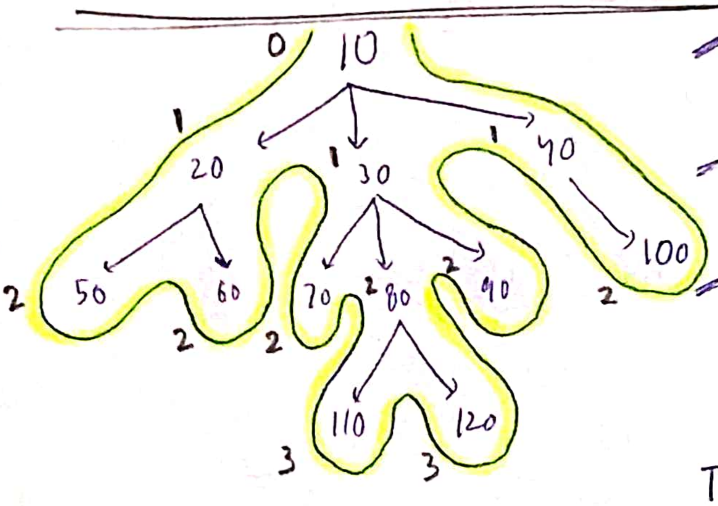
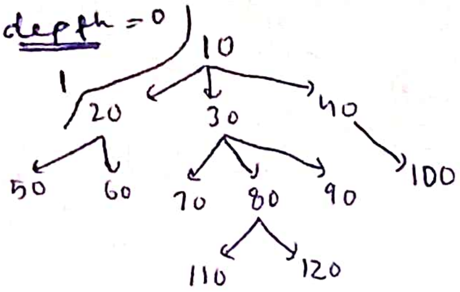
③ Multisolver For Generic Tree



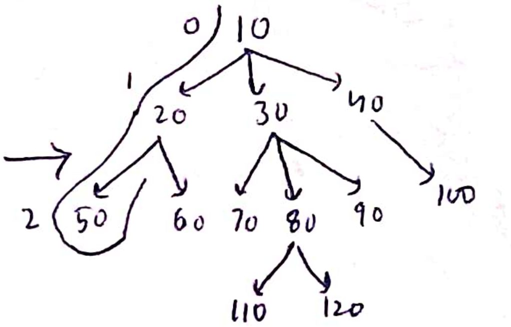
- Humne phele size, height, max, min Recursion se kiya tha!
- Aab hum Data Members use krengy aur will keep changing their values.
- That is we are going to traverse through a tree but not return anything!

TRAVERSE AND CHANGE RULE



Size = ~~0~~ ~~1~~ 2
min = ~~+∞~~ 10
max = ~~-∞~~ ~~10~~ 20
height = ~~0~~ 1

Heap



Size = ~~2~~ 3
min = 10
max = ~~20~~ 50
height = ~~1~~ 2

Eular path

```
public class Main {
    public static class Node {
      int data;
      ArrayList <Node> children = new ArrayList<>();
    }
    public static void display(Node node){
       String str = node.data + "->";
       for( Node child : node.children){
          str += child.data + ",";
       }
          str += ".";
          system.out.println(str);

       for( Node child: node.children) {
          display (child);
        }
    }
public static Node Construct (int [] arr) {
  Node root = null;

  Stack <Node> st = new Stack<>();
  for (int i = 0; i< arr.length; i++) {
     if (arr[i] == -1) {
         st.pop();
     } else {
       Node t = new Node();
       t.data = arr[i];
       if( st.size() >0) {
         st.peek().children.add(t);
       } else {
         root = t;
       }
       st.push(t);
     }
  }
  return root;
}
//*************// MULTISOLVER ***********//

static int size;   }
static int min;    }
static int max;    }--> heap                    Stack
static int height; }                              ↑
public static void multisolver(Node node, int depth){
  size ++;
  min = Math.min(min, node.data);
  max = Math.max (max, node.data);
  height = Math.max (height, depth);

  for(Node child: node.children){
   multisolver(child, depth+1);
  }
}
```

```
public static void main(String [] args){
BufferedReader br= new BufferReader(new InputStream
                         Reader(System.in));
int n = Integer.parseInt(br.readline());
int [] arr = new int [n];
String [] values = br.readLine().split(" ");
for(int i=0; i<n; i++){
  arr[i] = Integer.parseInt(values[i]);
}
Node root = construct(arr);
 size=0;
 max = Integer.MAX_VALUE;
 min = Integer.MN_VALUE;
 height = 0;
 multisolver(root,0);
 Syso("Size="+size);
 Syso("Max="+ max);
 Syso("Min="+ min);
 Syso("height"+ height);
 }
```