M. Hamza Shakeel

18L-2191

Data-Structures

Section-A

Question no 1:

We are given two functions. One function is called recursively in another function.

Time complexity for recursive function:

- int j=0 = O(1)
- j<N-1 = O(N)
- j++ = 2(N) * As 2 is constant so it does not affect. outer loop ends
- largest=j => O(N) =>(number of j)
- int i=j => O(N)
- i<N => N(N+1)

- i++ 2(N)(N+1).
- if() ⇒ if condition has time complexity = O N(N+1).

⇒ Function.

* Find top players.

Sort(playerlist, N) ⇒ it will give the order of N^2.

- int i ⇒ O(1)
- i < 3 ⇒ 3
- ++i ⇒ 6.

Now summing the total time complexity.

$= O(N^2) + O(1) + 3 + 6$

$= O(N^2)$

So, it is a order of N^2.

Question: 1

b.

The better solution to this problem can be done by using the sorting algorithm, that is heapsort.

The pseudo code to the heapsort is stated as follows.

⇒ Heapify (an Array, int x, int y)

```
{
    int p;
    int l, r;
    // we are using two variables
       for taking left and right child
    of that array.
    // As we formula for left child is
       2i. and right child is 2i+1.
       // l = 2i
          r = 2i+1.
```

// Now compare left with by and the value at left

~~if((*left<=y && Array(l~~

if (l<= y && Array[l] > A [x])

{ then

p = left // it is algo for heapsort on left side.

else

p = x.

}

Similarly we check for right side

if (r<=y && A[r] > A[p]).

~~ma~~ p = r

~~So~~

if (p != x)

swap both max and i index.

again call Heapify(Array, p).

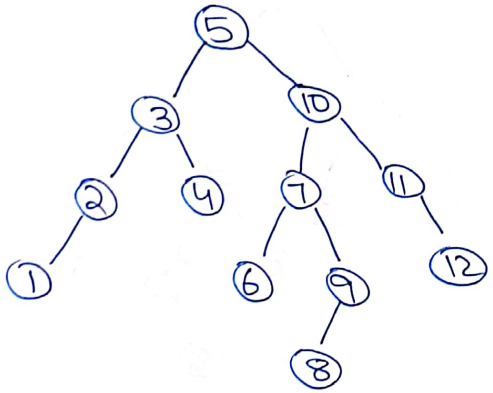
now we build heap.

Store elements.

=> Complexity of this pseudocode is O(nlogn)

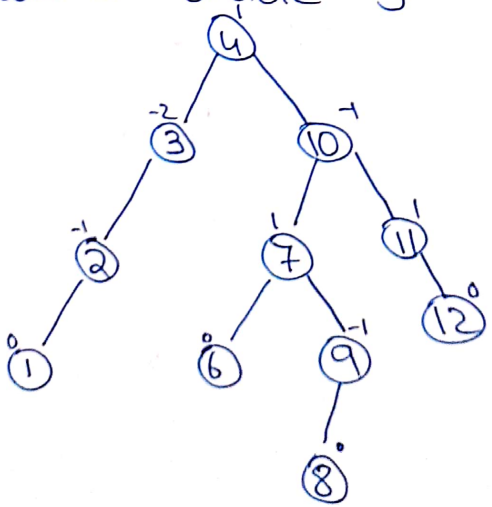
Question no 7:

Given AVL Tree

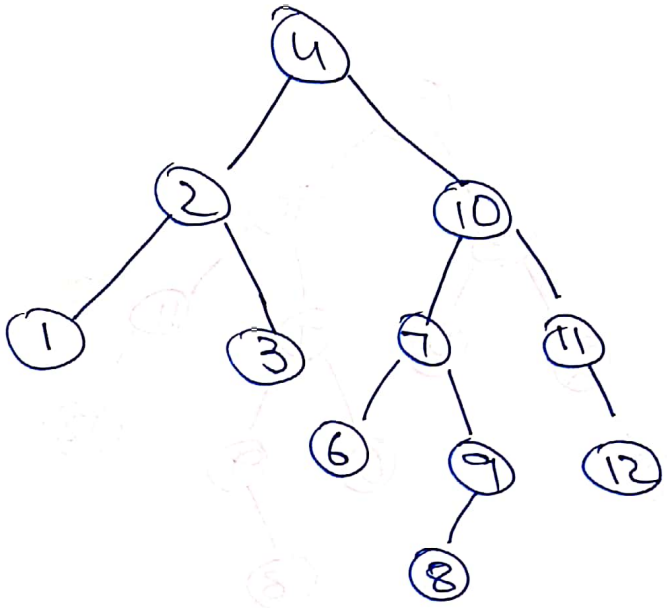


a: Delete 5 from the given AVL tree.

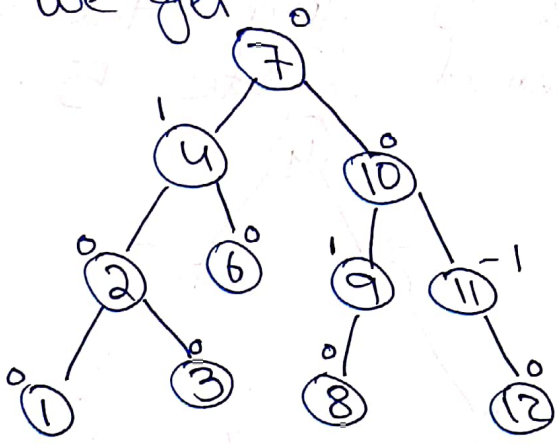
Replace 5 from the 4 of left child and delete 5.



b Now at node with value 3, is not balanced, so applying rotations.



Now applying R-L rotation on the given graph on the root => 4. and we get



So it is a balanced AVL tree after deletion of 5.

M. Hamza Shakeel
18L-2191
Data-Structures-A

Q 3:

Given data:

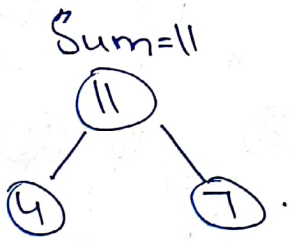
a = 15, h = 4, m = 8, t = 7, Space = 12

=> arranging in ascending.

4	7	8	12	15
h	t	m	Sp	a.

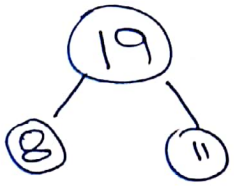
taking two smallest node.

4 + 7 = Sum = 11
h t



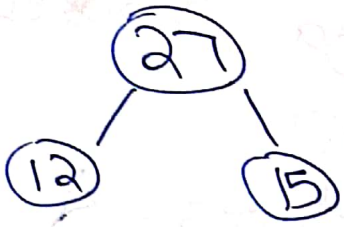
=> Now again inserting array:

8 11 12 15



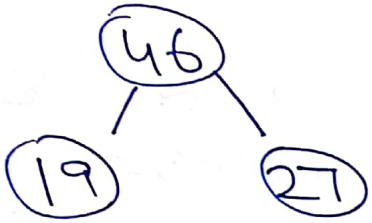
$\Rightarrow$ 12 15 19.

12+15 = 27.

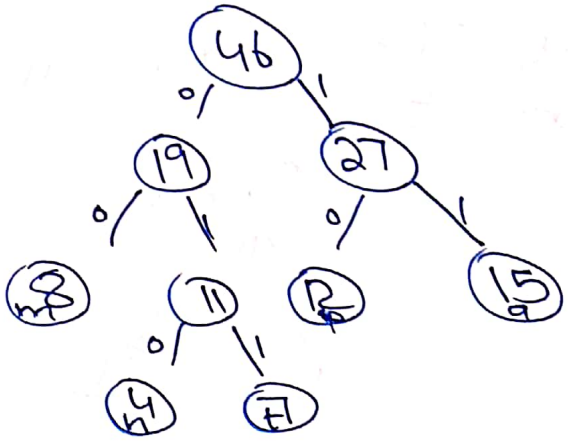


$\Rightarrow$ 19, 27.

19+27=46.



Over all tree



a = 11

Space = 10

h = 010

t = 011

m = 00.

Now by decoding this tree, we do not get the same ans of three given

Ans: all ans are incorrect.

Q # 5:

a. CPU capacity: 2.70 GHZ.

Memory = 6.5/7.9 (GB)

Cache:

L_1 : 128KB

L_2 : 512KB

L_3 4.0MB.

RAM: 8GB.

b

input Size	Reccursion Sol			Iterative Sol		
	in-order	Preorder	Post	inorder	Preorder	Postorder
10	3800	4100	4200	195990	277600	416000
50	5400	4800	5200	997500	648810	997100
500	4500	5100	4700	35967	59240	103669

c Efficiency and time complexity of Recursive solution is O(N)

Efficiency and time complexity of iterative solution is O(N)

d

The recursive solution of these traversals grows slower than the iterative solution. The difference of computational time of all traversals between both is mainly because of increase of size in interative solution. Also the space complexity of iterative solution is more.

Q 6:

a:

Worst Case scenario of insert function is O(n).

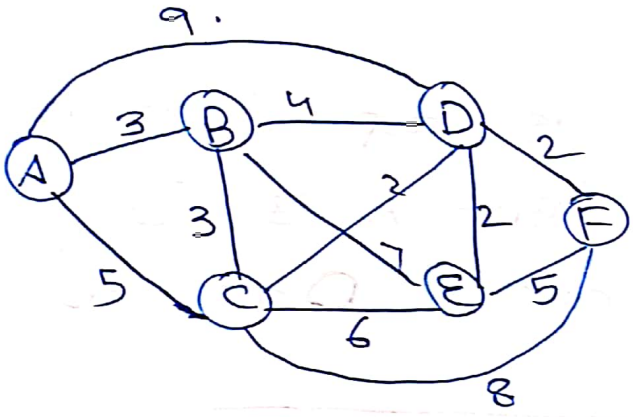
Average case is O(1).

b: Worst Case complexity for search is O(n)

Avg case is O(1).

Question no 4:

a.



As, we start from a and assuming it as a source vertex.

So, distance of a is 0 and assuming the distance of other vertices from a as infinity.

⇒ a has two child b, c.

A	

Now A is popped and b, c are insert b, c in queue.

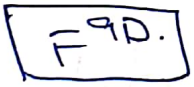
3A b	c 5A

=> After pop of B it's childs are inserted.

=> C is popped as lowest/shortest path from A is 5. Child of C are added.

=> Now D pops.

=> Now E pops.



Now for shortest distance from A

A = 0

B = A → B = 3

C = A → C = 5

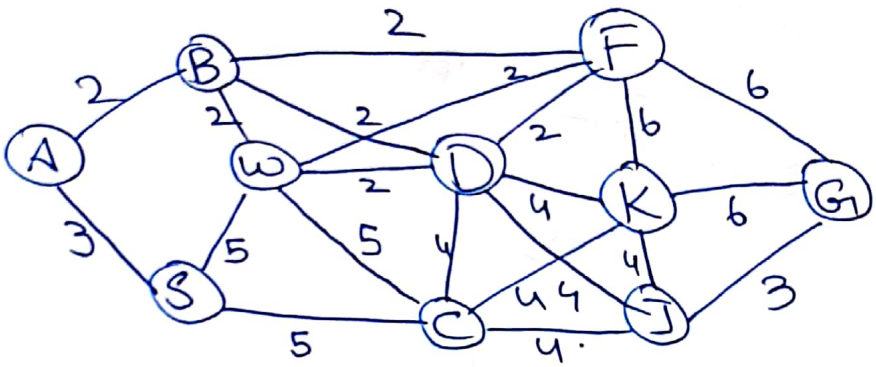
D = A → B → D = 7

E = A → B → D → E = 9

F = A → B → D → F = 9

So they are the shortest paths from A.

C



By the given view of ground floor I make this graph.

Now for finding spanning tree. we arrange all costs of the given tree in ascending order.

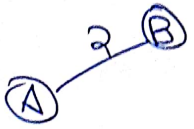
⇒

AB	BF	BD	BW	WD	WF	DF	AS	JG	CD	DK	CK	JK	JC
2	2	2	2	2	2	2	3	3	4	4	4	4	4

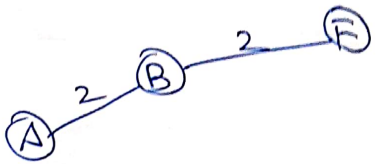
JD	SW	SC	WC	KG	FG	KF
4	5	5	5	6	6	6

⇒ Now Starting Kruskal.

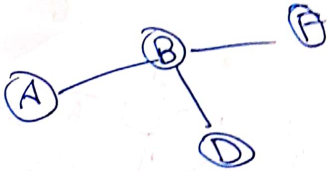
①



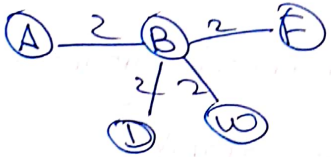
②



③

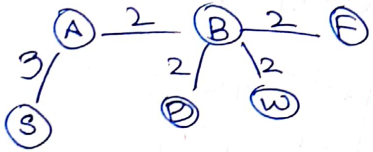


④

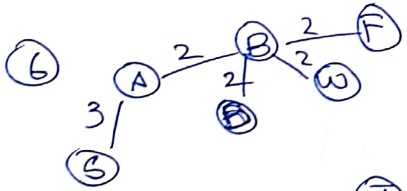


we dont include WD, WF bcz they make cycle.
Also missing DF.

⑤

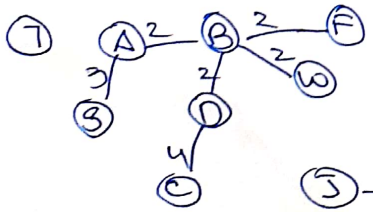


⑥



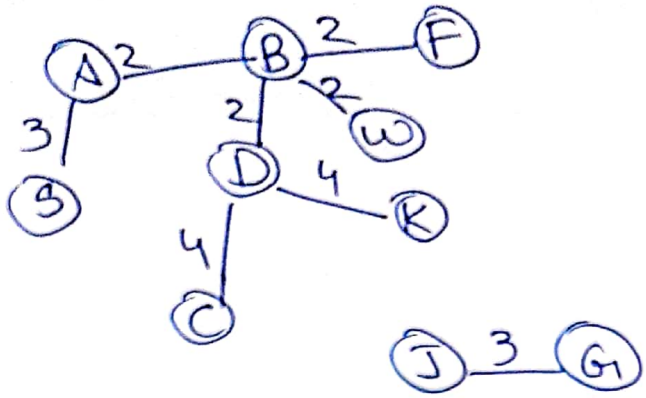
J 3 G

⑦



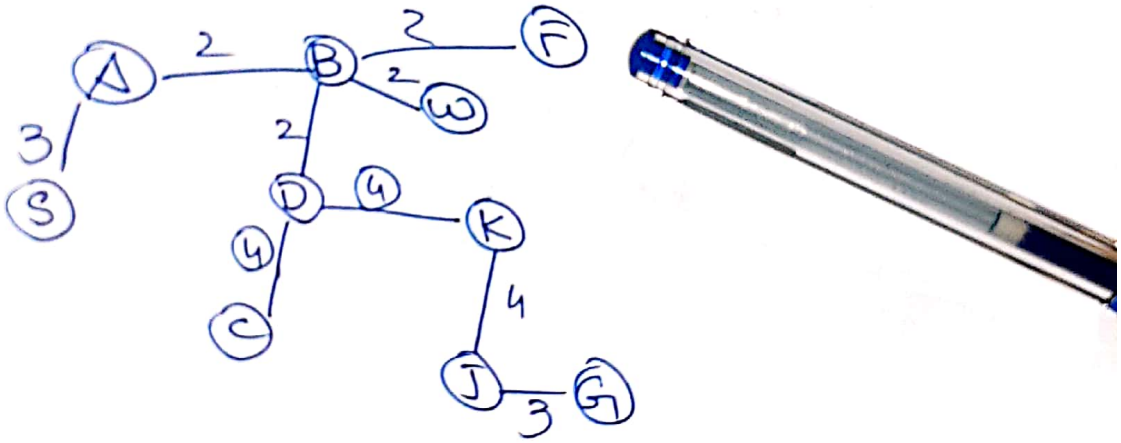
J 3 G

(8)



Now missing CK, beg it makes loop.

(9)



it is the minimum spanning tree for the given graph.

Cost = 26

⇒ it is not possible to walk through every doorway exactly once.

b.

it is not possible to add an edge in MST (max) because it will nullify the main concept of Spanning tree. As, we know, the spanning tree has min num of edges.

As, after adding one edge, the MST will become equal to vertices and now cycle will be formed.