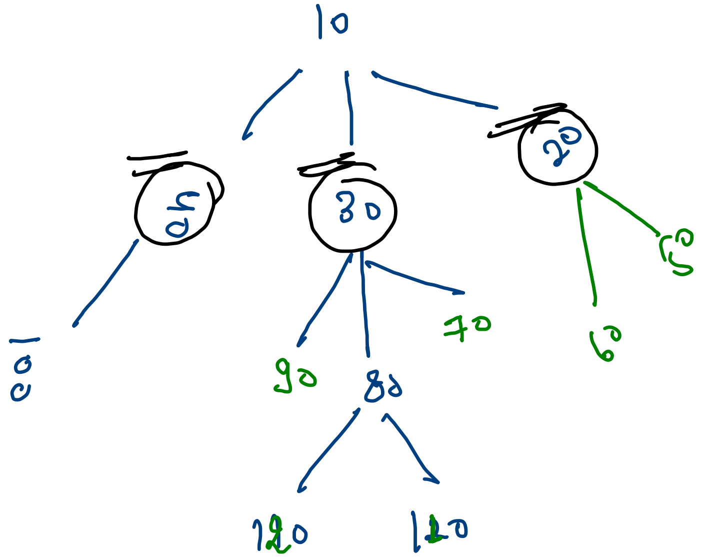
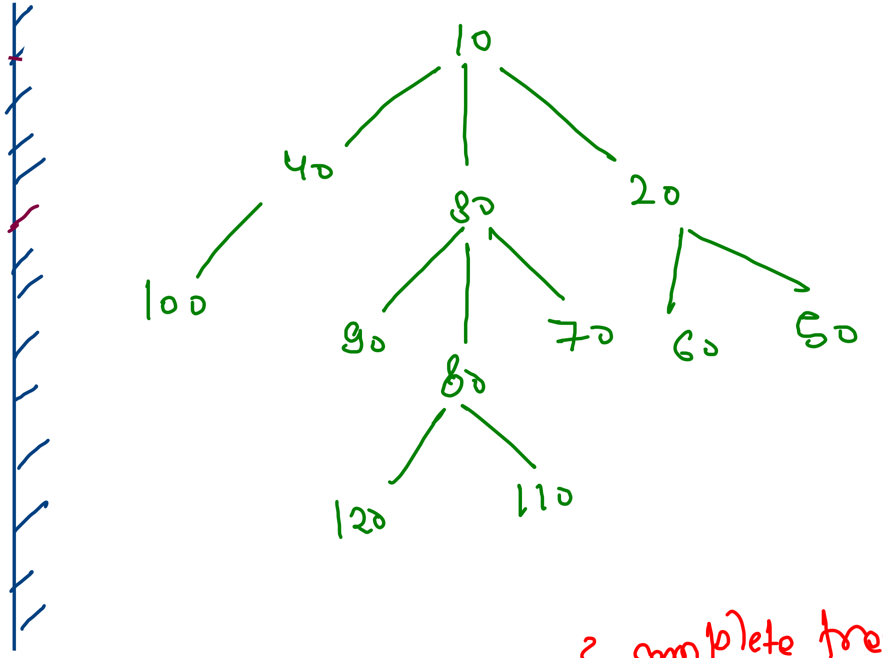
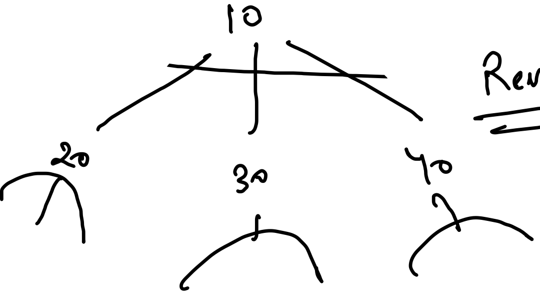
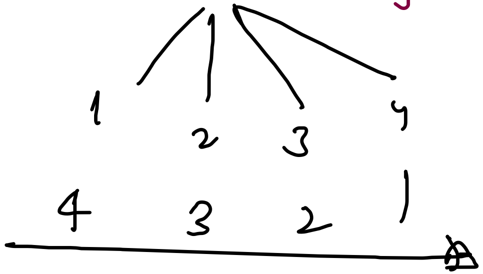
Mirror →







Reverses



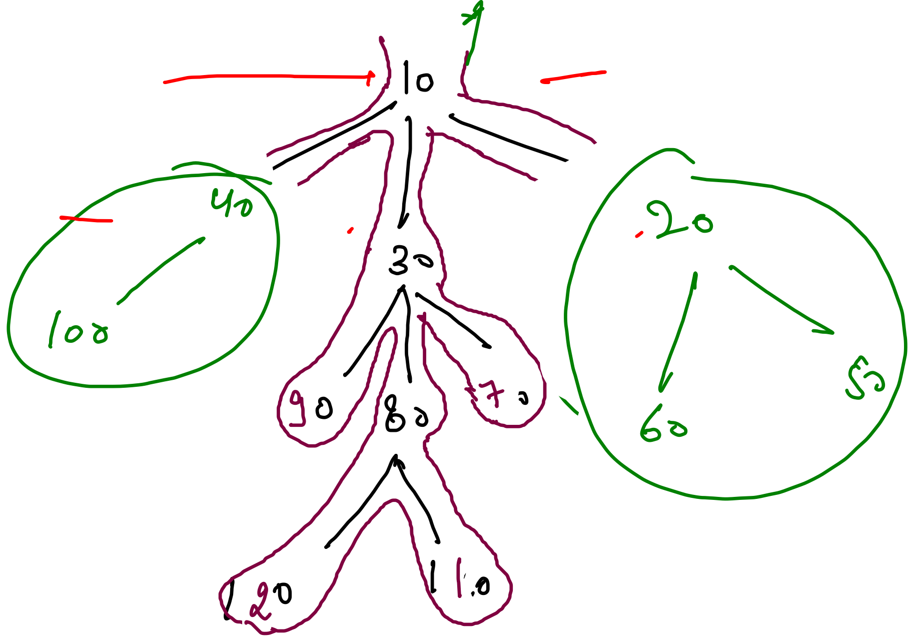
Expectation → mirror(10) } complete tree mirror

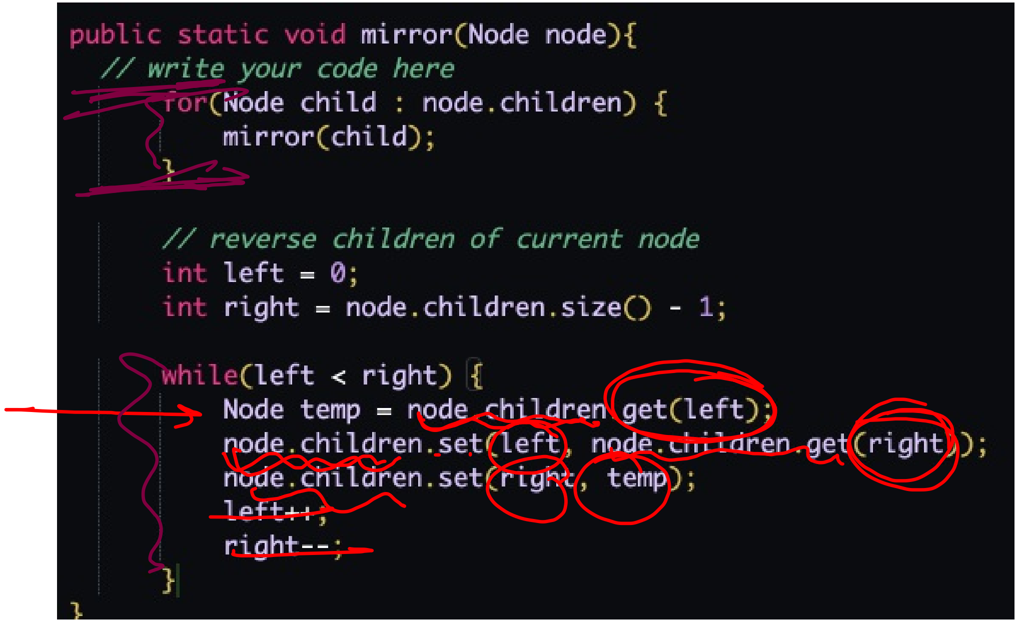
faith → * Chidren 10 → mirror(child) Done by recursion

mirror

* 10 child] → mirror

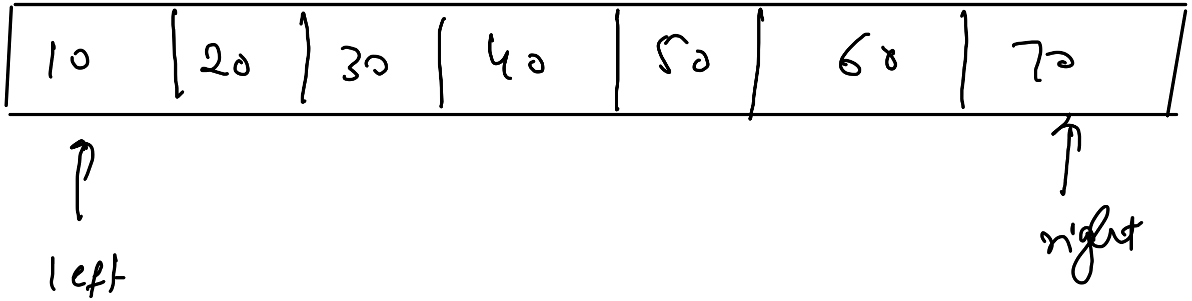
Done by me.



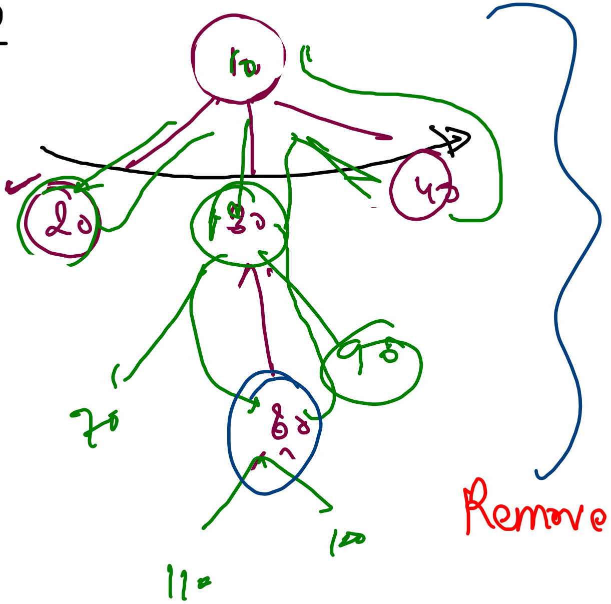


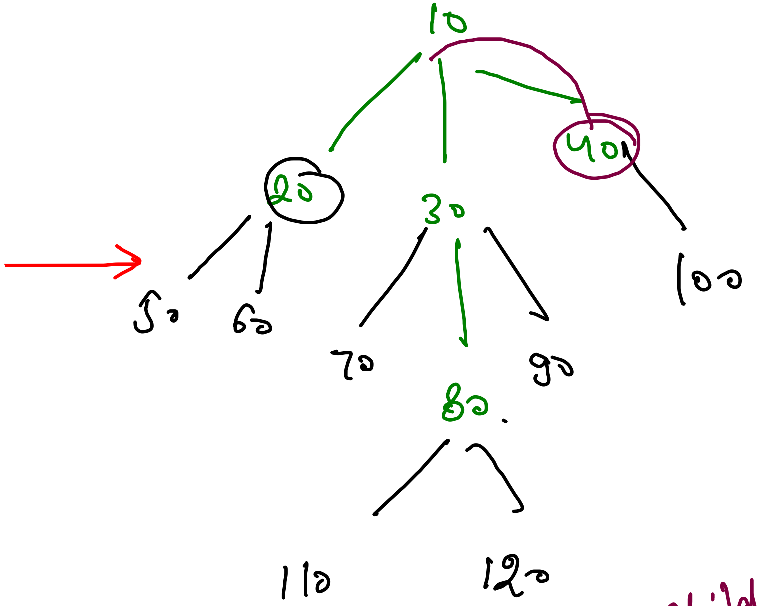
```
public static void mirror(Node node){
    // write your code here
    for(Node child : node.children) {
        mirror(child);
    }

    // reverse children of current node
    int left = 0;
    int right = node.children.size() - 1;

    while(left < right) {
        Node temp = node.children.get(left);
        node.children.set(left, node.children.get(right));
        node.children.set(right, temp);
        left++;
        right--;
    }
}
```

```
int temp = arr[left];
arr[left] = arr[right];
arr[right] = temp.
left ++;
right --;
```



Remove leaf





~~faith~~

~~self work~~

→

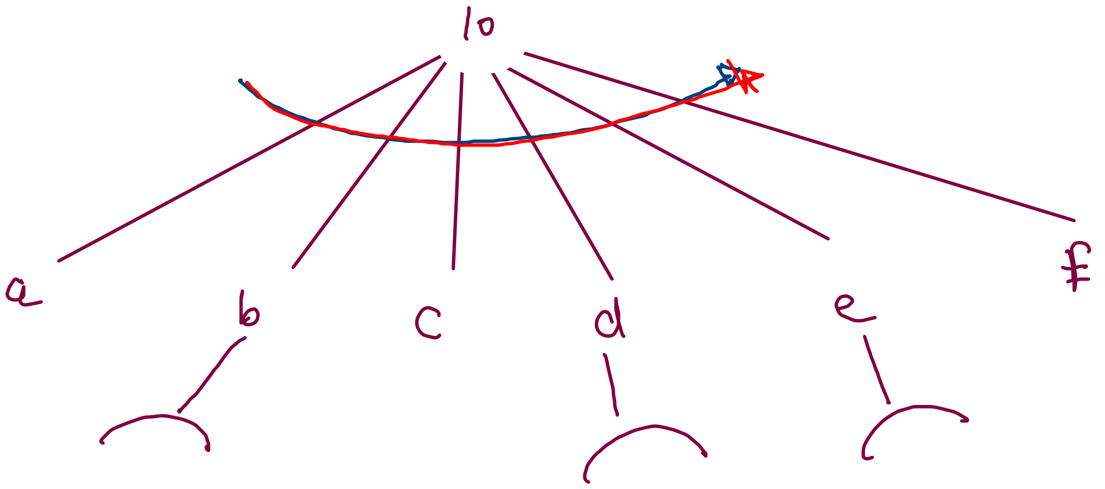
self work

faith

Pre Order

child.children.size()==0

child → leaf Node



iterator

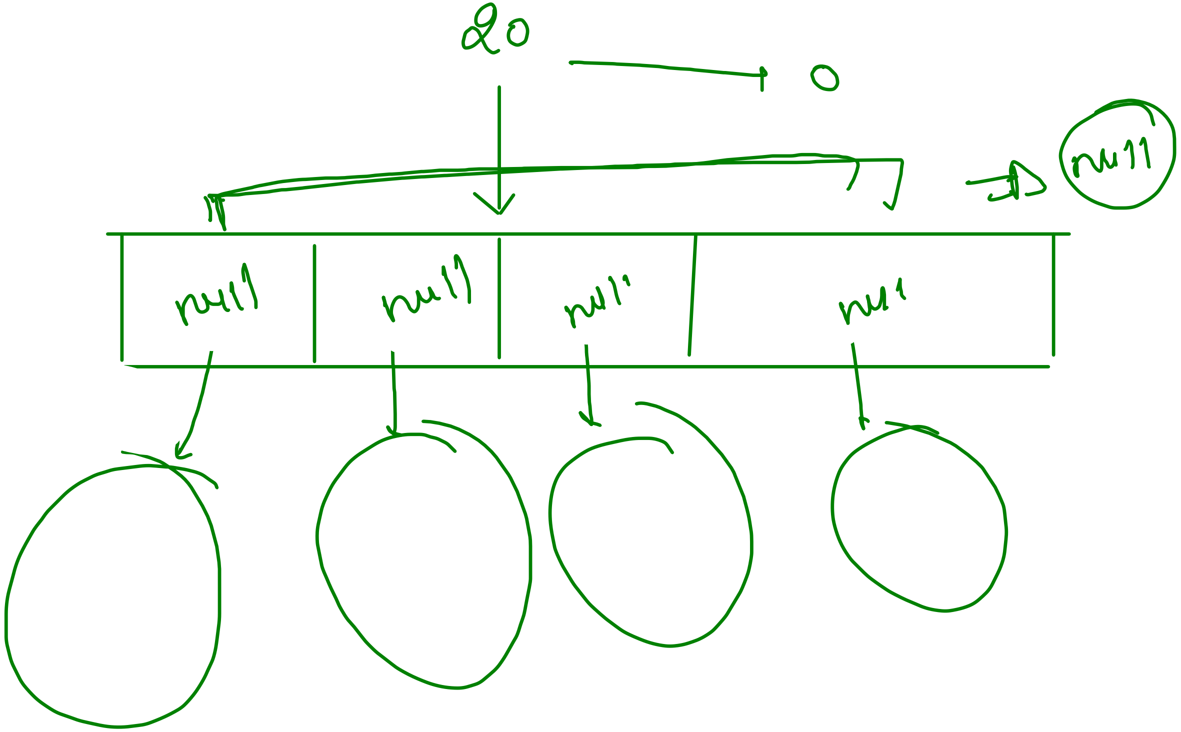
```
for( Node child: node.children) {
    if(child.children.size()==0){
        node.children.remove(child);
    }
}
```

remove - O(1)

10	20	30	40	50	60	

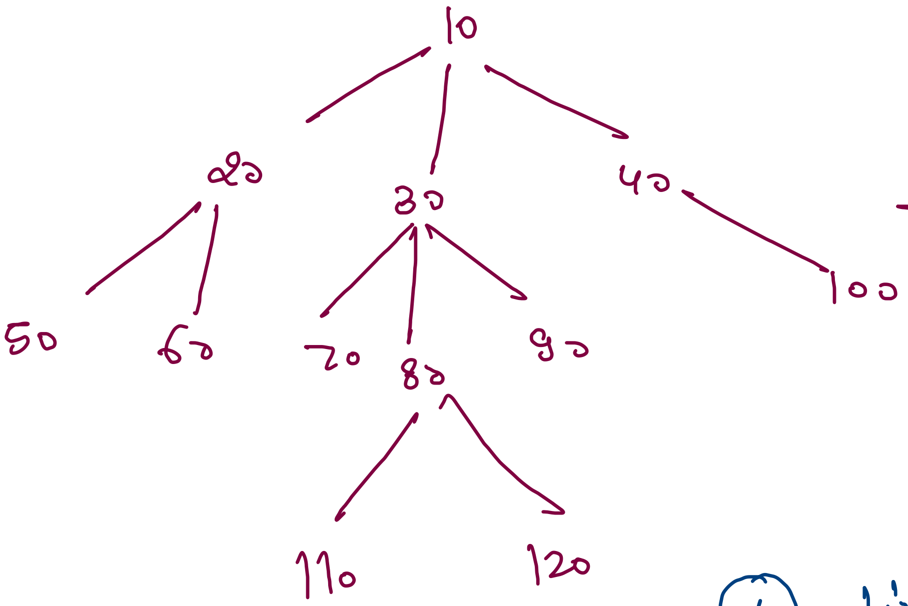
remove - O(n)

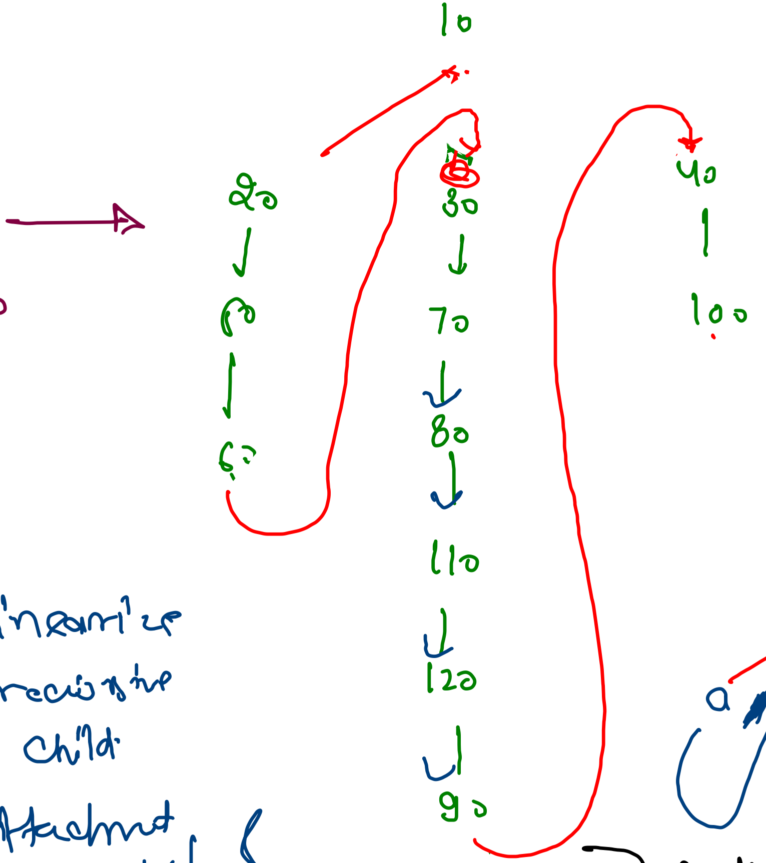
20 → nu, - - . . .

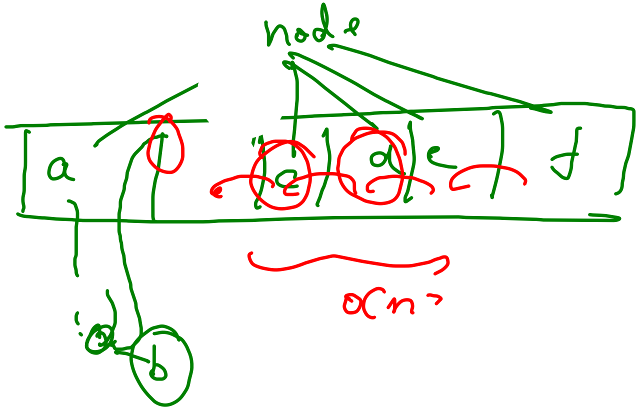


Linearize Generic Tree

Get Tail's



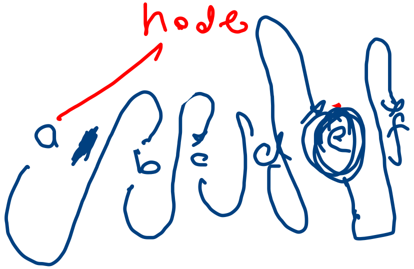


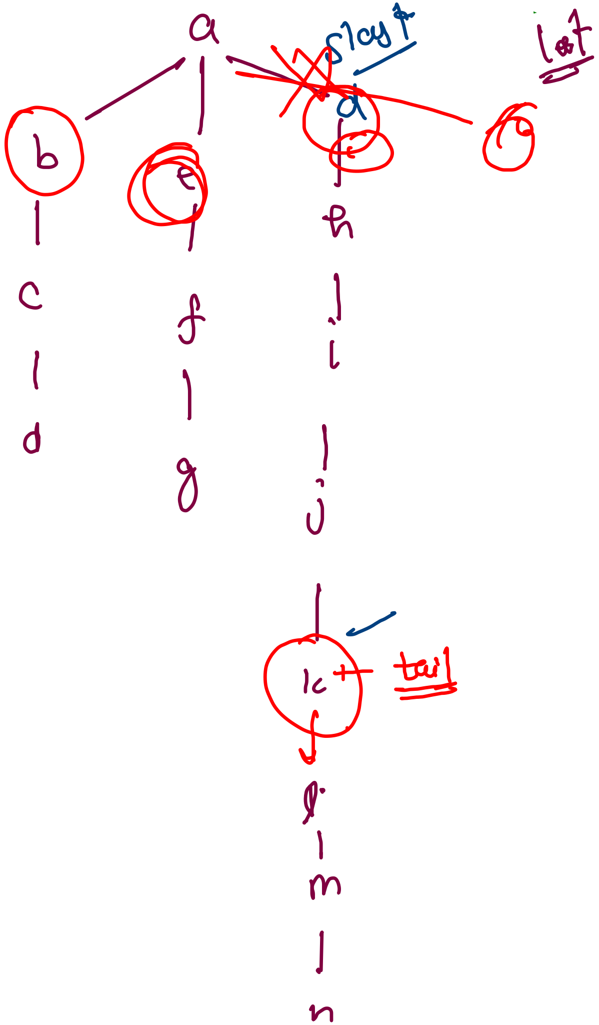


1. Linearize recursive child

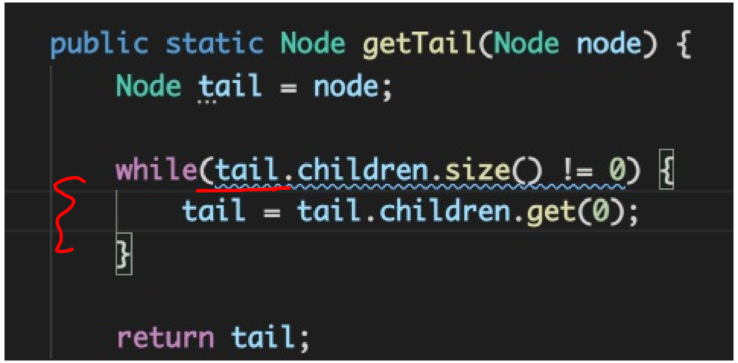
2. attachment of tail & head for linearisation

backward to first ?? Why

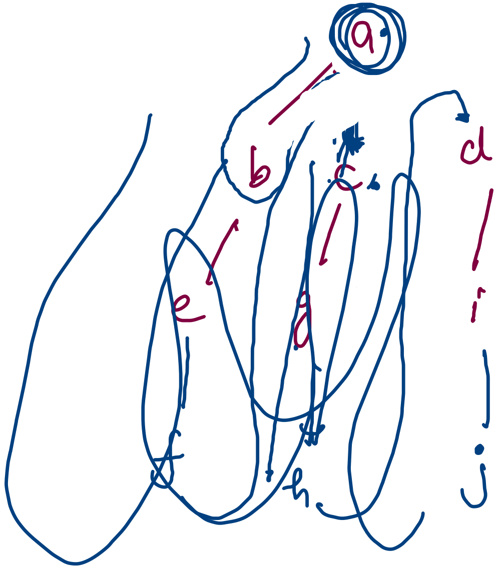


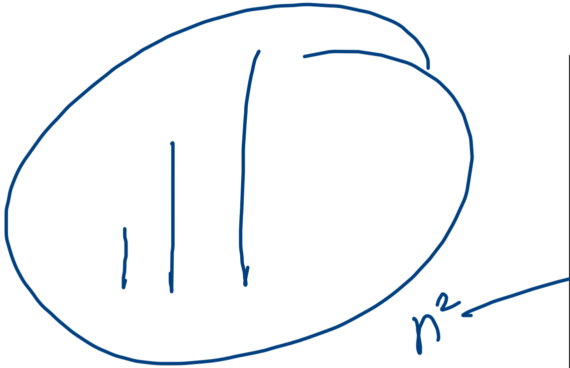


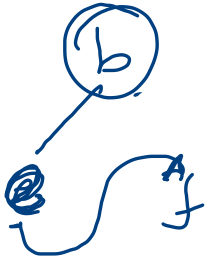
```
for( int i = arr.length - 2; i >= 0; i--) {
    Node rem = remove last Node
    get tail for last Node

    k. children.add(rem);

}
```

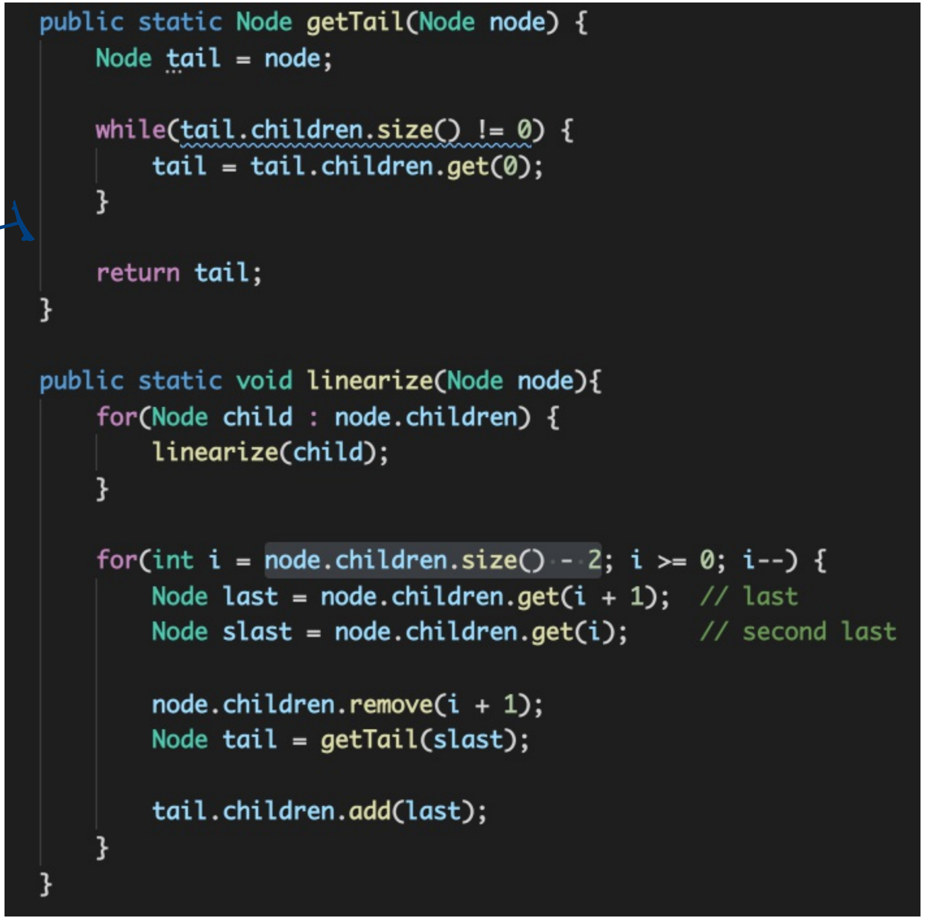


```java
public static Node getTail(Node node) {
    Node tail = node;

    while(tail.children.size() != 0) {
        tail = tail.children.get(0);
    }

    return tail;
```

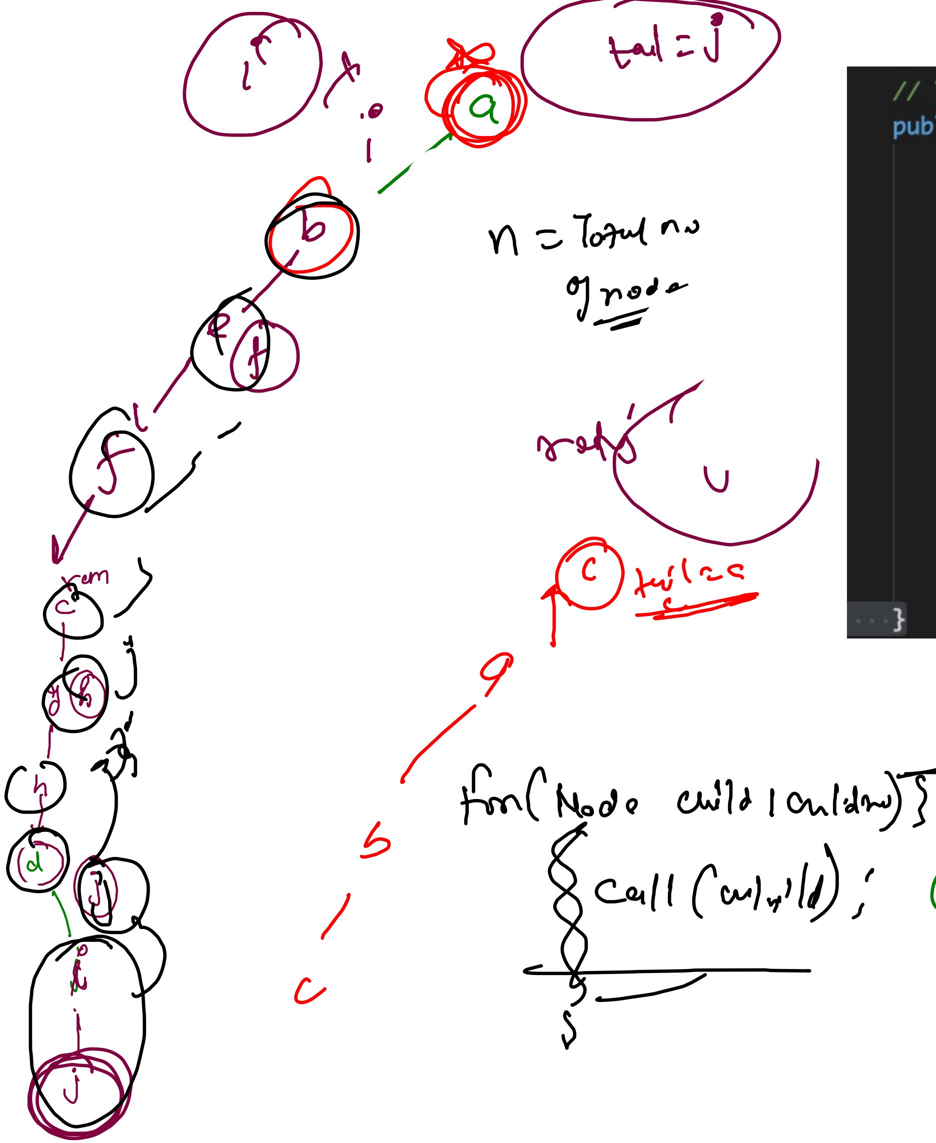


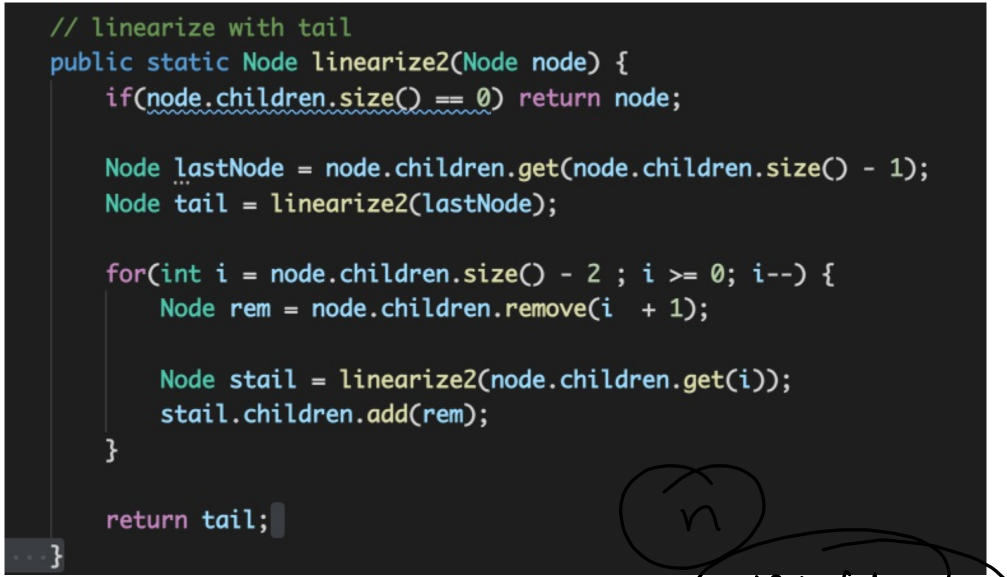




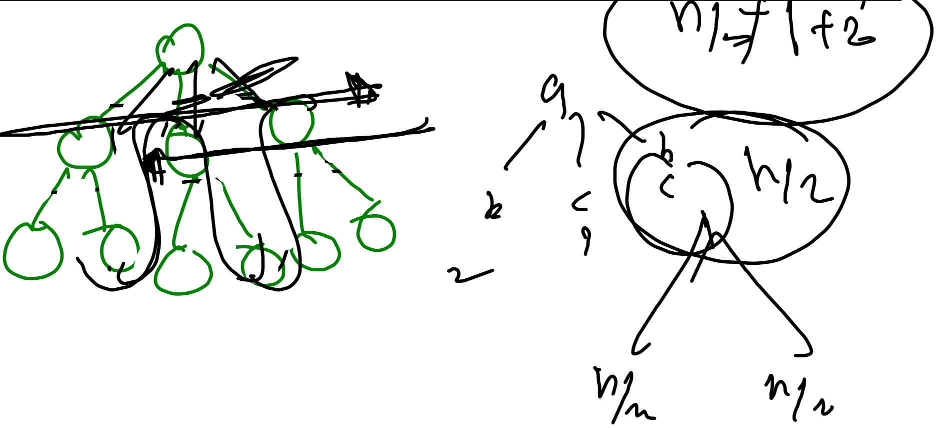


```
public static Node getTail(Node node) {
    Node tail = node;

    while(tail.children.size() != 0) {
        tail = tail.children.get(0);
    }

    return tail;
}

public static void linearize(Node node){
    for(Node child : node.children) {
        linearize(child);
    }

    for(int i = node.children.size() - 2; i >= 0; i--) {
        Node last = node.children.get(i + 1);  // last
        Node slast = node.children.get(i);     // second last

        node.children.remove(i + 1);
        Node tail = getTail(slast);

        tail.children.add(last);
    }
}
```





```java
// linearize with tail
public static Node linearize2(Node node) {
    if(node.children.size() == 0) return node;

    Node lastNode = node.children.get(node.children.size() - 1);
    Node tail = linearize2(lastNode);

    for(int i = node.children.size() - 2 ; i >= 0; i--) {
        Node rem = node.children.remove(i  + 1);

        Node stail = linearize2(node.children.get(i));
        stail.children.add(rem);
    }

    return tail;
}
```


n
n/2 + f2
n/2
n/2
n/2
a
b
c
2