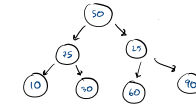
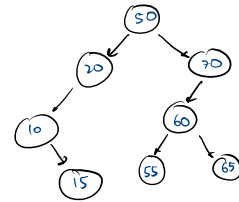
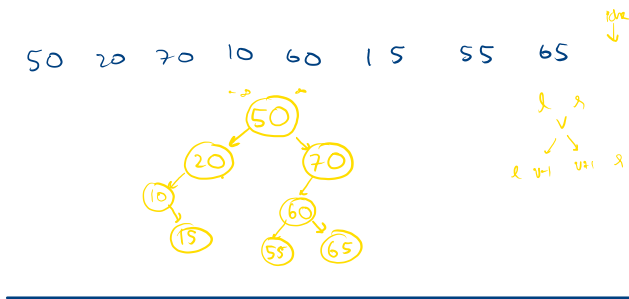


```

    parent
    pair = Node, l, r
    func(arr) {
    Q < Pair>
    root = new Node(arr[0])
    Q.push(P(root, -∞, r.val-1))
    Q.push(P(root, r.val+1, ∞))
    
```

```

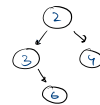
    idx = 1
    while(Q.size > 0) {
        Pair cur = Q.remove()
        if(idx == n) continue
        if(arr[idx] not in range) continue
        Node me = new Node(arr[idx])
        idx++
        if(me.val < cur.p.left) cur.p.left = me
        else cur.p.right = me
        Q.add(P(me, cur.l, me.v-1))
        Q.add(P(me, me.v+1, cur.r))
    }
    
```



```

    // Node constructor
    Node(int val) {
        this.val = val;
        this.left = null;
        this.right = null;
    }
    // Insert function
    void insert(Node n) {
        if(n.val < this.val)
            insertLeft(n);
        else
            insertRight(n);
    }
    // Insert left function
    void insertLeft(Node n) {
        if(this.left == null)
            this.left = n;
        else
            this.left.insert(n);
    }
    // Insert right function
    void insertRight(Node n) {
        if(this.right == null)
            this.right = n;
        else
            this.right.insert(n);
    }
    
```

③, 1
③, 2

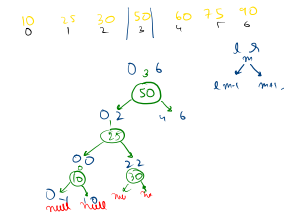
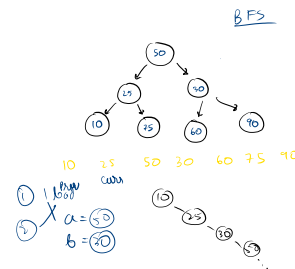
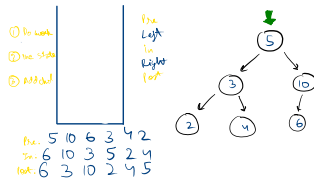


getNearestSmaller()

DFS
→ Preorder: N L R
→ Inorder: L N R
→ Postorder: L R N

```

    Stack<Pair> st;
    st.push(P(root, 1, 4))
    while(st.size > 0) {
        Pair top = st.peek();
        if(top.state == 1) {
            // Preorder
            // ++
            // left
        } else if(top.state == 2) {
            // Inorder
            // ++
            // right
        } else {
            // Postorder
            // POP
        }
    }
    
```



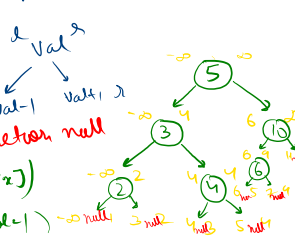
```

    func(arr, l, r) {
        if(l > r) return null;
        int mid = (l+r)/2;
        Node n = new Node(arr[mid]);
        n.left = func(arr, l, mid-1);
        n.right = func(arr, mid+1, r);
        return n;
    }
    
```

```

    idx = 0
    func(arr, l, r) {
        if(idx == n) return null
        if(l > r) return null
        if(arr[idx] < l || arr[idx] > r) return null
        Node me = new Node(arr[idx])
        idx++
        me.left = func(arr, l, me.val-1)
        me.right = func(arr, me.val+1, r)
    }
    
```

Pre: 5, 3, 2, 4, 10, 6



- 1) Val is not in range
- 2) Range is invalid
- 3) No more val left