COS 226 **Algorithms and Data Structures** **Spring 2016**

Final Exam

This final has 11 questions for a total of 105 points.

You have 3 hours. The exam is closed book, and no calculators or other electronic devices are allowed. You may use one 8.5×11 sheet (two sides) of notes in your own handwriting.

Name:

NetID:

Precept:

Room in which you're taking the exam:

P01	Th 9–9:50am	Maia Ginsburg
P02	Th 10–10:50am	Shivam Agarwal
P02A	Th 10–10:50am	Marc Leef
P03	Th 11–11:50am	Maia Ginsburg
P03A	Th 11–11:50am	Ming-Yee Tsang
P04	Th 12:30pm–1:20pm	Miles Carlsten
P05	Th 1:30pm–2:20pm	Sergiy Popovych
P06	F 10–10:50am	Andy Guna
P07	F 11–11:50am	Andy Guna
P07A	F 11–11:50am	Harry Kalodner
P99	M 7:30–8:20pm	Andy Guna

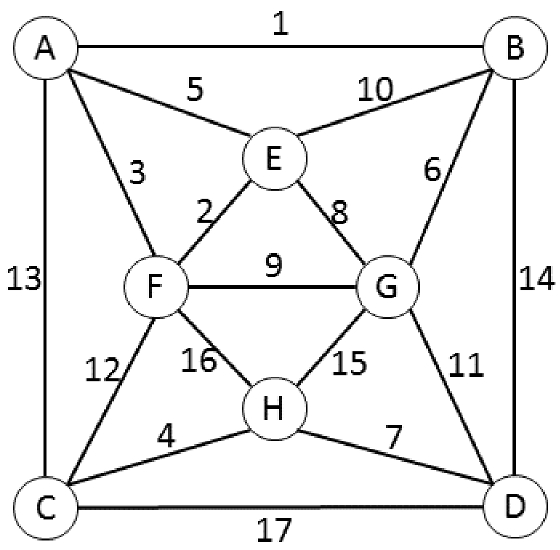
Write and sign:

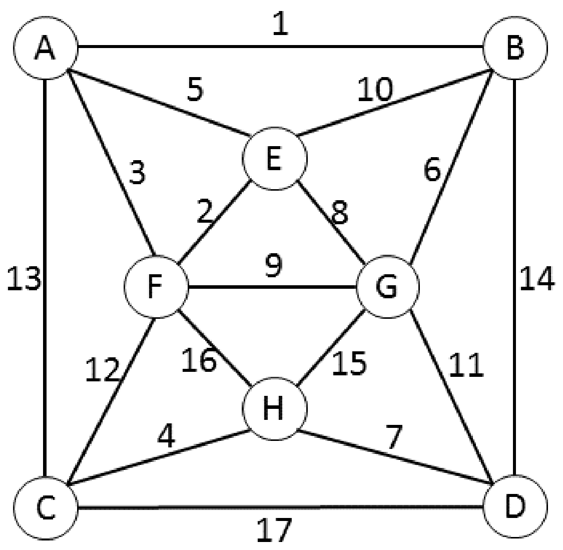
"I pledge my honor that I have not violated the Honor Code during this examination."

Optional: mark how you feel at the beginning of the test... ...and at the end

1. **Minimum Spanning Trees (8 points).**

Consider the following edge-weighted graph with 8 vertices and 17 edges. Note that the edge weights are distinct integers between 1 and 17. There are two copies of the graph in case you want to draw over them to work out the answer.





(a) Complete the sequence of edges in the MST in the order that Kruskal's algorithm includes them (by specifying their edge weights).

1 2 3 4 6 7 11

(b) Complete the sequence of edges in the MST in the order that Prim's algorithm includes them (by specifying their edge weights).

1 3 2 6 11 7 4

(c) If the weight of the A-B edge were increased to x then it would no longer be in the MST (circle all values of x that apply):

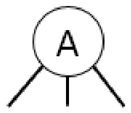
1.5 4.5 7.5 10.5

(d) If the weight of the C-F edge were decreased to y then it would be in the MST (circle all values of y that apply):

1.5 4.5 7.5 10.5

2. Ternary Search Trees and Tries (8 points).

Recall: the height of the tree is the maximum path length from the root to a leaf. For example, a TST with just the string `'A'` in it has a height of 1 because all paths from the root to a leaf (i.e., a null node) have length 1.



Consider the following set of strings:

`ARVI, GUNA, HARR, MAIA, MARC, MILE, MING, SERG, SHIV`

(a) Suppose that these strings are inserted into a TST in an order which minimizes its height. What is the height of the resulting TST?

6

(b) Suppose that they are instead inserted into a TST in an order which maximizes its height. What is the height of the resulting TST?

10

(c) Instead of TSTs, suppose we want to use an R-way trie ($R = 26$). What is the minimum and maximum height of the resulting trie?

5, 5

3. Analysis of algorithms (5 points).

Suppose that you collect the following running time data for a program as a function of the input size N.

N	Running time
1,000	0.10s
8,000	0.41s
64,000	1.61s
512,000	6.42s

Compute the approximate running of the program as a function of N and use tilde notation to simplify your answer. Hint: recall that $\log_b a = \frac{\lg a}{\lg b}$.

Assume $T(N) \sim a\,N^b$. Then $\frac{T(8000)}{T(1000)} \approx 8^b$.

From the table, $\frac{T(8000)}{T(1000)} \approx 4$.

$\Rightarrow\ 8^b \approx 4$

$\Rightarrow\ b \approx \frac{\log 4}{\log 8} = \frac{2}{3}$.

From the table, $a \cdot 1000^{2/3} \approx 0.1$

$\Rightarrow\ a \approx \frac{0.1}{100} = 10^{-3}$.

4. Finite automata (10 points).

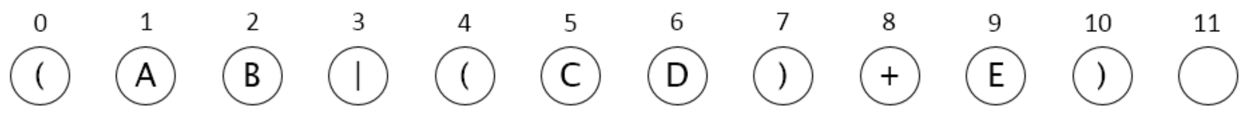
(a) Below is a partially-completed Knuth-Morris-Pratt DFA for a string of length 6 over the alphabet {A, B, C}. State 6 is the accept state. Fill in all the missing cells in the table.

j	0	1	2	3	4	5
A	0	2	0	0	5	0
B	1	1	1	4	1	6
C	0	0	3	0	0	3

List the string that the DFA searches for: BACBAB.

(b) Show the NFA corresponding to the regular expression `( AB | (CD)+ E )` generated by the procedure described in class. Recall that the | operator has the lowest precedence.

To show the NFA, write the list of match transitions and the list of epsilon transitions in the space below. You can draw arrows to visualize the NFA if you like, but we will ignore that for grading.



Match transitions:

$1 \rightarrow 2$, $2 \rightarrow 3$, $5 \rightarrow 6$, $6 \rightarrow 7$, $9 \rightarrow 10$

Epsilon transitions:

$0 \rightarrow 1$, $0 \rightarrow 4$, $3 \rightarrow 10$, $4 \rightarrow 5$, $7 \rightarrow 8$, $8 \rightarrow 4$, $8 \rightarrow 9$, $10 \rightarrow 11$

(c) List all the states the NFA can be in after reading the string CD.

4, 5, 7, 8, 9

5. Regular expressions (10 points).

(a) Consider the regular expression

(A*B | BA*C) D*

Circle all the strings that it matches among the following:

A B C D AB BC CD BD ABD BCD

(b) Write a regular expression corresponding to each of the following descriptions. If there is no such regular expression, write *Impossible*. Note: if the alphabet is not specified, then your answer has to be valid regardless of the alphabet.

- Strings containing an odd number of A's (and no other characters).

  A(AA)*

- Strings over the alphabet {A, B, C} not containing the substring AB.

  (B | A*C)* A*

- Strings containing more A's than B's.

  Impossible

- Strings matching the substring ABA at 2 or more distinct start indices. For example, the string ABACABAD matches the substring ABA at start indices 0 and 4, which are distinct.

  .* AB (A | A.*A) BA .*

6. Data compression (15 points).

(a) What is the approximate compression ratio achieved by the following algorithms and inputs? For Huffman and LZW, assume that the input is a sequence of 8-bit characters ($R = 256$). Recall that the compression ratio is the number of bits in the compressed message divided by the number of bits in the original message.

Assume an extremely long message. In particular, this means that the size of the trie in Huffman coding is negligible and that the LZW codeword table will get filled up.

[Continued from previous page] Fill in each blank with one of these possible answers A-P:

A: ≈ 16	B: ≈ 8	C: ≈ 1	D: $\approx \frac{3}{4}$	E: $\approx \frac{5}{16}$	F: $\approx \frac{1}{4}$	G: $\approx \frac{3}{16}$	H: $\approx \frac{1}{8}$
I: $\approx \frac{16}{255}$	J: $\approx \frac{1}{16}$	K: $\approx \frac{8}{255}$	L: $\approx \frac{1}{255}$	M: $\approx \frac{1}{256}$	N: $\approx \frac{1}{1280}$	O: $\approx \frac{1}{2560}$	P: $\approx \frac{1}{3840}$

I
__________ Run-length coding with 8-bit counts for a binary string of all 0's.

B
__________ Run-length coding with 8-bit counts for a binary string of alternating 0's and 1's.

G
__________ Huffman coding for the string `aabcaabcaabc`…

C
__________ Huffman coding for a random string.

N
__________ LZW coding for the string ababab… using 12-bit codewords.
Recall: no new codewords are added to the table if it already has $2^{12} = 4096$ entries. Initially there are 256 codewords, so there is room for 3840 more.

D
__________ LZW coding for a string of $\frac{N}{2}$ a's followed by $\frac{N}{2}$ b's using 12-bit codewords.

(b) What is the Burrows-Wheeler transform of `SAMIAM`? Circle your final answer.

5
SIMAAM

(c) What is the Burrows-Wheeler inverse transform of the following? Circle your final answer.
0
GKENPKOIOBE

BOOKKEEPING

7. Magical algorithms (8 points).

A wizard claims to be able to modify the complexity of a number of computational tasks.

(a) Mergesort's merge operation now takes constant time. What is mergesort's new worst case order-of-growth running time?
$O(N)$

(b) All priority queue operations now take constant time. What is Dijkstra's algorithm's new worst case order-of-growth running time? Assume a connected graph.
$O(E + V)$

(c) All priority queue operations now take constant time. What is Heapsort's new worst case order-of-growth running time?
$O(N)$

(d) Sometimes magic goes wrong. Enqueueing into a queue now takes logarithmic time (in the size of the queue), but dequeueing remains constant time. What is the new worst case order-of-growth running time of BFS? Assume a connected graph.
$O(E + V \log V)$

8. Assignments (9 points).

(a) Find the best-case order-of-growth running time for each of the following problems. As a reminder of how best-case analysis works, recall that in Wordnet, the best-case running time of finding the shortest common ancestor if you used `BreadthFirstDirectedPaths` was $O(V)$ because you initialized and iterated over the `distTo` array values to find the answer. You should consider any necessary initialization costs in your analysis.

(i) Finding a directed cycle in a digraph.
$O(V)$

(ii) Computing a topological sort in a digraph.
$O(V)$

(iii) Computing a topological sort in a digraph that's known to be strongly connected.
$O(1)$

(iv) Computing the minimal spanning tree using Kruskal's algorithm.
$O(E + V \log E)$

(b) Seamcarver. Below are the energies for a 4x6 image. The seam is shown by underlines. It is possible to recalculate only some of the energies after removing the marked seam. Mark the pixels that would have to be recalculated with an R and the pixels that would be moved with an M. (Three cells are already marked.)

241.26	262.86 RM	218.08 M	219.32 RM
130.86	275.66 RM	173.21 M	332.31 RM
286.21 R	173.21	321.01 RM	211.20 M
201.53 R	171.80	195.63 RM	310.70 M
127.53	270.93 RM	188.15 M	189.61 RM
316.45 R	176.88	326.63 RM	248.93 M

9. Analysis of algorithms II (10 points).

Give the order-of-growth running time of each function f1−f4 as a function of N:

```
public static int f1(int N) {
     int sum = 0;
     for (int i = 0; i * i < N; i++)
          for (int j = 0; j * j < N; j++)
               for (int k = 0; k < N; k++)
                    sum ++;
     return sum;
}
```

Running time: N^2

```
public static int f2(int N) {
     int sum = 0;
     for(int i = 0; i < N; i++)
          for(int j = 0; j < N * N; j++)
               for(int k = 0; k < j; k++)
                    sum++;
     return sum;
}
```

Running time: N^5

```
public static int f3(int N) {
     return g(N, N);
}

public static int g(int N, int K) {
     int sum = 0;
     if (N==0) return 1;
     for(int i = 1; i < K; i *= 2)
          sum += g(N-1, K)
     return sum;
}
```

Running time: $(\lg N)^N$

```
public static int f4(int N) {
     int sum = 0;
     for (int i = 0; i * i < N; i++)
          for (int j = 1; j < i; j *= 2)
               sum ++;
     return sum;
}
```

Running time: $\sqrt{N} \lg N$

10. COS 226 (12 points).

(a) When creating this course, imagine that Bob Sedgewick and Kevin Wayne made a digraph of dependencies between lectures. What algorithm could they have used to order the lectures in such a way that no lecture is dependent on lectures that come after it?

Topological sort (reverse DFS postorder).

(b) The instructors plan to grade the final tomorrow, Friday May 20. They have scheduled 8 hours for it. Each instructor prefers to grade only a certain subset of the 11 problems. They have represented their preferences in a graph in which a grader is connected to a problem if they are willing to grade it.

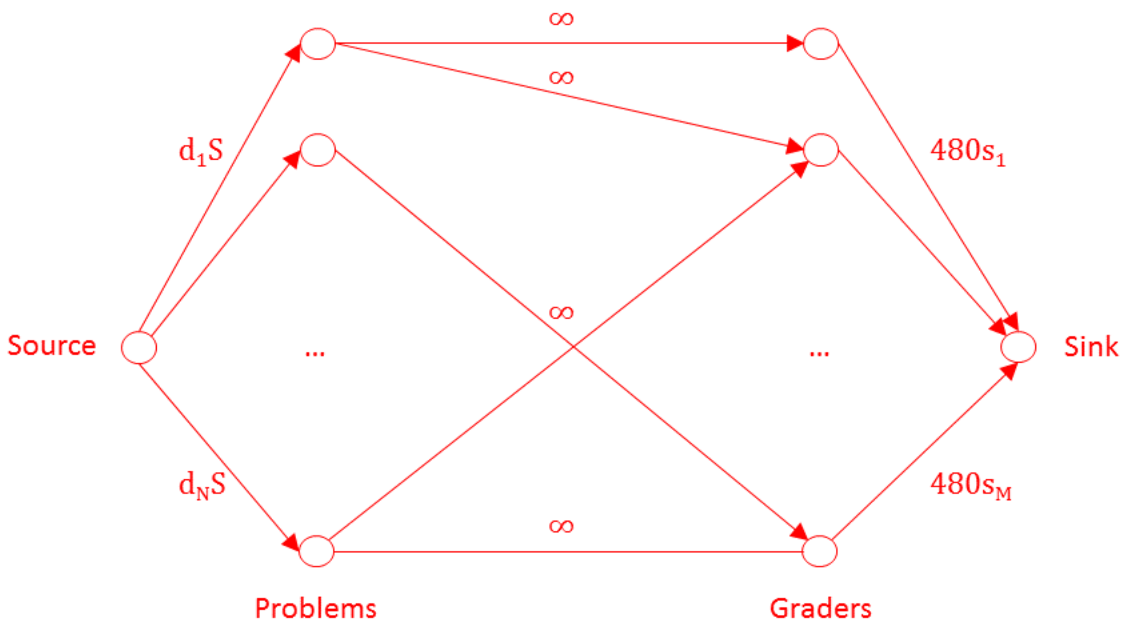
Each grader has a grading speed, and each problem has a grading difficulty. If a problem with difficulty d is graded by an instructor with speed s, it will take d/s minutes to grade. There are N problems and M instructors; S students took the exam.

[Continued from previous page] Design an algorithm to determine if the instructors will be able to finish grading within the scheduled 8 hours, respecting their grading preferences.

What is the order-of-growth worst-case running time of your algorithm in terms of M and N? Assume that the speeds and difficulties are bounded by a constant.

Hint: formulate it as a max-flow problem and invoke a known algorithm for solving max-flow. You might want to draw a graph to illustrate how you construct a max-flow problem.

Create a graph with one vertex for each problem and one for each as shown. The source $\rightarrow$ problem capacities represent the grading effort needed for each problem. The grader $\rightarrow$ sink capacities represent the grading capability of each instructor in 8 hours. The problem $\rightarrow$ grader edges are as in the input graph.



Check if the max flow equals $\sum_{i=1}^{N} d_i S$ using the Ford-Fulkerson algorithm. If it is, it means that the grading effort needed for each problem can be partitioned among the instructors without any instructor exceeding their 8-hour grading capability.

The running time is $O(VE^2)$ with the shortest-augmenting-path heuristic. Here $V = M + N + 2$ and $E \leq MN + M + N$, so the running time is $O(M^2N^2(M + N))$.

11. Reductions (10 points).

We define the Traveling Tourist Problem (TTP) as the following:

Given a weighted digraph G and a threshold t, determine if there is a cycle that visits each vertex in G at least once and has a total weight $\leq t$. You can think of a tourist who wishes to visit each of V cities at a total cost of at most t, the edge weights representing the cost of traveling from one city to another.[1]

For each of the following problems, construct a linear-time reduction of the problem to TTP. Note: it's enough to describe the reduction; you don't need to prove its correctness.

(a) TTP with the added assumption that the tourist incurs a city-specific cost penalty each time they visit a city that they have previously visited. In other words: given a weighted digraph G, a threshold t, and a penalty function $p(v)$, determine if there is a cycle that visits each vertex in G and has a total weight + total penalty $\leq t$. The penalty for visiting vertex v each time after the first is $p(v)$.

The problem reduces to TTP on a modified graph and threshold:

For each v, add weight $p(v)$ to every edge adjacent to v.

Add $\sum_v p(v)$ to the threshold.

(b) TTP with the restriction that each city be visited exactly once.

Hint: it is sufficient to reduce it to the problem in part (a).

Set the penalty for every vertex to $1 + t$.

[1] If you're familiar with the Traveling Salesperson Problem, beware that this definition is slightly different.

(c) [Extra credit; 2 points]. TTP but with the ability for the tourist to teleport — jump from the current city to any other, whether or not the pair is connected by an edge — at most once during the tour. The tourist doesn't have to teleport, but if they do, they will incur a (constant) cost penalty of p.

Solve these two instances of TTP and answer 'yes' if either answer is 'yes':

1. With the graph and threshold as given.
2. With a modified graph and threshold:
    - add a virtual vertex that is adjacent to and from every vertex
    - set edge weights of this vertex to $p + t$ in one direction and 0 in the other
    - set the threshold to $2t$.

Use this space for scratch. And don't forget to write your info and sign the honor code on the front page!

Problem	Score
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
Total	