Minimal Tree: Given a sorted array in ascending order with unique integer elements, write an algorithm to create a binary search tree with minimal height.

Example:

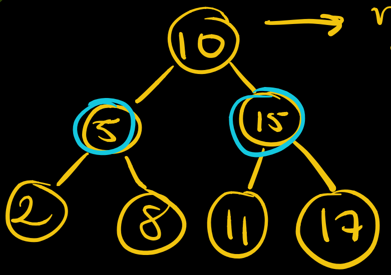
2	5	8	10	11	15	17
0	1	2	3	4	5	6

size = 7

$\log n = \lg(7) \approx 3$

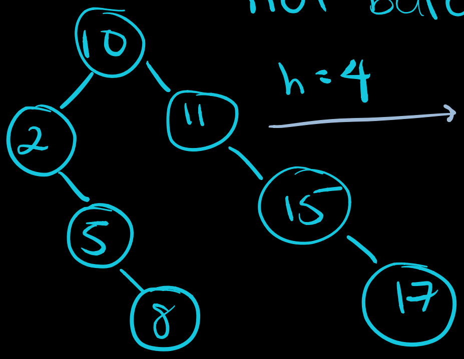
$h = 3$

Solution - end result:

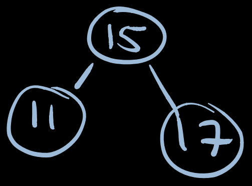


→ midpoint of the array became the root

Iteratively insert nodes — not balanced!



h = 4 → if this node has a right node, and that right node also has a right node (but null left node?) we can rebalance to

.

We should rebalance the root node's left node & right node, until both paths reach a null right node.

let's expand the array a bit

1	2	$\boxed{3}$	4	5	$\boxed{6}$	7	8	$\boxed{9}$	10	$\boxed{11}$	12
					↑						

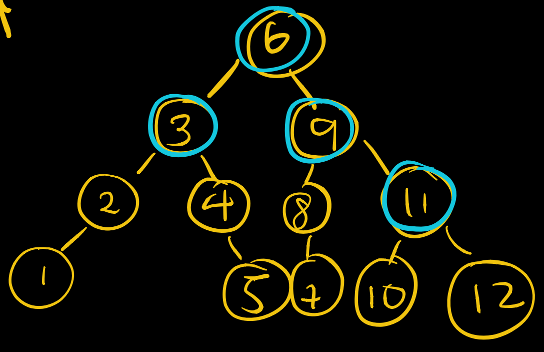
size = 13

$\log_2(13) = 3.7 \approx 4$

$h = 4$

$13/2 = 6$

3, 9, 11



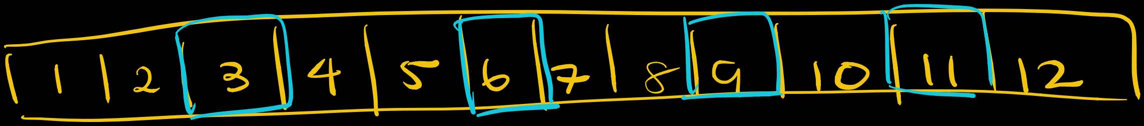
It looks like we can divide & conquer. We can take the midpoint of an array and -

- the first midpoint is the root node
- this divides the array into 2.
    - arr1 = elements < midpoint
    - arr2 = elements > midpoint
- keep dividing & tracking midpoints until the size of the arr <= 2.

Implementation:

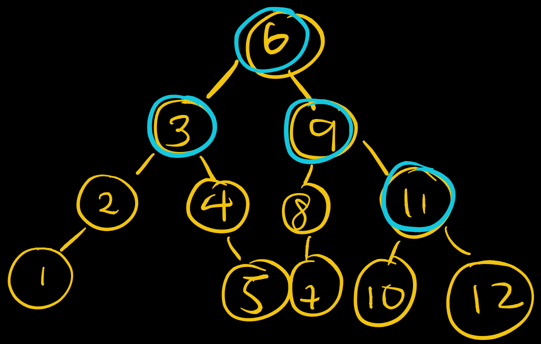
```
getMinimalTree (sortedArr){
    if(sortedArr.length === 0){ return null; }
    let midpoint = Math.floor(sortedArr.length/2);
    let parentNode = new TreeNode(sortedArr[midpoint]);
    let leftArr = sortedArr.splice(0, midpoint -1);
    parentNode.left = getMinimalTree(leftArr);

    let rightArr = sortedArr.splice(midpoint +1,
                        sortedArr.length -1);

    parentNode.right = getMinimalTree.(rightArr);
    return parentNode;
}
```

Illustrated Test:

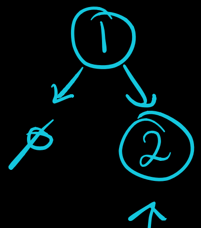


1) midpoint = 6
   parentNode = 6
   leftArr = [1 2 3 4 5]   rightArr = [7 8 9 10 11 12]

2) midpoint = 3
parentNode = $\textcircled{3}$
left Arr = $\boxed{1 \mid 2}$



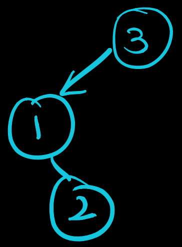
3) midpoint = 1
parentNode = $\textcircled{1}$
left Arr = ∅
return ∅ ⟶



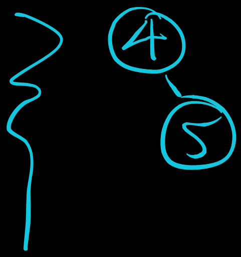
rightArr = 2

4) midpoint = 2
parentNode = $\textcircled{2}$
leftArr = ∅
rightArr = ∅
return $\textcircled{2}$

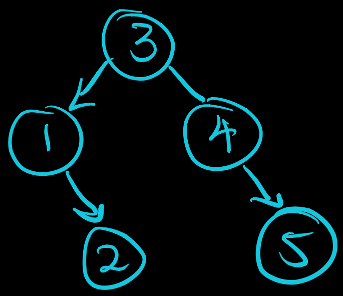
5) Go back to Step 2



rightArr = $\boxed{4 \mid 5}$
midpoint = 4
parentNode = $\textcircled{4}$
leftArr = ∅
right Arr = $\boxed{5}$



6)



7) go back to step 1 & repeat for the right array

... etc.