Assignment I: Trees

**Q.1 Discuss binary tree extension and what are its use?**

Solution :

**Types of binary trees:**

* A**rooted binary tree** is a rooted tree in which every node has at most two children.
* A **full binary tree**(sometimes proper binary tree or 2-tree or strictly binary tree) is a tree in which every node other than the leaves has two children.
* A perfect binary tree is a full binary tree in which all leaves are at the same depth or same level. (This is ambiguously also called a complete binary tree.)
* A **complete binary tree** is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.
* A **balanced binary tree** is where the depth of all the leaves differs by at most 1. Balanced trees have a predictable depth (how many nodes are traversed from the root to a leaf, root counting as node 0 and subsequent as 1, 2, ..., depth). This depth is equal to the integer part of log2(n) where n is the number of nodes on the balanced tree. Example 1: balanced tree with 1 node, log2(1) = 0 (depth = 0). Example 2: balanced tree with 3 nodes, log2(3) = 1.59 (depth=1). Example 3: balanced tree with 5 nodes, log2(5) = 2.32 (depth of tree is 2 nodes).
* **A**rooted complete binary tree**can be identified with a free magma.**
* **A**degenerate tree**is** a tree where for each parent node, there is only one associated child node. This means that in a performance measurement, the tree will behave like a linked list data structure.
* A rooted tree has a top node as root.

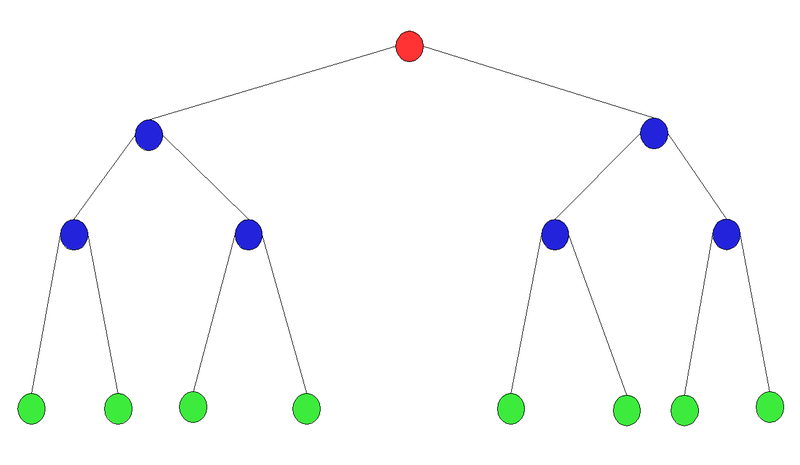
**Q.2 Describe the concept of binary tree and how we can convert general tree to binary tree**

Solution:

**Binary Trees**

 A binary tree is a tree data structure in which each node has at most two children. Typically the first node is known as the parent and the child nodes are called left and right. In type theory, a binary tree with nodes of type A is defined inductively as TA = μα. 1 + A × α × α. Binary trees are commonly used to implement binary search trees and binary heaps.

**Definition:** A binary tree is a finite set of nodes which is either empty or consists of a root and two disjoint binary trees called the left subtree and the right subtree.



Since the references now access either the first child or successive siblings, the process must use this type of information rather than magnitude as was the case for the binary search tree. Note that the resulting tree is a binary tree but not a binary search tree.

The process of converting the general tree to a binary tree is as follows:

\* use the root of the general tree as the root of the binary tree

\* determine the first child of the root. This is the leftmost node in the general tree at the next level

\* insert this node. The child reference of the parent node refers to this node

\* continue finding the first child of each parent node and insert it below the parent node with the child reference of the parent to this node.

\* when no more first children exist in the path just used, move back to the parent of the last node entered and repeat the above process. In other words, determine the first sibling of the last node entered.

\* complete the tree for all nodes. In order to locate where the node fits you must search for the first child at that level and then follow the sibling references to a nil where the next sibling can be inserted. The children of any sibling node can be inserted by locating the parent and then inserting the first child. Then the above process is repeated.

Given the following general tree:

A

D

C

B

F

E

J

I

H

K

G

The following is the binary version:

A

B

C

K

D

H

E

I

F

J

G

**Q.3 Discuss Huffman’s coding with example.**

**Huffman Tree**

* A Method for the Construction of Minimum Redundancy Codes.
* Not all characters occur with the same frequency.
* Yet all characters are allocated the same amount of space.
* Code word lengths vary and will be shorter for the more frequently used characters.

**The (Real) Basic Algorithm**

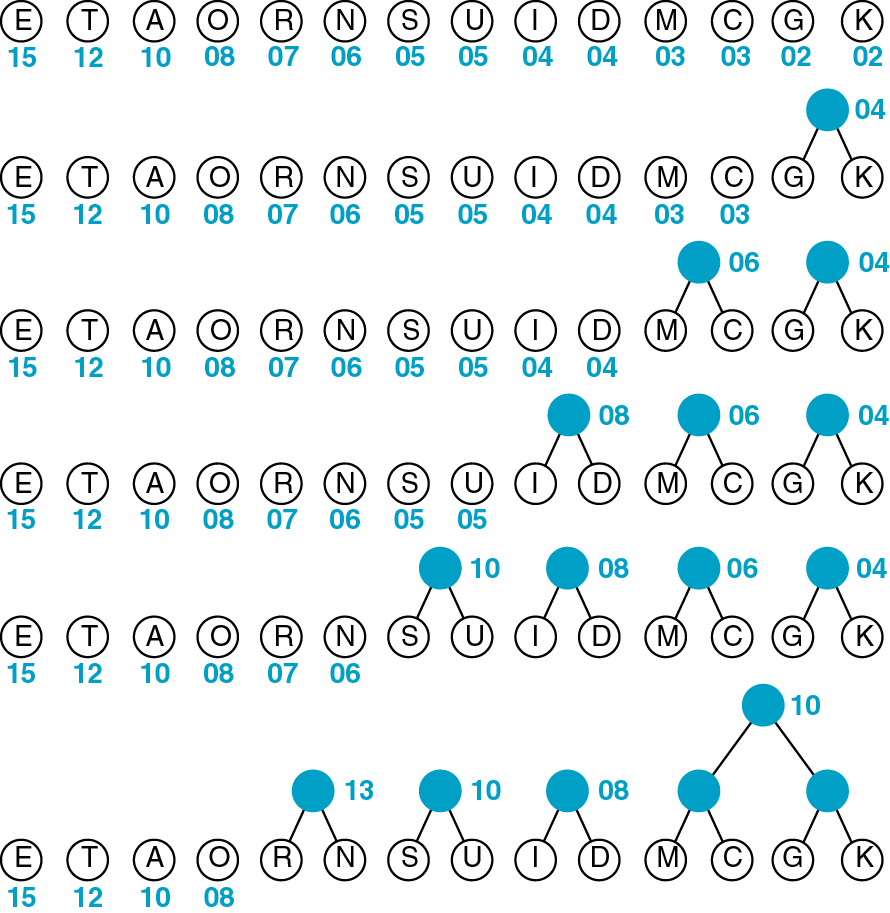
**1.** Scan text to be compressed and tally occurrence of all characters.

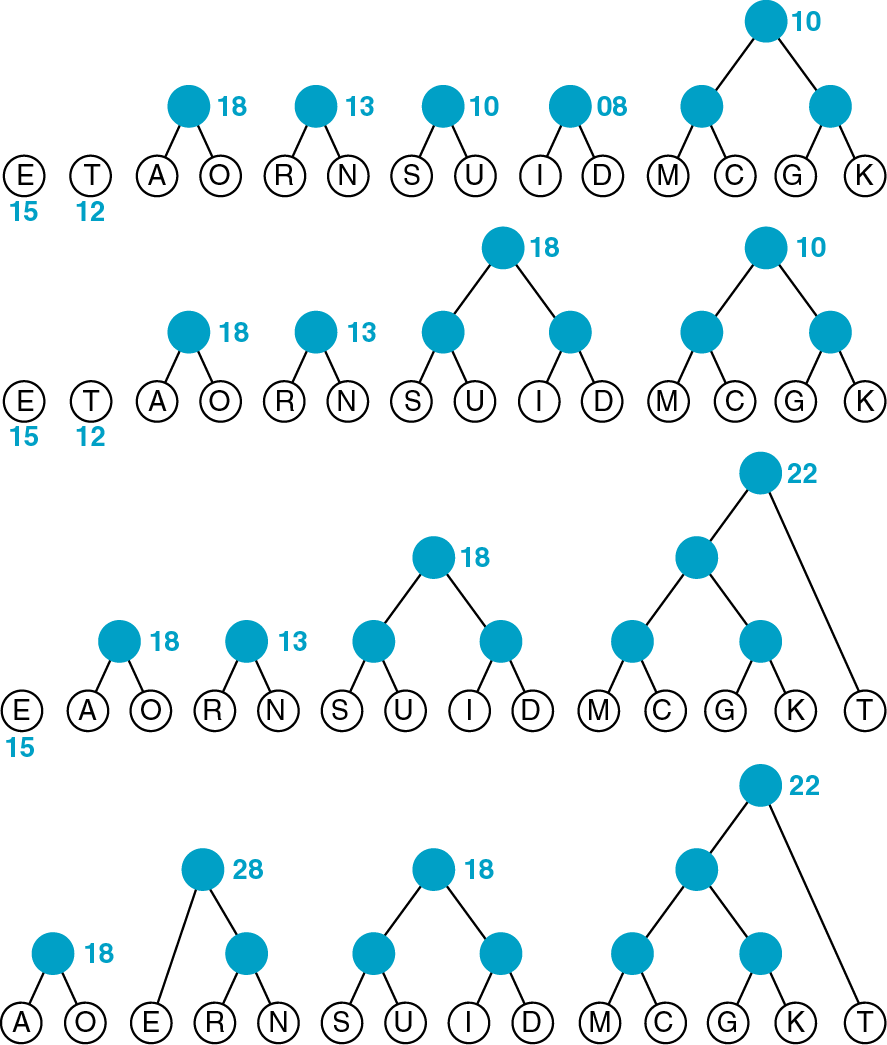
2. Sort or prioritize characters based on number of occurrences in text.

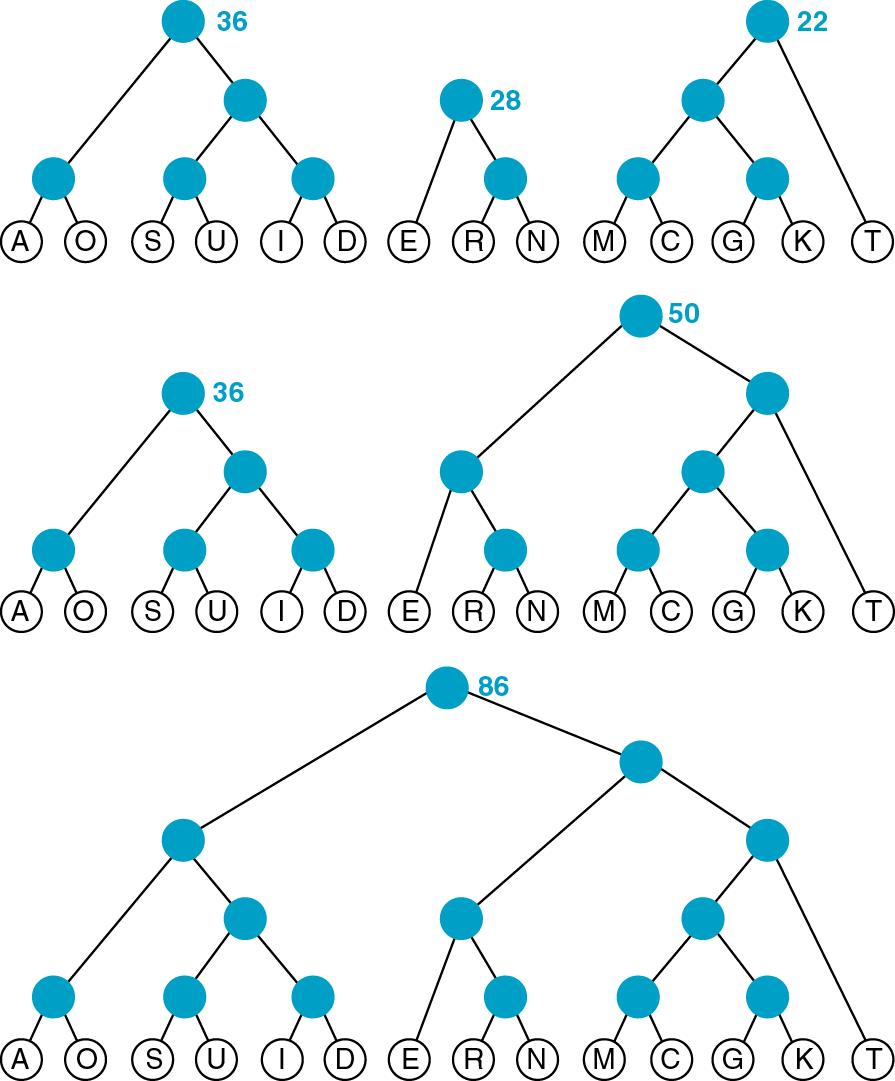
3. Build Huffman code tree based on prioritized list.

4. Perform a traversal of tree to determine all code words.

5. Scan text again and create new file using the Huffman codes**.**







**Q.4 Discuss the concept of Threaded binary tree.**

A Threaded Binary Tree is a binary tree in which every node if it does not have a right child and left child also has a THREAD (in actual sense, a link) to its INORDER successor and predecessor respectively. By doing this threading we avoid the recursive method of traversing a Tree, which makes use of stacks and consumes a lot of memory and time.

The node structure for a threaded binary tree varies a bit and its like this --

struct node

{

int data;

struct node \*lptr, \*rptr, \*lthread, \*rthread;

}

Let's make the Threaded Binary tree out of a normal binary tree...



The INORDER traversal for the above tree is -- D B A E C.

So, the respective Threaded Binary tree will be --



B has no right child and its inorder successor is A and so a thread has been made in

between them. Similarly, for D and E. C has no right child but it has no inorder successor even, so it has a hanging thread.

**Q.5 Use binary tree to evaluate expression tree with example.**