CS570
Analysis of Algorithms
Fall 2014
Exam I

Name: _____

Student ID: _____

Wednesday Evening Section

	Maximum	Received
Problem 1	20	
Problem 2	16	
Problem 3	16	
Problem 4	16	
Problem 5	16	
Problem 6	16	
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.

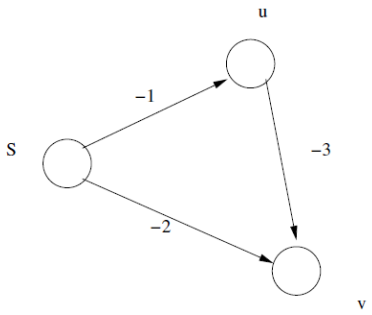
1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
Let T be a complete binary tree with n nodes. Finding a path from the root of T to a given vertex $v \in T$ using breadth-first search takes $O(logn)$.

False :
BFS requires $O(n)$ time. BFS examines each node in the tree in breadth-first order. The vertex v could well be the last vertex explored (Also, notice that T is not neceassarily stored).

[TRUE/FALSE]
Dijkstra's algorithm works correctly on a directed acyclic graph even when there are negative-weight edges.

False:
Consider Dijkstra's algorithm run from source *s* in the following graph. The vertex *v* will be removed from the queue with *d*[*v*] = -2 even though the shortest path to it is -4.



[TRUE/FALSE]
If the edge e is not part of any MST of G, then it must be the maximum weight edge on some cycle in G.

True:
Take any MST T. Since e is not part of it, e must complete some cycle T. e must be the heaviest edge on that cycle. Otherwise a smaller weight tree could be constructed by swapping the heavier edge on the cycle with e and thus T cannot be MST.

[TRUE/FALSE]
If $f(n) = O(g(n))$ and $g(n) = O(f(n))$ then $f(n) = g(n)$.

False:
f(n)= n and g(n)= n +1.

[TRUE/FALSE]
The following array is a max heap: [10; 3; 5; 1; 4; 2].
False:The element 3 is smaller than its child 4, violating the maxheap property.

[TRUE/FALSE]
There are at least 2 distinct solutions to the stable matching problem: one that is preferred by men and one that is preferred by women.
False:
Consider the example: two men X and Y; two women A and B.
Preference list (decreasing order):
X's list: A, B ; A's list: X, Y;
Y's list: A, B; B's list: X, Y.
The matching is unique: X-A and Y-B

[TRUE/FALSE]
In a binary max-heap with n elements, the time complexity of finding the second largest element is O(1).
True:
The second largest one should be the larger (or equal) one of the two children of the root node. Finding them takes time O(1). Note: it is possible that several elements have the same value including the first two layer of elements, then in this case, the largest element is equal to the second largest element, and returning one of the second layer elements is also fine.

[TRUE/FALSE]
Given a binary max-heap with n elements, the time complexity of finding the smallest element is O(lg n).
False:
The smallest element should be among the leaf nodes. Consider a full binary tree of n nodes. It has (n+1)/2 leafs (you can think of why). Then the worst case of finding the smallest element of a full binary tree (heap) is \Theta(n)

[TRUE/FALSE]
Kruskal's algorithm can fail in the presence of negative cost edges.
False:
The MST problem does not change even with the negative cost edges. So the Kruskal algorithm still works.

[TRUE/FALSE]
If a weighted undirected graph has two MSTs, then its vertex set can be partitioned into two, such that the minimum weight edge crossing the partition is not unique.
True: Let T1 and T2 be two distinct MSTs. Let e1 be an edge that is in T1 but not in T2. Removing e1 from T1 partitions the vertex set into two connected components. There is a unique edge (call it e2) that is in T2 that crosses this partition. By minimum-weight property of T1 and T2, both e1 and e2 should be of the same weight and further should be of the minimum weight among all edges crossing this partition.

2) 16 pts

The diameter of a graph is the maximum of the shortest paths' lengths between all pairs of nodes in graph G. Design an algorithm which computes the diameter of a connected, undirected, unweighted graph in $O(mn)$ time, and explain why it has that runtime.

Solution:

Suppose we suspected that a particular vertex v was an endpoint of the diameter of the graph. Since this graph is unweighted graph, we can use BFS to find the shortest path from any particular node v and report the largest layer number reached.

However, while we don't know any particular vertex v to be part of this, we do know that there is at least one vertex in the graph for which this is true. Accordingly, we can run BFS from every vertex v, and report the maximum layer reached in all of these runs.

The total time taken is O(n(m + n)); O(m + n) for a single breadth-first search, which is run O(n) times.

But notice that, when the graph is connected, n is O(m). Accordingly, O(m + n) is O(m), for a total runtime of O(mn), as desired.

3) 16 pts

Consider a stable marriage problem where the set of men is given by $M = \{m_1, m_2, ..., m_N\}$ and the set of women is $W = \{w_1, w_2, ..., w_N\}$. Consider their preference lists to have the following properties:

$$\forall w_i \in W: w_i \text{ prefers } m_i \text{ over } m_j, \forall\, j > i$$
$$\forall m_i \in M: m_i \text{ prefers } w_i \text{ over } w_j, \forall\, j > i$$

Prove that a unique stable matching exists for this problem.

Note: symbol $\forall$ means "for all".

We will prove that matching S where m_i is matched to w_i for all $i = 1, 2, \ldots, N$ is the unique stable matching in this case. We will use the notation $S(a) = b$ to denote that in matching S, a is matched to b.

It is evident that S is a matching because every man is paired with exactly one woman and vice versa. If any man m_j prefers w_k to w_j where $k < j$, then such a higher ranked woman w_k prefers her current partner to m_j. Thus, there are no instabilities and S is a stable matching. Now let's prove that this stable matching is unique.

By way of contradiction, let's assume that another stable matching S', which is different from S, exists. Therefore, there must exist some i for which $S'(w_i) = m_k, k \neq i$. Let x be the minimum value of such an i. Similarly, there must exist some j for which $S'(m_j) = w_l, j \neq l$. Let y be the minimum value of such a j. Since $S'(w_i) = m_i$ for all $i < x$, and $S'(m_j) = w_j$ for all $j < y$, $x = y$. $S'(w_x) = m_k$ implies $x < k$. Similarly, $S'(m_y) = w_l$ implies that $y = x < l$.
Given the preference lists, $m_y = m_x$ prefers w_x to w_l, and w_x prefers m_x to m_k. This is an instability and hence, S' cannot be a stable matching.

4) 16 pts

Ivan is a businessman and he has several large containers of fruits to ship. As he has only one ship he can transport only one container at a time and also it takes certain fixed amount of time per trip. As there are several varieties of fruits in the containers, the cost and depreciation factor associated with each container is different. He has n containers, $a_1, a_2, \ldots, a_n$. Initial value of each container is $v_1, v_2, \ldots, v_n$ and depreciation factor of each container is $d_1, d_2, \ldots, d_n$. So, if container a_i happens to be the j^{th} shipment, its value will be $v_i /(d_i \times j)$. Can you help Ivan maximize total value of containers after depreciation ($\sum v_i /(d_i \times j)$) by providing an efficient algorithm to ship the containers? Provide proof of correctness and state the complexity of your algorithm.

Solution: Ship the containers in decreasing order of v_i / d_i. We prove the optimality of this algorithm by an exchange argument.

Thus, consider any other schedule. This schedule should consist an inversion. Further, in fact, there must be an adjacent such pair i, j. Note that for this pair we have

$v_i / d_i <= v_j / d_j$, for $i < j$. If we can show that swapping this pair i, j does not decrease the total value, then we can iteratively do this until there are no more inversions, arriving at the greedy schedule without having decreased the value we are trying to maximize. It will then follow that our greedy algorithm is optimal.

Consider the effect of swapping i and j. The schedule of shipping all the other containers remains the same. Before the swap, the contribution of i and j to the total value was $(v_i / (d_i * i) + v_j / (d_j * j))$, while after the swap the total value is $(v_i / (d_i * j) + v_j / (d_j * i))$. The difference between value after the swap is $(v_j / d_j - v_i / d_i) * (1/i - 1/j)$. As both, $(v_j / d_j - v_i / d_i)$ and $(1/i - 1/j)$ terms are positive, the total value is not decreased by swapping as desired.

The complexity of this solution is O(n logn), for sorting in decreasing order.

5) 16 pts

Consider an undirected graph G = (V, E) with distinct nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G. Now suppose each edge weight is increased by 1: the new weights are $w'_e = w_e + 1$. Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

Solution:

This question is similar to 6b question in your hw4. Monotonicity is preserved in (.+.) function as in (.)2

6) 16 pts
Mark all the correct statements and provide a brief explanation for each.

a. Consider these two statements about a connected undirected graph with V vertices and E edges:
I. $O(V) = O(E)$
II. $O(E) = O(V^2)$

(a) I and II are both false.
(b) Only I is true.
(c) Only II is true.
(d) I and II are both true.

(d) If there are V vertices, there can be at most ${}^{V}C_2$ edges in the graph. For the graph to be connected, there must be at least $V - 1$ edges.
$E \leq V(V\text{-}1)/2 = O(V^2)$.

Also, $V \leq 2(V - 1) \leq 2E$
$\Rightarrow V = O(E)$

b. Suppose the shortest path from node i to node j goes through node k and that the cost of the subpath from i to k is D_{ik}. Consider these two statements:
I. Every shortest path from i to j must go through k.
II. Every shortest path from i to k has cost D_{ik}.

(a) I and II are both true.
(b) Only I is true.
(c) Only II is true.
(d) I and II are both false.

(c) I. is false because there can be multiple shortest paths from i to j, not necessarily going through k. For example, there could be an edge between i and j that has the same cost as the path through k.

II. is true because if there were a path P' from i to k that had cost less than D_{ik}, then the path from i to j consisting of P' and the path from k to j would be shorter than the shortest i-j path, which is a contradiction.

c. Consider the execution times of two algorithms I and II:
I. $O(n \log n)$

II. $O(log(n^n))$

(a) Only I is polynomial.
(b) Only II is polynomial.
(c) I and II are both polynomial.
(d) Neither I nor II is polynomial.

(c) $\log (n^n) = n \log n \leq n^2$

d. Suppose $f(n) = 3n^3 + 2n^2 + n$. Consider these statements:
I. $f(n) = O(n^3)$
II. $f(n) = O(n^4)$

(a) Only I is true.
(b) Only II is true.
(c) I and II are both true.
(d) I and II are both false.

(c) For all $n \geq 1$,
$f(n) \leq 3n^3 + 2n^3 + n^3 = 6n^3 \leq 6n^4$

Additional Space