Agenda

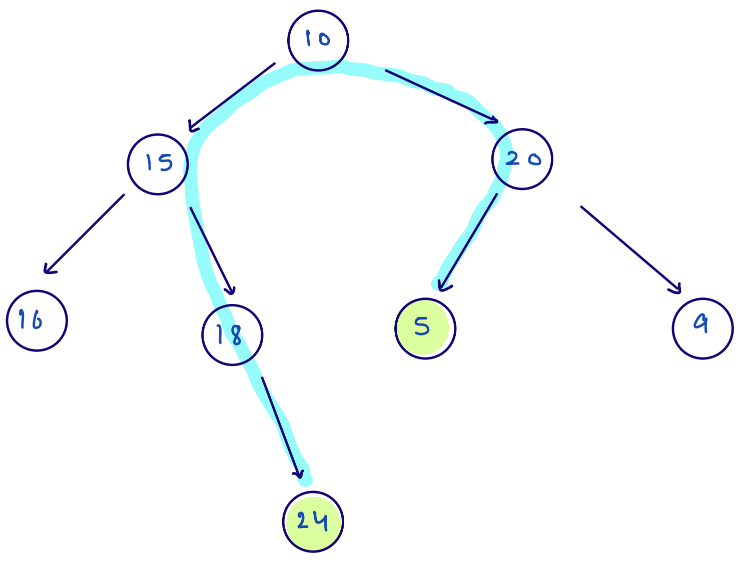
1) Diameter of Binary tree

2) Serialize & Deserialize Binary tree

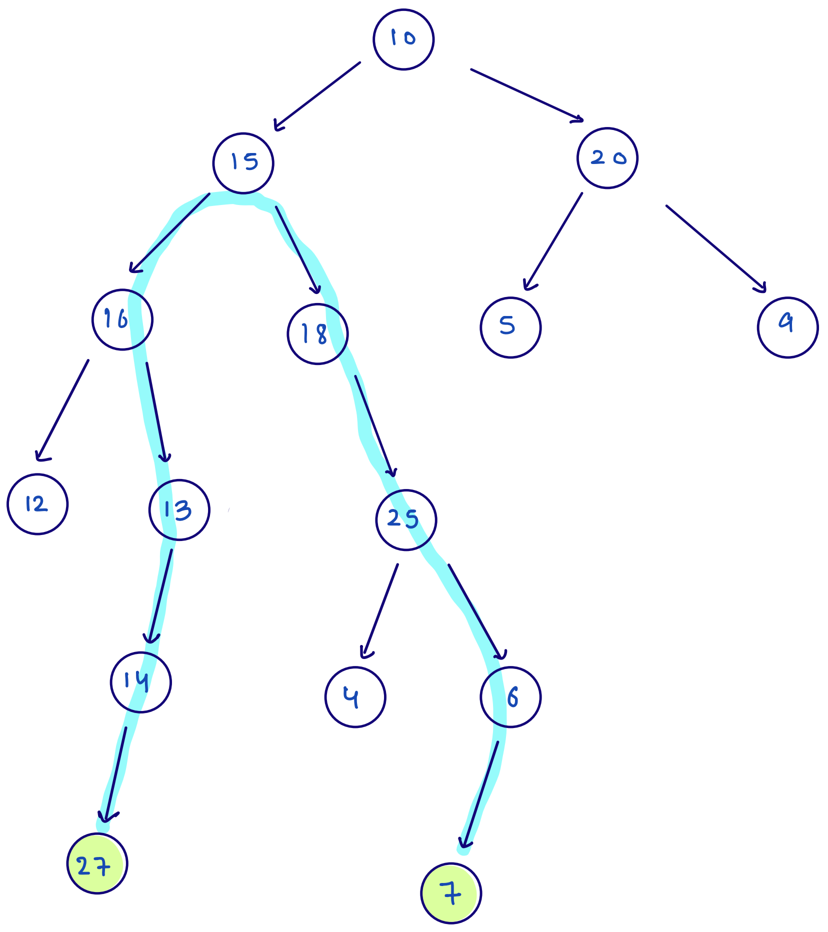
3) TreeMap Introduction & usage

Q.1 Given root of a binary tree, find its diameter.

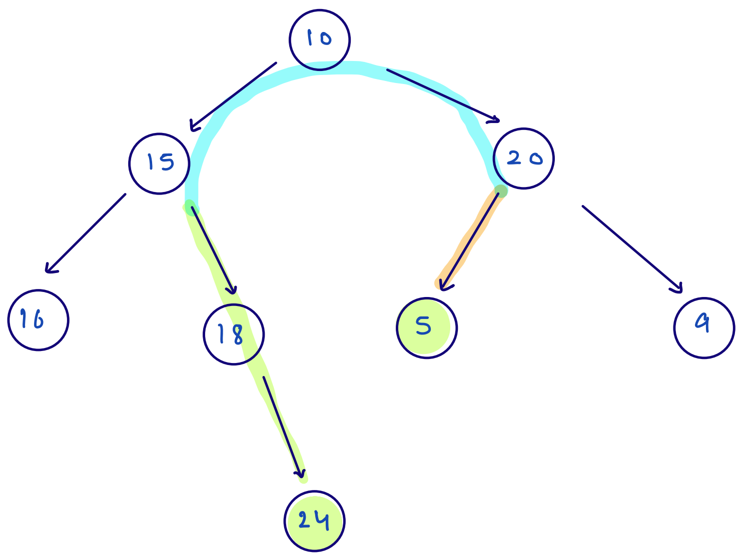
Note : The diameter of binary tree is the length of the longest path b/w any two nodes. This path may or may not pass through the root.



diameter = 5



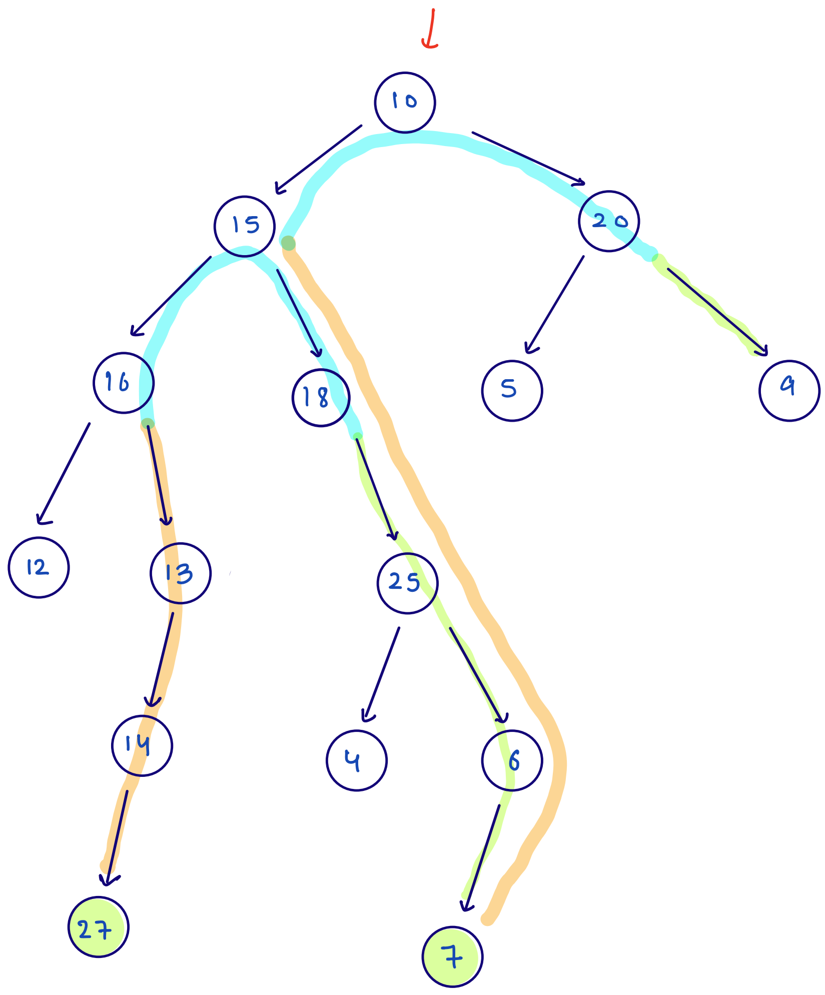
diameter = 8



lh = distance from left child to deepest node in left subtree

rh = distance from right child to deepest node in right subtree

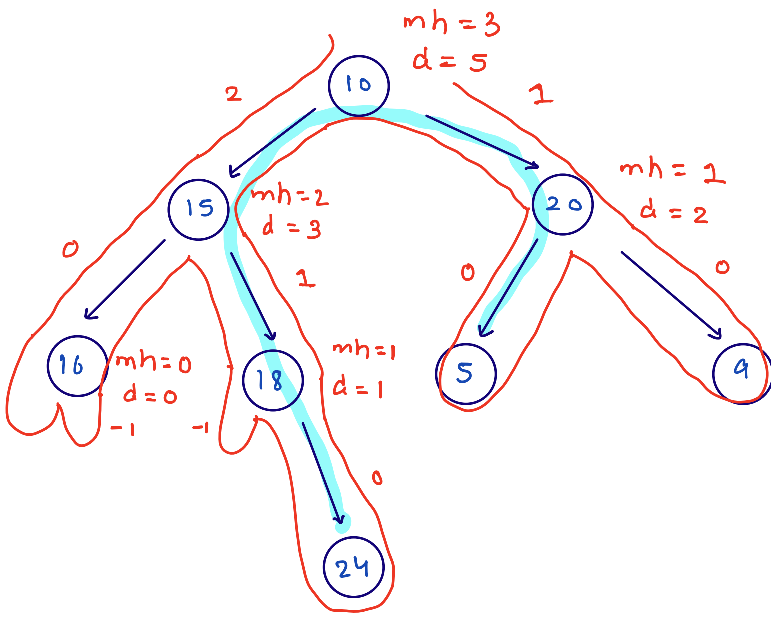
dist = lh + rh + 2



node	lh	rh	dist
15	3	3	8
10	4	1	7

```
int maxDist = 0;

int height (Node root) {
    if( root == null) {
        return -1;
    }

    int lh = height (root.left);

    int rh = height (root.right);

    int mh = Math.max (lh, rh) +1;

    int d = lh+rh+2;
    if (d > maxDist ) {
        maxDist = d;
    }

    return mh;

}

int diameter ( Node root) {
    maxDist=0;
    height (root);
    return maxDist;
}
```

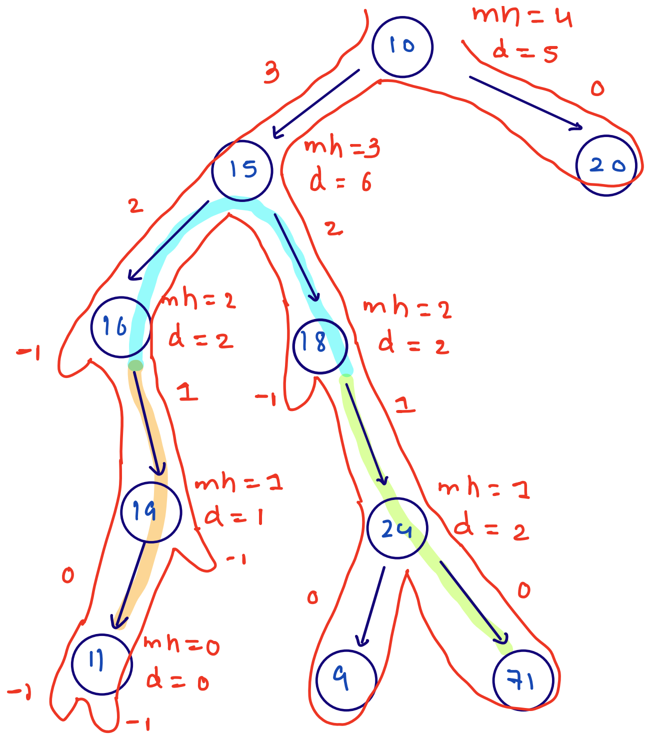
```
int maxDist = 0;

int height (Node root) {
    if( root==null) {
        return -1;
    }
    int lh= height (root.left);
    int rh= height (root.right);
    int mh= Math.max (lh, rh)+1;

    int d = lh+rh+2;
    if (d > maxDist) {
        maxDist = d;
    }
    return mh;
}
```



maxDist = ~~0~~ ~~1~~ ~~3~~ 5

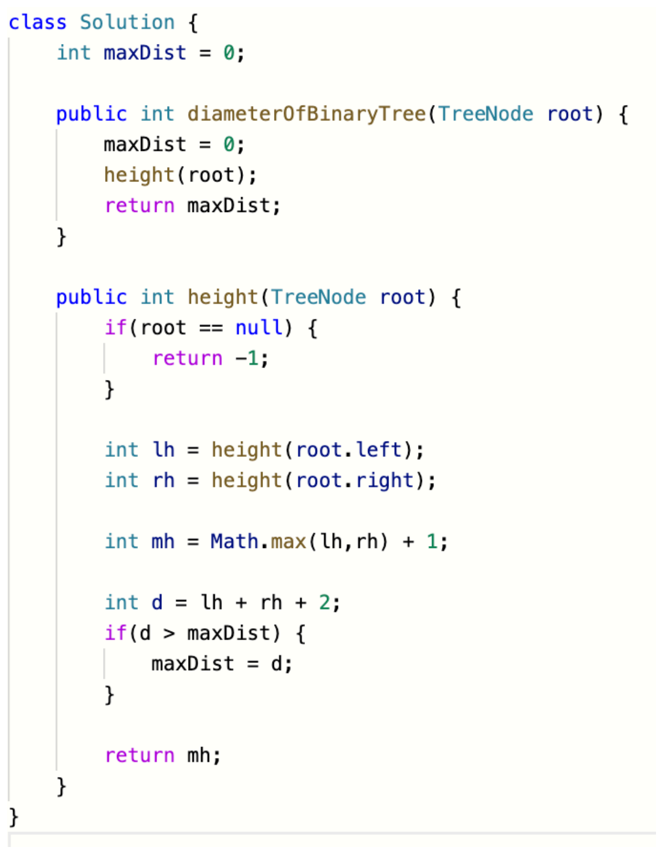
```
int diameter (Node root) {
    maxDist=0;
    height (root);
    return maxDist;
}
```

```
int maxDist = 0;

int height (Node root) {
    if( root = = null) {
        return -1;
    }
    int lh= height (root.left);
    int rh= height (root.right);
    int mh= Math.max (lh, rh) +1;
    int d = lh+rh+2;
    if (d > maxDist ) {
        maxDist = d;
    }
    return mh;
}

int diameter ( Node root) {
    maxDist=0;
    height (root);
    return maxDist;
}
```



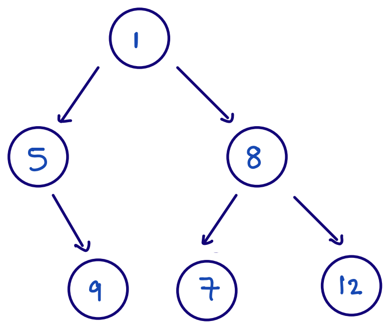
maxDist = ~~0~~ ~~2~~ ~~2~~ 6

https://leetcode.com/problems/diameter-of-binary-tree/



```java
class Solution {
    int maxDist = 0;

    public int diameterOfBinaryTree(TreeNode root) {
        maxDist = 0;
        height(root);
        return maxDist;
    }

    public int height(TreeNode root) {
        if(root == null) {
            return -1;
        }

        int lh = height(root.left);
        int rh = height(root.right);

        int mh = Math.max(lh,rh) + 1;

        int d = lh + rh + 2;
        if(d > maxDist) {
            maxDist = d;
        }

        return mh;
    }
}
```

Q. 2 Serialize and Deserialize a Binary tree.

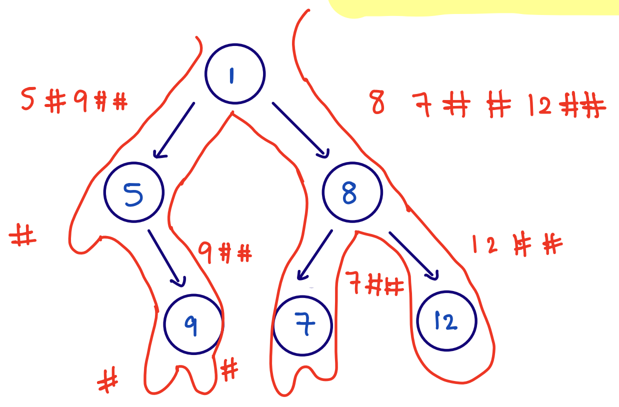


Serialize : convert info of BT into a string

Deserialize : with this same string convert back to the binary tree

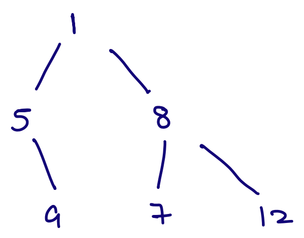
Serilization : Binary tree info into string.

1 5 # 9 # # 8 7 # # 12 # #



→ node.val + " " + lat
" " + ra;

1 5 # 9 # # 8 7 # # 12 # #



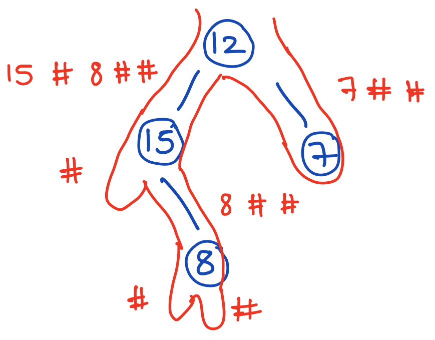
```
String serialization (Node root) {
    if(root == null) {
        return "#";
    }
    String la = serialization (root.left);
    String ra = serialization (root.right);
    String ma = node.val + " " + la + " " + ra;
    return ma;
}
```

---

dry run

```
String serialization (Node root) {
    if(root == null) {
        return "#";
    }
    String la = serialization (root.left);
    String ra = serialization (root.right);
    String ma = node.val + " " + la + " " + ra;
    return ma;
}
```

12 15 # 8 # # 7 # #



Deserialization : from the serialized string that you have created in above logic, give back the tree.

↓

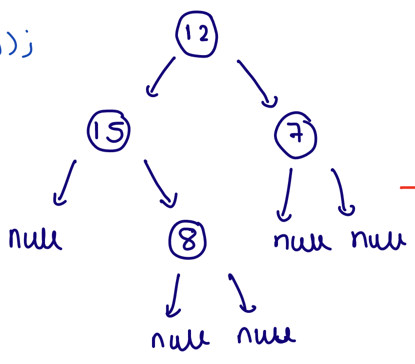
12 15 # 8 # # 7 # #

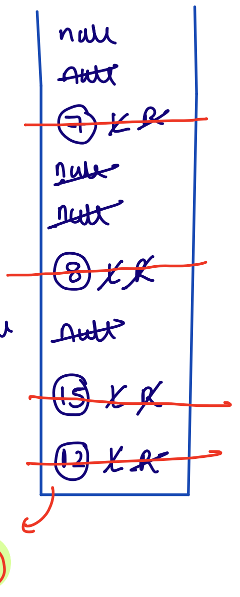
```
if I see a data -> node creation
        -> left call
        -> right call

else
    return null
```

```
int idx = 0;                          ↱ serialized string
Node Deserialize (String str) {
      String[] arr = str.split(" ");
       idx = 0;
       return helper (arr);
}

Node helper (String[] arr) {
      if (arr[idx].equals("#")) {
              idx++;
              return null;
      }
       else {
             int val = Integer.parseInt(arr[idx]);
             Node nn = new Node(val);
             idx++;
             nn.left = helper(arr);
             nn.right = helper(arr);
             return node;
       }
}
```

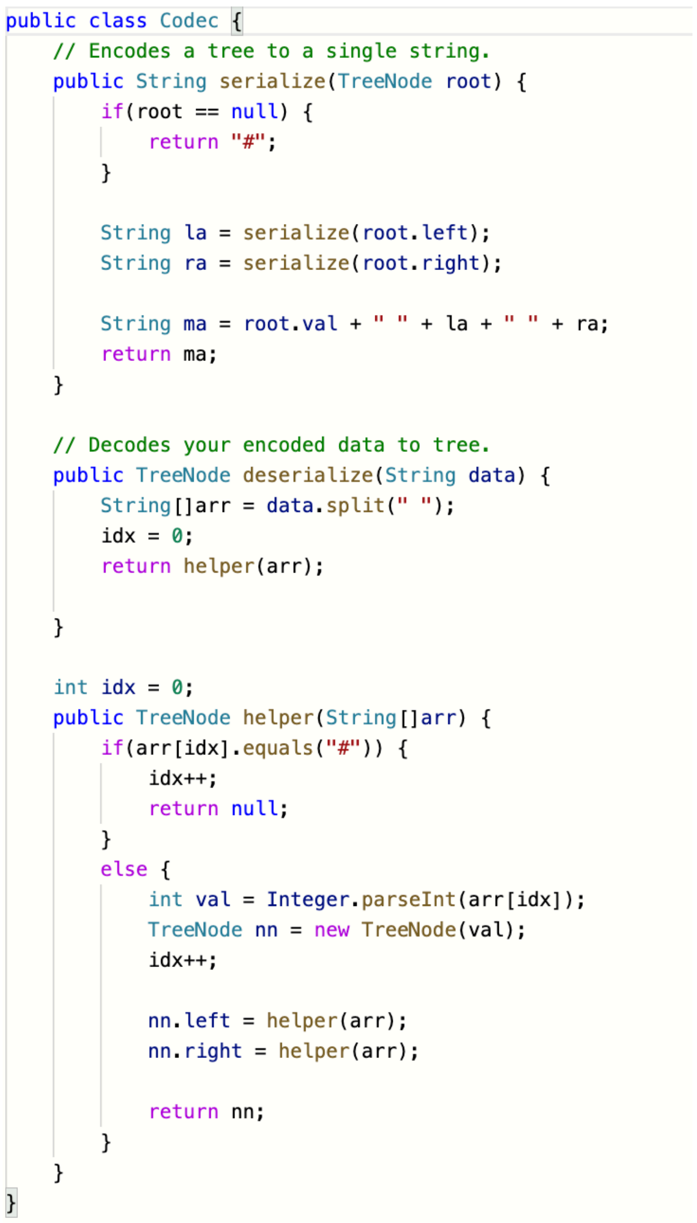
```
Node helper(String[] arr) {
    if (arr[idx].equals("#")) {
        idx++;
        return null;
    }
    else {
        int val = Integer.parseInt(arr[idx]);
        Node nn = new Node(val);
        idx++;
        nn.left = helper(arr);
        nn.right = helper(arr);
        return nn;
    }
}
```

arr: ["12", "15", "#", "8", "#", "#", "7", "#", "#"]
 0 1 2 3 4 5 6 7 8





https://leetcode.com/problems/serialize-and-deserialize-binary-tree/



```java
public class Codec {
    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        if(root == null) {
            return "#";
        }

        String la = serialize(root.left);
        String ra = serialize(root.right);

        String ma = root.val + " " + la + " " + ra;
        return ma;
    }

    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        String[]arr = data.split(" ");
        idx = 0;
        return helper(arr);

    }

    int idx = 0;
    public TreeNode helper(String[]arr) {
        if(arr[idx].equals("#")) {
            idx++;
            return null;
        }
        else {
            int val = Integer.parseInt(arr[idx]);
            TreeNode nn = new TreeNode(val);
            idx++;

            nn.left = helper(arr);
            nn.right = helper(arr);

            return nn;
        }
    }
}
```

Introduction to TreeMap

Treemap is a sorted HashMap.
↳ based on key values

```
TreeMap < String, Integer> tm = new TreeMap<>();
 tm.put ("India", 244);
  tm.put (" Australia", 444);
   tm.put (" China", 439);
    tm.put ("Bharat", 416);
     tm.put (" Pak", 389);
     sopln (tm.containskey (" Eng"))); → false
     sopln (tm.get (" Bharat")); → 416
```

HashMap	TreeMap
put, containskey, get, remove	put, containskey, get, remove, ceilingkey, floorkey
O(1)	O(log n)

Ide Link :

https://www.interviewbit.com/snippet/9bd6263c3cdcbe54e0fd/