PROBABILISTIC ANALYSIS OF GEOMETRIC ALGORITHMS

Mordecai J. Golin
(Thesis)

CS-TR-266-90

June 1990

PROBABILISTIC ANALYSIS OF GEOMETRIC ALGORITHMS

MORDECAI J. GOLIN

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE

June 1990

Dedicated with love and thanks to my parents
Fred Paul and Joyce Lee Golin

Probabilistic Analysis of Geometric Algorithms – Abstract

Mordecai J. Golin
Thesis Advisor – Professor Robert Sedgewick

This thesis is divided into four chapters. In the first chapter we describe the subtleties involved in probabilistically analyzing simple algorithms in computational geometry. We also work through a few easy examples of such analyses. The first example analyzes Quicksort, the second analyzes Quickhull, and the third analyzes interpoint distances between uniformly distributed points in hypercubes or hypertori.

In the second chapter we present a simple but effective preprocessing algorithm for calculating convex hulls. The algorithm is short and intuitive. It does a fast linear scan through the points, identifying many which are not on the convex hull and can therefore be eliminated. We perform an exact analysis of this algorithm showing that, given n points distributed uniformly in the unit square, only about $8\sqrt{n}$ of them remain after the preprocessing step: in higher dimensions only $c_d n^{1-1/d}$ will remain. We present the results of simulations comparing our mathematical analysis to reality. Finally, we end with a discussion of what distinguishes this algorithm from certain obvious variants.

In the third chapter we analyze closest pair algorithms. First, we analyze a sweep-line closest-pair algorithm, one similar in spirit to Hoey and Shamos' divide and conquer algorithm for solving the same problem. Our result is that, given n points uniformly distributed in the unit square and then sorted, there is a six line algorithm that finds the closest pair in $O(n)$ expected time. Moreover, this algorithm uses no complicated data structures. We then analyze a second algorithm, one that finds the closest pair using a modified version of Bentley and Papadimitriou's nearest neighbor projection algorithm. Our result, again, is that after the sorting stage, the linear scan stage of this new algorithm also finds the closest pair in $O(n)$ expected time.

The fourth chapter discusses the calculation of maxima. More specifically it analyzes a new heuristic, the Move-To-Front Algorithm, recently developed by Bentley, Clarkson and Levine. We prove mathematically that the Move-To-Front Algorithm is extremely efficient; on average it performs only one comparison per input point except on those in an asymptotically negligible subset.

Acknowledgments

It would be impossible to properly acknowledge everyone who helped me survive my time at Princeton. They made it a most rewarding period, both academically and otherwise. I would especially like to thank the faculty, graduate students, and staff of the Department of Computer Science.

Faculty members were always available and willing to answer questions. This "open door" policy greatly eased my transition from student to researcher and is something for which I continue to be grateful. I would like to signal out my readers for special praise. Bernard Chazelle introduced me to Computational Geometry and influenced my perceptions of the field. Jon Bentley came late, but more than almost anyone else, was an endless source of problems, critiques, and solutions: this thesis would have been much less without his input. Robert Sedgewick, my advisor, convinced me to become a Computer Scientist. His advice and example were the guiding forces behind this work.

Thanks to the graduate students for always being there. Without their friendship life would have been awfully boring. A special thanks goes to the officemates I have had through the years: Robert Abbott, Larry Aupperle, S.V. Krishnan, Sally Mckee, Richard Squier, and Jenny Zhao.

The staff (both clerical and technical) always made a point of avoiding the suffocating obfuscation that usually accompanies bureaucracy. I would like to thank them for their help and support. Special thanks and acknowledgment must go to Rebbeca Davies and Sharon Rodgers.

Finally, I would like to thank the friends who proofread the first version of this manuscript: Claire Kenyon, Burton Rosenberg, Norbert Schlenker, and Neal Young. Their corrections and comments were greatly appreciated.

Mordecai J. Golin
Princeton, New Jersey
May, 1990

Table of Contents

Chapter 1. Probabilistic Analysis of Geometric Algorithms

§1.1 Thesis Outline

This thesis describes and analyzes algorithms for solving some basic problems in computational geometry. The algorithms that we discuss are simple and robust. We show that, for specified input distributions, all of the algorithms discussed in this thesis have fast expected running times.

There are four chapters. In the first chapter we describe the subtleties of probabilistically analyzing algorithms in computational geometry. We also work through a few simple examples of average-case analysis. The first analyzes Quicksort, the second analyzes Quickhull, and the third analyzes interpoint distances between uniformly distributed points in hypercubes or hypertori.

In the second chapter we present a simple but effective preprocessing algorithm for convex hull calculations in all dimensions. The algorithm is intuitive and short (in two dimensions it requires only thirty lines of Pascal). It does a fast linear scan through the points, identifying many that are not on the convex hull and can therefore be eliminated. We perform an exact analysis of this algorithm showing that, given n points distributed uniformly in the unit square, only about $8\sqrt{n}$ of them remain after preprocessing. Extending this analysis, we show that in higher dimensions only $c_d n^{1-1/d}$ remain. We compare the predictions of the mathematical analysis to simulation results. Finally, we end with a discussion of what distinguishes this algorithm from certain obvious variants.

In the third chapter we present an analysis of the sweep-line closest-pair algorithm described in [**HNS**]. Our result is that, given n points uniformly distributed in the unit square and then sorted, a simple six-line algorithm finds the closest pair in $O(n)$ time. Moreover, this algorithm uses no complicated data structures.

The fourth chapter discusses the calculation of maxima. It analyzes a new heuristic, the Move To Front Algorithm, developed by Bentley, Clarkson and Levine [**BCL**]. We prove that the Move To Front Algorithm is extremely efficient; on average *it* performs only one comparison per input point except on those in an asymptotically negligible subset.

This thesis contains work both original and derivative. It is customary (and necessary) to distinguish the two. In Chapter One the analysis of Quicksort is

classic and well known [**Se1**]. The fact that the average running time of Quickhull is linear was also well known [**OL**]; the proof given here is new. The analysis of interpoint distances seems to be part of the folklore, known by many but never written down: the analysis of the nearest neighbor distance is implicit in prior probabilistic analyses of geometric algorithms, e.g. [**BWY**], while the analysis of closest pair distances doesn't seem to be recorded anywhere. Chapter Two is a rewritten, extended version of a paper presented at the Third ACM Symposium on Computational Geometry [**GS1**]. Chapter Three is an improved version of an unpublished technical report. The algorithm in Chapter Four is due to [**BCL**]; the analysis is original.

§1.2 Introduction

The work in this thesis grew out of concern about what constitutes a good "real-world" algorithm, one that an average programmer could and would implement if called upon to solve a problem. Two criteria immediately come to mind as being overwhelmingly important: simplicity and speed.

Simplicity means that the algorithm should be short, straightforward and austere. Short is self explanatory. Straightforward implies that it should take just a few minutes to describe and, hopefully be understandable directly from its implementation. Austere means that it requires neither special treatment for a plethora of unusual cases nor complicated data structures. If any of these conditions are not met then an average programmer, deterred by its complexity, will probably not attempt to implement the algorithm or, if he does, will have difficulties debugging it.

Speed is important for the obvious reason. If an algorithm is too slow then it is unusable. Here, as everywhere else in theoretical computer science, the underlying problem of complexity theory rears its head: how to measure the speed of an algorithm? We would like to measure worst-case complexity, the longest time the algorithm takes to run on any input of a given size. Unfortunately, it is unusual to find a *simple* algorithm that runs well on all inputs. Therefore we concentrate on expected running times, the average time the algorithm takes to run over all inputs of a given size. This introduces other difficulties, the most obvious of which is how to choose an input distribution that mirrors the *real-world problem* that the algorithm is supposed to address

There is an interesting connection between our two criteria. Simple, short

programs frequently have less overhead than more complicated algorithms do and thus often have faster expected running times [**Pi**]. We will exploit this fact often in the problems that we will address.

The search for fast, easily implementable algorithms is not a new goal. Much of the early history of computer science was devoted to finding just such algorithms for sorting and searching problems. For example, all of the early sorting techniques – Insertionsort, Mergesort, and Quicksort to name just a few – were exposed to intense scrutiny in attempts to optimize their running times. Many of the analytic techniques and approaches we rely upon today were developed during that search [**Kn1**] [**Kn3**].

What we do in this thesis is apply these criteria to some algorithms in computational geometry. Computational geometry, as its name implies, is the branch of computer science that develops and analyzes geometric algorithms: algorithms whose inputs and outputs are geometric objects such as point sets and line arrangements. For a specified set of points some typical problems are finding the closest pair it contains (Chapter Three), its convex hull (§1.2 and Chapter Two) or the set of its maxima (Chapter Four). Another typical problem involves *rectangular range searching*. For a given point set we want to preprocess the information so as to answer queries of the form: list all points in the set that are contained in the rectangle $\{(x,y)|\ x_{min} < x < x_{max},\ y_{min} < y < y_{max}\}$?

While the concept of an "algorithmic geometry" can be traced as far back as Euclid, computational geometry as we know it today evolved only during the mid 1970's. At that time researchers in diverse areas realized that many of the problems they were dealing with were naturally expressed in geometric terms. Computer graphics, for example, is such a natural source of problems that it is sometimes hard to know where it ends and computational geometry begins. Very Large Scale Integration (VLSI) chip design is another such area.

A third area of application is statistics. If we think of data points as existing in multidimensional space, each attribute reported corresponding to a dimension, then a statistical analysis frequently reduces to a geometric one. As an example it is known that the outlier problem – identifying those data points very different from the others in the set – is very similar to the convex layer problem in computational geometry that has been studied by Chazelle [**Ch**]. See also Shamos's discussion of Computational Statistics in [**Sha2**]. We will discuss the connection to statistics in more detail in Chapter Four.

For a comprehensive introduction to the field see Preparata and Shamos's book

[**PS**]. Shamos's 1978 Yale thesis [**Sha1**] was one of the first major works to address Computational Geometry as a distinct discipline. As such, besides being a good survey of the major problems, it also provides a historical perspective, a glimpse at a science in the making. Lee and Preparata [**LP**] provide a comprehensive bibliography up to 1984. Aggarwal and Wein's notes [**AW**] for the Spring 1988 Computational Geometry course at M.I.T. are an excellent introductory exposition. These notes also include a more up to date bibliography than [**LP**] and a collection of open problems.

To satisfy the purpose of this thesis, we must be able to design fast and simple algorithms in computational geometry. Designing simple algorithms is easy; guaranteeing that they run fast is often difficult. To find fast algorithms we must be able to perform probabilistic analyses on geometric algorithms. As we shall see later in this chapter such analyses can get very complicated very fast and be much more difficult than those of sorting and searching algorithms.

The question arises as to how the analysis of algorithms for sorting and searching differs from the analysis of those in computational geometry. A quick review of the literature points up a fundamental difference between how analyses in the two areas are approached. As an overbroad generalization we offer the following observations. In sorting or searching an attempt is usually made to analyze the algorithms directly. In computational geometry the attempt made is usually made to fit the algorithm to known geometric facts. To put it another way: in sorting or searching the analysis is tailored to follow the the twisted path of the algorithm's execution. In computational geometry, by contrast, the algorithm itself is frequently not analyzed directly. Instead we either exhume some old geometric fact or prove an appropriate new one and use this fact to derive the *general* behavior of the algorithm. Of course, we do not mean to imply that geometric algorithms are never analyzed directly (see for example [**BWY**] or [**Dw**]), just that they usually are not.

To make this discussion more concrete we contrast canonical examples of the two types of analyses. Quicksort is one of the older sorting algorithms [**Kn3**]. Working through its analysis (quickly reviewed in §**1.3**) we see that the analysis follows the execution of the algorithm step by step, aping its behavior on every possible input. Contrast this with the analysis of Giftwrapping, one of the older convex hull algorithms [**PS**]. In two dimensions it is known to run in $\Theta(nh)$ time on inputs of n points where h is the number of points on the convex hull. Let $\mathtt{E}(\cdot)$ be the expectation operator. The expected running time of the algorithm is then $n\mathtt{E}(h)$. A well known theorem of Renyi and Sulanke states:

Theorem [RS]: *Choose n points independently from a uniform distribution over a convex polygon. Let h be the number of points on their convex hull. Then* $\mathtt{E}(h) = O(\log n)$.

The Giftwrapping algorithm therefore runs in $O(n \log n)$ expected time when it is run on n points chosen independently from an appropriate distribution.

Without overly belaboring the point we would like to stress that in this analysis the algorithm itself wasn't analyzed. Rather, it was fitted to a previously known geometric result. As we said before this seems to be the standard technique used for analyzing algorithms in computational geometry.

There are at least three consequences of this method of analyzing algorithms. The first is that many algorithms simply do not have their expected running times examined. This is because it can be difficult to find appropriate geometric facts that let us analyze the algorithms. The second is that there are algorithms that are designed, not for ease of implementation, but so that they fit certain geometric facts that guarantee a good average case analysis. The third is that it is extremely difficult, if not impossible, to use such mathematical analysis to understand the subtleties of the algorithm's behavior. Understanding the subtleties is important because it lets us fine tune the algorithm to yield a variant that runs well in practice. It is known for example, that the ability to model Quicksort's behavior analytically led to a better understanding of variants of Quicksort (e.g.. median-of-three, finishing with an insertion-sort) that tend to run faster than the plain vanilla version. This idea, that a full analytic understanding of algorithms leads to the design of faster real world algorithms, runs through all of Knuth's work [**Kn1**] [**Kn2**] [**Kn3**]. Since the standard method of analyzing algorithms in computational geometry gives us only an understanding of gross behavior it is sometimes very difficult to use this information for fine tuning.

Obviously, theoretical computer scientists prefer to analyze algorithms directly. Why, then, do we see so few direct analyses in probabilistic computational geometry? The unsurprising answer seems to be that such analyses, even of simple algorithms, tend to be much more complicated than those of comparable algorithms for sorting and searching. A major reason for this complication is the extra degree of conditionality that geometric algorithms introduce. An algorithm, almost by definition, incorporates many *if...then* decisions which, in average case analyses, are dealt with by analyzing corresponding conditional probabilities. In geometric analyses these conditional terms are frequently hard to treat analytically. Another

reason that these analyses are more complicated is that a great many of the combinatorial tools developed over so many years by so many people to explain sorting and searching behavior do not seem to be applicable here. For sorting and searching algorithms the conditional terms described above can often be defined recursively and therefore analyzed using standard combinatorial techniques e.g. generating functions. For geometric algorithms these conditional terms are usually functions of geometric structures which often have no easy combinatorial description.

The remainder of this chapter is devoted to describing the difficulties inherent in attempting to perform average case analyses in computational geometry and what the tools are that we will use to overcome these difficulties. We will do this by examining three very different examples of analyses. The first example is of Quicksort. Its analysis illustrates how sorting and searching algorithms are approached. We show how what might be expected to be a geometric analysis is "combinatorialized" and therefore simplified. The second example is of Quickhull, an algorithm that finds convex hulls in two dimensions. Its analysis, although intrinsically geometric, is still easy. This is because we will reduce the analysis of the algorithm to the analysis of how the algorithm behaves on just one point, effectively removing worries about conditionality. The third and last example calculates the expected "nearest neighbor distances" between points uniformly distributed in a hypercube or a hypertorus. In this example we confront the problems of conditionality head on and introduce some standard techniques for dealing with them. The second, third, and fourth chapters present simple effective algorithms for finding, respectively, the convex hull, closest pairs, and maxima of point sets. The analyses of these algorithms will show that they are, in an expected sense, relatively fast. In order to perform these later analyses we need the tools and results that are developed in the rest of this chapter.

§1.3 Two Algorithms and their Analyses

§1.3.1 Quicksort

Figure 1.1 provides Pascal code and an example describing **Quicksort**, one of the older and more venerable sorting algorithms. Quicksort is a simple algorithm,

```
procedure quicksort(l, r : integer;)
  var  v, t, i, j : integer
  begin
  if r>l then
     begin
     v:=a[r]; i:= l-1; j:=r
     repeat
        repeat i:=i+1 until a[i]>=v;
        repeat j:=j-1 until a[j]<=v;
        swap(i,j);
     until j<=i;
     swap(i,j);
     quicksort(l, i-1);
     quicksort(i+1, r);
     end
  end;
```

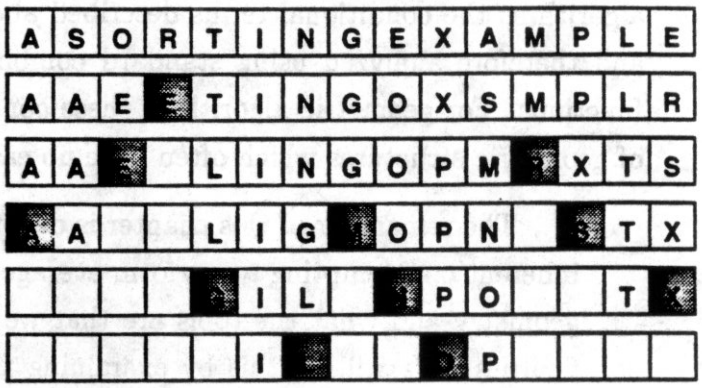


Figure 1.1. The left hand side of the figure provides Pascal code for Quicksort. *Swap(i,j)* is a procedure that swaps the values in $a[i]$ and $a[j]$. To ensure that i and j are always legal indices of $a[\,]$ we use sentinels: if the elements to be sorted are originally in $a[1], \ldots, a[n]$, then we set $a[0] = -\infty$, and $a[n+1] = \infty$, as sentinels. The right hand side of the figure is a worked example where the inputs are letters. Each level of the example corresponds to a recursive level of the program. The shaded letters on each level are the partitioning elements for their sets.

typically taught to most beginning computer scientists. Because it is so simple it has been analyzed in excruciating detail, making it ideal for our purpose of illustration.

A quick description of the algorithm will be helpful here[1]. It starts by getting n points (real numbers), $k_1, \ldots, k_n$, as input and storing them in an array $a[\,]$ where $a[i] = k_i$. It then designates the last point, k_n, as the *partitioning* element and partitions the remaining points based on it. This means that it rearranges the array (Figure 1.1) so that $a[d] = k_n$, $a[1], \ldots, a[d-1] \leq k_n$, and $a[d+1], \ldots, a[n] \geq k_n$. It finishes by recursively sorting the two point sets $a[1], \ldots, a[d-1]$, and $a[d+1], \ldots, a[n]$.

[1] Since our purpose is not to analyze Quicksort but to show how its probabilistic analysis differs from that of geometric algorithms what we provide is only a sketch. For a better description of the algorithm see [**Se1**] which is the source for the code and example given in Figure 1.1. For a complete description and analyses of efficient variants see [**Se2**].

In keeping with the purpose of this work we would like to perform a probabilistic analysis of Quicksort. For our model we will assume that the k_i-s are real numbers scaled so that they are all in the interval $[0, 1]$. An average case analysis of an algorithm like Quicksort would ask the following question:

> *Suppose that we are given n points (real numbers) independently identically uniformly distributed in the interval* $[0, 1]$. *What is the expected running time of Quicksort?*

This is a purely geometric quesion about an algorithm that manipulates points on a line segment. As such it seems to call for a (complicated) geometric analysis. Fortunately there is an easier approach. The crucial observation is that Quicksort's behavior is dependent only upon the relative ordering of the points and not upon their exact locations. Thus the point set (0.2, 0.1, 0.4, 0.3) is treated in exactly the same way as the point set (0.1, 0.05, 0.6, 0.3). All that matters is that they both have the same ordering as the permutation[2] **2 1 4 3**. If the n points are chosen independently identically distributed (I.I.D.) uniformly from the interval $[0, 1]$, then the probability of any two of them being equal is zero, i.e. with probability one, all points are unique. Therefore, there is some permutation that describes the relative ordering of the points. Furthermore the n points are indistinguishable so each of the $n!$ permutations occurs with equal probability. This transforms our original probabilistic question about a set of random points into one about random permutations:

> *Suppose that we are given a permutation of n items chosen at random uniformly from the n! possible permutations. What is the expected running time of Quicksort?*

The analysis of Quicksort is now standard and very well known. Let $\mathbf{k_1\ k_2\ \ldots k_n}$ be the given permutation of $[1 \cdots n]$ and let

C_n = Average number of comparisons made by Quicksort on n elements,

where the initial conditions are $C_0 = C_1 = 0$. We can now derive a recurrence relation for the C_n by examining the code in Figure 1.1. The first step, comparing each element to k_n, requires either n or $n + 1$ comparisons, depending on whether

[2] A permutation is a one-to-one mapping $f : [1 \cdots n] \rightarrow [1 \cdots n]$. It is frequently written in the form $\mathbf{f(1)\ f(2)\ \ldots f(n)}$. Thus **2 1 4 3** represents the permutation $f(1) = 2$, $f(2) = 1$, $f(3) = 4$, $f(4) = 3$.

the *repeat* loop terminates because $j = i$ or because $j < i$. For the rest of this section we will assume that the first step requires $n+1$ comparisons. (We will see later that replacing $n+1$ by n will only change low order terms in the analysis of C_n.) How much time does each of the recursive substeps take? The answer depends on the value of the partitioning element which, because the permutation was chosen uniformly from the set of all permuatations, takes on all values with equal probability, i.e.

$$\forall i \in [1 \cdots n], \quad \Pr(k_n = i) = 1/n.$$

If $k_n = i$ then the two recursive calls are on sets of size $i-1$ and $n-i$ respectively. Furthermore the permutation of the elements $[1 \cdots i]$ is random before the partitioning and therefore the permutations of the two new subsets are also random. This is because, in the partitioning step, the relative ordering of the elements is not used: the only facts used are whether the elements are greater or less than the partitioning element [**Kn3**]. The recursive call on the set of size $i-1$ will therefore perform, on average, C_{i-1} comparisons. Simlarly, the call on the set of size $n-i$ will perform C_{n-i}. Putting it all together

$$C_n = n + 1 + \frac{1}{n} \sum_{1 \le i \le n} (C_{i-1} + C_{n-i}) \tag{1.1}$$

which, by symmetry, is equivalent to

$$C_n = n + 1 + \frac{2}{n} \sum_{1 \le i \le n} C_{i-1}. \tag{1.2}$$

Multiplying both sides by n we find

$$nC_n = n(n+1) + 2 \sum_{1 \le i \le n} C_{i-1}. \tag{1.3}$$

Subtracting this equation (with $n-1$ substituted for n) from itself and simplifying gives

$$nC_n - (n+1)C_{n-1} = n(n+1) - (n-1)n + 2C_{n-1} \tag{1.4}$$

which gives

$$nC_n = (n+1)C_{n-1} + 2n.$$

We now divide both sides by $n(n+1)$ and telescope:

$$\begin{aligned}\frac{C_n}{(n+1)} &= \frac{C_{n-1}}{n} + \frac{2}{n+1} \\ &= \frac{C_{n-2}}{n-1} + \frac{2}{n+1} + \frac{2}{n} \cdots \\ &= \frac{C_2}{3} + \sum_{3 \le k \le n} \frac{2}{k+1} \\ &= 1 + 2\left(H_{n+1} - \frac{11}{6}\right).\end{aligned} \tag{1.5}$$

The harmonic sum $H_n = \sum_{1 \le i \le n} 1/i$ is approximated by $H_n \approx \ln n$ so

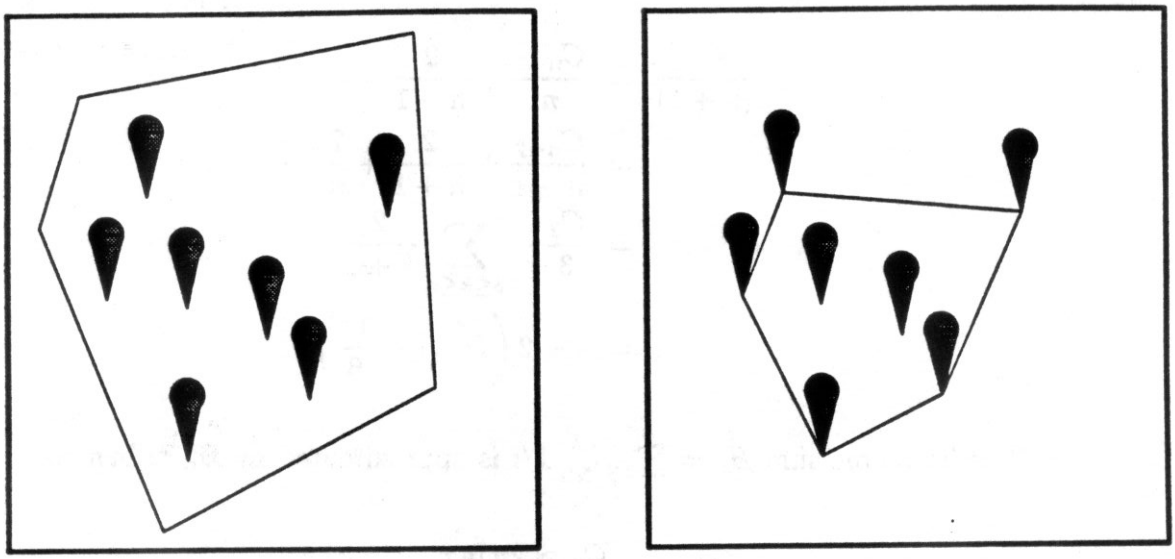
$$C_n \approx 2n \ln n.$$

Recall that we assumed that the first partitioning stage always makes $n+1$ comparisons. What would happen if we assumed that it always made n comparisons, if we replaced (1.1) by

$$C_n = n + \frac{1}{n} \sum_{1 \le i \le n} (C_{i-1} + C_{n-i})?$$

An analysis exactly like the one we performed for (1.1) again derives that $C_n \approx 2n \ln n$. Therefore our assumption did not bias the analysis and $C_n \approx 2n \ln n$.

This is not the end of Quicksort's analysis. There are myriad variants (median-of-three, ending with an insertion-sort, . . .) many of which were analyzed [**Se2**] so that their implementations can be fine-tuned to realize good running times on real computers. The point that we wish to make here is that what makes all of these analyses (and the implementations based on them) possible is the recasting of the geometric problem of points on a line into the combinatorial one of random permutations. If we can combinatorialize a problem we have a chance of bringing the entire arsenal of combinatorial analysis to bear on it. As we will see in the next section it is not always possible to combinatorialize a problem. Sometimes we have to perform the analysis in a purely geometric fashion.



(a) (b)

Figure 1.2. The diagram on the left shows a collection of nails on a board with a rubber band stretched so that it surrounds them. On the right the rubber band has been let go and snaps back to tightly surround the nails. The rubber band is the *convex hull* of the nails.

§1.3.2 Quickhull

Next we'll look at an algorithm whose analysis is purely geometric, and unlike that of Quicksort, requires knowledge of the physical locations of the points. The reason we present it here is to give an easy example of a geometric analysis that can't be combinatorialized. The algorithm is Quickhull [**OL**] (also known as Bykat's algorithm) and its purpose is the calculation of convex hulls in two dimensions. In order to understand the algorithm we'll need to provide a few simple properties of convex hulls.

First, some definitions. There is a simple physical system that provides geometric intuition into the structure of convex hulls. Pound nails into a flat board. Take a large rubber band and stretch it so that it surrounds all of the nails (Figure 1.2a). Next, let go of the rubber band and let it snap back tightly against the nails (Figure 1.2b). The rubber band is the convex hull of the nails.

Mathematically a set $P \subseteq \mathcal{R}^2$ is called *convex* if for every pair of points $p, q \in P$ the entire line segment

$$\overline{pq} = \{\lambda p + (1-\lambda)q \mid 0 \leq \lambda \leq 1\}$$

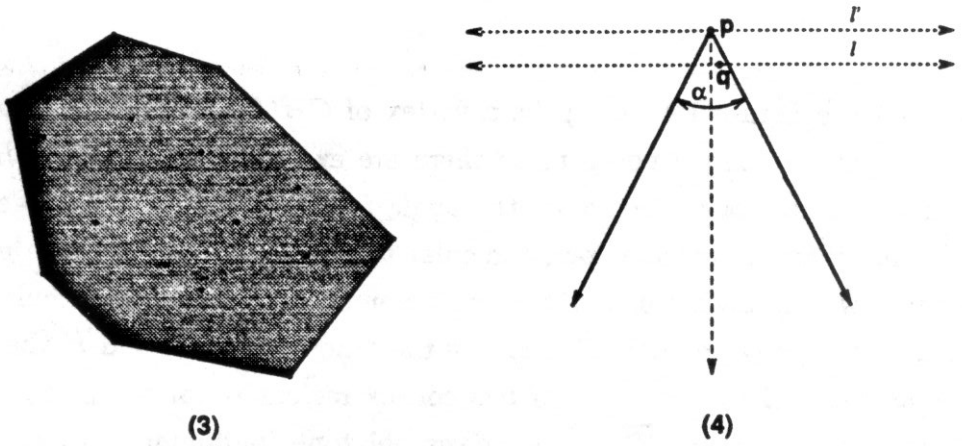


Figure 1.3. S is the collection of points, $CH(S)$ is the bold polygonal line, and $\overline{CH(S)}$ is the grey area.

Figure 1.4. Suppose $\mathbf{p} \notin S$ is a vertex of the covex hull of S. The bold lines are the two convex hull edges that have $\mathbf{p}$ as an endpoint. The dotted line is the bisector of the angle, α, that is formed by these two edges. l' is the line perpendicular to the bisector that goes through $\mathbf{p}$. Since $p \notin S$ we can slide l' a positive distance down the bisector until it hits some point $\mathbf{q} \in S$. Let l be the line through $\mathbf{q}$ that is parallel to l'.

---

is also in P. Given a finite set $S = \{p_1, \ldots, p_n\} \subseteq \Re^2$ we define the *convex hull* of S as

$$CH(S) \quad = \quad \text{the boundary of the smallest convex polygon containing } S. \quad (1.6)$$

There is some confusion in the literature over whether a convex hull is the *boundary* of the smallest convex polygon or whether it is the polygon itself. We follow the lead of [**PS**] and define it as in (1.6). Keeping our notation consistent we also denote

$$\overline{CH(S)} \quad = \quad \text{the smallest convex polygon containing } S.$$

Figure 1.3 illustrates these concepts. We will also need the following elementary facts about convex hulls:

Lemma 1.1: For any point set $P \subseteq \mathcal{R}^2$ and its convex hull $CH(S)$:
(i) The vertices of $CH(S)$ are points in S.
(ii) A point $p \in S$ is a vertex of $CH(S)$ if and only if there exists a line l through p such that all of the other points of S other than p lie completely on one side of l.

Proof: (i) See Figure 1.4. Let p be a vertex of $CH(S)$ and suppose that $p \notin S$. Convex hulls are simple polygons so there are exactly two edges of $CH(S)$ that have p as an endpoint. Let α be the angle formed at p by these edges. Let l' be the line through p that is perpendicular to the bisector of α. Since $p \notin S$ and S is finite we can always slide l' a small amount along the perpendicular bisector without hitting any points in S. Let l be the "slid" copy of l' and H the halfspace under it. Since the intersection of two convex regions is convex we see, from our construction, that $H \cap \overline{CH(S)}$ is a convex polytope containing S which is *smaller* than S. By contradiction p must be in S.
(ii) If p is a vertex of $CH(S)$ then the line l' through p that was defined in the proof of (i) has the given property.

Otherwise suppose that p is not a vertex. We will show that there is no line with the given property. There are two possible situations. The first is that p is on some edge $e = (r, s)$ of $CH(S)$. In this case any line through p either splits S into two nonempty sets (one containing r, the other s) or the line is tangent to e and itself contains r and s. The second situation is that p is interior to $\overline{CH(S)}$. But then any line through p intersects $CH(S)$ in two places and therefore partitions S into two proper subsets.

Q.E.D.

Corollary 1.2: If for a point $p \in S$, there exist three other points $q, r, s \in S$ such that p is inside the triangle $\triangle qrs$, then p is not a vertex of $CH(S)$.

Proof: A line through p splits the set $\{q, r, s\}$ into two nonempty sets and therefore also splits S.

We now know enough to understand Quickhull. First we describe the algorithm. Let $S = \{p_1, \ldots, p_n\}$. We want to construct $CH(S)$. Quickhull starts by finding the the leftmost and rightmost points of S and labeling them P_L and P_R. It partitions the points of S depending on whether they are above or below the line $\overline{P_L P_R}$ (points on the line can be disposed of since they can not be vertices of the convex hull).

$$Q_a = \{p \mid p \in S \text{ and above } \overline{P_L P_R}\}, \qquad Q_b = \{p \mid p \in S \text{ and below } \overline{P_L P_R}\}.$$

It then calls the routine $QH(P_L, P_R, Q_a)$ to find the upper hull of S, i.e. the convex hull of all the points on or above $\overline{P_L P_R}$. The lower hull can be found via a similar

routine (e.g. transform all of the points in S using $g((x,y)) = (x,-y)$. The lower hull of S is g of the upper hull of $g(S)$). After the upper and lower hulls have been found it patches them together to construct $CH(S)$. As output it prints the convex hull vertices in clockwise order. If three or more input points lie on the same convex hull edge the algorithm will return the endpoints of the edge. It might also return the points that are on the edge but not endpoints: whether it does or not depends on the order in which it examines the points.

$QH(r,s,Q)$:

Input: Two points, r and s, and a set of points Q such that, for every $p \in Q$, p is above the line $\overline{rs}$ and $r.x \leq p.x \leq q.x$.

Output: $CH(Q \cup \{r, s\})$.

Description:

Find the point p_f in Q that is the furthest distance from $\overline{rs}$. Divide the points in Q into three sets. Q_1 is the set of all points above $\overline{rp_f}$; Q_2, all points in the triangle $\triangle rp_fs$; Q_3 all points above $\overline{p_fs}$. We forget about the points in Q_2.

If Q_1 is empty let $C_1 = \emptyset$. Otherwise recursively call $QH(r, p_f, Q_1)$ to find C_1, the upper hull above $\overline{rp_f}$.

If Q_3 is empty let $C_2 = \emptyset$. Otherwise recursively call $QH(p_f, s, Q_3)$ to find C_2, the upper hull above $\overline{p_fs}$.

The convex hull of $Q \cup \{r,s\}$ is the concatenation $\{r\}\, C_1\, \{p_f\}\, C_2\, \{s\}$.

End.

Figure 1.5 illustrates a worked example of the algorithm. We must now show that the algorithm correctly finds the upper hull of S. For simplicity's sake we will assume that the points are in general position – no three of them are collinear. Then, from the comment immediately preceeding the algorithm's description, the algorithm should report all of the convex hull vertices and no other points.

First we have to prove that the algorithm always terminates. The only possible way for it not to terminate is to have some infinite nested sequence of calls to $QH(r,s,Q)$. But each time $QH()$ is called the size of Q decreases by at least one and therefore there can be at most $n-2$ nested $QH()$ calls.

Next we show that the points reported by the algorithm are all on the convex hull. A point p is only reported as a hull point if it is the furthest from the line $l' = \overline{rs}$. Therefore all of the points in S are to one side of the line l that goes through p and is parallel to l'. Lemma 1.1(ii) tells us that p is a vertex of the convex hull.

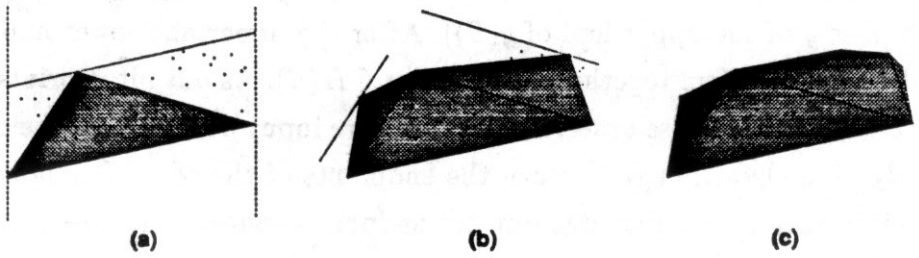


Figure 1.5. A worked example of Quickhull. In **(a)** we identify the left and rightmost points and the point furthest from the line connecting them. The points in the grey triangles are discarded. In **(b)** we perform the next level of the recursion, finding the furthest points in both the left and right sets. Again the points in the grey triangles are disposed of. Finally in **(c)** we perform the third level of the recursion (only on the right since there are no points remaining on the left). After this stage we have found the convex hull.

Finally we show that *all* of the vertices of $CH(S)$ are reported. From Lemma 1.1(i) we know that all of the vertices of $CH(S)$ are actually points in S. A point $p \in S$ is not reported as a hull point only if it is in Q_2 after some call $QH(r, s, Q)$. Therefore $p \in \triangle r p_f s$ and by Corollary 1.2 is not in $CH(S)$.

The algorithm is known as Quickhull because of its similarity to Quicksort. Both algorithms share the idea of finding a partioning element, using it to partition a set, and then recursively calling themselves on the two new, smaller subsets. Quickhull was independently discovered by Eddy [**Ed**] and Bykat [**Byk**]. Bykat at first thought that it had a worst case complexity of $O(n \log n)$ but it was later shown by [**Fo**] that there are inputs that force the algorithm to take $\Omega(n^2)$ time (Figure 1.6). On average though, the algorithm can be shown to take only $O(n)$ expected time when the points are drawn I.I.D. from a large number of input distributions. The following theorem was first presented in [**OL**] although the proof given here is different.

Theorem 1.3: If n points, $p_1, \ldots, p_n$, are chosen independently identically distributed from a uniform distribution in a bounded convex region $\mathcal{F}$ then Quickhull runs in expected $O(n)$ time on those points.

Proof: Before starting the proof we must point out that the probability that any three points are collinear is zero and that we can therefore assume that each p_i is either a vertex of the convex hull or interior to the hull.

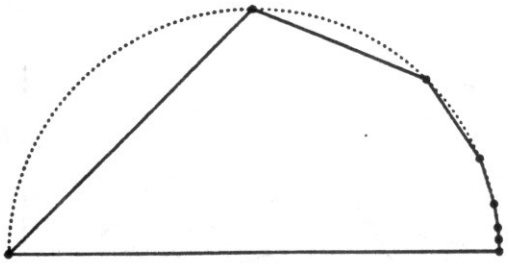


Figure 1.6. A degenerate case of Quickhull. Choose the n points so that they are on a circle: $p_i = (\cos\frac{\pi}{2^i}, \sin\frac{\pi}{2^i})$ for $0 \leq i < n-1$ and $p_{n-1} = (1,0)$. The solid polygon is the convex hull. Quickhull requires $\Omega(n^2)$ time for this point set; each call of $QH()$ will identify one point on the hull and then make only *one* recursive call on all of the remaining points.

The first part of the algorithm, identifying P_L and P_R and partioning the remaining points depending on whether they are above or below $\overline{P_L P_R}$, can be accomplished in two straight passes through the data and therefore only takes linear deterministic time. The second part of the algorithm, stitching together the upper and lower hulls, also can be done in linear time. Recall the definitions

$$Q_a = \{p \mid p \in S \text{ and above } \overline{P_L P_R}\}, \qquad Q_b = \{p \mid p \in S \text{ and below } \overline{P_L P_R}\}.$$

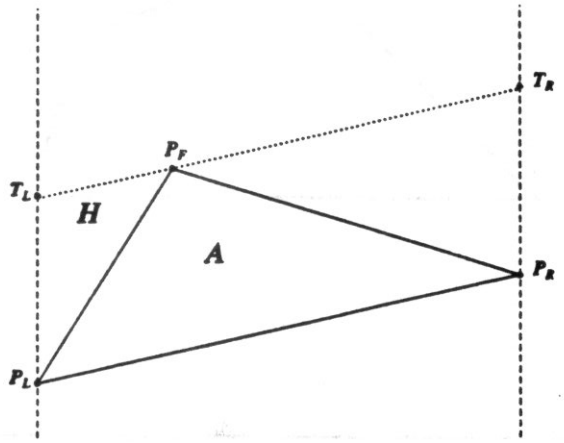
We must show that the expected amount of time taken by the calls $QH(P_L, P_R, Q_a)$ and $QH(P_L, P_R, Q_b)$ together is $O(n)$. Suppose, deterministically, that $|Q_a| = n_1$ and $|Q_b| = n_2$ where $n_1 + n_2 < n$. We will show that the expected amount of time taken by $QH(P_L, P_R, Q_a)$ is $O(n_1)$. A similar argument would show that the expected amount of time taken by $QH(P_L, P_R, Q_b)$ is $O(n_2)$. Combining the two proves the theorem.

Each recursive call, $QH(r, s, Q)$, requires $2|Q|$ time: $|Q|$ for the linear scan through the points that identifies p_f and another $|Q|$ for the second linear scan that partitions Q into Q_1, Q_2, Q_3. Using this fact we can calculate the total amount of work done by all the calls to $QH()$ by amortizing it over the points rather than over the actual procedure calls. We say that a point p *participates in a call on level* i if there is a series of directly nested calls

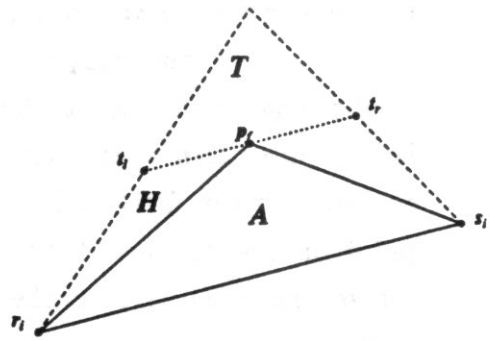
$$QH(r_1, s_1, Q^{(1)}),\ QH(r_2, s_2, Q^{(2)}), \cdots, QH(r_i, s_i, Q^{(i)})$$

where $p \in Q^{(i)} \subset \cdots \subset Q^{(2)} \subset Q^{(1)} = S'$. If we set

$$W(p) = \max_i \{p \textit{ participates in a call on level } i\}$$



(7)



(8)

Figure 1.7. The initial call of $QH(P_L, P_R, S')$. It finds the point p_f which is the furthest above $\overline{P_L P_R}$. It then disposes of all the points in the triangle $A = \triangle P_L p_f P_R$.

Figure 1.8. The general function call. It finds the point p_f which is the furthest above $\overline{r_i s_i}$. It then disposes of all the points in the triangle $A = \triangle r_i p_f s_i$.

---

then, from our definition of level and the way $QH()$ partions its input sets, p participates in exactly one call on every level less than $W(p)$ and none at any level above it. Thus the total amount of work done by all of the calls to $QH()$ will be $2\sum_i W(p_i)$ since, at each level a point participates, only two units of work are performed on that point. The first unit occurs during the scan that determines the furthest point from the current base line. The second occurs during the scan that throws away the points in the triangle. The total expected amount of work done will be $\mathrm{E}(2\sum_i W(p_i))$ which by the linearity of expectation and symmetry among the points will be equal to $2n_1\mathrm{E}(W(p_1))$. We will now show that $\mathrm{E}(W(p_1)) \leq 2$ and the proof of the theorem will follow.

We start by examining the first call, $QH(P_L, P_R, S')$. Let H be the parallelogram $P_L T_L T_R P_R$ and A the triangle $\triangle P_L p_f P_R$ as seen in Figure 1.7. Points in A are disposed of on the first level so $\Pr(W(p_1) > 1)$ is upper bounded[3] by $\Pr(p_1 \notin A)$. From the definition of P_L, P_R and p_f we know that $Q \subseteq H$ and from the convexity

---

[3] The probability is upper bounded by $\Pr(p_1 \notin A)$ but might not be equal to it because p_1 may be one of P_L, P_R, or p_f, the defining vertices of A. If this is the case then $\Pr(W(p_1) > 1) = 0$

of F, the support of the distribution, we know that $A \subseteq \mathcal{F}$. Therefore

$$\begin{aligned}\Pr(p_1 \in A) &= \frac{\text{Area}(A \cap \mathcal{F})}{\text{Area}(H \cap \mathcal{F})} \\ &= \frac{\text{Area}(A)}{\text{Area}(H \cap \mathcal{F})} \\ &\geq \frac{\text{Area}(A)}{\text{Area}(H)} = \frac{1}{2}\end{aligned}$$

and $\Pr(W(p_1) > 1) \leq 1/2$.

Next suppose that p_1 has participated in every level up to and including some level $i > 1$. What is the probability that it will participate in the $(i+1)$-st level? When we made the i-th level call $QH(r_i, s_i, Q^{(i)})$ with $p_1 \in Q^{(i)}$ there was some triangle T that contained all of the points in $Q^{(i)}$ and two of whose vertices were r_i and s_i. Let H be the quadrilateral $r_i t_l t_r s_i'$ and A the triangle $\triangle r_i p_f s_i$ as seen in Figure 1.8. Since $H \subseteq T$ we have $\text{Area}(A)/\text{Area}(H) \geq 1/2$ and the same reasoning we used for the first level gives $\Pr(p_1 \in A) \geq 1/2$. We have just shown that

$$\Pr(W(p_1) \geq i+1 \mid W(p_1) \geq i) \quad \leq \quad 1/2, \qquad\qquad i > 1$$

which put together with $\Pr(W(p_1) > 1) \leq 1/2$ yields $\Pr(W(p_i) \geq i) \leq 2^{-(i-1)}$. Using the standard formula for expectation we calculate

$$\text{E}(W(p_1)) \;=\; \sum_{i \geq 1} \Pr(W(p_i) \geq i) \;\leq\; \sum_{i \geq 1} 2^{-(i-1)} \;=\; 2.$$

Q.E.D.

What differentiates this analysis from that of Quicksort's is that for Quicksort we were able to combinatorialize the problem. Instead of focusing on point locations we were able to look at the combinatorics of permutations. In Quickhull's analysis, by contrast, we were unable to perform a similar combinatorialization and had to work from a totally geometric perspective. Our analysis was still easy though because we were able to reduce it to the analysis of how Quickhull acts on a single point. This enabled us to avoid the problems introduced by conditionality and interpoint dependencies. As we will see in the next section, we will not always be so lucky.

§1.4 Interpoint Distances

§1.4.1 Definitions and Mathematical Preliminaries

Our final foray into introductory geometric analysis will illustrate the inherent complexities of the subject. We examine the behavior of points taken from uniform distributions in the d-hypercube and the d-hypertorus. The d-hypercube $[0,1]^d$ is the region

$$\left\{\overline{\mathbf{x}} = (x_1, \ldots, x_d) \quad | \quad \forall i,\ 0 \leq x_i \leq 1\right\}$$

where the distance function between two points is the standard L_2 metric of $\mathcal{R}^d$

$$d(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \sqrt{\sum_{1 \leq i \leq d} (x_i - y_i)^2}.$$

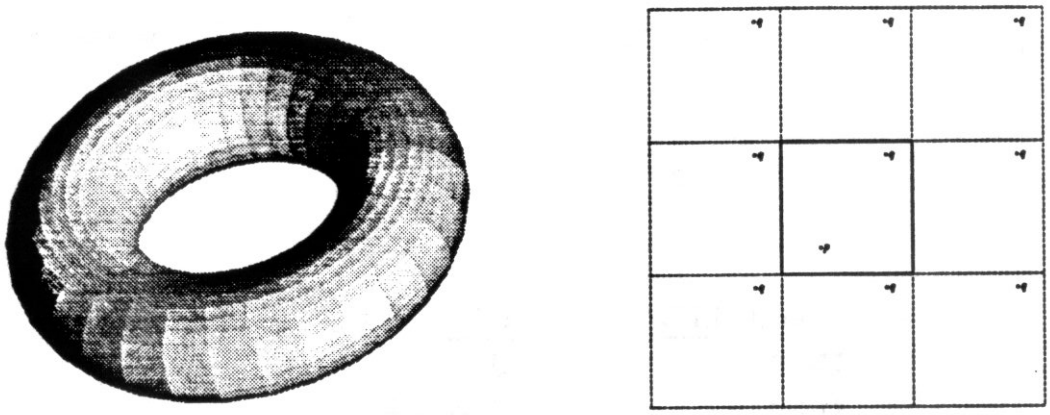
The d-hypertorus can be thought of as the d-hypercube with "wraparound": We fold the hypercube so that each of the bounding hyperplanes $x_i = 0$ touches the parallel hyperplane $x_i = 1$. The one dimensional torus is the circle with unit circumference (S^1). The two dimensional torus (Figure 1.9) can be constructed by taking a square and first taping its two horizontal edges together to form a cylinder and then connecting the cylinder's ends to form a doughnut. In higher dimensions the construction is similar but harder to visualize. The distance function for the one dimensional torus is

$$d_1(x, y) = \min\Big(|x - y|,\ 1 - |x - y|\Big)$$

while the distance function for the general torus is

$$d(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \sqrt{\sum_{1 \leq i \leq d} d_1(x_i, y_i)^2}.$$

To keep our notation consistent we will use the symbol $< 0,1 >^d$ for the d-hypertorus. Furthermore we will denote the uniform point distribution over $[0,1]^d$ and $< 0,1 >^d$ as $\mathbf{U}[0,1]^d$ and $\mathbf{U}< 0,1 >^d$ respectively. Finally, we will say that n points are I.I.D. if they are independently identically distributed over some distribution. Therefore when we talk about " *n points I.I.D.* $\mathbf{U}[0,1]^d$ " the intent is that the n points are drawn independently from the uniform distribution over the d-hypercube.



(a) (b)

Figure 1.9. Two representations of the two-dimensional torus. Figure **(a)** is a physical construction. Figure **(b)** provides some intuition as to how "wraparound" works in the plane. The heavy square in the middle is the actual torus. To find the distance between two points p and q in the torus construct copies of q in the surrounding squares. The distance in the torus between p and q is the minimum distance between p and the 9 different copies of q.

Let $\mathbf{p}_1, \ldots, \mathbf{p}_n$ be n points chosen I.I.D. from one of the distributions defined above. In this section we will analyse two random variables which are functions of such random point sets. The first is the minimum distance between $\mathbf{p}_1$ and all of the other points. We denote this distance by Y :

$$Y(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \min_{1<i\leq n} d(\mathbf{p}_1, \mathbf{p}_i).$$

The second random variable is the minimum distance between all pairs of points, which as known as the distance beween the *closest pair*. We denote this random variable by Z :

$$Z(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \min_{1\leq i<j\leq n} d(\mathbf{p}_i, \mathbf{p}_j).$$

Tables 1.1 and 1.2 present the results that we will prove later in this section.

We have two reasons for examining these random variables. The first, in keeping with the spirit of this chapter, is to illustrate the standard difficulties inherent in many probabilistic geometric computations. Our analysis of Quickhull, even though we were unable to combinatorialize its behavior, was still relatively trivial. This was because we were able to simplify the analysis by reducing it to the study of

n points I.I.D.	$\mathrm{E}(Y)$
$\mathbf{U}<0,1>$	$\frac{1}{2n}$
$\mathbf{U}[0,1]$	$\frac{1}{2n}+\frac{1}{n(n+1)}$
$\mathbf{U}<0,1>^d$	$\sigma_d n^{-1/d}\left[1+O\left(\frac{\ln n}{n}\right)\right]$
$\mathbf{U}[0,1]^d$	$\sigma_d n^{-1/d}\left[1+O\left(\left(\frac{1}{n}\right)^{1/d}\right)\right]$

Table 1.1: This table presents the results derived in §1.4.2. Y is the minimum distance between $\mathbf{p}_1$ and $\mathbf{p}_2,\ldots,\mathbf{p}_n$ where the $\mathbf{p}_i$ are chosen I.I.D. from some distribution. The first column is the appropriate distribution. The second column is $\mathrm{E}(Y)$ under this distribution. The value $\sigma_d = \frac{\Gamma(1/d)}{d\,\omega_d^{1/d}}$ where $\omega_d = 2\pi^{d/2}/d\Gamma(d/2)$ is the volume of the unit sphere $B(\mathbf{0},1)$.

n points I.I.D.	lower bounds on $\mathrm{E}(Z)$	upper bounds on $\mathrm{E}(Z)$
$\mathbf{U}<0,1>^d$	$2^{1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$	$2^{1+1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$
$\mathbf{U}[0,1]^d$	$2^{1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$	$2^{2+1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$

Table 1.2: This table presents the results derived in §1.4.3. Z is the minimum distance between the closest pair among $\mathbf{p}_1,\ldots,\mathbf{p}_n$ where the $\mathbf{p}_i$ are chosen I.I.D. from some distribution. The first column is the appropriate distribution. The second column contains lower bounds on $\mathrm{E}(Z)$; the third, upper bounds.

the behavior of one arbitrary point. By doing this we managed to sidestep dealing with how interpoint dependencies influence the running of the algorithm (such as *If point A is here then we don't have to check if point B is there except if point C is below it* ...). In this section, especially in §1.4.3, we won't be nearly as lucky. At least half of our calculations will be devoted to understanding how interpoint dependencies affect certain random variables, when they can be safely ignored, and how addressing them can introduce uncertainties into our equations. These types of calculations are present throughout this entire thesis – most importantly in Chapter Three, where we must understand Z to understand the behavior of an algorithm – so we begin to study them in these relatively simple situations. The second reason for examining them is to introduce the analytic techniques that we will use repeatedly throughout this paper. These include asymptotic expansions, Gamma and Beta function techniques, and nested area arguments.

The next few paragraphs contain mathematical definitions and concepts that we will need for our analyses. Most of them (except for Lemma 1.3) are fairly standard and are presented for the sake of completeness. This section can be safely skipped by those who are already familiar with the basics of real analysis.

<u>Mathematical Preliminaries:</u>

`Definition:` The *Gamma Function* of x is the definite integral

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt. \tag{1.7}$$

Stirling's Formula states that as $x \to \infty$

$$\Gamma(x) = \left(\frac{x}{e}\right)^x \sqrt{\frac{2\pi}{x}} \left[1 + O\left(\frac{1}{x}\right)\right]. \tag{1.8}$$

`Definition:` The *Incomplete Gamma Function* of u and x is

$$\Gamma(x,u) = \int_u^\infty t^{x-1}e^{-t}dt. \tag{1.9}$$

For fixed x, we have, as $u \to \infty$, the crude asymptotic bound

$$\Gamma(x,u) \leq \int_u^\infty e^{-2t/3}du = O(e^{-u/2}). \tag{1.10}$$

`Definition:` The *Beta Function* of x and y is the definite integral

$$\beta(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt. \tag{1.11}$$

It is known [**WW**] that

$$\beta(x,y) = \frac{\Gamma(x)\,\Gamma(y)}{\Gamma(x+y)}. \tag{1.12}$$

We will use the following very often:

Lemma 1.3: Let c be a real constant. As $x \to \infty$

$$\beta(c,x) = \Gamma(c)\,n^{-c}\left[1+O\left(\frac{1}{x}\right)\right]. \tag{1.13}$$

Proof: We know that

$$\beta(x,y) = \frac{\Gamma(c)\,\Gamma(y)}{\Gamma(x+c)}.$$

From Stirling's formula [**WW**] we find

$$\begin{aligned}\beta(c,x) &= \frac{\Gamma(c)\left(\frac{x}{e}\right)^x\sqrt{\frac{2\pi}{x}}\,\left[1+O\left(\frac{1}{x}\right)\right]}{\left(\frac{x+c}{e}\right)^{x+c}\sqrt{\frac{2\pi}{x+c}}\left[1+O\left(\frac{1}{x+c}\right)\right]} \\ &= \frac{\Gamma(c)\,e^c}{(x+c)^c}\left(\frac{x}{x+c}\right)^x\sqrt{\frac{x+c}{x}}\left[1+O\left(\frac{1}{x}\right)\right].\end{aligned}$$

As $x \to \infty$ standard asymptotic techniques show that

$$\begin{aligned}\sqrt{\frac{x+c}{x}} &= 1+O\left(\frac{1}{x}\right) \\ \frac{1}{(x+c)^c} &= x^{-c}\left[1+O\left(\frac{1}{x}\right)\right] \\ \left(\frac{x}{x+c}\right)^x &= e^{-c}\left[1+O\left(\frac{1}{x}\right)\right].\end{aligned}$$

Inserting these identities into the previous equation yields the desired result.

Q.E.D.

We will also need to be able to identify the open region surrounding a point.

`Definition:` The *neighborhood (ball)* of radius α around a point $\mathbf{p}$ is

$$B(\mathbf{p},\alpha) = \{\mathbf{y} \mid d(\mathbf{p},\mathbf{y}) < \alpha\}.$$

The volume of $B(\mathbf{p},\alpha)$ in $\mathcal{R}^d$ is $\nu_d = \omega_d\alpha^d$ where $\omega_d = 2\pi^{d/2}/d\,\Gamma(d/2)$ is the volume of the unit sphere $B(\mathbf{0},1)$.

Finally we will need the two asymptotic relations

$$\begin{aligned}e^x &= 1+O(x), \qquad x \to 0 \\ \ln(1-x) &= -x+O(x^2), \qquad x \to 0\end{aligned} \tag{1.14}$$

and the two summation formulae

$$\begin{aligned} \sum_{1 \le i < n} i &= \frac{1}{2}n^2 + O(n) \\ \sum_{1 \le i < n} i^2 &= \frac{1}{3}n^3 + O(n^2). \end{aligned} \tag{1.15}$$

§1.4.2 Nearest Neighbor to a Point

In this section we are given n points $\mathbf{p_1}, \ldots, \mathbf{p_n}$ drawn I.I.D. from one of our two distributions and pick some distinguished point at random. Without loss of generality we assume that we picked $\mathbf{p_1}$. Let Y be the distance from $\mathbf{p_1}$ to its nearest neighbor, i.e.

$$Y = \min_{1 < i \le n} d(\mathbf{p_1}, \mathbf{p_i}).$$

What is $\mathrm{E}(Y)$ *expressed as a function of* n? The answer, for n points I.I.D. both $\mathbf{U}< 0,1 >^d$ and $\mathbf{U}[0,1]^d$, is asymptotically $cn^{-1/d}$ where c is a constant dependent on d. Table 1.1 presents the results that will be derived in this section.

(a) First we will examine the simplest case: n points I.I.D. $\mathbf{U}< 0,1 >^1$. As was previously mentioned, the unit-length one-dimensional torus is a circle with circumference one. On this circle a pair of points is never more than distance $1/2$ from each other so $Y \le 1/2$. For $\alpha \le 1/2$ the event $Y \ge \alpha$ occurs if and only if none of the points $\mathbf{p_2}, \cdots, \mathbf{p_n}$ are in the neighborhood $B(\mathbf{p_1}, \alpha)$. Since the points are *independently* and *uniformly* distributed

$$\begin{aligned} \Pr(Y \ge \alpha) &= \Pr(\mathbf{p_2}, \cdots, \mathbf{p_n} \notin B(\mathbf{p_1}, \alpha)) \\ &= (\Pr(p_2 \notin B(\mathbf{p_1}, \alpha)))^{n-1} \\ &= (\Pr(1 - \text{Area}(B(\mathbf{p_1}, \alpha))))^{n-1} \\ &= (1 - 2\alpha)^{n-1}. \end{aligned} \tag{1.16}$$

The second equality follows from the independence; the third from the uniformity. We now calculate the expectation to be exactly

$$\begin{aligned} \mathrm{E}(Y) &= \int_0^{1/2} \Pr(Y \ge \alpha) d\alpha \\ &= \int_0^{1/2} (1 - 2\alpha)^{n-1} \, d\alpha = \frac{1}{2n}. \end{aligned} \tag{1.17}$$

(a') Now we look at the next simplest case, where the points are I.I.D. $\mathbf{U}[0,1]^1$. A priori we would expect $\mathrm{E}(Y)$ to be greater here than it was in the previous case. When $\mathbf{p_1}$ is near the middle of the interval $[0,1]$ the situation looks locally like the torus and we expect Y to behave the same way as it did for the torus. But, when $\mathbf{p_1}$ is the leftmost (rightmost) point in the interval its closest point has to be to its right (left) and Y is no longer the minimum of two numbers and, therefore, we expect Y to be bigger than it was in the case of the torus.

The analysis here differs from that of the previous case in the derivation of $\Pr(Y \geq \alpha)$. For the torus we implicitly used the fact that the points on the torus are indistinguishable from each other so

$$\Pr(Y \geq \alpha \,|\, \mathbf{p_1} = x)$$

was independent of x. For the unit interval this isn't true because $B(\mathbf{p_1}, \alpha)$ might not be totally contained in $[0,1]^1$ and the probability

$$\Pr(Y \geq \alpha \,|\, \mathbf{p_1} = x) \;=\; (1 - \mathrm{Area}\,(B(\mathbf{p_1}, \alpha) \cap [0,1]))^{n-1}$$

is a function of both α and x. For $x \leq 1/2$ we can explicitly calculate (Figure 1.10)

$$\mathrm{Area}\,(B(\mathbf{p_1}, \alpha) \cap [0,1]) = \begin{cases} 2\alpha & \text{if } 0 \leq \alpha \leq x \\ x + \alpha & \text{if } x \leq \alpha \leq 1 - x \\ 0 & \text{otherwise.} \end{cases} \tag{1.18}$$

For $x \geq 1/2$ the calculations are similar (replace x by $1 - x$ on the right hand side of brace). An expectation can be written as the weighted integral of conditional expectations so

$$\begin{aligned} \mathrm{E}(Y) &= \int_0^1 \mathrm{E}(Y \,|\, \mathbf{p_1} = x) dx \\ &= \int_0^1 \left[\int_0^1 \Pr(Y \geq \alpha \,|\, \mathbf{p_1} = x) d\alpha \right] dx. \end{aligned}$$

Substituting the values from (1.18) and using the symmetry between $x < 1/2$ and $x > 1/2$

$$\mathrm{E}(Y) = 2 \int_0^{1/2} \left[\int_0^x (1 - 2\alpha)^{n-1} d\alpha \;+\; \int_x^{1-x} (1 - x - \alpha)^{n-1} d\alpha \right] dx. \tag{1.19}$$

The two internal integrals can be evaluated

$$\begin{aligned} \int_0^x (1 - 2\alpha)^{n-1} d\alpha &= \frac{1}{2n}(1 - (1 - 2x)^n) \\ \int_x^{1-x} (1 - x - \alpha)^{n-1} d\alpha &= \frac{1}{n}(1 - 2x)^n. \end{aligned}$$

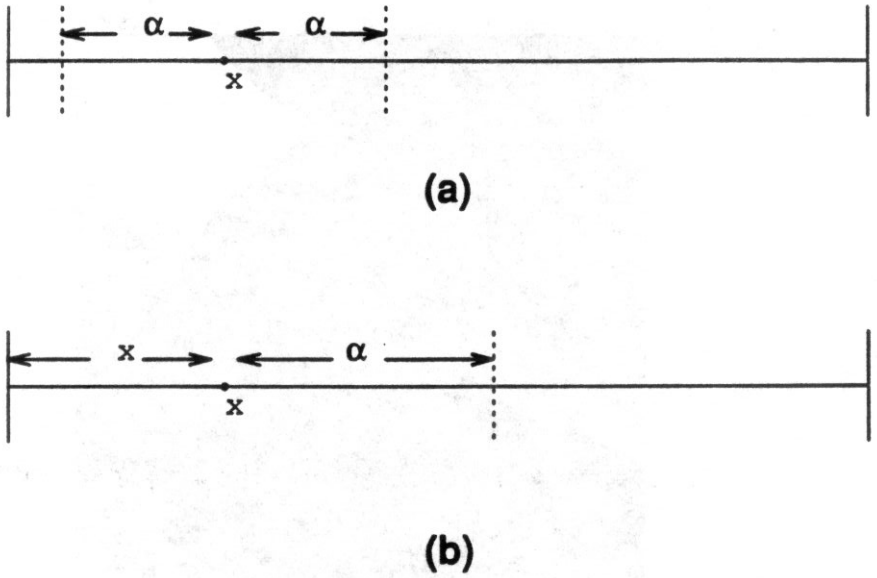


Figure 1.10. In (a) we see that if $\alpha \leq x$ then Area $(B(\mathbf{p_1}, \alpha) \cap [0,1]) = 2\alpha$. In (b) we see that if $x \leq \alpha \leq 1-x$ then Area $(B(\mathbf{p_1}, \alpha) \cap [0,1]) = x+\alpha$.

Substituting back into the original equation

$$\mathrm{E}(Y) = 2\int_0^{1/2} \frac{1}{2n}\left[1 + (1-2x)^n\right] dx = \frac{1}{2n} + \frac{1}{n(n+1)}. \tag{1.20}$$

We see that our original guess was correct: $\mathrm{E}(Y)$ is greater for the unit interval than it is for the unit torus. It's not that much greater though; the correction term is asymptotically negligible.

(*b*) Next we'll deal with n points I.I.D. $\mathbf{U}< 0,1 >^d$ with $d \geq 2$. We start the same way as before, by calculating

$$\Pr(Y \geq \alpha) = (1 - \text{Area}\,(B(\mathbf{p_1}, \alpha)))^{n-1}.$$

Observation shows that $0 \leq Y \leq \sqrt{d}/2$. If $\alpha \leq 1/2$ then Area $(B(\mathbf{p_1}, \alpha)) = \omega_d \alpha^d$. If $\alpha > 1/2$ then there is no longer a simple formula for the area of the neighborhood. This is because the boundary of the neighborhood overlaps itself (Figure 1.11). Fortunately $\alpha \geq 1/2$ is the exponentially low probability event:

$$\Pr(Y \geq 1/2) = \left(1 - \frac{\omega_d}{2^d}\right)^{n-1}. \tag{1.21}$$

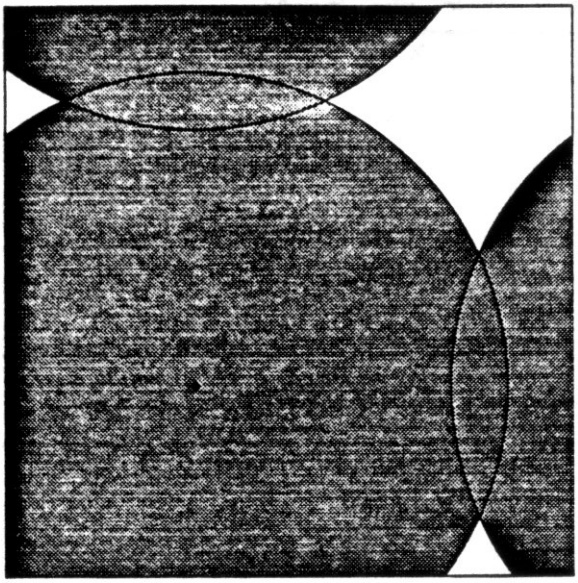


Figure 1.11. The diagram is of the two dimensional torus. The shaded area is the neighborhood $B(\mathbf{p}, 0.55)$ where $\mathbf{p} = (\frac{1}{3}, \frac{1}{3})$.

Therefore

$$\begin{aligned} \mathrm{E}(Y) &= \int_0^{\sqrt{d}/2} \Pr(Y \geq \alpha)\, d\alpha \\ &= \int_0^{1/2} \Pr(Y \geq \alpha)\, d\alpha + O\left[(1 - \omega_d/2^d)\right]. \end{aligned}$$

Furthermore we can add $\int_{1/2}^{\omega_d^{-1/d}} \left(1 - \omega_d \alpha^d\right)^{n-1} d\alpha = O\left[(1 - \omega_d/2^d)\right]$ to the previous expression so

$$\mathrm{E}(Y) = \int_0^{\omega_d^{-1/d}} \left(1 - \omega_d \alpha^d\right)^{n-1} d\alpha + O\left[(1 - \omega_d/2^d)\right]. \tag{1.22}$$

The integral can be evaluated by using the change of variable $u = \omega_d \alpha^d$ to get

$$\int_0^{\omega_d^{-1/d}} \left(1 - \omega_d \alpha^d\right)^{n-1} d\alpha = \frac{\beta(1/d, n)}{d\, \omega_d^{1/d}}. \tag{1.23}$$

Set

$$\sigma_d = \frac{\Gamma(1/d)}{d\, \omega_d^{1/d}}. \tag{1.24}$$

Substituting (1.23) into (1.22) and using Lemma 1.3 we have just shown

$$\mathrm{E}(Y) = \sigma_d n^{-1/d} \left[1 + O\left(\frac{1}{n}\right)\right]. \tag{1.25}$$

To review what we have just done: We saw that, for $\alpha \leq 1/2$, $\Pr(Y \geq \alpha) = \left(1 - \omega_d \alpha^d\right)^{n-1}$. We then used this to show that

$$\begin{aligned}
\mathrm{E}(Y) &= \int_0^{\sqrt{d}/2} \Pr(Y \geq \alpha) d\alpha \\
&\sim \int_0^{1/2} \Pr(Y \geq \alpha) d\alpha \\
&= \int_0^{1/2} \left(1 - \omega_d \alpha^d\right)^{n-1} d\alpha \\
&\sim \int_0^{\omega_d^{-1/d}} \left(1 - \omega_d \alpha^d\right)^{n-1} d\alpha \\
&= \frac{\beta(1/d, n)}{d\, \omega_d^{1/d}}.
\end{aligned}$$

This technique, of adding and subtracting asymptotically negligible quantities to massage an integral into a more tractable form, is very standard and will be used many times.

(b') If the n points are I.I.D. $\mathbf{U}[0,1]^d$ we no longer have to worry about the border of $B(\mathbf{p_1}, \alpha)$ overlapping itself. We *do* have to concern ourselves with the fact, similar to that observed in (a'), that $B(\mathbf{p_1}, \alpha)$ might not be totally contained in $[0,1]^d$. As a first step it is not difficult to see that, for $\alpha \leq 1/2$, at least $1/2^d$ of the volume of $B(\mathbf{p_1}, \alpha)$ (in two dimensions a quadrant, in three an octant, etc.) is contained in $[0,1]^d$. Therefore for $\alpha \leq 1/2$

$$\omega_d \alpha^d / 2^d \leq \text{Area}\left[B(\mathbf{p_1}, \alpha) \cap [0,1]^d\right] \leq \omega_d \alpha^d$$

and

$$\left(1 - \omega_d \alpha^d\right)^{n-1} \leq \Pr(Y \geq \alpha) \leq \left(1 - \omega_d \alpha^d / 2^d\right)^{n-1}.$$

Calculations exactly like those that took us from (1.21) to (1.25) yield

$$\sigma_d n^{-1/d} \left[1 + O\left(\frac{1}{n}\right)\right] \leq \mathrm{E}(Y) \leq 2\sigma_d n^{-1/d} \left[1 + O\left(\frac{1}{n}\right)\right] \tag{1.26}$$

With just a little more work we'll be able to show that the lower bound is tight although we will lose some asymptotic precision. The difficulty in explicitly calculating $\mathrm{E}(Y)$ is that if $\mathbf{p_1}$ is close enough to the border of $[0,1]^d$, then $B(\mathbf{p_1}, \alpha)$ isn't totally contained in $[0,1]^d$. We will show that $\mathbf{p_1}$ has a very low probability of being close enough to the border for this to happen.

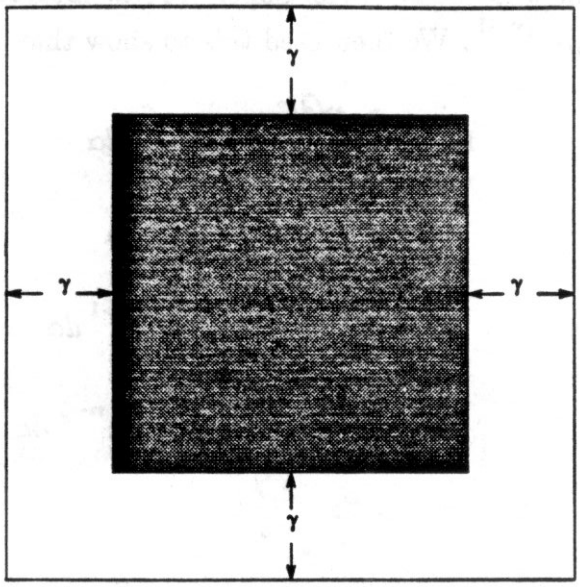


Figure 1.12. The unshaded region is $R(\gamma)$.

Let $R(\gamma)$ be the square ring (Figure 1.12) with width γ around the border of the hypercube:

$$R(\gamma) = \{\mathbf{x} = (x_1, \ldots, x_n) \,|\, \forall i \; 0 \leq x_i \leq \gamma \text{ or } (1-\gamma) \leq x \leq 1\}.$$

From the definition of conditional expectation

$$\mathrm{E}(Y) = \mathrm{E}(Y \,|\, \mathbf{p_1} \in R(\gamma)) \cdot \Pr(\mathbf{p_1} \in R(\gamma)) \; + \; \mathrm{E}(Y \,|\, \mathbf{p_1} \notin R(\gamma)) \cdot \Pr(\mathbf{p_1} \notin R(\gamma)). \tag{1.27}$$

If $\alpha \geq 1/2$, then independent of the location of $\mathbf{p_1}$, we know that $\Pr(Y \geq \alpha \,|\, \mathbf{p_1}) \leq \left(1 - \omega_d \alpha^2 / 2^d\right)^{n-1}$. Again the same type of calculations we performed in (b) show that

$$\mathrm{E}(Y \,|\, \mathbf{p_1} \in R(\gamma)) = \int \Pr\left(Y \geq \alpha \,|\, \mathbf{p_1} \in R(\gamma)\right) d\alpha = O(n^{-1/d}).$$

Also $\Pr(\mathbf{p_1} \in R(\gamma)) = \gamma$ so the first term on the right hand side of (1.27) is $O(\gamma n^{-1/d})$.

Now we calculate the second term. For $\mathbf{p_1} \notin R(\gamma)$ if $\alpha \leq \gamma$ then the entire neighborhood $B(\mathbf{p_1}, \alpha)$ is totally contained in the hypercube and

$$\Pr(Y \geq \alpha \,|\, \mathbf{p_1} \notin R(\gamma)) = (1 - \omega_d \alpha^d)^{n-1}.$$

If $\alpha' = \gamma' = (\ln n / n)^{1/d}$

$$\Pr(Y \geq \alpha' \mid \mathbf{p_1} \notin R(\gamma')) = O\left(\frac{1}{n}\right).$$

Integrating as before gives

$$\mathrm{E}(Y \mid \mathbf{p_1} \notin R(\gamma')) = \sigma_d n^{-1/d}\left[1 + O\left(\frac{1}{n}\right)\right].$$

Finally, $\Pr(\mathbf{p_1} \notin R(\gamma')) = 1 - 4\gamma'(1-\gamma') = 1 + O(\gamma')$, and putting it all together we have

$$\begin{aligned}\mathrm{E}(Y) &= O(\gamma n^{-1/d}) + \sigma_d n^{-1/d}\left[1 + O\left(\frac{1}{n}\right)\right][1 + O(\gamma')] \\ &= \sigma_d n^{-1/d}\left[1 + O\left(\left(\frac{\ln n}{n}\right)^{1/d}\right)\right].\end{aligned}$$

§1.4.3 Closest Pair

We are now going to look at a slightly different problem, calculating the expected distance between the closest pair along n points I.I.D. either $\mathbf{U}[0,1]^d$ or $\mathbf{U}< 0,1 >^d$. The random variable that we want to analyze is

$$Z(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \min_{1 \leq i < j \leq n} d(\mathbf{p}_i, \mathbf{p}_j).$$

We will prove that, under both distributions, $\mathrm{E}(Z) = \left(\Theta(n^{-1/2d})\right)$. Table 1.2 gives the precise results derived in this section.

From Z's definition we see that $Z \geq \alpha$ if and only $\mathbf{p_i}$ is outside the α-neighborhoods of all the $\mathbf{p_j}$ with $j < i$, i.e.

$$Z \geq \alpha \Leftrightarrow \forall i,\ \mathbf{p_i} \notin \bigcup_{j<i} B(\mathbf{p_j}, \alpha). \tag{1.28}$$

An alternate criterion is that all of the $\alpha/2$ neighborhoods around the points in the set are pairwise disjoint (Figure 1.13) i.e.

$$Z \geq \alpha \Leftrightarrow \forall i \neq j,\ B(\mathbf{p_i}, \alpha/2) \cap B(\mathbf{p_j}, \alpha/2) = \emptyset \tag{1.29}$$

Before we start analyzing $\mathrm{E}(Z)$ we prove a simple, well known, fact that deterministically bounds Z.

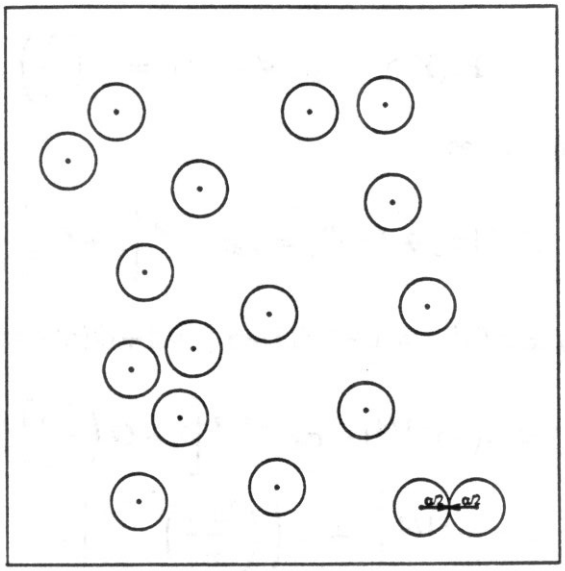


Figure 1.13. In this diagram Z, the distance between the closest pair, has value α. The large circles centered at the points have radius $\alpha/2$. Notice that all of the circles are disjoint and the two in the lower right hand corner are just touching each other.

Lemma 1.4: For any n points, $\mathbf{p}_1, \ldots, \mathbf{p}_n$, in $[0,1]^d$ or $< 0,1 >^d$

$$Z(\mathbf{p}_1, \ldots, \mathbf{p}_n) \leq sn^{-1/d}$$

where s is a constant dependent on both d and whether the distance function used is that of the hypercube or that of the hypertorus.

Proof: This follows directly from (1.29). For $\alpha \leq 1/2$ the area of the intersection of $B(\mathbf{p}_i, \alpha/2)$ with $[0,1]^d$ or $< 0,1 >^d$ is at least $c\alpha^d$ where c is a constant dependent on d and whether we are discussing the torus or the cube. Since these balls are disjoint we must have $cn\alpha^d \leq 1 = \text{Area}\left([0,1]^d\right) = \text{Area}\left(< 0,1 >^d\right)$. The proof of the lemma follows.

(a) As usual we will first examine n points I.I.D. $\mathbf{U}< 0,1 >^d$. We need to find the value of $\Pr(Z \geq \alpha)$. Think of constructing the event $Z \geq \alpha$ on a point-by-point basis. We use (1.28). First we place $\mathbf{p_1}$ anywhere. Next we place $\mathbf{p_2}$ anywhere outside of $B(\mathbf{p_1}, \alpha)$. Next we place $\mathbf{p_3}$ anywhere outside of $B(\mathbf{p_1}, \alpha) \cup B(\mathbf{p_2}, \alpha)$. We continue in this fashion until all of the points have been placed. The k-th step is

place $\mathbf{p_k}$ *anywhere outside of* $\cup_{i<k} B(\mathbf{p_i}, \alpha)$. Written in terms of probabilities

$$\begin{aligned}
\Pr(Z(\mathbf{p_1}, \ldots, \mathbf{p_n}) \geq \alpha) &= \Pr\left(Z\left(\mathbf{p_1}, \mathbf{p_2}\right) \geq \alpha\right) \\
&\quad \cdot \Pr\left(Z\left(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\right) \geq \alpha \mid Z\left(\mathbf{p_1}, \mathbf{p_2}\right) \geq \alpha\right) \\
&\quad \cdot \Pr\left(Z\left(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}\right) \geq \alpha \mid Z\left(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\right) \geq \alpha\right) \cdots \\
&= \prod_{k \leq n} \Pr\left(Z\left(\mathbf{p_1}, \ldots, \mathbf{p_k}\right) \geq \alpha \mid Z\left(\mathbf{p_1}, \ldots, \mathbf{p_{k-1}}\right) \geq \alpha\right) \\
&= \prod_{k \leq n} \Pr\left(\mathbf{p_k} \notin \bigcup_{i<k} B(\mathbf{p_i}, \alpha) \mid Z\left(\mathbf{p_1}, \ldots, \mathbf{p_{k-1}}\right) \geq \alpha\right)
\end{aligned}$$

While this might look forbidding, it is not hard to see[4] that the generic term is just

$$1 - \mathrm{E}\left(\mathrm{Area}\left(\bigcup_{i<k} B(\mathbf{p_i}, \alpha)\right) \mid Z\left(\mathbf{p_1}, \ldots, \mathbf{p_{k-1}}\right) \geq \alpha\right) \tag{1.30}$$

given that $Z(\mathbf{p_1}, \cdots, \mathbf{p_{k-1}}) \geq \alpha$. The problem we have in evaluating this expression is that the neighborhoods of the points might intersect and therefore we can't calculate the areas explicitly. We can bound (1.30) from below with $(1-(k-1)\omega_d\alpha^d)$. By using (1.29) we can also bound it from above ; even though the α neighborhoods might intersect we know that the $\alpha/2$ ones are disjoint and therefore (1.30) can be upper bounded by $(1-(k-1)\omega_d\alpha^d/2^d)$. Since $\mathrm{E}(Z) = \int_0^{sn^{-1/d}} \Pr(Z \geq \alpha)\, d\alpha$, we have just shown that

$$\int_0^{sn^{-1/d}} \prod_{i<n}(1 - i\omega_d\alpha^d)\, d\alpha \;\leq\; \mathrm{E}(Z) \;\leq\; \int_0^{sn^{-1/d}} \prod_{i<n}(1 - i\omega_d\alpha^d/2^d)\, d\alpha. \tag{1.31}$$

We will evaluate the left hand side; the right hand side is evaluated similarly. We use the asymptotic relations provided at the beginnning of this section.

$$\begin{aligned}
\prod_{i<n}(1 - i\omega_d\alpha^d) &= \exp\left(\sum_{i<n} \ln(1 - i\omega_d\alpha^d)\right) \\
&= \exp\left(\sum_{i<n} \left[-i\omega_d\alpha^d + O(i^2\alpha^{2d})\right]\right) \\
&= \exp\left(-\omega_d\alpha^d n(n-1)/2 + O(n^3\alpha^{2d})\right) \\
&= \exp\left(-\omega_d\alpha^d n^2/2\right)\, \left[1 + O(n\alpha^d) + O(n^3\alpha^{2d})\right].
\end{aligned}$$

[4] Because of Lemma 1.4 we no longer have to worry about $\alpha \geq 1/2$ since α will always be $O(n^{-1/d})$.

From this we see that if $\alpha > \left(\frac{2\ln n}{n^2}\right)^{1/d}$ then the product is $O(1/n)$ because the exponential term is the dominant factor. When $\alpha < \left(\frac{2\ln n}{n^2}\right)^{1/d}$ both of the $O()$ terms on the right hand side of the last line evaluate to $O\left(\frac{\ln^2 n}{n}\right)$. Therefore

$$\int_0^{sn^{-1/d}} \prod_{i<n}(1-i\omega_d\alpha^d)\,d\alpha = \int_0^{\left(\frac{2\ln n}{n^2}\right)^{1/d}} e^{-\omega_d\alpha^d n^2/2}\left[1+O\left(\frac{\ln^2 n}{n}\right)\right]d\alpha + O\left(\frac{1}{n}\right)$$

$$= \int_0^{\left(\frac{2\ln n}{n^2}\right)^{1/d}} e^{-\omega_d\alpha^d n^2/2} d\alpha + O\left(\frac{1}{n}\right).$$

We can evaluate this integral by the substitution $v = \omega_d\alpha^d n^2/2$ to get

$$\left(\frac{2}{w_d}\right)^{1/d}\left(\frac{n^{-2/d}}{d}\right)\int_0^{\omega_d \ln n} v^{\frac{1}{d}-1}e^{-v}\,dv$$

where the integral is 1 minus an incomplete Gamma function. Using (1.10) we find that

$$1-\Gamma(1/d,\omega_d\ln n) = \Gamma(1/d) + O\left(\frac{1}{n}\right).$$

We have just shown that

$$\mathrm{E}(Z) \geq 2^{1/d}\sigma_d n^{2/d}\left[1+O\left(\frac{1}{n}\right)\right].$$

The same calculations that were used to evaluate the left side of (1.31) can be used to evaluate its right hand side (substitute $\omega_d/2^d$ for ω_d) giving

$$\mathrm{E}(Z) \leq 2\left(\frac{2}{w_d}\right)^{1/d}\left(\frac{\Gamma(1/d)n^{-2/d}}{d}\right)\left[1+O\left(\frac{1}{n}\right)\right].$$

We have shown that

$$2^{1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right] \leq \mathrm{E}(Z) \leq 2^{1+1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]. \qquad (1.32)$$

(a') The analysis of Z when the n points are I.I.D. $\mathbf{U}[0,1]^d$ is similar. As in sections §1.4.2(a') and §1.4.2(b') we have to take into account the fact that not all of $B(\mathbf{p}_i,\alpha)$ is contained in $[0,1]^d$. But, again as before, we know that at least 2^{-d} of each of the balls is in $[0,1]^d$. Therefore

$$\int_0^{sn^{-1/d}} \prod_{i<n}(1-i\omega_d\alpha^d)\,d\alpha \;\leq\; \mathrm{E}(Z) \;\leq\; \int_0^{sn^{-1/d}} \prod_{i<n}(1-i\omega_d\alpha^d/4^d)\,d\alpha. \qquad (1.33)$$

This differs from (1.31) only in the extra factor of 2^{-d} in the right hand expression so the exact same calculations that we used to go from (1.31) to (1.32) transforms (1.33) to

$$2^{1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right] \leq \mathrm{E}(Z) \leq 2^{2+1/d}\sigma_d n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]. \qquad (1.34)$$

§1.4.4 Scaling

Let us return for the moment to §1.4.2(b) of this and rexamine the behavior of

$$Y(\mathbf{p_1},\cdots,\mathbf{p_n}) = \min_{1<i\leq n} d(\mathbf{p_1},\mathbf{p_i})$$

where the $\mathbf{p_i}$ are I.I.D. $\mathbf{U}< 0,1 >^d$. We showed there that

$$\mathrm{E}(Y) = \sigma_d n^{-1/d}\left[1+O\left(\frac{1}{n}\right)\right]. \qquad (1.25)$$

This is fine as far as it goes but what does it really tell us about the distribution of Y? Is it always "close" to $\mathrm{E}(Y)$ in the sense that its distribution is concentrated around $\mathrm{E}(Y)$, or is Y very spread out? Since[5] $\mathrm{E}(Y_n)$ is a fast decreasing function of n the word "close" will have to mean something very different for $n = 100$ than it does when $n = 1,000,000$. To make such a distinction rigorous we introduce the concept of *scaling*, multiplying the Y_n by appropriate factors and examining these scaled random variables on some absolute scale. In §1.4.2 we saw that the distance from a point to its nearest neighbor was $\Theta(n^{1/d})$. Intuitively then, $n^{1/d}$ might be an appropriate scaling factor since multiplying interpoint distances by it would transform them onto an absolute scale. In fact $n^{1/d}$ is a good scaling factor. We define

$$Y_n' = n^{1/d} Y_n.$$

Equation (1.25) tells us that

$$\mathrm{E}(Y_n') \to \sigma_d, \qquad n \to \infty \qquad (1.35)$$

but we can prove the following much stronger result:

---

[5] For the rest of this section we will write Y_n (and Z_n) instead of Y (and Z) to make the reliance on n more explicit.

Lemma 1.4: Let X be an exponential random variable with parameter ω_d. Then

$$Y_n' \to \sqrt[d]{X}, \qquad n \to \infty \tag{1.36}$$

where the mode of convergence is in distribution, i.e.

$$\forall x,\ \Pr(Y_n' > x) \to \Pr(\sqrt[d]{X} > x), \qquad n \to \infty. \tag{1.37}$$

We will delay the proof for a moment to see what type of information this gives us. If $f(n)$ is a function that goes to 0, e.g. $\ln^{-1} n$, then (1.18) tells us that

$$\Pr(Y_n' \le f(n)) \to 0 \quad \text{or} \quad \Pr(Y_n \le f(n)n^{-1/d}) \to 0.$$

Similarly if $f(n)$ is a function that goes to ∞, e.g. $\ln n$,

$$\Pr(Y_n' \ge f(n)) \to 0 \quad \text{or} \quad \Pr(Y_n \ge f(n)n^{-1/d}) \to 0.$$

Therefore, if α is outside the narrow range "a constant times $n^{-1/d}$" then with probability going to one, Y_n is not close to α. If α is some "constant times $n^{-1/d}$" then there is a positive probability that Y_n will be close to it. More specifically, from (1.18) we know that for all positive constants $0 \le c_1 < c_2$

$$\Pr(c_1 n^{-1/d} \le Y_n \le c_2 n^{-1/d}) = \Pr(c_1 n \le Y_n' \le c_2) \to e^{\omega_d c_1^d} - e^{\omega_d c_2^d} > 0.$$

Proof of the lemma: The analysis in the paragraph preceeding (1.32) shows that

$$\Pr(Y_n \ge \alpha) = (1 - \omega_d \alpha_d)^{n-1}. \tag{1.38}$$

Using the definition of Y_n' and the asymptotic relations (1.14) we find, for constant c,

$$\begin{aligned} \Pr(Y_n' \ge c) &= \Pr(Y_n \ge cn^{-1/d}) \\ &= \left(1 - \frac{\omega_d c^d}{n}\right)^{n-1} \\ &= e^{-\omega_d c^d}\left[1 + O\left(\frac{1}{n}\right)\right]. \end{aligned} \tag{1.39}$$

Since $\Pr(X \ge c^d) = e^{-\omega_d c^d}$ we are done.

Q.E.D.

The same scaling technique that we used for the problem of §1.4.2(b) also works for the problem of part §1.4.2(b') where the points are I.I.D. $\mathbf{U}[0,1]^d$. Writing an equation analogous to (1.38) for probabilities instead of expectations we find that

$$\begin{aligned}\Pr(Y \geq \alpha) = \Pr(Y \geq \alpha \,|\, \mathbf{p_1} \in R(\gamma)) \cdot \Pr(\mathbf{p_1} \in R(\gamma)) \\ + \Pr(Y \geq \alpha \,|\, \mathbf{p_1} \notin R(\gamma)) \cdot \Pr(\mathbf{p_1} \notin R(\gamma)).\end{aligned}$$

Setting $\gamma = \alpha$ and using the same techniques we used to analyze (1.29) we find that

$$\Pr(Y_n \geq \alpha) = \left(1 - \omega_d \alpha^d\right)^{n-1} + O\left[\left(1 - \omega_d \alpha^d / 2^d\right)^{n-1} \alpha\right]. \tag{1.40}$$

Again setting

$$Y_n' = Y_n n^{1/d}$$

we can prove Lemma 1.5 for this case as well by substituting (1.40) in place of (1.38). The same consequences follow.

We won't be able to get as nice a set of results for Z_n, the distance between the closest pair. This is because we weren't able to find an exact asymptotic formula like (1.38) for Z_n. The best we were able to accomplish when the points were $\mathbf{U}< 0,1 >^d$ was upper and lower bounding Z_n with

$$\begin{aligned}e^{\frac{-\omega_d \alpha^d n^2}{2}} \left[1 + O(n\alpha^d) + O(n^3\alpha^{2d})\right] \leq \Pr(Z_n \geq \alpha) \\ \leq e^{\frac{-\omega_d \alpha^d n^2}{2^{d+1}}} \left[1 + O(n\alpha^d) + O(n^3\alpha^{2d})\right].\end{aligned} \tag{1.41}$$

We still get some utility out of the scaling argument though. The appropriate scaling factor is $n^{2/d}$. Defining

$$Z_n' = Z_n n^{2/d}$$

we see that

$$e^{\frac{-\omega_d c^d}{2}} \left[1 + \left(\frac{1}{n}\right)\right] \leq \Pr(Z_n' \geq c) \leq e^{\frac{-\omega_d c^d}{2^{d+1}}} \left[1 + \left(\frac{1}{n}\right)\right].$$

Although we no longer can prove convergence in distribution to some random variable, this still shows that

$$\Pr(Z_n' \leq f(n)) \to 0, \qquad f(n) \to 0$$

and

$$\Pr(Z_n' \geq f(n)) \to 0, \qquad f(n) \to \infty$$

with the attendant consequences for Z_n. Furthermore there is still a positive probability of Z_n' being larger than any constant.

Finally notice that a similar phenomenon ocurrs if the points are $\mathbf{U}[0,1]^d$. The only difference is that we replace (1.41) with

$$e^{\frac{-\omega_d \alpha^d n^2}{2}} \left[1 + O(n\alpha^d) + O(n^3\alpha^{2d})\right] \leq \Pr(Z_n \geq \alpha)$$
$$\leq e^{\frac{-\omega_d \alpha^d n^2}{2^{2d+1}}} \left[1 + O(n\alpha^d) + O(n^3\alpha^{2d})\right]$$

and get

$$e^{-\omega_d c^d/2}\left[1 + \left(\frac{1}{n}\right)\right] \leq \Pr(Z_n' \geq c) \leq e^{-\omega_d c^d/2^{2d+1}}\left[1 + \left(\frac{1}{n}\right)\right].$$

To review: We choose n points, $\mathbf{p}_1, \ldots, \mathbf{p}_n$, I.I.D. $\mathbf{U}[0,1]^d$ or $\mathbf{U}< 0,1 >^d$ and define Y_n to be the minimum distance between $\mathbf{p}_1$ and its nearest neighbor. In this section we showed that as $n \to \infty$, the random variable $n^{1/d}Y_n$ converges in distribution to $\sqrt[d]{X}$ where X is an exponential random variable with parameter ω_d. This told us that Y_n is very concentrated around the value $n^{1/d}$, i.e.

$$\Pr(Y_n > n^{-1/d} \lg n) \to 0, \qquad n \to \infty$$

and

$$\Pr(Y_n < n^{-1/d}/\lg n) \to 0, \qquad n \to \infty.$$

We also applied a scaling argument to the random variable Z_n, the distance between the closest pair among the $\mathbf{p}_i$. We showed that $Z_n \sim n^{-2/d}$. Even though we were not able to prove that $n^{2/d}Z_n$ converged in distribution to some random variable, we were able to show that Z_n is very heavily concentrated around $n^{-2/d}$.

§1.4.5 Extensions to Other Metrics

In the previous subsections we analyzed the expected values of inter-point distances (closest point and closest pair) assuming that distance was defined under the L_2 metric. The L_2 metric defines the distance between two points $\overline{\mathbf{x}} = (x_1, \ldots, x_d)$ and $\overline{\mathbf{y}} = (y_1, \ldots, y_d)$ in $\mathcal{R}^d$ to be

$$d(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \sqrt{\sum_{1 \leq i \leq d} (x_i - y_i)^2}$$

and the distance between two such points in the d-dimensional hypertorus to be

$$d(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \sqrt{\sum_{1 \le i \le d} (\min(|x_i - y_i|, 1 - |x_i - y_i|))^2}.$$

In this section we generalize our results to other definitions of distance, specifically, for L_∞ and L_p distances. The L_∞ distance is defined in $\mathcal{R}^d$ by

$$d_\infty(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \max_{1 \le i \le d} |x_i - y_i|.$$

The analogous distance function in the d-dimensional hypertorus is

$$d_\infty(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \max_{1 \le i \le d} \big(\min(|x_i - y_i|, 1 - |x_i - y_i|)\big).$$

The L_p distance ($p \ge 1$) function in $\mathcal{R}^d$ is defined by

$$d_p(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \left(\sum_{1 \le i \le d} |x_i - y_i|^p \right)^{1/p}.$$

In the hypertorus we have

$$d_p(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \left(\sum_{1 \le i \le d} (\min(|x_i - y_i|, 1 - |x_i - y_i|)) \right)^{1/p}.$$

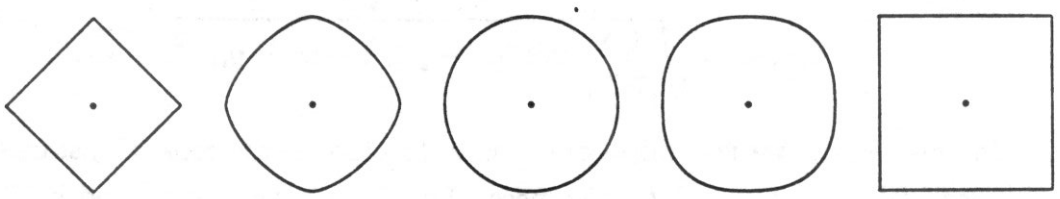
The L_2 metric is the standard distance function. The L_∞ metric can be thought of as $\lim_{p \to \infty} L_p$. In what follows, the phrase "L_p metrics" will be taken to mean both the L_p metrics and the L_∞ metric.

The first step in generalizing our analysis to include these new distance functions is to generalize our definition of a neighborhood. We do this in the natural way.

$$B_p(\overline{\mathbf{x}}, \alpha) = \{\overline{\mathbf{y}} \mid d_p(\overline{\mathbf{x}}, \overline{\mathbf{y}}) < \alpha\}.$$

Figure 1.14 illustrates the two-dimensional unit-neighborhoods under the $L_{1/2}$, L_1, $L_{3/2}$, L_2, and L_∞, metrics

The second step in generalizing our analysis is isolating those properties of $B(\overline{\mathbf{x}}, \alpha)$ that are used in §1.4.2 and §1.4.3 to analyze Y and Z as functions of random points $\mathbf{p}_1, \ldots, \mathbf{p}_n$. In §1.4.2 we analyze Y, the distance between $\mathbf{p}_1$ and its closest neighbor. We find that there are only two properties of $B(\overline{\mathbf{x}}, \alpha)$ that are used in §1.4.2. The first is that for $\alpha \le 1/2$



(a) (b) (c) (d) (e)

Figure 1.14. These are two-dimensional unit neighborhoods under different metrics: **(a)** is under the L_1 metric, **(b)** $L_{3/2}$, **(c)** L_2, **(d)** $L_{5/2}$, and **(e)** L_∞.

---

the area of an α neighborhood (in both a d-dimensional hypercube and a d-dimensional hypertorus) is directly proportional to α^d :

$$\text{Area}\left(B(\overline{\mathbf{x}}, \alpha)\right) = \omega_d \alpha^d, \qquad \alpha \leq 1/2. \tag{1.42}$$

The second is that, again for $\alpha \leq 1/2$, no matter where $\overline{\mathbf{x}}$ is located, at least $1/2^d$ of the area of $B(\overline{\mathbf{x}}, \alpha)$ is in $[0,1]^d$.

$$\omega_d \alpha^d / 2^d \leq \text{Area}\left[B(\overline{\mathbf{x}}, \alpha) \cap [0,1]^d\right] \leq \omega_d \alpha^d, \qquad \alpha \leq 1/2. \tag{1.43}$$

It is not hard to see that the L_p metrics also have these properties. To do this we must introduce some new notation. Let $w_{p,d}$ be the area/volume of the unit ball defined by the L_p metric:

$$w_{p,d} = \text{Area}\left(B_p(\mathbf{0}, 1)\right).$$

For example, $w_{2,d}$ is just the old w_d, while $w_{1,d} = 1/d!$ and $w_{\infty,d} = 4$. We can replace (1.42) and (1.43) by

$$\text{Area}\left(B_p(\overline{\mathbf{x}}, \alpha)\right) = \omega_d \alpha^d, \qquad \alpha \leq 1/2 \tag{1.44}$$

and

$$\omega_{p,d} \alpha^d / 2^d \leq \text{Area}\left[B_p(\overline{\mathbf{x}}, \alpha) \cap [0,1]^d\right] \leq \omega_{p,d} \alpha^d, \qquad \alpha \leq 1/2 \tag{1.45}$$

where p can also take on the value ∞.

We then substitute these properties in place of (1.42) and (1.43) in §1.4.2 to generalize its analysis of $\mathtt{E}(Y)$ to the L_p metrics. We present the generalized results in Table 1.3.

In §1.4.3 we analyze Z, the distance between the closest pair of points. This analysis also uses (1.42) and (1.43). It needs two other facts about $B(\overline{\mathbf{x}}, \alpha)$ as well. The first fact, the only other one needed to lower bound $\mathrm{E}(Z)$, is

$$Z \geq \alpha \Leftrightarrow \forall i, \mathbf{p_i} \notin \bigcup_{j<i} B(\mathbf{p_j}, \alpha). \tag{1.28}$$

This fact remains true under the L_p metrics. Therefore the lower bounds on $\mathrm{E}(Z)$ calculated in §1.4.3 under the L_2 metric are also valid under the L_p metrics after replacing w_d with $w_{p,d}$. The second fact, the only other one needed to upper bound $\mathrm{E}(Z)$, is

$$Z \geq \alpha \Leftrightarrow \forall i \neq j, B(\mathbf{p_i}, \alpha/2) \cap B(\mathbf{p_j}, \alpha/2) = \emptyset. \tag{1.29}$$

We will show that this fact is true for any L_p metric ($p \geq 1$. If $p < 1$ then, in general, this last statement isn't true). First we prove the $\Leftarrow$ direction. Suppose that $Z < \alpha$. There is some pair $\mathbf{p}_i, \mathbf{p}_j$ such that $d_p(\mathbf{p}_i, \mathbf{p}_j) < \alpha$. From the definition of an L_p metric we find that

$$d_p\left(\mathbf{p}_i, \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\right) = d_p\left(\mathbf{p}_j, \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\right) < \frac{\alpha}{2}.$$

Thus $B(\mathbf{p_i}, \alpha/2) \cap B(\mathbf{p_j}, \alpha/2) \neq \emptyset$.

Now we prove the $\Rightarrow$ direction. Suppose that $Z \geq \alpha$ but there is some pair $\mathbf{p}_i, \mathbf{p}_j$ and a point $\mathbf{q}$ such that $\mathbf{q} \in B(\mathbf{p_i}, \alpha/2) \cap B(\mathbf{p_j}, \alpha/2)$. The triangle inequality [**Ru**] yields

$$d_p(\mathbf{p}_i, \mathbf{p}_j) \leq d_p(\mathbf{p}_i, \mathbf{q}) + d_p(\mathbf{q}, \mathbf{p}_j) < \alpha/2 + \alpha/2 = \alpha.$$

This contradicts the assumption that $Z \geq \alpha$, an assumption which forces $d_p(\mathbf{p}_i, \mathbf{p}_j) \geq \alpha$.

We have just shown that (1.29) is true for any L_p metric. Therefore the upper bounds on $\mathrm{E}(Z)$ calculated in §1.4.3 under the L_2 metric are also valid under any L_p metric ($p \geq 1$) after replacing w_d with $w_{p,d}$. Table 1.4 presents the new upper and lower bounds.

> What differentiates the interpoint distances's analysis from Quickhull's is that here we had to worry about the how the locations of the points interacted with each other. The salient characteristics of this analysis were

n points I.I.D.	$\mathtt{E}(Y)$
$\mathbf{U}<0,1>$	$\frac{1}{2n}$
$\mathbf{U}[0,1]$	$\frac{1}{2n}+\frac{1}{n(n+1)}$
$\mathbf{U}<0,1>^d$	$\sigma_{p,d}n^{-1/d}\left[1+O\left(\frac{\ln n}{n}\right)\right]$
$\mathbf{U}[0,1]^d$	$\sigma_{p,d}n^{-1/d}\left[1+O\left(\left(\frac{1}{n}\right)^{1/d}\right)\right]$

Table 1.3: This table presents the results derived in §1.4.2 generalized to any L_p (L_∞) metric. Y is the minimum distance between $\mathbf{p}_1$ and $\mathbf{p}_2,\ldots,\mathbf{p}_n$ where the $\mathbf{p}_i$ are chosen I.I.D. from some distribution. The first column is the appropriate distribution. The second column is $\mathtt{E}(Y)$ under this distribution. The value $\sigma_{p,d}=\frac{\Gamma(1/d)}{d\,(\omega_{p,d})^{1/d}}$ where ω_d^p is the volume of the unit sphere $B(\mathbf{0},1)$ defined by the L_p (L_∞) distance function.

n points I.I.D.	lower bounds on $\mathtt{E}(Z)$	upper bounds on $\mathtt{E}(Z)$
$\mathbf{U}<0,1>^d$	$2^{1/d}\sigma_{p,d}\,n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$	$2^{1+1/d}\sigma_{p,d}\,n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$
$\mathbf{U}[0,1]^d$	$2^{1/d}\sigma_{p,d}\,n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$	$2^{2+1/d}\sigma_{p,d}\,n^{-2/d}\left[1+O\left(\frac{1}{n}\right)\right]$

Table 1.4: This table presents the results derived in §1.4.3 generalized to any L_p (L_∞) metric. Z is the minimum distance between the closest pair among $\mathbf{p}_1,\ldots,\mathbf{p}_n$ where the $\mathbf{p}_i$ are chosen I.I.D. from some distribution. The first column is the appropriate distribution. The second column contains lower bounds on $\mathtt{E}(Z)$; the third, upper bounds.

(1) Integration over continuous probability measures of functions that evaluate to Gamma and Beta functions.
(2) Identification of low probability events (such as $\mathbf{p_1} \notin R(\gamma)$ in §1.4.2(b')). Ignoring these events and conditioning the rest of the problem on their not occurring.
(3) Appropriate scaling of random variables.

In the remainder of this thesis these three characteristics will recur often.

Chapter 2. A Simple Convex Hull Algorithm

§2.1 Introduction

In this chapter we will present and analyze a simple, intuitive preprocessing algorithm for convex hulls. In §2.2 we present the algorithm. It is a simplified version of one proposed by Devroye and Toussaint in [**DT**]. A Pascal implementation is given along with a worked example. In §2.3 we analyze the expected number of points remaining after running the algorithm. Our probabilistic assumption is that the points are uniformly distributed in the unit square. Let S be the number of points remaining after running the algorithm on n points. The analysis given in [**DT**] shows that $\mathrm{E}(S) = O(\sqrt{n})$. This result was only of theoretical interest because the constant in the $O(\sqrt{n})$ was greater than $2,000,000$. Our main result is an analysis which proves that that $E(S) \leq c\sqrt{n}$ where c is a small constant less than 8. This shows that the algorithm is practical as well as simple. §2.4 discusses the combined behavior of the preprocessing routine followed by some standard convex hull algorithms. It also compares experimental results to those predicted by the mathematical analysis. §2.5 proves the benefit of performing a tight analysis of the algorithm. In it we present a variant of our convex hull preprocessing algorithm and discuss why we would prefer one algorithm over the other. over the other. To do this we need tight analyses of both algorithms. §2.6 discusses how to extend the algorithm and its analysis to higher dimensional spaces. The result there is that $E(S) = O(n^{(d-1)/d})$ where d is the index of the dimension. We conclude, in §2.7, by reviewing our results and presenting a simple intuitive argument why they should be true.

§2.2 The Algorithm

In §1.4 we introduced the concept of convex hulls in $\Re^2$. If $S = \{p_1, \ldots, p_n\} \subseteq \Re^2$ is a finite set of points then we defined $CH(S)$, the *convex hull* of S, to be the boundary of the smallest convex polygon containing S.

Convex Hulls are among the most intuitive and useful structures studied by geometers and as such they have been extensively examined and many elegant algorithms exist for their computation [**PS**]. In addition to these there is also a very simple heuristic attributed to W.F. Eddy and R.W. Floyd [**Ed**] [**Se1**] for

improving the performance of any such algorithm. Pick any four points from S. These points are the vertices of some quadrilateral – in the degenerate case a triangle or line segment – $\mathtt{Q}$. A straightforward generalization of corollary 1.2 shows that $\mathtt{Q}$ is contained within the convex hull of $p_1, \ldots, p_n$.

We can therefore eliminate all points that are inside $\mathtt{Q}$ without discarding any information essential to the construction of the convex hull (Figure 2.1). Our goal in this section is to identify a computationally simple method for choosing the four points such that after the elimination step only a few of the p_i will remain on average.

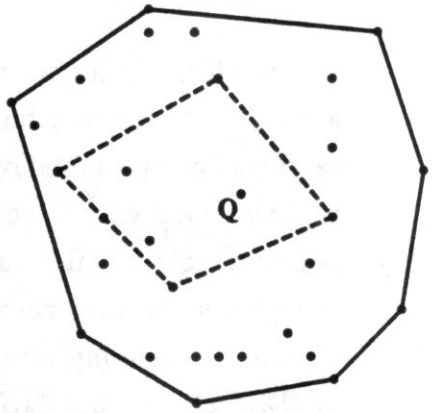


Figure 2.1

Specifically, suppose that $p_1, \ldots, p_n$ are I.I.D. $\mathbf{U}[0,1]^2$. In §2.3 we will prove that the following choice of points gives us good average case behavior: Pick the four points that minimize the four functions $\pm x \pm y$. Equivalently, these are the two points that minimize the functions $x \pm y$ along with the two that maximize the same functions, a fact that we will utilize in programming our algorithm. Note that these choices might not be well defined since there can be two or more points that minimize any of the functions. We would like our algorithm to always work properly, not just when its input points are in general position (points are in general position if no three of them are collinear). To insure this we will specify the points with the lowest indices that minimize the functions. Thus the four points (Figure 2.2a) are

$$\begin{aligned}
q_1 = (x_1^q, y_1^q) = p_k \qquad & k = \min_j \{\forall i : +x_j + y_j \leq +x_i + y_i\} \\
q_2 = (x_2^q, y_2^q) = p_k \qquad & k = \min_j \{\forall i : +x_j - y_j \leq +x_i - y_i\} \\
q_3 = (x_3^q, y_3^q) = p_k \qquad & k = \min_j \{\forall i : -x_j + y_j \leq -x_i + y_i\} \\
q_4 = (x_4^q, y_4^q) = p_k \qquad & k = \min_j \{\forall i : -x_j - y_j \leq -x_i - y_i\}.
\end{aligned}$$

The next step is to eliminate those p_i inside $\mathtt{Q}$. In reality, round-off error makes the problem of deciding whether a point is inside an arbitrary quadralateral computationally nontrivial. We therefore modify our elimination criteria somewhat. Instead of disposing of all of the points inside $\mathtt{Q}$ we find a large rectangle $\mathtt{R} \subseteq \mathtt{Q}$ and only dispose of those inside $\mathtt{R}$. Since these points are also inside $\mathtt{Q}$ we lose no essential information. If $\mathtt{Q}$ is not degenerate there is a nice geometric method for finding a

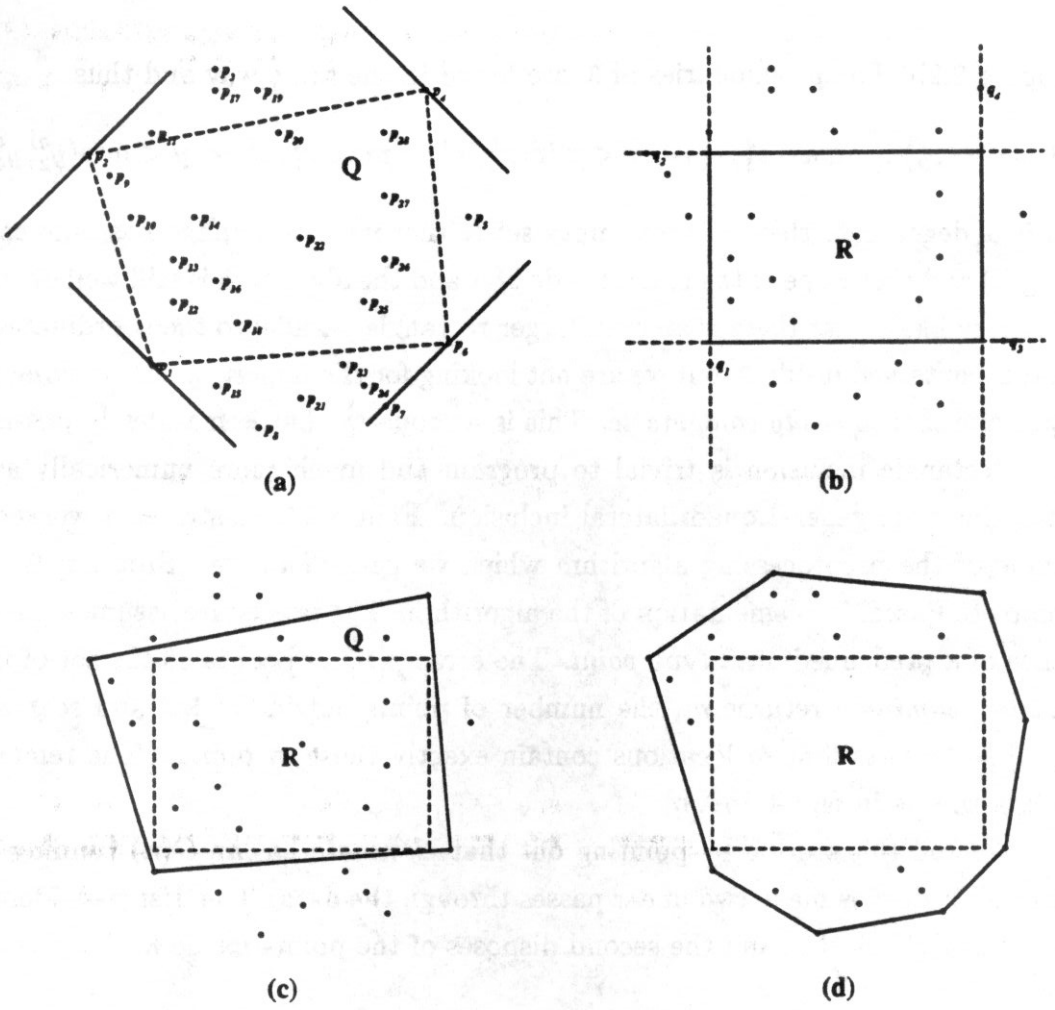


Figure 2.2. (a) shows the sweep lines hitting $q_1 = p_1$, $q_2 = p_2$, $q_3 = p_6$, and $q_4 = p_4$ and the quadrilateral Q they define. We resolve the tie between p_6 and p_7 by choosing p_6, the point with the lower index. (b) illustrates how the q_i are used to define R. In (c) we see that $\mathtt{R} \subseteq \mathtt{Q}$. After eliminating the points in R we run any convex hull algorithm to find (d).

---

good R. Minimizing the function $x + y$ can be thought of as sweeping in a 45° line from the lower left corner of infinity until it hits some p_i which will then be q_1. Similarly, minimizing $x - y$ sweeps in a line from the upper left hitting q_2, $-x + y$ sweeps in from the upper right hitting q_3, and $-x - y$ sweeps in from the lower right hitting q_4. By this reasoning the leftmost x-coordinate of R should be $\max(x_1^q, x_2^q)$ since these are the points encountered by sweeping in from the left. The rightmost x-coordinate will be the minimum of the two x-coordinates on the right, $\min(x_3^q, x_4^q)$

(Figure 2.2b).The y boundaries of R are found in the same way and thus

$$\mathtt{R} = \{(x,y): \quad \max(x_1^q, x_2^q) < x < \min(x_3^q, x_4^q) \quad \max(y_1^q, y_3^q) < y < \min(y_2^q, y_4^q)\}.$$

(If Q is degenerate then R is an empty set (either its x or y range has nonpositive length) and thus none of the p_i are inside of it and the algorithm is still well defined.) It is very likely that there is an even larger rectangle parallel to the coordinate axes that is contained inside Q but we are not looking for the *largest*, we are looking for a *large* one that is *easily computable.* This is a property that R manifestly possesses.

Rectangle inclusion is trivial to program and much more numerically stable than the more general quadrilateral inclusion. Figure 2.2 illustrates a worked example of the preprocessing algorithm which we call *eliminate*. Program 2.1 is a complete Pascal implementation of the algorithm. The procedure assumes the existence of a predefined data type *point*. The array $p[1 \ldots n]$ contains the set of input points. *eliminate* returns m, the number of points outside of R , and rearranges $p[\ldots]$ so that its first m locations contain exactly those m points. The remaining points will be in $p[m+1 \ldots n]$.

We end this section by pointing out that *eliminate* has an $O(n)$ running time since all it does is make two linear passes through the data. The first pass identifies the four vertices of Q and the second disposes of the points inside R.

§2.3 Analysis

Let $p_i = (x_i, y_i)$, $i = 1, \ldots, n$, be chosen I.I.D. $\mathbf{U}[0,1]^2$.Our goal is to analyze the asymptotic behavior of $\mathtt{E}(M)$, $n \to \infty$, where M is the number of points not deleted by *eliminate*:

$$\begin{aligned} M(p_1, \ldots, p_n) &= \textit{The number of points undeleted after running eliminate.} \\ &= \textit{The number of points outside} \ \mathtt{R}. \end{aligned}$$

We designed *eliminate* to be a simple algorithm, one that is easy both to understand and to implement. We *did not* design it to be simple to analyze. As we shall see the calculation of $\mathtt{E}(M)$ and related values is complicated by the fact that there is extreme probabilistic dependence between the values of q_k and even more so between the different boundaries of R. We will overcome this problem by finding the expectation of another simpler random variable and using it to upper bound $\mathtt{E}(M)$. To accomplish this we need to define the the following functions and regions (Figure 2.3a):

```
function eliminate(N : integer) : integer;
    var     i1, i2, i3, i4, j, M : integer;
            a1, a2, a3, a4, highx, lowx, highy, lowy : real;
            t : point;
begin
    i1:=1   i2:=1;  i3:=1;  i4:= 1;
    a1:=p[1].x + p[1].y;    a3:=a1;
    a2:=p[1].x - p[1].y;    a4:=a2;
    for j:=2 to N do
        begin
        if p[j].x + p[j].y < a1 then
            begin i1:=j; a1:=p[j].x + p[j].y;  end
        else if p[j].x +p [j].y > a4 then
            begin i4:=j; a4:=p[j].x + p[j].y;  end;
        if p[j].x - p[j].y < a2 then
            begin i2:=j; a2:=p[j].x - p[j].y;  end
        else if p[j].x - p[j].y > a3 then
            begin i3:=j; a3:=p[j].x - p[j].y;  end;
        end;
    lowx:= MAX(p[i1].x, p[i2].x);  highx:= MIN(p[i3].x, p[i4].x);
    lowy:= MAX(p[i1].y, p[i3].y);  highy:= MIN(p[i2].y, p[i4].y);
    M:=0;  j:=N;
    while j>M do
        if  lowx < p[j].x  and  p[j].x < highx
          and  lowy < p[j].y  and  p[j].y < highy then
            j:=j-1
        else
            begin
                M := M+1;
                t := p[M];  p[M] := p[j];    p[j] := t;
            end;
    eliminate := M;
end;
```

Program 2.1. A full Pascal implementation of *eliminate*. It is called with parameter N; the array $p[1 \ldots N]$ holds the points. After the function is initialized the **for** loop finds the four points that minimize the functions $\pm x \pm y$. *i1* holds the index of q_1, *i2* of q_2, etc. while *a1*, *a2*, *a3*, *a4* hold the respective minimized values of $\pm x \pm y$. The next step calculates the boundaries of R: *lowx*, *highx*, *lowy*, *highy*. The **while** loop rearranges the array so that the non-eliminated points are at its bottom. When it returns the function has value M, the number of points that weren't eliminated.

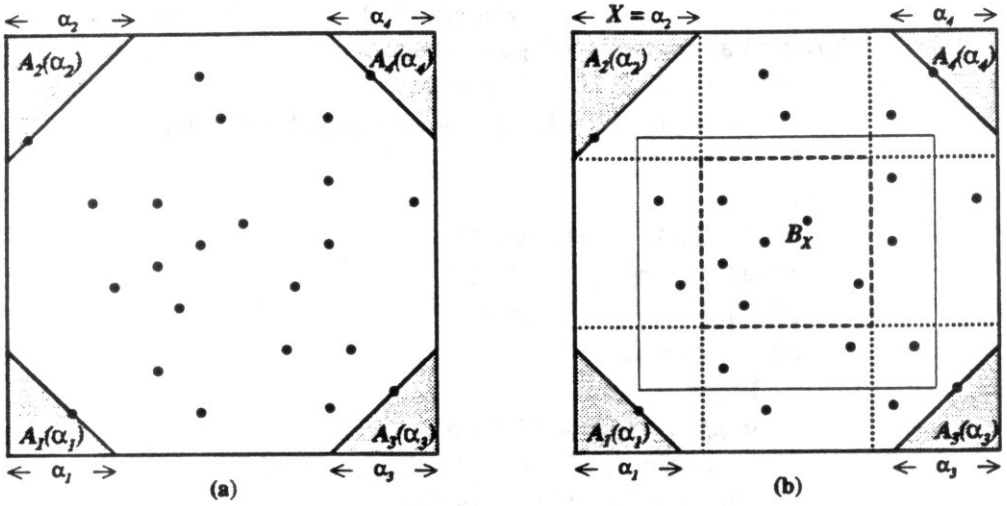


Figure 2.3. (a) gives some test points with the α_k noted and the $A_k(\alpha_k)$ (shaded areas) drawn. Note especially that $A_k(\alpha_k)$ are all empty with a point (q_k) on each of their defining diagonals. (b) is (a) with B_X (dashed box) and R (thin lined box) drawn in. The dotted boxes illustrate the relationship between the α_k and X. The q_k are all inside the dotted boxes so $B_X \subseteq \mathtt{R}$. By inspection $Z = 17$ and $M = 10$.

$$\alpha_1 = \min_{1\le i\le n} x_i + y_i \qquad A_1(\alpha) = \{(x,y):\ 0 \le x+y < \alpha\}$$

$$\alpha_2 = 1 + \min_{1\le i\le n} x_i - y_i \qquad A_2(\alpha) = \{(x,y):\ -1 \le x-y \le -(1-\alpha)\}$$

$$\alpha_3 = 1 - \max_{1\le i\le n} x_i - y_i \qquad A_3(\alpha) = \{(x,y):\ 1-\alpha < x-y \le 1\}$$

$$\alpha_4 = 2 - \max_{1\le i\le n} x_i + y_i \qquad A_4(\alpha) = \{(x,y):\ 2-\alpha < x+y \le 2\}$$

These definitions have very intuitive motivations: α_k is the distance from the k-th corner to the intersection of the corresponding diagonal and the horizontal (vertical) boundary of the unit square. $A_k(\alpha)$ is the right isosceles triangle with base sides α that fits into the k-th corner of the square. As defined above, α_k and A_k taken together have the following property upon which our analysis will strongly depend:

$A_k(\alpha_k)$ is empty and its defining hypotenuse has at least the one point q_k on it.

We further define

$$X = \max(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$$

$$B_X = \{(x,y) : (x,y) \in (X, 1-X)^2\}$$

$$Z = |\{p_i : p_i \notin B_X\}|.$$

B_X is the open square whose sides are parallel to, and a distance X away from, the sides of the unit square. Z is the number of points in $\overline{B_X}$, the hollowed out region surrounding B_X (Figure 2.3b). No matter what the placement of the p_i we always find that $\mathtt{R} \supseteq B_X$. Therefore, the number of points inside R, $n-M$, is always greater than the number of points inside B_X, $n-Z$: thus $M \leq Z$. The objective of this section is to show that there exists a $b > 0$ such that $E(Z) \sim b\sqrt{n}$ and therefore $E(M) = O(\sqrt{n})$. Before doing this we prove the following weaker result.

Theorem 2.1: Let $X = \max(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. Then, for n points chosen I.I.D. $\mathbf{U}[0,1]^2$,

$$\mathrm{E}(X) \sim \frac{c}{\sqrt{n}} \tag{2.1}$$

where $c = \sqrt{\pi}\left[\frac{7\sqrt{2}}{4} + 2\sqrt{\frac{2}{3}} - 3\right] = 1.9636\ldots$

Proof: We split the proof into four parts. In the first we derive a truncated integral representation for $\mathrm{E}(X)$. In the second we derive a closed form for the integrand. In the third we use the Gamma and Beta function techniques introduced in §1.4 to evaluate the general term of this closed form. In the fourth we combine the first three parts to prove the theorem.

(i) The values of the α_k range between 0 (a point at the k-th corner) and 2 (all of the points are clustered at the corner diagonally across from the k-th one). Thus $0 \leq X \leq 2$ and we can write

$$E(X) = \int_0^2 \alpha f_X(\alpha)\, d\alpha = \int_0^2 \Pr(X \geq \alpha)\, d\alpha. \tag{2.2}$$

where $f_X(\alpha)$ is the probability density function of X. Now $X \geq \alpha$ if and only if $\exists k$ such that $\alpha_k \geq \alpha$. This in turn is true if and only if at least one of the four isosceles right triangles $A_k(\alpha)$ is empty, thus

$$X \geq \alpha \Leftrightarrow \bigvee_i \{A_k(\alpha) \text{ is empty}\} \tag{2.3}$$

($\bigvee$ indicates the union of events). As occurred so many times in §1.4 we again have to concern ourselves with whether the region we are examining is totally contained within the support of our probability measure, i.e. whether $A_k(\alpha) \subseteq [0,1]^2$.

If $\alpha \leq 1$ then $A_k(\alpha) \subseteq [0,1]^2$ and, for any point p_i, we find that

$$\Pr(p_i \notin A_k(\alpha)) = 1 - \mathrm{Area}(A_k(\alpha)) = 1 - \alpha^2/2.$$

The p_i are independently distributed and therefore we can calculate

$$\begin{aligned}\Pr(A_k(\alpha) \text{ is empty}) &= \Pr(\forall i : p_i \notin A_k(\alpha)) \\ &= \prod_j \Pr(p_j \notin A_k(\alpha)) \\ &= (1-\alpha^2/2)^n.\end{aligned} \tag{2.4}$$

We now have to evaluate $\Pr(\bigvee_k \{A_k(\alpha) \text{ is empty}\})$. We do this by using the inclusion-exclusion principle **[GS2, p.6]** which requires taking the probability of the intersection of all possible combinations of one, two, three and four of the events $\{A_k(\alpha) \text{ is empty}\}$. If the $A_k(\alpha)$ are pairwise disjoint then this is very simple. For example

$$\{A_1(\alpha) \text{ is empty}\} \wedge \{A_2(\alpha) \text{ is empty}\} \Leftrightarrow \{A_1(\alpha) \cup A_2(\alpha) \text{ is empty}\} \tag{2.5}$$

and we can find the probability of the right side of (2.5) by using the same type of area argument that we used to derive (2.4): it is $(1-2\alpha^2/2)^n$. Notice that if $A_1(\alpha)$ and $A_2(\alpha)$ intersect then we have to subtract the area of their overlap greatly complicating our equations. This is especially complicated for the intersections of three or four events. Essentially this is the same difficulty that we had in §1.4(ii.b) with self overlapping neighborhoods. To avoid dealing with this difficulty we'll use the same technique we used there: showing that the bad event, here the $A_k(\alpha_k)$ not being pairwise disjoint, has an asymptotically negligent probability.

The $A_k(\alpha)$ are pairwise disjoint if and only if $\alpha \le \frac{1}{2}$ so we must prove that $\Pr(X \ge \alpha)$ is small for $\alpha > 1/2$. But by (2.3) and (2.4)

$$\begin{aligned}\Pr(X \ge \alpha) &\le \sum_k \Pr\left(A_k(\alpha) \text{ is empty}\right) \\ &= 4(1-\alpha^2/2)^n\end{aligned} \tag{2.6}$$

In the special case that $1/2 \le \alpha \le 2$ this shows that $\Pr(X \ge \alpha) \le 4 \cdot (7/8)^n$. We have really proven that this probability is asymptotically negligible for a much wider range of X. Suppose that $\ln n/\sqrt{n} \le \alpha \le 2$. Then

$$\Pr(X \ge \alpha) \le 4 \cdot \left(1 - \frac{\ln^2 n}{2n}\right)^n = O(n^{-\frac{1}{2}\ln n}) \tag{2.7}$$

and therefore we can restrict the range of (2.2) to yield

$$\mathrm{E}(X) = \int_0^{\frac{\ln n}{\sqrt{n}}} \Pr(X \ge \alpha)\, d\alpha + O(n^{-\frac{1}{2}\ln n}). \tag{2.8}$$

(ii) We have just shown that the only interesting situations are those where $\alpha < \ln n/\sqrt{n}$. assume that $n \geq 75$ and thus $\ln n/\sqrt{n} \leq 1/2$ and the $A_k(\alpha)$ are pairwise disjoint regions. We now apply the inclusion exclusion principle to (2.3);

$$\begin{aligned}\Pr(X \geq \alpha) = &\sum_k \Pr(A_k(\alpha) \text{ is empty}) - \sum_{k_1<k_2} \Pr(A_{k_1}(\alpha) \cup A_{k_2}(\alpha) \text{ is empty}) \\ &+ \sum_{k_1<k_2<k_3} \Pr(A_{k_1}(\alpha) \cup A_{k_2}(\alpha) \cup A_{k_3}(\alpha) \text{ is empty}) \\ &- \Pr(A_1(\alpha) \cup A_2(\alpha) \cup A_3(\alpha) \cup A_4(\alpha) \text{ is empty}) \\ = &\, 4(1-\alpha^2/2)^n - 6(1-2\alpha^2/2)^n + 4(1-3\alpha^2/2)^n - (1-4\alpha^2/2)^n. \end{aligned} \tag{2.9}$$

(iii) Plugging (2.9) into (2.8) reveals that the calculation of $\mathrm{E}(Z)$ involves the evaluation of four integrals of the general form

$$\int_0^{\frac{\ln n}{\sqrt{n}}} (1-k\alpha^2)^n d\alpha. \tag{2.10}$$

These can be evaluated by transforming them into Beta functions. The first step in this transformation is to change the limits of integration. For $\alpha \geq \ln n/\sqrt{n}$ we see that $(1-k\alpha^2)^n = O(n^{-k\ln n})$. Therefore we can replace (2.10) with

$$\int_0^{k^{-\frac{1}{2}}} (1-k\alpha^2)^n \, d\alpha + O(n^{-k\ln n}). \tag{2.11}$$

Setting $u = k\alpha^2$ gives

$$\begin{aligned}\int_0^{k^{-\frac{1}{2}}} (1-k\alpha^2)^n \, d\alpha &= \frac{1}{2\sqrt{k}} \int_0^1 u^{-1/2}(1-u)^n \, du \\ &= \frac{1}{2\sqrt{k}} \beta(\frac{1}{2}, n+1)\end{aligned} \tag{2.12}$$

which by Lemma 1.3 and the fact that $\Gamma(1/2) = \sqrt{\pi}$ evaluates out to

$$\frac{1}{2}\sqrt{\frac{\pi}{k}} n^{-\frac{1}{2}} + O(n^{-3/2}).$$

Our general formula is then

$$\int_0^{\frac{\ln n}{\sqrt{n}}} (1-k\alpha^2)^n \, d\alpha \sim \frac{1}{2}\sqrt{\frac{\pi}{k}} n^{-1/2} + O(n^{-3/2}). \tag{2.13}$$

(iv) We now prove the theorem.

$$\begin{aligned}
\mathrm{E}(X) &= \int_0^{\frac{\ln n}{\sqrt{n}}} \Pr(X \geq \alpha)\, d\alpha + O(n^{-\frac{1}{2}\ln n}) \\
&= \int_0^{\frac{\ln n}{\sqrt{n}}} \Big[4\left(1 - \frac{\alpha^2}{2}\right)^n - 6(1-\alpha^2)^n \\
&= \qquad\qquad + 4\left(1 - \frac{3\alpha^2}{2}\right)^n - \left(1 - 2\alpha^2\right)^n \Big] d\alpha + O(n^{-3/2}) \qquad (2.14) \\
&= \frac{\sqrt{\pi}}{2}\left[4\sqrt{2} - 6 + 4\sqrt{\frac{2}{3}} - \sqrt{\frac{2}{4}} \right] n^{-1/2} + O(n^{-3/2}) \\
&= cn^{-1/2} + O(n^{-3/2})
\end{aligned}$$

where we evaluate $c = 1.9636\ldots$. **Q.E.D.**

We now understand the behavior of $\mathrm{E}(X)$ but we still don't know how well it approximates X. One way of approaching this question is to use the concept of scaling introduced in §1.4. From (2.9) we see that

$$\begin{aligned}
\Pr\left(X \geq \frac{c}{\sqrt{n}}\right) &= 4(1 - c^2/2n)^n - 6(1 - c^2/n)^n + 4(1 - 3c^2/2n)^n - (1 - 2c^2/n)^n \\
&= \left(4e^{-c^2/2} - 6e^{-c^2} + 4e^{-3c^2/2} - e^{-2c^2}\right)\left[1 + \left(\frac{1}{n}\right)\right].
\end{aligned} \qquad (2.15)$$

Therefore X is concentrated in a strip which can best be described as "a constant over $\sqrt{n}$". That is,

$$\text{if } f(n) \to 0 \qquad \text{then} \qquad \Pr\left(X \leq \frac{f(n)}{\sqrt{n}}\right) \to 0.$$

Similarly,

$$\text{if } f(n) \to \infty \qquad \text{then} \qquad \Pr\left(X \geq \frac{f(n)}{\sqrt{n}}\right) \to 0.$$

Another way of approaching the question is by comparing $\mathrm{Var}(X)$ to $\mathrm{E}(X)$. Starting with

$$\mathrm{E}(X^2) = \int_0^2 \alpha^2 f_X(\alpha)\, d\alpha$$

we integrate by parts and use the same technique that yielded (2.8) to find

$$\mathrm{E}(X^2) = 2\int_0^{\frac{\ln n}{\sqrt{n}}} \alpha \Pr(X \geq \alpha)\, d\alpha + O(n^{-\frac{1}{2}\ln n}). \qquad (2.16).$$

The general term of $\Pr(X \geq \alpha)$ is of the form $(1 - k\alpha^2)^n$ and thus evaluating (2.16) means integrating terms of the general form

$$\begin{aligned} \int_0^{\frac{\ln n}{\sqrt{n}}} \alpha(1 - k\alpha^2)^n \, d\alpha &= \int_0^{k^{-\frac{1}{2}}} \alpha(1 - k\alpha^2)^n \, d\alpha + O(n^{-k \ln n}) \\ &= \frac{1}{2k(n+1)} + O(n^{-k \ln n}). \end{aligned} \tag{2.17}$$

Substituting (2.9) into (2.16) and evaluating using (2.17) we find that

$$\begin{aligned} \mathrm{E}(X^2) &= \frac{2(4 - \frac{6}{2} + \frac{4}{3} - \frac{1}{4})}{n} + O\left(\frac{1}{n^2}\right) \\ &= \frac{25}{6n} + O\left(\frac{1}{n^2}\right). \end{aligned} \tag{2.18}$$

Since $\mathrm{Var}(X) = \mathrm{E}(X^2) - \mathrm{E}^2(X)$ we have just proven

Theorem 2.2: Let $X = \max(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and $c = \sqrt{\pi}\left[\frac{7\sqrt{2}}{4} + 2\sqrt{\frac{2}{3}} - 3\right] = 1.9636\ldots$. Then, for n points chosen I.I.D. $\mathbf{U}[0,1]^2$

$$\mathrm{Var}(X) \sim \frac{c'}{n} \tag{2.19}$$

where $c' = \frac{25}{6} - c^2 = 0.3109\ldots$

We have just shown that $\sqrt{\mathrm{Var}(X)}$ and $\mathrm{E}(X)$ are of the same order of magnitude. This tells us the same thing as the scaling argument, namely that X is concentrated in a strip whose values are "a constant over $\sqrt{n}$". We will use this fact to analyse $\mathrm{E}(Z)$. Theorem 2.1 tells us that $\mathrm{E}(X)$ is $O(n^{-1/2})$. By definition $\overline{B_X}$, the hollowed out region surrounding B_X, is composed of four strips each of average area $O(n^{-\frac{1}{2}})$ and so it also has an area of $O(n^{-1/2})$. Since the value of X is heavily concentrated around its mean (up to constant multiples) and there are n points this would lead us to guess that $\mathrm{E}(Z)$, the expected number of points in $\overline{B_X}$ (Figure 2.4), is $n \cdot O(n^{-1/2}) = O(n^{1/2})$. It happens that this is true; what follows is a rigorous proof of the fact.

Theorem 2.3: Let $Z = |\{p_i : p_i \notin B_X\}|$ and $c = \sqrt{\pi}\left[\frac{7\sqrt{2}}{4} + 2\sqrt{\frac{2}{3}} - 3\right] = 1.9636\ldots$ Then, for n points chosen I.I.D. $\mathbf{U}[0,1]^2$

$$\mathrm{E}(Z) \sim 4c\sqrt{n}. \tag{2.20}$$

Proof: We prove this in three steps. In the first step we express $\mathrm{E}(Z)$ as a truncated integral of the form $\int \mathrm{E}(Z|X=\alpha) f_X(\alpha)\, d\alpha$. In the second we derive an asymptotic expression for $\mathrm{E}(Z|X=\alpha)$ as a function of α. In the third we prove (2.20).

(i)

$$\mathrm{E}(Z) = \int_0^2 \mathrm{E}(Z|X=\alpha) f_X(\alpha)\, d\alpha.$$

The first step in the evaluation of this integral is the same as in Theorems 2.1 and 2.2; we restrict the integral to the region $\left[0, \frac{\ln n}{\sqrt{n}}\right]$. It is always true that $Z \le n$. Therefore

$$\begin{aligned}\int_{\frac{\ln n}{\sqrt{n}}}^2 \mathrm{E}(Z|X=\alpha) f_X(\alpha)\, d\alpha &\le n \int_{\frac{\ln n}{\sqrt{n}}}^1 f_X(\alpha)\, d\alpha \\ &= n \Pr\left(X \ge \frac{\ln n}{\sqrt{n}}\right) \\ &= O(n^{1-\frac{1}{2}\ln n}).\end{aligned} \tag{2.21}$$

so

$$\mathrm{E}(Z) = \int_0^{\frac{\ln n}{\sqrt{n}}} \mathrm{E}(Z|X=\alpha) f_X(\alpha)\, d\alpha + O(n^{1-\frac{1}{2}\ln n}). \tag{2.22}$$

(ii) Intuitively this step is simple. Mathematically it is the bottleneck since we have to deal with the problems introduced by conditionality.

$\Pr(X \ge \alpha)$ was analyzed during the proof of Theorem 1 where we obtained the explicit formula given by equation (2.9). Differentiating that formula gives us an explicit formula for $f_X(\alpha)$ because $f_X(\alpha) = \frac{d}{d\alpha}\Pr(X \ge \alpha)$. The only unknown left to evaluate in (2.22) is the conditional expectation $E(Z|X=\alpha)$. Remember that $X = \max(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, so finding $E(Z|X=\alpha)$ requires integrating $E(Z|\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ over $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ conditioned on $X = \alpha$. Formally

$$E(Z|X=\alpha) = \int E(Z|\alpha_1, \alpha_2, \alpha_3, \alpha_4) g_\alpha(\alpha_1, \alpha_2, \alpha_3, \alpha_4)\, d\alpha_1\, d\alpha_2\, d\alpha_3\, d\alpha_4 \tag{2.23}$$

where $g_\alpha(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ is the appropriate probability density function that conditions on $X = \alpha$.

Referring to Figure 2.4 we see that the analysis of (2.23) is much simpler than it at first appears. Originally the n points are uniformly distributed in $[0,1]^2$. Conditioning on $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ is equivalent to conditioning on the fact that the $A_k(\alpha_k)$ are empty and there are four points, the q_k, on their boundaries. Thus the remaining $n-4$ points are uniformly distributed in the remaining region $[0,1]^2 - \cup A_k(\alpha_k)$. This implies that each of the $n-4$ points has probability p of being in

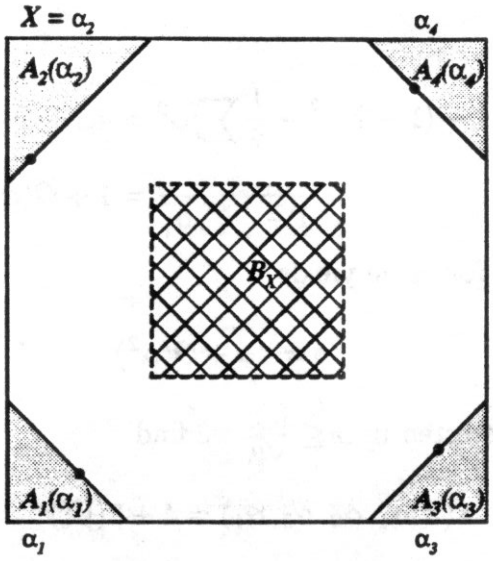


Figure 2.4. Fixing the α_k implies that all of the points (aside from the q_k) are independently uniformly distributed outside of $\cup_k A_k(\alpha_k)$-s (the grey areas). Thus the probability that a point is not in B_X (crosshatched region) is $p = \frac{\text{area of white region}}{\text{area of non gray region}} = \frac{1-Area(B_X)-Area(\cup_k A_k(\alpha_k))}{1-Area(\cup_k A_k(\alpha_k))}$.

$\overline{B_X}$ where p is the probability that a point will not be in B_X conditioned on the event that $\cup A_k(\alpha_k)$ is empty so

$$p = \frac{1 - Area(B_X) - Area(\cup_k A_k(\alpha_k))}{1 - Area(\cup_k A_k(\alpha_k))} = \frac{1 - (1-2\alpha)^2 - \frac{1}{2}\sum \alpha_i^2}{1 - \frac{1}{2}\sum \alpha_i^2}. \tag{2.24}$$

By the independence of the points, $Z|\alpha_1, \alpha_2, \alpha_3, \alpha_4$ is a random variable that has the same distribution as $4 + W$ where $W(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ is a $B(n-4, p)$ (Binomial) random variable[1]. All of the α_i are less than α because $\max(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = X = \alpha$ and thus, as $\alpha \to 0$

---

[1] It is not difficult to construct a formal mathematical proof out of this argument. First use limiting area arguments to calculate the probability density functions $f(Z = i, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and $f(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and then divide to find

$$f(Z = i|\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \frac{f(Z = i, \alpha_1, \alpha_2, \alpha_3, \alpha_4)}{f(\alpha_1, \alpha_2, \alpha_3, \alpha_4)} = \binom{n-4}{i-4} p^{i-4}(1-p)^{n-i}.$$

Thus $Z = 4 + W$.

$$1-(1-2\alpha)^2-\frac{1}{2}\sum\alpha_i^2=4\alpha+O(\alpha^2).$$

$$\frac{1}{1-\frac{1}{2}\sum\alpha_i^2}=1+O(\alpha^2).$$

Combining these two formulae yields

$$p=4\alpha+O(\alpha^2). \tag{2.25}$$

Since we are only interested in $\alpha \leq \frac{\ln n}{\sqrt{n}}$ we find

$$\begin{aligned}\mathrm{E}(Z|\alpha_1,\alpha_2,\alpha_3,\alpha_4) &= 4+\mathrm{E}(W)\\ &= 4+(n-4)p\\ &= 4n\alpha+O(\ln^2 n)\end{aligned}$$

Using this and the fact that $\int g_\alpha(\alpha_1,\alpha_2,\alpha_3,\alpha_4)\,d\alpha_1\,d\alpha_2\,d\alpha_3\,d\alpha_4 = 1$ we integrate (2.23) to get

$$\mathrm{E}(Z|X=\alpha)=4n\alpha+O(\ln^2 n). \tag{2.26}$$

(iii) Now we go back and evaluate (2.22).

$$\begin{aligned}\mathrm{E}(Z) &= \int_0^{\frac{\ln n}{\sqrt{n}}}\mathrm{E}(Z|X=\alpha)f_X(\alpha)\,d\alpha+O(n^{-\frac{1}{2}\ln n})\\ &= 4n\int_0^{\frac{\ln n}{\sqrt{n}}}\alpha f_X(\alpha)\,d\alpha+O\left(\ln^2 n\,\Pr\left(X\geq\frac{\ln n}{\sqrt{n}}\right)\right).\end{aligned} \tag{2.27}$$

Integrating by parts and applying (2.7) and (2.8) yields

$$\begin{aligned}\int_0^{\frac{\ln n}{\sqrt{n}}}\alpha f_X(\alpha)\,d\alpha &= -\alpha\Pr(X\geq\alpha)\Big|_0^{\frac{\ln n}{\sqrt{n}}}+\int_0^{\frac{\ln n}{\sqrt{n}}}\Pr(X\geq\alpha)\,d\alpha)\\ &= O(n^{-\frac{1}{2}\ln n})+\mathrm{E}(X)\end{aligned}$$

Combining this with (2.27) and the fact that $\Pr\left(X\geq\frac{\ln n}{\sqrt{n}}\right)=O(n^{-\frac{1}{2}\ln n})$ gives

$$\mathrm{E}(Z)=4cn^{1/2}+O(n^{-3/2}) \tag{2.28}$$

Q.E.D.

We close this section by calculating the variance and second moment of Z. Besides using this information to determine how closely $\mathrm{E}(Z)$ approximates Z we will also later use it to directly analyze certain convex hull procedures.

Theorem 2.4: Let $Z = |\{p_i : p_i \notin B_X\}|$ and $c' = \frac{25}{6} - c^2 = 0.3109\ldots$. Then, for n points chosen I.I.D. $\mathbf{U}[0,1]^2$

$$(a) \quad \mathrm{Var}(Z) \sim 16c'n \tag{2.29}$$

$$(b) \quad \mathrm{E}(Z^2) \sim \frac{25}{4}n. \tag{2.30}$$

Proof:

$$\mathrm{E}(Z^2) = \int_0^{\frac{\ln n}{\sqrt{n}}} \mathrm{E}(Z^2|X=\alpha) f_X(\alpha)\, d\alpha \; + \; O(n^{2-\frac{1}{2}\ln n}). \tag{2.31}$$

We again use the fact that $Z|\alpha_1,\alpha_2,\alpha_3,\alpha_4$ has the same distribution as $W+4$.

$$\begin{aligned} \mathrm{E}(Z^2|\alpha_1,\alpha_2,\alpha_3,\alpha_4) &= \mathrm{E}((W+4)^2) \\ &= \mathrm{E}(W(W-1)) + 9\mathrm{E}(W) + 16. \end{aligned} \tag{2.32}$$

Since W is a $B(n-4,p)$ random variable

$$\begin{aligned} \mathrm{E}(W(W-1)) &= (n-4)(n-5)p^2 \\ \mathrm{E}(W) &= (n-4)p. \end{aligned} \tag{2.33}$$

We know that $p = 4\alpha + O(\alpha^2)$ and $\alpha \le \frac{\ln n}{\sqrt{n}}$ so we can put (2.32) and (2.33) together to get

$$\mathrm{E}(Z^2|\alpha_1,\alpha_2,\alpha_3,\alpha_4) = 16n^2\alpha^2 + 36n\alpha + O(\ln^2 n).$$

Integrating this over $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ conditioned on $X=\alpha$ we get

$$\mathrm{E}(Z^2|\alpha) = 16n^2\alpha^2 + 36n\alpha + O(\ln^2 n). \tag{2.34}$$

Inserting this into (2.31) we find

$$\mathrm{E}(Z^2) = 16n^2 \int_0^{\frac{\ln n}{\sqrt{n}}} \alpha^2 f_X(\alpha)\, d\alpha \; + \; 9n \int_0^{\frac{\ln n}{\sqrt{n}}} \alpha f_X(\alpha)\, d\alpha \; + \; O(\ln^2 n).$$

These are integrals that we have previously calculated (to derive (2.18) and (2.1)) and thus we can write

$$\mathrm{E}(Z^2) = \frac{400}{3}n + O(\sqrt{n}).$$

We have proven (b); (a) will follow from

$$\mathrm{Var}(Z) = \mathrm{E}(Z^2) - \mathrm{E}^2(Z) = 16(\frac{25}{6}n - c^2 n) = 16c'n.$$

Q.E.D.

§2.4 Uses and Simulation Results

Eliminate has been used as a preprocessing step preceeding a standard convex hull algorithm. In the previous section we calculated how many points would be left untouched after its run. The question now is "How fast we can calculate a convex hull?" The answer obviously depends on what convex hull algorithm follows *eliminate*. Formally, let $\mathcal{A}$ be a convex hull algorithm. We would like to know how fast the *eliminate*/$\mathcal{A}$ pair runs. We also would like to know whether running the preprocessing heuristic is worthwhile: how fast is the *eliminate*/$\mathcal{A}$ pair as compared to $\mathcal{A}$ alone? The answer obviously depends on the choice of $\mathcal{A}$. In the paragraphs that follow we'll answer these questions for two standard convex hull algorithms: gift-wrapping and Graham's scan. (For full descriptions of these algorithms and derivations of their stand-alone worst-case and expected running times see [**PS**].) In this section we continue to assume that the n points are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$.

Theorem 2.5: The combined *eliminate*/gift-wrapping pair has an expected $O(n)$ running time.

Proof: The *eliminate*/gift-wrap pair has two stages. The *eliminate* stage is $O(n)$ worst case and discards all but M points. When run on x points gift-wrapping has an $O(x^2)$ worst case running time. The gift-wrapping stage will therefore require at most $O(M^2) \leq O(Z^2)$ where M is the number of points left over by *eliminate* and Z is the number of points in $\overline{B}_X$. The combined expected running time of the pair of algorithms will be $O(n) + O(\mathrm{E}(Z^2))$ which, by Theorem 2.4(b), is $O(n)$.

Q.E.D.

This result is much better than the expected $O(n \log n)$ running time for the stand-alone gift-wrapping algorithm with n points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$. It even compares favorably with more sophisticated algorithms such as Quickhull which was described in §1.4. Quickhull also has an $O(n^2)$ worst case and $O(n)$ expected and running time [**OL**] but has to employ a complicated nested recursion with high overhead and constants to achieve this. This is only to be expected. *Eliminate* is very much like the first step of Quickhull (eliminating some points) but we are are tailoring our choice of points to a particular distribution so one elimination step suffices and the algorithm is much simpler.

We can extend this idea further. Quickhull runs in expected linear time over a number of distributions, i.e. those where the points are uniformly distributed in

a convex bounded polygon[2]. If this distribution is known in advance then we can modify *eliminate* to work well: find the points that are "closest" to the corners of the convex polygon (where closeness of a point is measured by the length of the projection of the line between it and the corner on the perpendicular bisector of the angle at that corner) and eliminate all of the points inside the convex hull of the "close" points. It is not difficult to modify our analysis of *eliminate* to show that only $O(\sqrt{n})$ points will be left on average. It must be stressed though that the corresponding algorithm will not be as simple as *eliminate* because it will have to test for arbitrary polygon inclusion.

Graham's scan is another classic convex hull algorithm. First it sorts all of its input points radially (by angle) around some point internal to the hull. Next it does a linear scan through the points that decides which points are on the hull. The first stage is asymptotically dominant and will take $O(n \log n)$ time when run on n points. We can now use Theorem 2.1 to prove

Theorem 2.6: The combined *eliminate*/Graham's-scan pair has an expected $O(n)$ running time. Furthermore *eliminate*'s running time strongly dominates (in an average sense) that of Graham's scan.

Proof: As in the proof of Theorem 2.5 the *eliminate* stage discards all but $M \leq Z$ points. When run on m points Graham's scan has an $O(m \log_2 m)$ worst case running time. Thus the expected running time of the Graham's scan is upper bounded by

$$\mathrm{E}(Z \log_2 Z) \leq \mathrm{E}(Z) \log_2 n \sim 4c\sqrt{n} \log_2 n$$

which is dominated by *eliminate*'s $O(n)$ running time.

Q.E.D.

This tells us that the *eliminate*/Graham's-scan pair has an $O(n \log n)$ worst case time and $O(n)$ expected time, the best possible results that can be achieved by any convex hull algorithm. This is an extremely attractive result from a computational point of view especially because these bounds were achieved through an exceptoinally simple process; the bulk of the running time is devoted to *eliminate*'s two straightfoward linear passes through the data.

In Table 2.1 we present the results of test runs on random points. Each row

[2] Quickhull also runs in expected linear time even if the convex region is not a polygon, e.g. a circle. Our heuristic is not linear in such cases.

n	$\sqrt{n}\,\overline{X}_n$	$\overline{Z}_n/\sqrt{n}$	$\overline{M}_n/\sqrt{n}$
1000	1.874	7.025	3.410
2000	2.042	7.759	3.699
3000	1.996	7.714	3.645
4000	2.033	7.839	3.765
5000	1.849	7.235	3.386
6000	1.983	7.693	3.669
7000	1.966	7.675	3.558
8000	2.009	7.863	3.710
9000	1.966	7.696	3.645
10000	1.965	7.698	3.435

Table 2.1. For each value of n we chose 100 sets of n random points. For each of these sets we found X, Z, and M and then averaged over all the sets to calculate $\bar{X}_n$, $\bar{Z}_n$, and $\bar{M}_n$. Normalizing (multiplying/dividing by $\sqrt{n}$) yielded the values in the table.

contains the values $\sqrt{n}\,\overline{X}_n$ [3], $\overline{Z}_n/\sqrt{n}$, and $\overline{M}_n/\sqrt{n}$. These are the normalized values of the respective random variables averaged over 100 sets of n random points (n fixed). Theorems 2.1 and 2.3 predict that for large enough n

$$\sqrt{n}\,\mathrm{E}(X_n) \approx 1.964 \qquad \text{and} \qquad \mathrm{E}(Z_n)/\sqrt{n} \approx 7.854$$

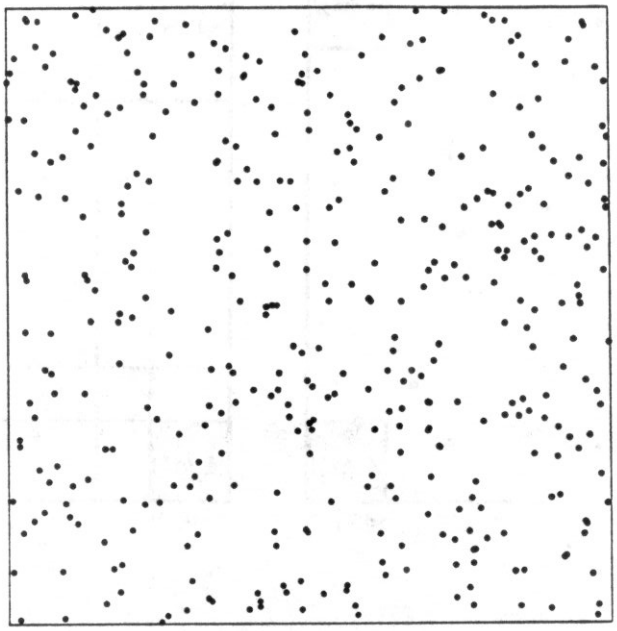
and we do see this behavior.

To close this section we present a medium-size example that illustrates *eliminate*'s utilility. We randomly picked the four hundred points in Figure 2.5a. After running *eliminate* only the 52 points of Figure 2.5b remain. The dotted line in 5b is the convex hull of the points (16 points). It is clear that running *eliminate* tremendously reduces the amount of time that the subsequent convex hull routine will require.

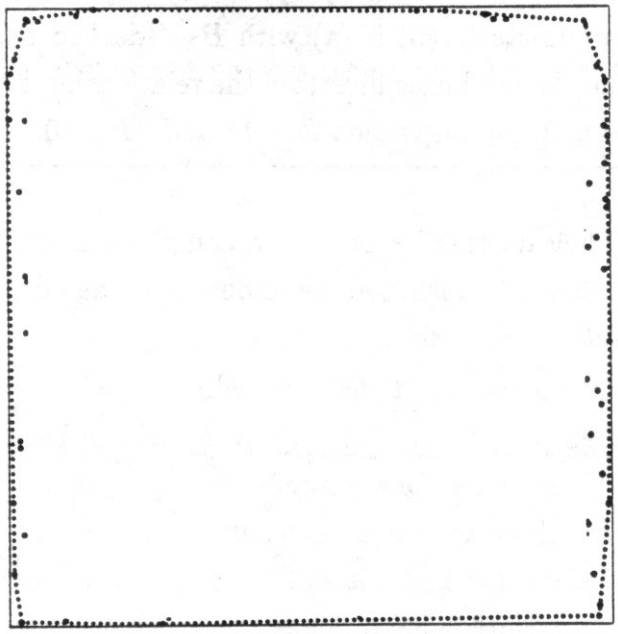
§2.5 A Variant

In this section we will try to illustrate one of the major advantages of tight algorithmic analyses: they allow us to fine-tune our algorithms to provide better

[3] Until now we only implicitly noted the fact that $E(X)$, $E(Z)$, and $E(M)$ are functions of n. We now make the dependency explicit and write X_n, Z_n, and M_n.



(a)



(b)

Figure 2.5: (a) shows 400 randomly chosen points in the unit square. Running *eliminate* erases all but $S = 52 = 2.6\sqrt{400}$ of the points as illustrated in (b). The convex hull (16 points) is represented by the dotted line.

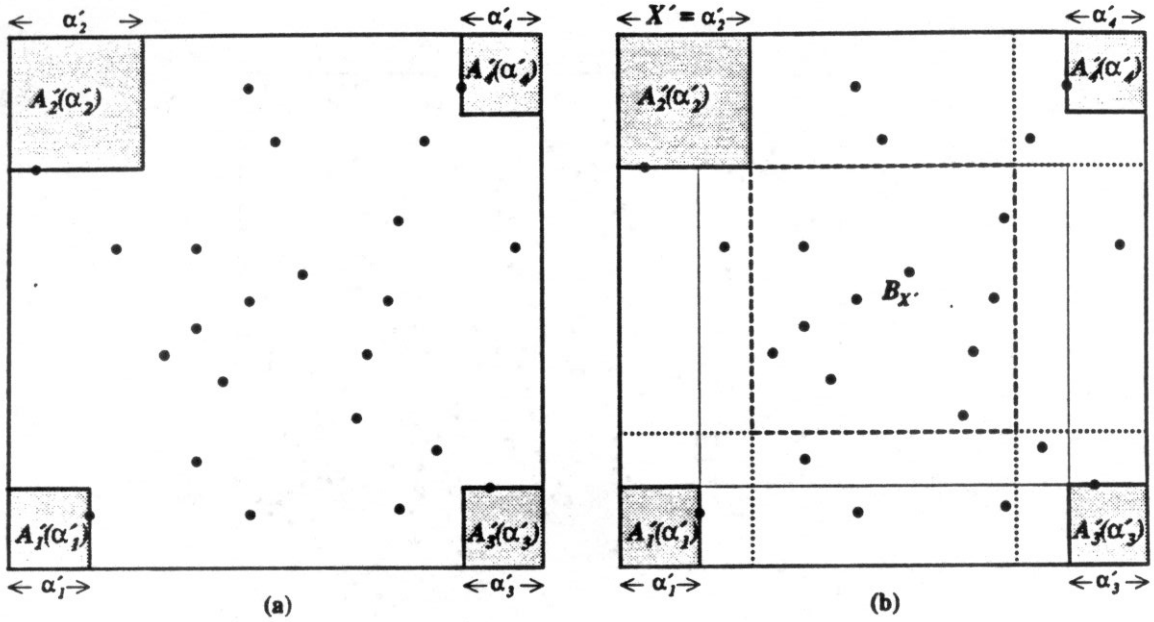


Figure 2.6. (6a) gives some test points with the α'_k noted and the $A'_k(\alpha'_k)$ (shaded areas) drawn. Note especially that $A'_k(\alpha'_k)$ are all empty with a point (q_k) on each of their defining perimeters. (b) is (a) with $B_{X'}$ (dashed box) and R (thin lined box) drawn in. The dotted boxes illustrate the relationship between the α'_k and X'. By definition $B_{X'} \subseteq$ R. By inspection $Z = 17$ and $M = 10$.

running times. They do this by providing complete information about the algorithm's behavior. This, in turn, lets us understand the effect of minor changes in the algorithm on its running time.

We will examine a variant of *eliminate* which we will name *eliminate2*. At first *elim2* will look even better than the original algorithm because we will be able to calculate an even lower bound for the number of points that it leaves untouched than we were able to for *eliminate*. But, as we shall see, *elim2* performs slightly more comparisons than *eliminate* and this will be enough to offset *elim2*'s advantage of a smaller output. We are only able to make the above claims because our analyses of the two algorithms are tight and provide the multiplicative constants of the leading order terms.

Our new algorithm is very simple. In the original version of *eliminate* we swept in 45 degree diagonals from each of the four corners until each hit a point. *elim2* will sweep in squares from each of the four corners until each one hits a point. This is most easily understood via a diagram such as Figure 2.6. Program 2.2 provides

```
function elim2(N : integer) : integer;
    var     i1, i2, i3, i4, j, M : integer;
            a1, a2, a3, a4, highx, lowx, highy, lowy : real;
            t : point;
begin
    i1:=1   i2:=1;  i3:=1;  i4:= 1;
    a1:=MAX(p[1].x,p[1].y);          a4:=MAX(1-p[1].x,1-p[1].y);
    a2:=MAX(p[1].x,1-p[1].y);        a3:=MAX(1-p[1].x,p[1].y);

    for j:=2 to N do
        begin
        if MAX(p[j].x,p[j].y) < a1 then
            begin i1:=j; a1:=MAX(p[j].x,p[j].y)        end;
        if MAX(1-p[j].x ,1-p [j].y) > a4 then
            begin i4:=j; a4:=MAX(1-p[j].x,1-p[j].y)    end;
        if MAX(p[j].x, 1-p[j].y) < a2 then
            begin i2:=j; a2:=MAX(p[j].x,1-p[j].y )     end;
         if MAX(1-p[j].x,p[j].y) > a3 then
            begin i3:=j; a3:=MAX(1-p[j].x,p[j].y )     end;
        end;

    lowx:= MAX(p[i1].x, p[i2].x);  highx:= MIN(p[i3].x, p[i4].x);
    lowy:= MAX(p[i1].y, p[i3].y);  highy:= MIN(p[i2].y, p[i4].y);
    M:=0;  j:=N;
    while j>M do
        if  lowx < p[j].x  and  p[j].x < highx
          and  lowy < p[j].y  and  p[j].y < highy then
            j:=j-1
        else
            begin
                M := M+1;
                t := p[M];   p[M] := p[j];     p[j] := t;
            end;
    elim2 := M;
end;
```

Program 2.2. A full Pascal implementation of *elim2*. It is called with parameter N; the array $p[1 \ldots N]$ holds the points. After the function is initialized the **for** loop finds the four points that minimize the appropriate functions. *i1, i2, i3, i4* hold the indices of these four points etc. while *a1, a2, a3, a4* hold the respective minimized values of thr functions.The next step calculates the boundaries of R: *lowx, highx, lowy, highy*. The **while** loop rearranges the array so that the non-eliminated points are at its bottom. When it returns the function has value M, the number of points that weren't eliminated.

a Pascal encoding. The analysis follows that of §2.3 almost step by step. First we make the appropriate definitions:

$$\alpha'_1 = \min_{1\le i\le n} \max(x_i, y_i) \qquad A'_1(\alpha) = \{(x,y):\ 0 \le x, y \le \alpha\}$$
$$\alpha'_2 = \min_{1\le i\le n} \max(x_i, 1-y_i) \qquad A'_2(\alpha) = \{(x,y):\ 0 \le x, 1-y \le \alpha\}$$
$$\alpha'_3 = \min_{1\le i\le n} \max(1-x_i, y_i) \qquad A'_3(\alpha) = \{(x,y):\ 0 \le 1-x, y \le \alpha\}$$
$$\alpha'_4 = \min_{1\le i\le n} \max(1-x_i, 1-y_i) \qquad A'_4(\alpha) = \{(x,y):\ 0 \le 1-x, 1-y \le \alpha\}.$$

These quantities are analogous to the α_k and $A_k(\alpha)$ that were defined in §2.3. The α'_k are the lengths of the base sides of the largest empty squares that fit into the k'th corner; the $A'_k(\alpha)$ are squares with base side α. Together they have the property that $A'_k(\alpha'_k)$ is empty and has at least one of the p_i (from the original point set) on its perimeter. We also define

$$X' = \max(\alpha'_1, \alpha'_2, \alpha'_3, \alpha'_4)$$
$$B_{X'} = \{(x,y) : (x,y) \in (X', 1-X')^2\}$$
$$Z' = |\{p_i : p_i \notin B'_{X'}\}|.$$

First we provide some intuition as to why we would guess $Z' < Z$. Since

$$\text{Area}(A'_k(\alpha)) > \text{Area}(A_k(\alpha))$$

it follows that

$$\Pr(A'_k(\alpha) \text{ is empty}) < \Pr(A_k(\alpha) \text{ is empty})$$

and therefore we expect $\alpha'_k < \alpha_k$ and $X' > X$. This would lead us to guess that $B_{X'} \supset B_X$ and $Z' < Z$. We will be able to prove that this is true and even more, that

$$\mathrm{E}(Z) \sim \sqrt{2}\,\mathrm{E}(Z').$$

The steps leading to the calculation of $\mathrm{E}(Z')$ parallel those that led to to that of $\mathrm{E}(Z)$. We will only fill in the high points. We start with the calculation of $\mathrm{E}(X')$. First notice that (2.4) can be replaced by

$$\Pr(A'_k(\alpha) \text{ is empty}) = (1-\alpha^2)^n. \tag{2.35}$$

This means that we can replace (2.9) by

$$\begin{aligned}\Pr(X' \geq \alpha) = &\sum_k \Pr(A'_k(\alpha) \text{ is empty}) - \sum_{k_1<k_2} \Pr(A'_{k_1}(\alpha) \cup A'_{k_2}(\alpha) \text{ is empty}) \\ &+ \sum_{k_1<k_2<k_3} \Pr(A'_{k_1}(\alpha) \cup A'_{k_2}(\alpha) \cup A'_{k_3}(\alpha) \text{ is empty}) \\ &- \Pr(A'_1(\alpha) \cup A'_2(\alpha) \cup A'_3(\alpha) \cup A'_4(\alpha) \text{ is empty}) \\ = &\, 4(1-\alpha^2)^n - 6(1-2\alpha^2)^n + 4(1-3\alpha^2)^n - (1-4\alpha^2)^n. \end{aligned} \tag{2.36}$$

All of the integrals in §2.3 that were of the form $\int (1-k\alpha^2)^n \, d\alpha$ will be replaced by integrals of the form $\int (1-2k\alpha^2)^n \, d\alpha$. Since

$$\int (1-2k\alpha^2)^n \, d\alpha = \frac{1}{\sqrt{2}} \int (1-2k\alpha^2)^n \, d\alpha$$

we can finish the proof to show

Theorem 2.1′: Given n points $\mathbf{U}[0,1]^2$ and X defined as above then

$$\mathrm{E}(X') \sim \frac{c}{\sqrt{2n}}. \tag{2.37}$$

where $c = \sqrt{\pi}\left[\frac{7\sqrt{2}}{4} + 2\sqrt{\frac{2}{3}} - 3\right] = 1.9636\ldots$.

We can then prove

Theorem 2.3′: Given n points $\mathbf{U}[0,1]^2$ and Z' as defined above, then

$$\mathrm{E}(Z') \sim 4c\sqrt{2n}.$$

Proof: The same as for Theorem 2.3. The only major difference occurs in the equation that replaces (2.24):

$$p = \frac{1 - Area(B'_{X'}) - Area(\cup_k A'_k(\alpha'_k))}{1 - Area(\cup_k A'_k(\alpha'_k))} = \frac{1-(1-2\alpha)^2 - \sum \alpha_i'^2}{1 - \sum \alpha_i'^2}. \tag{2.38}$$

This does not require any revision in the proof because the only use of (2.24) was to prove

$$p = 4\alpha + O(\alpha^2) \tag{2.25}.$$

This last equation can be derived from (2.38) so we are done.

We have just shown that, on average, Z' is definitely smaller than Z. Therefore it might seem that *elim2* is a better algorithm to implement than *eliminate*. We shall now see that this is not true. We compare the two programs by examining

the number of operations each performs within its **for** loop. First we examine the number of **if-then** statements: Program 2.1 executes two **if-then** statements each time it enters the **for** loop while Program 2.2 executes four (although through clever coding we might be able to reduce this to two statements). Next, we compare the number of additions: each program performs two. Program 2.1 calculates $p[j].x + p[j].y$ and $p[j].x - p[j].y$ while Program 2.2 calculates $1 - p[j].x$ and $1 - p[j].y$. Program 2.1 performs no other operations. Program 2.2 still has to find two **MIN**s and two **MAX**s (although clever coding might reduce them to one of each type). Therefore Program 2.2 performs substantially more work than Program 2.1 when run on large inputs.

Our mathematical analysis showed that *elim2* leaves fewer points than *eliminate*. We have just shown that *elim2* takes more time than *eliminate*. Therefore, the question of which algorithm is better reduces to the question of whether *elim2*'s advantage of producing a smaller Z' offsets the extra work it has to do. If we are going to feed the output of the preprocessing algorithm into an $n \log n$ convex hull finding algorithm such as Graham's-scan then *eliminate* should be used because, as proven in Theorem 2.6, the amount of work performed by the preprocessing algorithm overwhelms that performed by the convex hull algorithm.

§2.6 Extensions to Higher Dimensions

§2.6.1 The Algorithm

Until now we have been discussing convex hulls in $\Re^2$. We can easily extend *eliminate* to higher dimensional spaces. First suppose we are given a set of points $S = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$ in d-space, ie.

$$p_i = (p_i(1), \ldots, p_i(d)), \qquad i = 1, \ldots, n.$$

The convex hull of S is

$$CH(S) = \text{ boundary of the smallest convex polytope containing } S.$$

Our plan is the same as it was in two dimensions: to identify some polytope contained in $CH(S)$ and elimimate the points in that polytope.

The algorithm we present is a natural generalization of the two dimensional one. First we sweep in diagonal planes from each vertex of an "infinite" hypercube

until they each hit some point in S. We find a large rectangular box contained in the convex polytope which has the hit points as vertices. We then eliminate all of the points in that box. The two dimensional analysis of §2.3 can also be generalized. It will show that only $O(n^{(d-1)/2})$ points will, on average, remain after running the algorithm.

Before proceeding with the presentation of the algorithm and its analysis some words of both apology and caution. Conceptually, what follows is extremely simple. Mathematically and notationally, it is tedious and cumbersome. Unfortunately this is standard procedure for generalized multidimensional discussions.

In two dimensions we found the four points that minimized the functions $\pm x \pm y$. In d-dimensions we find the 2^d points that minimize the functions $\pm p(1) \pm p(2) \ldots \pm p(d)$. Let c_k be the 2^d vectors in $\{1,-1\}^d$ in the standard binary order.

$$\begin{aligned} c_1 &= (1,1,\ldots,1,1) \\ c_2 &= (1,1,\ldots,1,-1) \\ c_3 &= (1,1,\ldots,-1,1) \\ c_4 &= (1,1,\ldots,-1,-1) \\ &\vdots \\ c_{2^d} &= (-1,-1,\ldots,-1,-1). \end{aligned}$$

The 2^d functions $\pm p(1) \pm p(2) \ldots \pm p(d)$ can be written as the inner products $c_k \cdot p^T \quad k = 1,\ldots,2^d$ and the points that minimize them[4] will be

$$q_k = p_m : \quad m = \min_i \{\forall j : c_k \cdot p_i^T \leq c_k \cdot p_j^T\} \qquad k = 1,\ldots,2^d. \tag{2.39}$$

Let Q be the convex hull of $q_1,\ldots,q_{2^d}$. As before, we can eliminate all points inside Q without discarding any information necessary for the construction of the convex hull of $p_1,\ldots,p_n$. Unfortunately the calculation (and storage) of Q is complicated. Furthermore the elimination of points inside Q is even more prone to round off error now than it was in two dimensions. We sidestep these difficulties by again finding a large rectangular prism (in 2-d it was a rectangle) $\mathtt{R} \subseteq \mathtt{Q}$ and only eliminating the points inside R .

[4] In case of ties we again pick the point with lowest index that minimizes the inner product.

How will we find a good R? Recall that in two dimensions R was of the form

$$\mathtt{R} = \{(x,y):\ a(1) < x < b(1),\ a(2) < y < b(2)\}$$

where $a(1)$, $b(1)$, $a(2)$, and $b(2)$ all had nice geometrical definitions: $a(1)$ was the *max* of the two points hit by the two lines ($x \pm y = const$) in the increasing x direction sweeping in from the left while $b(1)$ was the *min* of the two points hit by the two lines ($-x \pm y = const$) sweeping in from the right. $a(2)$ and $b(2)$ were defined similarly for the y coordinate. This idea can be extended to d dimensions. R will be of the form

$$\mathtt{R} = \{(x(1),\ldots,x(d)):\ a(i) < x(i) < b(i),\quad i = 1,\ldots,d\}$$

where $a(i)$ will be $\max_k\{q_k(i)\}$, q_k running over those points that are hit by a sweep hyperplane ($c_k \cdot x^T = const$) that approaches in an increasing $x(i)$ direction. Similarly $b(i)$ will be $\min_k\{q_k(i)\}$, q_k running over those points hit by a sweep hyperplane that approaches in a decreasing $x(i)$ direction. The planes entering in an increasing direction are those whose defining equations have a $+1$ as their i-th coefficient while those entering in a decreasing direction have a -1 there. If we define the index sets

$$J_i^+ = \{k : c_k(i) = +1\},\quad J_i^- = \{k : c_k(i) = -1\},\qquad i = 1,\ldots,d \tag{2.40}$$

then we can write

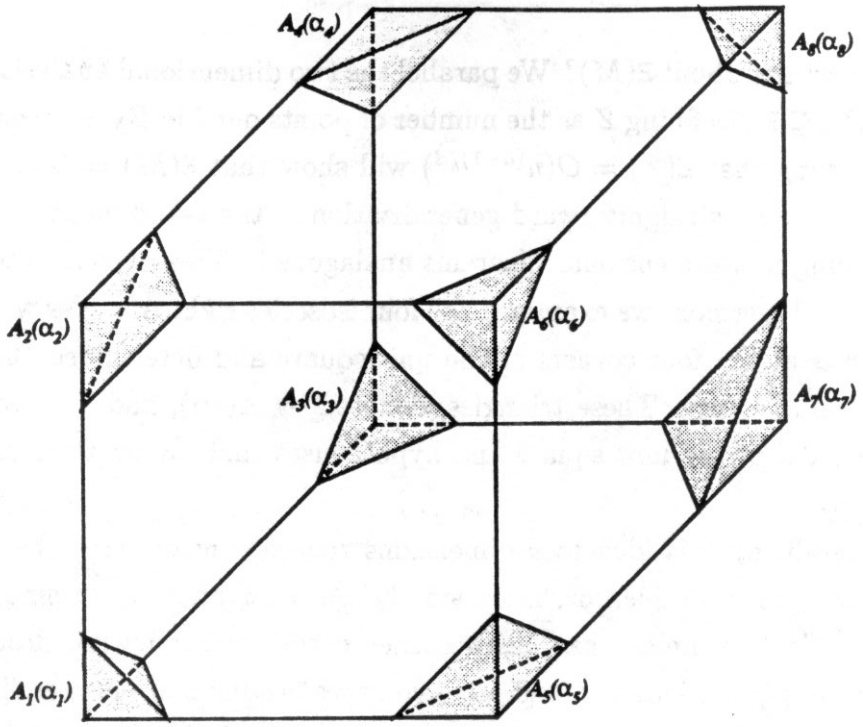
$$\mathtt{R} = \{(x(1),\ldots,x(d)):\ \max_{k\in J_i^+} q_k(i) < x(i) < \min_{k\in J_i^-} q_k(i),\quad i = 1,\ldots,d\}. \tag{2.41}$$

Calculating R and eliminating the points inside it is an $O(n)$ time operation. Figure 2.7 describes the three-dimensional version of this algorithm and shows that while the notation is somewhat dense the concepts behind each step are intuitively clear and the full algorithm is relatively easy to program. This is especially true in the lower dimensional cases ($d = 2,3$) where convex hull problems are frequently found.

§2.6.2 The Analysis

Suppose that $p_1, p_2, \ldots, p_n$ are chosen I.I.D. $\mathbf{U}[0,1]^{\mathbf{d}}$. What can we predict about the performance of the modified algorithm? More specifically, if we define

$$\begin{aligned} M(p_1,\ldots,p_n) &= \textit{The number of points undeleted after running eliminate.} \\ &= \textit{The number of points outside } \mathtt{R} \end{aligned}$$



(a)

$$
\begin{aligned}
q_1 = p_m \quad & m = \min_i\{\forall j : +p_i(1) + p_i(2) + p_i(3) \le +p_j(1) + p_j(2) + p_j(3)\} \\
q_2 = p_m \quad & m = \min_i\{\forall j : +p_i(1) + p_i(2) - p_i(3) \le +p_j(1) + p_j(2) - p_j(3)\} \\
q_3 = p_m \quad & m = \min_i\{\forall j : +p_i(1) - p_i(2) + p_i(3) \le +p_j(1) - p_j(2) + p_j(3)\} \\
q_4 = p_m \quad & m = \min_i\{\forall j : +p_i(1) - p_i(2) - p_i(3) \le +p_j(1) - p_j(2) - p_j(3)\} \\
q_5 = p_m \quad & m = \min_i\{\forall j : -p_i(1) + p_i(2) + p_i(3) \le -p_j(1) + p_j(2) + p_j(3)\} \\
q_6 = p_m \quad & m = \min_i\{\forall j : -p_i(1) + p_i(2) - p_i(3) \le -p_j(1) + p_j(2) - p_j(3)\} \\
q_7 = p_m \quad & m = \min_i\{\forall j : -p_i(1) - p_i(2) + p_i(3) \le -p_j(1) - p_j(2) + p_j(3)\} \\
q_8 = p_m \quad & m = \min_i\{\forall j : -p_i(1) - p_i(2) - p_i(3) \le -p_j(1) - p_j(2) - p_j(3)\}
\end{aligned}
$$

$$
\begin{aligned}
J_1^+ &= \{1,2,3,4\}, \qquad J_2^+ = \{1,2,5,6\}, \qquad J_3^+ = \{1,3,5,7\}. \\
J_1^- &= \{5,6,7,8\}, \qquad J_2^- = \{3,4,7,8\}, \qquad J_3^- = \{2,4,6,8\}
\end{aligned}
$$

(b)

Figure 2.7. (a) illustrates *eliminate* in 3 dimensions (for clarity the points are assumed to be inside the unit cube). Each of the 8 corner simplices $A_k(\alpha_k)$ are empty and have the point q_k on their defining hyperplanes. The q_k are found using the definitions of (b). *Eliminate* will dispose of all the points inside R as defined by (2.41) where the J_i^+ and the J_i^- are as given above.

what can we say about $\mathtt{E}(M)$? We parallel the two dimensional analysis and define a d-cube $B_X \subseteq \mathtt{R}$. Defining Z as the number of points outside B_X we see that $Z \geq M$. Thus showing that $\mathtt{E}(Z) = O(n^{(d-1)/d})$ will show that $\mathtt{E}(M) = O(n^{(d-1)/d})$. Our method will be a straightforward generalization of the two dimensional analysis of §2.4 yielding multdimensional theorems analagous to Theorems 2.1 and 2.3.

In two dimensions we examined the four isosceles right triangles with base edges of length α at the four corners of the unit square and determined the probability of their being empty. These triangles, denoted by $A_k(\alpha)$, had base edges that lay on the border of the unit square and hypotenuses that lay on their corresponding sweep lines.

Generalizing this idea to d-dimensions requires introducing the concept of a regular α-isoceles simplex or, more simply, an *α-simplex*. We define this to be a full dimensional simplex that has a corner p with the following properties: each edge touching p has length α and any two edges leaving p are perpendicular to each other. We will call p the *anchor* of the simplex. Equivalently we will say that the simplices are *anchored* at p. As examples both

$$\{x:\ x(1)+x(2)+\cdots+x(d)<\alpha,\quad 0\leq x(i)\ \ i=1,\ldots,d\}$$

and

$$\{x:\ x(1)+x(2)+\cdots+x(d)>d-\alpha,\quad x(i)\leq 1\ \ i=1,\ldots,d\}$$

are α-simplices. The first is anchored at $(0,0,\ldots,0)$ and the second at $(1,1,\ldots,1)$.

We will be interested in the 2^d α-simplices that are anchored at the 2^d corners of the unit d-cube and oriented so that their right triangular faces are flush with the faces of the d-cube. Each of these simplices is then fully identified by its remaining face. This face is of the form $c_k \cdot x^T = const$ (*const* a function of α to be determined below) which is just one of our sweep hyperplanes. Let $A_k(\alpha)$ be the α-simplex defined by the k-th hyperplane. Its anchor will be c_k^*, the first corner of the unit d-cube to be hit by the k-th hyperplane. This anchor satisfies

$$c_k^* \in [0,1]^d \ \text{ such that } \forall x \in [0,1]^d \qquad c_k \cdot c_k^{*T} \leq c_k \cdot x^T. \tag{2.42}$$

Note that c_k^* is the solution to a linear programming (minimization) problem. Since linear programming problems always have a vertex solution there is always a c_k^* that satisfies (2.42) and is a corner of the unit d-cube. We can find such a c_k^* by inspection.

By definition $c_k \cdot c_k^* = \sum_i c_k(i)\, c_k^*(i)$. If $c_k(i) = 1$ then setting $c_k^*(i) = 0$ minimizes the term $c_k(i)\, c_k^*(i)$. Similarly if $c_k(i) = -1$ we set $c_k^* = 1$. Letting $\mathbf{1} = (1, 1, \ldots, 1)$ we can rewrite this in vector notation as

$$c_k^* = \frac{1}{2}(\mathbf{1} - c_k). \tag{2.43}$$

Since this correspondence is one-one we can call c_k^* the *k-th corner of the unit d-cube.* This is notationally convenient; the k-th hyperplane first hits the cube at its k-th corner. With this notation it is easy to check that

$$A_k(\alpha) = \{x : \; c_k \cdot (x - c_k^*)^T \leq \alpha\} \tag{2.44}$$

is the α-isoceles simplex that is anchored at the k-th corner (Figure 2.6a) where we omit the appropriate feasibility conditions $0 \leq x(i)$ or $x(i) \leq 1$ because we already know that the points we are dealing with are distributed in the unit square.

Continuing with our generalization of the two-dimensional analysis we now need to define α_k such that the interior of $A_k(\alpha_k)$ is empty and there is at least one point on its defining hyperplane. By (2.39) we know that q_k will be on this hyperplane so

$$\alpha_k = c_k \cdot (q_k^T - c_k^{*T}). \tag{2.45}$$

Finally we set

$$\begin{aligned} X &= \max\{\alpha_1, \ldots, \alpha_{2^d}\}. \\ B_\alpha &= \{x : \; x \in (\alpha, 1-\alpha)^d\}. \\ Z &= |\{p_i : \; p_i \in B_X\}. \end{aligned} \tag{2.46}$$

It isn't hard to see[5] that $B_X \subseteq \mathrm{R}$ from which it follows

$$Z = \textit{number of points outside } B_X \geq \textit{number of points outside } \mathrm{R} = S.$$

---

[5] The main difficulty to seeing this is purely notational. From (2.45) $B_X \subseteq \mathrm{R}$ iff

$$\max_{k \in J_i^+} q_k(i) \leq X \qquad \text{and} \qquad 1 - X \leq \max_{k \in J_i^-} q_k(i) \qquad i = 1, \ldots, d. \tag{2.47}$$

To convince ourselves that (2.47) holds we note that from (2.45) $c_k^*(i) = 0$ then $q_k(i) \leq \alpha_k$ and if $c_k^*(i) = 1$ then $q_k(i) \geq 1 - \alpha_k$. The geometric meaning of this is that q_k is within the small d-cube with edge lengths α_k that is flush with the unit d-cube at c_k^*. If $k \in J_i^+$ then by (2.43) $c_k(i) = +1$ and thus by (2.41) $c_k^*(i) = 0$. Therefore $q_k(i) \leq \alpha_k \leq X$. Similarly if $k \in J_i^-$ then $c_k(i) = -1$ and $c_k^*(i) = 1$. Thus $1 - X \leq 1 - \alpha_k \leq q_k(i)$.

Establishing asymptotics for $\mathrm{E}(Z)$ will therefore also yield asymptotics for $\mathrm{E}(S)$.

Theorem 2.7: Given n points chosen I.I.D $\mathbf{U[0,1]^d}$and $X = \max\{\alpha_1, \ldots, \alpha_{2^d}\}$ then there is some constant c_d such that

$$\mathrm{E}(X) \sim c_d n^{-1/d}. \tag{2.48}$$

Proof: The proof is a generalization of the one given for Theorem 2.1 and follows the same format.

(i) For all k, $\alpha_k \leq d$ (otherwise $A_k(\alpha_k)$ would contain the entire unit d-cube), and therefore $X \leq d$. Thus

$$\mathrm{E}(X) = \int_0^d \Pr(X \geq \alpha)\, d\alpha \tag{2.49}$$

where

$$\begin{aligned} \Pr(X \geq \alpha) &= \Pr(\max_k\{\alpha_k\} \geq \alpha) \\ &= \Pr\left(\bigvee_k \{\alpha_k \geq \alpha)\}\right) \\ &= \Pr\left(\bigvee_k \{A_k(\alpha) \text{is empty}\}\right). \end{aligned} \tag{2.50}$$

If $\alpha < 1/2$ then $A_k(\alpha)$ is totally contained inside the unit d-cube and so by the independence of the p_i

$$\Pr(A_k(\alpha) \text{ is empty }) = (1 - \mathit{Volume}(A_k(\alpha))^n. \tag{2.51}$$

It is a well known fact that the volume of an α-simplex is $\alpha^d/d!$ and therefore

$$\Pr(A_k(\alpha) \text{ is empty}) = (1 - \alpha^d/d!)^n. \tag{2.52}$$

Combining (2.50) and (2.52) we find that for $\alpha \geq n^{-1/d} \ln n$

$$\begin{aligned} \Pr(X \geq \alpha) &\leq 2^d \left(1 - \frac{\ln^d n}{d!\, n}\right)^n \\ &= O(n^{-\ln^{(d-1)} n/d!}) \\ &= O(n^{-\ln n}) \end{aligned} \tag{2.53}$$

leading us to

$$\mathrm{E}(X) = \int_0^{n^{-1/d} \ln n} \Pr(X \geq \alpha)\, d\alpha \; + \; O(n^{-\ln n}). \tag{2.54}$$

(ii) If we assume that n is large enough so that in the range of the integral in (2.54) $n^{-1/d}\ln n \leq 1/2$ then the 2^d α-simplices are disjoint and it is not diffcult to use the inclusion-exclusion principle to show that

$$\begin{aligned}\Pr(X \geq \alpha) &= \Pr\left(\bigvee_k (A_k(\alpha) \text{ is empty})\right) \\ &= \sum_k \Pr(A_k(\alpha) \text{ is empty}) - \sum_{k_1<k_2} \Pr(A_{k_1}(\alpha) \cup A_{k_2}(\alpha) \text{ is empty}) \\ &\quad + \sum_{k_1<k_2<k_3} \Pr(A_{k_1}(\alpha) \cup A_{k_2}(\alpha) \cup A_{k_3}(\alpha) \text{ is empty}) + \cdots \\ &\quad + (-1)^{k+1} \Pr(\cup_k A_k(\alpha) \text{ is empty}) \\ &= \sum_{1\leq k\leq 2^d} \binom{2^d}{k} (-1)^{k+1}(1-k\alpha^d/d!)^n. \end{aligned} \tag{2.55}$$

(iii)

$$\begin{aligned}\int_0^{n^{-1/d}\ln n} (1-k\alpha^d)^n \, d\alpha &= \int_0^{k^{-1/d}} (1-k\alpha^d)^n \, d\alpha \; + \; O(n^{-\ln n}) \\ &= \frac{1}{dk^{1/d}} \int_0^1 (1-u)^n u^{\frac{1}{d}-1} \, du \; + \; + O(n^{-\ln n}) \\ &= \frac{1}{dk^{1/d}} \beta(\frac{1}{d}, n+1) + \; O(n^{-\ln n}) \end{aligned} \tag{2.56}$$

which by Lemma 1.5 is

$$\frac{\Gamma(\frac{1}{d})}{dk^{1/d}} n^{-1/d} + O(n^{-\frac{1}{d}-1}).$$

(iv) We can use this last fact to evaluate (2.54). We integrate the value for $P(X \geq \alpha)$ given by (2.55) and get an an integral over a sum. Evaluating each of the sum's terms separately we find

$$\mathrm{E}(X) = c_d n^{-\frac{1}{d}}$$

where

$$c_d = \frac{\Gamma(\frac{1}{d})\sqrt[d]{d!}}{d} \Big(\sum_{1\leq k\leq 2^d} \binom{2^d}{k} (-1)^{k+1} k^{-\frac{1}{d}} \Big)$$

Q.E.D.

We can also prove the following generalization of Theorem 2.3.

Theorem 2.8: Given n points chosen I.I.D. $\mathbf{U[0,1]^d}$, $Z = |\{p_i : p_i \in B_X\}$, and and c_d as defined in Theorem 2.7

$$\mathrm{E}(Z) \sim 2dc_d n^{\frac{d-1}{d}}. \tag{2.57}$$

Proof:

(i)

$$\begin{aligned} \mathrm{E}(Z) &= \int_0^d \mathrm{E}(Z|X=\alpha) f_X(\alpha)\, d\alpha \\ &= \int_0^{n^{-1/d}\ln n} \mathrm{E}(Z|X=\alpha) f_X(\alpha)\, d\alpha + O(n^{1-\ln n}). \end{aligned} \tag{2.58}$$

(ii) $Z|\alpha_1,\ldots,\alpha_{2^d}$ has the same distribution as 2^d+W where W is a $B(n-2^d,p)$ binomial random variable with p the ratio between the area outside of $B_X \cup \bigcup A_k(\alpha_k)$ and the area outside of $\bigcup A_k(\alpha_k)$. The reasoning behind this uses the same area argument as in two dimensions. Thus $\mathrm{E}(Z|\alpha_1,\ldots,\alpha_{2^d}) = 2^d + p(n-2^d)$. If $X = \alpha < 1/2$ then the $A_k(\alpha_k)$ are volume disjoint and using the fact that, $\alpha_k \le \alpha$ for all k, we can calculate asymptotics for p as follows:

$$\begin{aligned} p &= \frac{1-(1-2\alpha)^d - \frac{1}{d!}\sum \alpha_k^d}{1-\frac{1}{d!}\sum \alpha_k^d} \\ &= 2d\,\alpha + O(\alpha^2) \end{aligned}$$

Integrating, as in (2.26), over $\alpha_1,\ldots,\alpha_{2^d}$ conditioned on $X=\alpha$ we find

$$\mathrm{E}(Z|X=\alpha) = 2dn\alpha + nO(\alpha^2) + O(1). \tag{2.59}$$

If $\alpha \le n^{-\frac{1}{d}} \ln n$ then

$$\mathrm{E}(Z|X=\alpha) = 2dn\alpha + O(n^{-\frac{1}{d}}). \tag{2.60}$$

(iii) Combining (2.58) and (2.60) yields

$$\mathrm{E}(Z) = 2d\,n \int_0^{n^{-1/d}\ln n} \alpha f_X(\alpha)\, d\alpha \;+\; O(n^{-\frac{1}{d}}).$$

Integrating this by parts

$$\mathrm{E}(Z) = 2d\,n \int_0^{n^{-1/d}\ln n} \Pr(X \ge \alpha)\, d\alpha \;+\; O(n^{-\frac{1}{d}}). \tag{2.61}$$

While proving Theorem 2.7 we evaluated the integral in (2.61) to be $c_d n^{-\frac{1}{d}} + O(n^{-\frac{1}{d}-1})$ and thus

$$\mathrm{E}(Z) = 2d\,n^{(d-1)/d} + O(n^{-\frac{1}{d}}).$$

Q.E.D.

Theorem 2.8 tells us that the multidimensional version of *eliminate* does, on average, eliminate most of the original points. In the 3-dimensional case only $O(n^{\frac{2}{3}})$ will remain. We can prove a generalization of Theorem 2.5 and show that running gift-wrapping after *eliminate* takes $O(n^{\frac{4}{3}})$ time on average. There is a much better result though. Preparata and Hong [**PS**] give $O(n \log n)$ worst case algorithm for finding convex hulls in three dimensions. We can thus obtain a result analagous to Theorem 2.6:

Theorem 2.9: The combined *eliminate*/Preparata-Hong 3-dimensional convex hull algorithm has an average case $O(n)$ running time. Furthermore *eliminate*'s running time strongly dominates that of the Preparata-Hong part.

Proof: The same as for Theorem 2.6.

This is not the first three dimensional expected linear time algorithm; Bentley and Shamo's [**BS1**] divide and conquer algorithm is easily proven to have expected linear time for a large number of distributions. The major benefit of using *eliminate* is the same here as it was in two dimensions: It is simple to program, runs fast, and disposes of most of the points. Therefore the actual convex hull algorithm, which in three dimensions tends to have very high overhead, only runs on a very small, $n^{1-1/d}$, number of points.

§2.7 Conclusions

The purpose of this chapter was to prove that, given n points uniformly distributed in the unit square, *eliminate* eliminates all but $O(\sqrt{n})$ of the the original pointset without destroying any information essential to the construction of the convex hull. If the n points are distributed in the unit d-cube then *eliminate* eliminates all but $O(n^{(d-1)/d})$ of them. Therefore, *eliminate* followed by an $O(n \log n)$ worst case convex hull algorithm (such as exist for 2 and 3 dimensional space) yields an algorithm that constructs a convex hull in linear expected time. Furthermore, the expected running time of the second item in the pair, the actual convex hull algorithm, will be sublinear. From the computational point of view another benefit is that *eliminate* performs only simple comparisons (is $x > a$?) while most convex hull algorithms perform more complicated geometric ones (is point P above line l?). For extremely large random point sets, it is the case that the convex hull can be found in about the time it takes to access each point twice (once to participate in various comparisons to compute R and once to be eliminated).

In particular, for random points chosen I.I.D. uniformly in the unit square, the method we describe certainly outperforms "Quickhull" and is substantially easier to program. Often, the points used for the first stage of "Quickhull" are chosen to make the implementation of the recursive step more convenient: our results show that a proper choice makes the recursive step unnecessary.

It should be made clear that our choice of the unit square (d-cube) as the rectangular support of the point distribution was purely expository; the analysis of *eliminate* in section two (four) works just as well (after scaling) if the points are uniformly distributed in any rectangle (rectangular d-prism) oriented so that its sides are parallel to the Cartesian axes.

A final point that needs to be made is that there is another classic perspective on convex hulls which, while not yielding an analysis of *eliminate*, does provide some intuitive rationale as to why it performs as well as it does. If we are given a set S of n points I.I.D. uniformly in a bounded convex polygon P then, as $n \to \infty$, the convex hull of S will approach the perimeter of P very closely. An alternative way of expressing this is that the $CH(S)$ is a very close approximation to P. This suggests that the number of points on the hull will be comparatively small (and the number distributed inside will be relatively large). In fact this is the content of a famous theorem due to Rényi and Sulanke [**RS**]: If h signifies the number of points on the hull then $\mathtt{E}(h) \sim c \log n$, where c is a function of the number of edges of P.

In our case P is the unit square. Since the rectangle `R` constructed by *eliminate* is defined by the four points "closest" to the corners we can see `R` as a very close approximation of the square and therefore of the convex hull (X can be seen as a measure of this closeness). The points that are left will be squeezed between the perimeter of the hull and the perimeter of `R` and therefore, intuitively, should be few in number.

Chapter 3. Closest Pair Algorithms

§3.1 Introduction

The *closest pair* problem, finding the closest pair among a set of points, is one of the most basic problems in computational geometry. In §1.4 we showed that, given n points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$, the expected distance between the closest pair is $O(n)$. In this chapter we analyze two methods for finding the closest pair. A large number of algorithms, approaching the problem from many different perspectives, exist for its solution. There are "practical" $O(n \log n)$ time algorithms that are relatively simple and easy to program: Shamos and Hoey's [**Sha1**] divide-and-conquer algorithm and Hinrichs, Nievergelt, and Schorn's [**HNS**] sweep-line algorithm are representative. There are two randomized algorithms, one due to Rabin [**Ra**], the other to Weide [**We**], that both run in expected linear time on all inputs. There is a "theoretical" algorithm due to Fortune and Hopcroft [**FH**], really a modification of [**Ra**], that manages to get rid of the randomization by adding complexity and raising the running time. There are algorithms that solve the closest pair problem as a side effect of solving a more complicated problem such as constructing Voronoi diagrams [**SH**]. There is also an algorithm, due to Bentley, Weide, and Yao [**BWY**], that finds the closest pair by using a "gridding method", i.e. by using the floor function. When run on n points drawn I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$, this last algorithm requires $O(n)$ expected time.

In this chapter we will analyze the probabilistic performance of two algorithms for finding the distance between the closest pair. Both of these algorithms can be easily modified to return the points in the pair rather than the distance between them: we concentrate on distance finding procedures because they are conceptually easier to describe. Most of this chapter will be devoted to analyzing the probabilistic behavior of the first algorithm, the sweep-line algorithm described by [**HNS**]. We will then show how to use the techniques developed for analyzing that algorithm to analyze a variant of the projection algorithm described by Bentley and Papadimitriou [**BP**].

The purpose of this chapter is twofold: one practical, the other theoretical. The practical purpose is to show that, in a probabilistic sense, it is unnecessary to use complicated data structures such as 2-3 or AVL trees in the sweep-line algorithm.

The algorithm is very simple. Its first step sorts the points by their x coordinates using whatever efficient sorting algorithm is available. Its second step is a straight scan through the points. The main result of this chapter is to show that the scan will take, on average, linear time.

The theoretical purpose is the same as in the rest of this thesis: to provide direct average case analyses of algorithms that were designed to be simple but was not designed to be simply analyzed. We should stress that this chapter provides no new *theoretical* results in the sense of providing improved asymptotic ($O()$ notation) running time. It does, however, provide simple, fast algorithms.

A short description of the sweep-line algorithm is provided in §3.2. In §3.3 we perform its analysis and calculate the expected running time of the algorithm given that its n input points are chosen I.I.D. $\mathbf{U[0,1]^2}$. We demonstrate that, even though it's scan stage has $\Omega(n^2)$ worst case performance, it runs in linear expected time when the points are chosen I.I.D. $\mathbf{U[0,1]^2}$. We also compare our theoretical statements to experimentally derived results.

In §3.4 we analyze a a variant of Bentley and Papadimitriou's [**BP**] projection algorithm, one whose worst case performance can be shown to be $O(n^{3/2})$ when *closest* is defined using the L_∞ metric. We show that after performing two sorts, this algorithm's second stage will use only linear expected time when its input points are chosen I.I.D. $\mathbf{U[0,1]^2}$. Finally, in §3.5, we discuss extending the analysis to other distributions and why it can't be extended to higher dimensions

§3.2 Description of the Sweep Line Algorithm

The algorithm we consider here is the sweep line algorithm for computing the closest distance[1] among points in the plane. Before presenting our analysis we give a quick description of the algorithm. For a more complete description as well as a correctness proof and implementation details see [**HNS**].

Suppose we are given n points $p_i = (x_i, y_i), \quad i = 1 \ldots n$, in the plane. Sort them in increasing x order (arbitrarily breaking ties). We use the standard notation for order statistics. The i'th sorted point is denoted by $p_{(i)} = (x_{(i)}, y_{(i)})$ where

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n-1)} \leq x_{(n)}.$$

[1] In our description of the algorithm and in its analysis we will assume that *closest* is defined by the Euclidean (L_2) metric. Later, in §3.3.4, we will show how to generalize the description and the analysis to any L_p metric, $p \geq 1$.

The x-coordinate of p_i is the i-th smallest x-value. The y-coordinate of $p_{(i)}$ is the y coordinate that was associated with $x_{(i)}$. For example, if the original points were $p_1 = (5,3)$, $p_2 = (3,7)$, and $p_3 = (18,6)$, then the relabeled points are $p_{(1)} = (3,7)$, $p_{(2)} = (5,3)$, and $p_{(3)} = (18,6)$.

Let δ_i be the distance between between the closest pair among the first i points, i.e.

$$\delta_i = \min_{1 \le k < l \le i} d\left(p_{(k)}, p_{(l)}\right)$$

where $d((x,y),(x',y')) = \sqrt{(x-x')^2 + (y-y')^2}$. The closest pair distance problem is then seen to be the problem of determining δ_n. It follows from the definition of δ_i that

$$\delta_{i+1} = \min\left(\delta_i, \min_{1 \le k \le i} d\left(p_{(i+1)}, p_{(k)}\right)\right). \tag{3.1}$$

Updating δ_i only requires calculating the minimum among $d\left(p_{(i+1)}, p_{(k)}\right)$. Furthermore, if there is some $k \le i$ such that $d\left(p_{(i+1)}, p_{(k)}\right) < \delta_i$ then for that k, $x_{(i+1)} - x_{(k)} < \delta_i$. Thus we only have to check the distance between $p_{(i+1)}$ and those points whose x-coordinates are in the interval $(x_{(i+1)} - \delta_i, x_{(i+1)}]$, a region which we will call the δ_i-interval (see Figure 3.1(a)).

There is a geometric fact (proven in [**HNS**]) that is essential to understanding how to assure the algorithm $O(n \log n)$ worst case behavior. Suppose we have sorted the points in the δ_i-interval by their y-coordinates. Furthermore suppose that

$$\exists k \le i \quad \text{such that} \quad d\left(p_{(i+1)}, p_{(k)}\right) < \delta_i.$$

Then $p_{(k)}$ has to be one of the four points immediately above or one of the four points immediately below the point $p_{(i+1)}$ in the δ_i-interval. Thus, if the points in the δ_i-interval are stored in a balanced tree sorted by y-coordinate, we can calculate δ_{i+1} in $O(\log n)$ time by retrieving these eight points. All that remains is to describe how to update the δ_i-interval, i.e. update the y-sorted balanced tree so that it contains the points in the δ_{i+1}-interval. This is not difficult. First we insert $p_{(i+1)}$ into the balanced tree in $O(\log n)$ time. Then, using the fact that we already have the points sorted in increasing x-order (and have stored somewhere the index of the leftmost point in the δ_i-interval) we scan rightward deleting points from the tree until we enter the δ_{i+1}-interval (see Figure 3.1(b)). Since each point is deleted at most once during the execution of the algorithm the total time for deletion is $O(n \log n)$. Furthermore, since all other operations are $O(\log n)$ time per step we

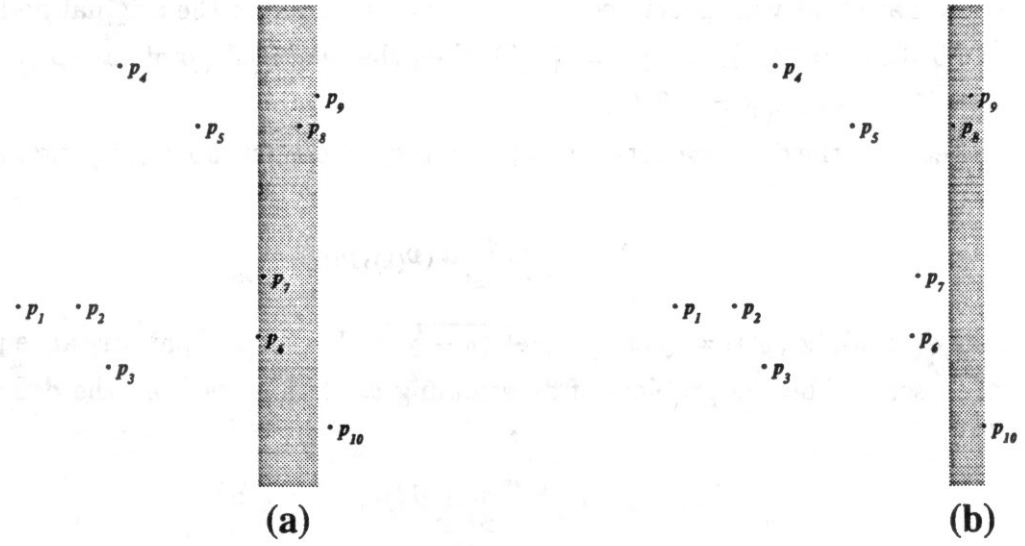


Figure 3.1. The minimum distance among all points preceeding p_9 is $d(p_1, p_2)$ so $\delta_8 = d(p_1, p_2)$. In (a) the shaded region has width δ_8 and is what we called the δ_8-interval. While checking all the points in the δ_8-interval against p_9 we find that $d(p_9, p_8) < \delta_8$ and thus $\delta_9 = d(p_9, p_8)$. We now find the δ_9 interval by scanning rightward from p_6 until we hit the first point (p_8) whose x-coordinate is within δ_9 of $p_{10}.x$. During this scan both p_7 and p_8 were deleted from the current δ interval. This gives us Figure (b).

---

find that the entire second stage of the algorithm can be implemented in $O(n \log n)$ time.

The main purpose of this chapter is to show that, probabilistically, the performance of the second stage of this algorithm – scanning through the sorted points while updating δ_i – will be much better than the worst case analysis suggests. In fact, we will show that it is possible to dispense totally with complicated balanced tree data structures and *still* get very good behavior: if we compare $p_{(i+1)}$ to *all* of the points in the δ_i-interval the algorithm will still run very fast on the average. We give a short but full listing of code for this version of the algorithm in Figure 3.2. It is not hard to see that the algorithm will require $\Omega(n^2)$ time in the worst case when run on n, x-sorted points, e.g. if the points are all on the same vertical line. We will show that, on average, it will take linear time.

Mathematically, given n points $p_1, \ldots p_n$, chosen I.I.D. $\mathbf{U}[\mathbf{0}, \mathbf{1}]^\mathbf{2}$, we set

$$\mathbf{N}_i = \text{ number of points in the } \delta_i\text{-interval .}$$

Figure 3.2. A Pascal listing of code that implements the linear scan to find the closest pair. We assume that the points are sorted by x-coordinate in the array $p[]$ and that there is a distance function, *dist(p,q)*, that returns the distance between points p and q. *Left* and i point to the leftmost and rightmost points in the δ_i-interval. The **while** loop updates the points in the δ_i-interval. The inner **for** loop updates the current value of δ_i by comparing $p[i+1]$ to all of the points in the δ_i-interval.

```
left := 1;   delta := dist(p[1], p[2]);
for i := 1 to n-1 do
   begin
      while (p[i+1].x - p[left].x) > delta do
         left := left + 1;
      for j:= left to i do
         if dist(p[j], p[i+1]) < delta then
            delta := dist(p[j], p[i+1]);
   end;
```

The amount of work needed to update δ_i to δ_{i+1} is $\mathbf{N}_i$ so the total amount of work performed by the algorithm is $\sum \mathbf{N}_i$. To prove that the algorithm runs in expected linear time we must show that

$$\mathrm{E}(\sum_{i=1}^{n} \mathbf{N}_i) = \sum_{i=1}^{n} \mathrm{E}(\mathbf{N}_i) = O(n). \tag{3.2}$$

It would be very satisfying to show that there exists some constant c such that $\mathrm{E}(\mathbf{N}_i) \leq c$ for all i. Linearity of the algorithm would follow immediately from (3.2). Unfortunately, there is no such c. As we shall soon see no better uniform bound than $\mathrm{E}(\mathbf{N}_i) = O(n^{1/3})$ can be found. This yields, at best $\mathrm{E}(\sum \mathbf{N}_i) = O(n^{4/3})$ which is not nearly good enough. While there is no uniform bound on the $\mathrm{E}(\mathbf{N}_i)$ we will be able to prove an adaptive one; the expected values decrease as i increases. Furthermore they decrease just fast enough for the cumulative work to be linear.

More specifically

$$\mathrm{E}(\mathbf{N}_i) = O\left(\sqrt{\frac{n}{i}}\right). \tag{3.3}$$

Using the fact that $\sum_{i \leq n} 1/\sqrt{i} = O(\sqrt{n})$ this proves that

$$\mathrm{E}(\sum_{i=1}^{n} \mathbf{N}_i) = O\left(\sum_{i=1}^{n} \sqrt{\frac{n}{i}}\right) = O(\sqrt{n})\, O\left(\sum_{1 \leq i \leq n} \frac{1}{\sqrt{i}}\right) = O(n). \tag{3.4}$$

We will also show that this bound is tight so a uniform bound can not exist: for large enough i there is a matching lower bound:

$$\mathrm{E}(\mathbf{N}_i) = \Omega\left(\sqrt{\frac{n}{i}}\right), \qquad i > n^{1/3} \lg n \tag{3.5}.$$

In the next section we will prove both of these facts.

§3.3 Analysis of the Sweep Line Algorithm

§3.3.1 Probabilistic Assumptions and Notation

We will assume that we are given n points $p_i = (x_i, y_i)$, $i = 1, \ldots, n$, chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^2$. The p_i-s are then sorted by x-coordinate[2] and renamed $p_{(i)} = (x_{(i)}, y_{(i)})$ where

$$x_{(1)} < x_{(2)} < \cdots < x_{(n-1)} < x_{(n)} \,. \tag{3.6}$$

We need to define the following random variables:

$$\begin{aligned} \mathbf{X}_{(i)} &= x_{(i)} \\ \delta_i &= \min_{1 \le k < l \le i} d\left(p_{(k)}, p_{(l)}\right). \end{aligned} \tag{3.7}$$

The distance function, $d(\cdot, \cdot)$, is the L_2 distance function. In §3.3.4 we will show how to generalize our analysis so that it is valid under any L_p metric. Given a real X we say that the *α-interval* is the strip of width α to the left of the line $x = X$. The number of points in an α-interval will be denoted by

$$\begin{aligned} L(X, \alpha) &= \text{ number of } p_i \text{ in the } \alpha \text{ strip to the left of } X \\ &= |\{p_j \mid X - \alpha \le x_j \le X\}| \end{aligned}$$

and the number of points in the δ_i-interval is

$$\mathbf{N}_i = L(\mathbf{X}_{(i+1)}, \delta_i).$$

Directly from the definitions we see that $\mathbf{N}_i \le i$. Our goal in this section is to analyze $\mathrm{E}(\mathbf{N}_i) = \mathrm{E}(L(\mathbf{X}_{(i+1)}, \delta_i))$. There is a major difficulty with any such analysis: if α is a constant then it is not difficult to calculate that the conditional expectation

$$\mathrm{E}(L(\mathbf{X}_{(i+1)}, \alpha) \mid \mathbf{X}_{(i+1)} = x) \;=\; \frac{i\alpha}{x} \tag{3.8}$$

---

[2] A technical quibble: this definition is not valid if there are two points with the same x-coordinate. We don't have to worry about this happening. If the points are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^2$ then two points sharing the same x-coordinate is a zero probability event which we can safely ignore. This restriction, that the points have unique x coordinates, is only made to simplify the analysis; the algorithm correctly finds the distance between the closest pair even when there are points with duplicated x-coordinates.

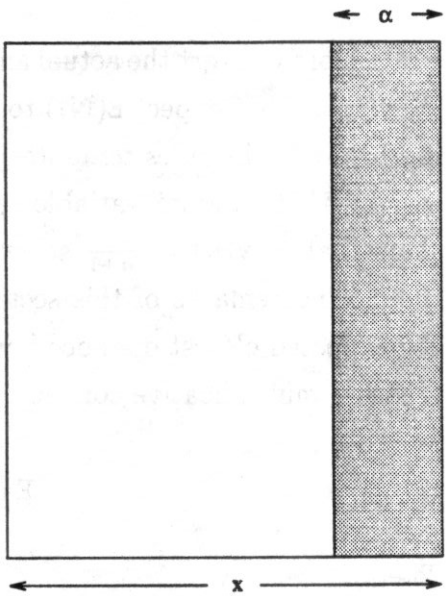


Figure 3.3. The large rectangle is R. The shaded rectangle is the α-interval. If a point is chosen from a uniform distribution in R then the point has probability α/x of being in the shaded region. If i points are chosen I.I.D. uniformly from R then the number of points in the shaded region is going to be a $\mathcal{B}(i, \alpha/x)$ binomially distributed random variable.

Conditioning on $\mathbf{X}_{(i+1)} = x$ means that the i points $p_{(1)}, \ldots, p_{(i)}$ are I.I.D. uniformly distributed in the rectangle

$$\mathtt{R} = [0, x] \times [0, 1]$$

which has area x. This makes $\left[L\left(\mathbf{X}_{(i+1)}, \alpha\right) \mid \mathbf{X}_{(i+1)} = x\right]$ a binomial[3] $\mathcal{B}(i, \alpha/x)$ random variable (Figure 3.3) with mean $i\alpha/x$.

This *does not imply* that

$$\mathtt{E}(\mathbf{N}_i \mid \mathbf{X}_{(i+1)} = x,\ \delta_i = \alpha) = \mathtt{E}(L(\mathbf{X}_{(i+1)}, \delta_i) \mid \mathbf{X}_{(i+1)} = x,\ \delta_i = \alpha) \quad = \quad \frac{i\alpha}{x}. \quad (3.9)$$

Conditioning on $\delta_i = \alpha$ totally changes the distribution of the points. It restricts all pairs of points from being less than a distance α from each other and also requires that there is some pair that is *exactly* a distance α apart. These extra conditions mean that we can no longer assume that $\left[\mathbf{N}_i \mid \mathbf{X}_{(i+1)} = x,\ \delta_i = \alpha\right]$ is binomially distributed. A good portion of our analysis will be devoted to developing a method of sidestepping this problem.

---

[3] A random variable Y is binomial $\mathcal{B}(m, p)$ if

$$\Pr(Y = k) = \begin{cases} \binom{m}{k} p^k (1-p)^{m-k} & \text{if } 0 \le k \le m \\ 0 & \text{otherwise.} \end{cases}$$

Before we start the actual analysis let's get some feel for why $\mathtt{E}(\mathbf{N}_i)$ is $O\left(\sqrt{n/i}\right)$. Why should we expect $\mathtt{E}(\mathbf{N}_i)$ to have this value? To answer this question we're going to break the rules temporarily and indulge in some illegal calculations; we will assume that random variables are *concentrated* at their mean. In §3.3.2 we will show that $\mathtt{E}(\mathbf{X}_{(i)}) = \frac{i}{n+1}$ so we start by setting $\mathbf{X}_{(i)} = \frac{i}{n+1}$. We can scale (see the closing remarks of this section) the techniques developed in §1.4.3 for finding the expected closest distance between n points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^2$ to show that, ignoring multiplicative constants,

$$\mathtt{E}\left(\delta_i \,|\, \mathbf{X}_{(i)} = x\right) \sim \frac{\sqrt{x}}{i}. \tag{3.10}$$

Therefore

$$\mathtt{E}\left(\delta_i \,|\, \mathbf{X}_{(i)} = \frac{i}{n+1}\right) \sim \frac{1}{\sqrt{in}}. \tag{3.11}$$

Again breaking the rules we'll assume both that δ_i is concentrated at its mean and that (3.9) is true. Then

$$\mathtt{E}\left(L(\mathbf{X}_{(i+1)}, \delta_i) \,|\, \mathbf{X}_{(i+1)} = \frac{i}{n+1}, \delta_i = \frac{1}{\sqrt{in}}\right) \quad = \quad \frac{i\frac{1}{\sqrt{in}}}{\frac{i}{n+1}} \sim \sqrt{\frac{n}{i}}. \tag{3.12}$$

Tragically neither real-life nor mathematics is ever this simple. This is especially true in probability theory where conditionality is a sad, inescapable fact of life. The laws of rigor intrude and force us to avoid all simplifying illegal assumptions, leaving the final proof much less straightforward.

§3.3.2 Order Statistics

We are now ready to start our analysis of $\mathtt{E}(\mathbf{N}_i)$. In this analysis we will find it useful to have upper and lower bounds for the $\mathbf{X}_{(i)}$. We take advantage of the fact that the $\mathbf{X}_{(i)}$ have been extensively studied [**De**] because they are the *order statistics* of random variables. More specifically, since the $p_i = (x_i, y_i)$ are I.I.D. uniformly in the unit square the x_i have to be I.I.D. uniformly distributed in $[0,1]$. The sorted x-coordinates, $\mathbf{X}_{(i)}$, are the order statistics of n uniformly distributed random variables on $[0,1]$. The probability density $f_{(i)}(x)$, of $\mathbf{X}_{(i)}$ is well known (see [**De**], p. 17, for a derivation) and is given by

$$f_{(i)}(x) = \frac{n!}{(i-1)!(n-i)!}\, x^{i-1}(1-x)^{n-i} \quad x \in [0,1]. \tag{3.13}$$

Using the standard properties of the Beta and Gamma functions the next lemma lets us calculate all of the moments of $\mathbf{X}_{(i)}$:

Lemma 3.1:

$$\mathrm{E}(\mathbf{X}_{(i)}^k) = \frac{n!\,(k+i-1)!}{(i-1)!\,(n+k)!}.$$

Proof.

$$\begin{aligned}
\mathrm{E}(\mathbf{X}_{(i)}^k) =& \frac{n!}{(i-1)!(n-i)!}\int_0^1 x^{k+i-1}(1-x)^{n-i}\,dx \\
=& \frac{n!}{(i-1)!(n-i)!}\beta(k+i, n-i+1) \\
=& \frac{n!}{(i-1)!(n-i)!}\frac{\Gamma(k+i)\Gamma(n-i+1)}{\Gamma(n+k+1)} \\
=& \frac{n!\,(k+i-1)!}{(i-1)!\,(n+k)!}.
\end{aligned}$$

These equations lead directly to

Corollary 3.2:

$$\begin{aligned}
\mathrm{E}(\mathbf{X}_{(i)}) &= \frac{i}{n+1}. \\
\mathrm{E}(\mathbf{X}_{(i)}^2) &= \frac{i\,(i+1)}{(n+1)\,(n+2)}. \\
\mathrm{Var}(\mathbf{X}_{(i)}) &= \mathrm{E}(\mathbf{X}_{(i)}^2) - \mathrm{E}^2(\mathbf{X}_{(i)}) = \frac{i\,(n+1-i)}{(n+2)\,(n+1)^2}.
\end{aligned} \tag{3.14}$$

This gives us the result that we will use in §3.3.3 to restrict the range of $\mathbf{X}_{(i+1)}$:

Corollary 3.3:

$$\Pr\left(\mathbf{X}_{(i+1)} \notin \left[\tfrac{1}{2}\tfrac{(i+1)}{(n+1)}, \tfrac{3}{2}\tfrac{(i+1)}{(n+1)}\right]\right) \leq \frac{4}{i+1}. \tag{3.15}$$

Proof. By Chebyshov's inequality and Corollary 1

$$\begin{aligned}
\Pr\left(\mathbf{X}_{(i+1)} \notin \left[\tfrac{1}{2}\tfrac{(i+1)}{(n+1)}, \tfrac{3}{2}\tfrac{(i+1)}{(n+1)}\right]\right) &= \Pr\left(|\mathbf{X}_{(i+1)} - \mathrm{E}(\mathbf{X}_{(i+1)})| > \frac{(i+1)}{2(n+1)}\right) \\
&\leq \mathrm{Var}(\mathbf{X}_{(i+1)})\left(\frac{2(n+1)}{(i+1)}\right)^2 \\
&\leq \frac{4}{(i+1)}.
\end{aligned}$$

§3.3.3 Asymptotics of $\mathtt{E}(\mathbf{N}_i) = \mathtt{E}(L(\mathbf{X}_{(i+1)}, \delta_i))$

Theorem 3.4: Choose n points I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$. Let $\mathbf{N}_i$ be the number of points in the δ_i-interval . Then

$$\mathtt{E}(\mathbf{N}_i) = \Theta\left(\sqrt{\frac{n}{i}}\right), \qquad i > n^{1/3}\lg n. \tag{3.16}$$

More specifically

$$\sqrt{\frac{2}{3\pi}}\sqrt{\frac{n}{i}} + O\left(\sqrt{\frac{n}{i^3}}\right) \le \mathtt{E}(\mathbf{N}_i) \le \sqrt{\frac{16}{\pi}}\sqrt{\frac{n}{i}} + 4 + O\left(\sqrt{\frac{n}{i^3}}\right).$$

Proof: The proof is complex and will be split into two parts. In the first we will show that we can effectively restrict $\mathbf{X}_{(i+1)}$ to the bounded subinterval $D_i = \left[\frac{1}{2}\frac{(i+1)}{(n+1)}, \frac{3}{2}\frac{(i+1)}{(n+1)}\right]$. In the second we will use this restriction to D_i to bound the conditional expectation $\mathtt{E}(L(\mathbf{X}_{(i+1)}, \delta_i) \,|\, \mathbf{X}_{(i+1)} = x)$ from above and from below.

Before starting the actual proof let's look at two reasons why the restriction $i > n^{1/3}$ is necessary. First, a quantitative reason: By definition $\mathbf{N}_i \le i$. If $i \ll n^{1/3}$ then $\sqrt{n/i} \gg n^{1/3}$ and the lower bound part of the theorem $\mathtt{E}(\mathbf{N}_i) = O\left(\sqrt{n/i}\right)$ can't be true[4].

Next, a qualitative one: Look at the average values "calculated" in (3.9) through (3.12). If $i \sim n^{1/3}$ then from Corollary 3.2 $\mathbf{X}_{(i+1)} \sim n^{-2/3}$. From (3.10) we also expect that $\delta_i \sim n^{-2/3}$. This leads us to believe that the δ_i-interval contains at least a constant fraction of the first i points and there is no appreciable savings made by the algorithm at this stage. Later we will see that for just this reason $i = n^{1/3}$ is an explicitly critical value in our calculations.

(i) Writing out the standard formula for conditional expectation we find that

$$\begin{aligned}\mathtt{E}(\mathbf{N}_i) &= \int_0^1 \mathtt{E}\left(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x\right) f_{(i+1)}(x)\, dx \\ &= \int_{x\in D_i} \mathtt{E}\left(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x\right) f_{(i+1)}(x)\, dx \\ &\qquad + \int_{x\notin D_i} \mathtt{E}\left(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x\right) f_{(i+1)}(x)\, dx.\end{aligned} \tag{3.17}$$

[4] At this point the attentive reader is probably asking why the the $\lg n$ factor in (3.16) is needed. The answer is that it isn't required for the proof. The theorem works just as well with any restriction $i > g(n)n^{1/3}$ where $g(n)$ montonically increases to infinity. The choice of $\lg n$ was made to simplify the proof.

From Corollary 3.3 and $\mathbf{N}_i \leq i$ we immediately see that

$$\int_{x \notin D_i} \mathrm{E}\left(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x\right) f_{(i+1)}(x)\, dx \leq i \cdot \Pr\left(\mathbf{X}_{(i+1)} \notin D_i\right)$$
$$\leq \frac{4i}{i+1} < 4.$$

We can therefore rewrite (3.17) as

$$\mathrm{E}(\mathbf{N}_i) = \int_{x \in D_i} \mathrm{E}\left(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x\right) f_{(i+1)}(x)\, dx \; + O(1). \tag{3.18}$$

(ii) We will show that, for $i > n^{1/3} \lg n$ and any $x \in D_i$,

$$\mathrm{E}\left(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x\right) = \Theta\left(\sqrt{\frac{n}{i}}\right).$$

The proof of the theorem will follow by the insertion of this last statement into (3.18).

From now on we assume that $\mathbf{X}_{(i+1)} = x$ for some constant $x \in D_i$. As before, R is the rectangular region to the left of the line $x = \mathbf{X}_{(i+1)}$:

$$\mathrm{R} \;=\; [0, x] \times [0, 1].$$

Conditioning on $\mathbf{X}_{(i+1)} = x$ we know that there are exactly i points in R. We can randomly relabel the points so that $q_1 = (x_1, y_1), \cdots, q_i = (x_i, y_i)$ are the ones in R. The crucial fact upon which our analysis will be based is that *after the points are relabeled the $q_1, \cdots, q_i$ are I.I.D. uniformly in* R.

We define new random variables (Figure 3.4):

$$M_j = \begin{cases} 1 & \text{if } x - \delta_i < x_j \leq x \\ 0 & \text{otherwise.} \end{cases}$$

This is the indicator function for the event that q_j *is in the δ_i-interval,* so $\mathbf{N}_i = \sum_{j \leq i} M_j$. Because the q_j-s are I.I.D., linearity of the expectation operator yields[5]

$$\mathrm{E}(\mathbf{N}_i) = i\mathrm{E}(M_1). \tag{3.19}$$

[5] As we said before, all random variables and expectations in this part are conditioned on $\mathbf{X}_{(i+1)} = x$. For the sake of clarity we don't write this explicitly. For example, (3.19) should be read as

$$\mathrm{E}(\mathbf{N}_i | \mathbf{X}_{(i+1)} = x) = i\mathrm{E}(M_1 | \mathbf{X}_{(i+1)} = x).$$

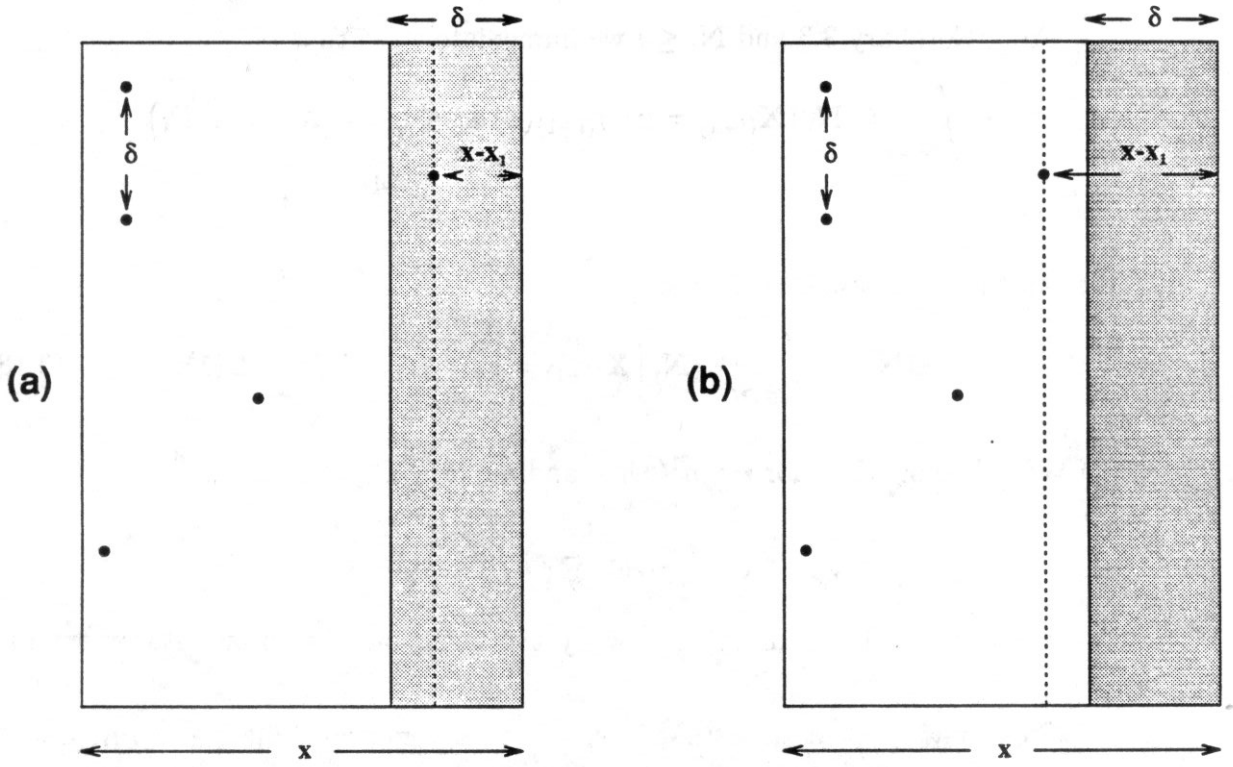


Figure 3.4. In both figures the large rectangle is R while the shaded one is the δ_i-interval. The point on the dotted line is q_1. In (a), q_1's distance, from the right boundary, $x - x_1$, is less than δ_i so $M_1 = 1$. In (b), the distance is greater than δ_i so $M_1 = 0$.

Since q_1 is (uniformly) distributed in R we can find $\mathrm{E}(M_1)$ by integrating over the conditional expectation given that q_1 is fixed, i.e.

$$\mathrm{E}(M_1) = \frac{1}{x} \int_{y_1=0}^{1} \int_{x_1=0}^{x} \mathrm{E}\left(M_1 \,|\, q_1 = (x_1, y_1)\right) \, dx_1 \, dy_1. \tag{3.20}$$

At first glance it might seem that all we've done is exchange one complicated integral, (3.17), for another. In the next few lines we give a few simple transformations that enable (3.20) to be rewritten in a more pliable form.

$$\begin{aligned}
\mathrm{E}(M_1) =& \frac{1}{x} \int_{y_1=0}^{1} \int_{x_1=0}^{x} \Pr\left(M_1 = 1 \,|\, q_1 = (x_1, y_1)\right) \, dx_1 \, dy_1 \\
=& \frac{1}{x} \int_{y_1=0}^{1} \int_{x_1=0}^{x} \Pr\left(x - \delta_i < x_1 \leq x \,|\, q_1 = (x_1, y_1)\right) \, dx_1 \, dy_1 \\
=& \frac{1}{x} \int_{y_1=0}^{1} \int_{x_1=0}^{x} \Pr\left(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)\right) dx_1 \, dy_1.
\end{aligned} \tag{3.21}$$

The first equality follows from M_1 being an indicator function, the second from the definition of M_1, and the third from the transformation $x_1 \rightarrow (x - x_1)$.

Our plan is to bound $\mathtt{E}(M_1)$ by calculating upper and lower bounds on $\Pr(\delta_i \geq x_1 \mid q_1 = (x - x_1, y_1))$ and integrating over them in the last line of (3.21). Calculating these bounds will require only minor modifications of the techniques derived in §1.4.3 for finding the closest distance between all pairs of points when the points are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$. The two major features that differentiate the current situation from that in the unit square are (a) that we are now conditioning over a particular placement of the first point $q_1 = (x_1, y_1)$ and (b) that the points are no longer I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$ but I.I.D. uniformly in $\mathtt{R}$.

(A) Lower bounds:

Recall the notation of §1.4.3. Given $q_1, \cdots, q_m$, points in the plane, Z was the distance between the closest pair, i.e.

$$Z(q_1, \cdots, q_m) = \min_{1 \leq k < l \leq m} d(q_k, q_l).$$

We proved that, when the points were I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$, then $\mathtt{E}(Z) = \Theta\left(n^{-1}\right)$. What happens if we change the distribution so that the points are now I.I.D. uniformly in $\mathtt{R} = [0, x] \times [0, 1]$ where $x \in D_i$?

In §1.4.3 we derived an alternative characterization of Z :

$$Z(q_1, \cdots, q_m) \geq \alpha \quad \Leftrightarrow \quad \forall k : \; q_k \notin \bigcup_{j<k} B(q_j, \alpha)$$

where $B(q_j, \alpha)$ is the Euclidean ball of radius α around q_j. We can think of this characterization as defining a point placement process. The points are sequentially placed in $\mathtt{R}$, each under the constraint that it is outside the α-neighborhoods of its predecessors. Viewing the situation from this perspective lets us write

$$\begin{aligned}
\Pr(Z(q_1, \cdots, q_m) \geq \alpha) &= \Pr(Z(q_1, q_2) \geq \alpha) \cdot \Pr(Z(q_1, q_2, q_3) \geq \alpha \mid Z(q_1, q_2) \geq \alpha) \\
&\quad \cdot \Pr(Z(q_1, q_2, q_3, q_4) \geq \alpha \mid Z(q_1, q_2, q_3) \geq \alpha) \cdots \\
&= \prod_{k \leq m} \Pr(Z(q_1, \cdots, q_k) \geq \alpha \mid Z(q_1, \cdots, q_{k-1}) \geq \alpha) \\
&= \prod_{k \leq m} \Pr\left(q_k \notin \bigcup_{j<k} B(q_j, \alpha) \mid Z(q_1, \cdots, q_{k-1}) \geq \alpha\right).
\end{aligned} \tag{3.22}$$

Because the q_k are uniformly distributed in R

$$\begin{aligned}
&\Pr\left(q_k \notin \bigcup_{j<k} B(q_j,\alpha) \mid Z(q_1,\cdots,q_{k-1}) \geq \alpha\right) \\
&\qquad = 1 - \mathrm{E}\left(\frac{Area\left[\bigcup_{j<k} B(q_j,\alpha)\cap \mathrm{R}\right]}{Area(\mathrm{R})} \;\middle|\; Z(q_1,\cdots,q_{k-1}) \geq \alpha\right) \\
&\qquad = 1 - \frac{\mathrm{E}\left(Area\left[\bigcup_{j<k} B(q_j,\alpha)\cap \mathrm{R}\right] \;\middle|\; Z(q_1,\cdots,q_{k-1}) \geq \alpha\right)}{x}.
\end{aligned} \tag{3.23}$$

The fact to be stressed here is that, no matter where the q_j are placed, we can always bound $Area\,(B(q_k,\alpha)\cap \mathrm{R}) \leq \pi\alpha^2$ so we can upper bound the expected area and therefore lower bound the product in (3.22) by

$$\Pr\left(Z(q_1,\cdots,q_m) \geq \alpha\right) \geq \prod_{k<m}\left[1 - \frac{k\pi\alpha^2}{x}\right]. \tag{3.24}$$

We will need a technical lemma to evaluate this product.

Lemma 3.5: For any constant $t > 0$

$$\prod_{k<m}\left[1 - tk\alpha^2/x\right] = e^{-tm^2\alpha^2/2x}\left[1 + O(m\alpha^2/x + m^3\alpha^4/x^2)\right]. \tag{3.25}$$

Proof: Standard asymptotic manipulations [**GKP**] give

$$\begin{aligned}
\prod_{k<m}\left[1 - tk\alpha^2/x\right] &= \exp\left[\sum_{k<m}\ln(1 - tk\alpha^2/x)\right] \\
&= \exp\left[\sum_{k<m}\left(-tk\alpha^2/x + O(k^2\alpha^4/x^2)\right)\right] \\
&= \exp\left[-tm(m-1)\alpha^2/2x + O(m^3\alpha^4/x^2)\right] \\
&= e^{-tm^2\alpha^2/2x}\left[1 + O(m\alpha^2/x + m^3\alpha^4/x^2)\right].
\end{aligned} \tag{3.26}$$

Q.E.D.

Return to (3.24) for a moment. Its derivation *never* used the fact that q_1 was chosen from a uniform distribution. This means the derivation remains valid even if we specify that q_1 has an arbitrary fixed value. Therefore

$$\Pr\left(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)\right) \geq \prod_{k<i}\left[1 - \frac{k\pi x_1^2}{x}\right]. \tag{3.27}$$

Remember that we are always assuming that $x \in D_i$ so we can write $x = ci/n$ where $1/2 \le c \le 3/2$. Setting $m = i$, $t = \pi$, and $\alpha = x_1$ we use Lemma 3.5 to evaluate the right hand side of the last equation.

$$\Pr(\delta_i \ge x_1 \,|\, q_1 = (x - x_1, y_1)) \ge e^{-\pi i n x_1^2/2c}\left[1 + O(nx_1^2 + in^2x_1^4)\right].$$

We can now upper bound $\mathrm{E}(M_j)$. Suppose[6] $x_1 = \sqrt{\lg n\,/in}$ then

$$e^{-\pi i n x_1^2/2c}\left[1 + O(nx_1^2 + in^2x_1^4)\right] = n^{-\Omega(\lg n)}. \tag{3.28}$$

Therefore, for any value of y_1

$$\begin{aligned}
&\int_0^1 \Pr(\delta_i \ge x_1 \,|\, q_1 = (x - x_1, y_1))dx_1 \\
&\qquad \ge \int_0^1 e^{-\pi i n x_1^2/2c}\left[1 + O(nx_1^2 + in^2x_1^4)\right] dx_1 \\
&\qquad = \int_0^{\frac{\lg n}{\sqrt{in}}} e^{-\pi i n x_1^2/2c}\left[1 + O(nx_1^2 + in^2x_1^4)\right] dx_1 + n^{-\Omega(\lg n)} \\
&\qquad = \sqrt{\frac{2c}{\pi i n}}\left[1 + O\left(\frac{1}{i}\right)\right].
\end{aligned} \tag{3.29}$$

Again using $x = ci/n$ we insert this back into (3.21) and get

$$\begin{aligned}
\mathrm{E}(M_1) &= \frac{1}{x}\int_0^1\int_0^1 \Pr(\delta_i \ge x_1 \,|\, q_1 = (x - x_1, y_1))dx_1 dy_1 \\
&\ge \frac{n}{ci}\sqrt{\frac{2c}{\pi i n}}\left[1 + O\left(\frac{1}{i}\right)\right]\int_0^1 dy_1 \\
&= \sqrt{\frac{2}{\pi c}}\sqrt{\frac{n}{i^3}} + O\left(\frac{\sqrt{n}}{i^5}\right).
\end{aligned}$$

Recalling (3.20) this gives the lower bound for $\mathrm{E}(N_i)$:

$$\mathrm{E}(N_i) = i\mathrm{E}(M_1) \ge \sqrt{\frac{2}{\pi c}}\sqrt{\frac{n}{i}} + O\left(\sqrt{\frac{n}{i^3}}\right) \tag{3.30}.$$

[6] This is where we use $i > n^{1/3}\lg n$. From the definitions of x and x_1 we must have $\sqrt{\lg n\,/in} = x_1 < x = ci/n$. If i was $O(n^{1/3})$ this would not be true, whereas for large enough n and $i > n^{1/3}\lg n$ it is always true.

Remember that all probabilistic statements this section were conditioned on $\mathbf{X}_{(i+1)} = x \in D_i$. Remember too, that for these x, we know that $1/2 \leq c \leq 3/2$. Therefore we've proven

$$\mathrm{E}(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x) \geq \sqrt{\frac{4}{3\pi}}\sqrt{\frac{n}{i}} + O\left(\sqrt{\frac{n}{i^3}}\right).$$

Reinserting this into (3.18) immediately proves the lower bound of the theorem.

(B) Upper bounds:

We will now prove an upper bound on $\Pr(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1))$. Inserted back into (3.21) this will give an upper bound on $\mathrm{E}(M_1)$ and thence $\mathrm{E}(N_i)$. In order to do this we will first find an upper bound on $\Pr(Z(q_1, \cdots, q_m) \geq \alpha)$ given that the q_j are I.I.D uniformly in R.

In §1.4.3 we provided a second characterization of Z:

$$Z(q_1, \cdots, q_m) \geq \alpha \quad \Leftrightarrow \quad \forall l \neq k, \quad B(q_l, \alpha/2) \cap B(q_k, \alpha/2) = \emptyset. \tag{3.31}$$

Assume for the moment that $\alpha \leq x \in D_i$. It follows that for every q_j at least one full quadrant (upper right, lower right, etc.) of $B(q_j, \alpha/2)$ is totally contained in R. If $Z(q_1, \cdots, q_{k-1}) \geq \alpha$ then combining this last fact with (3.31) gives

$$\begin{aligned} Area\left[\bigcup_{j<k} B(q_j, \alpha) \cap \mathrm{R}\right] &\geq Area\left[\bigcup_{j<k} B(q_j, \alpha/2) \cap \mathrm{R}\right] \\ &= \sum_{j<k} \mathrm{Area}\left[B(q_j, \alpha/2) \cap \mathrm{R}\right] \\ &\geq (k-1)\pi\alpha^2/4. \end{aligned} \tag{3.32}$$

This bounds the expected size of the area.

$$\mathrm{E}\left(Area\left[\bigcup_{j<k} B(q_j, \alpha) \cap \mathrm{R}\right] \;\middle|\; Z(q_1, \cdots, q_{k-1}) \geq \alpha\right) \geq (k-1)\pi\alpha^2/4.$$

We now mimic the transformations that yielded the lower bound in part (A). Insert the lower bound on the expected area into (3.28) and insert the resulting inequality into (3.22). This yields

$$\Pr(Z(q_1, \cdots, q_m) \geq \alpha) \leq \prod_{k<m}\left[1 - \frac{k\pi\alpha^2}{4x}\right]. \tag{3.33}$$

We now have an upper bound on Z but it looks like we still haven't developed a technique that enables us to deal with the conditioning on $q_1 = (x_1, y_1)$ in (3.21). In reality we don't need one: what we have is strong enough. No matter what the value of $q_1 = (x_1, y_1)$ it is always true that

$$\delta_i = Z(q_1, \cdots, q_m) \leq Z(q_2, \cdots, q_m).$$

Probabilistically this implies

$$\Pr(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)) \leq \Pr(Z(q_2, \cdots, q_i) \geq x_1).$$

Since $q_2, \cdots q_i$ are I.I.D. uniformly in R we can use (3.33) to bound

$$\Pr(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)) \leq \prod_{k<i-1} \left[1 - \frac{k\pi x_1^2}{4x}\right].$$

We again use Lemma 3.5 to evaluate the right side of this equation. The same sequence of steps that we used to find (3.29) now gives

$$\Pr(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)) \leq e^{-\pi i n x_1^2/8c} \left[1 + O(nx_1^2 + in^2x_1^4)\right]. \tag{3.34}$$

As we did for the upper bound, we again split our analysis into two parts depending on whether or not $x_1 > \sqrt{\lg n \,/in}$. For the moment we assume that $x_1 = \sqrt{\lg n \,/in}$. We would like to insert this value of x_1 into this last inequality and evaluate the resulting right hand side. Remember though that we based its derivation on (3.33). But in (3.33) we required that $\alpha < x$ which in (3.34) translates into $x_1 (= \alpha) \leq x$. But $\sqrt{\lg n \,/in} = x_1 \leq x = ci/n$ follows from the theorem's assumption that $i > n^{1/3} \lg n$. Therefore we can legally evaluate (3.34) with $x_1 = \sqrt{\lg n \,/in}$ giving

$$e^{-\pi i n x_1^2/8c} \left[1 + O(nx_1^2 + in^2x_1^4)\right] = n^{-\Omega(\lg n)}$$

and from the monotonicity of the probability distribution function

$$\Pr(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)) \leq n^{-\Omega(\lg n)}$$

for all $x_1 > \sqrt{\lg n \,/in}$.

To get the desired upper bound we now follow essentially the same sequence of transformations that we used in (3.29). We leave out the intermediate steps because they are almost exactly the ones we performed before.

$$\int_0^1 \Pr(\delta_i \geq x_1 \,|\, q_1 = (x - x_1, y_1)) dx_1 \leq \sqrt{\frac{8c}{\pi i n}} \left[1 + O\left(\frac{1}{i}\right)\right]$$

and

$$\mathrm{E}(M_1) \leq \sqrt{\frac{8}{\pi c}}\sqrt{\frac{n}{i^3}} + O\left(\frac{\sqrt{n}}{i^5}\right).$$

Inserted into (3.20) this yields the final result

$$\mathrm{E}(N_i) \leq i\mathrm{E}(M_1) \leq \sqrt{\frac{8}{\pi c}}\sqrt{\frac{n}{i}} + O\left(\sqrt{\frac{n}{i^3}}\right).$$

All probabilistic statements this section were conditioned on $\mathbf{X}_{(i+1)} = x \in D_i$. For these x, we know that $1/2 \leq c \leq 3/2$. Therefore we have actually proven that, for $x \in D_i$,

$$\mathrm{E}(\mathbf{N}_i \,|\, \mathbf{X}_{(i+1)} = x) \leq \sqrt{\frac{16}{\pi}}\sqrt{\frac{n}{i}} + O\left(\sqrt{\frac{n}{i^3}}\right)$$

Reinserting this into (3.18) immediately proves the upper bound of the theorem.

Q.E.D.

§3.3.4 Two Loose Ends Retied

In the preceding pages we made two claims that we did not prove because they were not needed by our analysis. One, made in in §3.3.1, was that we can directly evaluate $\mathrm{E}\left(\delta_i \,|\, \mathbf{X}_{(i)} = x\right)$. The other, made in §3.2 and §3.3.1, was that we can generalize the sweep-line algorithm and its analysis so that they both continue to be valid even when the distance function $d(q, q')$ is replaced by the distance function for the L_p metric ($p \geq 1$), $d_p(q, q')$. We now prove both of these claims.

We start by proving the first claim. In §3.3.1 we stated that the techniques of §1.4.3 could be scaled to derive

$$\mathrm{E}\left(\delta_i \,|\, \mathbf{X}_{(i)} = x\right) \sim \frac{\sqrt{x}}{i}. \tag{3.10}$$

We shall show that this statement is true whenever two conditions are fufilled. The first condition is the same one that Theorem 3.4 had: $i > n^{1/3} \lg n$. The second condition is more technical, that $x > i/(n \lg n)$. This is not a particularly onerous restriction since (3.13) shows that $\Pr(\mathbf{X}_{(i)} < i/(n \lg n))$ is extremely small.

Now, let $q_1, \cdots, q_i$ be chosen I.I.D. uniformly in $\mathtt{R} = [0, x] \times [0, 1]$. By definition

$$\Pr(Z(q_1, \cdots, q_i) \geq \alpha) = \Pr\left(\delta_i \geq \alpha \,|\, \mathbf{X}_{(i)} = x\right)$$

where Z is the distance between the closest pair among the points. During the proof of Theorem 3.4 we derived the following upper, (3.24), and lower, (3.33), bounds on this last expression:

$$\prod_{k<i}\left[1-\frac{k\pi\alpha^2}{x}\right] \le \Pr(Z(q_1,\cdots,q_i)\ge\alpha) \le \prod_{k<i}\left[1-\frac{k\pi\alpha^2}{4x}\right]. \tag{3.35}$$

The left inequality is always true. The right inequality is true when $\alpha < x$. Since

$$\mathrm{E}\left(\delta_i \,|\, \mathbf{X}_{(i)}=x\right) = \int_0^{\sqrt{2}} \Pr\left(\delta_i \ge \alpha \,|\, \mathbf{X}_{(i)}=x\right) \tag{3.36}$$

we may integrate the left and right sides of (3.35) to bound $\mathrm{E}\left(\delta_i \,|\, \mathbf{X}_{(i)}=x\right)$ from above and below. These bounds yield (3.10).

We will integrate the right side since we will need its value in §3.5. The integration of the left side is done similarly.

Lemma 3.5 lets us write rewrite the upper bound in (3.35) as

$$\begin{aligned}\Pr\left(\delta_i \ge \alpha \,|\, \mathbf{X}_{(i)}=x\right) &= \Pr(Z(q_1,\cdots,q_i)\ge\alpha) \\ &= e^{-\pi i^2\alpha^2/8x}\left[1+O(i\alpha^2/x + i^3\alpha^4/x^2)\right]\end{aligned} \tag{3.37}$$

where, as stated above, we assume that $\alpha < x$.

Let $\alpha = \sqrt{x}\lg n/i$. We would like to insert this value of α into (3.37). To do this we must show that $\sqrt{x}\lg n/i < x$. This will follow directly from our assumptions $i > n^{1/3}\lg n$ and $i/(n\lg n) < x$. The first assumption can be rewritten as $\lg^2 n/i^2 < i/(n\lg n)$. Combined with the second assumption this yields $\lg^2 n/i^2 < x$ which gives $\sqrt{x}\lg n/i < x$.

We can therefore insert $\alpha = \sqrt{x}\lg n/i$ into (3.37) to get

$$\Pr\left(\delta_i \ge \sqrt{x}\lg n/i \,|\, \mathbf{X}_{(i)}=x\right) = n^{-\Omega(\lg n)}.$$

We can now integrate the right side of (3.36) to calculate (3.10)'s upper bound.

$$\begin{aligned}\mathrm{E}\left(\delta_i \,|\, \mathbf{X}_{(i)}=x\right) \le& n^{-\Omega(\lg n)} + \int_0^{\sqrt{x}\lg n/i} e^{-\pi i^2\alpha^2/8x}\left[1+O(i\alpha^2/x+i^3\alpha^4/x^2)\right] \\ \le& n^{-\Omega(\lg n)} + \int_0^{\infty} e^{-\pi i^2\alpha^2/8x}\left[1+O(\frac{\lg^4 n}{i})\right] \\ =& O\left(\frac{\sqrt{x}}{i}\right).\end{aligned} \tag{3.38}$$

The $\lg^4 n/i$ value in the second inequality comes from substituting $x > i/n\lg n$ into the $O()$ term.

Now we prove the second claim. Let $p_i = (x_i, y_i)$, $i = 1, \ldots, n$. We must first show that the algorithm still finds the closest pair when closest is defined by

$$d_p(p_i, p_j) = \left(|x_i - x_j|^p + |y_i - y_j|^p\right)^{1/p}$$

where $d_p(p_i, p_j)$ is the L_p metric ($1 \leq p \leq \infty$). We must then show that our analysis of the algorithm, when the p_i are chosen I.I.D. $\mathbf{U}[\mathbf{0}, \mathbf{1}]^{\mathbf{2}}$, remains valid.

The validity of the algorithm for the standard L_2 metric was a result of

$$\delta_{i+1} = \min\left(\delta_i, \min_{1 \leq k \leq i} d\left(p_{(i+1)}, p_{(k)}\right)\right). \tag{3.1}$$

and the paragraph immediately following it. The salient fact used there was that "if there is some $k \leq i$ such that $d\left(p_{(i+1)}, p_{(k)}\right) < \delta_i$ then for that k, $x_{(i+1)} - x_{(k)} < \delta_i$." This fact and (3.1) are still true when we relace $d(p, q)$ by any $d_p(p, q)$ (where $1 \leq p \leq \infty$), so the algorithm does find the closest pair for any L_p metric.

Next we show that our analysis of the algorithm remains valid after replacing $d(p_i, p_j)$ with $d_p(p_i, p_j)$. Reviewing the proof of Theorem 3.4 we find that it uses only four properties of $d(p_i, p_j)$. The first two properties deal with the size of the α-ball surrounding a point and how much of such a ball must be in the unit square: they are

$$\text{Area}\left(B(p, \alpha)\right) = \pi\alpha^2, \qquad \alpha \leq 1/2,$$

and

$$\pi\alpha^2/4 \leq \text{Area}\left[B(p, \alpha) \cap [0, 1]^2\right] \leq \pi\alpha^2, \qquad p \in [0, 1]^2 \text{ and } \alpha \leq 1/2. \tag{3.39}$$

The second two properties are characterizations of $Z(p_1, \cdots, p_m)$, the distance between the closest pair of points:

$$Z(p_1, \cdots, p_m) \geq \alpha \quad \Leftrightarrow \quad \forall k: \; p_k \notin \bigcup_{j<k} B(p_j, \alpha),$$

and

$$Z(p_1, \cdots, p_m) \geq \alpha \quad \Leftrightarrow \quad \forall l \neq k, \quad B(p_l, \alpha/2) \cap B(p_k, \alpha/2) = \emptyset.$$

In §1.4.5 we proved that these four properties are all true for any L_p metric ($1 \leq p \leq \infty$) if, in (3.39), we replace π by the area of the unit ball under the L_p metric,

$$\omega_{p,2} = \text{Area}\left(B_p((0, 0), 1)\right)$$

where

$$B_p(q, \alpha) = \{q' \mid d_p(q, q') < \alpha\}.$$

Therefore we can generalize Theorem 3.4 to any L_p metric:

Theorem 3.4': Choose n points I.I.D. $\mathbf{U}[0,1]^2$. Let the distance function be some L_p metric. Let $\mathbf{N}_i$ be the number of points in the δ_i-interval. Then

$$\mathtt{E}(\mathbf{N}_i) = \Theta\left(\sqrt{\frac{n}{i}}\right), \qquad i > n^{1/3} \lg n$$

where the constants implicit in the $\Theta()$ notation depend upon the metric used.

Inserting this result into (3.4) proves that the algorithm takes expected linear time under any L_p metric ($1 \leq p \leq \infty$).

§3.3.5 Simulation Results

Numerically, we can calculate the constants implicit in the theta notation. Using the fact that $\sum \sqrt{n/i} \sim 2n$ and $\sum \sqrt{n/i^3} = O(\sqrt{n})$ we can add over i in (3.16) to get

$$0.6514\ldots\, n + O(\sqrt{n}) \;\leq\; \mathtt{E}(\mathbf{N}_i) \;\leq\; 6.2567\ldots\, n + O(\sqrt{n}).$$

The constant on the right hand side is small so the algorithm should run relatively fast. How well does this work in practice? In Table 3.1 we provide statistics on $\mathtt{E}(\sum \mathbf{N}_i)$ collected by running the algorithm on random point sets. For each value of n we ran 100 random trials and averaged the results. The statistics seem to show that the running time of the algorithm is much closer to the lower bound than to the upper.

§3.4 The Projection Algorithm

In §3.2 and §3.3 we discussed the sweep-line algorithm and showed that, after the sort, it finds the closest (defined by any L_p metric, $p \geq 1$) pair in linear expected time when the points are chosen I.I.D. $\mathbf{U}[0,1]^2$. We mentioned in passing that the sweep-line algorithm has $\Omega(n^2)$ worst case behavior. In this section we present and analyze another algorithm for finding the closest pair, one based on Bentley and Papadimitriou's projection method [**BP**]. This algorithm also starts with a sort and also has a second pass that will find the closest (defined by any L_p metric, $p \geq 1$) pair in linear expected time when the input points are I.I.D. $\mathbf{U}[0,1]^2$. It will follow

n	$\sum_i \mathbf{N}_i$	$\frac{1}{n}\sum_i \mathbf{N}_i$
1000	1287.9 ± 437.6	1.287900 ± 0.4376
2000	2578.0 ± 870.9	1.288995 ± 0.4355
3000	3886.7 ± 1355.5	1.295583 ± 0.4518
4000	5218.2 ± 1704.2	1.304547 ± 0.4260
5000	6665.0 ± 2179.6	1.332996 ± 0.4359
10000	14254.9 ± 4981.7	1.425485 ± 0.4982
15000	19627.2 ± 6512.7	1.308477 ± 0.4342
20000	27536.2 ± 9550.5	1.376812 ± 0.4775
25000	36641.9 ± 13681.5	1.465676 ± 0.5473
30000	38866.0 ± 12462.7	1.295532 ± 0.4154
35000	50043.5 ± 13718.0	1.429814 ± 0.3919
40000	56985.9 ± 17986.8	1.424649 ± 0.4497
45000	63917.4 ± 20805.6	1.420386 ± 0.4623
50000	67907.2 ± 19342.2	1.358143 ± 0.3868

Table 3.1. For each value of n we generated 100 sets of n random points. Each set was sorted and $\sum \mathbf{N}_i$ was calculated. The value in the second column is the average value of $\sum \mathbf{N}_i$ over all 100 point sets. The value in the third column is the value in the second normalized by division by n.

---

from a result of [**BP**] that this new algorithm has an $O(n^{3/2})$ worst case running time when closest is defined by the L_∞ metric.

This section is divided into two parts. In §3.4.1 we present the projection algorithm for finding the closest pair. This algorithm is a modified version of the "nearest-neighbor" algorithm presented in [**BP**]. For more information about the algorithm and for an analysis of its worst case running time see [**BP**]. In §3.4.2 we analyze the expected behavior of the projection algorithm. We find that the techniques developed in §3.3 to analyze the sweep-line algorithm are directly applicable to the analysis of this new algorithm. Our result is that, after the sort, the second stage of the projection algorithm requires only $O(n)$ expected time when its input points, $p_1, \cdots, p_n$, are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^\mathbf{2}$. As was the case for the sweep-line algorithm, our analysis remains valid when "closest" is defined by any L_p metric, $p \geq 1$.

§3.4.1 The Algorithm

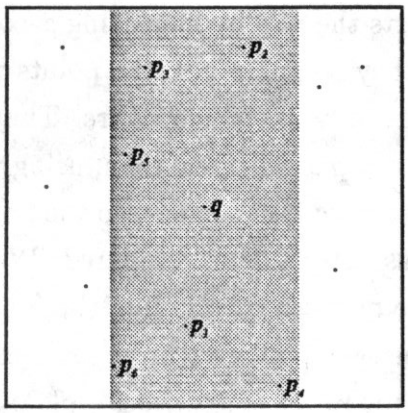
In [**BP**] Bentley and Papadimitriou describe an algorithm for solving the "post-office" problem. Given n points, $p_1, \cdots, p_n$, the "post-office" problem is to preprocess them so that, given a new point, q, we can always find the point p_i which is closest to q, i.e.

$$\forall j, \qquad d(q, p_i) \leq d(q, p_j).$$

The preprocessing step proposed by [**BP**] sorts the p_j's by increasing x-coordinate. To find the closest point to a new point, q, they compare q to the points which are nearest to q by projection on the x axis. This is a two step procedure. They first use a binary search to find p_i such that $p_i.x < q.x \leq p_{i+1}.x$. They then, using their own phrase, "search out," q's nearest neighbor by always comparing q to the point whose x-coordinate is closest to q and to which q has not yet been compared. While doing this they keep track of d, the closest distance between q and all of the points it has been compared to so far. They terminate the search when q has been compared to all points p_j whose x-coordinates are within the range $[q.x - d, q.x + d]$. We call this algorithm NNX : NNX(q) returns the distance between q and its nearest neighbor. In Figure 3.5(a) we provide a worked example of NNX and in Figure 3.5(b) we list pseudo-code implementing it.

The problem with this algorithm is that it can take $\Omega(n)$ time, e.g when all of the p_i lie on the same vertical line. In an effort to avoid this problem [**BP**] notes that there is nothing sacred about projection on the x-axis; we can just as easily write the algorithm so that it projects on the y-axis by "searching-out" the points whose y-coordinates are closest to q's y-coordinate. This algorithm, which we call NNY, also requires $\Omega(n)$ worst case time. We can also interleave projection on the x-axis with projection on the y-axis. That is, we can run the two versions, NNX and NNY, in lockstep: at each stage comparing q to the point with closest x-coordinate, and to the point with closest y-coordinate (Figure 3.6). This last version of the algorithm, which we call NN, has an interesting combinatorial worst case analysis. [**BP**] prove that if we run[7] NN(p_1), NN(p_2),... NN(p_{n-1}), and NN(p_n) the total time taken by the algorithm will be $O(n^{3/2})$ as compared to an $\Omega(n^2)$ worst case bound if we had run just the x-projection version NNX(p_1), NNX(p_2),... NNX(p_{n-1}), and

[7] By running NN(p_j) we mean finding the point $p_i \neq p_j$, such that p_i is closest to p_j. This requires us to slightly modify our code so that it p_j is not reported as its own nearest neighbor.



(a)

```
function NNX( q : point) : real;
   var     left, right, i : integer;
           delta, nextdist : real;
begin
   Use a binary search to find i
      such that   p[i].x < q.x <= p[i+1].x;

   left := i; right = i+1;
   delta := dist(q, p[i]);
   nextdist := min(p[left].x-q.x, q.x-p[right].x)

   while delta > nextdist do
      begin
         if q.x-p[left].x < p[right].x-q.x then
            begin
               delta := min(delta, dist(q, p[left]));
               left := left -1;
            end
         else
            begin
               delta := min(delta, dist(q, p[lright]));
               right := right +1;
            end
         end;
         nextdist := min(q.x-p[left].x, p[right].x-q.x);
      end;

   NN := delta
end;
```

(b)

Figure 3.5. **(a)** is a worked example of NNX. The algorithm scans through the p_i in the order defined by increasing size of $|q.x - p_i.x|$: it first looks at the p_i for which $|q.x - p_i.x|$ is minimal, then it looks at the one for which $|q.x - p_i.x|$ is second smallest, etc. It remembers which of the points scanned so far is closest to q. In this example it examines the points in the order $p_1, p_2, p_3, p_4, p_5, p_6$. The closest neighbor to q is p_5. After finding $d(q, p_5)$, the algorithm knows that it doesn't have to examine any points outside of the gray strip $\{(x, y) : |q.x - x| \leq d(q, p_5)\}$. Therefore it examines p_6, the last point that satisfies $|q.x - p_i.x| \leq d(q, p_5)$, and terminates, because it knows that p_5 is q's closest neighbor.

(b) is pseudo-code for NNX. We assume that the points are sorted by x coordinate so $p[i].x < p[i+1].x$ and that there are two dummy nodes $p[0] = (-\infty, -\infty)$ and $p[N+1] = (\infty, \infty)$. The function operates by always comparing q to the unchecked point with the closest x-coordinate. While doing this it updates *delta*, the minimum distance between q and all of the $p[j]$ that have been examined so far, i.e. the j which satisfy $left < j < right$. The function terminates when there is no unchecked point whose x-coordinate is within *delta* of $q.x$.

$\mathrm{NNX}(p_n)$.

We now modify this last algorithm so that it finds the distance between the closest pair. Obviously one way of doing this would be to calculate all of the values $\mathrm{NN}(p_1), \ldots, \mathrm{NN}(p_n)$ and report the minimum value. A more sophisticated technique uses information already calculated. Suppose we have already calculated

$$s_i = \min(\mathrm{NN}(p_1), \mathrm{NN}(p_2), \ldots, \mathrm{NN}(p_i)).$$

We will only be interested in calculating $\mathrm{NN}(p_{i+1})$ if $\mathrm{NN}(p_{i+1}) < s_i$. If we can convince ourselves that $\mathrm{NN}(p_{i+1}) \geq s_i$ then we can truncate the search performed by $\mathrm{NN}(p_{i+1})$. The truncation is implemented by stopping the execution of $\mathrm{NN}(p_{i+1})$ if there are no points in $\{(x,y) \,|\, p_{i+1}.x - s_i < x < p_{i+1}.x + s_i\}$ or if there are no points in $\{(x,y) \,|\, p_{i+1}.y - s_i < y < p_{i+1}.y + s_i\}$. In Figure 3.7 we present code for this truncated search closest-pair algorithm when we are only projecting/searching on the x-axis. We call this algorithm CPX. It is straightforward to generalize this code to perform interleaved x and y projections/searches. We name the interleaved closest pair algorithm CP. [**BP**]'s result implies that CP takes $O(n^{3/2})$ time in the worst case when *closest* is defined by the L_∞ metric. They also point out that, while, in general, they can't extend this $O(n^{3/2})$ bound to the L_2 metric, in practice, the use of finite precision arithmetic does let them extend it.

§3.4.2 Analysis

In this section we will prove that, after the initial sorting stage, algorithm CP will perform an expected linear number of computations if the input points are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^\mathbf{2}$. We will actually prove something stronger, that after its sorting stage, algorithm CPX will perform an expected linear number of computations if the input points are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^\mathbf{2}$. Since algorithm CP can not perform more than twice the number of comparisons that algorithm CPX performs, this implies the desired result. We mentioned before that [**BP**]'s analysis proves that algorithm CP takes $O(n^{3/2})$ worst case time when *closest* is defined by the L_∞ metric. Thus we have an algorithm that takes $O(n^{3/2})$ worst case time and, apart from the sorts, takes only $O(n)$ expected time.

Theorem 3.6: After the initial sorting stage, algorithm CPX performs only $O(n)$ expected comparisons when its input points, $p_1, \cdots, p_,$ are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^\mathbf{2}$.

Proof: As the first step in proving the theorem we must translate its statement into mathematical terms. We start by recalling some old definitions and introducing

```
function NN( q : point) : real;
    var     left, right, up, down, i : integer;
            delta, nextXdist, nextYdist : real;
begin
     Use a binary search to find i such that   p[i].x < q.x <= p[i+1].x;
     Use a binary search to find j such that   r[j].y < q.y <= r[j+1].y;

    left := i;  right := i+1; up := j+1; down := j;
    delta := dist(q, p[i]);
    nextXdist := min(p[left].x-q.x, q.x-p[right].x)
    nextYdist := min(r[up].y-q.y, q.y-r[down].x)

    while (delta > nextXdist ) AND (delta > nextYdist) do
        begin
            if q.x-p[left].x < p[right].x-q.x then
                begin
                    delta := min(delta, dist(q, p[left]));
                    left := left -1;
                end
            else
                begin
                    delta := min(delta, dist(q, p[iright]));
                    right := right +1;
                end;
            if q.y-r[down].y < r[up].y-q.y then
                begin
                    delta := min(delta, dist(q, r[down]));
                    down:= down -1;
                end
            else
                begin
                    delta := min(delta, dist(q, r[up]));
                    up := up+1;
                end
            end;
            nextXdist := min(q.x-p[left].x, p[right].x-q.x);
            nextYdist := min(q.y-r[down].y, r[up].y-q.y);
        end;

    NN := delta
end;
```

Figure 3.6. Pseudo-code for NN, the algorithm which finds q's nearest neighbor using interleaved projection on the x and y axes. We assume that the set of points is sorted by increasing x-coordinate so that $p[i].x < p[i+1].x$ and another copy of the points is sorted by increasing y-coordinate and stored in the $r[\,]$ array so that $r[i].y < r[i+1].y$. We also assume that there are two dummy nodes in each copy: $p[0] = r[0] = (-\infty, -\infty)$ and $p[N+1] = r[N+1] = (\infty, \infty)$. At each step by compares q to the previously unchecked point in $p[\,]$ with the closest x-coordinate and to the previously unchecked point in $r[\,]$ with the closest y coordinate. It is possible that a point will be checked twice: once during the x-scan and again during the y one. While performing these checks the algorithm updates *delta,* the minimum distance distance between q and all of the $p[i]$ and $r[k]$ that have been examined so far, i.e. the j that satisfy $left < i < right$ and the k which satisfy $down < k < up$. The function terminates when there is no unchecked point whose x-coordinate is within *delta* of $q.x$ *or* when there is no unchecked point whose y-coordinate is within *delta* of $q.y$.

```
function CPX( ) : real;
    var     left, right, i : integer;
            delta, nextdist : real;
begin
    left := i; right = i+1;
    delta := dist(p[1],p[2]);

    fori := 1 to N do
        begin
            nextdist := min(p[left].x-q.x, q.x-p[right].x)
            whiledelta > nextdist do
                begin
                    if q.x-p[left].x < p[right].x-q.x then
                        begin
                            delta := min(delta, dist(q, p[left]));
                            left := left -1;
                        end
                    else
                        begin
                            delta := min(delta, dist(q, p[lright]));
                            right := right +1;
                        end
                    end;
                    nextdist := min(q.x-p[left].x, p[right].x-q.x);
                end;
        end;
end;
```

Figure 3.7. Pascal code for CPX, the algorithm which finds the closest pair using projection on the x-axis. This code is a modification of the code given in Figure 3.5 for finding the nearest neighbor to a point. We assume that the points are sorted by increasing x-coordinate so $p[i].x < p[i+1].x$ and that there are two dummy nodes $p[0] = (-\infty, -\infty)$ and $p[N+1] = (\infty, \infty)$. The function operates by running through the $p[i]$ and, for each $p[i]$ performing a truncated search for its nearest neighbor: the nearest neighbor search is almost the same as it was in Figure 3.5. We truncate this procedure by halting the search for $p[i]'s$ nearest neighbor when we find that its nearest neighbor is further than *delta* away. Otherwise the nearest neighbor search is exactly the same as the one presented in Figure 3.5(a).

some new ones. Using these definitions will enable us to apply the results of §3.3 to this new problem. Recall our notation for order statistics. Given n points, $p_1, \cdots, p_i$, we sort them by increasing x-coordinate and rename them $p_{(i)} = (x_{(i)}, y_{(i)})$ where

$$x_{(1)} < x_{(2)} < \cdots < x_{(n-1)} < x_{(n)}\,. \tag{3.6}$$

As in the analysis of the sweep-line algorithm, we do not have to worry about two points sharing the same x-coordinate because this is a zero probability event when the points are chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$. We defined the random variables

$$\begin{aligned} \mathbf{X}_{(i)} &= x_{(i)} \\ \delta_i &= \min_{1 \le k < l \le i} d\left(p_{(k)}, p_{(l)}\right). \end{aligned} \tag{3.7}$$

$\mathbf{X}_{(i)}$ is the x-coordinate of the i'th sorted point and δ_i is the minimum distance between the first i sorted points. We also defined

$$\begin{aligned} L(X,\alpha) &= \text{ number of } p_j \text{ in the } \alpha \text{ strip to the left of } X \\ &= |\{p_j \,|\, X - \alpha \le x_j \le X\}|. \end{aligned} \tag{3.12}$$

We now introduce corresponding notation for the number of points in the α strip to the right of X :

$$\begin{aligned} R(X,\alpha) &= \text{ number of } p_j \text{ in the } \alpha \text{ strip to the right of } X \\ &= |\{p_j \,|\, X \le x_j \le X + \alpha\}|. \end{aligned} \tag{3.40}$$

It is very easy to see that, for $\alpha \le \beta$,

$$L(X,\alpha) \le L(X,\beta) \qquad \text{and} \qquad R(X,\alpha) \le R(X,\beta). \tag{3.41}$$

We also introduce a random variable whose value is the minimum distance between the first i points and all of the points:

$$\nu_i = \min_{\substack{1 \le k \le i \\ 1 \le l \le n \\ k \ne j}} d(p_{(k)}, p_{(j)}).$$

The definitions introduced above arise naturally in the analysis of algorithm CPX. The algorithm scans through the sorted points in order. It compares the current point it is scanning, $p_{(i)}$, to all of the points whose x-coordinates are within

ν_i of $\mathbf{X}_{(i)}$, i.e. while executing the truncated search for p_i's nearest neighbor it compares $p_{(i)}$ to all of the points in the vertical strip

$$\{(x,y) : |p_{(i)}.x - x| \leq \nu_i\}.$$

Therefore it performs exactly $L(\mathbf{X}_{(i)}, \nu_i) + R(\mathbf{X}_{(i)}, \nu_i)$ comparisons. We can now translate the statement of Theorem 3.6 into the following mathematical statement:

> "Given $p_1, \cdots, p_n$, n points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^\mathbf{2}$, sort and relable them by increasing x-coordinate so that $p_{(i)} = (x_{(i)}, y_{(i)})$ where
>
> $$x_{(1)} < x_{(2)} < \cdots < x_{(n-1)} < x_{(n)}.$$
>
> Then
>
> $$\mathtt{E}\left(\sum_{1\leq i\leq n} L(\mathbf{X}_{(i)}, \nu_i) + R(\mathbf{X}_{(i)}, \nu_i)\right) = O(n)." \tag{3.42}$$

We will prove (3.42) by separately proving that $\mathtt{E}\left(\sum_{1\leq i\leq n} L(\mathbf{X}_{(i)}, \nu_i)\right)$ and $\mathtt{E}\left(\sum_{1\leq i\leq n} R(\mathbf{X}_{(i)}, \nu_i)\right)$ are both $O(n)$.

(i) $\mathtt{E}\left(\sum_{1\leq i\leq n} L(\mathbf{X}_{(i)}, \nu_i)\right) = O(n)$:

We can prove this by applying the results of §3.3. First, from their definitions, we see that $\nu_i \leq \delta_i \leq \delta_{i-1}$. Therefore

$$\mathtt{E}(L(\mathbf{X}_{(i)}, \nu_i)) \leq \mathtt{E}(L(\mathbf{X}_{(i)}, \delta_{i-1})) = \mathtt{E}(\mathbf{N}_{i-1})$$

where $\mathbf{N}_i$ has been previously defined by

$$\mathbf{N}_i = L(\mathbf{X}_{(i+1)}, \delta_i).$$

Theorem 3.4 tells us that, for $i \geq n^{1/3}\lg n$, $\mathtt{E}(\mathbf{N}_i) = O(\sqrt{n/i})$. Therefore, following (3.4),

$$\begin{aligned}\mathtt{E}\left(\sum_{1\leq i\leq n} L(\mathbf{X}_{(i)}, \nu_i)\right) &\leq \mathtt{E}(\sum_{i=1}^{n} \mathbf{N}_i) \\ &= O\left(\sum_{i=1}^{n}\sqrt{\frac{n}{i}}\right) = O(\sqrt{n})\, O\left(\sum_{1\leq i\leq n}\frac{1}{\sqrt{i}}\right) = O(n).\end{aligned} \tag{3.43}$$

(ii) $\mathrm{E}\left(\sum_{1\le i\le n} R(\mathbf{X}_{(i)}, \nu_i)\right) = O(n)$:

We will prove this by deriving upper bounds on the individual $\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i))$. We start with an observation. Suppose that we fix $\mathbf{X}_{(i)} = x$. Futher, suppose that $\alpha < 1-x$ is a random variable that is independent of the locations of all of the points all of the points $p_{(i+1)}, p_{(i+2)}, \ldots, p_{(n)}$. Then $R(\mathbf{X}_{(i)}, \alpha)$ is a Binomially distributed $\mathcal{B}(n-i, \alpha/(1-x))$ random variable. This is because the points $p_{(i+1)}, p_{(i+2)}, \ldots, p_{(n)}$ are I.I.D. in the rectangle $[x,1] \times [0,1]$ which has area $1-x$: the probability that a particular point is in the strip $[x+\alpha, 1] \times [0,1]$ is $\alpha/(1-x)$. We have just shown that

$$\mathrm{E}(R(\mathbf{X}_{(i)}, \alpha) \,|\, \mathbf{X}_{(i)} = x) \;=\; \begin{cases} \frac{(n-i)\alpha}{1-x} & \text{if } \alpha \le x \\ \\ n-i & \text{otherwise.} \end{cases} \tag{3.44}$$

Notice that we have proven something quite powerful, namely that

$$\mathrm{E}(R(\mathbf{X}_{(i)}, \delta_i) \,|\, \mathbf{X}_{(i)} = x, \delta_i = \alpha) \;=\; \begin{cases} \frac{(n-i)\alpha}{1-x} & \text{if } \alpha \le x \\ \\ n-i & \text{otherwise.} \end{cases} \tag{3.45}$$

This is true because, even though δ_i is a random variable, it is independent of the random variables $p_{(i+1)}$, $p_{(i+2)}, \ldots, p_{(n)}$. Contrast this with the problems we encountered in §3.3.1 when trying to calculate $\mathrm{E}(L(\mathbf{X}_{(i)}, \delta_i) \,|\, \mathbf{X}_{(i)} = x, \delta_i = \alpha)$. There, the random variable was not Binomially distributed because conditioning on $\delta_i = \alpha$ skewed the distribution.

We will also need the following Lemma which lets us restrict the location of $\mathbf{X}_{(i)}$.

Lemma 3.7: Let $p_1, \cdots, p_n$ be chosen I.I.D. $\mathbf{U}[0,1]^2$ with $\mathbf{X}_{(i)}$ their order statistics as defined previously. Then the following statements are true for all i :

(1) $$\Pr\left(\mathbf{X}_{(i)} < \frac{i}{n \lg n}\right) \le O\left(\frac{1}{2^i}\right)$$

(b) $$\Pr\left(\mathbf{X}_{(i)} > 1 - \frac{(n-i)}{n \lg n}\right) \le O\left(\frac{1}{2^i}\right).$$

Proof: (1) We use the fact that we explicitly know $\mathbf{X}_{(i)}$'s probability distribution function

$$f_{(i)}(x) = \frac{n!}{(i-1)!(n-i)!} x^{i-1}(1-x)^{n-i}, \quad x \in [0,1]. \tag{3.13}$$

If $x \leq i/(n \lg n)$ then, for large enough n, Stirling's formula tells us

$$f_{(i)}(x) \leq \frac{n^i}{i!} \left(\frac{i}{n \lg n} \right)^{i-1} = O\left(\frac{n \lg n}{i 2^i} \right).$$

Therefore

$$\Pr\left(\mathbf{X}_{(i)} < \frac{i}{n \lg n} \right) = \int_0^{\frac{i}{n \lg n}} f_{(i)}(x)\, dx = O\left(\frac{1}{2^i} \right).$$

(2) This follows from (1) and the fact that $\mathbf{X}_{n-1}$ and $1 - \mathbf{X}_{(i)}$ have the same distribution function.

Q.E.D.

We are now ready to upper bound $\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i))$. We split this section of the proof into three parts depending on whether (a) $i \leq n^{1/3} \lg n$, (b) $i \geq n - n^{1/3}$, or (c) $n^{1/3} \lg n < i < n - n^{1/3}$.

(a) First we assume that $i \leq n^{1/3} \lg n$. Let μ_i be the minimum distance between $p_{(i)}$ and all of the points to its right:

$$\mu_i = \min_{j>i} d(p_i, p_j).$$

By definition, $\nu_i \leq \mu_i$, so (3.41) tells us that $R(\mathbf{X}_{(i)}, \nu_i) \leq R(\mathbf{X}_{(i)}, \mu_i)$ and thus $\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i)) \leq \mathrm{E}(R(\mathbf{X}_{(i)}, \mu_i))$. We would like to evaluate $\mathrm{E}(R(\mathbf{X}_{(i)}, \mu_i))$ by applying (3.44): we can not apply it because μ_i is not independent of the points to the right of $p_{(i)}$. To avoid this problem we will bound μ_i from above by some function of n, $g(n)$. Then $\mathrm{E}(R(\mathbf{X}_{(i)}, \mu_i)) \leq \mathrm{E}(R(\mathbf{X}_{(i)}, g(n)))$ and, since $g(n)$ is independent of the points to the right of $p_{(i)}$, we may use (3.44). to evaluate $\mathrm{E}(R(\mathbf{X}_{(i)}, g(n)))$. First, a simple area argument yields

$$\begin{aligned} \Pr(\mathbf{X}_{(n^{1/3} \lg n)} > 1/2) &= \sum_{0 \leq i \leq n^{1/3} \lg n} \binom{n}{i} 2^{-n} \\ &\leq \frac{n^{n^{1/3} \lg n}\, n^{1/3} \lg n}{2^n} \\ &= O\left(2^{-\sqrt{n}} \right). \end{aligned}$$

For all $i \leq n^{1/3} \lg n$, we know that $\mathbf{X}_i \leq \mathbf{X}_{n^{1/3} \lg n}$, so we have just proven that

$$\Pr(\mathbf{X}_{(i)} > 1/2) = O\left(2^{-\sqrt{n}} \right), \qquad i \leq n^{1/3} \lg n.$$

Since this is such a low probability we may assume that $\mathbf{X}_{(i)} < 1/2$. Under this assumption

$$\begin{aligned}\Pr(\mu_i \geq 1/\sqrt{n}) &\leq \left(1 - \frac{1/\sqrt{n}}{x}\right)^{n-i} \\ &\leq \left(1 - \frac{1}{2\sqrt{n}}\right)^{n/2} = O\left(e^{-\sqrt{n}/4}\right).\end{aligned}$$

Therefore we may also assume that $\mu_i < 1/\sqrt{n}$. This value is independent of the points $p_{(i+1)}, p_{(i+2)}, \ldots, p_{(n)}$ so we may finally use (3.44). Formally,

$$\begin{aligned}\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i)) \leq& \mathrm{E}(R(\mathbf{X}_{(i)}, \mu_i)) \\ \leq& \mathrm{E}\left(R(\mathbf{X}_{(i)}, \mu_i) \,|\, \mathbf{X}_{(i)} \leq 1/2,\, \mu_i \leq 1/\sqrt{n}\right) + O\left(e^{-\sqrt{n}/4}\right) \\ \leq& \frac{n-i}{1/2}\frac{1}{\sqrt{n}} + O\left(e^{-\sqrt{n}/4}\right) \\ =& O(\sqrt{n}).\end{aligned}$$

We have just shown that

$$\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i)) = O(\sqrt{n}) \qquad i \leq n^{1/3} \lg n \tag{3.46}$$

so

$$\sum_{1 \leq i \leq n^{1/3} \lg n} \mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i)) = O(n). \tag{3.47}$$

(b) Now we assume that $i \geq n - n^{1/3}$. The very definition of $R(X, \alpha)$ tells us that, deterministically, $R(\mathbf{X}_{(i)}, \nu_i)$ is less than or equal to the number of points to the right of $p_{(i)}$, i.e.

$$\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i)) \leq n - i \leq n^{1/3}, \qquad i \geq n - n^{1/3}.$$

Therefore we have just proven that

$$\sum_{n - n^{1/3} \leq i \leq n} \mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i)) = O(n). \tag{3.48}$$

(c) We have upper bounded $\mathrm{E}(R(\mathbf{X}_{(i)}, \nu_i))$ when i is small and we have upper bounded it when i is large. We must now upper bound it when i is in the midrange. We assume

$$n^{1/3} \lg n < i < n - n^{1/3}. \tag{3.49}$$

We start by using the same bounding argument that we used in part (i). We mentioned there that $\nu_i \leq \delta_i$ so

$$\mathtt{E}(R(\mathbf{X}_{(i)}, \nu_i)) \leq \mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i)).$$

It will therefore be sufficient for us to upper bound $\mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i))$.

Before calculating the upper bound we should provide some intuition as to where we are headed. Suppose for the moment that we have fixed $\mathbf{X}_{(i)} = x$ and $\delta_i = \alpha$. Then, subject to the constraint $\alpha \leq 1 - x$, we can use (3.45) to evaluate

$$\mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i) \,|\, \mathbf{X}_{(i)} = x, \delta_i = \alpha) = \frac{(n-i)\alpha}{1-x}. \tag{3.50}$$

Keeping $\mathbf{X}_{(i)}$ fixed and taking expectations over δ_i we find that

$$\mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i) \,|\, \mathbf{X}_{(i)} = x) = \frac{(n-i)}{1-x} \mathtt{E}(\delta_i \,|\, \mathbf{X}_{(i)} = x).$$

Now we can use the result of (3.38) which states that

$$\mathtt{E}\left(\delta_i \,|\, \mathbf{X}_{(i)} = x\right) = O\left(\frac{\sqrt{x}}{i}\right). \tag{3.51}$$

Plugging this in the previous equation would prove that

$$\mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i) \,|\, \mathbf{X}_{(i)} = x) = O\left(\frac{\sqrt{x}(n-i)}{(1-x)i}\right). \tag{3.52}$$

We finish by integrating over (3.52) to calculate an upper bound on $\mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i))$:

$$\mathtt{E}(R(\mathbf{X}_{(i)}, \delta_i)) = O\left(\int_0^1 \mathtt{E}\left(R(\mathbf{X}_{(i)}, \delta_i) \,|\, \mathbf{X}_{(i)} = x\right) f_{(i)}(x)\, dx\right). \tag{3.53}$$

This last equation can be evaluated using the Beta function techniques we employed earlier in the chapter.

The reason that we can not perform the analysis in the manner just described is that, in the foregoing sketch, we did not take into account the conditions that must be satisfied in order for the formulas to be valid. For example, to use (3.45) we must satisfy $\alpha \leq 1 - x$. Similarly to use (3.51) we must satisfy $x > i/(n \lg n)$.

Now, we start the real analysis. Assume for the moment that $\mathbf{X}_{(i)} = x$ where

$$\frac{i}{n \lg n} < x < 1 - \frac{n-i}{n \lg n}. \tag{3.54}$$

Our goal is to use (3.44) to show that

$$\mathrm{E}(R(\mathbf{X}_{(i)},\alpha)\,|\,\mathbf{X}_{(i)}=x) \;=\frac{(n-i)\alpha}{1-x}. \tag{3.55}$$

To use (3.44) we must restrict $\alpha < 1-x$. Therefore we would like to show that $\Pr(\delta_i > 1-x\,|\,\mathbf{X}_{(i)}=x)$ is small. To do this we use a formula derived in §3.3.4:

$$\Pr\left(\delta_i \geq \alpha\,|\,\mathbf{X}_{(i)}=x\right) = e^{-\pi i^2\alpha^2/8x}\left[1+O(i\alpha^2/x+i^3\alpha^4/x^2)\right]. \tag{3.37}$$

This equation is valid whenever $i > n^{1/3}\lg n$ and $x > i/(n\lg n)$, both of which are currently assumed to be true (3.49) (3.54). Evaluating this equation with $\alpha = 1-x$ we find that

$$\Pr\left(\delta_i \geq 1-x\,|\,\mathbf{X}_{(i)}=x\right) = e^{-\pi i^2(1-x)^2/8x}\;\left[1+O(n^5)\right]. \tag{3.56}$$

(We used the fact that $x > i/(n\lg n)$ to get the $O(n^5)$ term.) The important thing to realize is that, since we are assuming $(1-x) > (n-i)/(n\lg n)$ (3.49) and $n^{1/3} \leq i, n-i \leq n-n^{1/3}$ (3.54) we know that

$$i^2(1-x)^2 \geq \frac{n^{2/3}\left(n-n^{1/3}\right)^2}{n^2\lg^2 n} \geq \frac{n^{1/4}}{2}.$$

Inserting this back into (3.56) we get

$$\Pr\left(\delta_i \geq 1-x\,|\,\mathbf{X}_{(i)}=x\right) = O\left(e^{-n^{1/4}}\right). \tag{3.57}$$

If $\alpha < 1-x$ then we can apply (3.45) to get

$$\mathrm{E}(R(\mathbf{X}_{(i)},\delta_i)\,|\,\mathbf{X}_{(i)}=x,\delta_i=\alpha) \;=\; \frac{(n-i)\alpha}{1-x}.$$

Combining this with (3.57) yields

$$\mathrm{E}(R(\mathbf{X}_{(i)},\delta_i)\,|\,\mathbf{X}_{(i)}=x) \;=\; \frac{(n-i)}{1-x}\,\mathrm{E}(\delta_i\,|\,\mathbf{X}_{(i)}=x) + O\left(ne^{-n^{1/4}}\right). \tag{3.58}$$

We will use one more result from §3.3.4: (3.38) states that, when $i > n^{1/3}\lg n$ and $x > i/(n\lg n)$, then

$$\mathrm{E}\left(\delta_i\,|\,\mathbf{X}_{(i)}=x\right) = O\left(\frac{\sqrt{x}}{i}\right).$$

Combining this with (3.58) proves that

$$\mathrm{E}(R(\mathbf{X}_{(i)},\alpha)\,|\,\mathbf{X}_{(i)}=x)=O\left(\frac{(n-i)\sqrt{x}}{i(1-x)}\right),\qquad \frac{i}{n\lg n}<x<1-\frac{n-i}{n\lg n}.$$

We are now almost done. Inserting this last result into (3.53), applying Lemma 3.7, and using the definition of $f_{(i)}(x)$ given in (3.13) we find that

$$\begin{aligned}\mathrm{E}(R(\mathbf{X}_{(i)},\delta_i)) &= \int_{i/(n\lg n)}^{1-(n-i)/(n\lg n)} O\left(\frac{(n-i)\sqrt{x}}{i(1-x)}\right) f_{(i)}(x)\,dx + O\left(n/2^i\right)\\ &= O\left(\int_0^1 \frac{(n-i)\sqrt{x}f_{(i)}(x)}{i(1-x)}\,dx\right) + O\left(n/2^i\right)\\ &= O\left(\frac{n!}{i!(n-i-1)!}\int_0^1 x^{i-1/2}(1-x)^{n-i-1}\,dx\right) + O\left(n/2^i\right)\\ &= O\left(\frac{n!}{i!(n-i-1)!}\beta(i+1/2,n-i)\right) + O\left(n/2^i\right).\end{aligned}$$

Using $\beta(a,b)=\Gamma(a)\Gamma(b)/\Gamma(a+b)$ and Stirling's formula we see that we have just proven

$$\mathrm{E}(R(\mathbf{X}_{(i)},\delta_i))=O\left(\sqrt{\frac{n}{i}}\right),\qquad n^{1/3}\lg n<i<n-n^{1/3}.$$

We started this part of the proof by showing that

$$\mathrm{E}(R(\mathbf{X}_{(i)},\nu_i))\le\mathrm{E}(R(\mathbf{X}_{(i)},\delta_i))$$

so we have just proven that

$$\sum_{n^{1/3}\lg n<i<n-n^{1/3}}\mathrm{E}(R(\mathbf{X}_{(i)},\nu_i))\le\sum_{1\le i\le n}\sqrt{\frac{n}{i}}=O(n).$$

Combining this with (3.47) and (3.48) proves

$$\sum_{1\le i\le n}\mathrm{E}(R(\mathbf{X}_{(i)},\nu_i))=O(n)$$

and we are done.

Q.E.D.

In §3.3.4 we showed how to extend our analysis of the sweep line algorithm so that it remains valid when the L_2 metric is replaced by any L_p metric ($p\ge 1$). The

reasoning used there applies here as well. Therefore, we have really shown that, for n input points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$, algorithm CPX performs only $O(n)$ expected comparisons after the sort when closest is defined by any L_p metric ($p \geq 1$).

§3.5 Conclusions

We proved that – given n points uniformly distributed in the unit square and then sorted – a simple linear scan through the points of the type described in §3.2 will determine the closest pair in expected $\Theta(n)$ time. This gives a practical algorithm for finding the closest pair. First, sort the points by x coordinate. Next, run the linear scan. The $n \lg n$ time required by the sorting stage is dominant since we have proven that the scanning stage is linear. The advantage in using this algorithm is that you can use your favorite fine tuned sorting routine and therefore get an algorithm that runs fast on your particular machine. We also showed how to use the techniques we developed for analyzing the sweep-line algorithm to analyze the use of [**BP**]'s projection method for finding the closest pair.

An interesting open question is whether it is possible to tighten the results obtained here to show that there is some evauable constant σ such that

$$\frac{1}{n}\sum_{i\leq n}\mathtt{E}(\mathbf{N}_i) \rightarrow \sigma, \qquad n \rightarrow \infty.$$

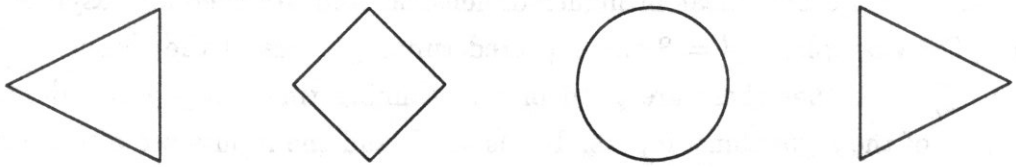
In order to find such a σ (at least if we continue using the techniques of this chapter) we would have to be able to calculate much tighter bounds on δ_i. Unfortunately, to the best knowledge of the author, no such bounds are known, even for the simple case of $\delta_n = Z(p_1, \cdots, p_n)$. It is not even known if there exists an evaluable γ such that

$$nZ(p_1, \cdots, p_n) \rightarrow \gamma, \qquad n \rightarrow \infty$$

where the $p_1, \cdots, p_n$ are $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$. If such a γ were known then it might be feasible to use this fact to get tighter bounds on the $\mathtt{E}(\delta_i)$ and thence on $\mathtt{E}(\mathbf{N}_i)$. One approach to finding such a γ could be to modify the Poisson process techniques utilized by Steele [**St**] for finding tight bounds on the probability that a given vertex of the Euclidean minimal spanning tree of a random point set has a specified degree. This approach offers the possibility of success.

Another open question is whether it is possible to extend the results of this chapter to points that are picked from other distributions. The underlying principle

behind the behavior of the $\mathbf{N}_i$ seems to be that (a) if i is small then $\mathbf{N}_i$ must be small whereas (b) if i is large then δ_i is small so $\mathbf{N}_i$ must be small as well. It is conceivable that this principal still holds for points chosen uniformly from other regions, e.g. the ones below.



A final question, and one that we must answer in the negative, is whether it is possible to extend the algorithm to $d > 2$ dimensions. The first step of the algorithm would still be to sort the points by x-coordinate and then relable them $p_{(i)} = (x_{(i)}, y_{(i)}, z_{(i)})$,

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n-1)} \leq x_{(n)}.$$

It would then scan through the points updating

$$\delta_i = \min_{1 \leq k < l \leq i} d(p_{(k)}, p_{(l)}).$$

As before it would do this by comparing $p_{(i+1)}$ to all of the points in the δ_i-interval which is now

$$(x_{(i+1)} - \delta_i, x_{(i+1)}] \times [0,1] \times \cdots \times [0,1].$$

The same style of off-the-cuff analysis ((3.10) through (3.12)) that told us that the (post-sort stage of the) algorithm is linear in two dimensions will show us that it is not linear in higher dimensions. As before we assume $x_{(i+1)} \sim i/n$. Scaling the techniques of §1.4(ii) we can show that

$$\mathrm{E}\left(\delta_i | x_{(i+1)} = \frac{i}{n}\right) \sim \frac{x_{(i+1)}^{1/d}}{i^{2/d}} \sim \left(\frac{1}{in}\right)^{1/d}.$$

Since we expect $\mathbf{N}_i$, the number of points in the δ_i-interval to be almost binomially $B(i, \delta_i/x)$ distributed we expect $\mathbf{N}_i$ to be about

$$\frac{i\delta_i}{x} \sim \frac{i\left(\frac{1}{in}\right)^{1/d}}{\frac{i}{n}} = \frac{n^{1-1/d}}{i^{1/d}}.$$

This would mean that

$$\sum_{i<n} \mathbf{N}_i \sim \sum_{i<n} \frac{n^{1-1/d}}{i^{1/d}} \sim n^{2(1-1/d)}.$$

The same result would hold for the analysis of the d-dimensional projection algorithm: after the sorting stage it would perform $\Theta\left(n^{2(1-1/d)}\right)$ expected comparisons.

We have just shown that the second stages of the two algorithms described in this chapter are not linear in higher dimensions and are therefore asymptotically bad. For example, if $d = 3$ the expected running times of the second stages are $\Theta\left(n^{4/3}\right)$. Whether these are good or bad running times in practice depends on the size of the algorithms' inputs. If d is small and the inputs are of medium size, e.g. $d = 3$ and $n = 10000$, then it might make sense to use the simple sweep line algorithm. If n is very large then it probably wouldn't.

Chapter 4. The Move To Front Maxima Algorithm

§4.1 Introduction

The purpose of this chapter is to address a conjecture posed by Bentley, Clarkson, and Levine in [**BCL**]. In that paper they constructed and analyzed a linear expected-time algorithm for finding the maxima of sets of points. They also presented another algorithm, a Move-To-Front (MTF) heuristic, that was simpler to code and more robust than the first algorithm

Experimental evidence suggested that the MTF algorithm was also faster than the first algorithm. More specifically, their simulations showed that, when run on n points, the MTF algorithm required, on average, only $n + o(n)$ point comparisons. They conjectured that this could be mathematically proven.

In this chapter we analyze the MTF algorithm under the assumption that the input points are I.I.D. from a Component Independent (CI) distribution in $\mathcal{R}^2$. Our main result is that, with probability $1 - n^{-\Omega(\lg n)}$, the heuristic will perform only $n + O\left(n^{6/7} \lg^5 n\right)$ point comparisons.

In §4.2 we define the maxima of a point set. We also introduce the concept of a CI distribution and calculate the distribution of the number of maxima in a set of n points that are chosen I.I.D. from a CI distribution.

In §4.3 we introduce [**BCL**]'s MTF algorithm. We also present the results of their simulations which led them to believe that it performs only $n + o(n)$ point comparisons.

In §4.4 we analyze their algorithm. We use an amortization argument to show that, with probability $1 - n^{-\Omega(\lg n)}$, it requires only $n + O\left(n^{6/7} \lg^5 n\right)$ point comparisons when run on inputs of n points in $\mathcal{R}^2$ chosen I.I.D. from a distribution with the CI property.

Finally, in §4.5, we conclude by discussing our results and the open problem of proving [**BCL**]'s conjecture in higher dimensions.

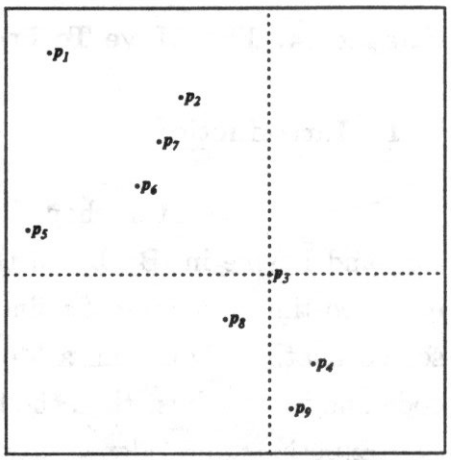


Figure 4.1: The maxima of this set of points are p_1, p_2, p_3, p_4. Drawing a pair of Cartesian axes through p_3 we see that the upper right quadrant is empty since p_3 is maximal. The maxima listed are the (++) maxima. The (−+) maxima are p_1 and p_5. The (−−) maxima are p_5, p_8, and p_9. The (+−) maxima are p_4 and p_9.

§4.2 Introduction To Maxima

§4.2.1 Definitions

In this section we define the maxima of a point set and describe some of their applications. We also discuss the relationship between maxima and convex hulls.

We start by examining the two dimensional case. Let $p = (p.x, p.y)$ and $q = (q.x, q.y)$ be points in $\mathcal{R}^2$. We say that p *dominates* q if $p.x > q.x$ and $p.y > q.y$. Let S be a set of points in $\mathcal{R}^2$. We say that $p \in S$ is a *maximal point* of S (sometimes known as a *maximum*) if p is not dominated by any point in S. An equivalent definition involves drawing Cartesian axes parallel to the standard ones and whose origin is p: p will be a maximal point if and only if the upper right quadrant formed by these new axes contains no points in S (Figure 4.1). The maximal points of S are known as the *maxima* of S.

We can easily extend the definition of maxima to higher dimensions. Let $p = (p_1, \ldots, p_d)$, $q = (q_1, \ldots, q_d)$ be points in d-space. We say that p *dominates* q if $p_i > q_i$ for all $1 \leq i \leq d$. The *maximal elements* of a set $S \subseteq \mathcal{R}^d$ are the points in S that are not dominated by any point in S. As before we call these points the *maxima* of S.

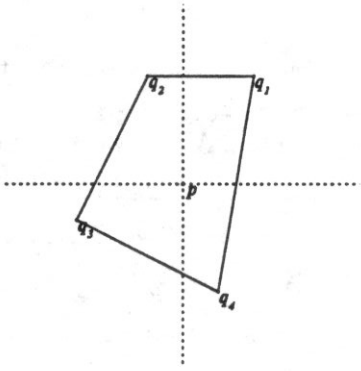
Maxima finding occurs naturally in a great many applications: statistics, economics and operations research to name a few [**PS**]. Another application is found in [**BCL**]. They discuss how [**BDMW**] use maxima to rate 329 cities using data found in [**BS2**]. The cities are assigned nine values describing characteristics such

as climate, health care, and housing costs. The cities can be therefore be thought of as 329 points in nine-dimensional space. [**BDMW**] use the dominance relation to rate the cities.

Another reason for being interested in maxima is that, like convex hulls, they describe the "boundary" of a point set. In fact, there is a very natural relationship between maxima and convex hulls. To describe it we need to generalize the definition of maxima. As defined in the beginning of this section the maxima of a set S are the points in S that have no points to their upper right. We will call these points the *maxima in the* (++) *orientation.* It is also possible to define maxima in the (+−) sense (Figure 4.1). These are elements of S that have no points in S to their upper left. Mathematically, p is a maximal point in the (−+) sense if there is no point $q \in S$ such that $q.x < p.x$ and $q.y > p.y$. Similarly, we can define a (−−) maximal point (no points to its lower left) and a (+−) one (no points to its lower right). With these definitions we can prove the following lemma:

Lemma 4.1: Let $S \subseteq \mathcal{R}^2$. If $p \in S$ is on the convex hull of S then p is a maxima of S in at least one of the four orientations.

Proof: Suppose $p \in S$ is not maximal in any of the four orientations. We can then find four points $q_1, q_2, q_3, q_4 \in S$ such that q_i is in the i-th quadrant of Cartesian axes that have p as an origin. This means that p is inside $CH(\{q_1, q_2, q_3, q_4\})$ and can not be on $CH(S)$.



§4.2.2 Distributions of Maxima

In §4.4 we will need to answer the question, "How many maxima are there in a set of n points?" It is of course possible that every point in the set is maximal, e.g. the n points lie on a line of negative slope. We are more interested in the probabilistic question, "Given n points I.I.D. from some distribution what is the distribution of the number of maxima?"

We start by addressing the question when the n points are I.I.D. $\mathbf{U[0,1]^2}$. Afterwards we will define the component independence property and examine the situation where the n points are chosen I.I.D. from any distribution over $\mathcal{R}^2$ with the component independence property. We will conclude by quoting the results of [**BKST**] on the expected number of maxima in point sets in higher dimensions.

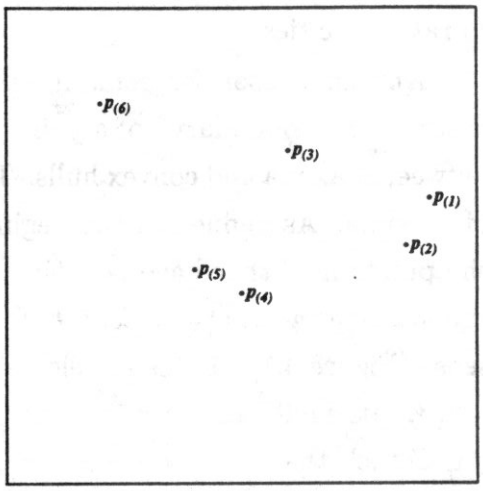


Figure 4.2: The points have previously been sorted by decreasing x-coordinate and relabeled. $p_{(1)}$, $p_{(3)}$, and $p_{(6)}$ are maximal so $Z_1 = Z_3 = Z_6 = 1$. The other three points are not maximal so $Z_{(2)} = Z_{(4)} = Z_{(5)} = 0$.

Lemma 4.2: Suppose that $p_1, \ldots, p_n$ are I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^2$. Let M_n be the number of maxima in the set containing $p_1, \ldots, p_n$,

$$M_n = |\{p_i : p_i \text{ maximum in } \{p_1, \ldots, p_n\}\}|.$$

Then M_n has the same distribution as $Z_1+Z_2+\cdots+Z_n$ where the Z_i are independent Bernoulli random variables with $\Pr(Z_i = 1) = 1/i = 1 - \Pr(Z_i = 0)$.

Proof: (See [**Kn1**], 1.2.10, for a different proof.) We start with labeled points $p_i = (x_i, y_i)$, $1 \leq i \leq n$. Since the x_i's are I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]$ we can assume that the x_i's are all unique (repeated values are a zero probability event). Sort the points by decreasing x coordinate and relabel them as $p_{(1)}, \ldots, p_{(n)}$ where

$$x_{(1)} > x_{(2)} > \cdots > x_{(n)}. \tag{4.1}$$

Thus, if the original points were $p_1 = (5,3)$, $p_2 = (3,7)$, and $p_3 = (18,6)$, then the relabeled points are $p_{(1)} = (18,6)$, $p_{(2)} = (5,3)$, and $p_{(3)} = (3,7)$.

Let Z_i be the indicator function of $p_{(i)}$ being a maximal point. Since $p_{(i)}$ is maximal if and only if its y-coordinate is greater than the y-coordinate of all points to its right (Figure 4.2) we can write

$$Z_i = \begin{cases} 1 & \forall j < i, \quad y_{(j)} < y_{(i)}. \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

By definition, $M_n = Z_1 + Z_2 + \cdots + Z_n$. To prove the lemma we have to show that, for all i, $\Pr(Z_i = 1) = 1/i$ and Z_i is independent of $Z_1, \ldots, Z_{i-1}$. The y_i's were originally chosen I.I.D $\mathbf{U}[\mathbf{0},\mathbf{1}]$ and are independent of the x_i's. Therefore,

after the sorting, the $y_{(i)}$'s can be assumed chosen I.I.D. from $\mathbf{U}[\mathbf{0},\mathbf{1}]$. This is a continuous distribution so we may assume that there are no repeated values among the $y_{(i)}$ (repeated values are a zero probability event). The $y_{(i)}$ are indistinguishable so

$$\Pr(Z_i = 1) = 1/i.$$

Independence follows from the fact that the value of Z_i is independent of all order information on $y_{(1)}, \ldots, y_{(i-1)}$. By this we mean that Z_i is independent of any knowledge of the relative order among $y_{(1)}, \ldots, y_{(i-1)}$:

$$\Pr(Z_i = 1 \,|\, y_{(\sigma(1))} > y_{(\sigma(2))} > \cdots > y_{(\sigma(i-1))}) = \Pr(Z_i = 1)$$

where σ is any permutation of $1, \ldots, i-1$. The only information that is contained in $Z_1, \ldots, Z_{i-1}$ is order information, so Z_i must be independent of them.

Next, we will extend the lemma to points that are I.I.D. from any distribution that has the *component independence property*. We say that a distribution over $\mathcal{R}^2$ has the component independence (CI) property if the x and y coordinates are chosen independently from continuous distributions. The $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$ distribution has the CI property. CI distributions can be much more general: The x coordinate might be chosen from a normal $N(0,1)$ distribution and the y coordinate from a Cauchy distribution. As long as the x and y coordinates are chosen from independent distributions and those distributions are both continuous we say that the combined distribution has the CI property. We can now generalize Lemma 4.1:

Lemma 4.3: Suppose that $p_1, \ldots, p_n$ are chosen I.I.D. from a distribution in $\mathcal{R}^2$ with the CI property. Let M_n be the number of maxima in the set containing $p_1, \ldots, p_n$,

$$M_n = |\{p_i : p_i \text{ maximum in } \{p_1, \ldots, p_n\}\}|.$$

Then M_n has the same distribution as $Z_1 + Z_2 + \cdots + Z_n$ where the Z_i are independent Bernoulli random variables with $\Pr(Z_i = 1) = 1/i = 1 - \Pr(Z_i = 0)$.

Proof: The proof is essentially the same as the proof of Lemma 4.3. Looking back at Lemma 4.3 we see that the only properties of $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$ used in its proof were that the x and y coordinates are independent and that both of them are drawn from continuous distributions. Since these two properties exactly define the CI property the proof follows.

Corollary 4.4: Suppose $p_1, \ldots, p_n$ are chosen I.I.D. from a distribution with the CI property. Let M_n be defined as above. Then

$$\mathrm{E}(M_n) = \sum_{1 \le i \le n} 1/i = H_n.$$

Proof: Apply Lemma 4.3 and take expectations.

$$\mathrm{E}(M_n) = \sum_{1 \le i \le n} \mathrm{E}(Z_i) = \sum_{1 \le i \le n} 1/i = H_n.$$

With very little extra effort we can also prove the following lemma bounding the expected number of points on a convex hull.

Corollary 4.5: Suppose $p_1, \ldots, p_n$ are chosen I.I.D. from a distribution with the CI property. Let C be the number of points on the convex hull of $p_1, \ldots, p_n$. Then $\mathrm{E}(C) \le 4H_n$.

Proof: Lemma 4.3 and Corollary 4.4 were proven for maxima defined in the (++) orientation but are also true for maxima defined in the (−+), (−−), and (+−) orientations. Let M_{++}, M_{-+}, M_{--}, and M_{+-} be the number of p_i that are maximal in the respective orientations. Lemma 4.1 tells us that $C \le M_{++} + M_{-+} + M_{--} + M_{+-}$ and the result follows by taking expectations.

We conclude this section by mentioning a result about maxima in higher dimensions. A distribution over $\mathcal{R}^d$ is said to have the CI property if the d coordinates are chosen from d independent one dimensional distributions, each of which is continuous. Bentley, Kung, Schkolnick, and Thompson [**BKST**] prove the following extension of Corollary 4.4:

Lemma 4.6: If $p_1, \ldots, p_n$ are chosen I.I.D. from a distribution in $\mathcal{R}^d$ with the CI property then the expected number of maxima is $O(\lg^{d-1} n)$.

Later, in §4.3, we will present a long quote from [**BCL**]. Part of that quote is a graph (Figure 4.4) giving the number of maxima found in point sets chosen I.I.D. from $\mathbf{U}[0,1]^d$, $d = 2, 3, 4, 5$.

§4.3 The Move-To-Front Algorithm

In §4.2 we defined the maxima of a point set and mentioned some applications. In this section we discuss algorithms for finding the maxima.

Program 4.1: This is [**BCL**]'s pseudo-code for the MTF maxima finding algorithm. The queue of current maxima is stored in the array $Max[\,]$. $TopMax$ is the current number of points in the queue. The input points are read in sequentially from another array: when the J'th point is read it is compared to the current points in the queue. If a current maximal point dominates the J'th point then that maximal point is moved to the front of the queue. If the J'th point dominates a current maximal point then that maximal point is thrown off the queue. If the J'th point isn't dominated by any maximal point in the queue then it is placed at the end of the queue. The code signals the special case of a duplicate point by setting $J := TopMax + 2$.

```
TopMax := 1
Max[TopMax] := 1
for I := 2 to N do
    J := 1
    while J<= TopMax do
        if pt Max[J] dominates pt I then
            move Max[J] to front of Max[1..J]
            J := TopMax + 2
        else if pt I dominates pt Max[J] then
            shift  Max[J+1..TopMax]
                to Max[J..TopMax-1]
            TopMax := TopMax - 1
        else if pt I equals pt Max[J] then
            J := TopMax + 2
        else // pts I, Max[J] incomparable
            J := J+1
    if J = TopMax + 1 then
        TopMax := TopMax + 1
        Max[TopMax] := I
```

First we need some background. It is possible to show that finding the maxima of a set of n points in two dimensions requires $\Omega(n \lg n)$ time in the comparison tree model of computation: this is done via a reduction to sorting [**PS**]. There is an algorithm [**PS**] for finding maxima that matches this lower bound; it requires only $O(n \lg n)$ comparisons. [**KLP**] give an algorithm that finds maxima in d-dimensions. It runs in $O(n \lg^{d-2} n + n \lg n)$ time and is extremely complicated to implement.

Deterministic running times are not our major concern here. As in the rest of this thesis, we are interested in simple algorithms with fast expected running times. In [**BCL**] Bentley, Clarkson, and Levine present an algorithm that runs in $O(n)$ expected time on a set of n points I.I.D. $\mathbf{U}[0,1]^d$. They then extend their algorithm so that it runs in $O(n)$ expected time even if the the points are chosen I.I.D. from an arbitrary distribution with the CI property.

After presenting this second algorithm they comment that it was "designed to be efficient for CI inputs and easy to analyze for that case but ... not necessarily robust for point sets from other distributions." They then continue "We will now study an algorithm that is easy to implement, very efficient for CI distributions,

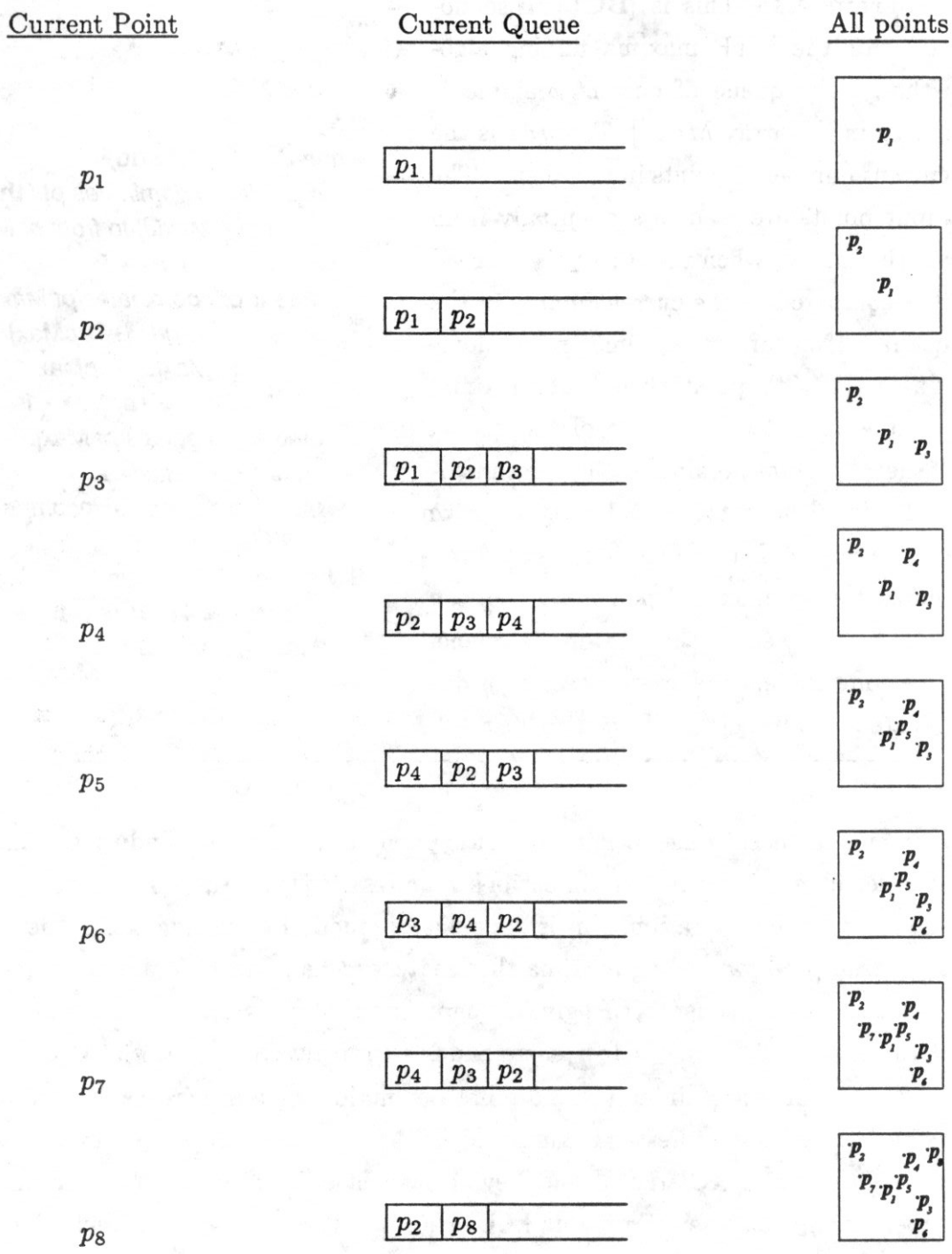


Figure 4.3: A worked example of the move to front maxima finding algorithm. It shows the status of the queue after comparing the current point against the current maxima.

and somewhat robust for other distributions." The algorithm referred to in the second quote is the Move-To-Front (MTF) algorithm. We present their pseudo-code implementing it in Program 4.1 and a worked example in Figure 4.3. The algorithm scans the points $p_1, \ldots, p_n$ in sequential order maintaining a queue T that contains the current maximal elements among the currently scanned points. The algorithm updates the queue by comparing p, the current point being scanned, to the maxima in the queue in left-to-right order.

If p dominates some point q in the queue then q is no longer maximal and is thrown away. If p is not dominated by any point in the queue then p is currently maximal and is placed at the end of the queue. Otherewise p is dominated by some maximal element, q, in the queue. This point q is moved to the front of the queue.

How fast is the MTF algorithm? Obviously the algorithm runs in $O(n^2)$ time since, when p_i is scanned, it is compared to at most $i-1$ points in the queue. It is not hard to construct a degenerate two-dimensional example where these $i-1$ comparisons are all made: the points $(p.x, p.y)$ are all on the line $p.x = -p.y$. As always we are more concerned with discovering the expected running time of the algorithm.

[**BCL**] observed that the total number of point comparisons performed by the MTF algorithm is bounded from above by $\sum_{i<n} M_i$ where M_i is the number of maximal elements in the set $p_1, \ldots, p_i$:

$$M_i = |\{p_i \,:\, p_i \text{ maximum in } \{p_1, \ldots, p_i\}\}|.$$

If the n points are chosen I.I.D. from any CI distribution in d-dimensions then Lemma (4.6) tells us that $\mathtt{E}(\sum M_i) = \sum O(\lg^{d-1} i) = O(n \lg^{d-1} n)$.

The next few paragraphs are an extended quote from [**BCL**] describing their experiments on the MTF algorithm and their subsequent conjecture about its actual running time. Figures 1, 2, and 3 referred to in the quote are Figures 4.4, 4.5, and 4.6 in this paper. Algorithm M3 is the MTF algorithm and "*the sequence T* " is the queue in which the current maxima are kept.

We will turn now to an analysis of the run time of Algorithm M3. The expected size of the sequence T after M elements have been examined is precisely the expected number of maxima in a set of M points, which is monotone increasing in M. The expected size of T is therefore $O(\log^{K-1} N)$, *and the expected total cost of the N searches is* $O(N \log^{K-1} N)$. *To understand the constants in the big-oh, a series of experiments investigated the average number of maxima. Ten point sets were*

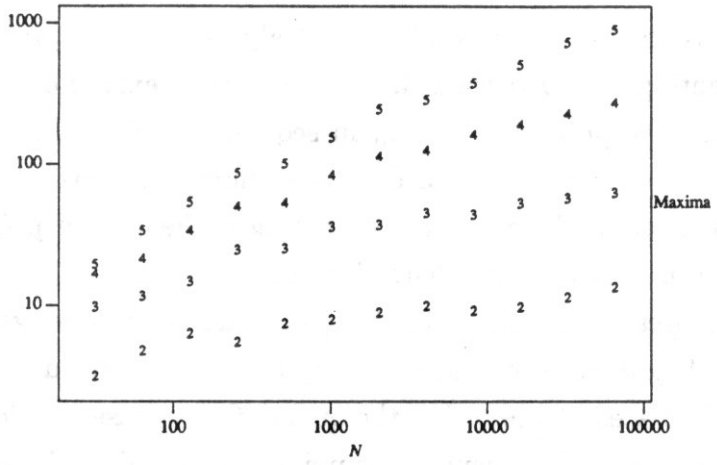


Figure 4.4: The number of maxima.

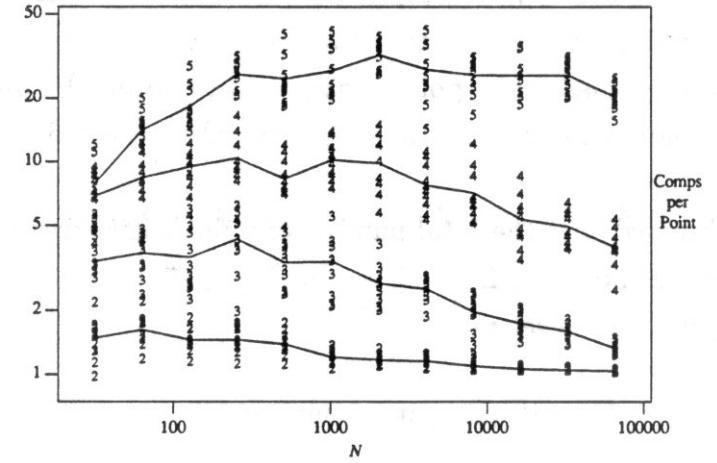


Figure 4.5: Comparisons per point.

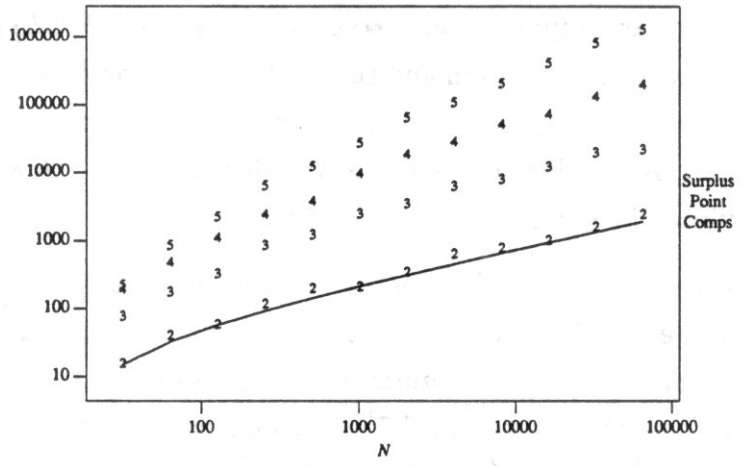


Figure 4.6: Surplus point Comparisons.

generated for N at each power of two from 32 to 65536 and for each K from 2 to 5. The N points were uniformly distributed over the K-dimensional hypercube. Figure 1 plots the number of maxima observed (averaged over the ten point sets), using the dimension as the plot symbol.

The distribution of the maxima and the move-to-front heuristic might, however, make the algorithm substantially faster than the $O(N \log^{K-1} N)$ expected bound. We examined the running time in the same series of experiments shown in Figure 1. Figure 2 shows the average number of point comparisons used, divided by N. The plot symbol is the dimension; the lines denote mean values.

The average number of point comparisons per point is substantially less than the number of maxima in the set. At $N = 65536$, the mean ratios were 1.037, 1.331, 3.998, and 20.49, for $D = 2$, 3, 4, and 5, while the mean number of maxima were 13.2, 61.6, 268.9, and 881.6. When we first ran the experiments, we expected the ratios to increase with N (though we hoped not quite as quickly as the $O(log^{K-1}N)$ bound). When we examined Figure 2, however, we conjectured that the number of comparisons per point is approaching a different constant for each dimension K.

In Figure 2, though, each curve appears to grow to a peak and then to decrease again. After running a handful of large experiments (N up to 1,000,000), we conjectured that the ratio approaches 1 for all values of K. To test this hypothesis, we re-expressed the data in Figure 2 by plotting the number of point comparisons beyond N, which we shall refer to as the surplus comparisons.† The means of surplus comparisons are shown Figure 3.

A weighted least squares regression shows that for $K = 2$ the number of surplus comparisons grows as $7.6N^{.515} - 27.7$; the regression yields well-behaved residuals (apparently normally distributed). We have therefore plotted the function $7.6\sqrt{N} - 28$ on the Figure 3, which is a close fit to the experimental data. Algorithm M1 helps to explain why Algorithm M3 might behave this way: a single point very near $(1, 1)$

---

† *Our first attempt to examine the surplus comparisons resulted in a warning from the regression/plotting package that it had tried to take the logarithm of a negative number. Investigation showed that on one set of $N = 32$ points in $K = 2$-space, Algorithm M3 found 2 maxima using 31 point comparisons. We first feared that this was evidence of a program bug, but examining the input point set showed us how it happened. We leave the explanation as a puzzle for the reader. (Hint: the probability of Algorithm M3 finding the maxima of a planar set in exactly $N - 1$ point comparisons is at least $1/N^2$.)*

tends to stay near the front of the sequence, and dominates all but roughly $\sqrt{N}$ *of the other inputs. Regressions for the other surplus values yield* $37.1N^{.594} - 237$ *for* $K = 3$, $31.8N^{.799} - 333$ *for* $K = 4$, *and* $32.1N^{.966} - 733$ *for* $K = 5$ *(though the residuals are not particularly well behaved). This data and the heuristic arguments support the following:*

Conjecture 2-6. *Algorithm M3 finds the maxima of* N *points chosen from a* K*-dimensional CI distribution in* $O(N)$ *time, using* $N + o(N)$ *point comparisons, for any fixed dimension* K.

We suspect that one might even be able to tighten the number of surplus comparisons to near $O(N^{1-1/K})$.

In the next section we shall address their conjecture when $K = 2$. We shall show that, for n points chosen I.I.D. from a distribution over $\mathcal{R}^2$ that has the CI property, the number of comparisons point comparisons made by the MTF algorithm is $n + O(n^{6/7} \lg^5 n)$.

§4.4 Analysis of the Algorithm

§4.4.1 Plan and Definitions

We first analyze the MFT algorithm's performance under the assumption that it is run on $p_1, \ldots, p_n$, n points I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$. Afterwards we show how to extend the analysis to n points chosen I.I.D. from any CI distribution. Our analysis bounds the number of surplus comparisons performed by the MTF algorithm. The algorithm must perform at least one point comparison for each input point p_i, $i > 1$ – the comparison that compares p_i to the first element in the queue. Surplus comparisons are the extra point comparisons the algorithm performs when it compares p_i to elements deeper in the queue; if the algorithm performs a total of C point comparisons then it performs $C - n + 1$ surplus comparisons. Surplus comparisons are a logical measure of the cost of the algorithm since any comparison-based maxima finding algorithm must perform at least one point comparison for each input point.

The analysis is divided into two parts. The first part calculates a limit on the number of surplus comparisons where the limit is a function of geometric random variables. The second part derives probabilistic bounds on the geometric random variables. Inserting these bounds into the first part yields a probabilistic bound

$$\mathcal{C} = [1-\alpha, 1] \times [1-\alpha, 1]$$
$$\mathcal{T}_1 = [1-\alpha, 1] \times [1-\alpha-\beta, 1-\alpha]$$
$$\mathcal{T}_2 = [1-\alpha-\beta, 1-\alpha] \times [1-\alpha, 1]$$
$$\mathcal{D} = [1-\alpha-\beta, 1-\alpha-\beta] \times [1-\alpha, 1-\alpha]$$
$$\mathcal{A}_1 = [1-\alpha, 1] \times [0, 1-\alpha-\beta]$$
$$\mathcal{A}_2 = [0, 1-\alpha-\beta] \times [1-\alpha, 1]$$
$$\mathcal{W}_1 = [1-\alpha-\beta, 1-\alpha] \times [0, 1-\alpha-\beta]$$
$$\mathcal{W}_2 = [0, 1-\alpha-\beta] \times [1-\alpha-\beta, 1-\alpha]$$
$$\mathcal{V} = [0, 0] \times [1-\alpha-\beta, 1-\alpha-\beta]$$

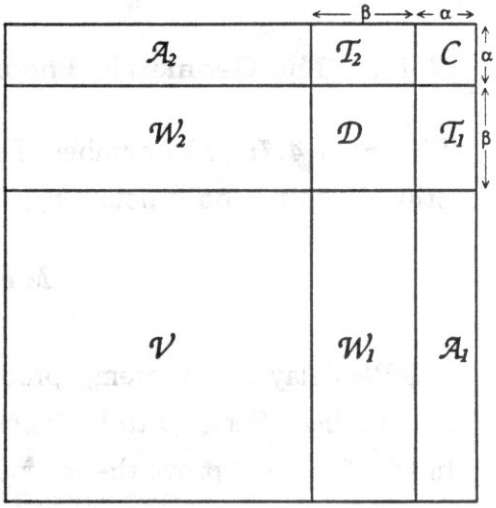


Figure 4.7: A partition of the square into rectangular regions that are functions of α and β.

on the number of surplus comparisons. We also define random variables that are functions of the distribution of the points in the rectangles.

As mentioned above we first prove the theorem for points in the unit square. Let α and β be positive real numbers such that $\alpha + \beta < 1$. We divide the square into rectangles as illustrated in Figure 4.7.

Random Variables:

$$M_i = |\{j : j \leq i, p_j \text{ maximal in } p_1, \ldots, p_i\}| \tag{4.3a}$$
$$M = \max_{1 \leq i \leq n} M_i \tag{4.3b}$$
$$A = |\{p_i : p_i \in \mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2\}| \tag{4.3c}$$
$$W = |\{p_i : p_i \in \mathcal{W}_1 \cup \mathcal{D} \cup \mathcal{W}_2\}| \tag{4.3d}$$
$$F_{\mathcal{C}} = \min_{1 \leq i \leq n} \{i : p_i \in \mathcal{C}\} \tag{4.3e}$$
$$G_{\mathcal{D}} = \max_{1 \leq i \leq n} \min_{k \geq 0} \{k : p_{i+k} \in \mathcal{D} \text{ or } i + k = n\} \tag{4.3f}$$

M_i is the queue size after reading the i'th point. M is the maximal queue size encountered while executing the MTF algorithm. A and W are the number of points found in their respective regions. $F_{\mathcal{C}}$ is the index of the first point in $\mathcal{C}$. $G_{\mathcal{D}}$ is the maximum number of points, k, that we must scan to the right of any p_i to be sure of reaching a point in $\mathcal{D}$, i.e. the maximum gap between any two points in $\mathcal{D}$. As an example, suppose that $n = 10$, $p_2 \in \mathcal{D}$, $p_7 \in \mathcal{D}$, and the remaining eight points are not in $\mathcal{D}$: $G_{\mathcal{D}} = 7 - 2 = 5$.

§4.4.2 The Geometric Theorem

Theorem 4.7: The number of surplus comparisons performed by the MTF algorithm when run on n points $p_1, \ldots, p_n \in [0,1]^2$ is less than

$$MF_{\mathcal{C}} + MAG_{\mathcal{D}} + MW. \tag{4.4}$$

We delay the theorem's proof for the moment and instead concentrate on how it is applied. Set α, β to be functions of n : $\alpha = n^{-\alpha'}$, $\beta = n^{-\beta'}$ where $\alpha', \beta' > 0$. In §4.4.3 we will prove the following probabilistic facts:

$$\Pr\left(M > \lg^3 n\right) < n^{-\Omega(\lg n)} \tag{4.5a}$$

$$\Pr\left(A > 6n^{1-\alpha'}\right) < 2^{1-n} \tag{4.5b}$$

$$\Pr\left(W > 6n^{1-\beta'}\right) < 2^{1-n} \tag{4.5c}$$

$$\Pr\left(F_{\mathcal{C}} > n^{2\alpha'} \lg^2 n\right) < n^{-\Omega(\lg n)} \tag{4.5d}$$

$$\Pr\left(G_{\mathcal{D}} > n^{2\beta'} \lg^2 n\right) < n^{-\Omega(\lg n)} \tag{4.5e}$$

Inserting these back into Theorem 4.7 we see that the MTF algorithm performs

$$O\left(\max\left(n^{2\alpha'} \lg^5 n,\, n^{1-\alpha'+2\beta'} \lg^5 n,\, n^{1-\beta'} \lg^3 n\right)\right) \tag{4.6}$$

surplus comparisons with probability $1 - n^{-\Omega(\lg n)}$. We minimize this expression by setting

$$n^{2\alpha'} = n^{1-\alpha'+2\beta'} = n^{1-\beta'}. \tag{4.7}$$

Solving gives $\alpha' = 3/7$, $\beta' = 1/7$ and (4.7) becomes $O(n^{6/7} \lg^5 n)$. This proves our main theorem:

Theorem 4.8: The number of surplus comparisons performed by the MTF algorithm when run on n points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$ is $O\left(n^{6/7} \lg^5 n\right)$ with probability $1 - n^{-\Omega(\lg n)}$.

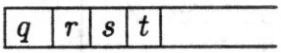
Since the MTF algorithm never performs more than n^2 comparisons we have also proven the following corollary:

Corollary 4.9: The expected number of point comparisons performed by the MTF algorithm when run on n points chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$ is $n + O\left(n^{6/7} \lg^5 n\right)$.

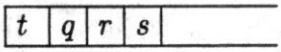
Proof of Theorem 4.7: The heart of the theorem is a technical lemma. It bounds the number of surplus comparisons performed while checking subsequences of points

that conform to a very restrictive type. We first prove the lemma and afterwards show how to partition $p_1, \ldots, p_n$ into restricted subintervals of the desired type. Taken together the two parts prove the theorem.

First we introduce some new notation. By *scanning* a point or a sequence of points we mean comparing the point(s) to the maxima in the queue using the MTF algorithm. The number of surplus comparisons performed while scanning a point, p, is the number of points in the queue that p is compared to, minus one. For example, suppose the queue holds



and we are currently scanning the point p to determine if it is maximal. If t is the first maximal point in the queue that dominates p then 3 surplus comparisons are performed while scanning p and t is moved to the front. The new queue is



.

The maximum number of surplus comparisons that can be performed for any point is $M - 1$ since, by definition, M is the maximal queue size.

Lemma 4.10: Suppose that at some point during the MTF algorithm's execution the following conditions are fulfilled:

(i) There is some maximal point $c \in \mathcal{C}$ in the queue.

(ii) The current maximal point at the front of the queue is $t \in \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2$.

(iii) The next m points to be scanned, $q_1, \ldots, q_m$, are all in $\mathcal{V} \cup \mathcal{W}_1 \cup \mathcal{D} \cup \mathcal{W}_2$.

Then the total number of surplus comparisons performed while scanning the q's can be bounded depending on the location of t.

(I) $t \in \mathcal{C}$: No surplus comparisons are performed.

(II) $t \in \mathcal{T}_1$: The number of surplus comparisons is less than $M \cdot W_m$ where

$$W_j = |\{q_i : q_i \in \mathcal{W}_2 \cup \mathcal{D}, 1 \leq i \leq j\}|.$$

W_j is the number of points in $q_1, \ldots, q_j$ that are in $\mathcal{W}_2 \cup \mathcal{D}$.

(III) $t \in \mathcal{T}_2$: The number of surplus comparisons is less than $M \cdot W'_m$ where

$$W'_j = |\{q_i : q_i \in \mathcal{W}_1 \cup \mathcal{D}, 1 \leq i \leq j\}|.$$

W'_j is the number of points in $q_1, \ldots, q_j$ that are in $\mathcal{W}_1 \cup \mathcal{D}$.

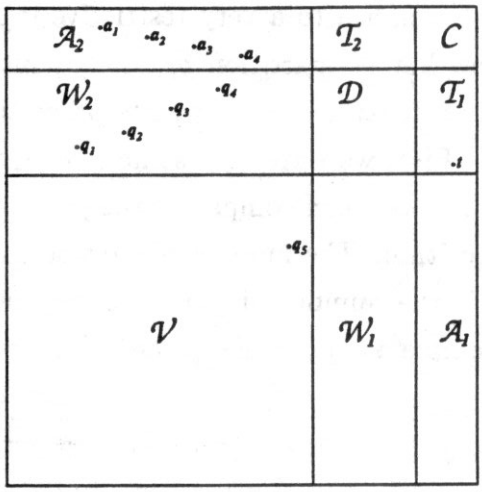
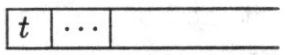


Figure 4.8: This illustrates how to force a point in $\mathcal{V}$ to participate in a large number of surplus comparisons. We start with $t \in \mathcal{T}_1$ at the front of the queue. After scanning q_1, q_2, q_3, and q_4, the front four maxima in the queue are a_1, a_2, a_3, and a_4 forcing $q_5 \in \mathcal{V}$ to participate in four surplus comparisons.

Proof: The main purpose of condition (i) is to guarantee that no maximal points are taken off or added to the queue while scanning the q_j : the only operations that transform the queue are of the move to front type. This is because c dominates the q_j, precluding any of them from being maximal. Furthermore no maximum t in the queue can be dominated by any of the q_j since this would imply that t is dominated by c leading to a contradiction. This forces all of the maxima currently in the queue to be in $\mathcal{A}_1 \cup \mathcal{T}_1 \cup C \cup \mathcal{T}_2 \cup \mathcal{A}_2$.

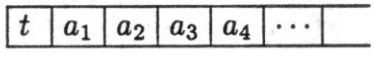
(I) $t \in C$: Before starting the scan the queue looks like



, $t \in C$.

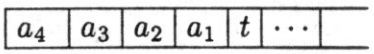
Because of condition (iii) this t dominates all of the points $q_1, \ldots, q_m \in \mathcal{V}$ so no surplus comparisons are ever performed.

(II) $t \in \mathcal{T}_1$: In Figure 4.8 we have drawn an example where $t \in \mathcal{T}$ dominates all of the q_i in $\mathcal{V} \cup \mathcal{W}_1$. The lemma's result makes it seem that we are going to prove that points in $\mathcal{V} \cup \mathcal{W}_1$ are always dominated by the first point in the queue and only points in $\mathcal{D} \cup \mathcal{W}_2$ can cause surplus comparisons. This isn't true. In Figure 4.8 we illustrate a counterexample. The queue starts off as



.

After scanning the points q_1, q_2, q_3, q_4 the queue looks like



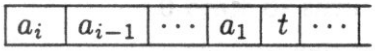
.

The next point scanned, q_5, is in $\mathcal{V}$ but participates in four surplus comparisons. It is not hard to see that we can generalize this example to force a point in $\mathcal{V}$ to participate in arbitrarily many surplus comparisons.

There are two properties in the above example that we would like to stress. The first is that the a_i-s moved to the front of the queue are sorted by decreasing x-coordinate. The second is that the number of surplus comparisons performed while scanning q_5 is not arbitrary. Rather, it is the number of points in $\mathcal{W}_1 \cup \mathcal{D}$ that were scanned since the last time t was at the front of the queue.

We will show that these two properties are not particular to our specific example. This fact will let us take all of the surplus comparisons performed while scanning $q_1, \ldots, q_m$ and *amortize* them over the W_m occurrences of points in $\mathcal{W}_2 \cup \mathcal{D}$. By amortization we mean distributing the total cost of a long sequence of operations over the number of appearances of certain specific occurrences (see [**Ta**] for more on the theory of amortization). In order to proceed with our proof we will need one more definition:

`Definition:` A queue is in *state* i, $i = 0, 1, 2 \ldots$, if it has the form



$$, \qquad \begin{array}{l} t \in \mathcal{T}_1 \cup \mathcal{C} \\ a_j \in \mathcal{A}_2 \cup \mathcal{T}_2, \quad 1 \le j \le i \\ a_i.x > a_{i-1}.x > \cdots > a_1.x. \end{array} \tag{4.8}$$

The first i maxima in the queue are in $\mathcal{A}_2 \cup \mathcal{T}_2$ and are sorted by decreasing x-coordinate. The $i+1$'st maximal point is in $\mathcal{T}_1 \cup \mathcal{C}$. The remaining maxima can be anywhere. Figure 4.9 illustrates the first four queue states. In the example given at the beginning of this proof the queue starts in state 0: while scanning q_1, q_2, q_3, q_4 it goes into states $1, 2, 3, 4$, and finally returns to state 0 after scanning q_5. In this new terminology, condition (ii) of the lemma states that the queue is in state 0 before q_1 is scanned.

Suppose now that the queue is in state i. We will show that after scanning a new point, q, the queue will be in one of only three possible states: 0, i, $i+1$. Therefore the numbered states defined in (4.8) contain all of the possible queue configurations that can occur while scanning $q_1, \ldots, q_m$. While showing this we will also count how many surplus comparisons are performed during the state transitions. Table 4.1 summarizes the possible outcomes and associated costs.

When $q \in \mathcal{V} \cup \mathcal{W}_1$ there are two possibilities. If the maximal point at the front of the queue dominates q then we perform no surplus comparisons and remain in

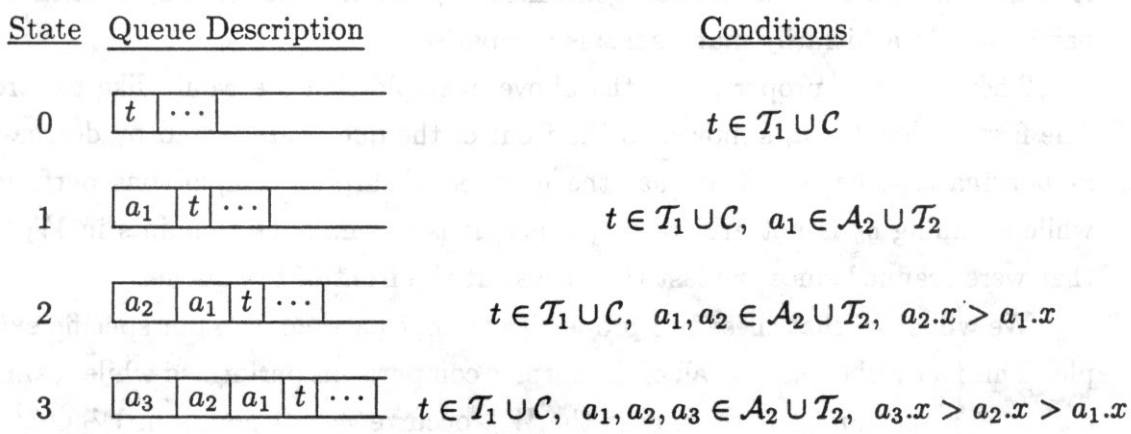


Figure 4.9: The first four queue states and their associated conditions.

Outcome	Location of q	Transition	Surplus Comps.
(a)	$q \in \mathcal{V} \cup \mathcal{W}_1$	queue remains in state i	0
(b)	"	queue returns to state 0	i
(c)	$q \in \mathcal{W}_2 \cup \mathcal{D}$	queue remains in state i	0
(d)	"	queue goes to state $i+1$	$M-1$
(e)	"	queue returns to state 0	$M-1$

Table 4.1: Possible queue state transition outcomes and the maximum number of surplus comparisons they can cost.

state i : this is outcome (a). Otherwise, q is not dominated by a_i. Condition (iii) of the lemma implies that $a_j.y > q.y$ for all $1 \leq j \leq i$. It follows from the definitions of domination and the queue states that $q.x > a_i.x > a_{i-1}.x > \cdots > a_1.x$. Therefore t is the first maximal point in the queue that dominates q and we performed i surplus comparisons discovering this fact: outcome (b).

When $q \in \mathcal{W}_2 \cup \mathcal{D}$ there are also two possibilities. Again, if the maximal point at the front of the queue dominates q, then no surplus comparisons are performed and we remain in state i : outcome (c). Otherwise, again as before, we find that $q.x > a_i.x > a_{i-1}.x > \cdots > a_1.x$ and none of the a_j dominate q. Therefore the first maximal point that dominates q, the one that will be moved to the front of the queue, is not one of the a_j. Since we do not know where this maximal point

comes from we can only bound the number of surplus comparisons performed by $M-1$. If this new maxima is in $\mathcal{A}_2 \cup \mathcal{T}_2$ then call it a_{i+1}. From the definition of domination $a_{i+1}.x > q.x > a_i.x$: outcome (d). If it isn't in $\mathcal{A}_2 \cup \mathcal{T}_2$ then it can't be in $\mathcal{A}_1$ because a point in $\mathcal{A}_1$ dominates no point in $\mathcal{W}_2 \cup \mathcal{D}$. Therefore the new maximal point must be in $\mathcal{T}_1 \cup \mathcal{C}$: outcome (e).

We now use these facts about transitions to finish the proof of this part of the lemma. The general idea is that the only expensive outcomes are of type (b), (c), and (d), and the type (b) outcomes can be charged to the type (d) outcomes that precede them. The proof will be by induction on m where the base case, $m = 0$, is obviously true.

Assume now that the lemma is true for all valid sets of less than than m points. We know that before scanning q_1 the queue is in state 0.

Suppose first, that while scanning q_1 the queue enters state 1 and subsequently, while scanning $q_2, \ldots, q_m$, it never returns to state 0. Referring to Table 4.1 we see that all surplus comparisons that occur must result from outcomes of type (d). These occur only while scanning points in $q_1 \in \mathcal{W}_2 \cup \mathcal{D}$ and there are W_m of these. Therefore at most $(M-1)W_m$ surplus comparisons are performed and we have proven the lemma.

Otherwise the the queue does return to state 0 while scanning $q_1, \ldots, q_m$. Let

$$m' = \min_{i \geq 1}\{i : \text{queue returns to state 0 after scanning } q_i\}.$$

By definition the queue never returns to state 0 while scanning $q_1, \ldots, q_{m'-1}$ so the analysis performed in the previous paragraph tells us that at most $(M-1)W_{m'-1}$ surplus comparisons are performed while scanning $q_1, \ldots, q_{m'-1}$. Furthermore, we know that after scanning $q_{m'-1}$ the queue is in state i where $i \leq W_{m'-1}$ because only transitions leading to outcome (d) can increase the state number and there are at most $W_{m'-1}$ of these.

It remains to calculate how many surplus comparisons are performed while scanning $q_{m'}$, a step which we know returns us to state 0. If $q_{m'} \in \mathcal{V} \cup \mathcal{W}_1$ then we know that we see outcome (b) and $W_{m'} = W_{m'-1}$. Therefore the maximum number of surplus comparisons performed while scanning $q_1, \ldots, q_{m'}$ is less than

$$(M-1)W_{m'-1} + i \leq (M-1)W_{m'} + W_{m'} = MW_{m'}.$$

Otherwise, $q \in \mathcal{W}_2 \cup \mathcal{D}$ and we see outcome (e), $W_{m'} = W_{m'-1} + 1$, and the maximum number of surplus comparisons performed while scanning $q_1, \ldots, q_{m'}$ is

$$(M-1)W_{m'-1} + M - 1 = (M-1)W_{m'}.$$

In both cases we have proven that at most $MW_{m'}$ surplus comparisons are performed while scanning $q_1, \ldots, q_{m'}$. Furthermore, after scanning $q_{m'}$ the queue is back in state 0 so inductively applying the lemma yields that fewer than $M(W_m - W_{m'})$ surplus comparisons are performed while scanning $q_{m'+1}, \ldots, q_m$. Combining these two facts proves part (II).

(III) $t \in \mathcal{T}_2$: The proof for this case follows that of part (II)'s.

Q.E.D. (Lemma 4.10)

(Continuation of the proof of Theorem 4.7:) Our next step is to partition our original point set $p_1, \ldots, p_n$ into parts, many of which are in the form required by the lemma. First we divide $p_1, \ldots, p_n$ into stages which we define as follows: Starting with $i_1 = F_C$, let i_j be the j'th point found in $\mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2$ while scanning from left to right:

$$
\begin{aligned}
i_0 &= 1 \\
i_1 &= F_C \\
i_2 &= \min_{i_2 > i_1} \{i : p_i \in \mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2\} \\
i_3 &= \min_{i_3 > i_2} \{i : p_i \in \mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2\} \\
i_4 &= \min_{i_4 > i_3} \{i : p_i \in \mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2\} \\
&\vdots
\end{aligned}
$$

Now, we define the *j'th stage* to be the set of points[1]

$$p_{i_j}, p_{i_j+1}, \cdots p_{i_{j+1}-1}.$$

These stages are almost in the form required by the Lemma 4.10. After scanning the first stage we know that there will always be a maximal point $c \in \mathcal{C}$ in the queue and thus condition (i) will be satisfied while scanning all subsequent stages. Furthermore, every point except for the first in a stage is in $\mathcal{V} \cup \mathcal{W}_1 \cup \mathcal{D} \cup \mathcal{W}_2$ and so satisfies condition (iii). To be able to apply the lemma we must also satisfy condition (ii). We can not satisfy this condition for the entire stage but, fortunately, we will be able to satisfy it for a significant portion of the stage.

For each stage $j > 0$ let t_j be the first point in $\mathcal{D}$ in the stage:

$$t_j = \min_{i \geq i_j} \{i : p_i \in \mathcal{D}\}.$$

[1] The final stage is a special case. If p_{i_f} is the point in $\mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2$ with the highest index then we say that the i_f'th stage consists of the points $p_{i_f}, \ldots, p_n$.

Point location	Surplus Comps.
Stage 0	$MF_{\mathcal{C}}$
Initial parts of all stages	$MAG_{\mathcal{D}}$
Terminal parts of all stages	MW

Table 4.2: Maximum number of surplus comparisons performed on all points that appear in the given location.

Next, split the j'th stage into two parts: an initial and a terminal.

$$\text{initial part} = p_{i_j}, p_{i_j+1}, \cdots, p_{t_j}$$
$$\text{terminal part} = p_{t_{j+1}}, p_{t_{j+2}}, \cdots, p_{i_{j+1}-1}.$$

As we shall soon see, scanning p_{t_j} forces a maximal point in $\mathcal{T}_\infty \cup \mathcal{C} \cup \mathcal{T}_2$ to the front of the queue (condition (ii)). This fact will let us employ the lemma to bound the number of surplus comparisons done while scanning each stage.

The 0'th stage contains $F_{\mathcal{C}}-1$ points, so at most $(M-1)F_{\mathcal{C}}$ surplus comparisons are performed on it. After scanning the 0'th stage there is some maximal point in $\mathcal{C}$ in the queue and therefore condition (i) of the lemma is always true during the remaining stages.

How many comparisons are performed while scanning the remaining stages? Since each stage starts with a point in $\mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2$ there are at most A stages. From definition (4.8) we know that, no matter where we start, after scanning $G_{\mathcal{D}}$ points we will encounter some point in $\mathcal{D}$. Therefore, for every j the size of the initial part of the j'th stage, $t_j - i_j + 1$, is less than $G_{\mathcal{D}}$. Thus the total number of surplus comparisons performed while scanning the initial parts of *all* of the stages taken together is less than $(M-1)AG_{\mathcal{D}}$.

We would like to employ Lemma 4.10 to analyze the terminal parts of the stages. To do this we must prove that conditions (i), (ii), and (iii) all apply. We have already seen that (i) is true. The definition of a stage requires that all of its points (except the first) are in $\mathcal{W}_1 \cup \mathcal{D} \cup \mathcal{W}_2 \cup \mathcal{V}$ so (iii) is true.. To show (ii) we examine what happens when $p_{t_j} \in \mathcal{D}$ is scanned. We know that p_{t_j} can't be a maximal point because it is dominated by c. Therefore a maximal point that dominates p_{t_j} is moved to the front of the queue. Since any such point must be in $\mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2$ condition (ii) is true immediately before scanning p_{t_j+1}.

Applying the lemma we find that the number of surplus comparisons performed while scanning the points in the terminal part of the j'th stage is less than

$$M \cdot |\{p_i : t_j < i < p_{j+1}, p_i \in \mathcal{W}_1 \cup \mathcal{D} \cup \mathcal{W}_2\}|.$$

Summing over all of the stages proves that the total number of surplus comparisons performed while scanning the terminal parts of all of the stages is less than MW.

Combining the bounds for the 0'th stage, the initial parts of all stages, and the terminal parts (Table 4.2) proves the theorem.

Q.E.D. (Theorem 4.7)

§4.4.3 Probabilistic Facts

It is left for us to prove the various probabilistic facts that, inserted into Theorem 4.7, let us prove Theorem 4.8. These facts were

$$\Pr\left(M > \lg^3 n\right) < n^{-\Omega(\lg n)} \tag{4.5a}$$

$$\Pr\left(A > 6n^{1-\alpha'}\right) < 2^{1-n} \tag{4.5b}$$

$$\Pr\left(W > 6n^{1-\beta'}\right) < 2^{1-n} \tag{4.5c}$$

$$\Pr\left(F_C > n^{2\alpha'} \lg^2 n\right) < n^{-\Omega(\lg n)} \tag{4.5d}$$

$$\Pr\left(G_{\mathcal{D}} > n^{2\beta'} \lg^2 n\right) < n^{-\Omega(\lg n)} \tag{4.5e}$$

Remember that we assume that the n points are chosen I.I.D. $\mathbf{U}[0,1]^2$. The proofs of the above facts follow from standard probabilistic manipulations. (4.5a) is the hardest to prove. It states that, with high probability the maximum queue size is less than $\lg^3 n$. The proof of (4.5a) will follow from the next lemma which bounds the queue size at any given time.

Lemma 4.11: Given n points, $p_1, \ldots, p_n$, chosen I.I.D. $\mathbf{U}[\mathbf{0},\mathbf{1}]^{\mathbf{2}}$, let M_i be the queue size after the MTF algorithm has scanned p_i. Then, for all i less than or equal to n,

$$\Pr(M_i > \lg^3 n) \leq n^{-\Omega(\lg n)}.$$

Proof: Lemma (4.3) permits us to write

$$M_i = Z_1 + Z_2 + \cdots + Z_i$$

where the Z_j's are independently distributed Bernoulli random variable with $\Pr(Z_j = 1) = 1/j$. Let $l = \lfloor \lg n \rfloor$. We split the Z_j's up into $l+1$ sets of geometrically in-

creasing size by defining new random variables Y_k :

$$\begin{aligned}
Y_0 &= Z_1 \\
Y_1 &= Z_2 + Z_3 \\
Y_2 &= Z_4 + Z_5 + Z_6 + Z_7 \\
&\vdots \quad \vdots \\
Y_k &= Z_{2^k} + Z_{2^k+1} + \cdots + Z_{2^{k+1}-1} \\
&\vdots \quad \vdots \\
Y_l &= Z_{2^l} + Z_{2^l+1} + \cdots + Z_i.
\end{aligned}$$

By definition $M_i = \sum_{0 \le k \le l} Y_j$, so to prove the lemma it is enough to prove that, for each k,

$$\Pr(Y_k \ge \lg^2 n) \le n^{-\Omega(\lg n)}.$$

If $k < 2 \lg \lg n$ then, deterministically, $Y_k \le 2^k < \lg^2 n$. We may therefore assume that $k > 2 \lg \lg n$. We know[2] that

$$Y_k = Z_{2^k} + Z_{2^k+1} + \cdots + Z_{2^{k+1}-1}.$$

For any $j \ge 2^k$ it is true that $\Pr(Z_j = 1) \le 1/2^k$. Therefore for any constant c we can write

$$\Pr(Y_k = c) \le \binom{2^k}{c} \frac{1}{(2^k)^c} < \frac{1}{c!}. \tag{4.9}$$

Suppose $c \ge \lg^2 n$. Applying Stirling's formula to (4.9)

$$\Pr(Y_k = c) = O\left(n^{-\Omega(\lg n)}\right).$$

Summing over all $c \ge \lg^2 n$ shows that

$$\Pr(Y_k \ge \lg^2 n) = O\left(n^{-\Omega(\lg n)}\right)$$

and proves the lemma.

From this we can derive (4.5a).

Corollary 4.12:

$$\Pr(M > \lg^3 n) < n^{-\Omega(\lg n)} \tag{4.5a}$$

[2] It is possible that the l'th row isn't the sum of 2^l random variables, i.e. $i \ne 2^{l+1} - 1$. When this occurs we pad Y_l by defining $Z_{i+1} = Z_{i+2} = \cdots = Z_{2^{l+1}-1} = 0$.

Proof:

$$\Pr(M > \lg^3 n) < \sum_{1 \le i \le n} \Pr(M_i > \lg^3 n) < n^{-\Omega(\lg n)}.$$

Next we are going to prove (4.5b), a probabilistic bound on A. Let q be a point chosen $\mathbf{U}[0,1]^2$. The probability that $q \in \mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2$ is

$$p = \text{Area}(\mathcal{A}_1 \cup \mathcal{T}_1 \cup \mathcal{C} \cup \mathcal{T}_2 \cup \mathcal{A}_2) = 2n^{-\alpha'} - n^{-2\alpha'} \tag{4.10}$$

It is not hard to see that A is a $B(n,p)$ binomially distributed random variable. Therefore

$$\begin{aligned} \Pr(A = i+1) &= \binom{n}{i+1} p^i (1-p)^{n-i} \\ &= \frac{n-i}{i+1} \frac{p}{1-p} \Pr(A = i). \end{aligned}$$

If $i \ge 2np$ this implies $\Pr(A = i+1) \le \Pr(A = i)/2$. Telescoping this inequality gives

$$\Pr(A = 3np) \le 2^{-n} \Pr(A = 2np) < 2^{-n}.$$

Telescoping again gives

$$\Pr(A > 3np) = \Pr(A = 2np) \sum_{i \ge 0} 2^{-i} < 2^{1-n}. \tag{4.11}$$

Substituting $p = 2n^{-\alpha} - n^{\alpha^2}$ back into this equation yields (4.5b).

The same technique will prove (4.5c). Replace A with W and (4.10) with

$$p = \Pr(p_i \in \mathcal{W}_1 \cup \mathcal{D} \cup \mathcal{W}_2) = 2n^{-\beta'} - n^{-2\beta'} - 2n^{\alpha'+\beta'}.$$

Evaluating (4.11) gives (4.5c).

Next we are going to prove (4.5d). This time we let

$$p = \Pr(p_i \in \mathcal{C}) = 2n^{-2\alpha'}$$

Then

$$\Pr(F_{\mathcal{C}} > j) = \Pr(p_i \notin \mathcal{C}, i \le j) = (1-p)^j$$

giving

$$\Pr(F_c > n^{-2\alpha'} \lg^2 n) = (1-p)^{n^{-2\alpha'} \lg^2 n} < n^{-\Omega(\lg n)}. \tag{4.12}$$

$$\mathcal{C} = [\alpha, \infty] \times [\alpha, \infty]$$
$$\mathcal{T}_1 = [\alpha, \infty] \times [\beta, \alpha]$$
$$\mathcal{T}_2 = [\beta, \alpha] \times [\alpha, \infty]$$
$$\mathcal{D} = [\beta, \alpha] \times [\beta, \alpha]$$
$$\mathcal{A}_1 = [\alpha, \infty] \times [-\infty, \beta]$$
$$\mathcal{A}_2 = [-\infty, \beta] \times [\alpha, \infty]$$
$$\mathcal{W}_1 = [\beta, \alpha] \times [-\infty, \beta]$$
$$\mathcal{W}_2 = [-\infty, \beta] \times [\beta, \alpha]$$
$$\mathcal{V} = [-\infty, \beta] \times [-\infty, \beta]$$

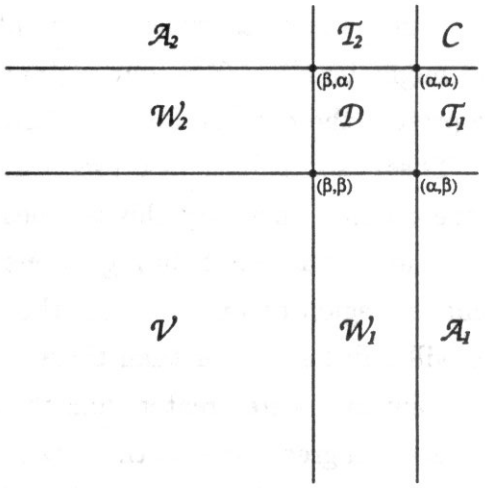


Figure 4.10: We partition the plane into rectangular regions that are functions of α, β.

A similar technique proves (4.5e). Let

$$p = \Pr(p_i \in \mathcal{D}) = 2n^{-2\alpha'}.$$

For each $i \leq n$ let S_i be the minimum number of points to the right of p_i that we must scan before encountering a point in $\mathcal{D}$: $S_i = \min_{k>0}\{k : p_{i+k} \in \mathcal{D}\}$. A calculation similar to (4.12) shows that $\Pr(S_i > n^{-2\beta'} \lg^2 n) < n^{-\Omega(\lg n)}$. By definition $G_{\mathcal{D}} = \max_i S_i$ so

$$\Pr(G_{\mathcal{D}} > n^{-2\beta'} \lg^2 n) \leq \sum_i \Pr(S_i > n^{-2\beta'} \lg^2 n) < n^{-\Omega(\lg n)}.$$

§4.4.4 Extension to CI Distributions

In §4.4.2 and §4.4.3 we analyzed the probabilistic behavior of the MTF algorithm given that its input points were chosen I.I.D. $\mathbf{U}[\mathbf{0}, \mathbf{1}]^{\mathbf{2}}$. We promised there that we would later extend this analysis to apply when the points are chosen I.I.D. from any distribution over $\mathcal{R}^2$ with the CI property. The purpose of this section is to fulfill that promise.

Our first step will be to modify the definitions of the rectangular regions that were presented in Figure (4.7). We assumed there that the input points were confined to $[0, 1]^2$, so our rectangular regions partitioned $[0, 1]^2$. Now, dealing with

points that can be anywhere in the plane (the support of an arbitrary CI distribution might be all of $\mathcal{R}^2$), our rectangular regions must partition the entire plane. We present the new, modified, definitions in Figure (4.10).

These new definitions preserve all of the ordering information that was present in the original ones. By this we mean that if, under the old definition, the x (y) coordinates of all points in region one were greater than the x (y) coordinates of all points in region two then, under the new definition the x (y) coordinates in region one will still be greater than those in region two. For example, the x coordinates of a point in $\mathcal{C}$ were greater than those of all points in $\mathcal{D}$ under the old definitions: they are still greater under the new ones. The reason that we stress the preservation of the ordering information is that this information contained the only geometric facts used in the proof of Theorem 4.7. Therefore Theorem 4.7 is still valid under these new definitions.

Next, we choose suitable values for the parameters α and β. If $p = (X, Y)$ is chosen from a CI distribution then the probability density functions

$$F_X(x) = \Pr(X \leq x), \qquad F_Y(y) = \Pr(Y \leq y)$$

are, by definition, continuous. This, together with the fact that probability distribution functions are always monotonic implies that $F_X(x)$ and $F_Y(y)$ are invertible. For positive constants α', β' we can therefore define

$$\alpha = F_X^{-1}\left(1 - n^{-\alpha'}\right), \qquad \beta = F_Y^{-1}\left(1 - n^{-\alpha'} - n^{-\beta'}\right). \tag{4.13}$$

The rationale behind these definitions is to ensure that the probability that a point will be in a particular region remains invariant: We want the probability that a point will be in a particular region to be the same under the definitions in Figure 4.10 and the parameter settings of (4.13) that it was under the old definitions in Figure 4.7 and the settings $\alpha = n^{-\alpha'}$, $\beta = n^{-\beta'}$. For example, the probability that a point was in $\mathcal{C}$ under the old definitions/settings is the same as it is under the new: the probability is $n^{-2\alpha'}$. This is a consequence of second part of the CI property's definition, which states that the x and y coordinates are independent, i.e.

$$F_{X,Y}(x, y) = \Pr(X \leq x, Y \leq y) = F_X(x)\, F_Y(y).$$

These probabilities (of whether a point is in a region) were the only facts used in the proofs of (4.5b)-(4.5e) so these facts continue to be true. The proof of (4.5a) was based on Lemma 4.3. Lemma 4.3 is true for any CI distribution and not just

$\mathbf{U}[0,1]^2$ and therefore (4.5a) is also true for any distribution with the CI property. To finish this section we note that the proof of Theorem 4.8 followed directly from the Theorem 4.7 and (4.5a)-(4.5f) so we have just generalized Theorem 4.8 to get

Theorem 4.8′: The number of surplus comparisons performed by the MTF algorithm when run on n points chosen I.I.D. from a distribution with the CI property is $O\left(n^{6/7}\lg^5 n\right)$ with probability $1 - n^{-\Omega(\lg n)}$.

We have also generalized its attendant corollary

Corollary 4.9′: The expected number of point comparisons performed by the MTF algorithm when run on n points chosen I.I.D. from a distribution with the CI property is $n + O\left(n^{6/7}\lg^5 n\right)$.

§4.5 Conclusions

In this chapter we proved the following conjecture of Bentley, Clarkson, and Levine [**BCL**]: the MTF algorithm performs only $n + o(n)$ expected point comparisons when run on n points chosen I.I.D. from any distribution over $\mathcal{R}^2$ with the CI property. More specifically, we proved that when the points are chosen from such a distribution then with probability $1 - n^{-\Omega(\lg n)}$ only $n + O(n^{6/7}\lg^5 n)$ point comparisons will be made. The proof depends on an amortized probabilistic technique.

There are two open questions left to answer. The first one is whether it is possible to improve our result to match the experimental evidence gathered by [**BCL**]. The second one is whether it is possible to extend our result so that it remains valid in higher than two dimensions.

We start with the first question. Recall (Figure 4.6 and the extended quote presented in §4.3) that [**BCL**] found experimental evidence that the expected number of surplus comparisons performed on n points chosen I.I.D. $\mathbf{U}[0,1]^2$ is $7.6n^{.515\cdots} - 27.7$. Unfortunately it does not seem possible to extend Theorem 4.8 to show this. The proof of the theorem was based on a balancing argument that minimized (4.7). This balancing argument seems to show that the techniques employed by our analysis can't be pushed to provide even tighter results.

Now, we come to the second question: can Theorem 4.8 be generalized to show that with high probability the number of surplus comparisons is sublinear no matter what the dimension of the space that underlies the CI distribution? Unfortunately it does not seem possible to extend the techniques of this chapter in that direction. This is because the techniques made implicit use of the ordering properties of points

in two dimensions. More explicitly, Lemma 4.10 can not be extended. In Lemma 4.10 we used the fact that all of the maxima that would dominate $q_1, \ldots, q_m$ are either in $\mathcal{T}_1$ or in $\mathcal{A}_2 \cup \mathcal{T}_2 \cup \mathcal{C}$. Furthermore the maxima in $\mathcal{A}_2 \cup \mathcal{T}_2 \cup \mathcal{C}$ at the front of the queue would appear sorted by their x-coordinates. It is this last fact which does not seem to generalize.

REFERENCES:

[**AW**] Alok Aggarwal and Joel Wein, "Computational Geometry (Lecture Notes for 18.409)," *MIT/LCS/RSS Research Seminar Series,* (August 1988).

[**BDMW**] R.A. Becker, L. Denby, R. McGill, and A.R. Wilks, "Analysis of Data from the Places Rated Almanac," *The American Statistician,* **41**(3) (August, 1987) 169-186.

[**BCL**] J.L. Bentley, K.L. Clarkson, and D.B. Levine, "Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls," *Symposium on Discrete Algorithms,* (1990).

[**BKST**] J.L. Bentley, H.T. Kung, M. Schkolnick, and C.D. Thompson, "On the Average Number of Maxima in a Set of Vectors and Applications," *Journal of the Association for Computing Machinery,* **25**(4) (October 1978) 536-543.

[**BP**] J. L. Bentley and C. H. Papadimitriou, "A Worst-Case Analysis of Nearest-Neighbor Searching by Projection," *7th Int. Conf. on Automata, Languages and Programming,* (August 1980) 470-482.

[**BS1**] J.L. Bentley and M.I. Shamos, "Divide and Conquer for Linear Expected Time," *Information Processing Letters,* **7**(2) (Feb. 1978) 87-91.

[**BWY**] J.L. Bentley, B.W. Weide, and A.C. Yao, "Optimal Expected-Time Algorithms for Closest Point Problems," *ACM Trans. on Mathematical Software,* **6**(4) (Dec. 1980) 563-580.

[**BS2**] R. Boyer and D. Savageau, *Places Rated Almanac,* Rand McNally. (1985).

[**Byk**] A. Bykat, "Convex Hull of a Finite Set of Points in Two Dimensions," *Information Processing Letters,* **7** (1978) 296-298.

[**Ch**] Bernard Chazelle, "On the Convex Layers of a Planar Set," *IEEE Transactions on Information Theory,* **IT-31**(4) (July 1985) 509-517.

[**De**] Luc Devroye, "How to Reduce the Average Complexity of Convex Hull Finding Algorithms," *Comp. and Maths. with Appls.,* **7** (1981) 299-308.

[**DT**] L. Devroye and G. Toussaint, "A Note on Linear Expected Time Algorithms for Finding Convex Hulls," *Computing,* **26** (1981) 361-366.

[**Dw**] Rex Allen Dwyer, "Average-Case Analysis of Algorithms For Convex Hulls and Voronoi Diagrams," *Thesis - Carnegie-Mellon University,* **CMU-CS-88-132** (1988).

[**Ed**] W.F. Eddy, "A New Convex Hull Algorithm for Planar Sets," *ACM Transactions on Mathematical Software,* **3** (1977) 398-403.

[**FH**] S. Fortune and J.E. Hopcroft, "A Note on Rabin's Nearest Neighbor Algorithm," *Information Processing Letters,* **8**(1) (1979) 20-23.

[**Fo**] A. Fournier, "Comments on Convex Hull of a Finite Set of Points in Two Dimensions," *Information Processing Letters,* **8** (1979) 173.

[**GKP**] Ronald Graham, Donald Knuth, and Oren Patashnik, *Concrete Mathematics: A Foundation For Computer Science,* Addison-Wesley, Reading, Mass. (1988).

[**GS1**] M.J. Golin and R. Sedgewick, "Analysis of a Simple but Efficient Convex Hull Algorithm," *Proceedings of the Fourth Annual Symposium on Computational Geometry,* (1988) 153-163.

[**GS2**] G.R. Grimmet and D.R. Stirzaker, *Probability and Random Processes,* Clarendon Press, Oxford. (1985).

[**HNS**] Klaus Hinrichs, Jurg Nievergelt, and Peter Schorn, "Plane-Sweep Solves the Closest Pair Problem Elegantly," *Information Processing Letters,* **26** (January 11, 1988) 255-261.

[**Kn1**] Donald E. Knuth, *The Art of Computer Programming: Fundamental Algorithms (2nd ed),* Addison-Wesley, Reading, Mass. (1973).

[**Kn2**] Donald E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms (2nd ed),* Addison-Wesley, Reading, Mass. (1981).

[**Kn3**] Donald E. Knuth, *The Art of Computer Programming: Sorting and Searching,* Addison-Wesley, Reading, Mass. (1973).

[**KLP**] H.T. Kung, F. Luccio, and F.P. Preparata, "On Finding the Maxima of a Set of Vectors," *Journal of the Association for Computing Machinery,* **22**(4) (October 1975) 469-474.

[**LP**] D.T. Lee and F.P. Preparata, "Computational Geometry - A Survey," *IEEE Transactions on Computers,* **c-33** (12) (1984) 1072-1101.

[**OL**] Mark H. Overmars and Jan van Leeuwen, "Further Comments on Bykat's Convex Hull Algorithm," *Information Processing Letters,* **10** (July 5, 1980) 209-212.

[**Pi**] Rob Pike, "Notes on Programming in C," *Unpublished Manuscript,.*

[**PS**] Franco P. Preparata and Michael Ian Shamos, *Computational Geometry, An Introduction,* Springer-Verlag, New York. (1985).

[**Ra**] M.O. Rabin, "Probabilistic Algorithms," *Algorithms and Complexity: New Directions and Recent Results (J.F. Traub ed.),* (1976) 21-39.

[**RS**] A. Renyi and R. Sulanke, "Ueber die konvexe hulle von n zufallig gewahlten punkten, I," *Z. Wahrschien,* **2** (1963) 75-84.

[**Ru**] Walter Rudin, *Real and Complex Analysis,* McGraw Hill, New York. (1966).

[**Se1**] Robert Sedgewick, *Algorithms,* Addison-Wesley, Reading, Mass. (1983).

[**Se2**] Robert Sedgewick, *Quicksort,* Garland Publishing, New York. (1980).

[**Sha1**] Michael Ian Shamos, "Computational Geometry," *Thesis (Yale),* (1978).

[**Sha2**] M.I. Shamos, "Geometry and Statistics: Problems at the Interface," *Algorithms and Complexity: New Directions and Recent Results (J.F. Traub ed.),* (1976) 251-280.

[**SH**] Michael Ian Shamos and Dan Hoey, "Closest Point Problems," *16'th Annual Symposium on Foundations of Computer Science (FOCS),* (1975) 151-161.

[**St**] J.M. Steele, "Growth Rates of Euclidean Minimal Spanning Trees with Power Weighted Edges," *The Annals of Probability,* **16**(4) (1988) 1767-1787.

[**Ta**] Robert E. Tarjan, "Amortized Computational Complexity," *SIAM J. Alg. Disc. Math,* **6**(2) (April 1985) 28-40.

[**We**] Bruce W. Weide, "Statistical Methods in Algorithm Design and Analysis," *Thesis (Carnige-Mellon University). CMU-CS-78-142,* (August 1978).

[**WW**] E.T. Whittaker and G. N. Watson, *A Course in Modern Analysis,* Cambridge University Press, London. (1973).