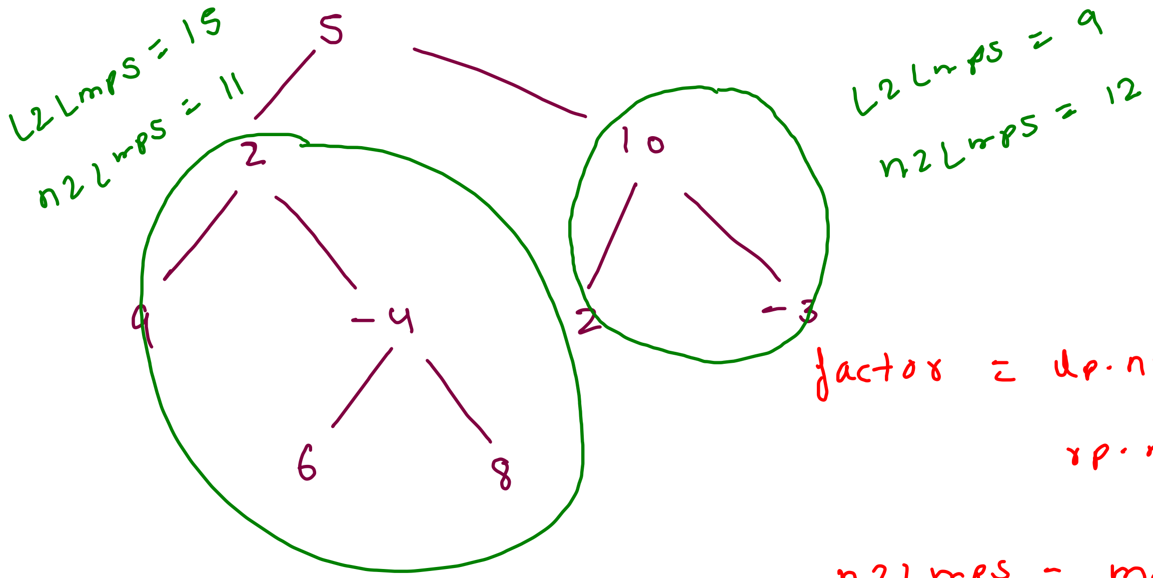
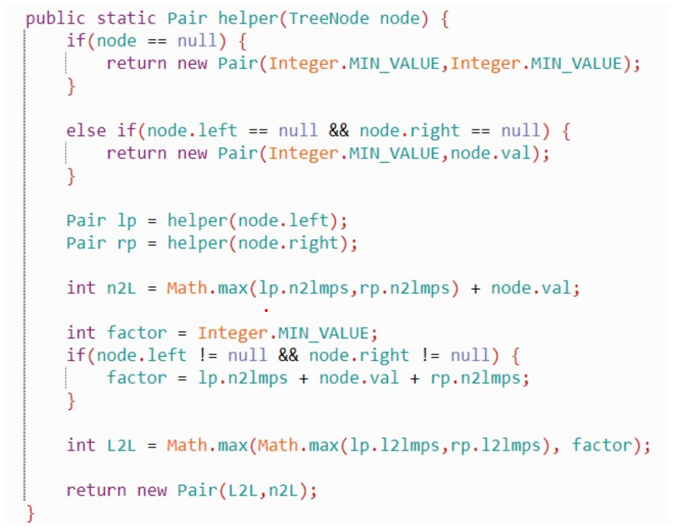
max path sum L2L



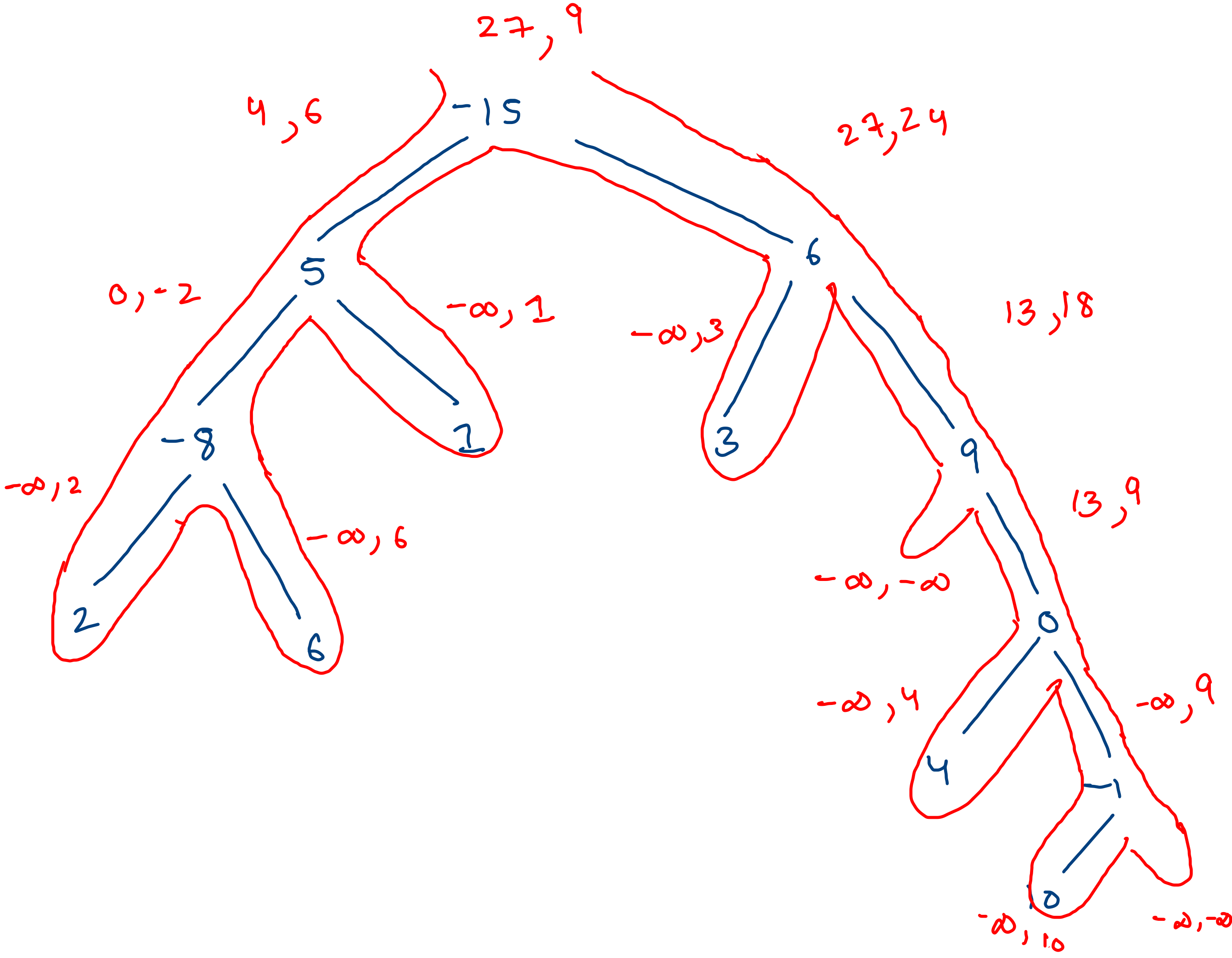
factor = lp.n2Lmps + node.data + rp.n2Lmps.

n2Lmps = max (lp.n2Lmps, rp.n2Lmps) + node.data ;

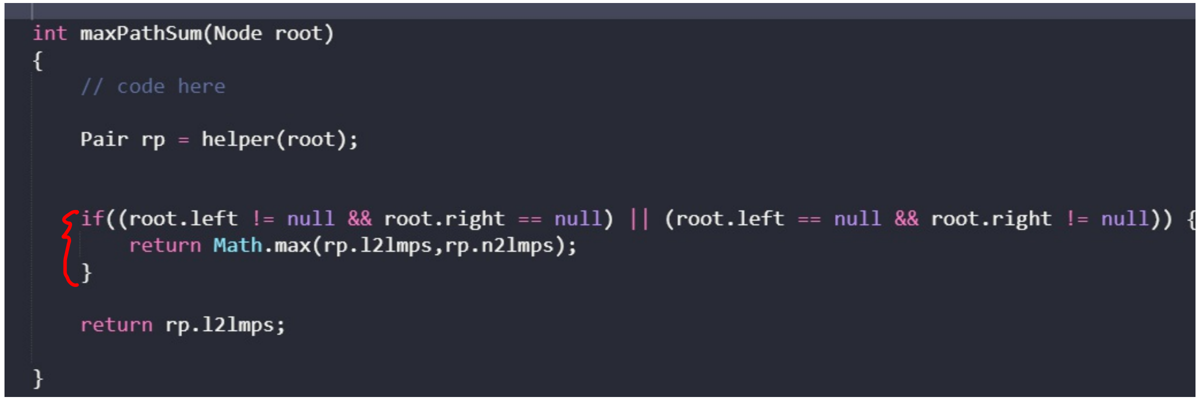
L2Lmps = max (lp.L2Lmps, rp.L2Lmps, factor)



```java
public static Pair helper(TreeNode node) {
    if(node == null) {
        return new Pair(Integer.MIN_VALUE,Integer.MIN_VALUE);
    }

    else if(node.left == null && node.right == null) {
        return new Pair(Integer.MIN_VALUE,node.val);
    }

    Pair lp = helper(node.left);
    Pair rp = helper(node.right);

    int n2L = Math.max(lp.n2lmps,rp.n2lmps) + node.val;

    int factor = Integer.MIN_VALUE;
    if(node.left != null && node.right != null) {
        factor = lp.n2lmps + node.val + rp.n2lmps;
    }

    int L2L = Math.max(Math.max(lp.l2lmps,rp.l2lmps), factor);

    return new Pair(L2L,n2L);
}
```

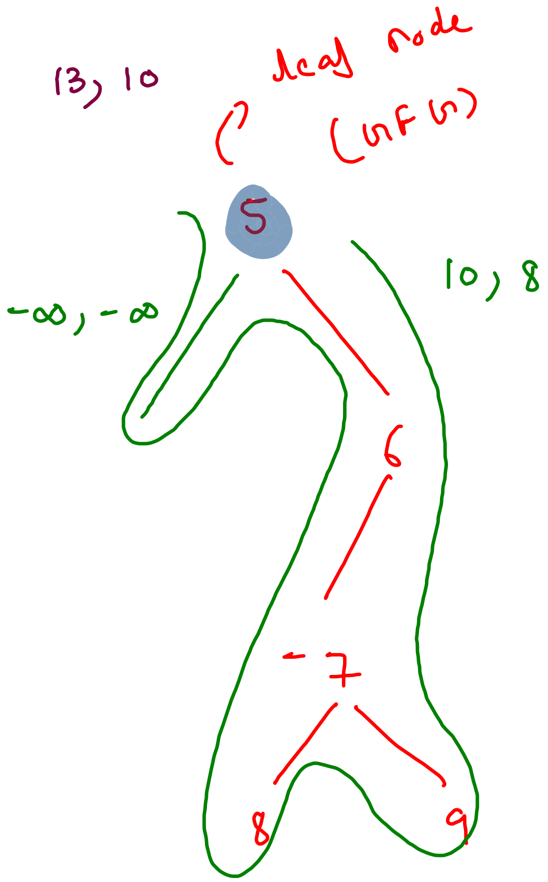


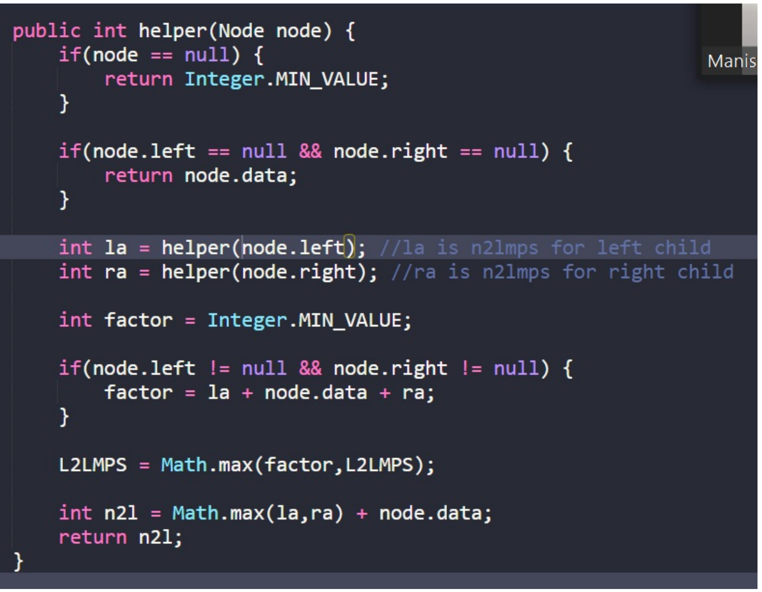
(l2lmps, n2lmps)



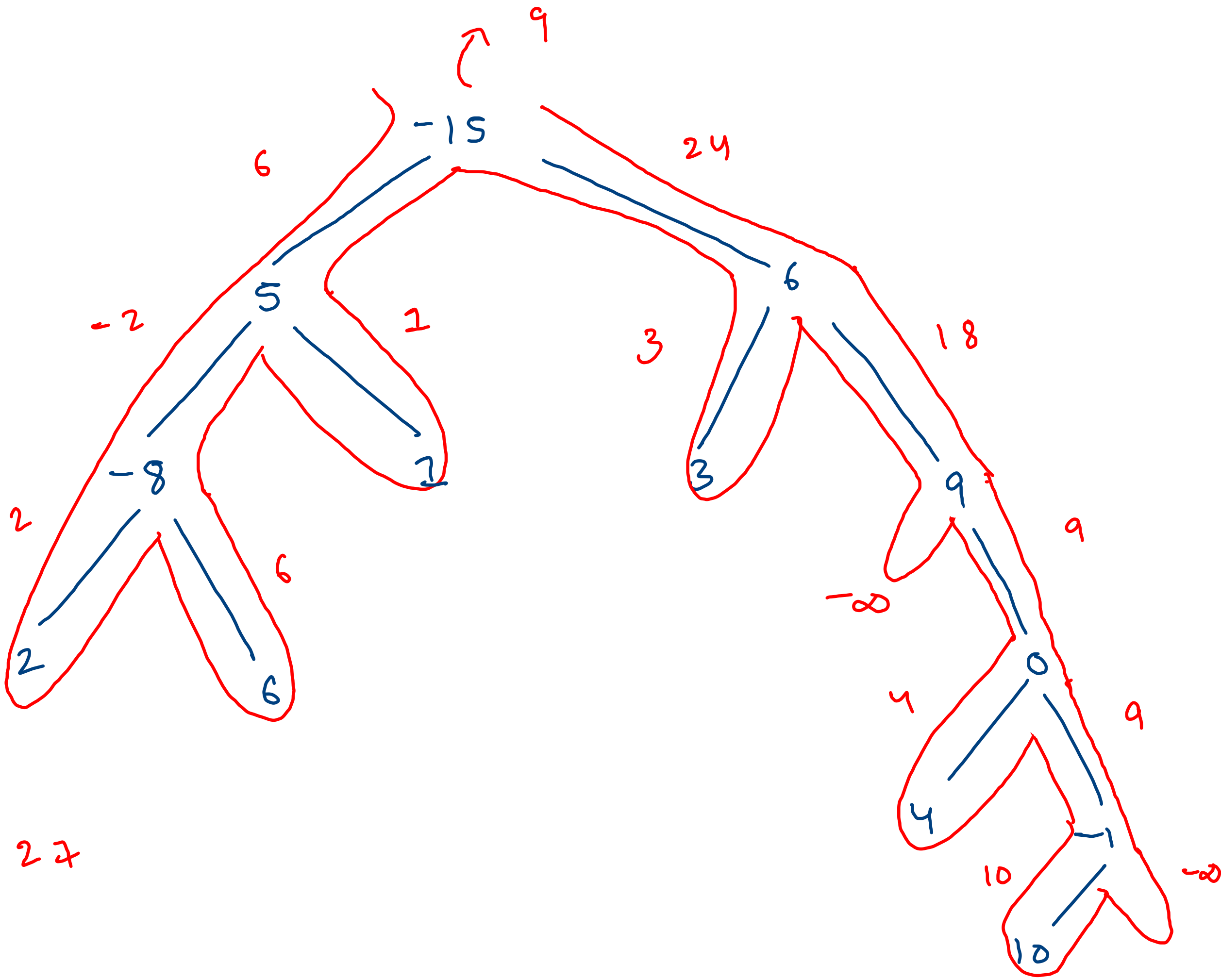
```
int maxPathSum(Node root)
{
    // code here

    Pair rp = helper(root);


    if((root.left != null && root.right == null) || (root.left == null && root.right != null)) {
        return Math.max(rp.l2lmps,rp.n2lmps);
    }

    return rp.l2lmps;

}
```

ans -> 13



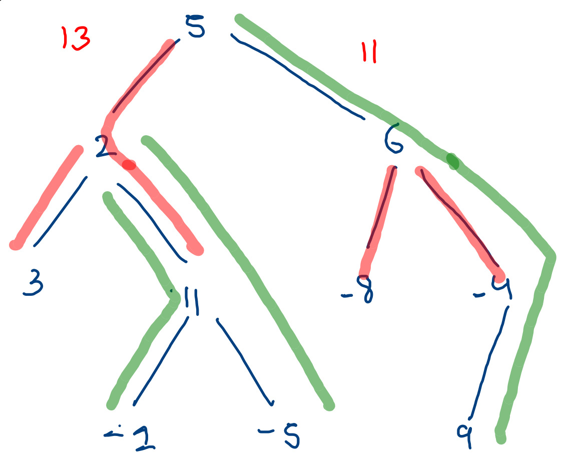


```java
public int helper(Node node) {
    if(node == null) {
        return Integer.MIN_VALUE;
    }

    if(node.left == null && node.right == null) {
        return node.data;
    }

    int la = helper(node.left); //la is n2lmps for left child
    int ra = helper(node.right); //ra is n2lmps for right child

    int factor = Integer.MIN_VALUE;

    if(node.left != null && node.right != null) {
        factor = la + node.data + ra;
    }

    L2LMPS = Math.max(factor,L2LMPS);

    int n2l = Math.max(la,ra) + node.data;
    return n2l;
}
```



n2l = ~~-∞~~ ~~0~~ ~~4~~ ~~13~~ 27

node to node
max path sum

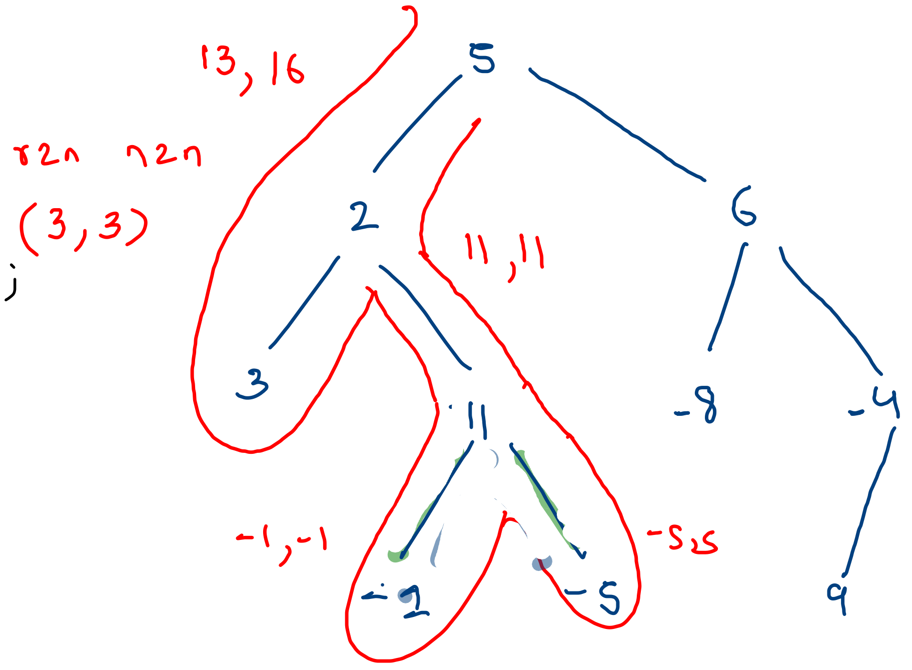


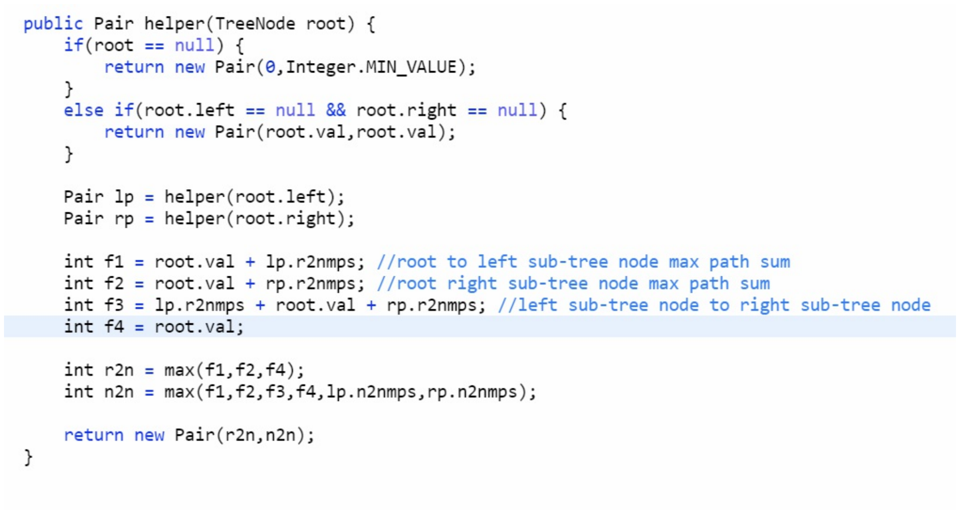
```
Pair {
    int n2nmps;
    int r2nmps;
}
```

dp, rp

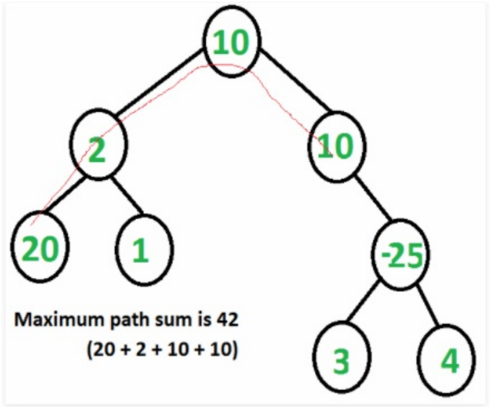
```
int j1 = dp.r2nmps + node.data;
int j2 = rp.r2nmps + node.data;
int j3 = dp.r2nmps + node.data + rp.r2nmps;
int j4 = node.data;

r2nmps = max(j1, j2, j4);

n2nmps = max(j1, j2, j3, j4, dp.n2nmps,
                    rp.n2nmps);
```

```
int j1 = lp.r2nmps + node.data;

int j2 = rp.r2nmps + node.data;

int j3 = lp.r2nmps + node.data + rp.r2nmps;

int j4 = node.data;


r2nmps = max(j1, j2, j4);


n2nmps = max( j1, j2, j3, j4, lp.n2nmps,
                         rp.n2nmps);
```



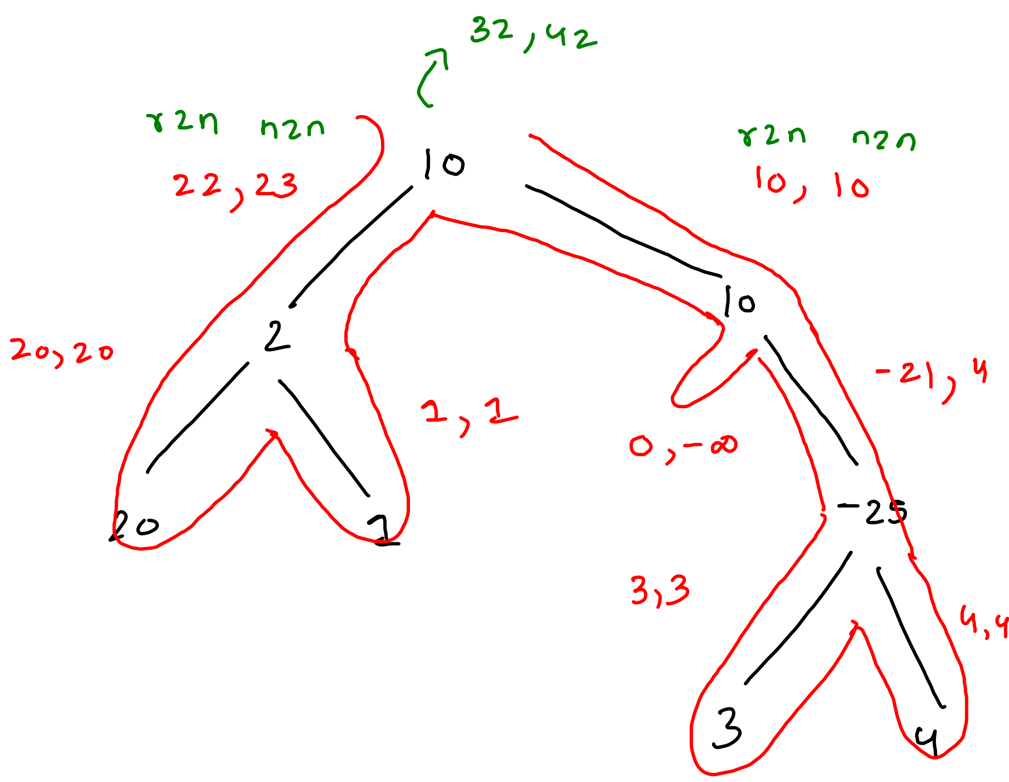


```
public Pair helper(TreeNode root) {
    if(root == null) {
        return new Pair(0,Integer.MIN_VALUE);
    }
    else if(root.left == null && root.right == null) {
        return new Pair(root.val,root.val);
    }

    Pair lp = helper(root.left);
    Pair rp = helper(root.right);

    int f1 = root.val + lp.r2nmps; //root to left sub-tree node max path sum
    int f2 = root.val + rp.r2nmps; //root right sub-tree node max path sum
    int f3 = lp.r2nmps + root.val + rp.r2nmps; //left sub-tree node to right sub-tree node
    int f4 = root.val;

    int r2n = max(f1,f2,f4);
    int n2n = max(f1,f2,f3,f4,lp.n2nmps,rp.n2nmps);

    return new Pair(r2n,n2n);
}
```



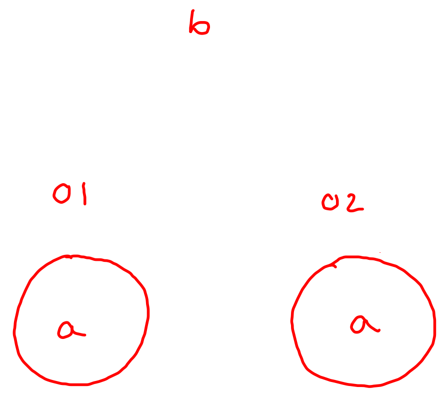
Maximum path sum is 42
(20 + 2 + 10 + 10)

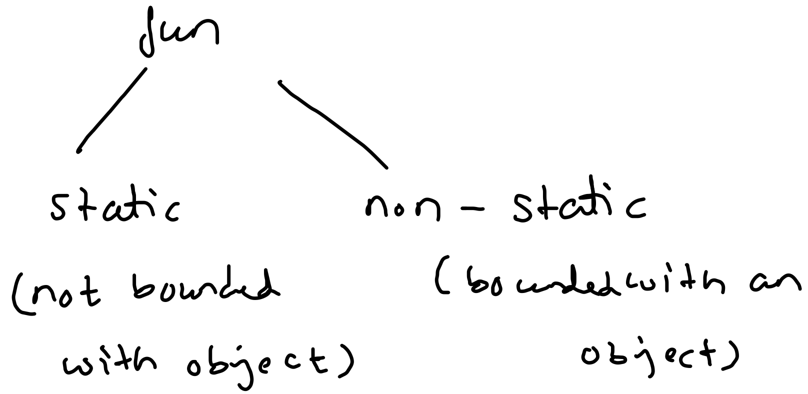
(r2n, n2n)



Static meaning

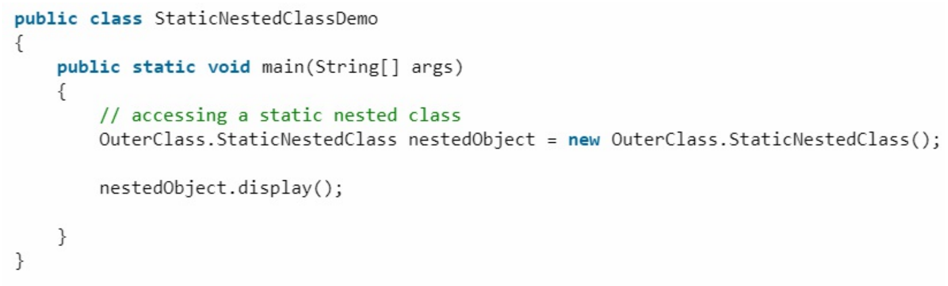
```
Class A {

data
member [ int a;

class    [ Static int b;
variable

}
main {
        A o1 = new A();
        A o2 = new A();
}
```





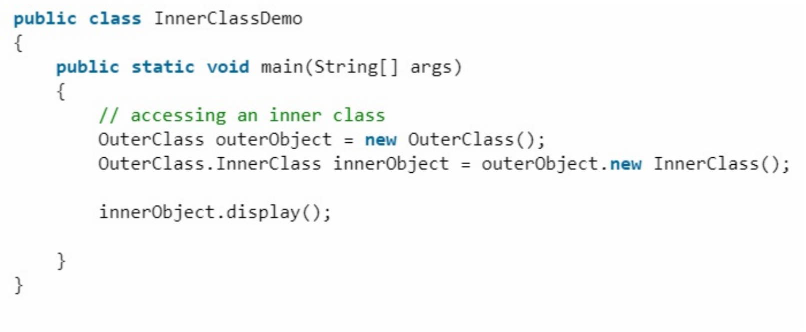
```
class     LL {
        Node  head;
        int  size;
        public  void  ns(){

        }
        public   static  fun2 ( ) {


        }
}

main  ( ) {
      LL  obj =  new  LL();
      obj. fun1( );
      LL. fun2( d1, d2 );
}
```

```
class   LL {
      class  Node {

      }
}
   new   Node( )
```



```java
public class StaticNestedClassDemo
{
    public static void main(String[] args)
    {
        // accessing a static nested class
        OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();

        nestedObject.display();

    }
}
```

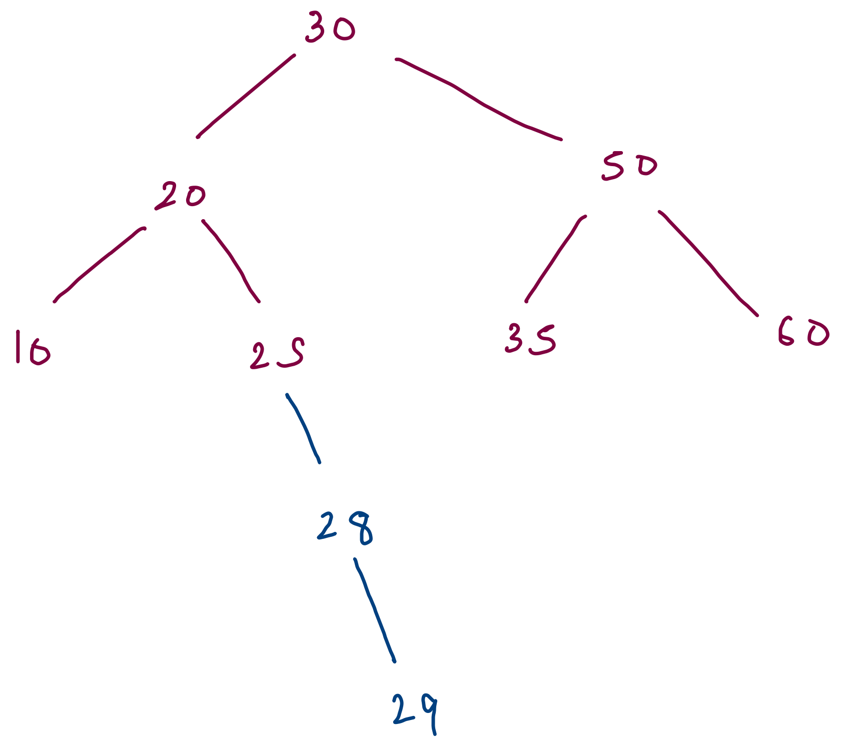
```
outer class {

        static   inner class {

        }

}
```



```java
public class InnerClassDemo
{
    public static void main(String[] args)
    {
        // accessing an inner class
        OuterClass outerObject = new OuterClass();
        OuterClass.InnerClass innerObject = outerObject.new InnerClass();

        innerObject.display();

    }
}
```

```
outer class {

        inner class {

        }

}
```

AVL



add(28)

add(29)