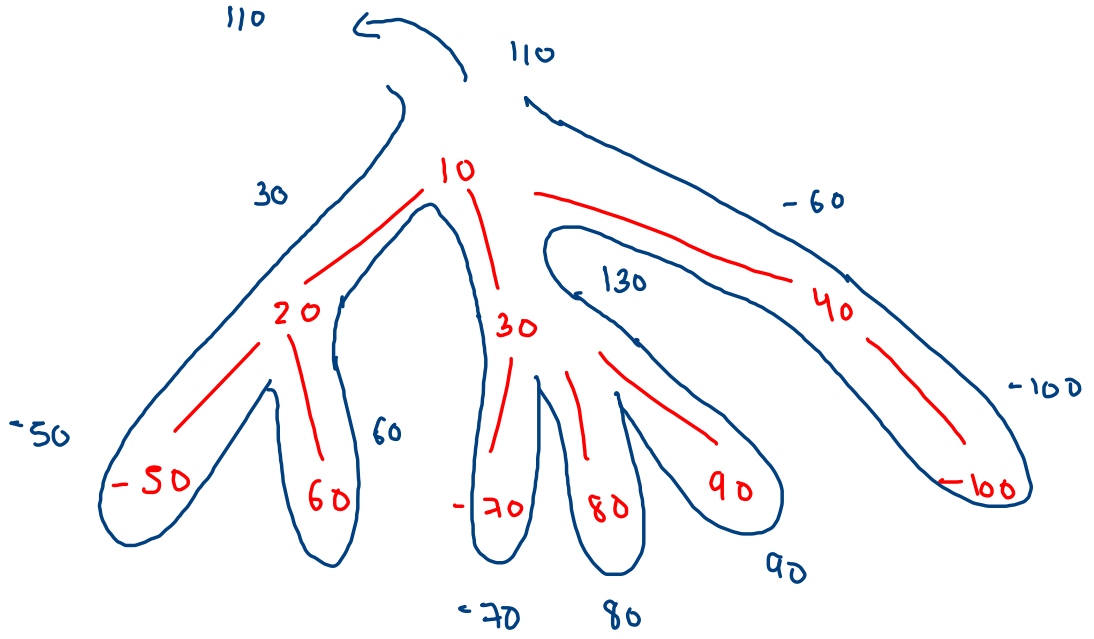
Node With Maximum Subtree Sum

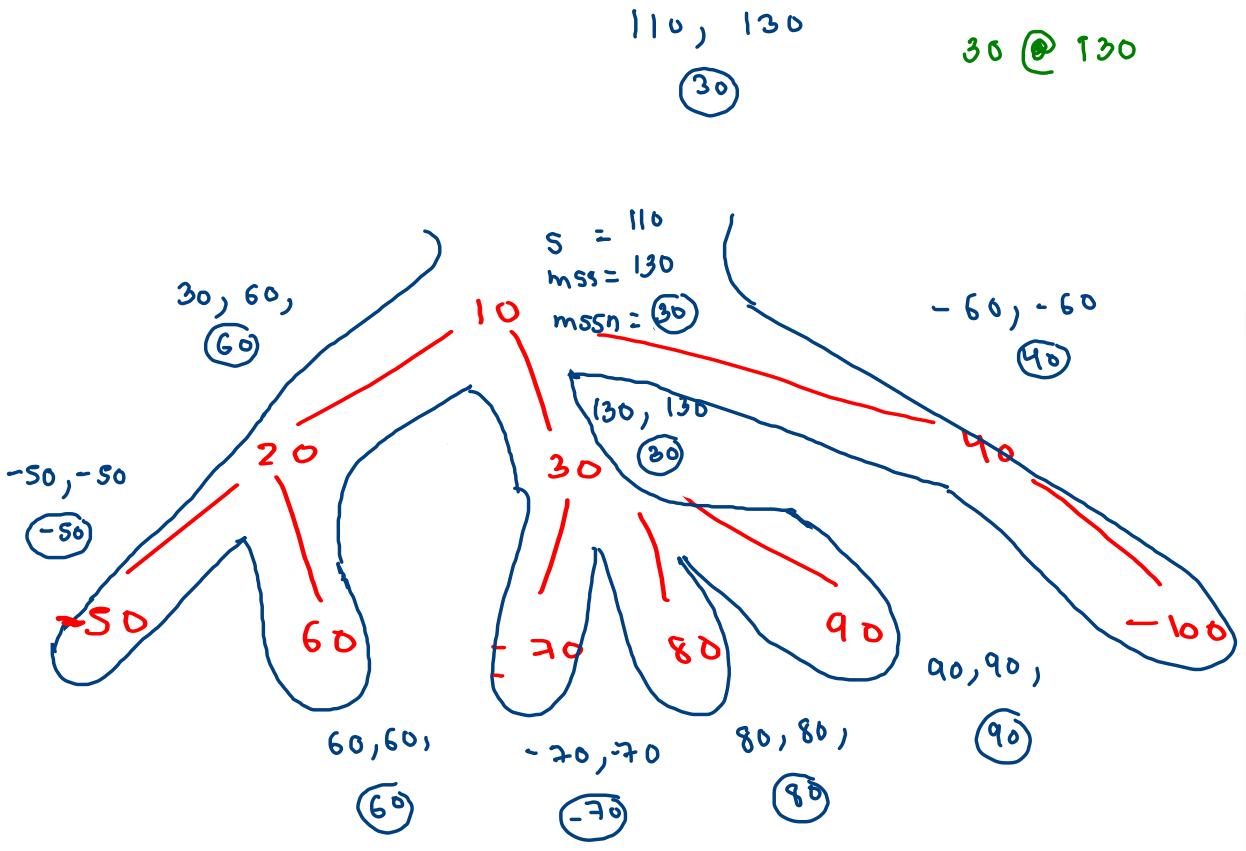
(i) static variable



mss = ~~-∞~~ ~~-50~~ ~~80~~ ~~60~~ ~~90~~ 130

mssn = ~~-∞~~ ~~-50~~ ~~60~~ ~~80~~ ~~90~~ 30

30 @ 130



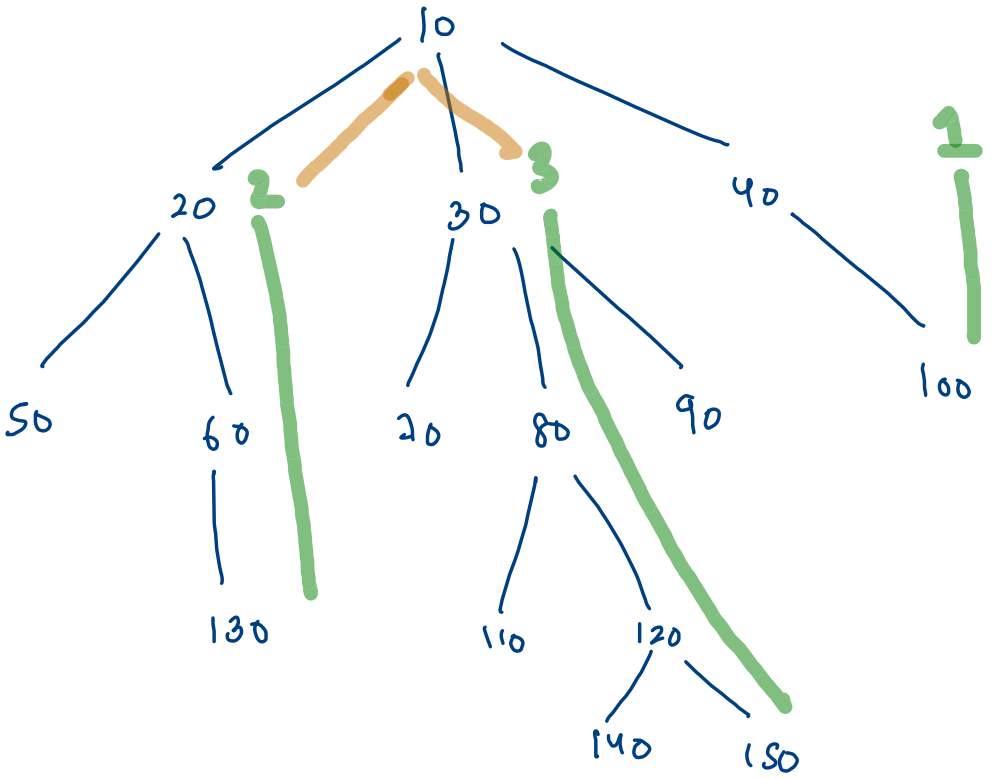

```java
public static class Pair {
    int sum;
    int mss; //max subtree sum
    Node mssn; //max subtree sum node

    Pair() {

    }

    Pair(int sum,int mss,Node mssn) {
        this.sum = sum;
        this.mss = mss;
        this.mssn = mssn;
    }
}
```

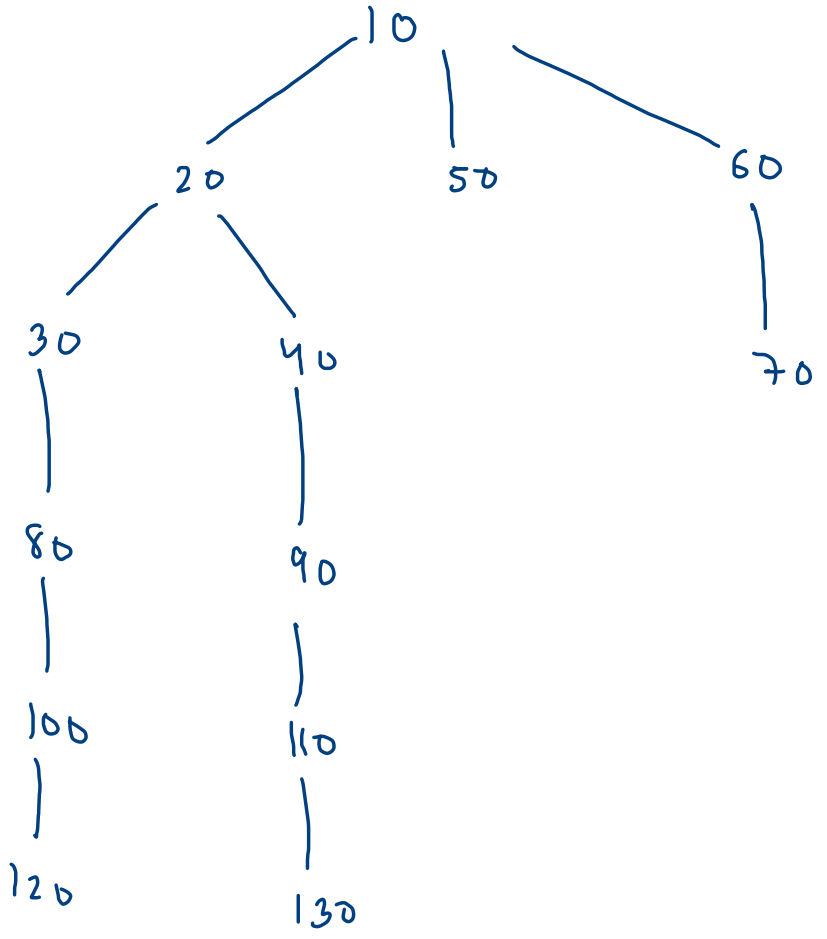
```java
public static Pair maxSubtreeSum(Node node) {
    int sum = node.data;
    int mss = Integer.MIN_VALUE;
    Node mssn = null;

    for(int i=0; i < node.children.size();i++) {
        Node child = node.children.get(i);
        Pair cp = maxSubtreeSum(child);

        sum += cp.sum;

        if(cp.mss > mss) {
            mss = cp.mss;
            mssn = cp.mssn;
        }
    }


    //node's contendor (subtree rooted at node)
    if(sum > mss) {
        mss = sum;
        mssn = node;
    }

    return new Pair(sum,mss,mssn);
}
```

Diameter Of Generic Tree

2. You are required to find and print the diameter of tree. THe diameter is defined as maximum number of edges between any two nodes in the tree. Check the question video for clarity.

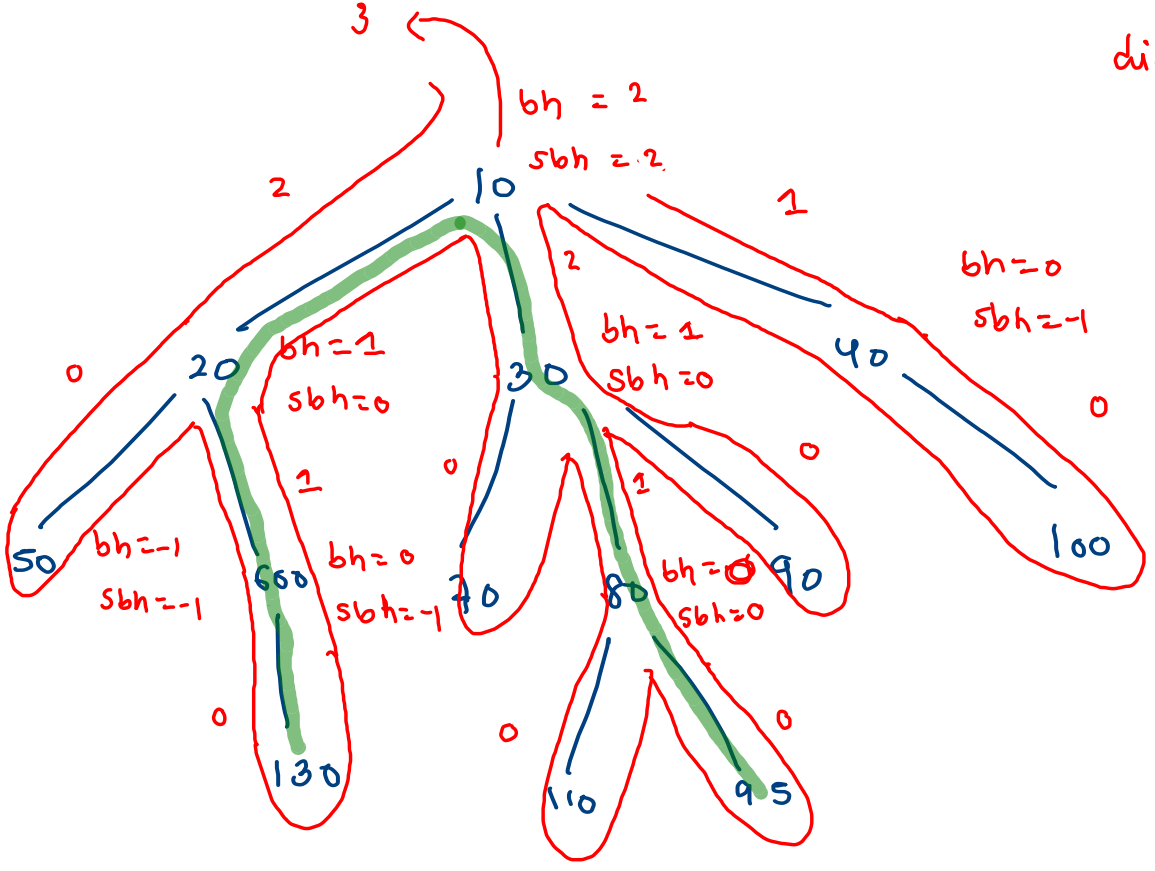
O(n)





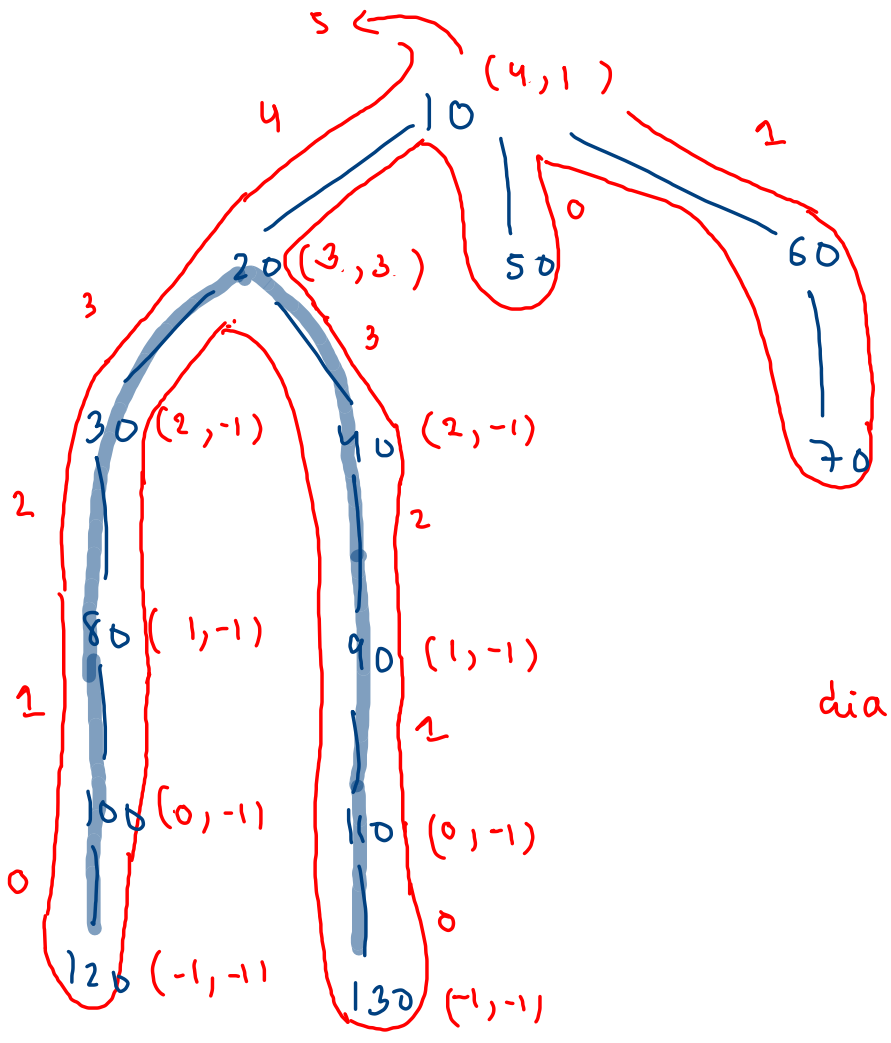
dia →

cont ; bht + sbht + 2



dia = ~~0~~ ~~2~~ ~~3~~ 6

```java
public static int height(Node node) {
    int bcht = -1; //best child height
    int sbcht = -1; //second best child height

    for(int i=0; i < node.children.size();i++) {
        Node child = node.children.get(i);
        int cht = height(child);

        if(cht > bcht) {
            sbcht = bcht;
            bcht = cht;
        }
        else if(cht > sbcht) {
            sbcht = cht;
        }
    }

    int dist = bcht + sbcht + 2; //node's contendor

    if(dist > dia) {
        dia = dist;
    }

    return bcht + 1;
}
```

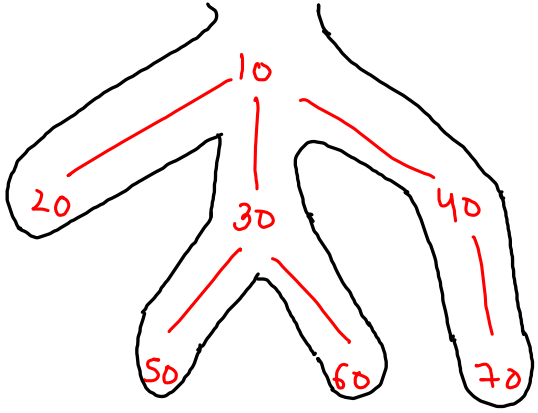


(bh, sbh)

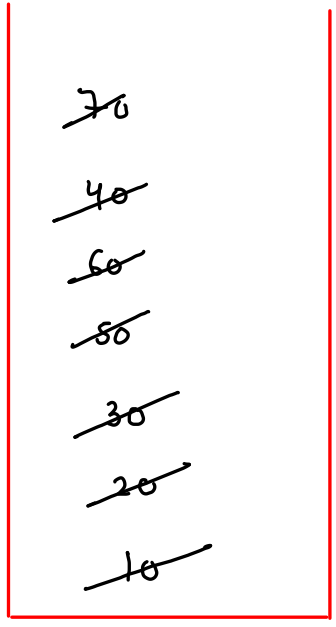
dia = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 8

```java
public static int height(Node node) {
    int bcht = -1; //best child height
    int sbcht = -1; //second best child height

    for(int i=0; i < node.children.size();i++) {
        Node child = node.children.get(i);
        int cht = height(child);

        if(cht > bcht) {
            sbcht = bcht;
            bcht = cht;
        }
        else if(cht > sbcht) {
            sbcht = cht;
        }
    }

    int dist = bcht + sbcht + 2; //node's contendor

    if(dist > dia) {
        dia = dist;
    }

    return bcht + 1;
}
```

Iterative Preorder And Postorder Of Generic Tree

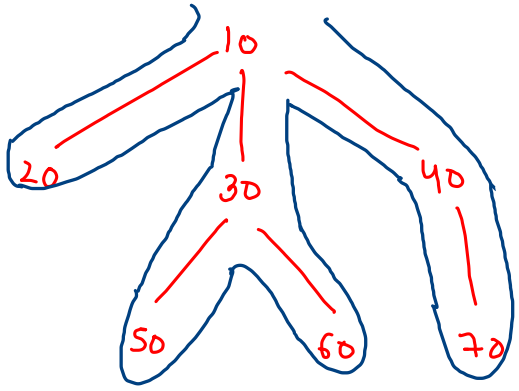



```
fun (node) {
    pre += node.data;
    [ loop : child
    post += node.data
}
```



pre = 10 20 30 50 60 40 70

post = 20 50 60 30 70 40 10



Pair :

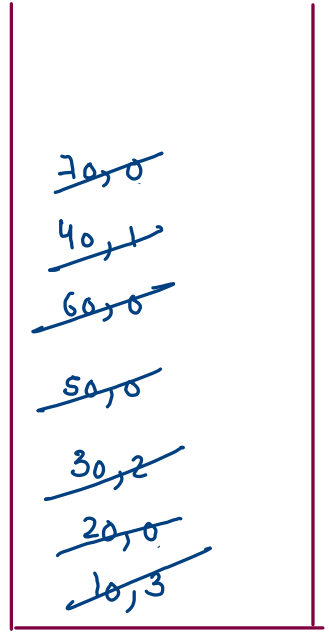
Node node;

int state;

-1 : pre, st++

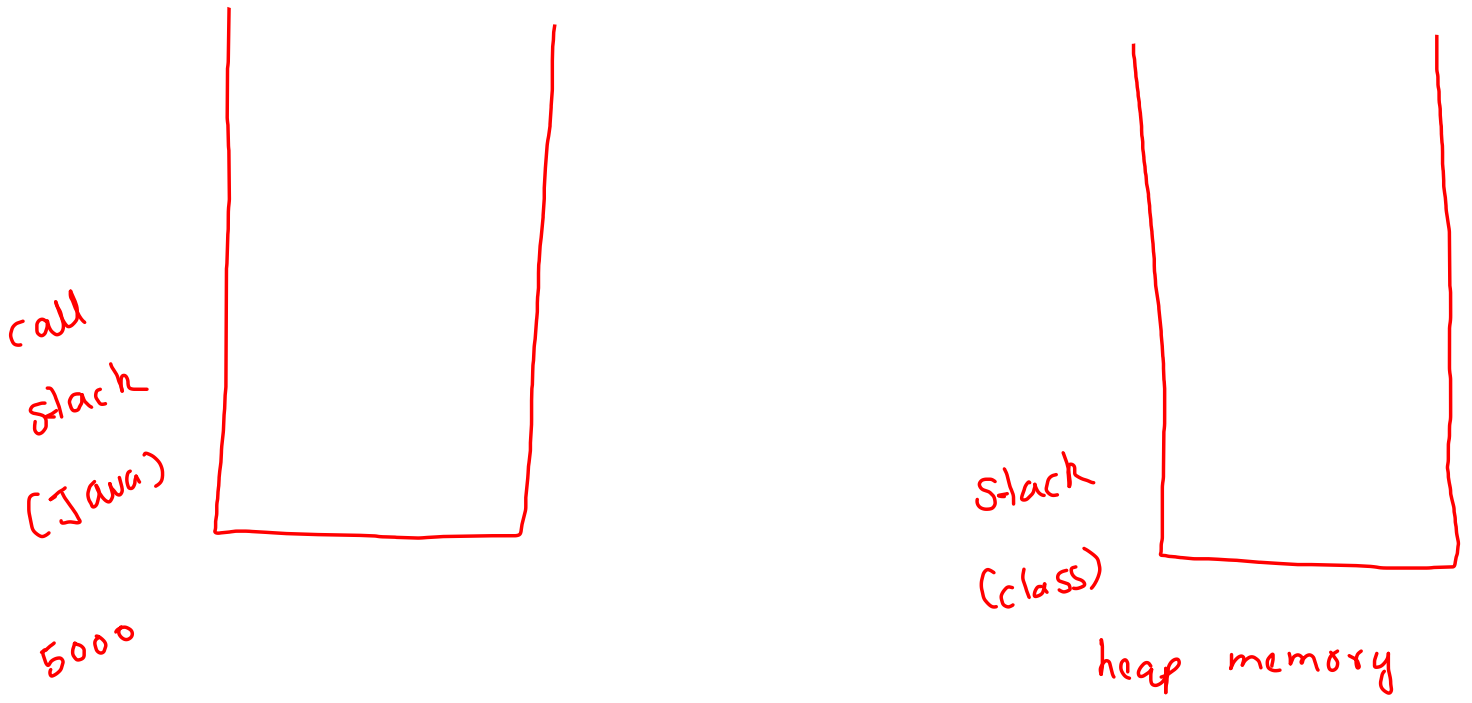
0 to child.size-1 : push st's index child, st++

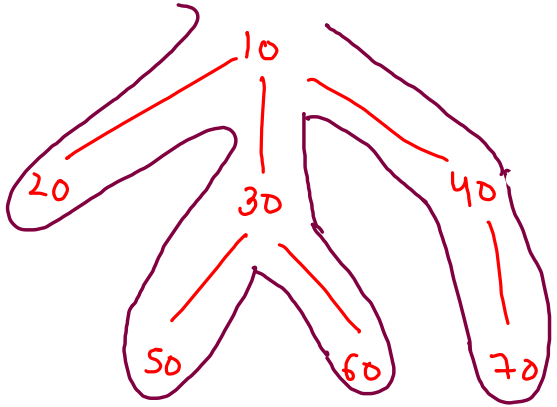
child.size : post, pop



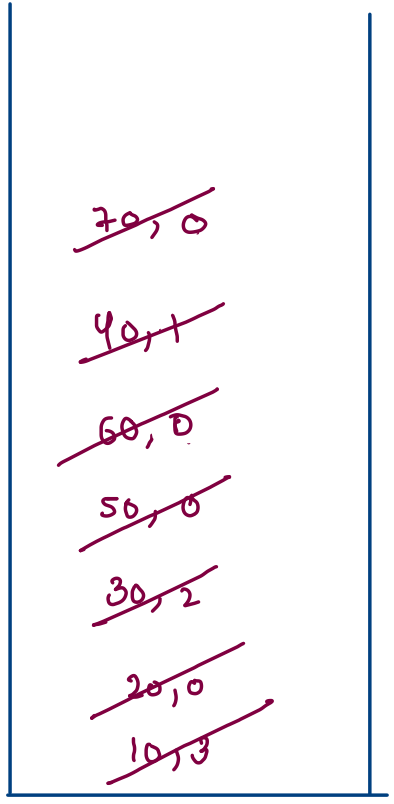
pre : 10 20 30 50 60 40 70

post : 20 50 60 30 70 40 10


call
stack
(Java)
5000
stack
(class)
heap memory




```
while(st.size() > 0) {
    Pair top = st.peek();
    Node node = top.node;
    int state = top.state;

    if(state == -1) {
        //pre area
        pre += (node.data + " ");
        top.state++;
    }
    else if(state < node.children.size()) {
        Node child = node.children.get(state);
        st.push(new Pair(child,-1));
        top.state++;
    }
    else if(state == node.children.size()) {
        //post
        post += (node.data + " ");
        st.pop();
    }
}
```



pre: 10 20 30 50 60 40 70

post: 20 50 60 30 70 40 10

Pair {

node;

state;

}