CS570
Analysis of Algorithms
Spring 2012
Exam I

Name: _____

Student ID: _____

____12:30 – 1:50 session ____ 2:00 – 3:20 session

	Maximum	Received
Problem 1	20	
Problem 2	20	
Problem 3	20	
Problem 4	20	
Problem 5	10	
Problem 6	10	
Total	100	

2 hr exam
Close book and notes

If a description to an algorithm is required please limit your description to within 150 words, anything beyond 150 words will not be considered.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[FALSE]
Dijkstra's algorithm solves the Minimum Spanning Tree problem.

[TRUE]
Let e be a maximum-weight edge on some cycle of $G = (V, E)$. Then there is a minimum spanning tree of $G' = (V, E - \{e\})$ that is also a minimum spanning tree of G.

[TRUE]
Let e be a minimum-weight edge in a graph G. Then e must belong to some minimum spanning tree of G.

[FALSE]
Given a directed graph G with n vertices, any BFS tree corresponding to graph G will have n-1 edges.

[FALSE]
Given an undirected graph G with n vertices, any BFS tree corresponding to graph G will have n-1 edges.

[TRUE]
A constant $n_0 \geq 1$ exists such that for any $n \geq n_0$, there is an array of n elements such that insertion sort runs slower than merge sort on that input.

[TRUE]
The array [20, 15, 18, 7, 9, 5, 12, 3, 6, 2] forms a MaxHeap.

[FALSE]
Having done two BFS and DFS searches starting from the same point s in graph G, the DFS tree is never as the same as the BFS tree.

[TRUE]
A greedy algorithm always makes the choice that looks best at the moment.

2) 20 pts

a- Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge (i.e., a unique edge of smallest cost) crossing the cut.

Definition: a cut in a graph divides the nodes of a graph into two sets. For example, in class we talked about the node sets S and V-S when discussing some MST algorithms. The boundary between S and V-S is an example of a cut.

Solution:

Assume for a contradiction that the graph has two distinct minimum spanning trees T and T'. Let (u, v) be an edge in T which is not in T'. Removing edge (u, v) cuts the tree T into two components (trees). Let T_u and T_v be the vertices in the component containing u and v, respectively. Consider the cut (T_u, T_v) and let (x, y) be the unique light edge crossing the cut. If $(x, y) \neq (u, v)$ then $w(x, y) < w(u, v)$ and the spanning tree $T - \{(u, v)\} \cup \{(x, y)\}$ has a better cost than T, a contradiction.
Hence, assume that $(x, y) = (u, v)$, i.e., the unique light edge (u, v) crossing the cut (T_u, T_v) doesn't belong to T'. Consider the path p from u to v in T'. Path starts in T_u and ends in T_v, hence there must be an edge e on it crossing the cut (T_u, T_v) (there might be several of them, but take one). As (u, v) is the unique light edge crossing this cut, $w(u, v) < w(e)$. If we add (u, v) to T', we get a cycle composed of (u, v) and p. By removing any edge from the cycle we get again a spanning tree. Hence $T' \cup \{(u, v)\} - \{e\}$ is a spanning tree and by above it has a better cost than T', again a contradiction.

b- Show that the converse is not true, i.e. it is possible for a graph to have non-unique minimum cost edges crossing a cut and still only have one unique MST.

Solution:

This can be proved by the following counterexample for the converse. Consider a graph with 3 vertices a, b, c and weights $w(a, b) = w(a, c) = 1$ and $w(b, c) = 2$.
The graph has a unique minimal spanning tree (containing edges (a, b) and (a, c)), however cut $(\{a\}, \{b, c\})$ doesn't have a unique light edge crossing the cut.

3) 20 pts

You are given n events where each takes one unit of time. Event i will provide a profit of g_i dollars ($g_i > 0$) if started at or before time t_i where t_i is an arbitrary real number. (Note: If an event is not started by t_i then there is no benefit in scheduling it at all. (All events can start as early as time 0.) Given the most efficient algorithm you can to find a schedule that maximizes the profit.

Solution:

First, we sort the jobs according to $\lfloor t_i \rfloor$ (sorted from largest to smallest). Let time t be the current time being considered (where initially $t = \lfloor t_1 \rfloor$). All jobs i where $\lfloor t_i \rfloor = t$ are inserted into a priority queue with the profit g_i used as the key. An extractMax is performed to select the job to run at time t. Then t is decremented and the process is continued. Clearly the time complexity is O(nlog n). The sort takes O(n log n) and there are at most n insert and extractMax operations performed on the priority queue, each which takes O(log n) time.

4) 10 pts

Consider two positively weighted graphs $G = (V, E, w)$ and $G' = (V, E, w')$ with the same vertices V and edges E such that, for any edge e in E, we have $w'(e) = w(e)^2$.
Prove or disprove: For any two vertices u, v in V , any shortest path between u and v in G' is also a shortest path in G.

Solution:

False. Assume we have two paths in G, one with weights 2 and 2 and another one with weight 3. The first one is shorter in G' while the second one is shorter in G.

5) 10 pts

Assume that A is a very large unsorted array of integers. Describe an algorithm that picks the largest *m* integers in A, where *m* is small relative to the size of the array (*n*). The time complexity should be less than $O(nlgn)$, i.e. sorting the array is not going to be fast enough. Also, the amount of additional memory that you are given is O(1).

Solution:

Two ways solving this problem is provided as follows:
The first one to just do *m* linear search. Since $m<<n$, the time complexity $O(mn)$ is $O(n)$.
The second one is to max-heapify the array, then do *m* extract-max. It takes $O(n)$ + $O(mlogn)$, less than $O(nlogn)$.

6) 20 pts
   a- Describe how Strassen's algorithm is capable of reducing the complexity of dense matrix multiplication. You do not have to provide exact formulas. You only need to describe the approach and to do complexity analysis on the resulting algorithm.

Solution:

Please refer to the lecture notes or the following link for description and complexity analysis for Strassen's algorithm: http://en.wikipedia.org/wiki/Strassen_algorithm

b- Solve the following recurrences by giving tight Θ-notation bounds. You do not need to justify your answers, but any justification that you provide will help when assigning partial credit.

i. $T(n)=4T(n/2)+n^2 \log n$

Solution:

Case 2 of the Master Method: $T(n)=n^2 \log^2 n$

ii. $T(n)=8T(n/2)+n\log n$

Solution:

Case 1 of the Master Method: $T(n)=n^3$

iii. $T(n)=(6006)^{1/2} * T(n/2)+n^{\sqrt{6006}}$

Solution:

Case 3 of the Master Method to obtain $T(n)=\Theta(n^{\sqrt{6006}})$, checking that the regularity condition $af(n/2)= (6006)^{1/2}(n/2)^{\sqrt{6006}}+n^{\sqrt{6006}} \leqq c\, n^{\sqrt{6006}}$ for some $c < 1$. Condition holds for any $c > (6006)^{1/2}/ 2^{\sqrt{6006}}$