1. Suppose you have the following runtime data for an algorithm. What complexity class do they indicate?

Input	Seconds
1000	11
2000	90
4000	729
8000	5840
16000	46700

Answer: - O(N^3)

Explanation: - Every time the input is increased by two times, the time duration is increased by 8 roughly times,

2^1 ➔ 2^3

We can conclude if the Input increases by N times, the time duration will increase in N^3 times.

2. Consider the following code fragment. Based on its structure, what is its complexity class?

```
int m = 0;

for(int i = 0; i < n; i++)

        for(int j = 0; j < 5; j++)

                for(int k = 0; k < j; k++)

                        m += i;
```

Answer: - O(N)

Explanation :-The first for loop is dependent on input n. So, the time complexity is O(N). The second for loop depends on the constant value “5”. So its time complexity is O(1). The third for loop is dependent on the variable “j” from the 2nd for loop. Which means it is also indirectly dependent on constant value “5”. Which gives it the time complexity O(1). Because of this the total time complexity is O(N).

3. Suppose we run Selection Sort to sort the following array in increasing order. What does the array look like after each of the first 3 iterations, and why?

index	0	1	2	3	4	5	6
value	17	2	3	5	7	11	13

Answer: - {2,3,5,17,11,13}

Explanation: - In the first iteration we take the 0th index "17" and check whether it is smaller than the minimum value from the unsorted part. The minimum value from the unsorted part is "2". So, we swap "17" and "2".

In the second iteration we take the 1st index "17" and check whether it is smaller than the minimum value from the unsorted part. The minimum value from the unsorted part is "3". So, we swap "17" and "3".

In the third iteration we take the 2nd index "17" and check whether it is smaller than the minimum value from the unsorted part. The minimum value from the unsorted part is "5". So, we swap "17" and "5".

After the first 3 iterations the array will look like this,

{2,3,5,17,11,13}

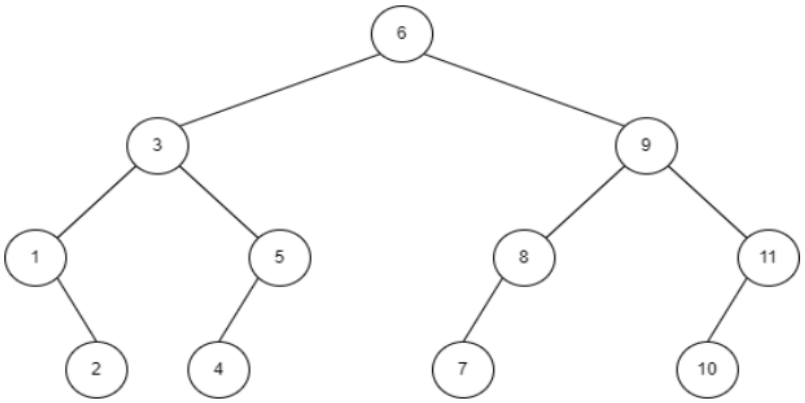
4. Suppose we use Binary Search to find the value 11 on the following array. Which array elements get checked, in which order, and why?

index	0	1	2	3	4	5	6
value	4	9	16	25	36	49	64

Answer: - 25,9,16

Explanation: - First we find the middle index of the array. Which is 3rd index. We check whether 11 is equal to the value of index 3(25). As it is not, we check whether it is larger than 25. Since it is we take the left side half of index 3. From that half we find the middle index, which is the 1st index. Then we again check whether the value of that index(9) is equal to 11. As it is not we check whether 11 is smaller than 9. Since it is not, we take the Right half of the middle index. And find the middle index of that half. Which is the 2nd index. Then again we check whether 11 is equal to the value of that index(16) and since it is not, and there aren't any indexes left in that part we conclude 11 is not in the array.

5. Suppose we remove the value 6 from the following binary search tree. What are the possible results?



To enter your answer, write the contents of each tree one level at a time, using "-" to indicate missing nodes. For example, the above tree would be written
6

3 9

1 5 8 11

- 2 4 - 7 - 10 –

Answer: -

5

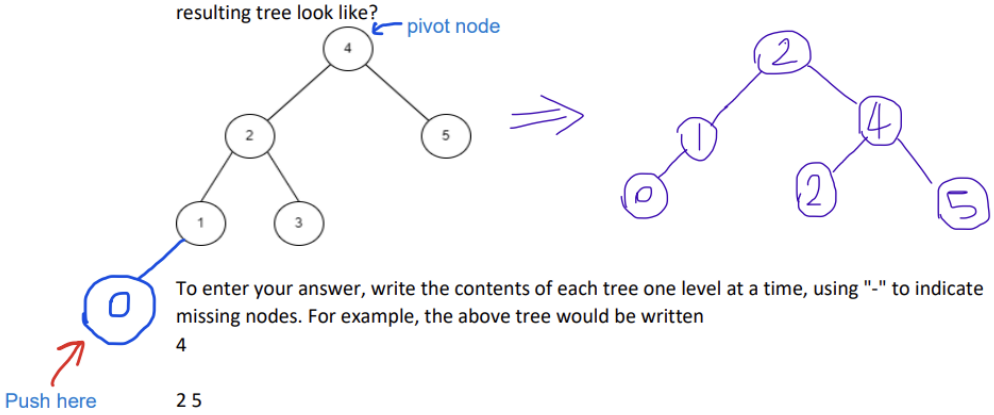
3 9

1 4 8 11

- 2 - - 7 - 10 –

Explanation :- Since the deleted node has two children, we have to replace the node with the rightmost value of the left subtree of the node. That node is 5. But 5 has a child, 4. So we connect 4 to 5's parent 3.

6. Suppose we insert the value 0 in the following AVL tree, and re-balance it. What does the resulting tree look like?



To enter your answer, write the contents of each tree one level at a time, using "-" to indicate missing nodes. For example, the above tree would be written

4

2 5

1 3 - -

Answer:-

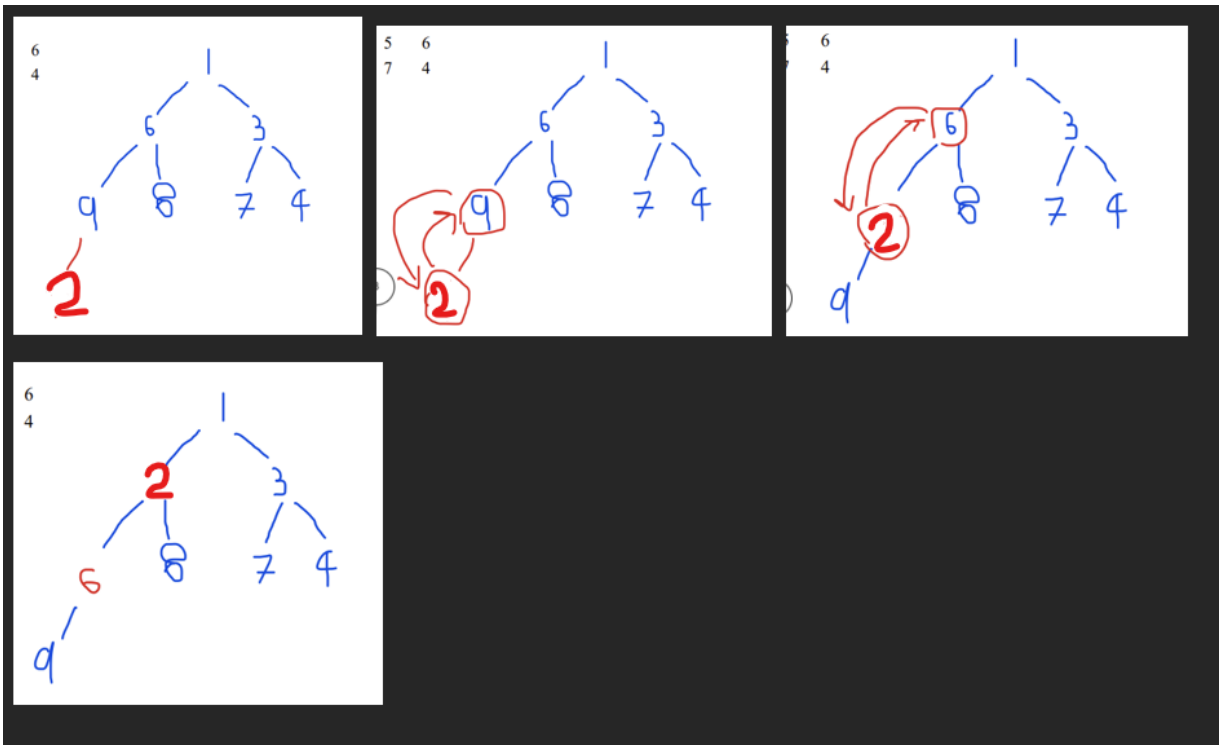
2

1 4

0 – 2 5

7. Consider the following array representation of a Min-Heap. How does this change after inserting the value 2?

index	0	1	2	3	4	5	6
value	1	6	3	9	8	7	4

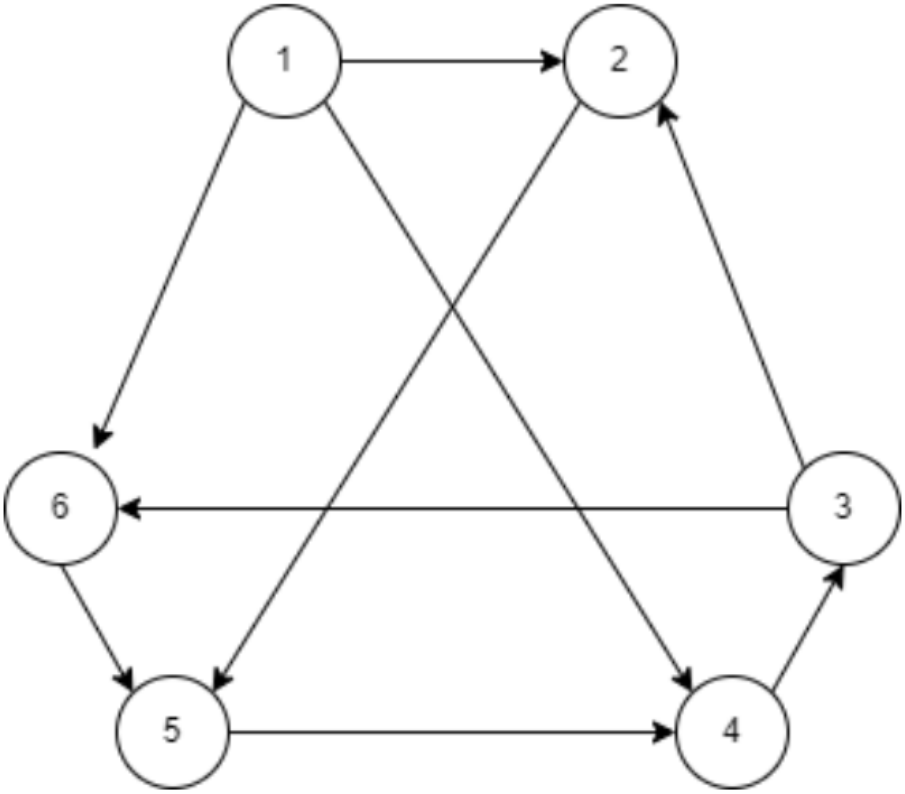


Answer: -

Index: 0 1 2 3 4 5 6

Value: 1 2 3 6 8 7 4

8. Consider the following directed graph:

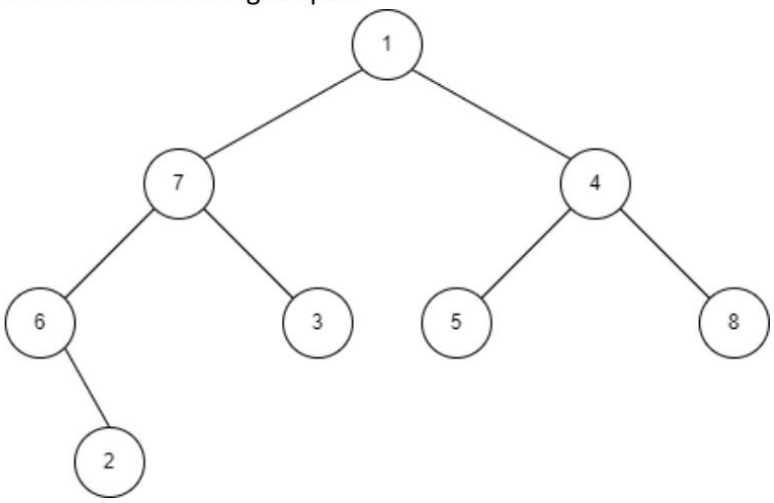


Write down the adjacency matrix of this graph.

Answer: -

	1	2	3	4	5	6
1	0	1	0	1	0	1
2	0	0	0	0	1	0
3	0	1	0	0	0	1
4	0	0	1	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	1	0

9. Suppose we use a pre-order traversal to output the following tree. What does this traversal do? What is the resulting output?



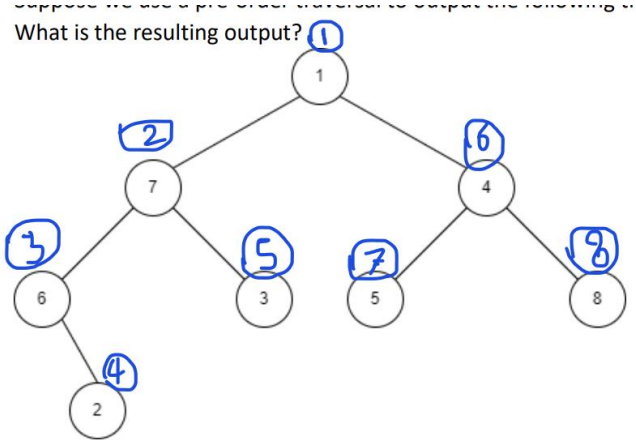
In order traversal = Left , Root, Right

Pre order Traversal = Root, Left, Right

Post order Traversal = Left, Right, Root

Asnswer :-

Pre-order traversal outputs the Root first and then left and right children.



1,7,6,2,3,4,5,8