Assignment 1 of CISC 3000

ZHANG HUAKANG

February 14, 2022

1

- $T(n) = 3n^2 + 5n \log_2 n = O(n)$. False
- $T(n) = 4^{\log_2 n} + \sqrt{n} = \Omega(n^2)$. True
- $T(n) = 3n^2 + 9n = O(n^3)$. True
- $T(n) = 4(\log_2 n)^5 + 5\sqrt{n} + 10 = \Theta(\sqrt{n})$. False
- $T(n) = (\log_2 n)^{\log_2 n} + n^4 = \Theta(n^4)$. True

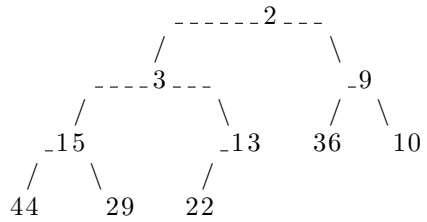
2

Algorithm 1: Sum of three(A, K)

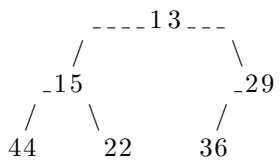
let $n \leftarrow |A|$ and assume $A = \{a_1, a_2, ..., a_n\}$.
for $i == 1, 2, ..., n-2$ **do**
    $j \leftarrow i + 1$ and $k = n$.
    while $k > j$ **do**
        if $a_i + a_j + a_k = K$ **then**
            Output: (i, j, k)
        else if $a_i + a_j + a_k < K$ **then**
            $j \leftarrow j + 1$
        else if $a_i + a_j + a_k > K$ **then**
            $k \leftarrow k - 1$
Output: do not exist

3

3.1

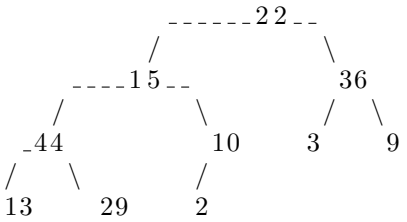


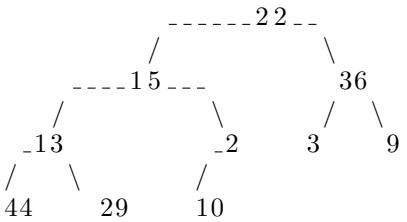
3.2

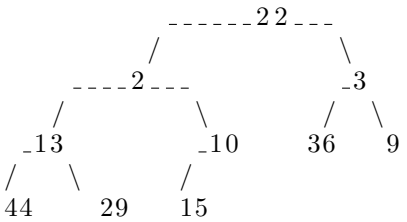


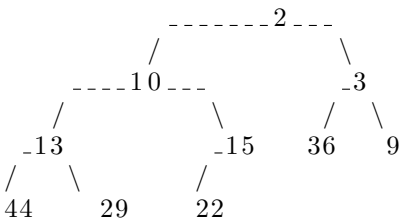
4

4.1

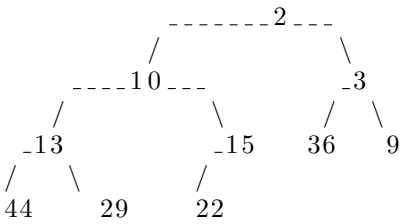

22
15
36
44
10
3
9
13
29
2

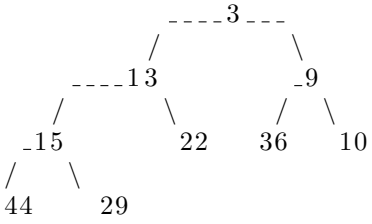

22
15
36
13
2
3
9
44
29
10

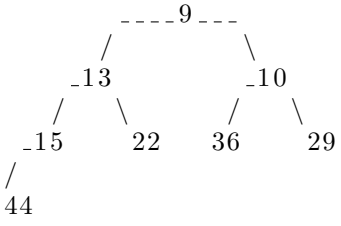

22
2
3
13
10
36
9
44
29
15

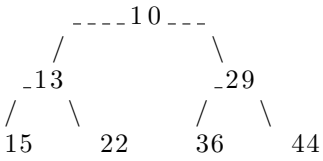

2
10
3
13
15
36
9
44
29
22

4.2


2
10
3
13
15
36
9
44
29
22

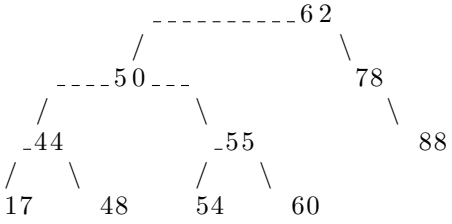

3
13
9
15
22
36
10
44
29


9
13
10
15
22
36
29
44

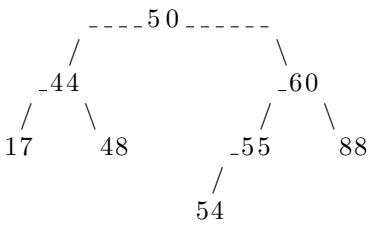

10
13
29
15
22
36
44

5

5.1


62
50
78
44
55
88
17
48
54
60

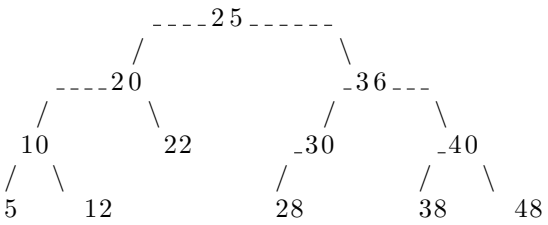
5.2



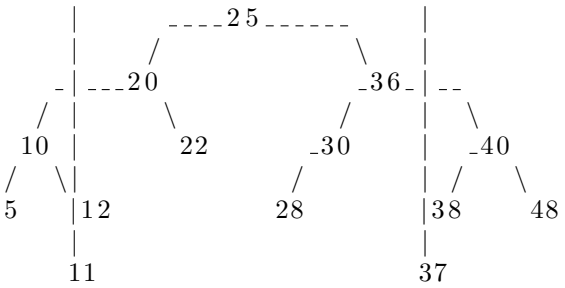
6

```
def findAll(k1:int,k2:int):
    return _findAll(k1,k2,tree.root)
def _findAll(k1:int,k2:int,node:Node):
    if node is None:
        return None
    if node.val<k1:
        if node.right is None:
            return None
        else:
            return _findAll(k1,k2,node.right)
    elif node.val>k2:
        if node.left in None:
            return None
        else:
            return _findAll(k1,k2,node.left)
    else:
      return {
          _findAll(k1,k2,node.left),
          node.key,
          _findAll(k1,k2,node.right)
      }
```

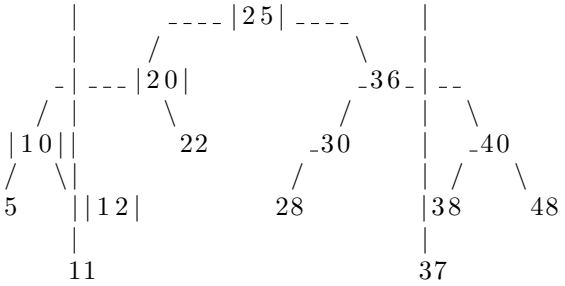
Suppose we have a BST:



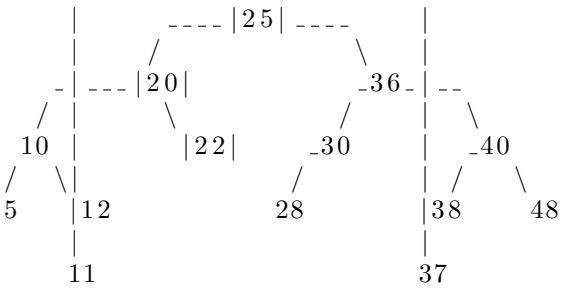
Let $k_1 = 11$ and $k_2 = 37$



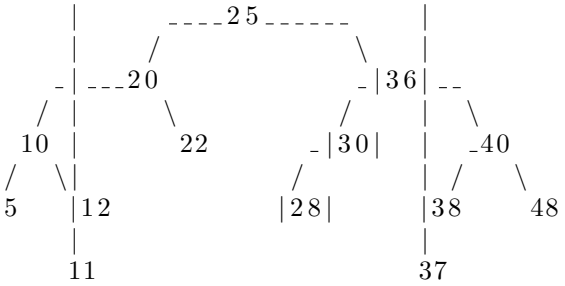
We will check node 25, 20, 10 and 12 at first. And add 12 to the result.



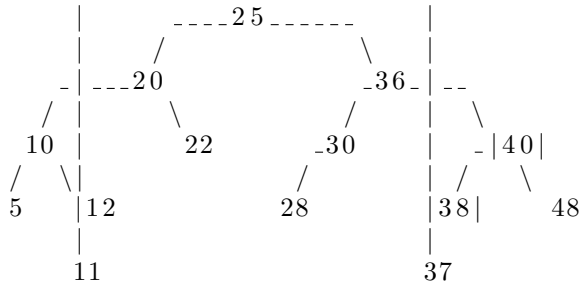
Check node 22 and add 20, 22 and 25 to the result.



Check node 36 30 and 28 and add 28 30 and 36 to the result.



Check node 40 and 38 and end the process.



Proof. The first thing we need to do is find the leftmost node and rightmost in BST that they are in the range $[k_1, k_2]$. The worst case is that all node in BST is in the range. So find each node we need $O(\log n)$ and two nodes is $2O(\log n)$.

The second step us add all node between leftmost node and rightmost node into result. If we have s node, we will need $O(s)$ to do this.

Thus, the total complexity of this algorithm is

$$2O(\log n) + s = O(\log n + s)$$

□