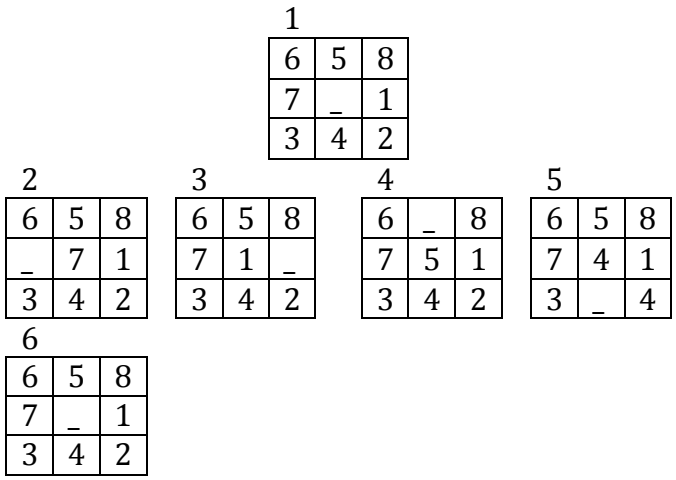
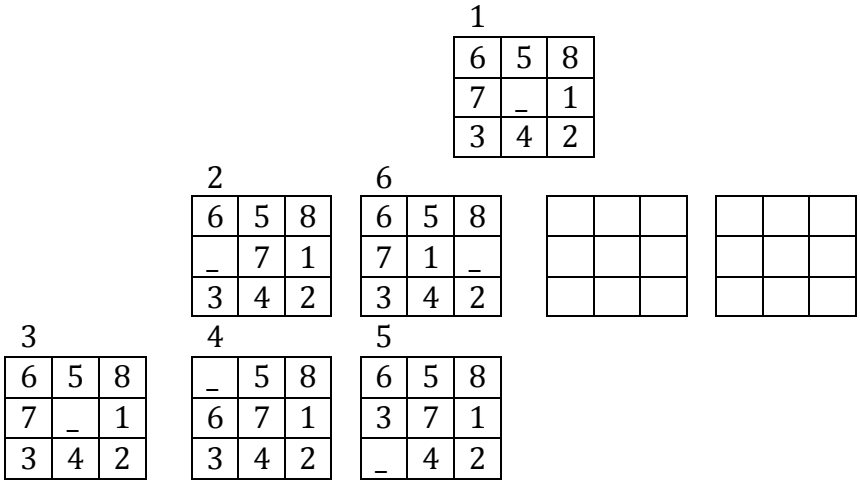
PS4

1- Uninformed state-space search (6pts)

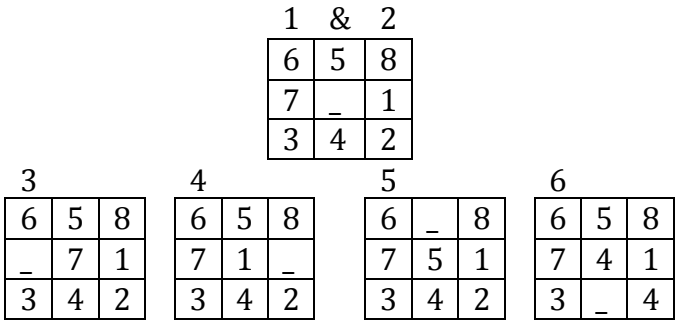
a. Breadth first search



b. Depth first search



c. Iterative deepening search



2- Determining the depth of a node (12pts)

a. Time complexity of binary tree

Tree of n nodes.

Best case:

key is the root
key == root.key at first recursive call.
Time complexity: O(1)

Worst case:

key is not in the tree.
balanced tree:
we have to visit all the nodes
time complexity: O(n)
unbalanced tree:
equivalent to a linked list, and we have to visit all the list.
time complexity: O(n)

b. Revised depthInTree()

```
private static int depthInTree (int key, Node root) {
    if (key == root.key)
        return 0;

    if (root.left != null && key < root.key){
        int depthInLeft = depthInTree (key, root.left);
        if (depthInLeft != -1)
            return depthInLeft + 1;
       }

   if (root.right != null && key >= root.key) {
        int depthInRight = depthInTree (key, root.right);
        if (depthInRight != -1)
            return depthInRight + 1;
      }

   return -1;
}
```

c. Time complexity of binary SEARCH tree

Tree of n nodes.

Best case:

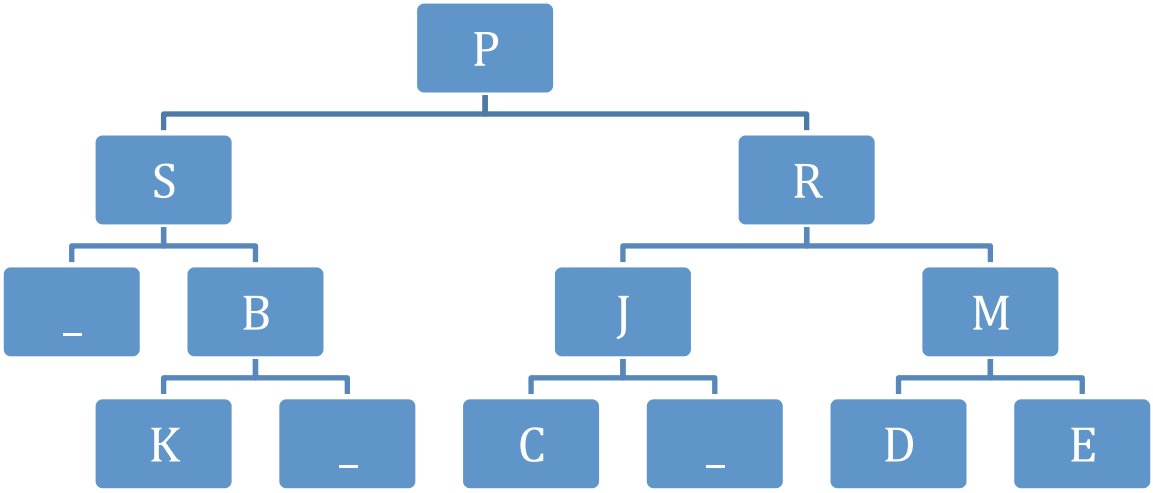
key is the root
key == root.key at first recursive call.
Time complexity: O(1)

Worst case:

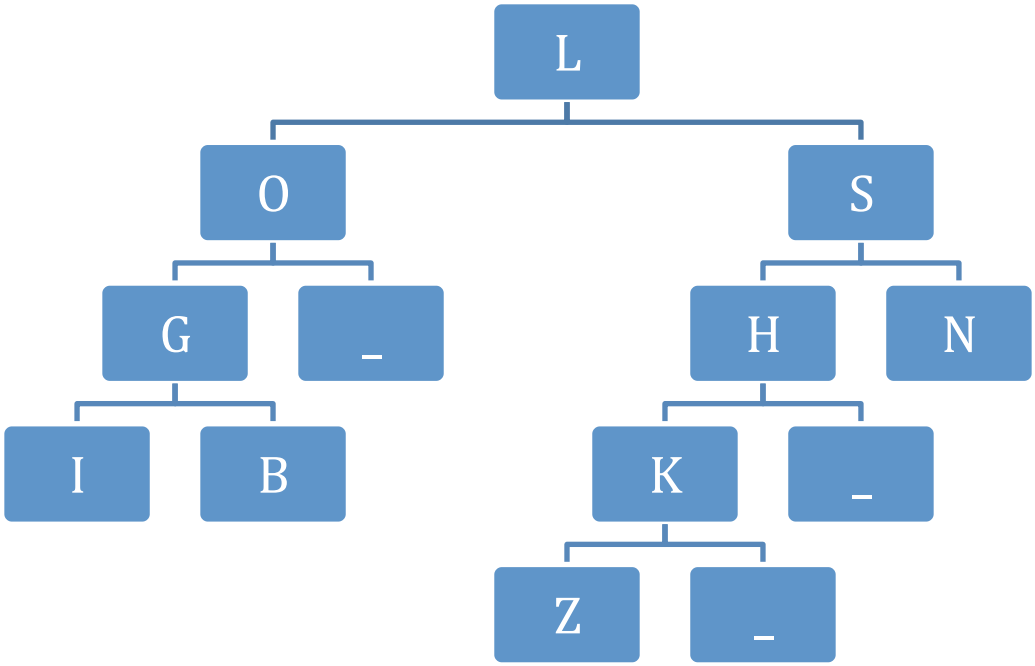
key is not in the tree.
balanced tree:
we divide the problem in 2 at each recursion
time complexity: $O(\log_2 n)$
unbalanced tree:
equivalent to a linked list, and we have to visit all the list.
time complexity: O(n)

3- Tree traversal puzzle (10pts)

a. inorder: SKBPCJRDME, preorder: PSBKRJCMDE



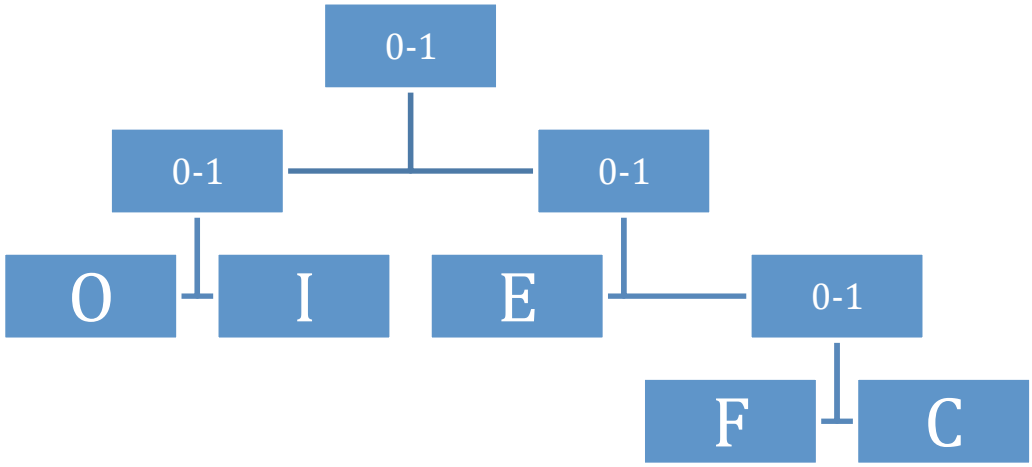
b. postorder: IBGOZKHNSL, preorder: LOGIBSHKZN



4- Huffman encoding (7pts)

a. Huffman tree

Ch	Fq
F	8
C	14
O	17
I	20
E	40



b. Encoding of a string

From the tree, we get the following encoding:

O = 00
F = 110
I = 01
C = 111
E = 10

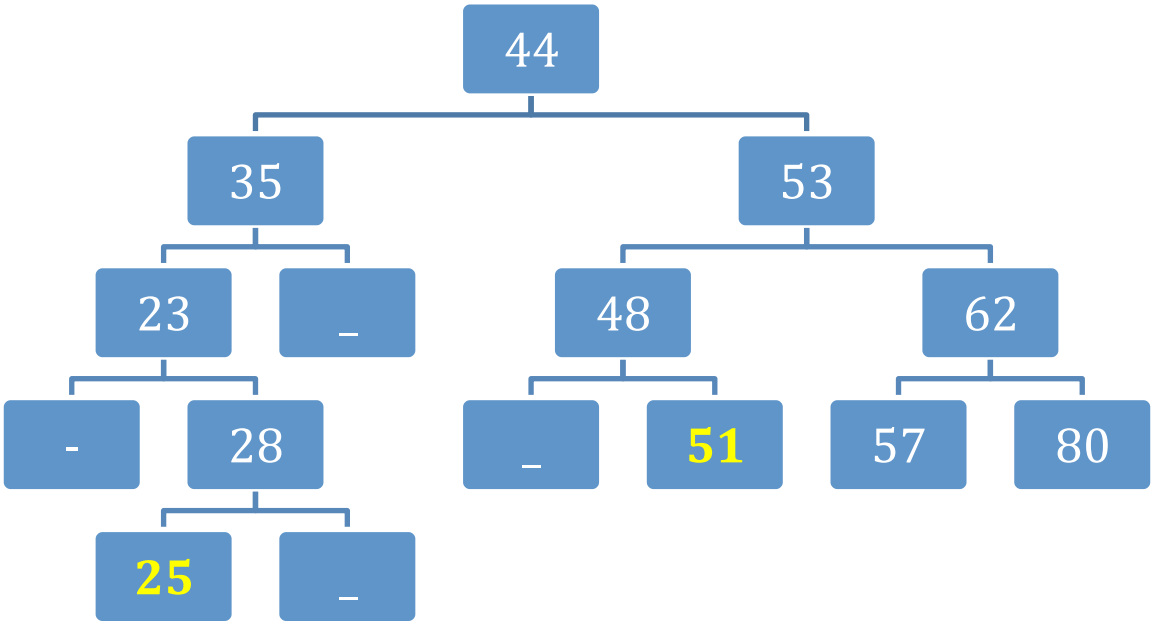
Therefore: **office = 00 110 110 01 111 10**

5- Binary search trees (10pts)

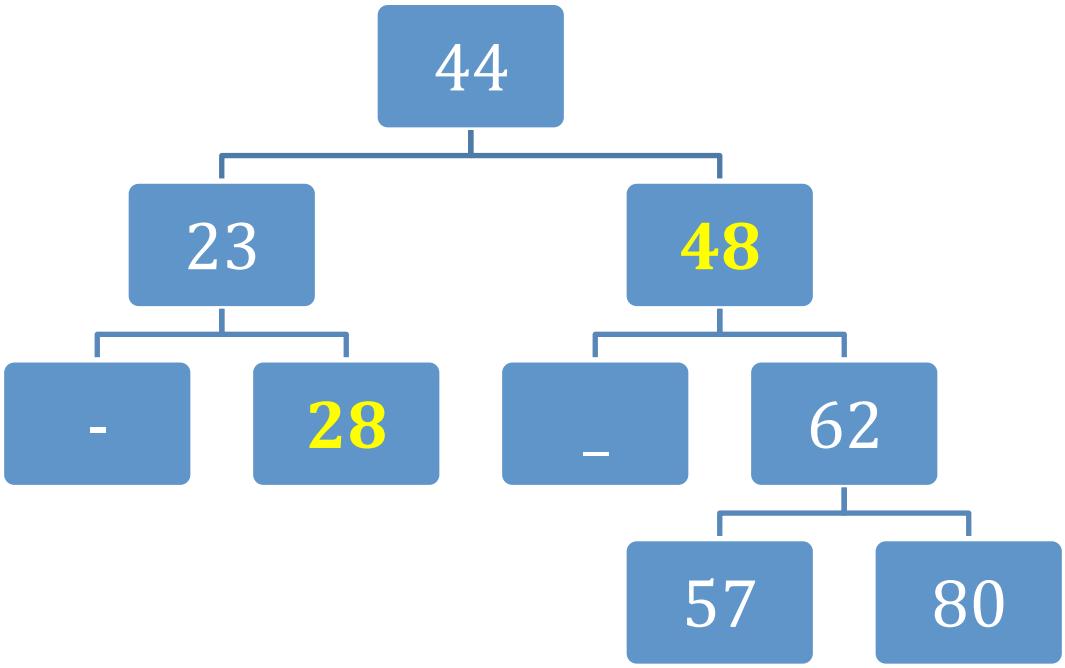
a. preorder, print at right (print if a node is a right child)
28, 53, 62, 80

b. post order
28, 23, 35, 48, 57, 80, 62, 53, 44

c. insert 25, insert 51



d. delete 53, delete 35



e. balanced tree

The original tree **is not** balanced because:
The right sub-tree of 35 has a height of 0.
The left sub-tree of 35 has a height of 2.
They differ by more than 1. Therefore, the tree is not balanced.

6- “2-3 Trees” and “B-Trees” (10pts)

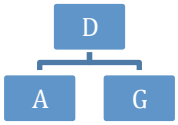
a. empty 2-3 tree

1- insert A

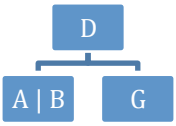
2- insert D

3- insert G

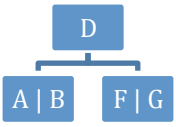
split



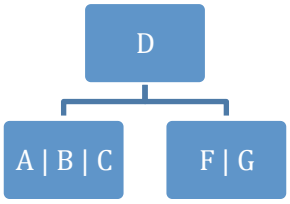
4- insert B



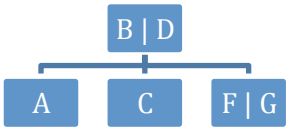
5- insert F



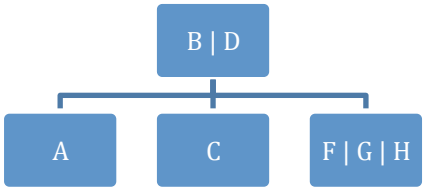
6- insert C



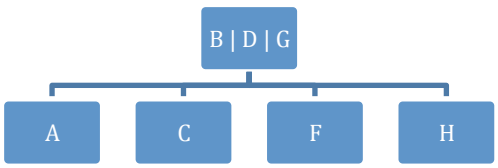
split

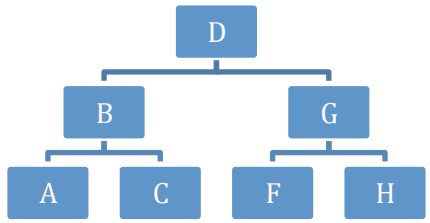


7- insert H



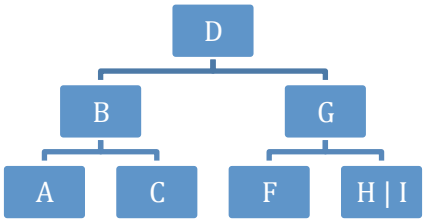
split



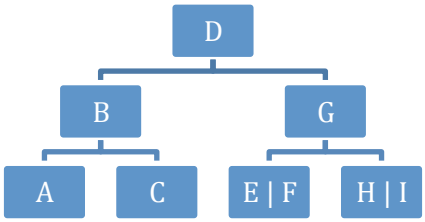


split

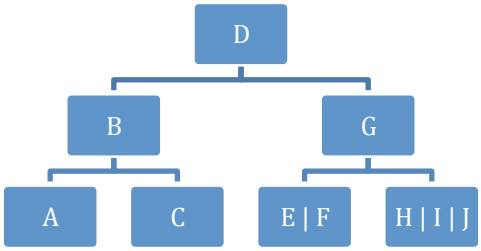
8- insert I

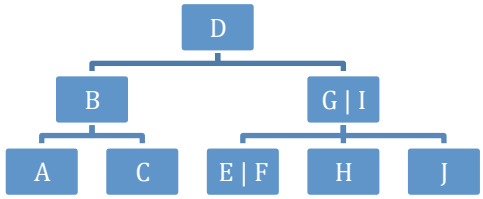


9- insert E



10- insert J





split

b. empty B-tree of order 2

1- insert A

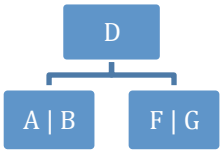
2- insert D

3- insert G

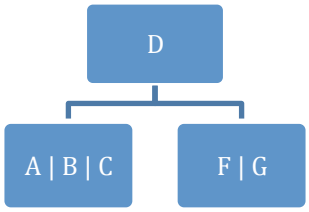
4- insert B

5- insert F

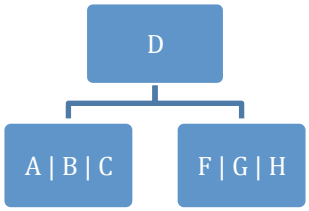
split



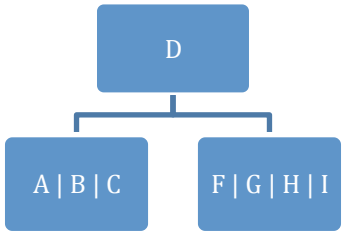
6- insert C



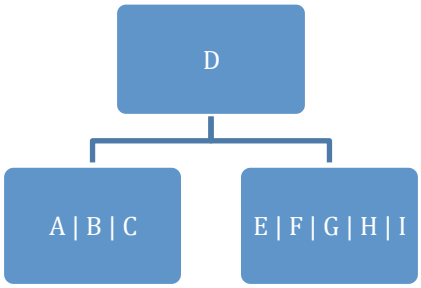
7- insert H

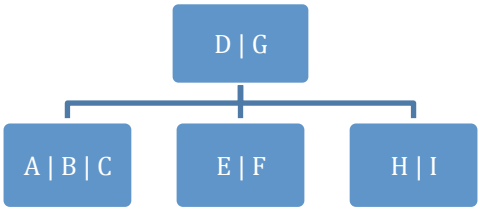


8- insert I



9- insert E





split

10- insert J