Name: Tofik Mussa
Submitted to: Dr. David Sullivan

Problem Set 4, Part I

Problem 1: Uninformed state-space search
1-1) a -> b -> c -> d -> e -> f

1-2) a -> b -> d -> h -> e -> i

1-3) I am showing each iteration for clarity
a
a -> b -> c
a -> b -> d -> e -> c -> f -> g
a -> b -> d -> h -> e -> i -> j -> k

Problem 2: Determining the depth of a node

2-1)

It is O(1) in the best case, when the key we are looking for is at the root itself resulting in no recursive calls. In the worst case, we have to visit each node once to determine the depth of the key in the tree, so O(n). Whether the tree is balanced or not, we have to traverse the entire tree to find the key in the worst case. Since we are not cutting the problem size in half as in BST tree, it is still O(n) in both scenarios of a worst case

2-2)

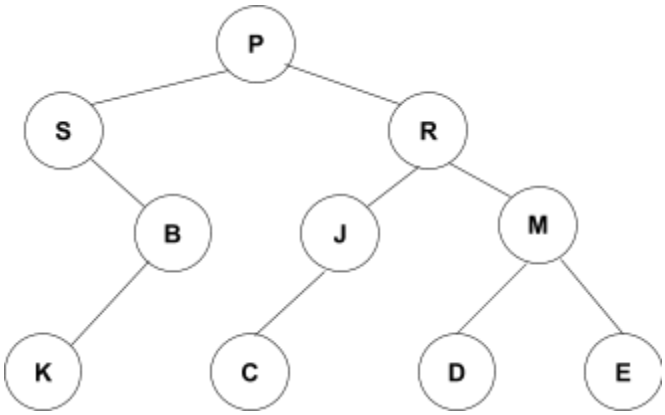
```
private static int depthInTree(int key, Node root) {
   if (root == null || key == root.key) {
       return 0;      // found
   } else {
       if(key < root.key){
           return 1 + depthInTree(key, root.left);
       } else if(key > root.key){
           return 1 + depthInTree(key, root.right);
       }
   }
   return -1;     // not found in either subtree
}
```

2-3)

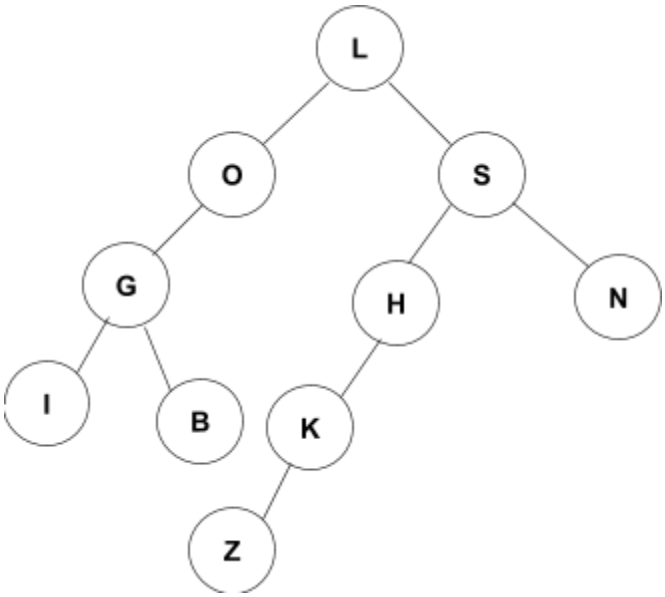
The best time complexity is O(1) when the key that we are looking for is at the root of the tree. For the worst case, if the tree is balanced, it is O(log n) because we are cutting the problem size in half each time we decide to go left or right. If the tree is not balanced in the worst case, it will be more like a linkedlist and it will be O(n). For example, if the tree was constructed from a sorted array and the key we are looking for was the maximum, this situation might happen

Problem 3: Tree traversal puzzles

3-1)

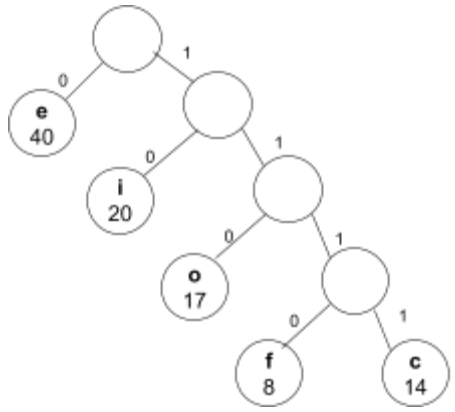


3-2)



Problem 4: Huffman encoding

4-1)



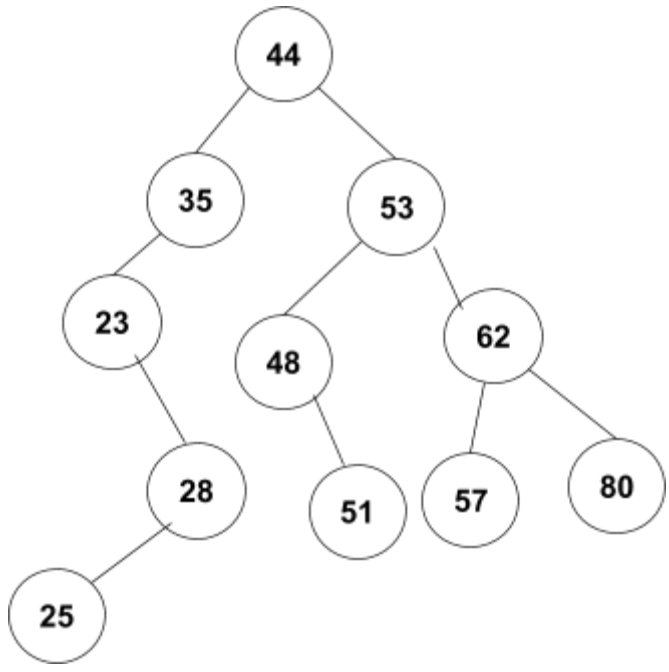
4-2) 110111011101011110

Problem 5: Binary search trees

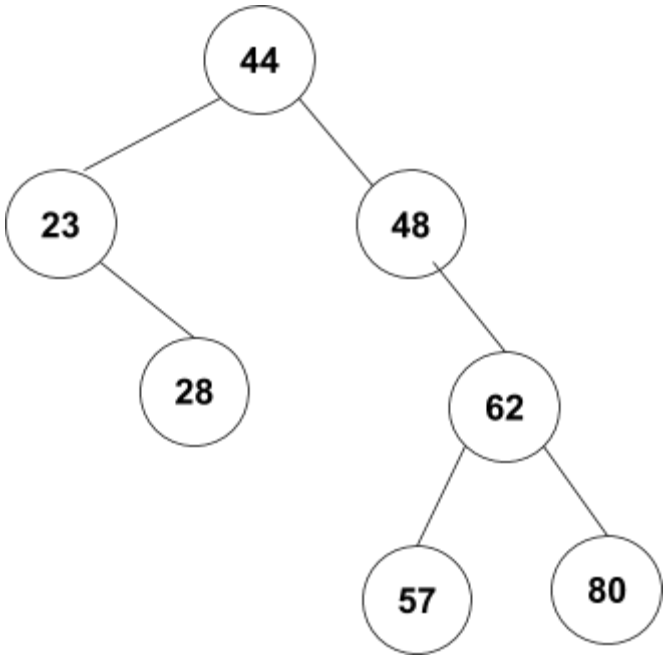
5-1) 44 35 23 28 53 48 62 57 80

5-2) 28 23 35 48 57 80 62 53 44

5-3)



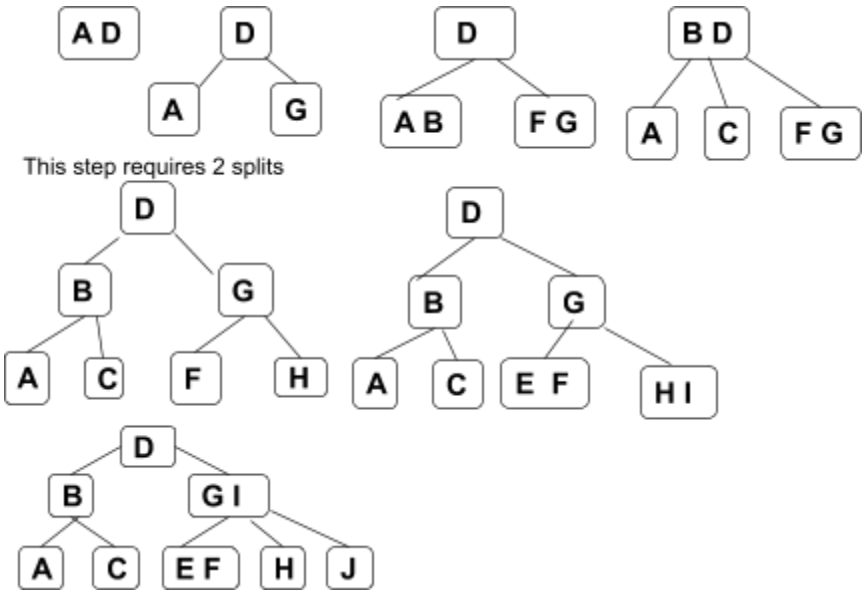
5-4)



5-5) No. Because for node 35, the length of the right subtree is -1 and the length of the left subtree is 1. 1 - (-1) = 2 which is bigger than 1

Problem 6: 2-3 Trees and B-trees

6-1)



6-2)