Spring 2013 **CMSC 351: Final** Hamid Mahini
Time: 2 Hours

Type A

WAIT FOR INSTRUCTIONS BEFORE BEGINNING

HONOR PLEDGE: "I pledge on my honor that I have not given or received any unauthorized assistance on this examination."

Signature and UID: _____

Print name: _____

- ***Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should PROVE the correctness of your algorithms either directly or by referring to a proof in the book.***
- ***The sum of the grades is 110, but your grades would be out of 100 (thus you get 10 bonus points by solving all the problems).***
- ***Select the best choice for the first 5 problems and mark it by X in the table below***

Problem	1	2	3	4	5	6	7	8
a						X		
b		X			X			
c				X				
d			X				X	
e	X							X

DO NOT WRITE BELOW THIS LINE

Problem 1-8:	
Problem 9:	
Problem 10:	
Problem 11:	
Problem 12:	
TOTAL:	

Multiple-choice Problems (Answer ONLY in the TABLE in the FIRST PAGE and NOT HERE):

Problem 1. **(5 points)** Let $T(n) = 3T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 5n$ and $T(0) = 7$. Then $T(n)$ is?

a) Θ(n) b) $\Theta(n^{log_2^3} \log n)$ c) $\Theta(n \log n)$
d) $\Theta(n^{1.5})$ e) $\Theta(n^{log_2^3})$

Problem 2. **(5 points)** Let $f(n) = 3n^2 + logn$, $g(n) = nlogn + nlog^2 n$, and $h(n) = 4n^2 - 5n$. Select the correct answer.

a) $f(n) = O(g(n)), f(n) = O(h(n))$ b) $f(n) = \Omega(g(n)), f(n) = \Theta(h(n))$

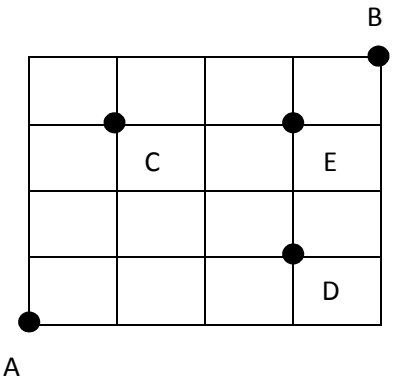
c) $f(n) = O(g(n)), f(n) = \Theta(h(n))$ d) $f(n) = \Theta(g(n)), f(n) = \Omega(h(n))$

e) $None$

Problem 3. **(5 points)** How many ways can you walk from A to B such you do not pass through C, D, and E. In each move you are allowed to go one step up or one step to the right. (Hint: use dynamic programming technique)

a) 8 b) 10 c) 12

d) 14 e) 18



Problem 4. **(5 points)** Problem A is polynomially reducible to problem B. Which of the following statements is correct?

I. If problem A is solvable in a polynomial time then problem B is solvable in polynomial time.

II. If problem A is NP-complete then problem B is NP-complete.

III. If problem A is NP-hard then problem B is NP-hard

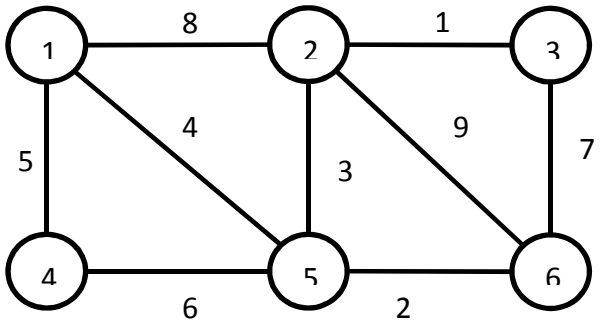
a) I b) II c) III
d) I and II e) I, II, and III

Problem 5. (5 points) For dense graphs with $\Omega(n^2)$ weighted edges what is the best running time for Prim's algorithm?

a) $O(nlogn)$ b) $O(n^2)$ c) $O(n^2logn)$ d) $O(n^3)$ e) $O(nlog^2n)$

Problem 6. (5 points) Consider the weighted graph below. Run Dijkstra's algorithm starting from vertex 1. Write the vertices in the order which they are marked.

a) 1, 5, 4, 6, 2, 3
b) 1, 5, 4, 2, 6, 3
c) 1, 5, 6, 2, 4, 3
d) 1, 5, 4, 2, 3, 6
e) 1, 5, 6, 2, 3, 4



Problem 7. (5 points) Consider the weighted graph of Problem 6. Run Prim's algorithm starting from vertex 1. Write the vertices in the order which they are marked.

a) 1, 5, 4, 6, 2, 3

b) 1, 5, 2, 3, 6, 4

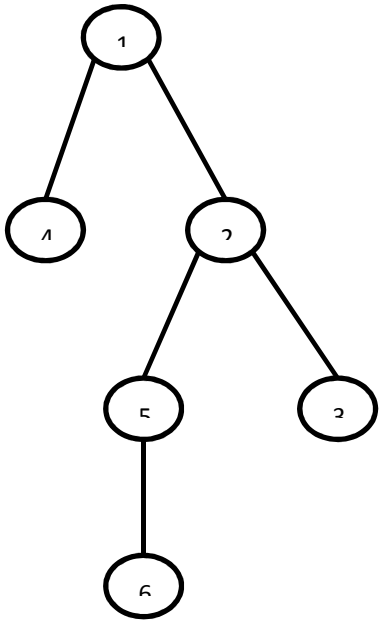
c) 1, 5, 6, 3, 2, 4

d) 1, 5, 6, 2, 3, 4

e) 1, 5, 4, 2, 6, 3

Problem 8. (5 points) Consider tree T below. Assume tree T is the DFS tree of undirected graph G. Which of the following cannot be an edge of graph G?

a) (1, 5)
b) (2, 6)
c) (1, 6)
d) (1, 3)
e) (4, 3)



Regular Problems:

Problem 9. (20 points) The longest Increasing Subsequence (LIS) problem is to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order. For example, length of LIS for (10, 22, 9, 33, 21, 50, 41, 60, 80) is 6 and LIS is (10, 22, 33, 50, 60, 80). Design an algorithm to find the longest increasing subsequence of a given sequence. The running time of your algorithm should be $O(n^2)$.

Remark 1: The first element of the sequence need not belong to the LIS. For example, consider the sequence (25, 8, 22, 9, 13). The LIS is (8, 9, 13).

Remark 2: Many of you thought about the following algorithm: from each number, find the maximum increasing subsequence starting at that number and then maximum of these lengths. For a fixed number, you tried to find the longest increasing subsequence starting at that number by greedily appending larger numbers as you see them. Again this does not work. Consider the sequence (8, 22, 9, 13). According to the greedy choice the longest increasing subsequence starting at 8 is (8,22). But the actual LIS is (8,9,13).

We solve the problem using dynamic programming. Let the sequence be $(s_1, s_2, \dots, s_n)$. For $1 \leq i \leq n$ let L_i be the length of the longest increasing subsequence of $(s_1, s_2, \dots, s_i)$ that ends in s_i.

Base Case: $L_1 = 1$

Recurrence: $L_i = 1 + \max_{\{j<i \,\&\, s_j<s_i\}} L_j$. (the 1 comes from the s_i at the end)

The final answer is $\max_{1\leq i\leq n} L_i$. This gives the length of the LIS. It is easy to see that we can also construct the LIS by maintain the current longest increasing subsequence at each of the above steps.

Let us now analyze the running time of the algorithm. To compute L_i we need $i-1$ comparisons. Therefore the total number of comparisons is $1 + 2 + 3 + \cdots + (n-1) = O(n^2)$.

Problem 10. (20 points) Let *D* be the shortest path matrix of weighted graph *G*. It means *D[u,v]* is the length of shortest path from vertex u to vertex v, for every two vertices *u* and *v*. Graph *G* and matrix *D* are given. Assume the weight of an edge *e* is decreased from w_e to w'_e. Design an algorithm to update matrix *D* with respect to decrement of weight of edge *e*. The running time of your algorithm should be $O(n^2)$.

Assume graph is undirected. Let $e = \{a, b\}$. For every pair of vertices i, j let $D(i,j), D'(i,j)$ be the old, new entries respectively. Now we have two possibilities: either the new shortest path from i to j can pass through e, or not. Therefore, $D'(i,j) = \min\{\, D(i,j)\,;\ D(i,a) + w'_e + D(b,j);\ D(i,b) + w'_e + D(a,j)\}$. Since we can update each entry in constant time, the whole algorithm runs in $O(n^2)$ time.

Note: If graph is directed and $e = (a, b)$ then$D'(i,j) = \min\{\, D(i,j)\,;\ D(i,a) + w'_e + D(b,j)\}$

Problem 11. (15 points) Given two sets S_1 and S_2 of natural numbers, and a natural number a. Find whether there exists an element x from S_1 and an element y from S_2 such that $2x + 5y = a$. The algorithm should run in time $O(n)$, where n is the total number of elements in both sets.

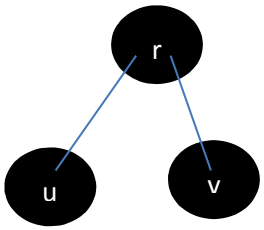
This is very similar to Problem 1 from Homework 4.

Use a Hash data structure to store all the members of S_2. For every member $x \in S_1$, we simply search for $\frac{a-2x}{5}$ in the data structure. In a hash data structure the running time of insertion and search are O(1) in average , thus the algorithm runs in O(n) time in average

Problem 12. (15 points) Let $d[v]$ be the discovery time and $f[v]$ be the finishing time of vertex v in DFS. Prove that it is not possible to find two vertices v and u such that $d[u] < d[v] < f[u] < f[v]$.

$d[u] < d[v]$ means that we discover u before v. Now we have two cases either:

1. u is an ancestor of v
2. u is not an ancestor of v (many students have missed considering this case; see figure below)



If Case 1 happens then we know that every predecessor of u finishes before finishing u as DFS is a recursive function which contradicts with the fact that $f[u] < f[v]$.

If Case 2 happens then we know that u had to be finished by the discovery of v which contradicts with the fact that $d[v] < f[u]$.

Therefore the only valid sequences are : d[u] < d[v] < f[v] < f[u] or d[u] < f[u] < d[v] < f[v].