Q1 14

1) 20 pts

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

[TRUE/FALSE]
Any depth-first search in a directed graph will produce the same number of tree edges in the DFS tree. True

[TRUE/FALSE]
For any connected graph with n vertices and m edges we can say that m+n = O(m). TRUE

[TRUE/FALSE]
In a binomial heap, for a given set of input values, the number of binomial trees in the heap will depend on the order of insertions False

[TRUE/FALSE]
The amortized cost of an operation can be as high as the worst-case cost of an operation in a data structure. True

[TRUE/FALSE]
A recurrence equation of the type $T(n) = a * T(n/b) + f(n)$ where $a * f(n/b) = 2 * f(n)$ cannot be solved using the Master Theorem. True

[TRUE/FALSE]
For a given problem instance, an algorithm with a $\Theta(n^2)$ worst case performance can run faster than an algorithm with a $\Theta(n \log n)$ worst case performance. True

[TRUE/FALSE]
The shortest path between two nodes in a weighted directed graph can be found using BFS if all edges except for those leaving the starting point have the same weight. True

[TRUE/FALSE]
Suppose that G is a directed, acyclic graph and has a topological ordering with v_1 the first node in the ordering and v_n the last node in the ordering. Then there is a path in G from v_1 to v_n. False

[TRUE/FALSE]
Consider an execution of Kruskal's algorithm on graph G with n nodes where all the first n-1 edges that we pick from the sorted list end up in the MST. Then we can conclude that G has no cycles. False

[TRUE/FALSE]
Kruskal's algorithm for minimum weight spanning trees is an example of a greedy algorithm. True

Q2 16

2) 16 pts

Consider a special case of the Minimum Spanning Tree problem where the graph G has edge costs of either 1, 3, or 5. We know that we can find an MST in this graph using Prim's algorithm in $\theta(m \log n)$ when a binary heap is used as the data structure to hold the nodes (in the set V-S). Given the fact that edge costs can either be 1, 3, or 5, create a special data structure (replacing the heap) that will improve this time complexity. In other words, the resulting algorithm should have a worst-case performance strictly better than $\theta(m \log n)$ on this graph.

The resulting data structure has the following properties.

~~4) Every node~~

1) It contains a bucket for each edge length of 1, 3, 5. Hence it contains 3 buckets (@ 1, 3, 5).

2) Each node goes into a particular bucket depending on its edge distance from S. ie a node with an edge distance of 'i' from S will go into bucket 'i'.

~~3) Using this we can extract~~

~~3) Since in each bucket,~~

3) We keep picking the ~~edge~~ node from bucket 1 first, then 3, then 5.

4) Hence every extract-min operation takes. O(1) instead of O(logn).

Hence the performance can be improved to O(m) from O(mlogn) by using the hash like structure

Q3 2

5) 16 pts

Suppose you are organizing a company party. The corporation has a hierarchical ranking structure; that is, the CEO is the root node of the hierarchy tree, and the CEO's immediate subordinates are the children of the root node, and so on in this fashion. To keep the party fun for all involved, you will not invite any employee whose immediate superior is invited. Every employee is equally valued, so we would like to maximize the total number of employees invited. Everyone in the corporation hierarchy (including the CEO) is considered an employee.

A) Provide a greedy algorithm to solve this problem. (8 pts)

1) Traverse the entire tree and create a list of employees with all subordinates.
2) Sort that list in decreasing order of number of subordinates
3) Iterate through the list and for each element, invite all the subordinates (if they are not already uninvited) and don't invite the superior.
4) Add this superior to the list of uninvited.

Note: All employees with no subordinates will be invited.

B) Prove that your algorithm produces an optimal solution. (8 pts)

In each step, we are selecting an employee with the max. no. of employees.

In the list created, assume there is an inversion i.e we select an employee with lesser no. of subordinates.

I

Q4 10

4) 10 pts
Arrange these functions under the O notation using only = or $\subset$ (strict subset of):
E.g. for the functions n, $n+1$, n^2, the answer should be $O(n+1) = O(n) \subset O(n^2)$. No justifications are required.

n^3-613, ln(n+4), sin(n^4), $\log_2(n^{88n})$, 9999, n^n, $n\log_3(n^3)$, $n^{1.1}$

$n^3 - 613 = O(n^3)$

$\ln(n+4) = \log_e(n+4) = O(\log_e n) = O(\log_{2.71} n)$

$\sin(n^4) = O(1)$

$\log_2(n^{88n}) = 88n \log_2 n = O(n \log_2 n)$

$9999 = O(1)$

$n^n = O(n^n)$

$n\log_3(n^3) = 3n \log_3 n = O(n \log_3 n)$

$n^{1.1} = O(n^{1.1})$

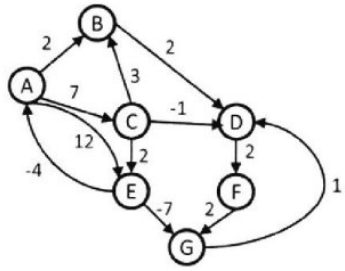
Answer

$O(\sin(n^4)) = O(9999) \subset O(\ln(n+4)) \subset O(n\log_3(n^3)) = O(\log_2(n^{88n})) \subset O(n^{1.1})$

$\subset O(n^3 - 613) \subset O(n^n)$

Q5 4

5) 18 pts



Consider the following directed, weighted graph:

(a) Even though the graph has negative weight edges, step through Dijkstra's algorithm to calculate the supposedly shortest paths from A to every other vertex. Show your steps in the table below. Start with initial distances in the top row of the table and cross out old values (distances) and write in new ones in the row below whenever distances change. (6 pts)

A	B	C	D	E	F	G
0	∞	∞	∞	∞	∞	∞
	2	7	4	9	6	8
			3			2

(b) List the vertices in the order which you marked them known. (3 pts)

(c) Dijkstra's algorithm found the wrong path to some of the vertices. For just the vertices where the wrong path was computed, indicate both the path that was computed and the correct path. (6 pts)

(d) What single edge could be removed from the graph such that Dijkstra's algorithm would happen to compute correct answers for all vertices in the remaining graph? (3 pts)

Q6 12

6) 12 pts
Use the Master's Theorem to solve the following recurrence equations (if it applies). Indicate if the Master's Theorem does not apply and give a reason why.

a) $T(n) = 3T(n/4) + O(n)$ (3 pts)

$f(n)$ is $O(n)$

Case 3 $n^{\log_b a} = n^{\log_4 3}$ Since $\log_4 3$ is less than 1,

$f(n)$ is $\Omega(n^{\log_4 3})$ and the two functions differ by a polynomial

$3f(n/4) \le cf(n) \Rightarrow 3O(n/4) \le cO(n) \Rightarrow c$ exists ($\frac{3n}{4} \le cn \Rightarrow c \ge \frac{3}{4}$ and $c < 1$)

$\therefore T(n)$ is $\Theta(n)$

b) $T(n) = 8T(n/4) + O(n^2)$ (3 pts)

$n^{\log_4 8} = n^{1.5} = O(n^{1.5})$

Case 3 $\therefore f(n)$ is $\Omega(n^{1.5})$ and they differ by a polynomial

$8\left(\frac{n^2}{16}\right) \le cn^2 \Rightarrow c \ge \frac{1}{2}$ and $c < 1 \Rightarrow c$ exists

$\therefore T(n)$ is $\Theta(n^2)$

c) $T(n) = 2^nT(n/2) + O(n)$ (3 pts)

Master's theorem does not apply because the number of subproblems (a) cannot be dependent on n. 'a' needs to be a constant value > 1.

d) $T(n) = 9T(n/3) + O(n^2)$ (3 pts)

$n^{\log_b a} = n^{\log_3 9} = n^2$.

Case 2. $\therefore f(n)$ is $\Theta(n^{\log_b a})$

$\therefore T(n)$ is $\Theta(n^2 \log n)$.

EDB43222-692F-49C1-9303-781C871B3E40

csci570-20191-midterm1

#62 8 of 10

Q7 8

7) 8 pts

Assume you are creating an array data structure that has a fixed size of n. You want to backup this array after every so many insertion operations. Unfortunately, the backup operation is quite expensive, it takes exactly n operations to do the backup regardless of how many elements are currently in the data structure (e.g. after 5 insertions, there are 5 elements in the data structure, but the backup still requires exactly n operations). Insertions without a backup only take one operation to perform.

a) What is the amortized cost of the insertion operation (after n insertions) if you perform a backup after every $n/5$ insertions? (4 pts)

If we charge insert 6 and backup 0, with actual costs 1 and n respectively,
After $n/5$ iterations, we have $5 \times \frac{n}{5} = n$ credits in the bank, which we can use to pay for the expensive backup operation.

∴ Amortized cost of insertion is $O(1)$

b) What is the amortized cost of the insertion operation if you perform a backup after every 5 insertions? (4 pts)

If we charge insert $\left(\frac{n}{5}+1\right)$ and backup 0, with actual costs of 1 and n respectively,
After every 5 iterations, we have $\left(\frac{n}{5}+1\right) \times 5 = n+5$ credits in the bank. which we can use to pay for the expensive backup operation and be left with 5 credits

∴ Amortized cost of insertion is $O(n)$.

Additional Space

98E20B69-875D-4154-95F5-BDECA3033B7F

csci570-20191-midterm1

#62 10 of 10

Additional Space