Jakob Varmose Bentzen
Xu Zhen Yang

Studypoint exercise - collection & algorithms

Problem 1

Den øverste løkke afhænger af (pass, n) som kører **n** gange, hvor dér nederste løkke afhænger af (index,n) som kører **n** gange, og ind i den er der (count =1) en **konstant**, som er ligegyldigt.
Så svaret er: O(n*n*1), **$O(n^2)$**

Problem 2

Code:

Jakob

```
 public static void freq(int[] arr) {
    Arrays.sort(arr);
    int val = 0, count = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == val) {
            count++;
        } else {
            val = arr[i];
            count = 1;
        }
        if (i + 1 == arr.length || arr[i+1] != arr[i]) {
            System.out.println(val + "\t" + count);
        }
    }
}
```

Xu

```
public void freq(int[] arr)
    {
        //Write a method
        //public void freq(int[] arr)
        //int[] elementer =
{2,5,2,9,7,1,100,2,3,5,77,9,1,2,6,5};
        HashMap<Integer, Integer> hm = new HashMap<>();
        for (int e : arr)
        {
            if(hm.containsKey(e))
                hm.put(e, hm.get(e)+1);
            else
                hm.put(e,1);
        }
        //sæt elementer i hm
        ArrayList<Integer> l = new
ArrayList<>(hm.keySet());
        Collections.sort(l);
        for (int i = 0; i < l.size(); i++)
        {
```

Jakob Varmose Bentzen
Xu Zhen Yang

```
            System.out.println("Key - Count " + l.get(i) +
" " + hm.get(l.get(i)));

        }
  }
```

Testing:

```
Input:
[2, 5, 2, 9, 7, 1, 100, 2, 3, 5, 77, 9, 1, 2, 6, 5]
Result:
1          2
2          4
3          1
5          3
6          1
7          1
9          2
77         1
100        1

Input:
[0, 0, 7, 0]
Result:
0          3
7          1

Input:
[]
Result:

Input:
[-10, -20, -10, -30]
Result:
-30        1
-20        1
-10        2
```

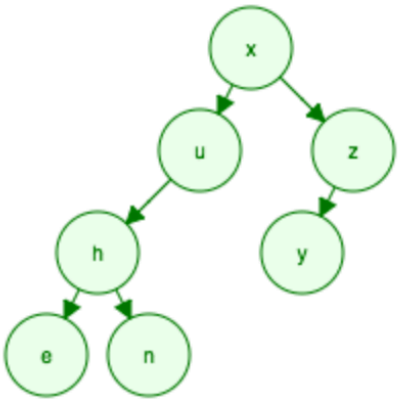
Problem 3

Jeg vil bruge HashMap funktion, fordi det er konstanttid med insert og lookup.
I andre tilfælde kan man overveje med TreeMap, på grund af den er fint sorteret.

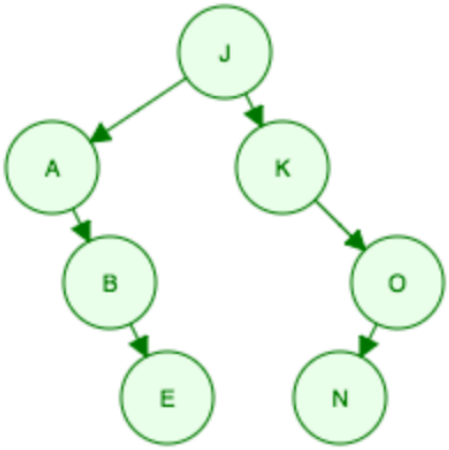
Problem 4

a)

Jakob Varmose Bentzen
Xu Zhen Yang

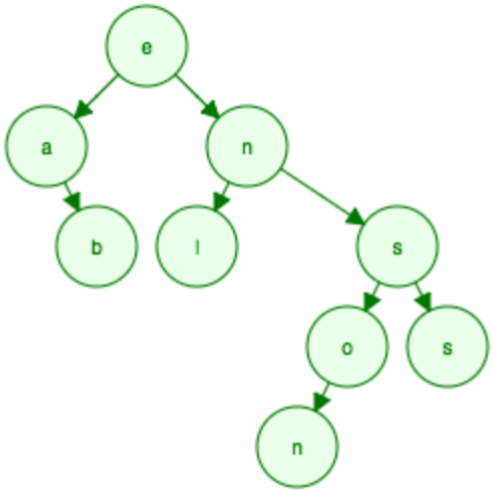


7 bogtaver: XuZhenY



7 bogtaver: Jakoben

b)



Insert b

Jakob Varmose Bentzen
Xu Zhen Yang
b blev indsættes på venstre siden af træet, bogstavet b er før H så placerer den på venstre siden, og derefter placerer den til højre siden at bogstavet a.

Delete H
Begge siden af træet er lige gode, nu har vi valgt at fjerne et tal fra venstre træet, og finder den største tal på venstre siden og erstattet med H som root.

Problem 5

```
public int sum() {
    if (root == null) {
        return 0;
    }
    return sumRec(root);
}

private int sumRec(BinaryTreeNode node) {
    int res = (Integer)node.element;
    if (node.left != null) {
        res += sumRec(node.left);
    }
    if (node.right != null) {
        res += sumRec(node.right);
    }
    return res;
}
```

Problem 6

```
public String preorder() {
    if (root == null) {
        return "";
    }
    return preorderRec(root);
}

private String preorderRec(BinaryTreeNode node) {
    String res = node.element.toString();
    if (node.left != null) {
        res += " " + preorderRec(node.left);
    }
    if (node.right != null) {
        res += " " + preorderRec(node.right);
    }
    return res;
}
```