Lec-36

08/02/25 1:00 PM

```
Cars[] arr = new Cars[5];
```

```
public static void Sort(int[] arr) {
    for (int trurn = 1; trurn < arr.length; trurn++) {
        for (int i = 0; i < arr.length - trurn; i++) {
            if (arr[i] > arr[i + 1]) {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }
    }
}
```

```
public static <T>void Sort(T[] arr) {

        for (int trurn = 1; trurn < arr.length; trurn++) {
            for (int i = 0; i < arr.length - trurn; i++) {
                if (arr[i] > arr[i + 1]) {
                    T temp = arr[i];
                    arr[i] = arr[i + 1];
                    arr[i + 1] = temp;
                }
            }
        }
    }
```

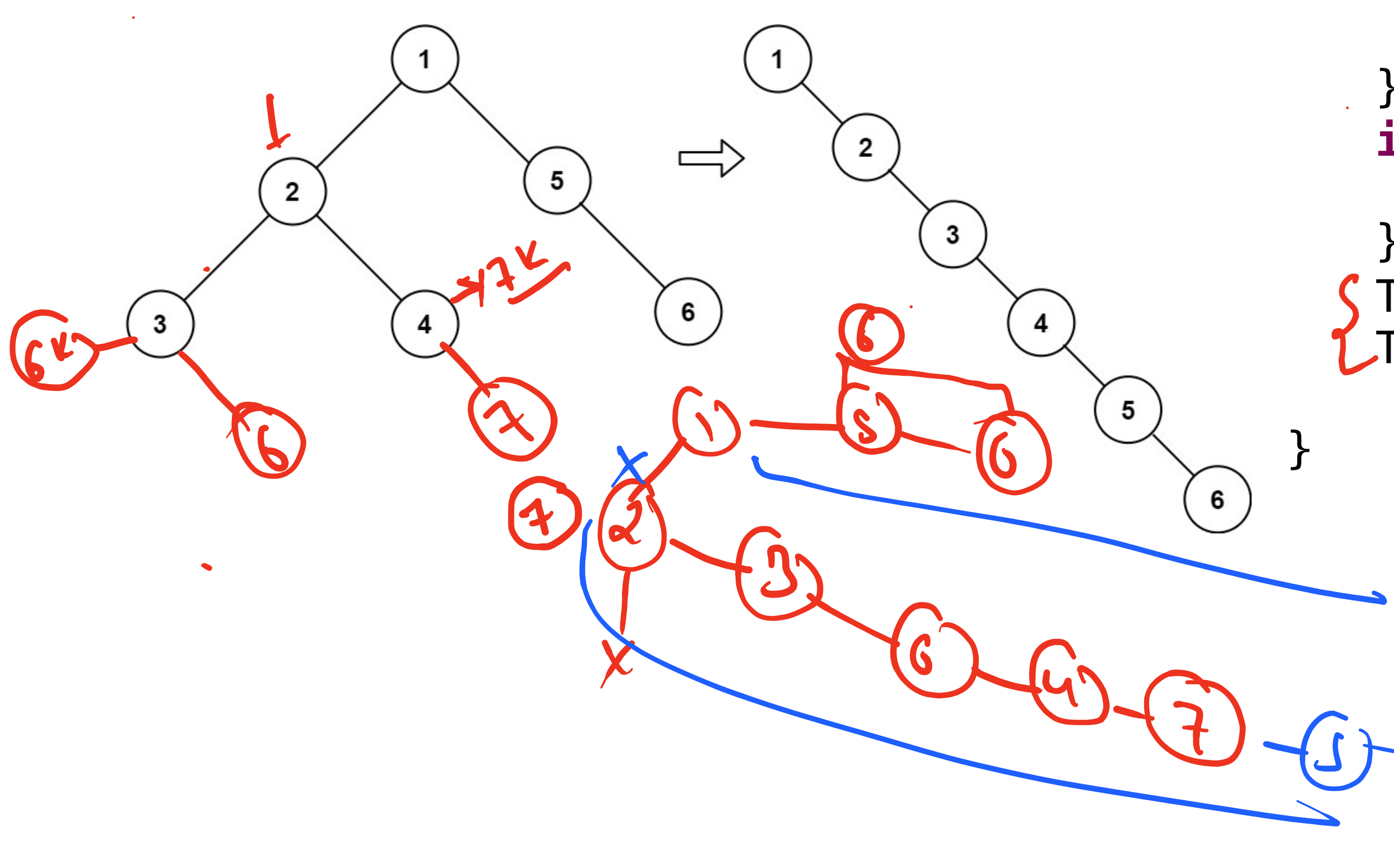
```
Cars[] arr = new Cars[5];
    arr[0] = new Cars(200, 10, "White");// P S C
    arr[1] = new Cars(1000, 20, "Black");
    arr[2] = new Cars(345, 3, "Yellow");
    arr[3] = new Cars(34, 89, "Grey");
    arr[4] = new Cars(8907, 6, "Red");
```

```
@Override
    public int compareTo(Cars o) {
        // TODO Auto-generated method stub
        return 0;
    }
```

```
public static <T extends Comparable<T>> void Sort(T[] arr) {

        for (int trurn = 1; trurn < arr.length; trurn++) {
            for (int i = 0; i < arr.length - trurn; i++) {
                if (arr[i].compareTo(arr[i + 1]) > 0) {
                    T temp = arr[i];
                    arr[i] = arr[i + 1];
                    arr[i + 1] = temp;
                }
            }
        }
    }
```

```
6
7 9
0 10
4 5
8 9
4 10
5 7
```



```
private TreeNode MakeLL(TreeNode root) {
            // TODO Auto-generated method stub
            if(root==null) {
                return null;
            }
            if(root.left==null && root.right==null) {
                return root;
            }
            TreeNode left_tail=MakeLL(root.left);
            TreeNode right_tail=MakeLL(root.right);
    }
```

```
private TreeNode MakeLL(TreeNode root) {
            // TODO Auto-generated method stub
            if(root==null) {
                return null;
            }
            if(root.left==null && root.right==null) {
                return root;
            }
            TreeNode left_tail=MakeLL(root.left);
            TreeNode right_tail=MakeLL(root.right);
            if(left_tail!=null) {
                left_tail.right=root.right;
                root.right=root.left;
                root.left=null;
            }
            return right_tail !=null ? right_tail : left_tail;

        }
```