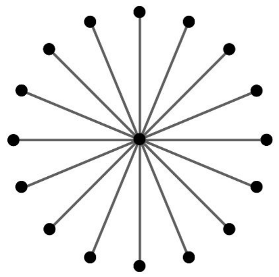
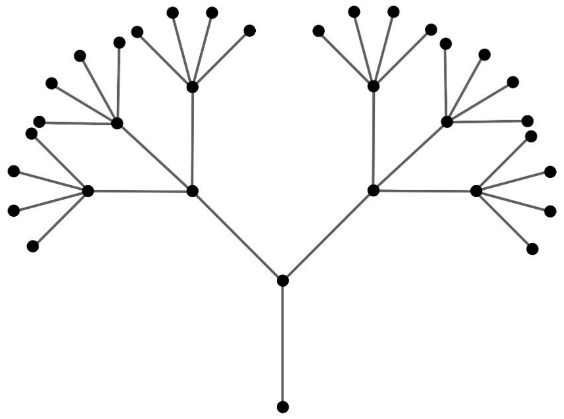
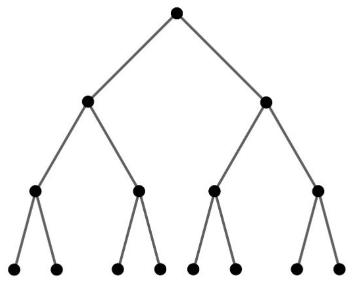
Trees

DATA STRUCTURES & ALGORITHMS

Tree

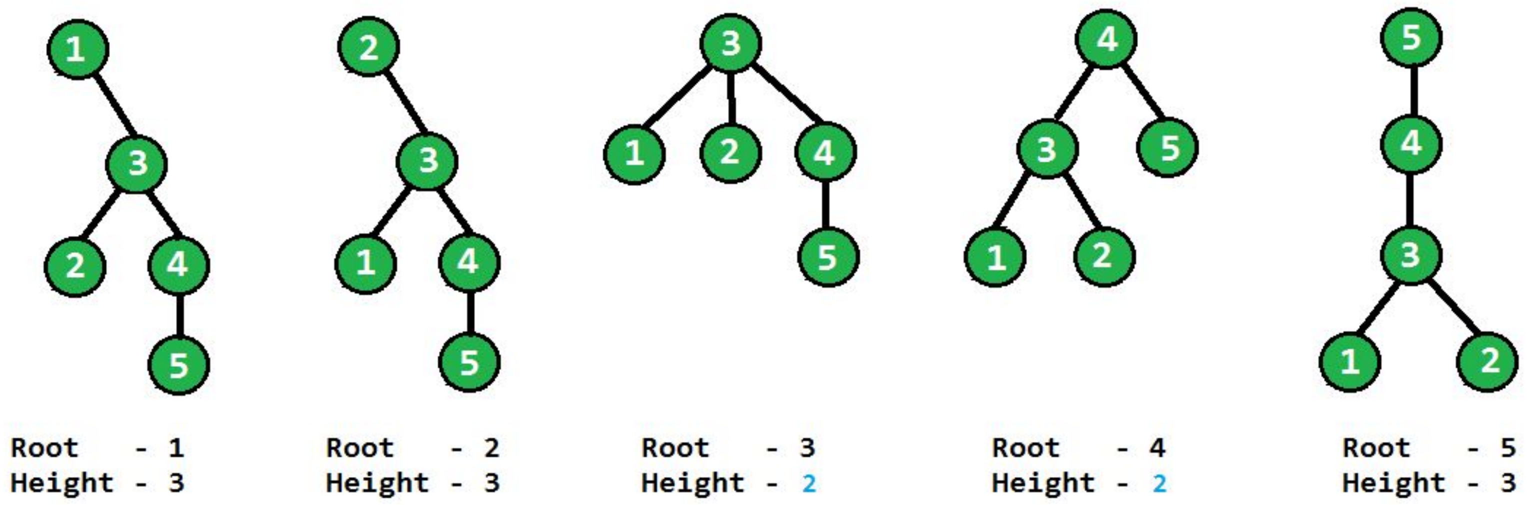
A tree is an undirected graph G that satisfies any of the following equivalent conditions:

- G is connected and acyclic (contains no cycles).
- G is acyclic, and a simple cycle is formed if any edge is added to G.
- G is connected, but would become disconnected if any single edge is removed from G

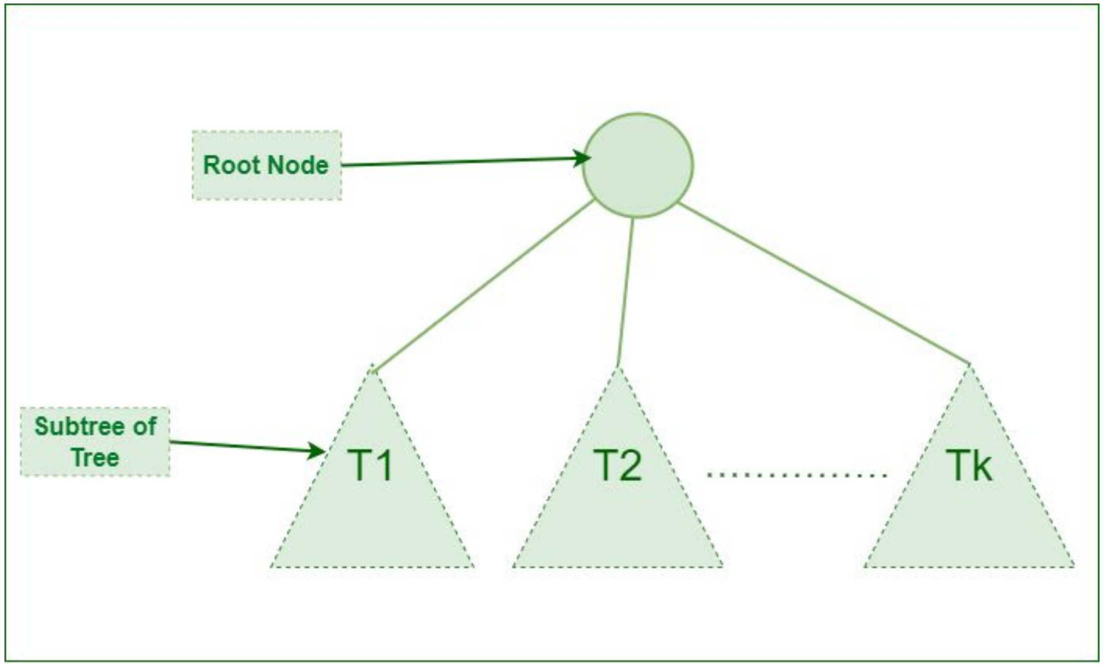
Rooted tree

A rooted tree is a tree in which a special ("labeled") node is singled out. This node is called the "root" of the tree.



Recursive definition

A tree **T** of order **n** is either empty or consists of a node called root of **T** together with at most n trees,called the subtrees of **T**, each of whose roots are connected by an edge from the root of **T**.

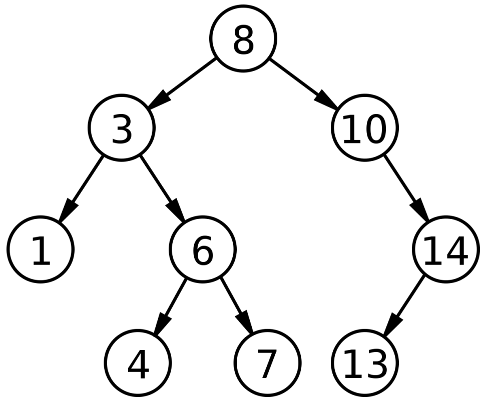


Representing rooted trees

- represent each node of a tree by a node.
- As with linked lists, we assume that each node contains a key attribute.
- The remaining attributes of interest are pointers to other nodes, and they vary according to the type of tree.

Types of trees

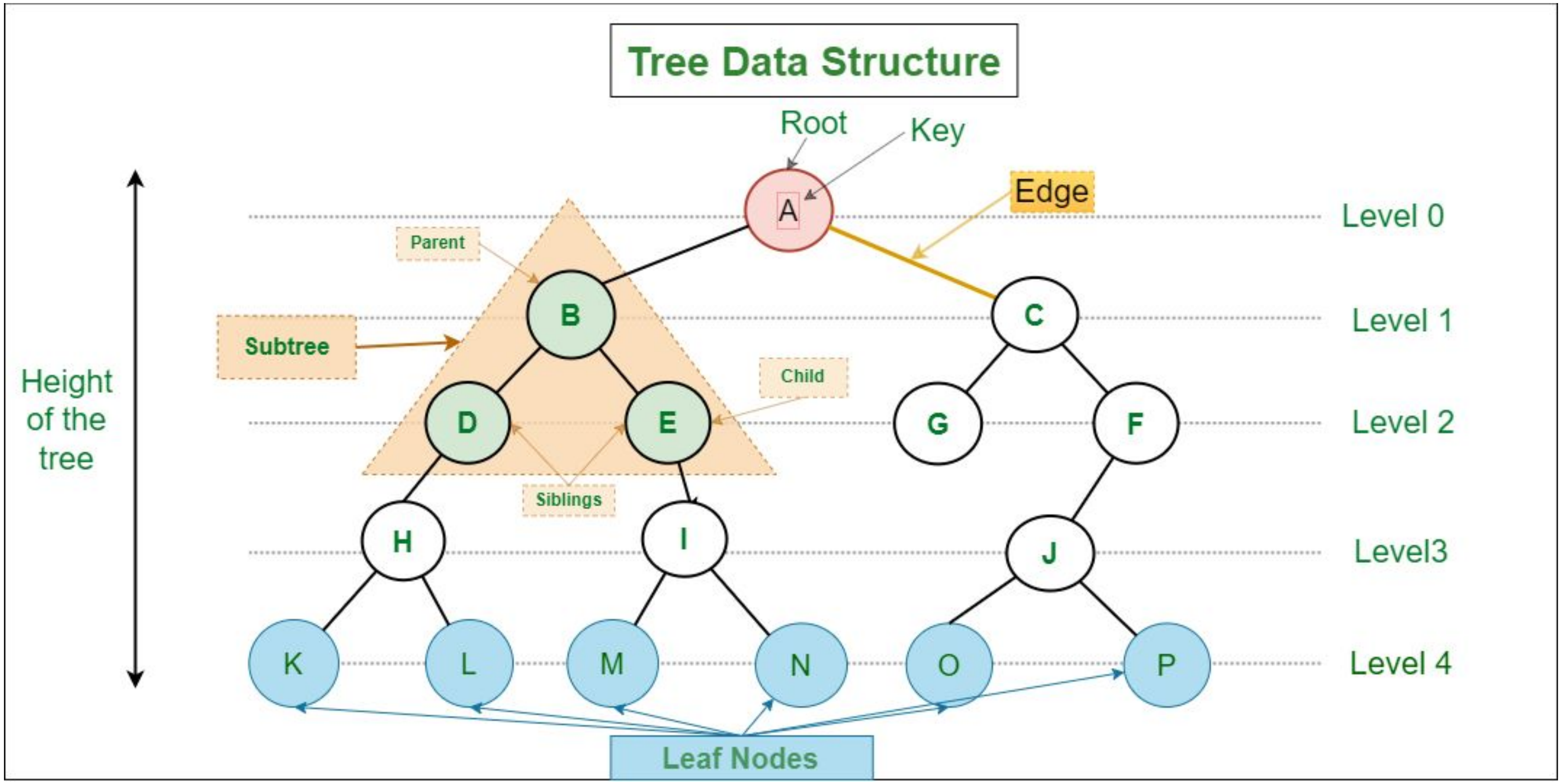
- **Binary tree:**
    - a Tree data structure with at most 2 children. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.
- **Ternary tree:**
    - a tree data structure in which each node has at most three child nodes, usually distinguished as “left”, “mid” and “right”.
- **N-ary(Generic) tree**



Tree attributes

- **The depth of a node** in a binary tree is the number of edges from the root node to another node. It is calculated by counting the number of edges traversed from the root node to the given node.
- **The height of a node** in a binary tree is defined as the number of edges from the node to the deepest leaf node.
- The height of the root node is therefore called the **height of the tree**.

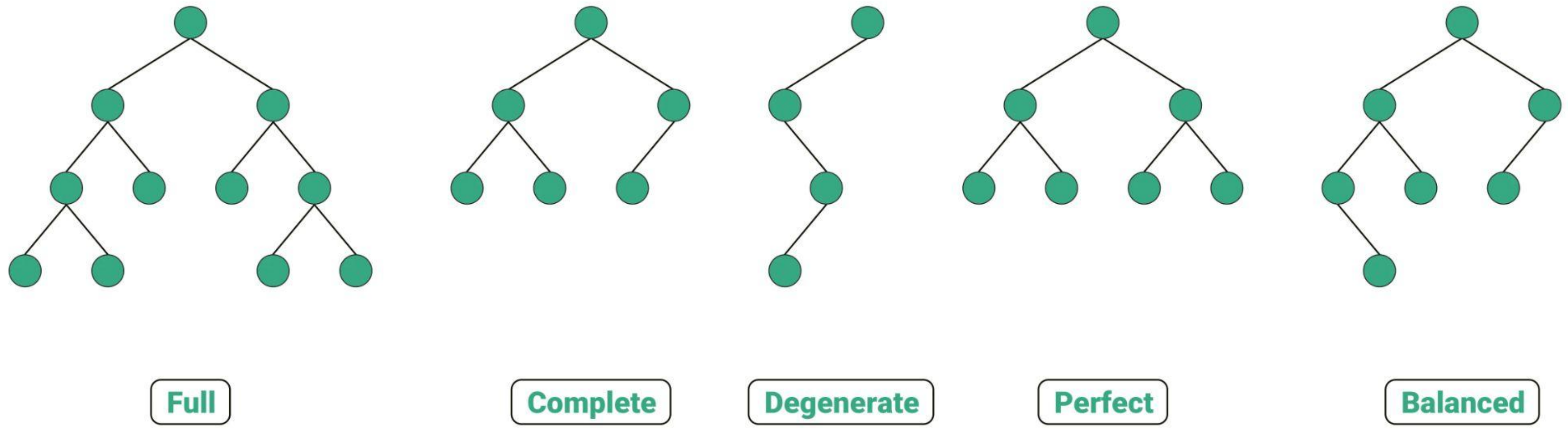
Tree attributes



Types of binary tree

- Based on the **number of children**:
    - Full Binary tree: every node has 0 or 2 children
    - Degenerated Binary tree
- Based on **completion of levels**:
    - Complete Binary Tree: all levels completely filled with nodes except the last level and in the last level, all the nodes are as left side as possible
    - Perfect Binary Tree: all internal nodes have 2 children and all the leaf nodes are at the same depth or same level.
    - Balanced Binary Tree: height of the left and the right sub-trees of every node may differ by at most 1

Types of binary tree



Exercise

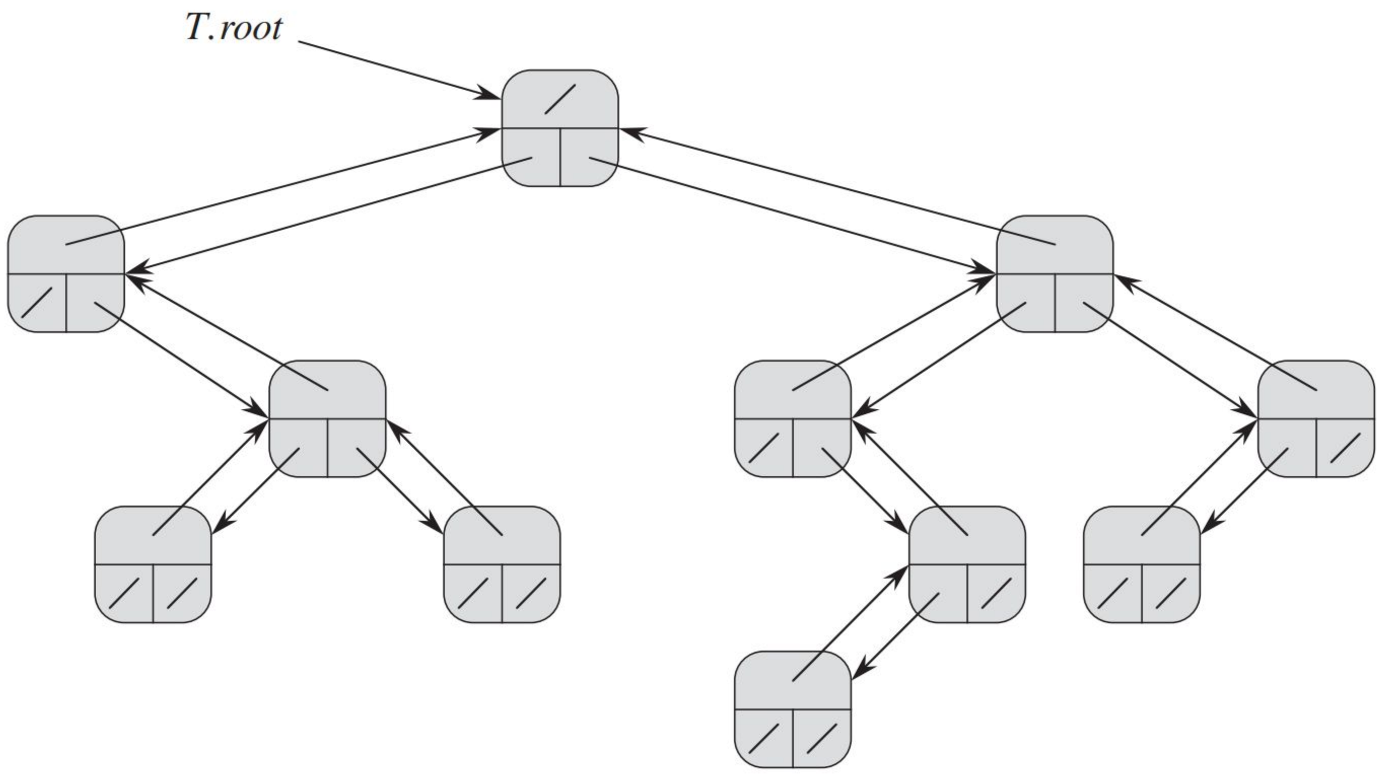
For Full Binary Tree:

Number of Leaf nodes = Number of Internal nodes + 1

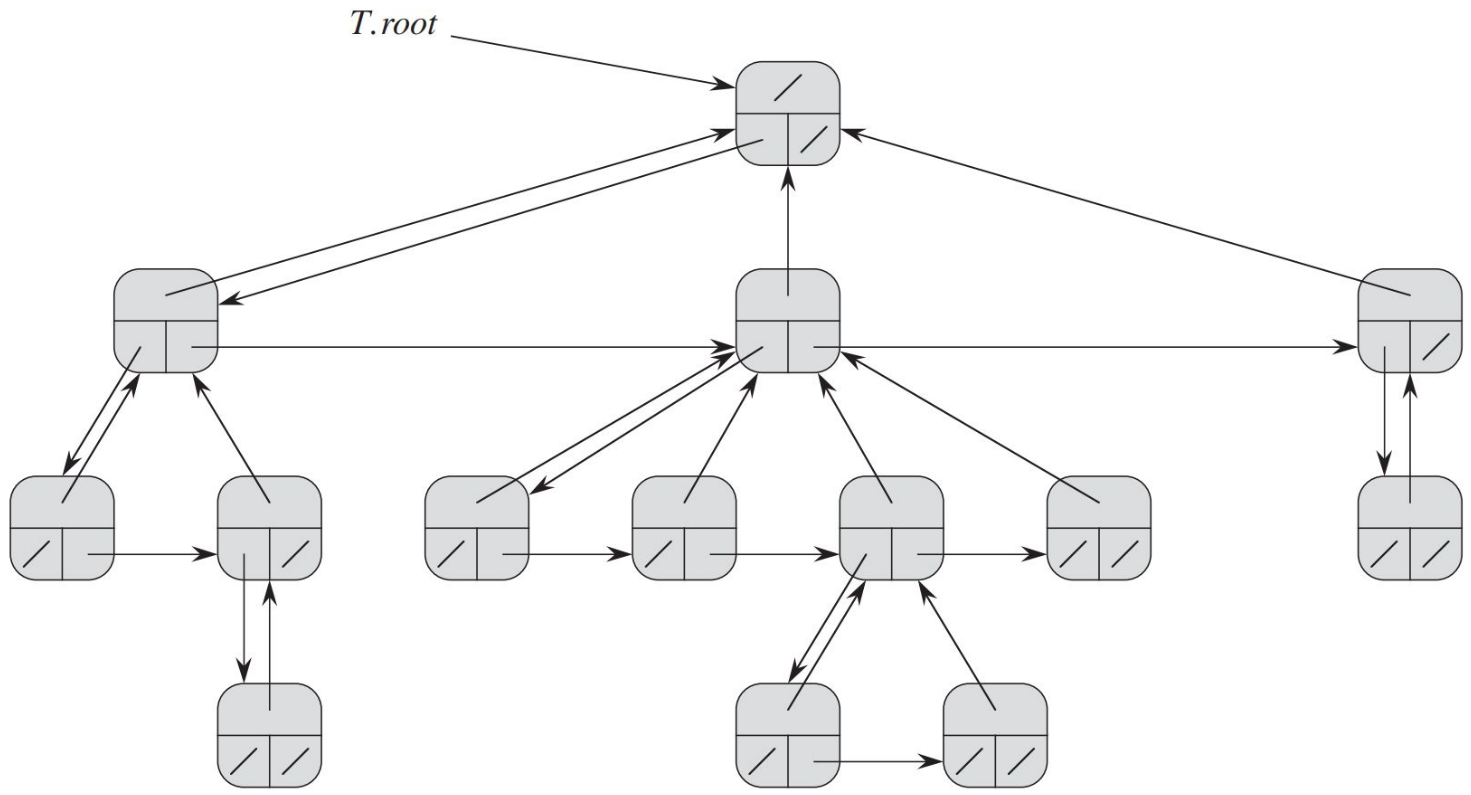
For Perfect Binary Tree:

Total number of nodes in a with height H is …….

Rooted binary tree representation



Rooted tree representation



Exercise

10.4-2

Write an $O(n)$-time recursive procedure that, given an n-node binary tree, prints out the key of each node in the tree.

10.4-3

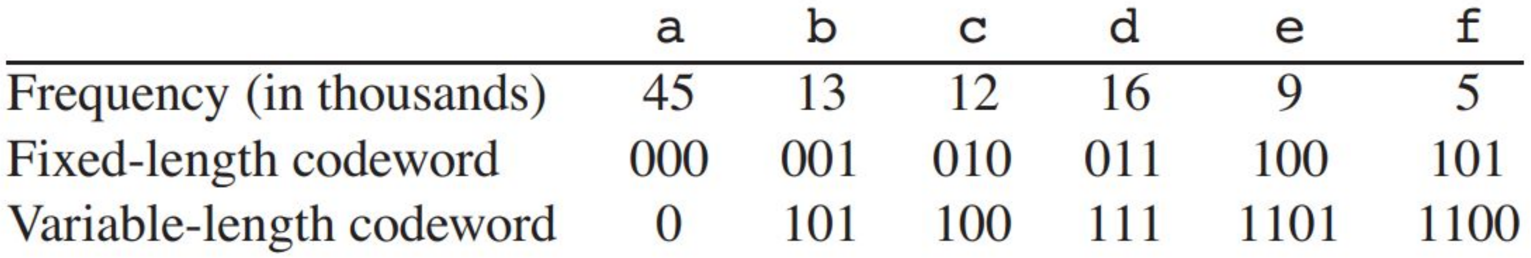
Write an $O(n)$-time nonrecursive procedure that, given an n-node binary tree, prints out the key of each node in the tree. Use a stack as an auxiliary data structure.

Some Apllications

- Unix directory structure is tree
- Data Compression
- Database Indexing
-

Huffman Coding

. A data file of 100,000 characters contains only the characters a–f, with the frequencies indicated. If we assign each character a 3-bit codeword, we can encode the file in 300,000 bits. Using the variable-length code shown, we can encode the file in only 224,000 bits.

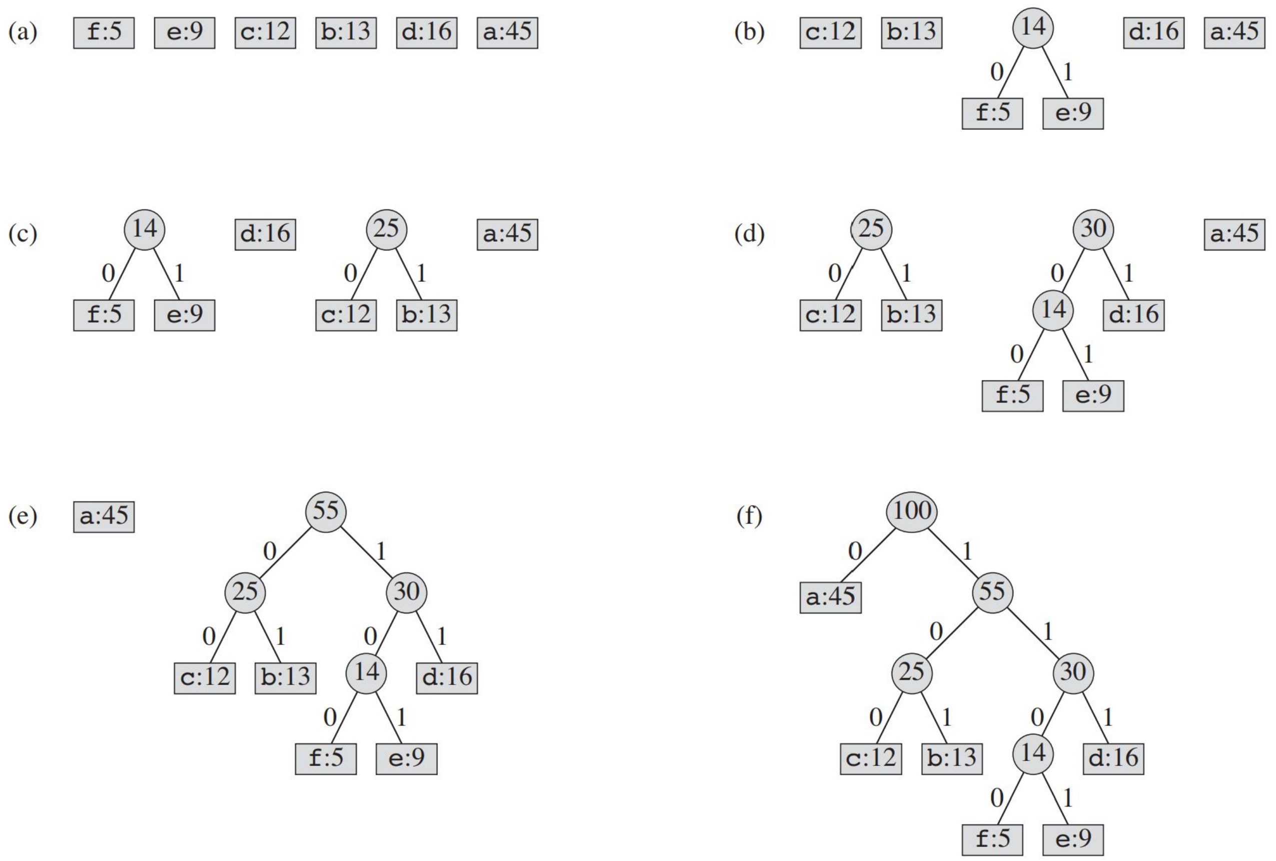


	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

Huffman Coding

```
HUFFMAN($C$)
n = |C|
Q = C
for $i = 1$ to $n - 1$
    allocate a new node $z$
    $z.left = x =$ EXTRACT-MIN($Q$)
    $z.right = y =$ EXTRACT-MIN($Q$)
    $z.freq = x.freq + y.freq$
    INSERT($Q, z$)
return EXTRACT-MIN($Q$)        // return the root of the tree
```

Huffman Coding


(a)
f:5
e:9
c:12
b:13
d:16
a:45
(b)
c:12
b:13
14
d:16
a:45
f:5
e:9
(c)
14
d:16
25
a:45
f:5
e:9
c:12
b:13
(d)
25
30
a:45
c:12
b:13
14
d:16
f:5
e:9
(e)
a:45
55
25
30
c:12
b:13
14
d:16
f:5
e:9
(f)
100
a:45
55
25
30
c:12
b:13
14
d:16
f:5
e:9