CS570

Analysis of Algorithms

Spring 2015

Exam I

Name: ______________________

Student ID: __________________

Email Address:________________

_____Check if DEN Student

	Maximum	Received
Problem 1	20	
Problem 2	20	
Problem 3	14	
Problem 4	13	
Problem 5	20	
Problem 6	13	
Total	100	

Instructions:
1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.

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

[[✓ TRUE]/FALSE]
For some graphs BFS and DFS trees can be the same.

[[✓ TRUE]/FALSE]
The number of cycles in a bipartite graph may be odd.

[[✓ TRUE]/FALSE]
Stable matching algorithm presented in class is based on the greedy technique.

[TRUE/[✓ FALSE]]
To delete the ith node in a binary min heap, you can exchange the last node with the ith node, then check the nodes below the ith node to see if the ith node should move down the heap to "re-heapify" it.

[TRUE/[✓ FALSE]]
Kruskal's algorithm can fail in the presence of negative cost edges.

[[✓ TRUE]/FALSE]
Consider an instance of the Stable Matching Problem in which there exists a man m and a woman w such that m is ranked first by w, and w is ranked first by m. Then (m, w) must appear in every stable matching.

[[✓ TRUE]/FALSE]
If a connected undirected graph G has the same weights for every edge, then a minimum spanning tree can be found in linear time.

[TRUE/[✓ FALSE]]
Given n numbers, one could construct a binary heap using the n numbers, then using the binary heap produce a sorted list of the numbers in O(n) time.

[[redacted]
In a [redacted] the insert operation has an amortized cost of $O(1)$ time, but the worst case cost is higher.

[TRUE/[✓ FALSE]]
Function $10n^{10}2^n + 3^n \log(n)$ is O($n^{10}2^n$).

2) 20 pts

A pharmacist has W pills and n empty bottles. Let $\{p_1, p_2, \ldots, p_n\}$ denote the number of pills that each bottle can hold.
Describe a greedy algorithm, which, given W and $\{p_1, p_2, \ldots, p_n\}$, determines the fewest number of bottles needed to store the pills. Prove that your algorithm is correct.

Solution:
Sort the n bottles in non-increasing order of capacity. Pick the 1st bottle, which has largest capacity, and fill it with pills. If there are still pills left, fill the next bottle with pills. Continue filling bottles until there are no more pills. We can sort the bottles in O(n log n) and it takes time O(n) to fill the bottles, so our greedy algorithm has running time O(n log n).

To show correctness, we want to show there is an optimal solution that includes the 1st greedy choice made by our algorithm. Let k be the fewest number of bottles needed to store the pills and let S be some optimal solution. Denote the 1st bottle chosen by our algorithm by p'. If S contains p', then we have shown our bottle is in some optimal solution. Otherwise, suppose p' is not contained in S. All bottles in S are smaller in capacity than p' (since p' is the largest bottle) and we can remove any of the bottles in S and empty its pills into p', creating a new set of bottles S' = S –{p}∪{p'}that also contains k bottles { the same number of bottles as in S'. Thus S' is an optimal solution that includes p' and we have shown there is always an optimal solution that includes the 1st greedy choice.
Because we have shown there always exists an optimal solution that contains p', the problem is reduced to finding an optimal solution to the subproblem of finding k-1 bottles in $\{p_1, p_2, \ldots, p_n\}$-{p'}to hold W-p' pills. The subproblem is of the same form as the original problem, so that by induction on k we can show that making the greedy choice at every step produces an optimal solution.

3) 14 pts
We are given a graph G = (V; E) with uniform cost edges and two vertices s and t within G. We are told that the length of the shortest path from s to t is more than n/2 (where n is the number of vertices within G). Give a linear time algorithm to find a vertex v (other than s and t) such that every path from s to t contains v. Justify your solution.

Solution:

Algorithm:

a) Based on the uniform edge cost assumption, we run BFS from s and find the shortest path from s to t.
b) Since the shortest distance from s to t is more than n/2, other than the s layer and t's belonged layer in the BFS tree, there must be at least one layer crossed by the shortest path that has only 1 vertex. Choose this single vertex that forms a layer by itself, and this vertex is a valid one we want.
c) Running BFS and searching takes linear time in total.

Proof (Justification):
The key part of the justification is to prove that, other than the s layer and t's belonged layer in the BFS tree, there must be a layer crossed by the shortest path that has a single vertex.

Suppose this is not true, i.e., other than the s layer and t's belonged layer, each layer of the BFS tree crossed by the shortest path has at least two vertices. As we know, the shortest path from s to t has distance more than n/2, i.e., the shortest distance is at least $\underline{n/2}$+1, where $\underline{n/2}$ means the largest integer smaller than or equal to n/2. Therefore, apart from the s layer and t's belonged layer, the number of layers in the BFS tree crossed by the shortest path from s to t should be at least $\underline{n/2}$, and the total number of vertices in these layers must be at least $2*\underline{n/2} \geq n-1$. Adding s and t, the total number of vertices in G turns out to be no less than n+1, which obviously contradicts the fact that n is the number of vertices in G.

Thus, other than the s layer and t's belonged layer, there must be a layer between s and t that only has a single vertex, denoted as v. Then every possible path from s to v must pass through node v.

4) 13 pts

Arrange the following functions in increasing order of asymptotic complexity. If f(n)=Θ(g(n)) then put f=g. Else, if f(n)=O(g(n)), put f < g.

$4n^2$, $\log^2 n$, $5n^2$, $\log^3 n$, n^n, 3n, $nlog(n)$, $2n$, 2n+1

Solution:

$\log^2 n < \log^3 n < 2n = 2n + 1 = 3n < nlog(n) < 4n^2 = 5n^2 < n^n$

5) 20 pts

Let $G = (V, E)$ be an (undirected) graph with costs $C_e \geq 0$ on the edges $e \in E$. Assume you are given a minimum-cost spanning tree T in G. Now assume that an edge connecting two nodes $v, w \in V$ with cost c is deleted from graph G and let G' be the new graph. Assume that the graph G' is connected.

a. If the removed edge doesn't appear in the MST T, will T still be the MST of G'? Please justify your answer.

Yes, If there is another MST T' which is better than T for the graph G' then it will also be better than T for the graph G, which is a contradiction that T is a MST for graph G.

b. If the removed edge appears in the MST T, give an [redacted] algorithm to find a MST for the graph G'.

After an edge is removed from the MST T, let A and B be the two connected components that are formed. From the graph G', find an edge e' that connects A and B and has minimum cost among all such edges. Join components A and B with the e' to give a new tree T', which will be MST of G'.

c. Prove that the output produced by your algorithm in part b is an MST

By applying cycle property, if a "better edge" is added to a tree, and the cycle has a more expensive edge than "better edge", then the tree is not MST; otherwise, "better edge" does not exist anywhere, tree is MST.

1. The two sub-trees after disconnection are both MST, using cycle property, no "better edge" within each sub-tree.
2. If a "better edge" is connecting two sub-trees, some edge in sub-tree is more expensive. This means the original tree before the disconnection is not a MST. We have contradiction, so the "better edge" is not connecting two sub-trees.

"better edge" does not exist anywhere, the tree from Problem 5(2) is a MST.

6) 13 pts

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. $T(n)=5T(n/2)+n^2 \log n$

Solution: From master theorem case 1, i.e. for the recurrence relation T(n) = aT($\frac{n}{b}$) + f(n), if f(n) = O(n^c), where c < $log_b a$ then T(n)=$\theta(n^{\log_b a})$

T(n) = $\Theta(n^{\log_2 5})$

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

From master theorem case 1, T(n) = $\theta(n^2)$

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

From master theorem case 3, i.e. if f(n) = $\Omega(n^c)$, where $c > log_a b$, and $af(\frac{n}{b}) \leq$ kf(n) for constant k < 1 then T(n) = $\theta(f(n))$

T(n) = $\theta(n^{\sqrt{6006}})$

Additional Space