Assignment 4

2) Binary search trees

a)

//write a recursion function according to this!

For a binary search tree that contains n nodes with keys 1,2,3…n, the minimal height is log(n). Try to make every level in the tree except the last level is full.

So insert the keys into the binary search tree in this way:

For the level 1 (root): n/2

For the level 2 (from left to right): n/4 3n/4

For the level 3 (from left to right): n/8 3n/8 5n/8 7n/8

For the level n (from left to right):

(2-1)/(2^n) (2^2-1)/(2^n) (2^3-1)/(2^n) …… (2^n-1)/(2^n)

b)

Find:

Start from the root, choose to go either left or right and repeat this until the key is found. In the worst case is failing to find the key until we go down to the bottom of the tree. Since there is log(n) levels, we check log(n) nodes, thus the running time is: O(log(n))

Insert:

Find the leaf where the insertion will take place by going down the tree as for the "find" algo. In the worst case, we go down to the bottom of the tree, checking log(n) nodes, thus the running time is: O(log(n))

Remove:

Find the node N to be removed using the "find" algo. In the worst case, N has two children, we need to follow the right child of N and then go down left children until no left child is found to get N'. Overwrite N by N'. In the worst case, we need to go the bottom of the tree, thus the running time is: O(log(n))

c)

Insert in the way that makes the binary search tree somewhat likes a linked list. We starts with the smallest key, let it be the root, and go down insert the other keys with increasing value one by one as right children. (There is no left children) The maximal height is n.

d)

Find:

In the worst case, we go down to the bottom of the tree of n levels, therefore the running time O(n).

Insert:

O(n)

Remove:

O(n)

3) Finding the k-th elemtn

a)

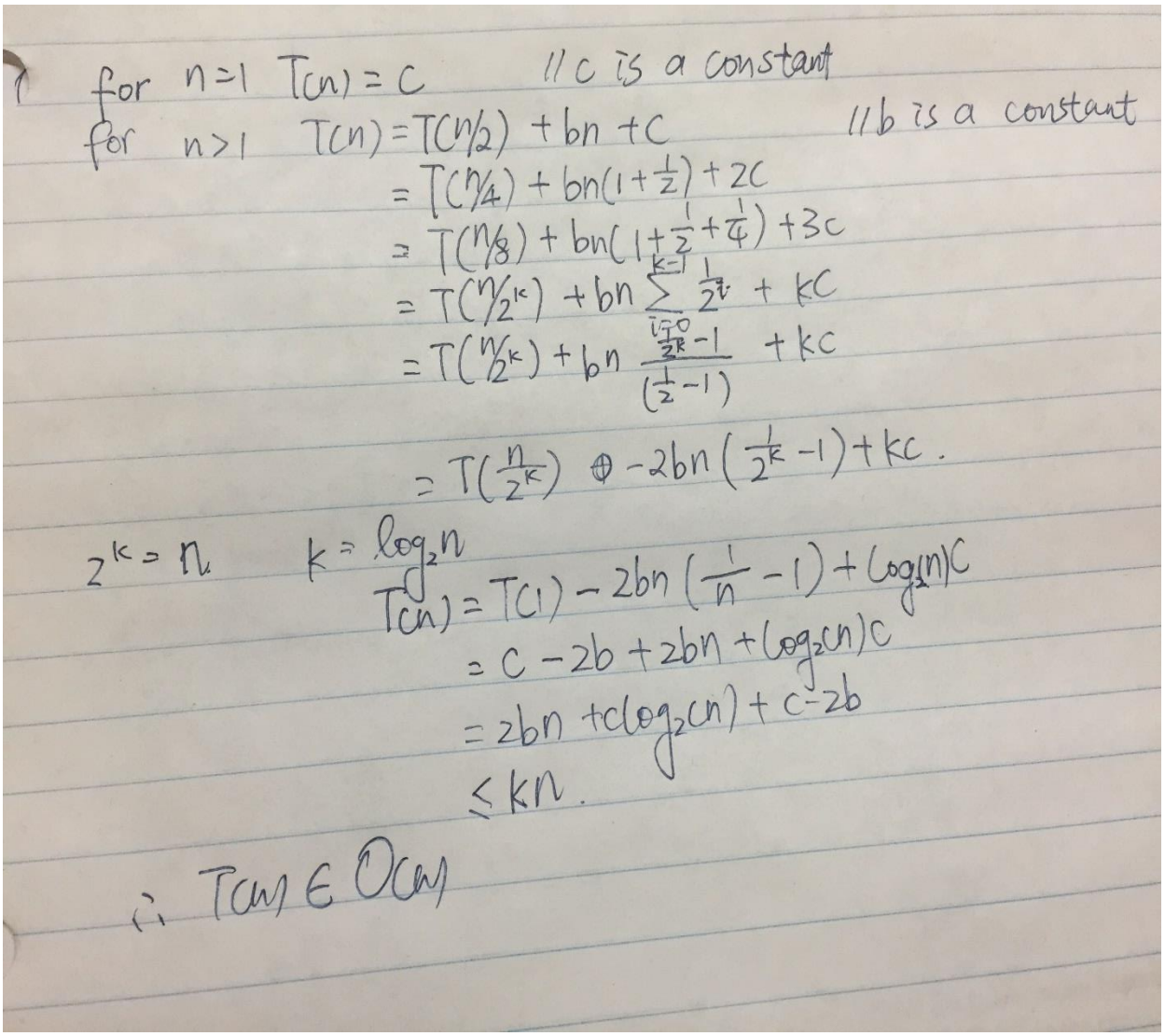
Algorithm findKth(A, start, stop, k)

Input: An unsorted array A[start...stop] of numbers, and a number k between 0 and stopstart-1

Output: Returns the k-th smallest element of A[start...stop]

```
If(k>(stop-start)){
    print “k should be less than (stop – start);
    return null
}
If(start < stop) {
    pivot ← partition(A, start, stop)
    if (pivot == k) return A[k]
    if (pivot > k) return findKth(A, start, pivot-1, k)
    if (pivot < k) return findKth(A, pivot+1, stop, k)
}
If(start == stop) return A[start]
```

b) see picture in next page



4.

a)

7 9 7 8 9 3 6 2 2 1

b)

I am not sure about the question since part of it is covered by the tree.

If the question is “Write the output being printed when weirdPostOrder(root) is executed on the following binary tree:”, then:

9 9 8 7 6 2 1 2 3 7

c)

7 3 9 2 6 8 9 1 2 7

Breath-first search of the tree.

d)

7 9 9 8 7 3 6 2 2 1

Depth-first search of the tree

5.

Algorithm isIsomorphic(treeNode A, treeNode B)

Input: Two treeNodes A and B

Output: Returns true if the trees rooted at A and B are isomorphic

```
If (A==null && B==null) return true
If (A==null && B!=null || A!=null &&B==null) return false
If (A.getValue()!=B.getValue()) return false
else{
   LL=isLsomorphic(A.getLeftChild(), B.getLeftChild())
   RR= isLsomorphic(A.getRightChild(), B.getRightChild())
   LR= isLsomorphic(A.getLeftChild(),B.getRightChild())
   RL= isLsomorphic(A.getRightChild(),B.getLeftChild())
}
```