18L-1051
Rida Fatima

QNo3

h	t	m	space	a
4	7	8	12	15

7+4 = 11

11+8 = 19

m:8 | ht: 11 (0: h:4, 1: t:7) | space: 12 | a:15

12+15 = 27

Space: 12 a:15 | mht: 19 (0: m:8, 1: ht: 11 (0: h:4, 1: t:7))

mht: 19 (0: m:8, 1: ht: 11 (0: h:4, 1: t:7)) | space a: 27 (0: space: 12, 1: a:15)

mht space a: 46 (0: mht: 19 (0: m:8, 1: ht: 11 (0: h:4, 1: t:7)), 1: space a: 27 (0: Space: 12, 1: a:15))

QN i) All codes are wrong

CP Because code for

a is 11

Rf

Co

m is 00

t is 011

N

space is 10

(b)

Inpu

i

s

QNO1

a) Time complexity analysis

```
for( int j=0 ; j< N-1 ; j++)  ---- (n-1)
{  largest = j ;
   for ( int i= j+1 ; i<N ; i++)
   { if ( playsList[i] -> getScore() > playList[largest] -> getScore() )
       largest = i ;
   }
   swap( playersList[j], playersList [largest]) ;
```

(c)

(d) According to above code the time complexity is $O(n^2)$ because.

To find maximum element from array of n element here n-1 comparisons are to be performed Then, after putting max element to its proper position, size of unsorted array reduces to n-1 and then n-2 comparisons are required.

18L-1051

(2)

So,

(a) $(n-1) + (n-2) + \dots + 1 = \frac{n*(n-1)}{2}$ and have

n swappings

So, $O(n^2)$

(b) **Merge sort** will be a better solution because it's time complexity in $\theta(n \log n)$

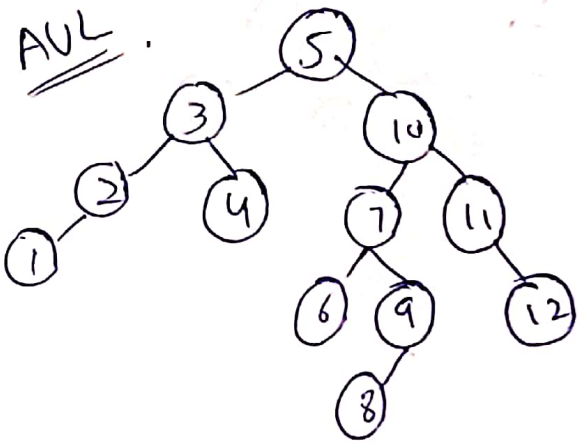
(b)
```
MergeSort (playerList, left, right)
{ if (r > l)
    // we find middle point
       middle m = (left + right)/2

    // call sort
       call Mergesort (arr, l, m)
       call Mergesort (arr, m+1, r)
       call Mergesort (arr, l, m, r)
}
```

(c)

QNo 7

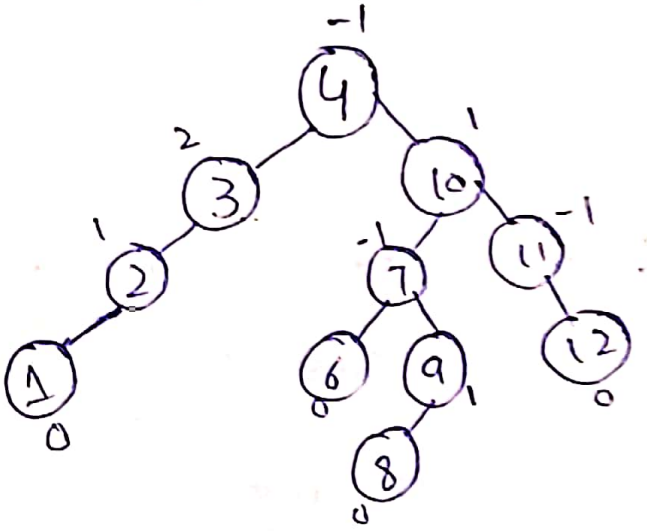
(d) AVL



18L-1051

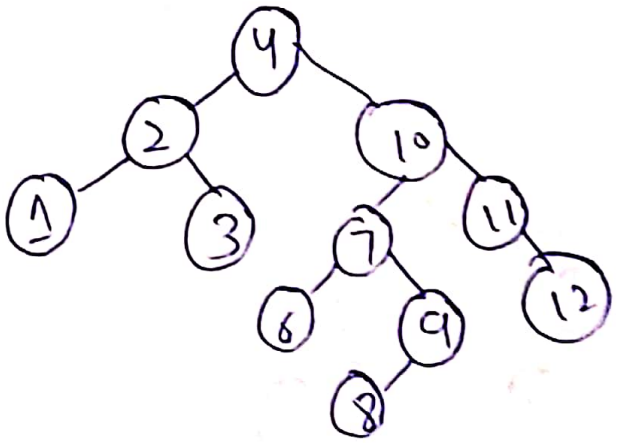
(a) BST after 5 is removed

Now replace 5 with 4 after removal



(b) Rebalance the tree

The balancing factor of Node 3 is 2 So, There occur right rotation that include Nodes => 3, 2, 1



18L-1051

(3)

QNo5

CPU capacity Intel(R) Core(TM) i7-8550U CPU @ 1.80 GHz 1.99 GHz.

RAM 8.00 GB

Cache L1 cache: 256 kB
L2 cache: 1.0 MB
L3 cache: 8.0 MB

Memory : 8 × 1024 × 1.5 = 12288 MB

(b)

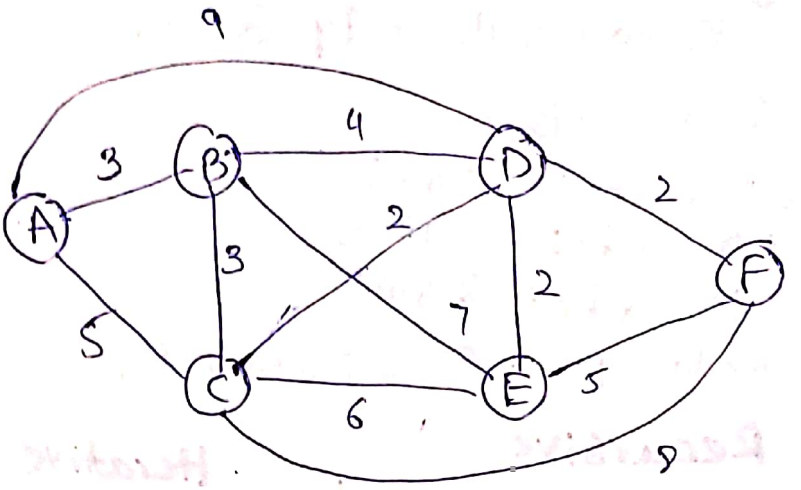
	Recursive			Iterative		
Inputsize	Inorder	Preorder	Postorder	Inorder	Preorder	Postorder
10	600	800	700	34200	46200	60300
50	1600	2000	2500	121100	115500	~~263500~~
500	12600	14500	14500	1164100	1107300	263900 / 2805800

(c) It is O(n) where n is number of nodes. Because algo needs to visit each node It is O(n+v) where v is number of edges Although in a tree v is n-1 so O(n) for trees

(d) Iterative is more good because it takes less memory. Recursive take a lot a memory in stack. So Iterative is best

18L-1051

QNO4



	dist	prev	dist
A	∞	undef	0
B	∞	undef	∞
C	∞	"	"
D	∞	"	"
E	∞	"	"
F	∞	"	"

Q = {A, B, C, D, E, F}

u = A

Q = {B, C, D, E, F}

① v = B, ② v = C

③ v = D

Now u = B

	dist	prev.
(v) B	3	A
C	5	A
D	9	A

A

dist 0

alt 0 + 3 = 3 (B)

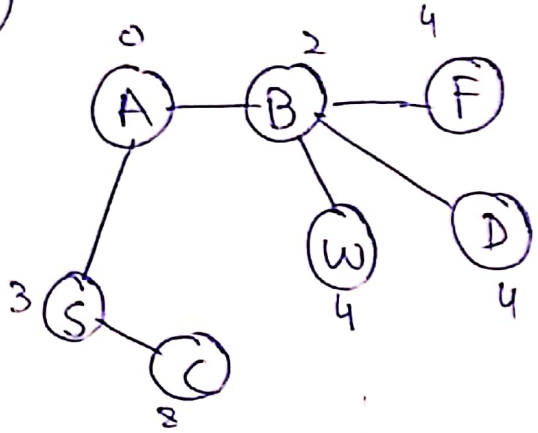
alt 0 + 5 = 5 (C)

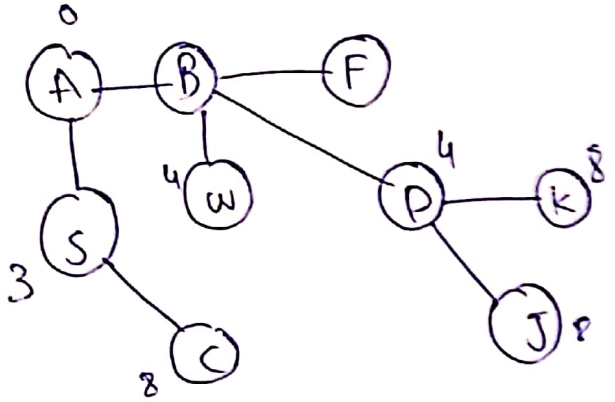
0 + 9 = 9 (D)

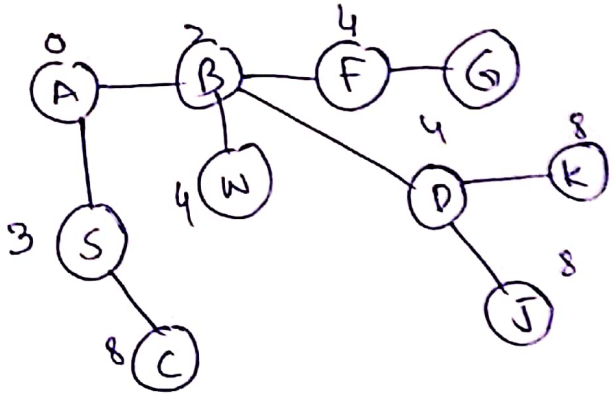
(a) B:

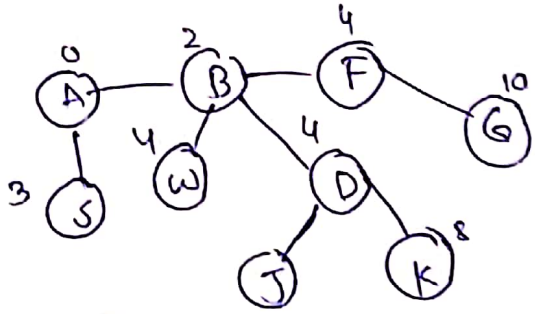
Q No 4

(c)









Q No 6

(c) In worst cast time complexity is O(n)