Princeton University – Computer Science
COS226: Data Structures and Algorithms

Final, Spring 2013

This test has 14 questions worth a total of 116 points. The exam is closed book, except that you are allowed to use a one page written cheat sheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Write 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."

	Score		Score
0		8	
1		9	
2		10	
3		11	
4		12	
5		13	
6		14	
7			
Sub 1		Sub 2	

Total	/116

Name:

NetID:

Exam Room: McCosh 10
McCosh 46

Circle your precept:

P01	Josh	Th 11	P05	Jennifer	F 11	P07	Nico	F 230
P02	Maia	Th 1230	P05A	Stefan	F 11	P08	Maia	F 10
P03	Arvind	Th 130	P06	Diego	F 230			
P04	Diego	F 330	P06A	Dushyant	F 230			

Tips:

- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. Do not get overly captivated by interesting design issues.
- Not all information provided in a problem may be useful.
- On any design problem, you may use the uniform hashing assumption unless otherwise stated.

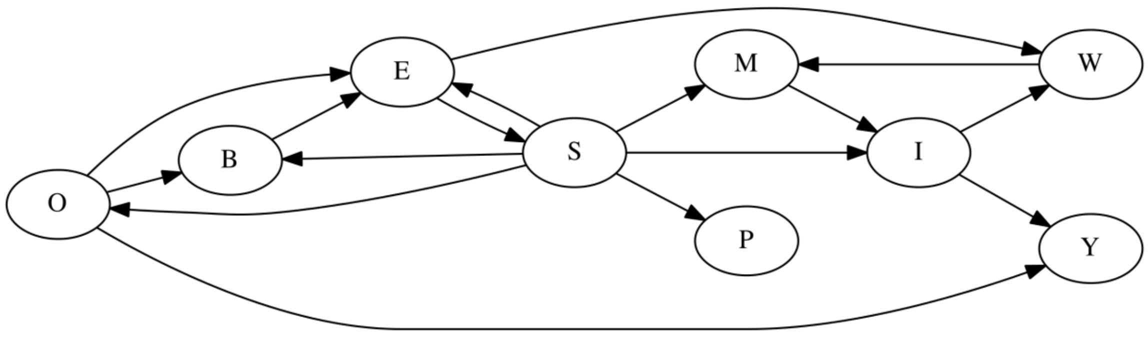
Optional. Mark along the line to show your feelings on spectrum between ☹ and ☺.

Before exam: [☹________________☺].
After exam: [☹________________☺].

0. **(1 point)** Write your name and Princeton NetID on the front page. Circle the exam room. Circle your precept. Enjoy your free point.

1. Graph search (8 points).

Consider the digraph below. Assume the adjacency lists are in sorted order. For example, follow the edge $O \rightarrow B$ before following $O \rightarrow E$.



(a) Give the reverse postorder of the graph when visited by DFS. Start from vertex O.

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

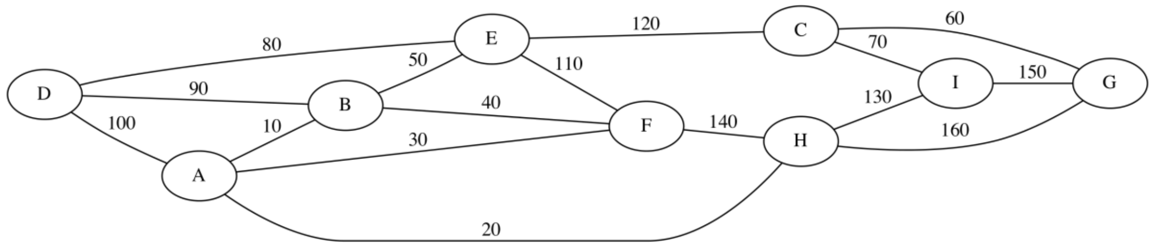
(b) Give the order in which vertices are enqueued using BFS. Start from vertex O.

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

(c) Provide a topological sort that is different than the reverse postorder. If no such sort exists, explain why.

2. **MSTs (8 points).**

Consider the weighted undirected graph below.



(a) Give the order in which the edges are added to the MST using Kruskal's algorithm.

A-B
---- ---- ---- ---- ---- ---- ---- ----

(b) Give the order in which the edges are added to the MST using Prim's algorithm.

A-B
---- ---- ---- ---- ---- ---- ---- ----

(c) Suppose the edge H-C is added to the MST. For what values of H-C does this new edge belong to an MST?

3. **TSTs (8 points).**

(a) If the key value pairs below are inserted into a TST in the order shown, what is the height of the resulting TST? (The height of a 1-node TST is 0), circle your answer. You do not need to draw the TST, but we may use it to award partial credit.

```
maam  : 0
beware: 1
a     : 2
shaman: 3
sham  : 4
in    : 5
town  : 6
```

(b) What is the height of the tree if the keys are inserted in worst case order? Circle your answer.

4. **Analysis of Algorithms (5 points).**

For each of the functions shown in the left column below, give the best matching order of growth of the *running time* on the right.

```
__D__ 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 x = 0;
          for (int i = 0; i < n; i++)
                for (int j = 0; j < i*i; j++)
                      x++;
          return x;
      }

_____ public static int f3 (int n) {
          if (n <= 1) return 1;
          return f3(n-1) + f3(n-1)
      }

_____ public static int f4 (int n) {
          if (n <= 1) return 1;
          return f4(n/2) + f4(n/2);
      }

_____ public static int f5 (int n) {
          if (n <= 1) return 1;
          return f1(n) + f5(n/2) + f5(n/2);
      }

_____ public static void f6(int n) {
          // 1<<i is the same as 2^i.
          // Ignore integer overflow.
          // 1<<i takes constant time.
          for (int i = 0; i < n; i=1<<i);
      }
```

A. Constant

B. Log* N

C. Log N

D. N

E. N log N

F. N^2

G. N^3

H. 2^N

I. 3^N

J. N!

5. **Shortest Paths (10 points).**

Suppose you're using Dijkstra's algorithm starting from some source vertex s. The table on the right shows the shortest paths tree (`edgeTo[]` and `distTo[]`) immediately after vertex 4 has been relaxed.

edge	weight
0 → 2	3.0
0 → 3	1.0
1 → 3	1.0
1 → 6	5.0
2 → 1	2.0
3 → 1	17.0
3 → 2	13.0
3 → 5	3.0
3 → 7	8.0
4 → 7	1.0
5 → 1	4.0
6 → 4	-11.0
7 → 4	2.0

v	distTo[]	edgeTo[]
0	∞	*null*
1	7.0	5
2	13.0	3
3	0.0	*null*
4	10.0	7
5	3.0	3
6	12.0	1
7	8.0	3

(a) Give the order in which the first 5 vertices were deleted from the priority queue and relaxed.

				4

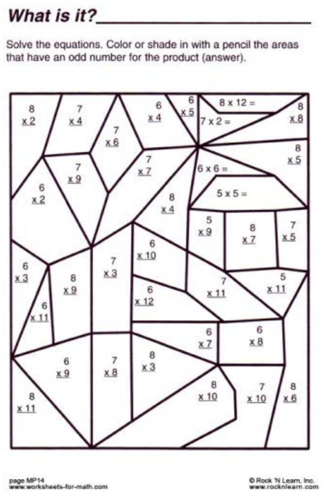
(b) Fill in the table below to reflect the new shortest paths tree (by changing `edgeTo[]` and `distTo[]`) after the next vertex is relaxed. Only fill in those entries which change from the values shown in the table above.

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

(c) As you are no doubt aware, Dijkstra’s algorithm is not guaranteed to provide the correct shortest paths tree when a graph contains a negative weight. For the graph above, it will not.

Suppose we create a new graph that is different only in the weight between 3 → 7. Provide a new weight for the edge 3 → 7 that ensures that Dijkstra’s provides the correct shortest paths tree (assuming all other weights in the graph stay the same). Include a brief explanation of why this new edge weight works.

Look at all this space below, and think of all that free time you have while working on this exam! Fill free to fill it in this space whatever you’d like. If you’re feeling uninspired, there’s a worksheet in the corner from rocknlearn.com, the only website that combines a fresh rock n’ roll attitude with math.



What is it?____________________

Solve the equations. Color or shade in with a pencil the areas that have an odd number for the product (answer).

page MP14
www.worksheets-for-math.com

© Rock 'N Learn, Inc.
www.rocknlearn.com

6. **Substring Matching (9 points).**

Below is a partially completed KMP-DFA.

(a) Fill in the rest of the string in the bottom row. You do not need to fill in the top three rows, but they will be considered for partial credit.

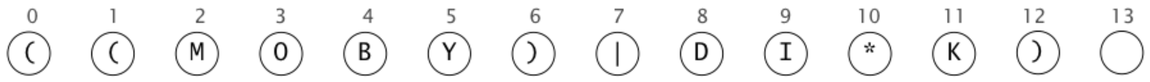
	0	1	2	3	4	5	6	7	8	9	10	11
A												3
B	1	2										12
C												6
s	B	B										B

(b) Consider a text editor that uses KMP DFA for substring search, where everything is optimized to be as fast as possible. Suppose you're using the `find next` button with a single pattern of M characters on a text of length N. In the worst case, after clicking the `find next` button F times, how many times must the KMP DFA be constructed?

(c) Assume N is much larger than FM. Assume that the initial search is from the beginning of the document, and that search does NOT wrap around. Give your answers in terms of order of growth: What is the worst case run time to handle F clicks to `find next` in terms of F, M, N? What is the best case run time?

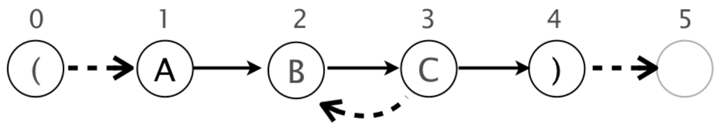
7 **Regular Expressions (7 points).**

(a) Draw the NFA corresponding to the regular expression `((MOBY)|DI*K)` generated by the procedure described in class. Clearly distinguish your epsilon transitions.

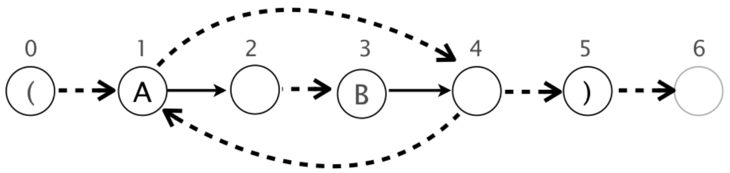


(b) Give a valid regular expression corresponding to the NFAs below. These NFAs were NOT generated using the procedure described in class. **States without a displayed character (i.e. empty circles) do not necessarily correspond to a regular expression meta-character.** Write your answer in the blank provided. You are permitted to use any standard regular expression meta-character. It is also ok if you stick to the basic metacharacters: `( ) * |`

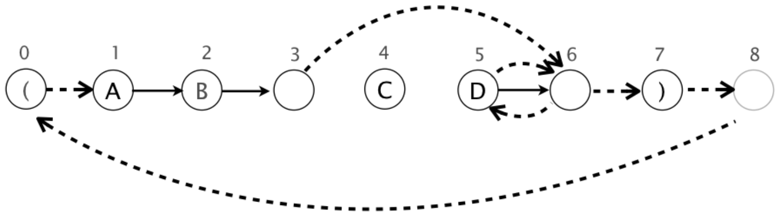
Epsilon transitions are shown with dotted lines. First answer is provided for you.



ABB*C



____

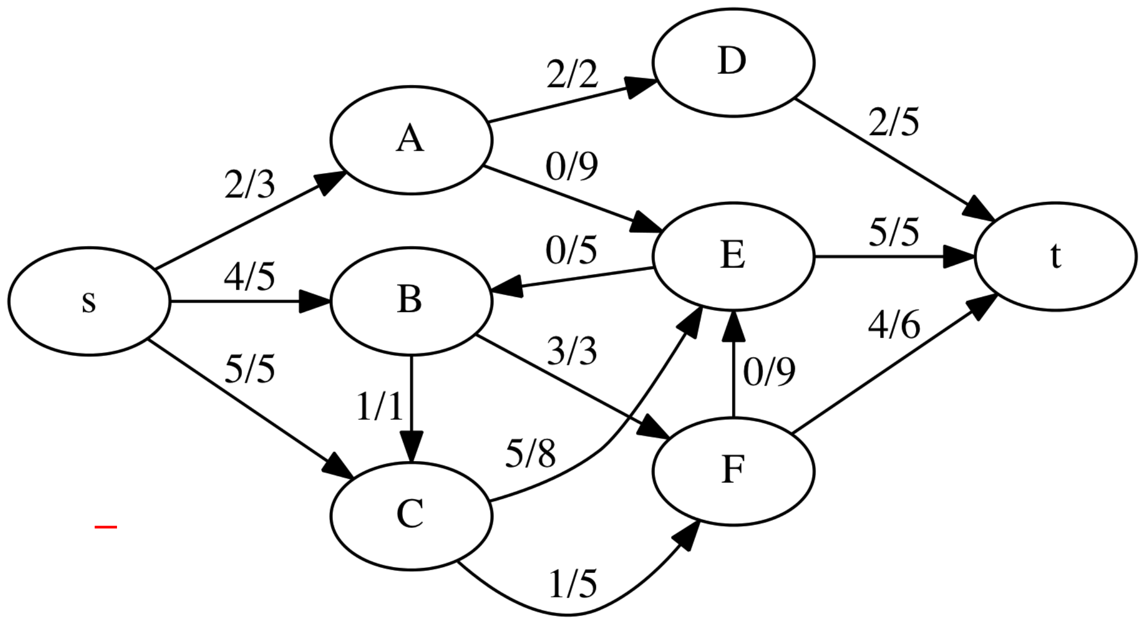


____

(c) Given a string of length N and any regular expression of length M, we can determine if the string matches the regular expression in N time if we use a discrete finite automaton (DFA), and in NM time if we use an nondeterministic finite automaton (NFA). Why do we use an NFA if they are slower?

8. **Reductions and MaxFlow (8 points).**

Consider the graph below.



(a) What is the value of the flow shown above?

(b) What is the value of the max flow?

(c) Which vertices are on the s side of the min cut?

(d) Suppose we are given the following facts:

i. An OctopusWrangling problem of size N reduces to a MaxFlow problem with order N^2 edges in quadratic time.
ii. Dinitz, Edmonds, and Karp have discovered an algorithm that can solve MaxFlow in time E^3.
iii. Maia has discovered an $E \log E$ lower bound for solving MaxFlow.

Which of the following can you definitively infer from these three facts? Mark each blank with Y if you can infer, and N if you cannot infer.

----- MaxFlow provides an N^6 solution to OctopusWrangling.

----- $N^2 \log N^2$ is a lower bound for OctopusWrangling.

----- There exists an $N^2 \log N^2$ solution for OctopusWrangling.

----- An $N^2 \log N^2$ solution to OctopusWrangling provides an order $E \log E$ solution to MaxFlow.

9. **True. False. Pain. (10 points).**

Mark each proposition as True or False.

----- Adding a constant to every edge weight in an undirected graph can change the set of edges that belong to the minimum spanning tree. Assume unique weights.

----- Adding a constant to every edge weight in a directed graph can change the set of edges that belong to a shortest paths tree. Assume unique weights.

----- Finding a Hamilton tour on a directed acyclic graph is NP-Complete.

----- Assuming the graph is connected, the following algorithm completes and results in a correct MST: Pick a random cut. Identify the minimum crossing edge. Add that edge to the MST if it is not already present. Repeat until V-1 edges are in the MST.

----- Assuming the graph is connected, the following algorithm completes and results in a correct MST: Pick a random vertex. Identify the minimum edge from that vertex. Add to the MST if it is not already present and does not form a cycle, otherwise continue to the next iteration. Repeat until V-1 edges are in the MST.

----- Rerunning Dijkstra's algorithm on a graph V times will result in the correct shortest paths tree, even if there are negative edges (but no negative cycles). Assume that edgeTo[] and distTo[] persist between calls to Dijkstra's algorithm, but not marked[].

----- "Does there exist a tour of city-set Q of length less than 10000?" is in NP.

----- "Is tour X the shortest tour of city-set Q?" is in NP.

----- Being able to solve any NP Complete problem in polynomial time would allow you to find the Kolmogorov complexity of any string in polynomial time.

----- Proving that a problem reduces to 3SAT means that this problem is NP Complete.

10. **Recursive Code / BinaryStdIn (6 points).**

```
private static class Node {
   private char ch;
   private final Node left, right;

   Node(char ch, Node left, Node right) {
       this.ch = ch;
       this.left = left;
       this.right = right;
   }
}

private static Node readTrie() {
   if (BinaryStdIn.readBoolean()) {                 //readBoolean: reads 1 bit
      char nextChar = BinaryStdIn.readChar();       //readChar   : reads 8 bits
      return new Node(nextChar, null, null);
   }
   return new Node(0, readTrie(), readTrie());
}
```

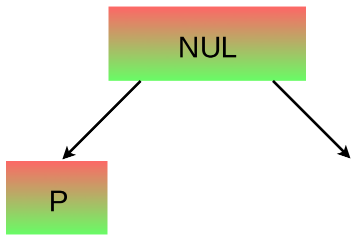
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Suppose that the following bitstream (spread across two lines for your annotating convenience) is read via BinaryStdIn by `readTrie()`. Hint: The underlined part of the bittream represents P.

0 1 **0 1 0 1 0 0 0 0** 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0

1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1

Complete the tree below. Represent each node by the ASCII character stored in `ch`. Your nodes need not have a fancy gradient.



11. **Compression (6 points).**

(a) What is the inverse Burrows Wheeler Transform of the following input?

```
0
Z  H  E  C  E
```

(b) In the blanks below, write the name of the compression algorithm (Huffman or LZW) that performs better for the string in question.

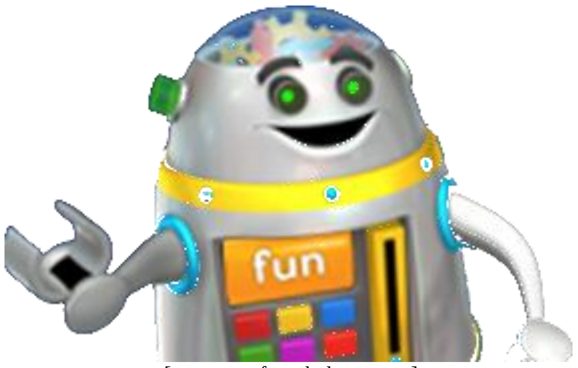
------------ A string of length 1 million consisting of only repeats of the letter `a`.

------------ `abacadaeafagahbcbdbebfbgcacbcccdcecfcgdedfdgefegfg`

(c) One point question (come back later): Given a string consisting solely of L repeats of the 8-bit ASCII letter `a`, consider a version of LZW that replaces the least recently used codeword when the table fills up. Assuming 8 bit codewords, what is the order of growth of the compression ratio for LZW as a function of L?

(d) One point bonus question: Recall that the Kolmogorov complexity $K(s)$ of a string s is the length of the shortest program in some programming language (say, Java, but immaterial for the purposes of this question) that prints s and exits.

Let π_N be the string consisting of the first N decimal digits of π. (For example, π_5 = "31415".) What is the order of growth of $K(\pi_N)$ as a function of N? Provide a short explanation.



[courtesy of: rocknlearn.com]

12. **Graph Algorithm Design (8 points).**

Let `G=(V,E)` be an unweighted, directed graph. Let `s` and `t` be two vertices of `G`. Design an algorithm with `O(V+E)` worst-case running time that finds the number of distinct shortest paths (distinct paths may share some but not all vertices) from `s` to `t`. You may assume there are no parallel edges or self-edges. Precisely list all data structures utilized (e.g. if you are using arrays, list each array and its meaning).

13. **Substring Search Algorithm Design (8 points).**

The Great Firewall of China maintains a list of over a hundred censored keywords. Each TCP packet entering China's borders is searched for these keywords, and the connection is dropped if a certain number of keywords are found in the packet. This is an example of the multiple pattern search problem.

If there are K patterns, each of length M, that we want to search for in a text of length N, a straightforward approach is to search for each pattern separately and count the number of matches. This is K times slower than single pattern search.

Of the three pattern search algorithms we've studied — Knuth-Morris-Pratt, Boyer-Moore, and Rabin-Karp — is there one that can be generalized in a way that's faster than searching for K patterns independently?

(a) Circle your choice of algorithm

KMP Boyer-Moore Rabin-Karp

(b) Describe the data structures you need to build when beginning a search, and how to build them.

(c) Describe the changes you need to make to the search algorithm to handle multiple patterns.

(d) What is the order of growth of the run time to build the data structures required for search? What is the order of growth of the run time required to process a text? Give your answers in terms of N, M, and K.

---

14. Legume Grime Pop (**14 points).**

Congratulations, you've graduated, and have found employment at Peg Leg Emporium, providing the highest quality peg legs at the lowest legal prices.

Sadly, the peg leg business is really hurting, and your boss wants a truly memorable advertising slogan. To this end, she proposes the following task: Design an algorithm for finding the largest **anagram set** in the English language. An **anagram set** is a set of words such that all words are anagrams of one another. For example, {tars, arts, rats, star} is an anagram set.

Your algorithm should run in NL^2 time, where L is the length of the longest English word, and N is the number of words. You may assume you have a text file containing all English words. You do not need to describe how to read in this text file, nor do you need to count reading this text file in your runtime.

Your program should print all members of the largest anagram set to the screen. If there is more than one, you may print any of them. You may use any algorithm discussed in class.

[provide answers on next page]

(a) What data structures do you use (you’ll explain how to build these in part b)? Include the data structure for storing the set of all English words.

(b) Describe the process by which you construct your anagram set. Write the order of growth of the run time of every step of your algorithm in terms of N and L, as well as the total run time (you may continue on to the next page if necessary).

(c) Your marketing campaign was a success, and you got a pretty good high five at last week's meeting, but now the boss has an even crazier idea, slyly mentioning that you might just be able to earn Employee of the Month. Now she wants the longest **anagram ladder** in English. An **anagram ladder** is a sequence of words such that the k+1th word is an anagram of the kth word plus any character in the English alphabet. For example {to, lot, lost, toils, tonsil, lotions, colonist, locations, coalitions, dislocation, conditionals, consolidation, consolidations} is an anagram ladder.

Provide an algorithm (and precisely list any data structures you use) that produces the longest anagram ladder in English. Write the runtime of every step of your algorithm, as well as the total runtime**.** If you built some data structure in part b that is useful to you here, you may reuse it. For full credit complete this task in order of growth NL^2 time or better in the worst case.