COS 226 Algorithms and Data Structures Spring 2014

Final Exam

This test has 15 questions worth a total of 100 points. You have 180 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11, both sides, in your own handwriting). No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

"I pledge my honor that I have not violated the Honor Code during this examination."

Problem	Score
0	
1	
2	
3	
4	
5	
6	
7	
Sub 1	

Problem	Score
8	
9	
10	
11	
12	
13	
14	
Sub 2	

Total	

Name:

netID:

Room:

Precept:

P01	Th 11	Andy Guna
P02	Th 12:30	Andy Guna
P03	Th 1:30	Chris Eubank
P04	F 10	Jenny Guo
P05	F 11	Madhu Jayakumar
P05A	F 11	Nevin Li
P06	F 2:30	Josh Hug
P06A	F 2:30	Chris Eubank
P06B	F 2:30	Ruth Dannenfelser
P07	F 3:30	Josh Hug

0. **Initialization. (1 point)**

   In the space provided on the front of the exam, write your name and Princeton netID; circle your precept number; write the name of the room in which you are taking the exam; and write and sign the honor code.

1. **Analysis of algorithms. (8 points)**

   (a) You observe the following running times for a program with an input of size N.

N	time
1,000	0.1 seconds
2,000	0.3 seconds
4,000	2.5 seconds
8,000	19.8 seconds
16,000	160.1 seconds

   Estimate the running time of the program (in seconds) on an input of size $N = 80,000$.

seconds

   (b) Consider the following implementation of a binary trie data type:

```
public class BinaryTrieST<Value> {
   private Node root;        // root of trie
   private int N;            // number of nodes in the trie

   private class Node {
       private Value val;
       private Node left;
       private Node right;
   }
   ...
}
```

   Using the 64-bit memory cost model from lecture and the textbook, how much memory (in bytes) does a `BinaryTrieST` object use to store M key-value pairs in N nodes?

   Use tilde notation to simplify your answer. Do not include the memory for the values themselves but do include all other memory (including pointers to values).

$\sim$ *bytes*

(c) For each function on the left, give the best matching order of growth of the *running time* on the right. You may use an answer more than once or not at all.

__B__
```
public static int f1(int N) {
   int x = 0;
   for (int i = 0; i < N; i++)
       x++;
   return x;
}
```

_____
```
public static int f2(int N, int R) {
   int x = 0;
   for (int i = 0; i < R; i++)
      x += f1(i);
   return x;
}
```

_____
```
public static int f3(int N, int R) {
   int x = 0;
   for (int i = 0; i < R; i++)
      for (int j = 0; j < N; j++)
         x += f1(j);
   return x;
}
```

_____
```
public static int f4(int N, int R) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 1; j <= R; j += j)
          x++;
   return x;
}
```

_____
```
public static int f5(int N, int R) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 1; j <= R; j += j)
          x += f1(j);
   return x;
}
```

A. R

B. N

C. $N + R$

D. $N \log R$

E. $R \log N$

F. NR

G. R^2

H. N^2

I. $NR \log N$

J. $NR \log R$

K. NR^2

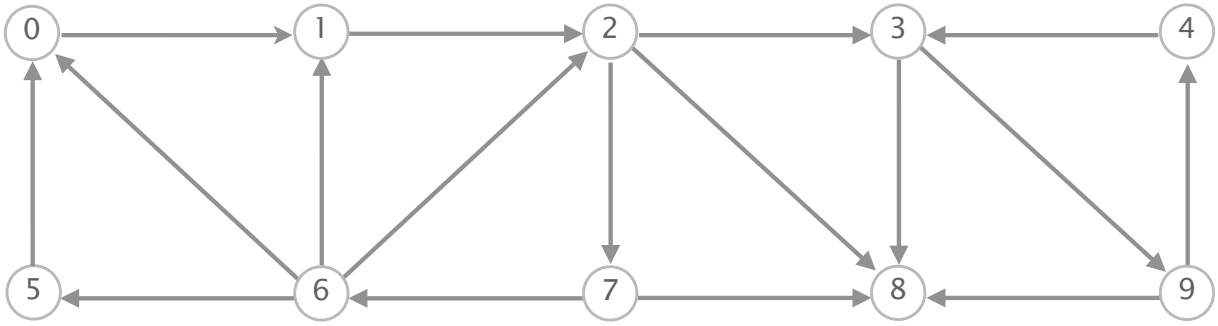
L. RN^2

M. R^3

N. N^3

2. **Graph search. (6 points)**

Consider the following digraph. Assume the adjacency lists are in sorted order: for example, when iterating through the edges pointing from 2, consider the edge $2 \to 3$ before either $2 \to 7$ or $2 \to 8$.



Run depth-first search on the digraph, starting from vertex 0.

(a) List the vertices in *reverse postorder*.

0

--- --- --- --- --- --- --- --- --- ---

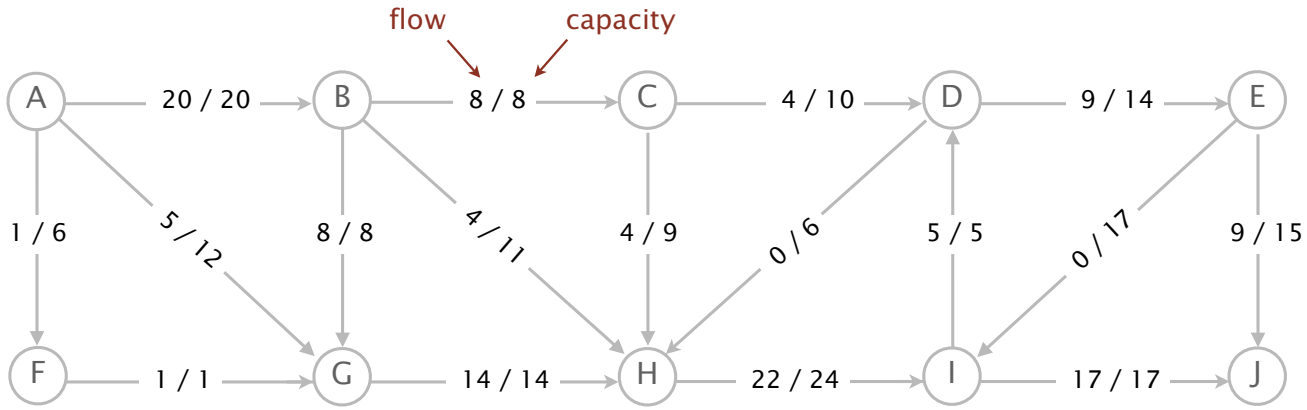
(b) List the vertices in *preorder*.

0

--- --- --- --- --- --- --- --- --- ---

3. **Maximum flow. (10 points)**

Consider the following flow network and feasible flow f from from the source vertex A to the sink vertex J.



(a) What is the value of the flow f?

(b) Starting from the flow f given above, perform one iteration of the Ford-Fulkerson algorithm. List the sequence of vertices on the augmenting path.

(c) What is the value of the maximum flow?

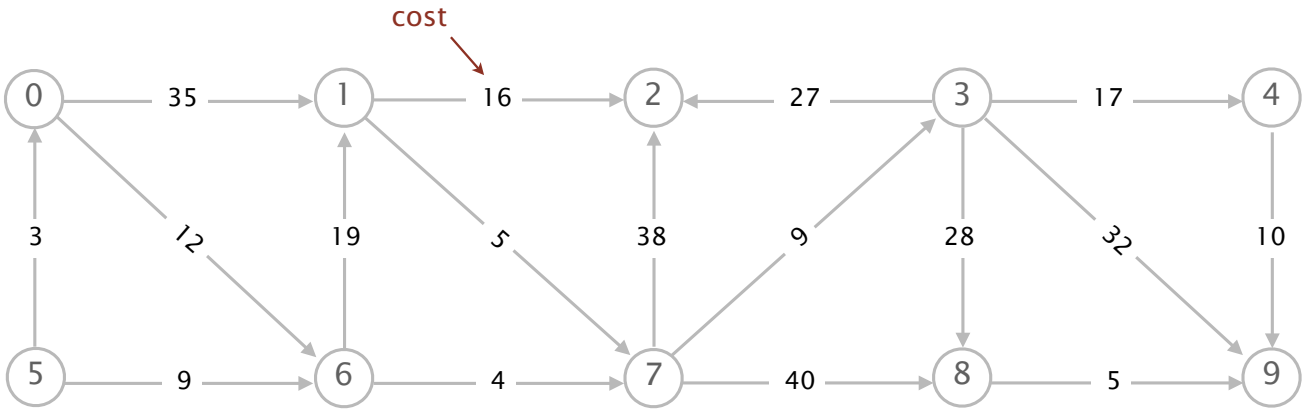
(d) Circle the vertices on the source side of a minimum cut.

A B C D E F G H I J

(e) Give one edge such that if its capacity were decreased by one, then the value of the maxflow would decrease.

4. **Shortest paths. (6 points)**

Suppose that you are running Dijkstra's algorithm on the edge-weighted digraph below, starting from some vertex s (not necessarily 0).



The table below gives the `edgeTo[]` and `distTo[]` values immediately after vertex 7 has been deleted from the priority queue and relaxed.

v	`distTo[]`	`edgeTo[]`
0	`3.0`	$5 \to 0$
1	`28.0`	$6 \to 1$
2	`51.0`	$7 \to 2$
3	`22.0`	$7 \to 3$
4	∞	*null*
5	`0.0`	*null*
6	`9.0`	$5 \to 6$
7	`13.0`	$6 \to 7$
8	`53.0`	$7 \to 8$
9	∞	*null*

(a) Give the order in which the first 4 vertices were deleted from the priority queue and relaxed.

(b) Which is the *next* vertex after 7 to be deleted from the priority queue and relaxed?

0 1 2 3 4 5 6 7 8 9

(c) In the table below, fill in those entries (*and only those entries*) in the `edgeTo[]` and `distTo[]` arrays that change (from the corresponding entries on the facing page) immediately after the next vertex after 7 is deleted from the priority queue and relaxed.

v	`distTo[]`	`edgeTo[]`
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

5. **String sorting algorithms. (7 points)**

The column on the left is the original input of 24 strings to be sorted; the column on the right are the strings in sorted order; the other 7 columns are the contents at some intermediate step during one of the 3 radix sorting algorithms listed below. Match up each column with the corresponding sorting algorithm. You may use a number more than once.

mink	bear	bear	calf	crow	myna	crab	bear	bear
moth	calf	calf	lamb	lamb	crab	toad	crow	calf
crow	crow	crow	hare	deer	lamb	swan	calf	crab
myna	crab	crab	wasp	crab	toad	bear	crab	crow
swan	deer	hare	hawk	hare	mule	deer	deer	deer
wolf	hare	kiwi	ibex	bear	hare	ibex	hare	hare
mule	hawk	deer	bear	kiwi	sole	hoki	hawk	hawk
slug	hoki	hawk	deer	calf	wolf	mule	hoki	hoki
hare	ibex	ibex	mink	hawk	calf	sole	ibex	ibex
bear	kiwi	hoki	lion	ibex	slug	wolf	kiwi	kiwi
kiwi	lion	lion	kiwi	hoki	moth	calf	lion	lamb
calf	lynx	lynx	slug	lion	kiwi	lamb	lynx	lion
hawk	lamb	lamb	toad	lynx	hoki	myna	lamb	lynx
ibex	mink	mink	hoki	mink	mink	mink	mink	mink
oryx	moth	mule	sole	mule	hawk	lynx	moth	moth
lion	myna	myna	wolf	myna	swan	lion	myna	mule
sole	mule	moth	moth	moth	lion	crow	mule	myna
wasp	oryx	wasp	crab	wasp	wasp	hare	oryx	oryx
lynx	swan	sole	crow	sole	bear	wasp	swan	slug
hoki	slug	oryx	oryx	oryx	deer	moth	slug	sole
crab	sole	slug	mule	slug	crow	slug	sole	swan
deer	toad	wolf	swan	wolf	ibex	kiwi	toad	toad
lamb	wolf	toad	myna	toad	oryx	hawk	wolf	wasp
toad	wasp	swan	lynx	swan	lynx	oryx	wasp	wolf
----	----	----	----	----	----	----	----	----
0								4

(0) Original input

(1) LSD radix sort

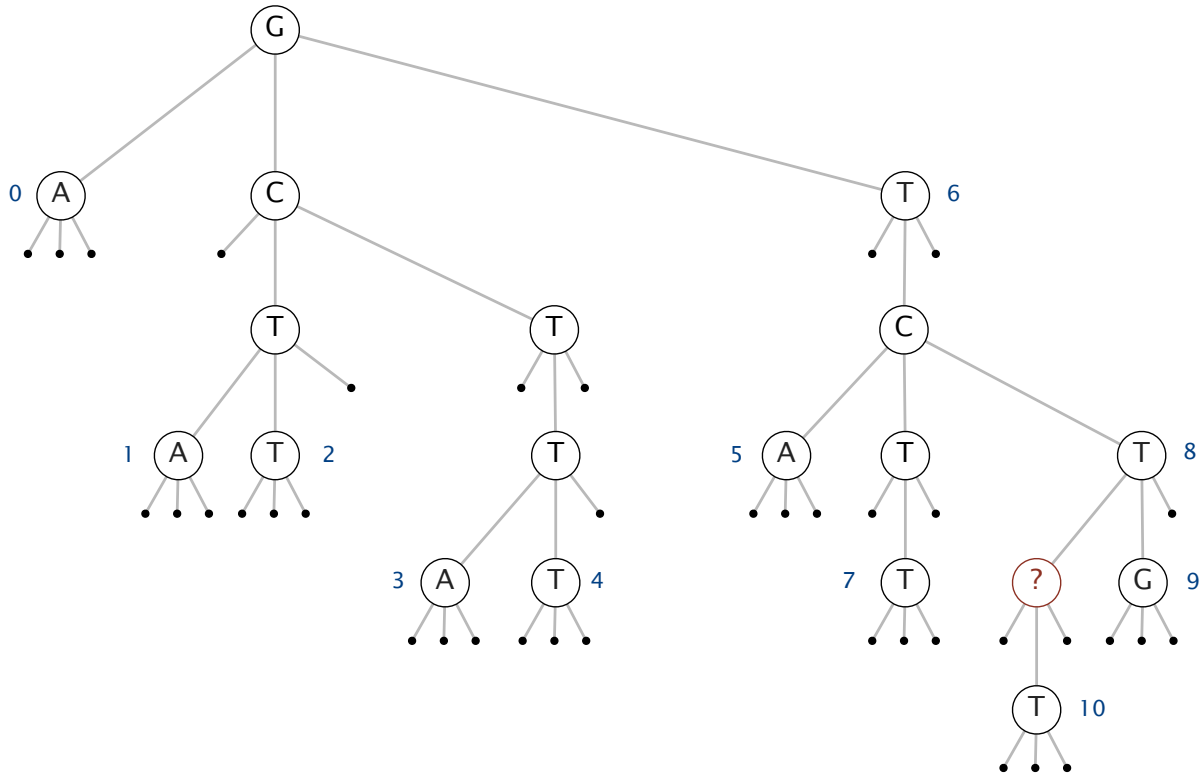
(2) MSD radix sort

(3) 3-way radix quicksort (no shuffle)

(4) Sorted

6. **Ternary search tries. (5 points)**

Consider the following ternary search trie over the alphabet { A, C, G, T }, where the values are shown next to the nodes of the corresponding string keys. The node containing `?` contains one of the characters { A, C, G, T }.



Circle which one or more of the following string keys are (or could be) in the TST above.

`A`	`CT`	`GCA`	`GCG`	`GT`	`GTT`	`TA`
`TCA`	`TAT`	`TCT`	`TCTT`	`TGT`	`TTT`	`TTTT`

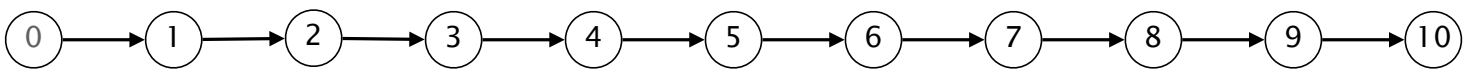
7. **Knuth-Morris-Pratt substring search. (6 points)**

Below is a partially-completed Knuth-Morris-Pratt DFA for a string s of length 10 over the alphabet { `A`, `B`, `C` }.

	0	1	2	3	4	5	6	7	8	9
`A`				4	5		2			
`B`				0	0		7			3
`C`				0	0		0			10
s										

(a) Reconstruct the string s in the last row of the table above.

(b) Complete the first row of the table above (corresponding to the character `A`).



Feel free to use this diagram for scratch work.

8. **Boyer-Moore substring search. (6 points)**

Suppose that you run the Boyer-Moore algorithm (the basic version considered in the textbook and lecture) to search for the pattern

```
R O W S T H E
```

in the text

```
O C I E T Y E X C E P T T H E S C A R E C R O W S T H E
```

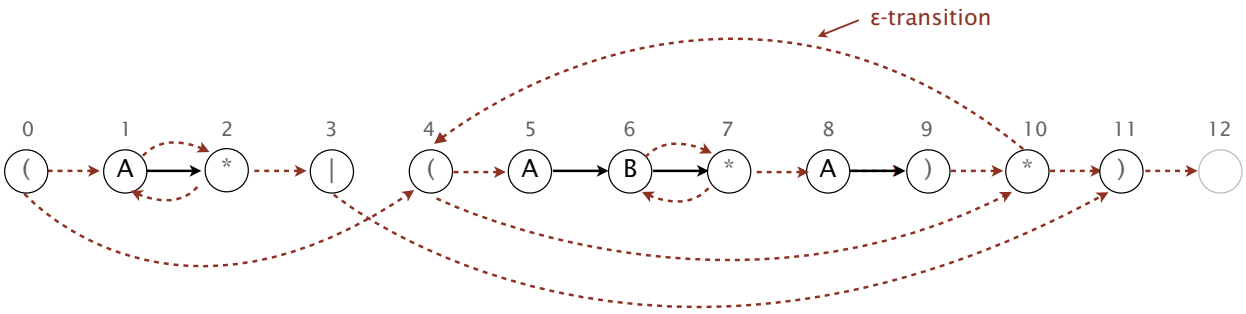
(a) Give the trace of the algorithm in the grid below, circling the characters in the pattern that get compared with characters in the text.

O	C	I	E	T	Y	E	X	C	E	P	T	T	H	E	S	C	A	R	E	C	R	O	W	S	T	H	E
R	O	W	S	T	(H)	(E)																					

(b) Give a pattern string of length 7 that would result in the `Y` in the text being compared twice when running the Boyer-Moore algorithm.

9. **Regular expressions. (7 points)**

The following NFA is the result of applying the NFA construction algorithm from lecture and the textbook to some regular expression.



(a) What is the regular expression?

(b) Suppose that you simulate the following sequence of characters on the NFA above:

```
A  A  A  A  A  A  A
```

In which one or more states could the NFA be?

0 1 2 3 4 5 6 7 8 9 10 11 12

(c) Suppose that you want to construct an NFA for the regular expression

```
( A * | ( A B * A ) + )
```

where the operator `+` means *one or more copies*. What minimal change(s) would you make to the NFA above?

10. **LZW compression. (5 points)**

What is the result of expanding the following LZW-encoded sequence of 11 hexadecimal integers?

```
43  41  42  42  82  43  81  41  87  82  80
```

Assume the original encoding table consists of all 7-bit ASCII characters and uses 8-bit codewords. Recall that codeword 80 is reserved to signify end of file.

C	A	B	B											

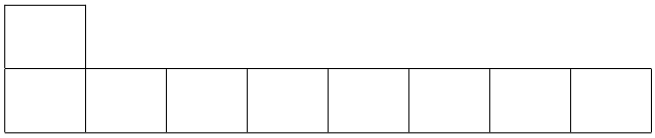
For reference, below is the hexademical-to-ASCII conversion table from the textbook:

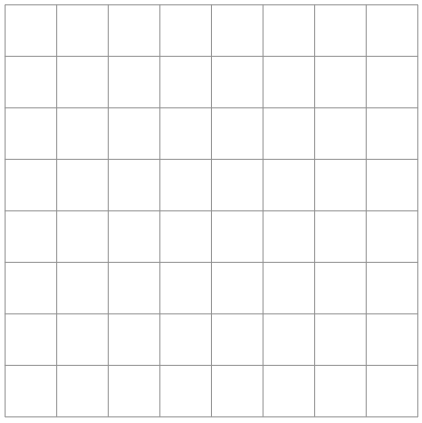
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

11. **Burrows-Wheeler transform. (8 points)**

(a) What is the Burrows-Wheeler transform of the following?

B D A B A C A C

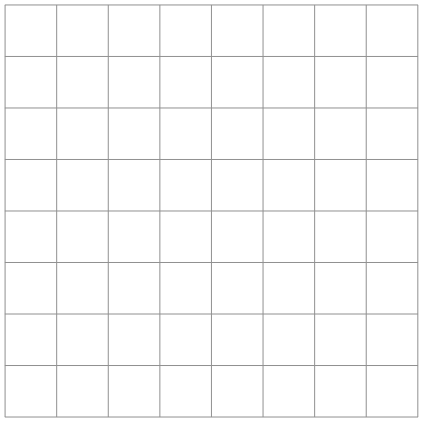




Feel free to use this grid for scratch work.

(b) What is the Burrows-Wheeler inverse transform of the following?

4
D A D C C C D B



Feel free to use this grid for scratch work.

12. **Problem identification. (9 points)**

You are applying for a job at a new software technology company. Your interviewer asks you to identify which of the following tasks are possible, impossible, or unknown.

A. Possible.

B. Impossible.

C. Unknown.

_____ Given an undirected graph, determine if there exists a path of length $V - 1$ with no repeated vertices in time proportional to EV in the worst case.

_____ Given a digraph, determine if there exists a directed path between every pair of vertices in time proportional to $E + V$ in the worst case.

_____ Given a digraph, design an algorithm to determine whether it is a *rooted DAG* (i.e., a DAG in which there is a path from every vertex to some root r) in time proportional to $E + V$ in the worst case.

_____ Given a flow network (a digraph with positive edge capacities) and two vertices s and t, find the the value of the min st-cut in time proportional to $E + V$ in the worst case.

_____ Given a digraph where each edge is colored black or orange and two vertices s and t, find a path from s to t that uses the fewest number of black edges in time proportional to $E + V$ in the worst case.

_____ Given an array a of N 64-bit integers, determine whether there are two indices i and j such that $a_i = -a_j$ in time proportional to N in the worst case.

_____ Given an array of N integers between 0 and $R - 1$, *stably sort* them in time proportional to $N + R$ in the worst case.

_____ Determine *how many times* a pattern string of length M appears as a substring in a text string of length N in time proportional to $M + N$ in the worst case. For simplicity, assume the binary alphabet.

_____ Design an algorithm that compresses at least half of all 10,000-bit messages by one (or more) bits.

13. **Reductions. (8 points)**

Consider the following two string-processing problems:

- SUFFIX-ARRAY. Given a string s, compute its suffix array `sa[]`.
- CIRCULAR-SUFFIX-ARRAY. Given a string s, compute its circular suffix array `csa[]`.

For example, the suffix array `sa[]` and circular suffix array `csa[]` of the string s = ABAAB are given below, along with the the corresponding suffixes and circular suffixes (in parentheses).

i	s[i]	sa[i]	csa[i]
0	A	2 (AAB)	2 (AABAB)
1	B	3 (AB)	0 (ABAAB)
2	A	0 (ABAAB)	3 (ABABA)
3	A	4 (B)	1 (BAABA)
4	B	1 (BAAB)	4 (BABAA)

Show that SUFFIX-ARRAY over the binary alphabet linear-time reduces to CIRCULAR-SUFFIX-ARRAY over the binary alphabet by completing parts (a) and (b).

(a) Show that SUFFIX-ARRAY over the binary alphabet $\{\mathrm{A}, \mathrm{B}\}$ linear-time reduces to CIRCULAR-SUFFIX-ARRAY over the base-4 alphabet $\{0, 1, 2, 3\}$.

i. Given a string input s to SUFFIX-ARRAY over the alphabet $\{\mathrm{A}, \mathrm{B}\}$, how do you construct the corresponding string input s' to CIRCULAR-SUFFIX-ARRAY over the alphabet $\{0, 1, 2, 3\}$?

ii. Given the string input s = ABAAB, what is the corresponding string input s'?

You need not use all of the boxes.

iii. Given the solution `csa[]` to s', how do you construct the solution `sa[]` to s?

(b) Show that CIRCULAR-SUFFIX-ARRAY over the base-4 alphabet $\{0, 1, 2, 3\}$ linear-time reduces to CIRCULAR-SUFFIX-ARRAY over the binary alphabet $\{\mathrm{A}, \mathrm{B}\}$.

i. Given a string input s to CIRCULAR-SUFFIX-ARRAY over the alphabet $\{0, 1, 2, 3\}$, how do you construct the corresponding string input s' to CIRCULAR-SUFFIX-ARRAY over the alphabet $\{\mathrm{A}, \mathrm{B}\}$?

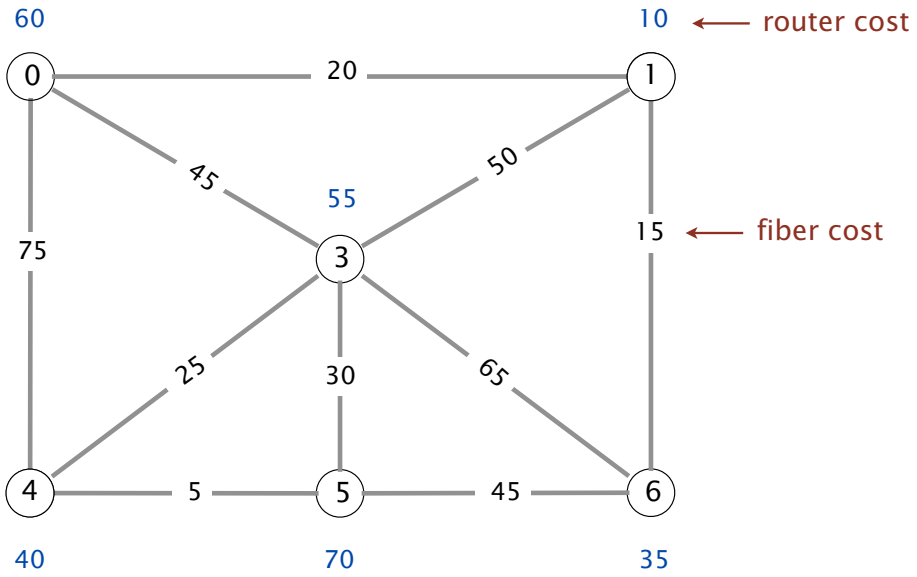
ii. Given the string input $s = 03122$, what is the corresponding string input s'?

You need not use all of the boxes.

iii. Given the solution `csa'[]` to s', how do you construct the solution `csa[]` to s?

14. **Algorithm design. (8 points)**

There are N dorm rooms, each of which needs a secure internet connection. It costs $w_i > 0$ dollars to install a secure router in dorm room i and it costs $c_{ij} > 0$ dollars to build a secure fiber connection between rooms i and j. A dorm room receives a secure internet connection if either there is a router installed there or there is some path of fiber connections between the dorm room and a dorm room with an installed router. The goal is to determine in which dorm rooms to install the secure routers and which pairs of dorm rooms to connect with fiber so as to minimize the total cost.



This instance contains 6 dorm rooms and 10 possible connections. The optimal solution installs a router in dorm rooms 1 and 4 (for a cost of 10 + 40) and builds the following fiber connections: 0-1, 1-6, 3-4, 4-5 (for a cost of 20 + 15 + 25 + 5).

Formulate the problem as a *minimum spanning tree* problem. To demonstrate your formulation, modify the figure above to show the MST problem that you would solve to find the minimum cost set of routers and fiber connections.