Part B Complexity Analysis Jacob Hathaway 11/2/22

```
Find ClosestManager(parent, head) {

if(Parent == null) {
    return
}
```

if we have the node that we are looking for and understand what the tree looks like then we will have to traverse the tree looking for the node, resulting in O(h) complexity

```
if(node != found) {

  Search left
  and return node if found
}
```

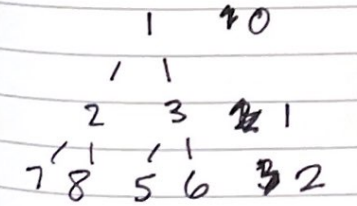
} based on height

```
if(node != found) {
  search the right side
  and return if found
}
```

} based on height

return null; if node is not found then return null after searching the whole tree.

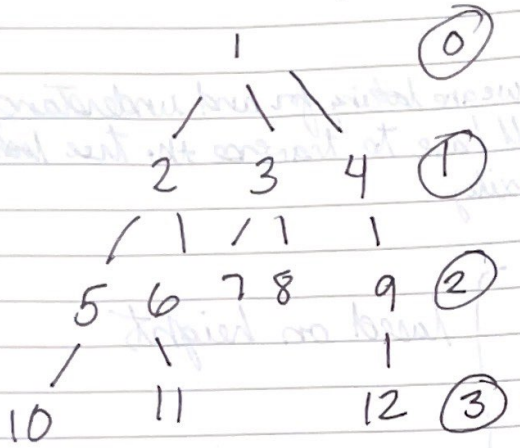
For worst case it understands the node it is looking for so it begins to search the tree and can't find it, resulting in O(h) → height.



Code starts at root making its way down the left then right stopping at the node it is looking for.

in order for the tree to be O(h) we need a BST which can traverse without recursion

Tree example



Starts here and if it finds it will stop

↓ otherwise it will move down

↓ Continues down and h increases

↓ Gets to bottom and if not found it will give you O(h) time.