COL106: Data Structures. I semester, 2018-19. Major Exam.
17 November 2018. Maximum Marks: 30 marks

Name Mohammad Atif	I	Ent. No. 2017EE10462

Problem 1 (4 marks) You are given n numbers stored in an input queue (you can look at the front of the queue and deque, but you cannot enqueue anything into it). You are also given two stacks and a *constant* amount of extra storage apart from that. Using these stacks and the constant storage sort the numbers in ascending order. At the end of the algorithm the numbers should be available in one of the stacks with the smallest at the top of the stack. What is the worst case running time of your algorithm?

Exam Instructions. Because exam papers will be scanned you need to carefully follow these instructions.
1. Answer clearly *within* the space provided. **No extra sheet will be given or collected.**
2. Workout your answers in rough and then transfer carefully to the printed sheet.
3. Rough sheets will **not** be collected.
4. If you need a new printed sheet, you can get one but you will have to transfer your entire exam onto the new sheet. We will collect only two sheets of paper from you.

Queue → Q
Stacks → S_1, S_2

i) Deque the queue and push the element coming out into stack S_1.

ii) Look at the top of que and compare it with the element on the top of the stack (S_1) number at top of stack / element in front of que
Deque(Q) and push the element coming out into the stack S_1.

else

Pop(S_1) until it is empty or front of queue becomes $\geq$ element of top of stack S_1.
Then deque(Q) and push element into S_1

iii) Pop(S_2) and keep pushing elements into S_1 until S_2 becomes empty.

iv) Keep Repeating from step (ii) until Queue becomes empty.

In worst case at every step of insertion we will need to pop(S_1)

Worst case time $= 1 + (2(1)+1) + (2(2)+1) + \cdots + (2(n-1)+1)$

$= n + 2\sum_{i=1}^{n} i$

$= n + 2\left(\frac{n(n+1)}{2}\right)$

$= n^2$

$\approx \Theta(n^2)$

At any step we need to remove all elements of S_1 then deque and add and then again push all elements of S_2 into S_1 i.e. $2(i)+1$, i = no of elements in S_1

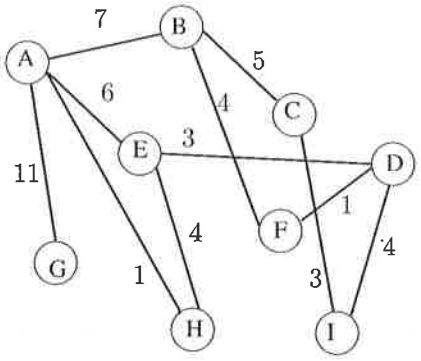
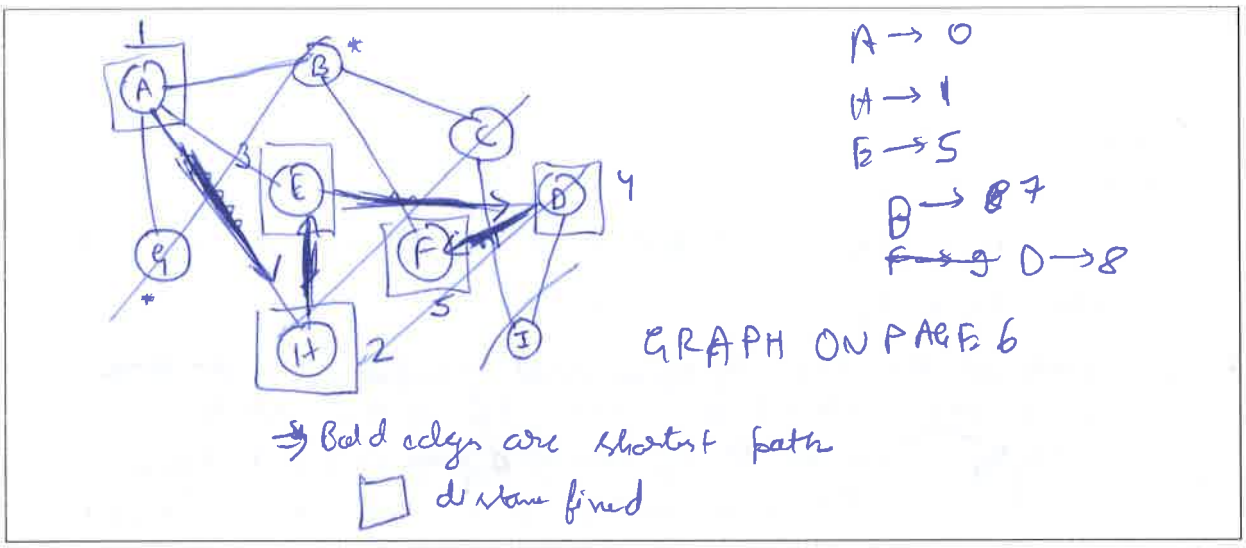
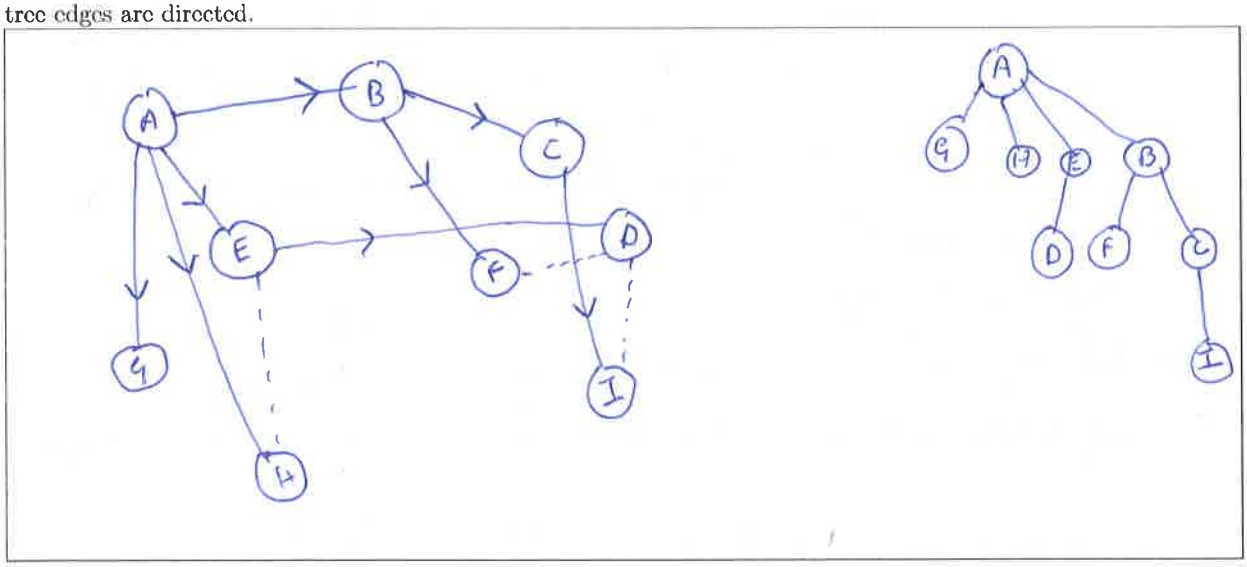


Figure 1: Graph with edge weights

Problem 2 (3 marks) Consider the edge-weighted graph in Figure 1. Suppose we start Dijkstra's algorithm from vertex A. If we consider A to be the 1st vertex, show the state of the graph after the distance label of the 5th vertex has been fixed. To be precise: I want you to redraw the graph in the space below, give each vertex a distance label, put a box around the distance labels that are guaranteed not to change (i.e. have been fixes), and indicate which edges of the shortest path tree have been identified up to this point.

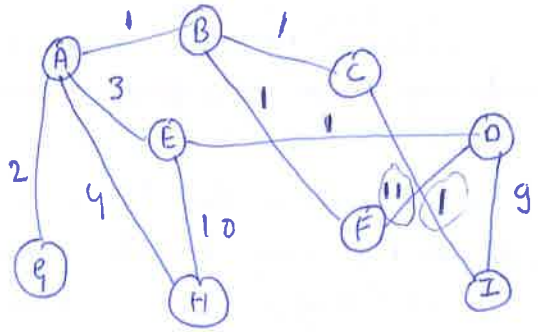


Problem 3 (3 marks) Ignoring the edge weights perform BFS starting from A on the graph in Figure 1. Redraw the graph indicating the tree edges with solid lines and the cross edges with dotted lines. Make sure that your tree edges are directed.



Name Muhammad Atif	II	Ent. No. 2017EE10462

Problem 4 (3 marks) Change the weights of the edges of the graph in Figure 1 so that the shortest path tree computed by Dijkstra's algorithm is exactly the same as the BFS tree you have drawn for Problem 3. To make it easier for us to check please redraw the graph and unambiguously label the edges with the new weights. Note that if you haven't attempted Problem 3 then you cannot get any marks for this problem.



Problem 5 (3 marks) Prove clearly and completely that $n^{1-\delta}$ is $O\left(\frac{n}{\log n}\right)$ for any constant $\delta > 0$.

$$n^{1-\delta} = \frac{n}{n^{\delta}}$$

We know for $n \geq 1$

$$n > \log n > 0$$

$$\Rightarrow n^{\delta} > \log(n^{\delta})$$

$$\Rightarrow \frac{1}{\log n^{\delta}} > \frac{1}{n^{\delta}}, \quad n \geq 1$$

$$\frac{1}{\delta \log n} \geq \frac{1}{n^{\delta}}, \quad n \geq 1$$

$$\frac{n}{n^{\delta}} \leq \frac{1}{\delta}\left(\frac{n}{\log n}\right), \quad n \gg 1$$

$$\Rightarrow \frac{n}{n^{\delta}} \in O\left(\frac{n}{\log n}\right)$$

$$n^{1-\delta} \in O\left(\frac{n}{\log n}\right)$$

Problem 6 (4 marks) Explain how to delete *any* key from a heap with n keys in it, not just the minimum key. *No pseudocode* allowed for this problem. You must explain the steps required in words. You may assume all keys are distinct and that you are given a pointer to the node containing the key to be deleted. The time complexity of your method must be $\theta(\log n)$. Assuming it's a "heap structure"

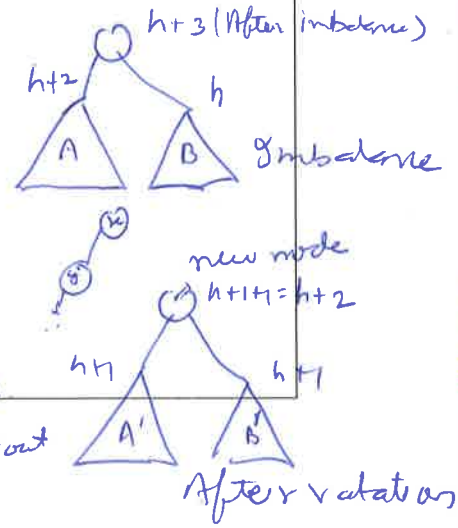
1. If the node to which pointer is given is a leaf then simply remove it.
2. If it is an internal leaf then exchange the key with that in last node of the lowest level of the heap. and then delete the last node of the lowest level
3. Bubble down from the node to which pointer was given. (Note: The node to which this pointer points now has a different key as we exchanged it in step 2.)

Bubble down algorithm:
① Starting from level i of node, if key in the node > min of keys in children, then swap.
② Continue step down to until you reach an internal node or key at node < min. of key at children

Problem 7 (3 marks) Suppose we start with an AVL tree T. We do an insertion of a new key in the tree just using the binary search tree algorithm without yet doing any rotations. The result is a tree T'. It turns out that the deepest node in T' which is not balanced is the root, so we do an appropriate rotation at the root to produce the final AVL tree T''. If the height of T'' is 10, what were the heights of T and T' respectively ? Give an argument for your answer.

The T'' has same height as T. because rotation at node doesn't change height of the node where imbalance occured. Height of T'' = Height of T = 10.

Explanation: During rotation one of the nodes of A goes to the left of B. Height A dec. and becomes h+2-1 = h+1, and height of B increases to h+1. (Clearly height of new root = h+2 (same as before) ← because then A would have been of height h+1. Adding one more in height by 1 led to imbalance



In the imbalance tree we can see height of root will be h+2+1 = h+3

Height of T'' = h+2 = 10
h = 8
Height of T' = h+3 = 11

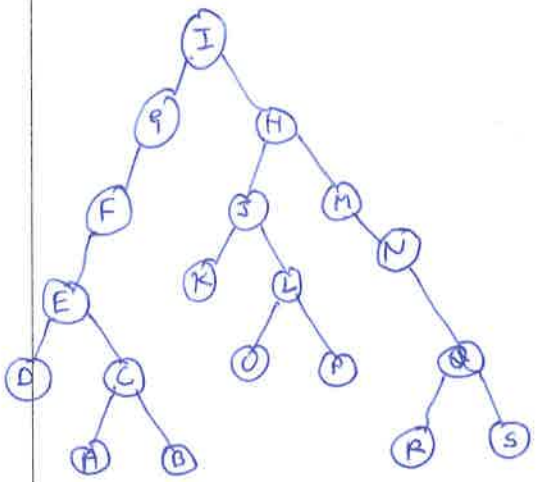
Name	Mohammad Atif	III	Ent. No.	2017EE10462

Problem 8 (3 marks) Given n numbers that all lie between 0 and n^4 explain how to sort them in worst case $\theta(n)$ time assuming that all mathematical operations take $\theta(1)$ time.

We will use radix sort ~~or~~ with small modification.
We have numbers from range 0 to n^4. ~~of we change~~
(1) We change the base of numbers from base 10 to base n.
Now all given numbers $\leq n^4$. Maximum No. of digits in n^4 in base $n = \lfloor \log_n(n^4) \rfloor + 1$
$= \lfloor 4 \rfloor + 1 = 5$
This gives ~~both dig~~ that maximum number of digits = 5
(2) We will use radix sort. Radix sort takes $O(d(n+N))$ time where N is range of input, d is number of digits, n is numbers of inputs. Range is now 0 to $n-1$ ~~as we have changed base~~ as we have changed base to n. Sorting time = $O(5(n+n-1))$
~~Running time = $O(5(n+n-1)) = O(10n-5) = O(n)$~~ $O(n)$ $= O(n)$
(3) All mathematical operations take $O(1)$. So Since max. no. of digits in base n, numbers is 5. So for conversion of one number we require $O(5)$ time times and for n numbers $O(5n) = O(n)$ time

CONTINUED on page 6

Problem 9 (4 marks) The preorder and postorder traversals of a tree are IGFEDCABHJKLOPMNQRS and DABCEFGKOPLJRSQNMHI respectively. Draw the tree and give its inorder traversal.



DEACBFGIKJOLPHMNQRQS

↑ inorder traversal

Use this space to continue or rewrite any solution that you may have messed up in the allotted space. Make sure you clearly mention the problem number here and indicate clearly that *the solution is continued on page 6* in the allotted space.

(8) From arguments (2) and (3) total running time = $O(n)$

Radix sort algorithm.

i) First make a bucket of given numbers. Add min. no. of zeros in front to make equal ~~digits in each~~ characters in each number

ii) ~~Sort by using~~ Perform bucket sort as ~~on~~ least significant digits at keys. Then, sort using the second last significant digits (stable sort) and so on.

(2)