CS570

Analysis of Algorithms

Summer 2017

Exam I

Name: Kavish Jadwani

Student ID: 6917934471

Email Address: jadwani@usc.edu.

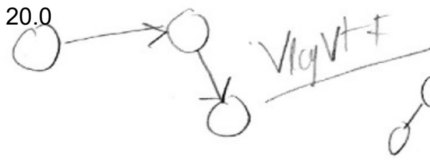
_____**Check if DEN Student**

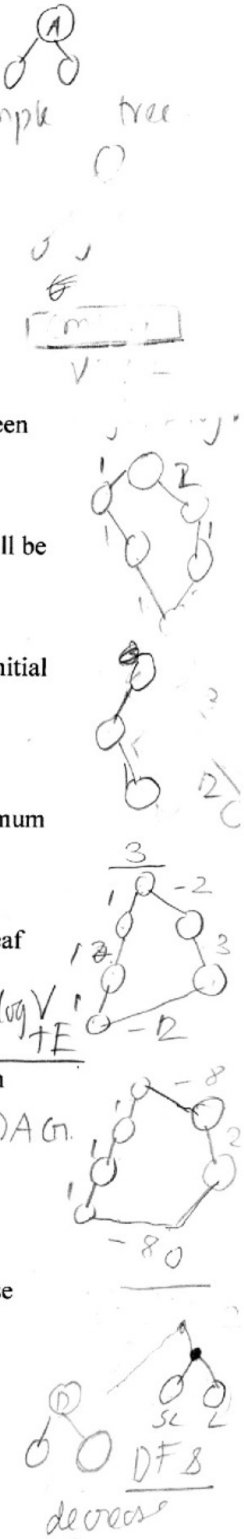
	Maximum	Received
Problem 1	20	
Problem 2	10	
Problem 3	12	
Problem 4	12	
Problem 5	13	
Problem 6	13	
Problem 7	20	
Total	100	

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.

Q1 Score: **12.0** / 20.0





1) 20 pts

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

[TRUE/FALSE] False
The depths of any two leaves in a binomial heap differ by at most 1.

[TRUE/FALSE]
Given that all edges in a graph G are of equal weight, then the shortest path between any two vertices in G can be computed in linear time. False.

[TRUE/FALSE]
Given a connected graph G with at least two edges having the same cost, there will be at least two distinct minimum spanning trees in G. False

[TRUE/FALSE]
Provided the BFS and DFS trees of a connected undirected graph with the same initial vertex, the distance between two vertices in the DFS tree cannot be smaller than distance in the BFS tree. True.

[TRUE/FALSE].
In a graph, if one raises the lengths of all edges to the power 3, the minimum spanning tree will stay the same. True

[TRUE/FALSE]
In Huffman coding, the item with the second-lowest probability is always at the leaf that is furthest from the root. True

[TRUE/FALSE]
The shortest path between two nodes in a graph could change if the weight of each edge is increased by an identical positive number. True

[TRUE/FALSE]
The shortest path in a weighted DAG can be found in linear time. False

[TRUE/FALSE]
If an operation takes O(1) amortized time, then that operation takes O(1) worst case time. False

[TRUE/FALSE]
In Fibonacci heaps, the decreaseKey operation takes O(1) worst case time.
True

Q2 Score: **10.0** / 10.0

$f(n) = O g(n)$
$f(n) = \theta g(n)$
$f(n) = \Omega g(n))$

n^2 Ω. n θ
n^2 Ω.
$n^{1.5}$

2) 10 pts
In the following table, for each pair of expressions *(f(n), g(n))*, tell whether *f(n)* is $O(g(n)), \Theta(g(n)), \Omega(g(n))$ of these expressions. Put symbol O, Θ, Ω in the empty cells in the table. The first row, $f(n) = n^2$ is filled to help you understand this problem.

$n\sqrt{n}$ $\log n$
$1.5 \log n$ $\log n$

		g(n)			
		n	n^2	$n\log(n)$	2^n
f(n)	n^2	Ω	Θ	Ω	O
	$n^{1.5}$	Ω	O	Ω	O
	$n\log(n)$	Ω	O	Θ	O
	$\log^{10}(n)$	O	O	O	O
	$\sqrt{2^n}$	Ω	Ω	Ω	O
	$n^{\log n}$	Θ	O	O	O

$(\log n)^{10}$

2
$= 1$
$n/2$
$2 \log n$
$\log_2$

4 n^2

n

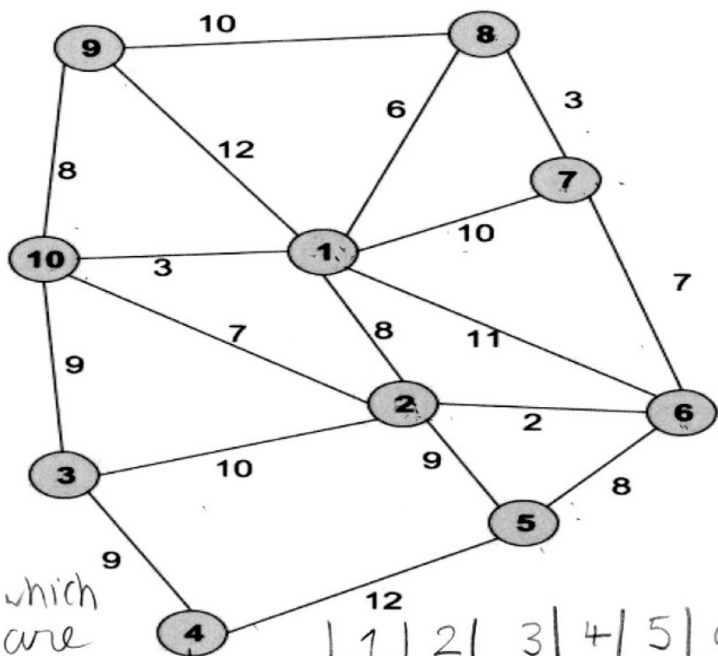
$2 \log n$

2 $1 =$
4 2

Q3 Score: **12.0** / 12.0

3) 12 pts

Run Dijkstra's algorithm on the following directed graph, to compute distances from node 1 to all the other nodes. List the vertices in the order they are added to the shortest path tree. You do not need to demonstrate all details of the algorithm execution.



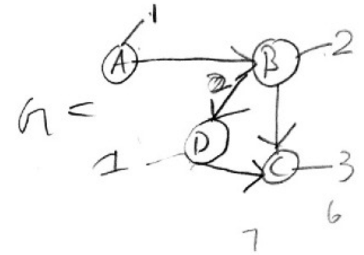
The order in which the vertices are added are :-
1, 10, 8, 2, 7, 6, 3, 9, 5, 4

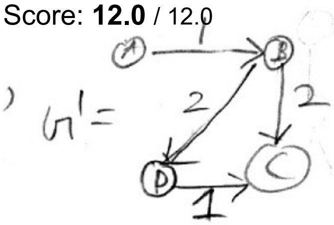
Shortest distances are as follows from node 1

Node	Distance
2	8
3	12
4	21
5	17
6	10
7	9
8	6
9	11
10	3

	1	2	3	4	5	6	7	8	9	10
	0	∞	∞	∞	∞	∞	∞	∞	∞	∞.
1	–	8				11	10.	6	12	3
10	–	8.	12			11	10	6	11	–
8		8	12			11	9	–	11	–
2	–	–	12		17	10	9.	–	11	–
7	–	–	12		17	10	–	–	11	–
6			12		17	10	–	–	11	
3	–	–	–	21	17	–	–	–	11.	–
9	–	–	–	21	17	–	–	–	–	–
5	–	–	–	21	–	–	–	–	–	–
4	–	–	–	–	–	–	–		–	

Q4 Score: **12.0** / 12.0





← Illustration

4) 12 pts
Suppose we define a new kind of directed graph in which positive weights are assigned to the vertices but not to the edges. If the length of a path is defined by the total weight of all nodes on the path, describe an algorithm that finds the shortest path between two given points A and B within this graph. Hint: modify a given graph.

Consider a graph G, with each node having positive weights. In this graph, we need to find the shortest distance between two point A & B.
The length of path is the sum of the vertice weights in the graph. Thus our minimum path will be at least the sum of weights of A & B.
Now, let's construct another graph G' such that the edge (u,v) carries weight of vertice u and edge (v,w) carries weight of vertice v where (u,v) is edge from u to v and (v,w) is edge from v to w
Thus, now G' is a weighted directed graph and we can run Dijkstra's algorithm on G' to find shortest distance between A & B within this graph

Q5 Score: **4.0** / 13.0

5) 13 pts
Consider a singly linked list data structure that stores a sequence of items in any order. To access the item in the i-th position requires time i. Also, any two contiguous items can be swapped in constant time. The goal is to allow access to a sequence of n items in a minimal amount of time. The TRANSPOSE is a heuristic that improves accessing time if the same is accessed again. TRANSPOSE works in the following way: after accessing x, if x is not at the front of the list, swap it with its next neighbor toward the front. Prove that the amortized cost of a single access is at least linear, i.e. $\Omega(n)$.

item at i, time $= i$

swapping of contiguous $=$ constant time

We need to prove that amortized cost of single access is $\Omega($

Position	Access Time
1	1
2	2
3	3

→ Consider x which is at position i. and consider that in best case, x is accessed again and again, then the time taken by the first access of x is i and for n recursive access to x, the time taken will be constant. Thus, we can conclude that the amortized cost of a single access is at least linear ie $\Omega(n)$ when $i = 1$ ie the best case for accessing n items is at least linear.

Q6 Score: **13.0** / 13.0

Start Here → Here, we assume that each item in same cluster has constand distance between them and each cluster has a constant distance between each other given by some abstract representation.

6) 13 pts

Suppose you have *n* objects with defined distances *d(u,v)* between each pair of them. *d(u, v)* may be an actual distance, or some abstract representation of how dissimilar two objects are. Your task is to group these objects in such a way that objects within a single group have small distances between them, and objects across groups have larger distances between them. This problem is called *clustering*. Propose an algorithm to separate given *n* objects into $k \leq n$ clusters (groups), so that the minimum distance between items in different groups is maximized.

We have distances d(u,v) between each pair of items. For a graph G with all items connected to each other. Now, find MST for this graph.

it is not a complete graph

The MST will connect the items with smaller distance between them and thus group them all together forming a cluster. And, thus there will be $K < n$ clusters. These clusters will be connected to each other in such a way that the clusters of items are least different from each other.

Our goal is to maximize distance between each cluster.

P.T.O

Please see back side of this page for explanation

Q7 Score: 20.0 / 20.0

7) 20 pts

Suppose there are n children at the birthday party, and the birthday cake is sliced into n slices with sizes $s_1, s_2, \ldots, s_n$ (where s_k is the weight of slice k) The cake was not perfectly sliced, so no slice has the same weight as others. Given the appetite of each child $a_1, a_2, \ldots, a_n$ (where a_k is a measure of child k's appetite in terms of cake weight), we want to distribute the slices (one slice per child) to fulfill every child's appetite. We say, a child's appetite is fulfilled if his/her slice weighs greater than or equal to his/her appetite.

a) Describe an algorithm to determine whether it is possible for such a distribution of the slices. (8 pts)

→ We maintain two min-heaps.
A → One will have the weights of the cake slices
B → The other will have the appetite of the children
Let the first min heap be A, and second min heap be B
We take the minimum of both heaps and compare
if min(B) ≤ min(A), we form a pair and remove the min element from A & B both.
If min(B) > min(A), we remove the min(A) element and repeat the above steps.
The total number of pairs will give the number of children whose appetite is satisfied. The remaining children will not have their apetite satisfied

b) Determine the runtime complexity of your algorithm. (4 pts)

To make heaps, it takes O(n) time.
To find and delete min elements, it takes O(log n) and we do this for n times in worst case
∴ total = O(n + n log n)

1 @ 2 3 5

1 9 11 12.

child with, a1

gel 81

wl 81 < a

1	2	3	5
12	11	9	1

c) Prove the correctness of your algorithm.(8 pts)

In our algorithm, we choose a child with minimum appetite and try to mate it with the cake of minimum weight.

If the cake slice of minimum weight cannot satisfy the appetite of child having lowest appetite, then that slice of cake cannot satisfy the appetite of any other child ~~if~~ because the appetites are in min heap and the next children will have appe greater than the previous one.

Hence our algorithm is correct.

space Not graded

Additional Space

Additional Space