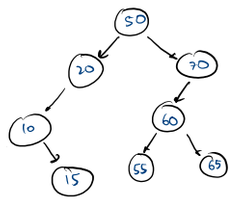
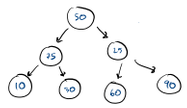
Pair = ~~Node~~ Pair, l, r

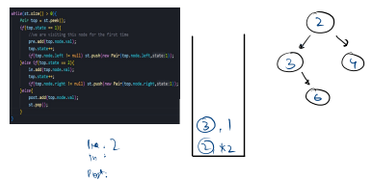
```
func (arr) {
  Q < Pair >
  Root = new Node (arr[0])
  Q.push ( P (Root, -∞, R.val-1) )
  Q.push ( P (Root, R.val+1, ∞) )
  idx = 1
  while ( q.size > 0 ) {
    Pair curr = q.remove()
    if ( idx == n ) continue
    if (arr[idx] not in range ) continue
    NN ne = nn(arr[idx]);
    idx++
    if( ne.val < curr.pair) curr.p.left = ne
    else curr.p.right = ne
    q.add ( P (ne, curr.l, ne.v-1) )
    q.add ( P( ne, ne.v+1, curr r) ).
  }
```

50 20 70 10 60 15 55 65

pre



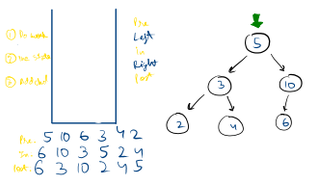




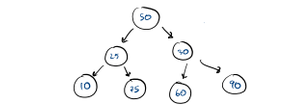
getNextInorder()

DFS


```
Stack <Pair> st;
st.push ( {root, 1} )
while(st.size > 0 ) {
  Pair top = st.peek();
  if(top.state == 1){
    pre
    ++
    left
  }else if( top.state == 2){
    in
    ++
    right
  }else{
    post
    POP
  }
}
```



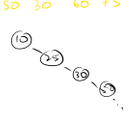
BFS

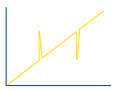


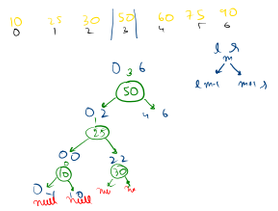
10 25 50 30 60 75 90

curr

1) Pop
2) a = 50
b = 20








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

```
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, m.val-1)
    me.right = func( arr, m.val+1, r)
}
```

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

idx

l Val r

Val-1 Val+1 r

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