Chapter 10
Trees

In this chapter we consider an important class of graphs, called *Trees*. Some basic properties of trees and their immediate consequences and related concepts are presented. The topic of *Prefix Codes* is introduced and illustrated.

10.1 Trees and their Basic properties

A graph G is said to be a ***tree*** *if it is connected and has no cycles.*

It immediately follows that a tree has to be a simple graph; because loops and parallel edges form cycles.

The graphs shown in Figure 10.1 are all trees. We observe that each of these trees possesses at least two pendant vertices. A pendant vertex of a tree is also called a ***leaf***.

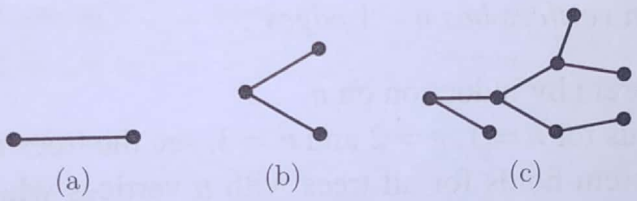


Figure 10.1

A graph which is a tree is usually denoted by T (instead of G) to emphasize the structure.

The graphs shown in Figure 10.2 are not trees. Observe that the first of these contains a cycle whereas the second is not connected. However, each component of the second (disconnected) graph is a tree. Such a graph is called a ***forest***.

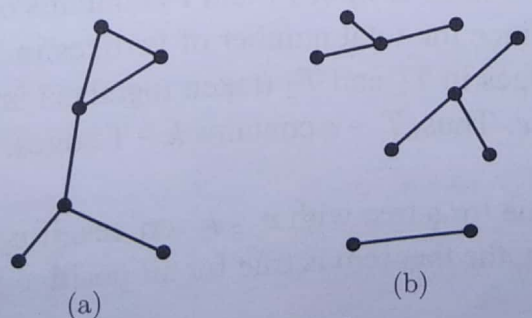


Figure 10.2

The following theorems contain some basic properties of trees.

Theorem 1 . *In a tree, there is one and only one path between every pair of vertices.*

Proof: Let T be a tree. Then T is a connected simple graph. Since T is connected, there must be at least one path between every two vertices. If there are two paths between a pair of vertices of T, the union of the paths will become a cycle, and T cannot be a tree. Thus, between every pair of vertices in a tree there must exist one and only one path. •

Theorem 2 . *If in a graph G there is one and only one path between every pair of vertices, then G is a tree.*

Proof: Since there is a path between every pair of vertices in G, it is obvious that G is connected. Since there is only one path between every pair of vertices, G cannot have a cycle. Because, if there is a cycle, then there exist two paths between two vertices on the cycle. Thus, G is a connected graph containing no cycles. This means that G is a tree. •

The above two theorems may be combined together and put in the following form:

A graph G is a tree if and only if there is one and only one path between every pair of vertices in G.

Theorem 3 . *A tree with n vertices has $n - 1$ edges*.*

Proof: We prove the theorem by induction on n.

The theorem is obvious for $n = 1$, $n = 2$ and $n = 3$; see the trees in Figure 10.1.

Assume that the theorem holds for all trees with n vertices where $n \leq k$, for a specified positive integer k.

Consider a tree T with $k + 1$ vertices. In T, let e be an edge with end vertices u and v. Since T is a tree, it has no cycles and therefore there exists no other edge or path between u and v. Hence, deletion of e from T will disconnect the graph and $T - e$ consists of exactly two components, say T_1 and T_2. Since T does not contain any cycle, the components T_1 and T_2 too do not contain any cycles. Hence, T_1 and T_2 are trees in their own right. Both of these trees have less than $k + 1$ vertices each, and therefore, according to the assumption made, the theorem holds for these trees; that is, each of T_1 and T_2 contains one less edge than the number of vertices in it. Therefore, since the total number of vertices in T_1 and T_2 (taken together) is $k + 1$, the total number of edges in T_1 and T_2 (taken together) is $(k + 1) - 2 = k - 1$. But T_1 and T_2 taken together is $T - e$. Thus, $T - e$ contains $k - 1$ edges. Consequently, T has exactly k edges.

Thus, if the theorem is true for a tree with $n \leq k$ vertices, it is true for a tree with $n = k + 1$ vertices. Hence, by induction, the theorem is true for all positive integers n. •

*In other words, for a tree of n vertices and m edges, we have $m = n - 1$. Said alternatively, this means that for a tree $T = (V, E)$, we have $|E| = |V| - 1$, or equivalently $|V| = |E| + 1$.

Theorem 4 . *Any connected graph with n vertices and n – 1 edges is a tree.*

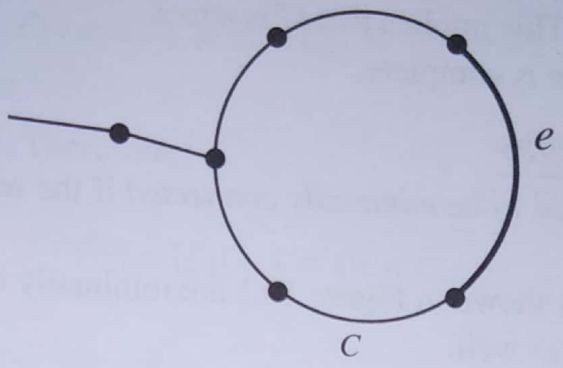


Figure 10.3

Proof: Let G be a connected graph with n vertices and $n - 1$ edges. Assume that G is not a tree. Then G contains a cycle, say C. Let e be an edge in C. The graph G will not become disconnected if e is deleted (see Figure 10.3). Thus, $G - e$ is a connected graph. But, on the other hand, $G - e$ has n vertices and $n - 2$ edges; therefore, it cannot be connected.* This is a contradiction. Hence, G must not have a cycle; this means that G must be a tree.

This completes the proof of the theorem. •

Remark: A disconnected graph with n vertices and $n - 1$ edges need not be a tree; see the graph shown below (which is disconnected and has 4 vertices and 3 edges).

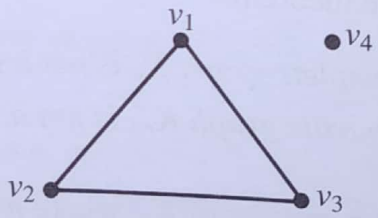


Figure 10.4

Theorems 3 and 4 can be put together in the following combined form:
A graph with n vertices is a tree if and only if it is connected and has n – 1 edges.

Theorem 5 . *A connected graph G is a tree if and only if adding an edge between any two vertices in G creates exactly one cycle in G.*

Proof: Suppose a connected graph G is a tree. Then G has no cycles and there is exactly one path between any two vertices, u, v. If we add an edge between u and v, then an additional path is created between u and v and the two paths constitute a cycle. Since G had no cycles earlier, this is the only cycle which G now possesses.

*Recall Theorem 2, Section 9.7.

Conversely, suppose G is connected and adding an edge between any two vertices u and v in G creates exactly one cycle in G. This implies that, before adding this edge, exactly one path was there between u and v. This implies that G is a tree.

The proof of the theorem is complete. •

Minimally connected graphs

A connected graph is said to be *minimally connected* if the removal of any one edge from it disconnects the graph.

For example, the graphs shown in Figure 10.1 are minimally connected. As already noted, all of these graphs are trees as well.

Theorem 6 . *A connected graph is a tree if and only if it is minimally connected.*

Proof: Suppose G is a connected graph which is not a tree. Then G contains a cycle C. The removal of any one edge e from this cycle will not make the graph disconnected. Therefore, G is not minimally connected. Thus, if a connected graph is not a tree then it is not minimally connected. This is equivalent to saying that if a connected graph is minimally connected then it is a tree (contrapositive).

Conversely, suppose G is a connected graph which is not minimally connected. Then there exists an edge e in G such that $G - e$ is connected. Therefore, e must be in some cycle in G. This implies that G is not a tree. Thus, if a connected graph is not minimally connected then it is not a tree. This is equivalent to saying that if a connected graph is a tree, then it is minimally connected (contrapositive).

This completes the proof of the theorem. •

Example 1 **(a)** *Show that the complete graph K_n is not a tree when $n > 2$.*

(b) *Show that the complete bipartite graph $K_{r,s}$ is not a tree when $r \geq 2$.*

► (a) If v_1, v_2, v_3 are any three vertices of K_n, where $n > 2$, then the closed walk $v_1v_2v_3v_1$ is a cycle in K_n. Since K_n has a cycle, it cannot be a tree.

(b) Let v_1 and v_2 be any two vertices in the first bipartite and v_1', v_2' be any two vertices in the other bipartite of $K_{r,s}$, with $s \geq r > 1$. Then, the closed walk $v_1v_1'v_2v_2'v_1$ is a cycle in $K_{r,s}$. Since $K_{r,s}$ has a cycle, it cannot be a tree. ■

Example 2 *Prove that a graph with n vertices, $n - 1$ edges and no cycles is connected.*

► Consider a graph G which has n vertices, $n - 1$ edges and no cycles. Suppose G is not connected. Let the components of G be H_i, $i = 1, 2, \ldots k$. If H_i has n_i vertices, we have $n_1 + n_2 + \cdots + n_k = n$. Since G has no cycles, H_is also do not have cycles. Further, they are all connected graphs. Therefore, they are trees. Consequently, each H_i must have $n_i - 1$ edges. Therefore, the total number of edges in these H_is is

$$(n_1 - 1) + (n_2 - 1) + \cdots + (n_k - 1) = n - k.$$

This must be equal to the total number of edges in G; that is $n - k = n - 1$. This is not possible, since $k > 1$. Therefore, G must be connected. ■

Example 3 *Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be two trees. If $|E_1| = 19$ and $|V_2| = 3|V_1|$, determine $|V_1|$, $|V_2|$, and $|E_2|$.*

▶ It is given that $|E_1| = 19$. Therefore,

$$|V_1| = |E_1| + 1 = 19 + 1 = 20.$$

Since $|V_2| = 3|V_1|$ as given, we get

$$|V_2| = 3|V_1| = 3 \times 20 = 60.$$

Lastly,

$$|E_2| = |V_2| - 1 = 60 - 1 = 59.$$ ■

Example 4 *If a tree has 2020 vertices, find the sum of the degrees of the vertices.*

▶ Since the number of vertices in the given tree is $|V| = 2020$, the number of its edges is $|E| = |V| - 1 = 2019$. Therefore (by hand-shaking property) the sum of degrees of its vertices $= 2|E| = 4038$. ■

Example 5 *Prove that a tree with two or more vertices contains at least two leaves (pendant vertices).*

▶ Consider a tree T with n vertices, where $n \geq 2$. Then it has $n - 1$ edges. Therefore (by the handshaking property), the sum of the degrees of the n vertices must be equal to $2(n-1)$. Thus, if $d_1, d_2, \ldots, d_n$ are the degrees of vertices of T, we have

$$d_1 + d_2 + \cdots + d_n = 2(n - 1) = 2n - 2.$$

If each of $d_1, d_2, \ldots, d_n$ is ≥ 2, then their sum must be at least $2n$. Since this is not true, at least one of the d's is less than 2. Thus, there is a d which is equal to 1. (Since T is connected, no d can be zero). Without loss of generality, let us take this to be d_1. Then

$$d_2 + d_3 + \cdots + d_n = (2n - 2) - 1 = 2n - 3.$$

This is possible only if at least one of $d_2, d_3, \ldots, d_n$ is equal to 1. So, there is at least one more d which is equal to 1. Thus, in T, there are at least two vertices with degree 1; that is, there are at least two pendant vertices (leaves). ■

Example 6 *Show that if a tree has exactly two pendant vertices, the degree of every non-pendant vertex is two.*

► Let n be the number of vertices in a tree T. Suppose it has exactly two pendant vertices (so that their degrees are 1 each). Let $d_1, d_2, \ldots, d_{n-2}$ be the degrees of the other (non-pendant) vertices. Then, since T has exactly $n-1$ edges, we have

$$1 + 1 + d_1 + d_2 + \cdots + d_{n-2} = 2(n-1)$$

or $$d_1 + d_2 + \cdots + d_{n-2} = 2n - 4 = 2(n-2)$$

The left hand side of this condition has $n-2$ terms, and none of these is one or zero. Therefore, this condition holds only if each of the d_is is equal to two. ■

Example 7 *Show that, in a tree, if the degree of every non-pendant vertex is 3, the number of vertices in the tree is an even number.*

► Let n be the number of vertices in a tree T. Of these, let k be the number of pendant vertices. Then, if each non-pendant vertex is of degree 3, the sum of the degrees of vertices is $k+3(n-k)$. This must be equal to $2(n-1)$, by the handshaking property. Thus,

$$k + 3(n-k) = 2(n-1), \quad \text{or} \quad n = 2(k-1).$$

This shows that n is an even number. ■

Example 8 *Suppose that a tree T has two vertices of degree 2, four vertices of degree 3 and three vertices of degree 4. Find the number of pendant vertices in T.*

► Let N be the number of pendant vertices in T. It is given that T has two vertices of degree 2, four vertices of degree 3 and three vertices of degree 4. Therefore,

Total number of vertices $= N + 2 + 4 + 3 = N + 9$, and

Sum of the degrees of vertices $= (N \times 1) + (2 \times 2) + (4 \times 3) + (3 \times 4) = N + 28$.

Since T has $N + 9$ vertices, it has $N + 9 - 1 = N + 8$ edges.

Therefore, by handshaking property, we have $N + 28 = 2(N + 8)$ which gives $N = 12$.

Thus, the given tree has 12 pendant vertices. ■

Example 9 *If a tree T has four vertices of degree 2, one vertex of degree 3, two vertices of degree 4 and one vertex of degree 5, find the number of leaves in T.*

► Let N be the number of leaves (pendant vertices) in T. Then:

Total No. of vertices $= N + 4 + 1 + 2 + 1 = N + 8$

Sum of the degrees of vertices $= (N \times 1) + (4 \times 2) + (1 \times 3) + (2 \times 4) + (1 \times 5) = N + 24$.

Since T has $N + 8$ vertices, it has $N + 7$ edges. Therefore, by hand shanking property.

$$N + 24 = 2(N + 7) \quad \text{which gives} \quad N = 10.$$

Thus, the given tree has 10 leaves. ■

Example 10 *Suppose that a tree T has N_1 vertices of degree 1, N_2 vertices of degree 2, N_3 vertices of degree 3, ..., N_k vertices of degree k. Prove that*

$$N_1 = 2 + N_3 + 2N_4 + 3N_5 + \cdots + (k-2)N_k.$$

▸ From what is given, we note that, in T,

Total number of vertices $= N_1 + N_2 + \cdots + N_k$, and

Sum of the degrees of vertices $= N_1 + 2N_2 + 3N_3 + 4N_4 + 5N_5 + \cdots + kN_k$.

Therefore, the total number of edges in T is $N_1 + N_2 + \cdots + N_k - 1$, and the handshaking property gives

$$N_1 + 2N_2 + 3N_3 + 4N_4 + 5N_5 + \cdots + kN_k = 2(N_1 + N_2 + \cdots + N_k - 1)$$

Rearranging terms, this gives

$$N_3 + 2N_4 + 3N_5 + \cdots + (k-2)N_k = N_1 - 2.$$

This is the required result. ■

Exercises

1. Prove that a graph with n vertices is a tree if and only if it has $n-1$ edges and no cycles.
2. Show that a tree with exactly two leaves must be a path.
3. If a tree has four vertices of degree 3, two vertices of degree 4 and one vertex of degree 5, show that it should have 10 pendant vertices.
4. In a tree with 14 pendant vertices, the degree of every non-pendant vertex is either 4 or 5. Show that the tree has 3 vertices of degree 4 and 2 vertices of degree 5.
5. In a tree with $r + s$ vertices, if r vertices are pendant vertices and s vertices have degree 4 each, prove that $2s = r - 2$.
6. Prove that, in a tree, the minimum number of pendant vertices is equal to the maximum degree of a vertex.
7. Prove that every tree with two or more vertices is 2-chromatic.
8. Prove that in a tree with $n(\geq 2)$ vertices the number of distinct paths is ${}^nC_2$.
9. Let T be a tree with n vertices, where $n \geq 4$, and v be a vertex of maximum degree in T. Prove that T is a path if and only if $d(v) = 2$.
10. Let T be a tree with n vertices, $n \geq 3$. Show that there is a vertex v of T with degree at least two such that every vertex adjacent to v, except possibly one, is a pendant vertex.

10.2 Rooted Trees

Let D be a *directed graph* and G be its underlying graph. We say that D is a ***directed tree*** whenever G is a tree. Thus, a directed tree is a directed graph whose underlying graph is a tree.

A directed tree T is called a ***rooted tree*** if (i) T contains a unique vertex, called the *root*, whose in-degree is equal to 0, and (ii) the in-degrees of all other vertices of T are equal to 1.

Figures 10.5 and 10.6 depict two directed trees. The first of these is not a rooted tree whereas the second is a rooted tree.

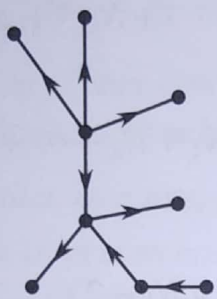


Figure 10.5

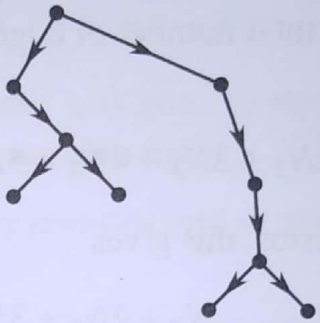


Figure 10.6

In a rooted tree, we denote the root by r and draw the (diagram of the) tree downward from an upper level to a lower level, so that the arrows can be dropped. Then the root r will be at the uppermost level (zeroth level) and all other vertices will be at lower levels.

A vertex v (other than the root r) of a rooted tree is said to be at the *k-th level* or has ***level number k*** if the path from r to v is of length k. If v_1 and v_2 are two vertices such that v_1 has a lower level number than v_2 and there is a path from v_1 to v_2, then we say that v_1 is an ***ancestor*** of v_2, or that v_2 is a ***descendant*** of v_1. In particular, if v_1 and v_2 are such that v_1 has a lower level number than v_2 and there is an edge (- directed edge, actually) from v_1 to v_2, then v_1 is called the ***parent*** of v_2, or v_2 is called the ***child*** of v_1. Two vertices with a common parent are referred to as ***siblings***.

In a rooted tree a vertex whose out-degree is 0 is called a ***leaf*** and a vertex which is not a leaf is called an ***internal vertex***.

For example, suppose we redraw the directed tree of Figure 10.6 as shown below without arrows (- which are understood) and with vertices labeled.

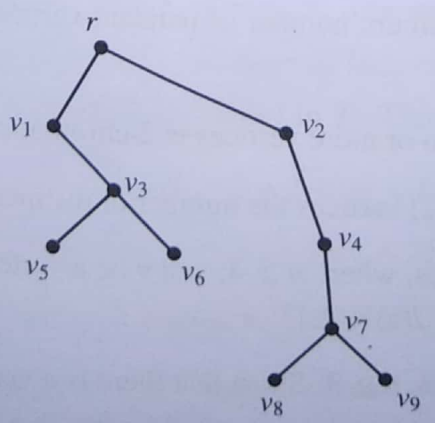


Figure 10.7

In this rooted tree, we note that

(1) v_1 and v_2 are at the *first level*, v_3, v_4 are at the *second level*, v_5, v_6, v_7 are at the *third level*, and v_8 and v_9 are at the *fourth level*.

(2) v_1 is the *ancestor* of v_3, v_5, v_6 (or v_3, v_5, v_6 are the *descendants* of v_1), and v_2 is the *ancestor* of v_4, v_7, v_8, v_9 (or v_4, v_7, v_8, v_9 are the *descendants* of v_2).

(3) v_1 is the *parent* of v_3 (or v_3 is a *child* of v_1).

(4) v_5 and v_6 are *siblings*, and v_8 and v_9 are *siblings*.

(5) v_5, v_6, v_8, v_9 are *leaves*, and all other vertices are *internal vertices*.

m-ary Tree

A rooted tree T is called an *m-ary tree* if every internal vertex of T is of out-degree $\leq m$; that is if every internal vertex of T has at most m children.

A rooted tree T is called a ***complete m-ary tree*** if every internal vertex of T is of out-degree m; that is if every internal vertex of T has exactly m children.

Binary Tree

An m-ary tree for which $m = 2$ is called a *binary tree*.

In other words, a rooted tree T is called a *binary tree* if every vertex of T is of out-degree ≤ 2; that is if every vertex has at most two children.

A complete m-ary tree for which $m = 2$ is called a ***complete binary tree***.

In other words, a rooted tree T is called a complete binary tree if every internal vertex of T is of out-degree 2; that is if every internal vertex has exactly two children.

The rooted tree shown in Figure 10.7 is a binary tree; it is not a complete binary tree. The rooted tree shown in Figure 10.8 is a complete binary tree.

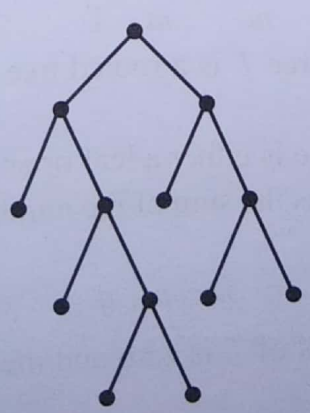


Figure 10.8

Balanced Tree

If T is a rooted tree and h is the largest level number achieved by a leaf of T, then T is said to have *height* h. A rooted tree of height h is said to be *balanced* if the level number of every leaf is h or $h-1$.

The tree shown in Figure 10.7 is of height 4 and is balanced too. The tree shown in Figure 10.8 is of height 4 but is not balanced.

Full binary Tree

Let T be a complete binary tree of height h. Then T is called a ***full binary tree*** if all the leaves in T are at level h.

The complete binary tree shown in Figure 10.8 is not a full binary tree. The tree shown in Figure 10.9 is a full binary tree.

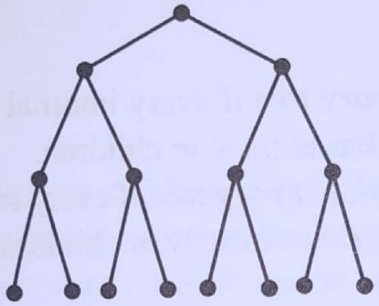


Figure 10.9

Example 1 *Let T be a complete m-ary tree of order n with p leaves and q internal vertices. Prove the following:*

$$\text{(a)} \quad n = mq + 1 = \frac{mp - 1}{m - 1}$$

$$\text{(b)} \quad p = (m-1)q + 1 = \frac{(m-1)n + 1}{m}$$

$$\text{(c)} \quad q = \frac{n-1}{m} = \frac{p-1}{m-1}$$

► We recall that a complete m-ary tree T is a rooted tree in which every internal vertex is of out-degree m.

Since every vertex in a rooted tree is either a leaf or an internal vertex, the total number of vertices in T (that is, the order of T) is the sum of the number of leaves in T and the number of internal vertices in T. Thus, we have

$$n = p + q \tag{i}$$

Since the out-degree of every leaf of T is zero and the out-degree of every internal vertex of T is m,

The sum of the out-degrees of vertices of T

$$= (p \times 0) + (q \times m) = qm \tag{ii}$$

Since in a rooted tree of order n, the in-degree of the root is zero and the in-degrees of the remaining $(n-1)$ vertices are 1 each,

The sum of the in-degrees of vertices of T

$$= (1 \times 0) + ((n-1) \times 1) = n - 1 \qquad \text{(iii)}$$

By the First Theorem of Digraph Theory*, the two sums given by (ii) and (iii) must be equal. Thus, $qm = n - 1$, or

$$n = qm + 1, \qquad \text{(iv)}$$

or

$$q = \frac{n-1}{m} \qquad \text{(v)}$$

Using expression (iv) in expression (i), we get $qm + 1 = p + q$, which gives

$$p = (m-1)q + 1 \qquad \text{(vi)}$$

This result readily yields

$$q = \frac{p-1}{m-1} \qquad \text{(vii)}$$

Putting this into expression (iv) we get

$$n = m\left(\frac{p-1}{m-1}\right) + 1 = \frac{m(p-1) + (m-1)}{m-1} = \frac{mp-1}{m-1} \qquad \text{(viii)}$$

Lastly, putting expression (vii) into expression (i), we get

$$n = p + \frac{p-1}{m-1}, \quad \text{or} \quad (m-1)n = p(m-1) + p - 1 = pm - 1$$

or

$$p = \frac{(m-1)n + 1}{m} \qquad \text{(ix)}$$

Thus, all the required results are proved. ■

Remark: In the case of a *complete binary tree* (for which $m = 2$), the results of the above Example become

(a) $n = 2q + 1 = 2p - 1$

(b) $p = q + 1 = \frac{1}{2}(n+1)$

(c) $q = \frac{1}{2}(n-1) = p - 1$

*See Section 9.1.

Example 2 *Find the number of vertices and the number of leaves in a complete binary tree having 10 internal vertices.*

► Let n be the number of vertices in the tree being considered. Since this tree has $q = 10$ internal vertices, the expression $n = 2q + 1$ yields $n = 21$. Consequently, the number of leaves is $p = n - q = 11$. ■

Example 3 (a) *Find the number of internal vertices in a complete 5-ary tree with 817 leaves.*

(b) *Find the number of leaves in a complete 6-ary tree of order 733.*

► (a) From Example 1, we have

$$q = \frac{p-1}{m-1}$$

For $m = 5$ and $p = 817$, this gives $q = 204$. Thus, the given tree has 204 internal vertices.

(b) From Example 1, we have

$$p = \frac{(m-1)n + 1}{m}$$

For $m = 6$ and $n = 733$, this gives $p = 611$. Thus, the given tree has 611 leaves. ■

Example 4 *The computer laboratory of a school has 10 computers that are to be connected to a wall socket that has 2 outlets. Connections are made by using extension cords that have 2 outlets each. Find the least number of cords needed to get these computers set up for use.*

► Here, the wall socket may be regarded as the root of a complete binary tree having the computers as its leaves and the internal vertices, other than the root, as extension cords.

Then the number of leaves in the tree is $p = 10$. Therefore, the number of internal vertices in the tree is $q = p - 1 = 10 - 1 = 9$. Accordingly, the number of extension cords needed (namely the number of internal vertices minus the root) is $q - 1 = 8$. (See Figure 10.10 for illustration).

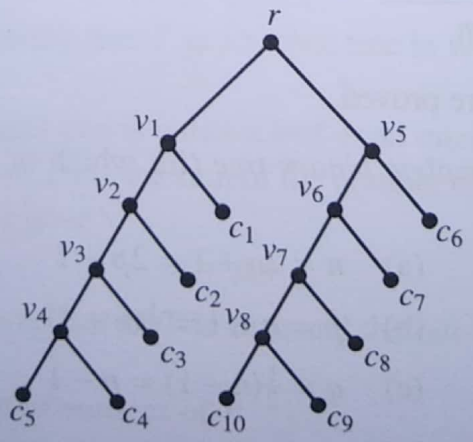


Figure 10.10 ■

Example 5 *A class room contains 25 microcomputers that must be connected to a wall socket that has 4 outlets. Connections are made by using extension cords that have 4 outlets each. Find the least number of cords needed to get this computer set up for the class.*

▶ Here, the wall socket may be regarded as the root of a complete 4-ary tree with the computers as its leaves and the internal vertices other than the root as extension cords.

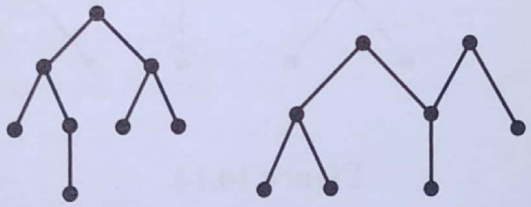
Thus, here, $m = 4$ and $p = 25$. Therefore, the number of internal vertices is (see Example 1)

$$q = \frac{p-1}{m-1} = \frac{24}{3} = 8.$$

Consequently, the number of extension cords needed (namely the number of internal vertices minus the root) is $q - 1 = 7$. ■

Exercises

1. Which of the following is a rooted tree?



(a) (b)

Figure 10.11

2. For the rooted tree shown below, find the levels of the vertices A, H, F, M.

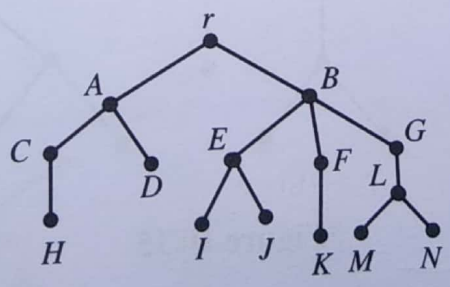


Figure 10.12

3. In the rooted tree shown in Figure 10.12, identify

(i) all ancestors of L, (ii) all descendants of A,
(iii) the children of B, (iv) the parent and siblings of C.

4. In the tree shown in Figure 10.13, identify the following:

   (i) vertices which are leaves (ii) the root (iii) the parent of g

   (iv) descendants of c (v) siblings of s (vi) vertices having the level number 4.

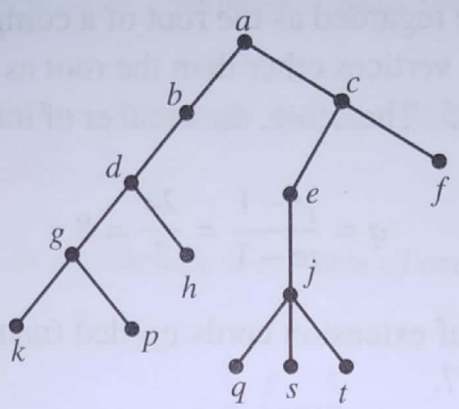


Figure 10.13

5. Which of the following is a binary tree? complete binary tree?

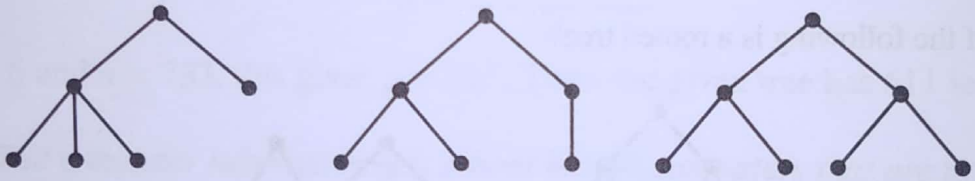


Figure 10.14

6. Which of the following is a balanced tree? full binary tree?

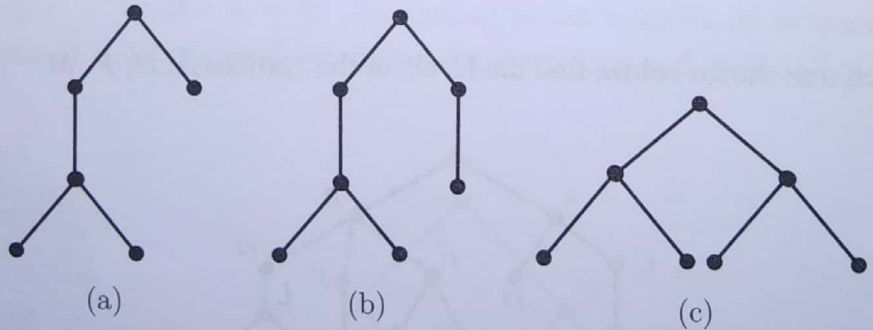


Figure 10.15

7. A complete binary tree has 20 leaves. How many vertices does it have?
8. Find the number of leaves in a complete binary tree if it has 29 vertices.
9. In a single-elimination singles tennis tournament, a player is eliminated after a single loss. If 32 players compete in the tournament, how many matches must be played to determine the number-

one player? (Hint : Consider a complete binary tree with the number-one player as the root, matches as internal vertices and the competitors as leaves).

10. A complete 3-ary tree T has 34 internal vertices. How many leaves does T have?

Answers

1. (a) rooted tree, (b) not a rooted tree. **2.** $A:1, H:3, F:2, M:4$.

3. (i) r, B, G (ii) C, D, H (iii) E, F, G (iv) parent:A, sibling:D

4. (i) k, p, h, q, s, t, f (ii) a (iii) d (iv) e, f, j, q, s, t (v) q, t (vi) k, p, q, s, t

5. (a) not a binary tree (b) binary tree, but not complete (c) complete binary tree.

6. (a) not a balanced tree (b) balanced tree, but not full (c) full binary tree.

7. 39 **8.** 15 **9.** 31 **10.** 69

10.2.1 Sorting

Suppose we wish to sort (– rearrange/reorganize) a given list of n integers in nondecreasing order. The most common (and the easiest) way of carrying out this sorting consists of two parts. In the first part, we recursively split the given list and all subsequent lists in half (or as close as possible to half) until each sublist contains a single element. In the second part, we merge the sublists in nondecreasing order until the original n integers have been sorted. The splitting and merging process is done by the use of balanced complete trees. This method of sorting a list is known as ***Merge Sort***.

Example 1 *Using the merge-sort method, sort the list* $7, 3, 8, 4, 5, 10, 6, 2, 9$.

▶ First, we recursively split the given list and all subsequent lists in half or as close as possible to half until each sublist contains a single element. This splitting process is represented in the tree shown in Figure 10.16(a).

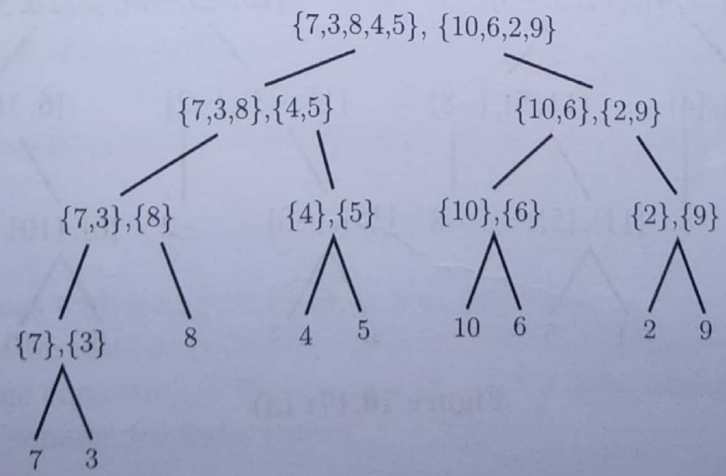


Figure 10.16: (a)

Now we merge the sublists in nondecreasing order until the items in the original list have been sorted. This merging process is represented in Figure 10.16(b).

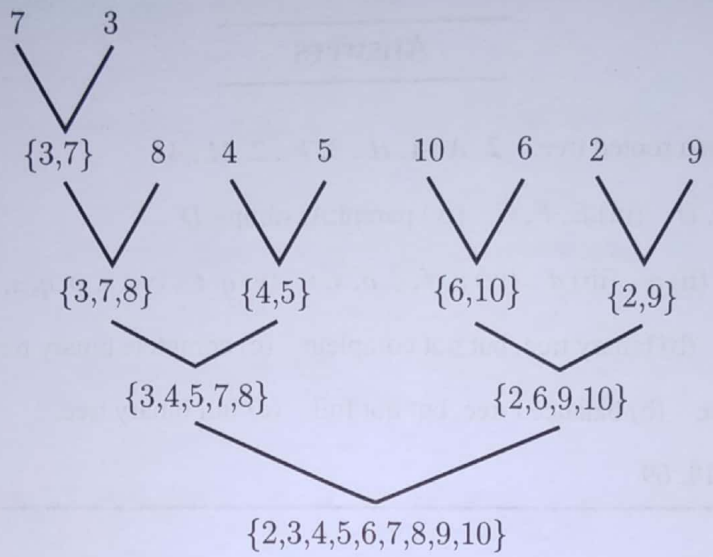


Figure 10.16: (b)

Thus, the sorted form of the given list is 2, 3, 4, 5, 6, 7, 8, 9, 10. ■

Example 2 *Apply merge-sort to the list* $-1, 7, 4, 11, 5, -8, 15, -3, -2, 6, 10, 3$.

► The given list is first split into sublists as depicted in the tree shown in Figure 10.17(a).

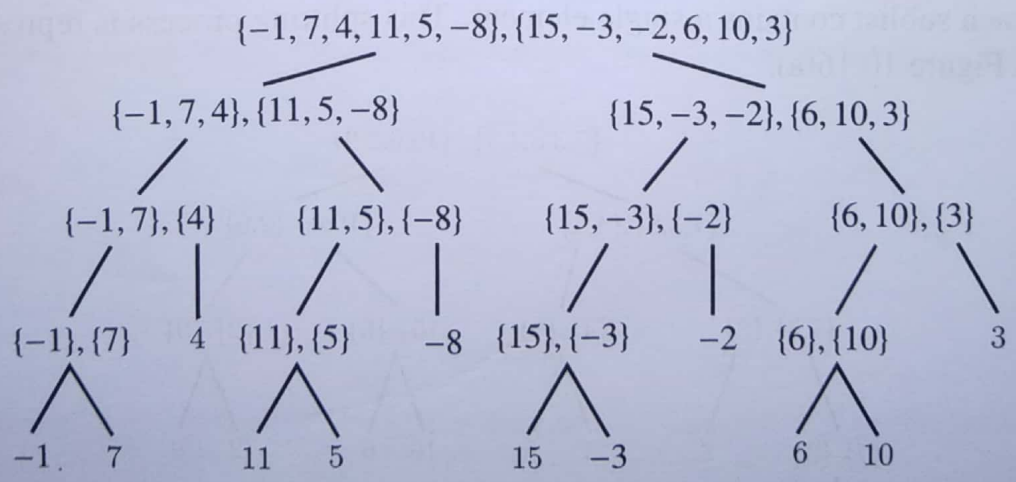


Figure 10.17: (a)

The sublists obtained above are now merged as depicted in Figure 10.17(b).

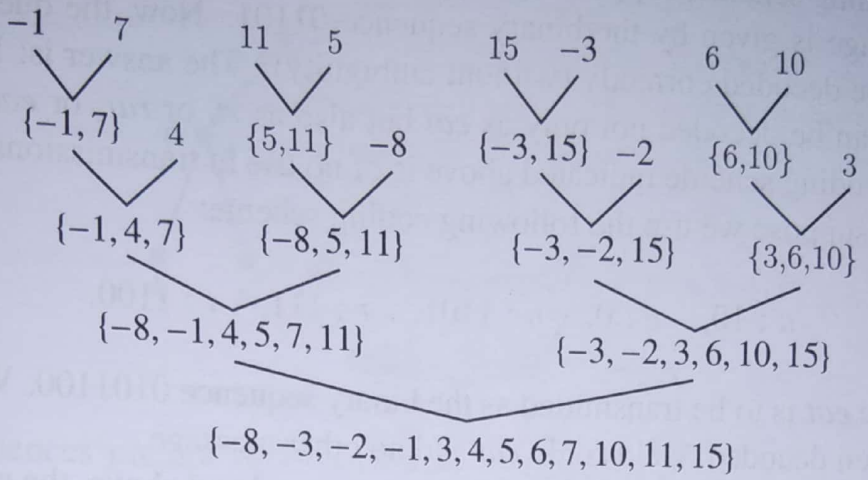


Figure 10.17: (b)

Thus, the sorted-out version of the given list is −8, −3, −2, −1, 3, 4, 5, 6, 7, 10, 11, 15. ■

Exercises

Apply the merge-sort to each of the following lists:

1. 6, 2, 7, 3, 4, 9, 5, 1, 8.
2. −1, 0, 2, −2, 3, 6, −3, 5, 1, 4.
3. −9, 6, 5, −3, 4, 2, −7, 6, −5, 10, −11, 0, 1.

10.3 Prefix codes and Weighted trees

Recall that a sequence is a set whose elements are listed (arranged) in order as the first element, second element, third element, and so on. The number of elements contained in a sequence is called its *length*.

A sequence consisting of only 0 and 1 is called a ***binary sequence*** or a ***binary string***.

Thus,

$$01, 001, 101, 11001, 1000100$$

are all binary sequences with lengths 2, 3, 3, 5, 7, respectively.

Binary sequences are used as codes for messages sent through transmitting channels.

Suppose a message consisting of the letters a, e, n, r, t is to be transmitted. Suppose we use the following coding scheme for these letters:

$$a : 1, \quad e : 0, \quad n : 10, \quad r : 01, \quad t = 101.$$

Under this coding scheme, suppose the message *eat* is to be transmitted. Then, the coded form of the message is given by the binary sequence 01101. Now, the question is: can this binary sequence be decoded correctly (without ambiguity)? The answer is: No!. Because the sequence 01101 can be decoded not only as *eat* but also as *rt*, or *rar*, or *eaar* (for example). Accordingly, the coding scheme indicated above is of no use in transmissions.

Alternatively, suppose we use the following coding scheme:

$$a : 10, \quad e : 0, \quad n : 1101, \quad r : 111, \quad t : 1100.$$

Then, the message *eat* is to be transmitted as the binary sequence 0101100. We can check that this sequence, when decoded, yields only *eat* and no other message.

We observe that in the first of the coding schemes considered above, the code 1 is assigned to a, the code 10 is assigned to n and the code 101 is assigned to t and that the code assigned to t contains the codes assigned to a and n as prefixes. This is why there arises ambiguity in decoding. In the second of the coding schemes, the code of any letter is *not a prefix* of the code of any other letter. This is why there cannot be ambiguity in decoding. Coding schemes of this type are of interest in coding theory and a brief account of the same is given in the following paragraphs.

Prefix Codes

Let P be a set of binary sequences that represent a set of symbols. Then P is called a *prefix code*[†] if no sequence in P is the prefix of any other sequence in P.

For example, the sets

$$P_1 = \{10, 0, 1101, 111, 1100\}, \quad \text{and} \quad P_2 = \{000, 001, 01, 10, 11\}$$

are prefix codes, whereas the sets

$$A_1 = \{01, 0, 101, 10, 1\} \quad \text{and} \quad A_2 = \{1, 00, 01, 000, 0001\}$$

are *not* prefix codes.

Prefix codes can be represented by binary trees as illustrated below.

Consider the prefix code

$$P_2 = \{000, 001, 01, 10, 11\}$$

indicated above. In this code, the longest sequence has length 3. Keeping this in mind, let us construct a *full binary tree* of height 3, and assign the symbol 0 to every edge that is directed towards the child in the left from its parent vertex and 1 to every edge that is directed towards the child in the right, as shown in Figure 10.18*.

[†]prefix codes are also called *Huffman codes*, in honor of their inventor D. A. Huffman.

*Assigning 0 and 1 to edges of a tree according to the scheme indicated and illustrated here is conventional in nature and is called *labeling*. The tree got after labeling is called the *labeled tree*.

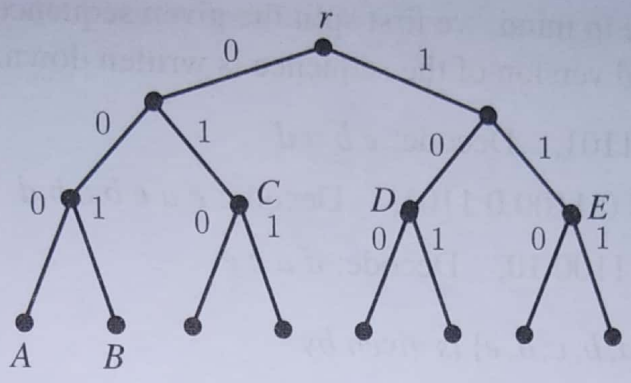


Figure 10.18

The five sequences present in P_2 can now be identified with five vertices of the above tree. We note that the vertex marked A can be reached from the root r through the three edges labeled 0, 0, 0. Accordingly, the vertex A can be assigned the sequence 000. Similarly, the vertex marked B can be assigned the sequence 001, the vertex marked C can be assigned the sequence 01, the vertex marked D can be assigned the sequence 10, and the vertex marked E can be assigned the sequence 11. Thus, all the five sequences present in P_2 can be assigned to the five vertices, marked A, B, C, D, E, of the tree being considered.

The subtree extracted from the full binary tree of Figure 10.18 that contains the root r and the vertices A, B, C, D, E is shown in Figure 10.19. This subtree represents the prefix code given by the set P_2.

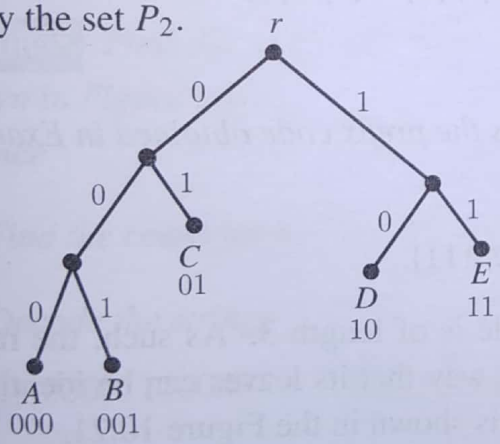


Figure 10.19

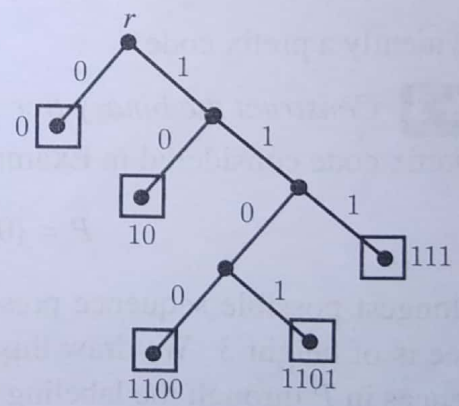


Figure 10.20

The binary tree representing the prefix code given by the set P_1 (indicated earlier) is shown in Figure 10.20. The vertices of this tree to which the sequences in P_1 can be assigned are boxed.

Example 1 *Consider the prefix code:*

$$a : 111, \quad b : 0, \quad c : 1100, \quad d : 1101, \quad e : 10.$$

Using this code, decode the following sequences:

(i) 1001111101, (ii) 10111100110001101, (iii) 1101111110010.

► Keeping the given code in mind, we first split the given sequences into appropriate number of parts. Then, the decoded version of the sequence is written down.

(i) Splitting: 10 0 111 1101, Decode: $e\ b\ a\ d$

(ii) Splitting: 10 111 10 0 1100 0 1101, Decode: $e\ a\ e\ b\ c\ b\ d$

(iii) Splitting: 1101 111 1100 10, Decode: $d\ a\ c\ e$ ■

Example 2 *A code for* $\{a, b, c, d, e\}$ *is given by*

$$a : 00,\quad b : 01,\quad c : 101,\quad d : x10,\quad e : yz1,$$

where $x, y, z \in \{0, 1\}$. *Determine* x, y *and* z *so that the given code is a prefix code.*

► If we set $x = 0$, then the code for d reads 010 which contains the code 01 for b as a prefix. Therefore, we cannot take $x = 0$. This implies that $x = 1$. Then, $d : 110$.

If we set $y = 0$, $z = 0$ then the code for e reads 001, which contains the code for a as a prefix. If we set $y = 0$, $z = 1$, then the code for e reads 011, which contains the code for b as a prefix. If we set $y = 1$, $z = 0$, then the code for e becomes identical with that for c. Hence, we take $y = 1$, $z = 1$ so that $e : 111$.

With the choices of x, y, z as obtained above the given code reads

$$a : 00,\quad b : 01,\quad c : 101,\quad d : 110,\quad e : 111.$$

This is evidently a prefix code. ■

Example 3 *Construct the binary tree that represents the prefix code obtained in Example* 2.

► The prefix code considered in Example 2 is

$$P = \{00, 01, 101, 110, 111\}.$$

The longest possible sequence present in this code is of length 3. As such, the required binary tree is of height 3. We draw this tree in such a way that its leaves can be identified by the sequences in P through the labeling rule. The tree is shown in the Figure 10.21.

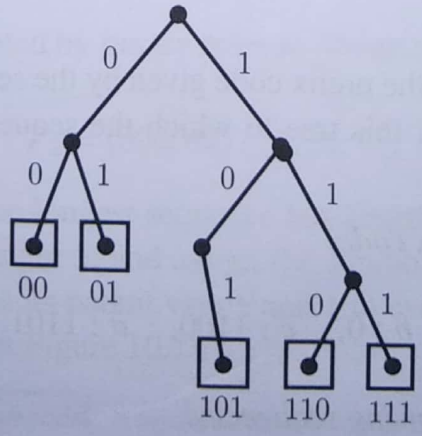


Figure 10.21 ■

Example 4 *Obtain the prefix code represented by the following labeled complete binary tree:*

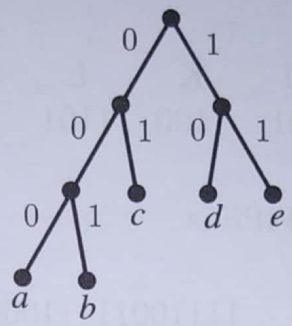


Figure 10.22

▶ The leaves of the given tree are represented by the symbols a, b, c, d, e. These leaves are identified by the sequences as indicated in the following Table:

Leaf:	a	b	c	d	e
Sequence:	000	001	01	10	11

This table determines the required prefix code as

$$P = \{000, 001, 01, 10, 11\}$$

■

Example 5 *Find the prefix codes for the letters* B, E, I, K, L, T, P, S *if the coding scheme is as shown in Figure* 10.23.

Hence

(a) *Find the codes for the words* PIPE *and* BEST

(b) *Decode the strings*

(i) 000011100001 **(ii)** 11111111101101011110

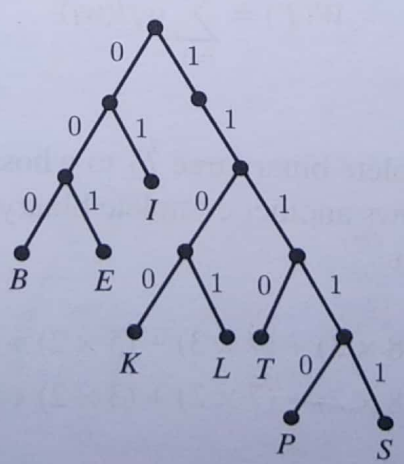


Figure 10.23

▶ The given letters are the leaves of the given tree. The codes for these leaves are indicated below.

Leaf :	B	E	I	K	L	T	P	S
Code :	000	001	01	1100	1101	1110	11110	11111

(a) It follows that the code for the PIPE is

$$111100111110001$$

and the code for the word BEST is

$$000001111111110.$$

(b) Keeping the codes obtained above in mind, we first split the given strings as shown below:

(i) 000 01 1100 001 **(ii)** 11111 11110 1101 01 1110

It now follows that the decoded versions of the these strings are BIKE and SPLIT respectively. ■

Weighted Trees

Consider a set of n positive integers $w_1, w_2, \ldots, w_n$, where $w_1 \leq w_2 \leq \ldots \leq w_n$. Suppose we assign these integers to the n leaves of a complete binary tree $T = (V, E)$ in any one-to-one manner. The resulting tree is called a *complete, weighted, binary tree* with $w_1, w_2, \ldots w_n$ as *weights*. If $l(w_i)$ is the level number of the leaf of T to which the weight w_i is assigned, then $W(T)$ defined by

$$W(T) = \sum_{i=1}^{n} w_i \, l(w_i)$$

is called the ***weight*** of the tree T.

Figure 10.24(a) shows a complete binary tree T_1 to whose leaves the weights 3, 5, 7, 8 are assigned, and Figure 10.24(b) shows another complete binary tree T_2 to whose leaves the same weights are assigned. We find that

$$W(T_1) = (8 \times 3) + (7 \times 3) + (5 \times 2) + (3 \times 1) = 58,$$

and

$$W(T_2) = (8 \times 2) + (7 \times 2) + (3 \times 2) + (5 \times 2) = 46.$$

This illustrates the fact that for a given set of weights, the value of $W(T)$ depends upon the tree T chosen.

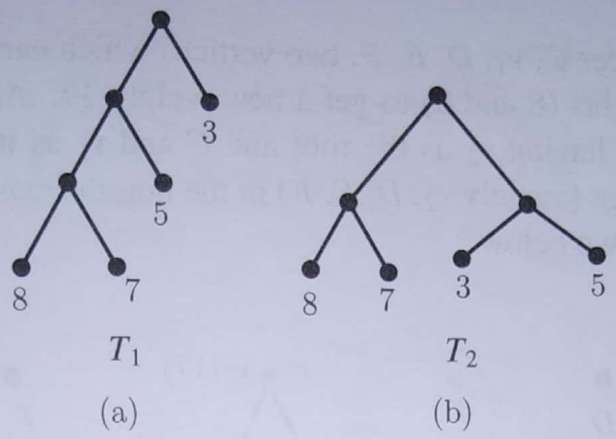


Figure 10.24

Optimal Tree

Given a set of weights, suppose we consider the set of all complete binary trees to whose leaves these weights are assigned. A tree in this set which carries the minimum weight is called an ***optimal tree*** for the weights. For a given set of weights, there can be more than one optimal tree.

The construction of an optimal tree for a given set of weights is illustrated below.

Let $\{4, 15, 25, 5, 8, 16\}$ be a set of six weights. Let us first arrange these six weights in non-decreasing order and assign them to six isolated vertices A, B, C, D, E, F as shown in Figure 10.25(i). The weights are indicated in brackets.

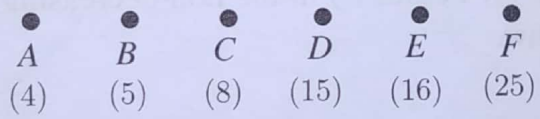


Figure 10.25: (i)

We note that the vertices A and B carry the smallest weights, 4 and 5. Add these weights to get the weight 9 and assign it to a new vertex v_1. Draw a tree having v_1 as the root and A and B as its children. Rearrange the vertices present at this stage (namely the new vertex v_1 and the old vertices C, D, E, F) in the non-decreasing order of their weights. The resulting graph is shown in Figure 10.25(ii).

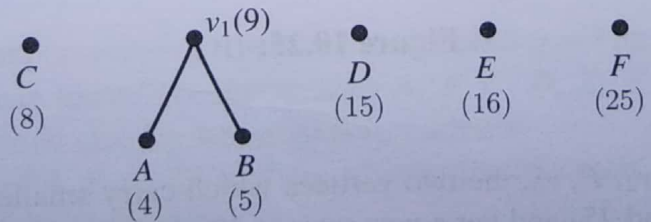


Figure 10.25: (ii)

Now, among the vertices C, v_1, D, E, F, two vertices which carry the smallest weights are C and v_1. Add their weights (8 and 9) to get a new weight 17. Assign this weight to a new vertex v_2 and draw a tree having v_2 as the root and C and v_1 as its children. Rearrange the vertices present at this stage (namely v_2, D, E, F) in the non-decreasing order of their weights. The resulting graph is shown below.

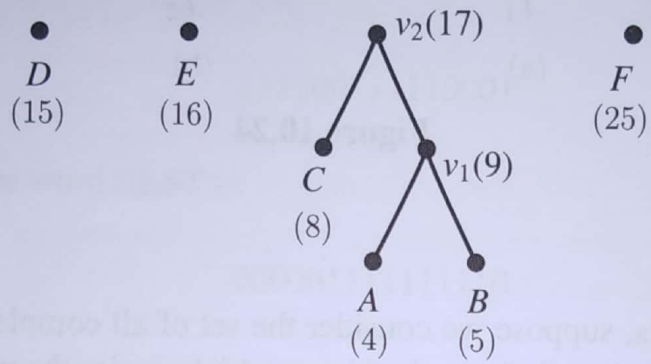


Figure 10.25: (iii)

Among the vertices D, E, v_2, F, the two vertices which carry minimum weights are D and E. Add their weights (15 and 16) to get a new weight 31. Assign this weight to a new vertex v_3 and draw a tree having v_3 as the root and D and E as its children. Rearrange the vertices present at this stage (namely v_2, v_3 and F) in the non-decreasing order of their weights. The resulting graph is shown below.

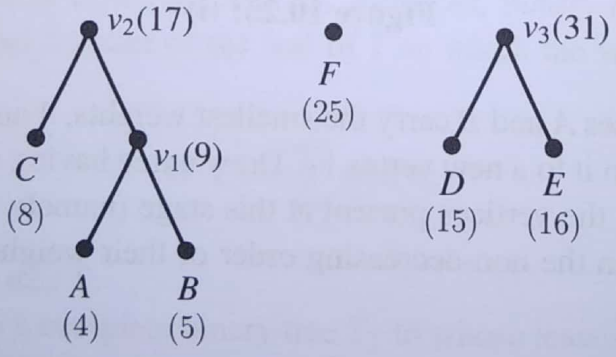


Figure 10.25: (iv)

Among the vertices v_2, F, v_3, the two vertices which carry smallest weights are v_2 and F. Add their weights (17 and 25) and get a new weight 42. Assign this weight to a new vertex v_4 and draw a tree having v_4 as the root and v_2 and F as its children. Rearrange the vertices v_3 and v_4 in the non-decreasing order of their weights. The resulting graph is shown below.

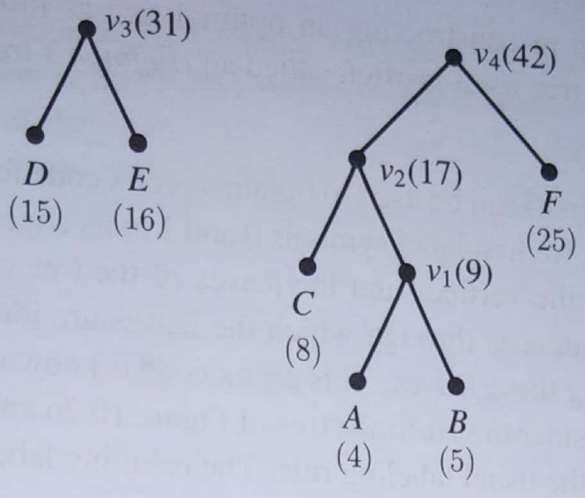


Figure 10.25: (v)

The disconnected graph shown above consists of two trees having v_3 and v_4 as roots. Add their weights (31 and 42) and get a new weight 73. Assign this weight to a new vertex r and draw a tree having r as the root and v_3 and v_4 as its children. This tree is shown in Figure 10.26.

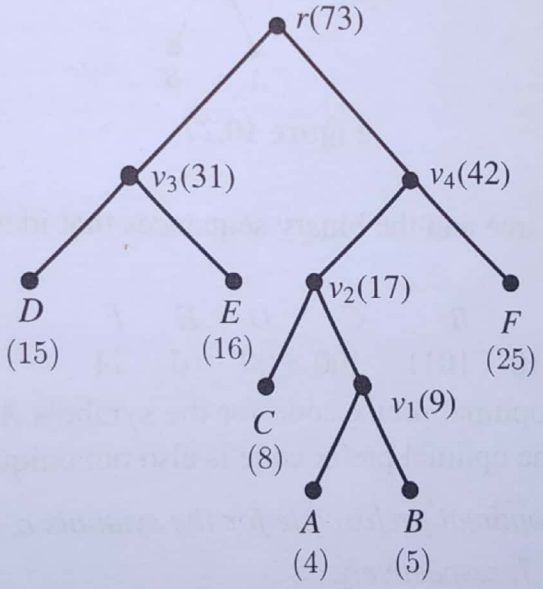


Figure 10.26

The tree obtained as explained above (that is the tree shown in Figure 10.26) is a complete, weighted binary tree whose leaves are the vertices A, B, C, D, E, F with which we started. This tree serves as an optimal tree for the weights considered.

Keeping the weights of A, B, C, D, E, F in mind and noting their levels in the above tree, we find that the weight of the optimal tree is

$$W(T) = (4 \times 4) + (5 \times 4) + (8 \times 3) + (15 \times 2) + (16 \times 2) + (25 \times 2) = 172.$$

The procedure adopted in constructing an optimal tree as illustrated above is known as *Huffman's procedure*. The tree itself is often called an *Huffman's tree*. This tree is *not unique*.*

Optimal Prefix Code

Huffman tree (optimal tree) can be used to obtain a prefix code for the symbols representing its leaves. For this purpose, we first label symbols 0 and 1 to its edges by the labeling procedure indicated earlier. Then all the vertices and the leaves of the tree can be identified by binary sequences. The binary sequences through which the leaves are identified yield a prefix code for the symbols representing these leaves. This prefix code is known as an ***optimal prefix code***.

For example, let us consider the Huffman tree of Figure 10.26 and assign the symbols 0 and 1 to its edges according to the usual labeling rule. The resulting labeled tree is shown below:

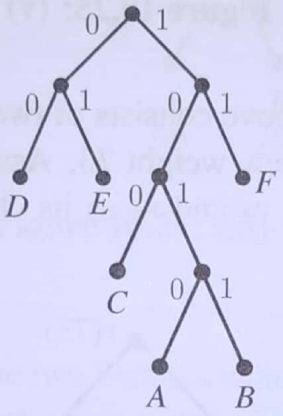


Figure 10.27

The leaves of the above tree and the binary sequences that identify these leaves are shown in the following Table:

Leaf:	A	B	C	D	E	F
Binary sequence:	1010	1011	100	00	01	11

This Table displays an optimal prefix code for the symbols A, B, C, D, E, F. Since the optimal tree is not unique, the optimal prefix code is also not unique.

Example 6 *Construct an optimal prefix code for the symbols a, o, q, u, y, z that occur with frequencies* 20, 28, 4, 17, 12, 7, *respectively.*

► Treating the given frequencies as the weights and the corresponding symbols as the isolated vertices, we first arrange the symbols such that their frequencies are in nondecreasing order. This is shown below:

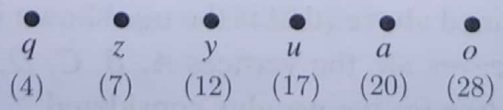


Figure 10.28: (i)

*This is because of multiple choices one may have in the rearrangement of vertices at different steps and the choice in selecting left and right subtrees.

Now, we construct an optimal tree having q, z, y, u, a, o as leaves by using the Huffman's procedure. The construction is done step-by-step and the graphs obtained in these steps are shown below in the order of their occurrence. (See Figures 10.28(ii)-(vi)). The final graph is labeled (using the labeling procedure) and depicted in Figure 10.29.

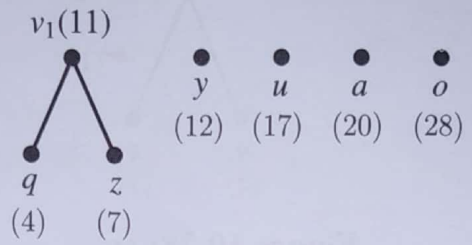


Figure 10.28: (ii)

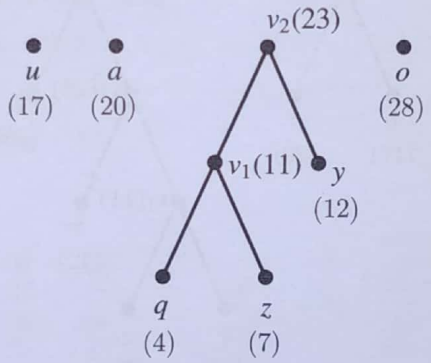


Figure 10.28: (iii)

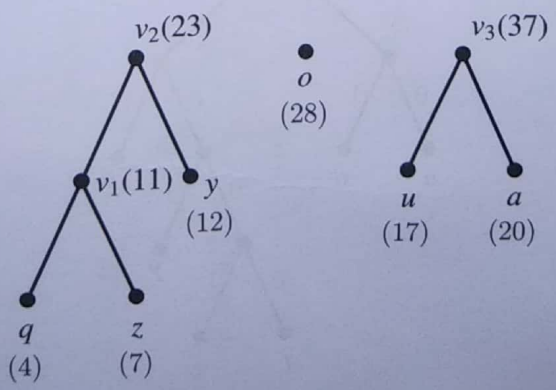


Figure 10.28: (iv)

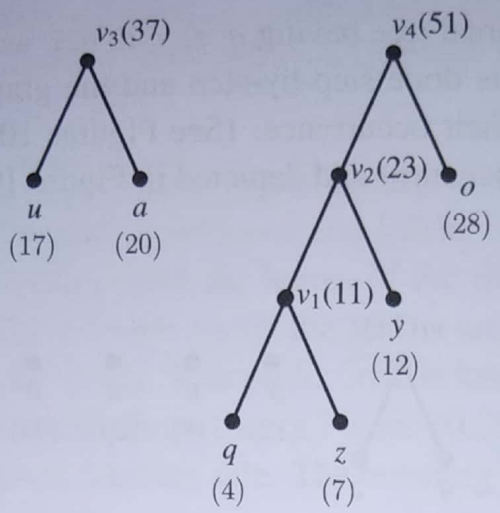


Figure 10.28: (v)

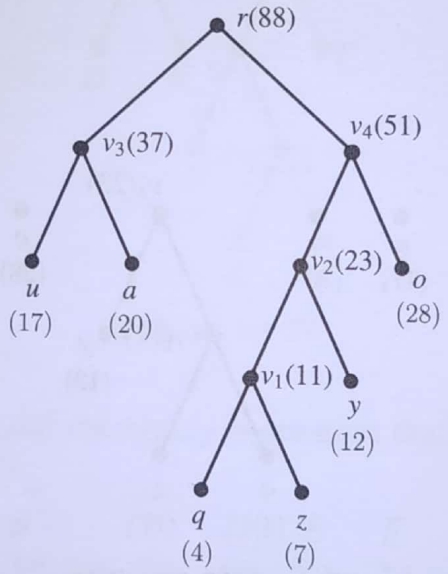


Figure 10.28: (vi)

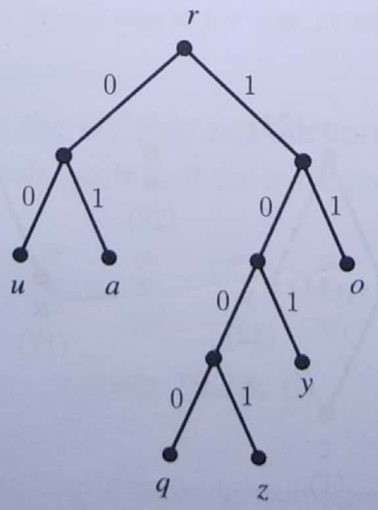


Figure 10.29

The tree shown in Figure 10.29 is an Huffmann tree (optimal tree) for the given data. From this graph, we obtain the following optimal prefix code for the given symbols:

$$a : 01, \quad o : 11, \quad q : 1000, \quad u : 00, \quad y : 101, \quad z : 1001. \qquad \blacksquare$$

Example 7 *Construct an optimal prefix code for the symbols A, B, C, D, E, F, G, H, I, J that occur with respective frequencies* 78, 16, 30, 35, 125, 31, 20, 50, 80, 3.

▶ First, we arrange the given symbols in the non-decreasing order of their weights (frequencies). Their representation as isolated vertices is shown below:

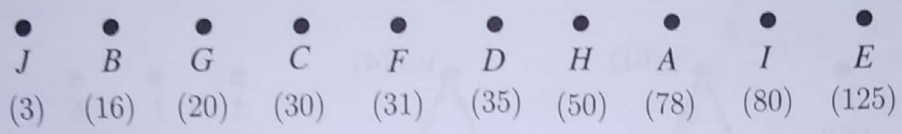


Figure 10.30: (i)

We now construct an optimal tree having $A, B, C, D, E, F, G, H, I, J$ as leaves by using the Huffman procedure. The construction is done step-by-step and the graphs obtained in these steps are shown below in Figures 10.30(ii)-(x) in the order of their occurrence. The final graph is labeled and depicted in Figure 10.31.

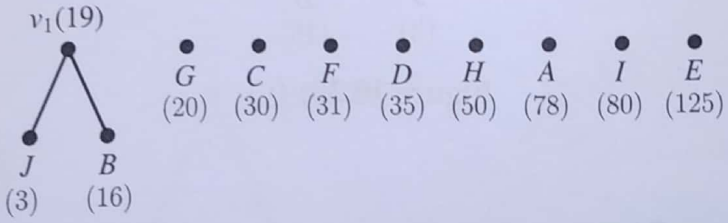


Figure 10.30: (ii)

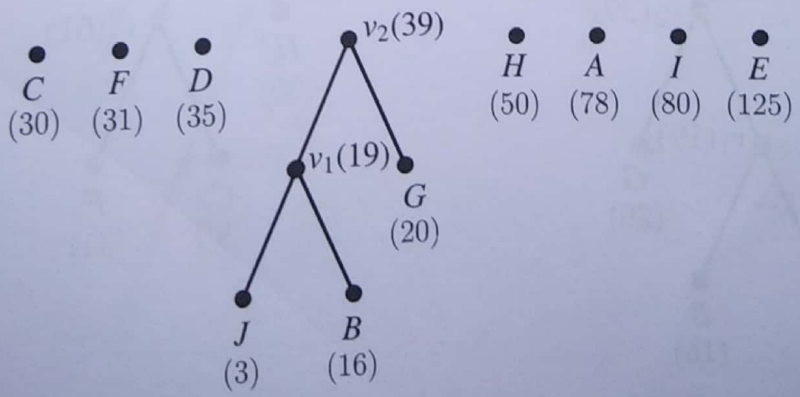


Figure 10.30: (iii)

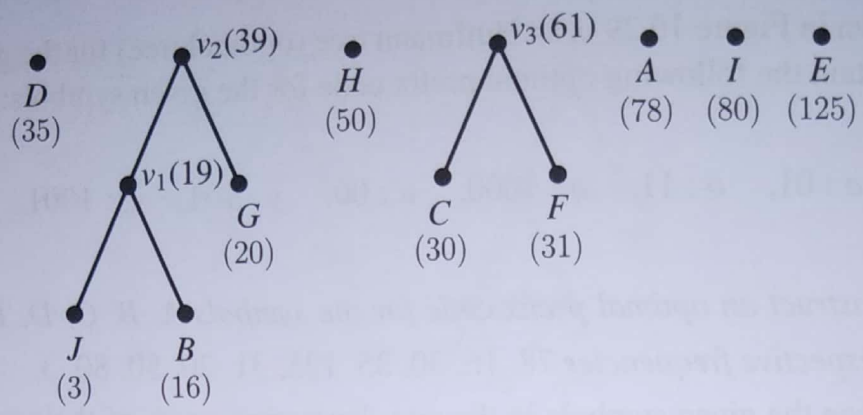


Figure 10.30: (iv)

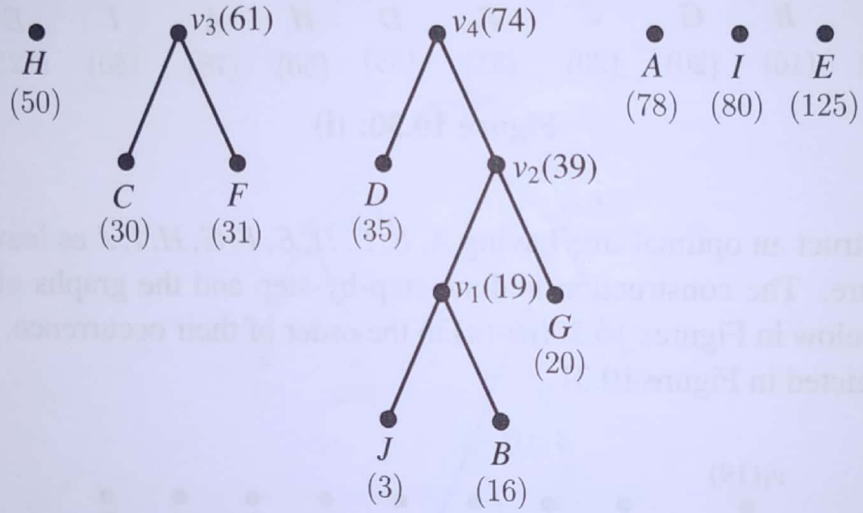


Figure 10.30: (v)

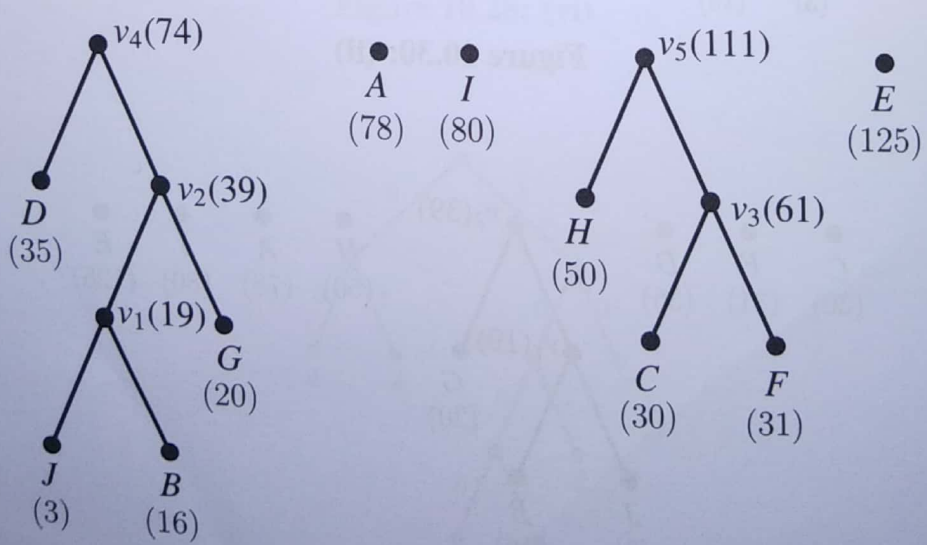


Figure 10.30: (vi)

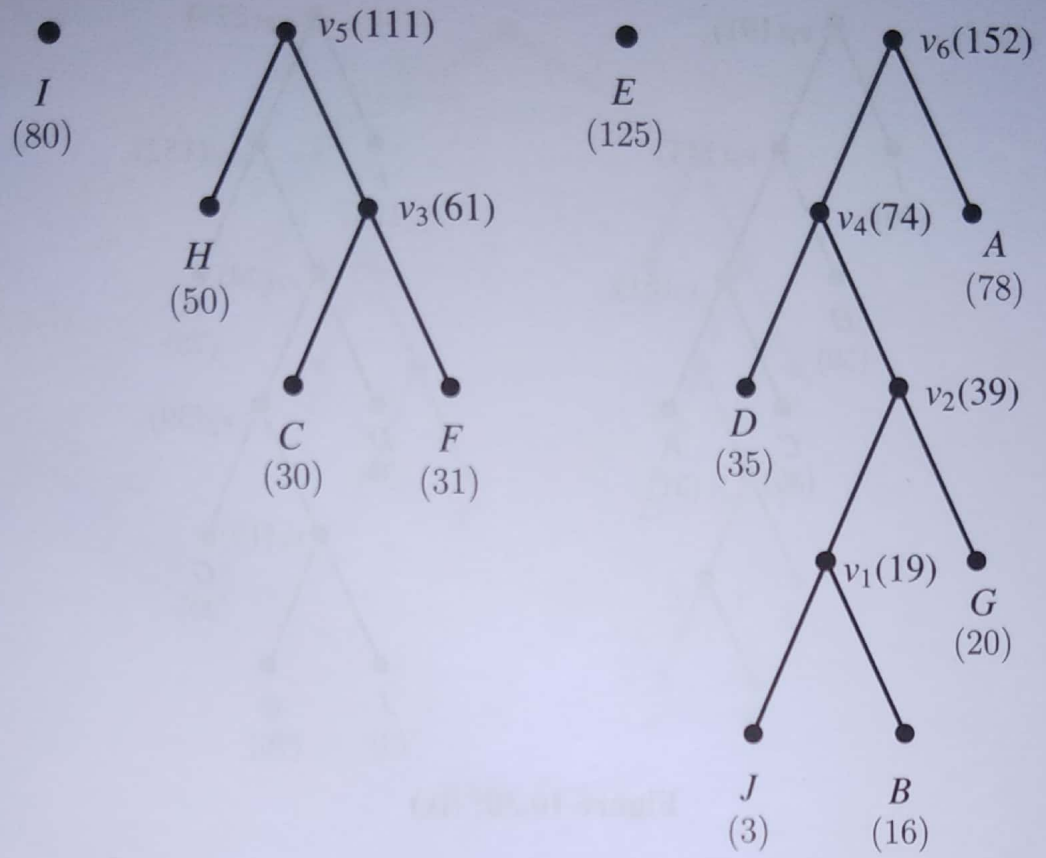


Figure 10.30: (vii)

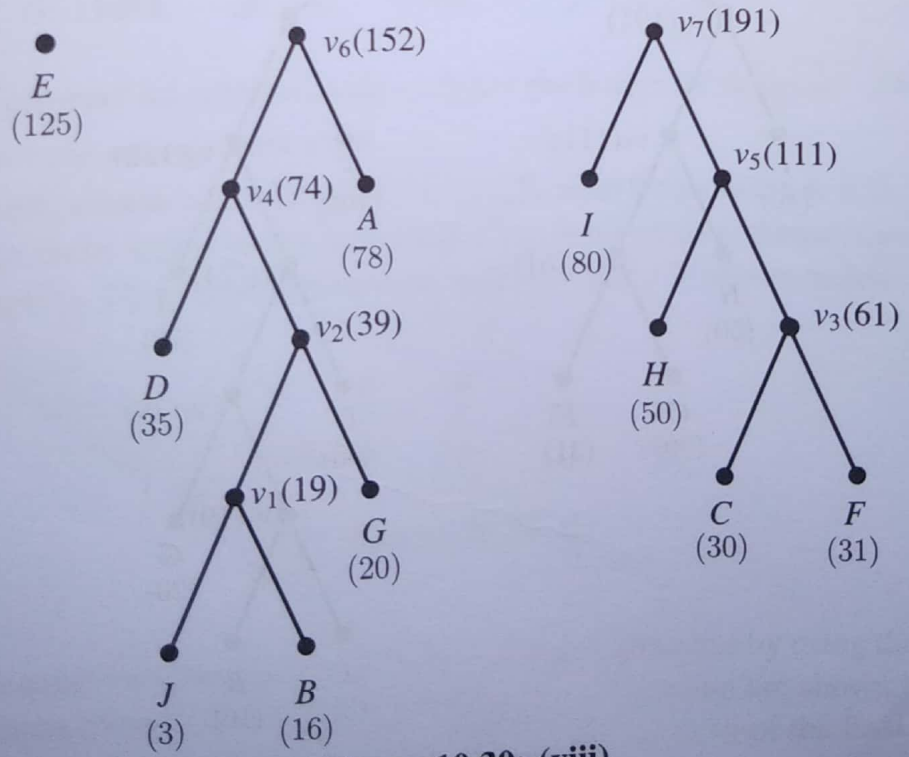


Figure 10.30: (viii)

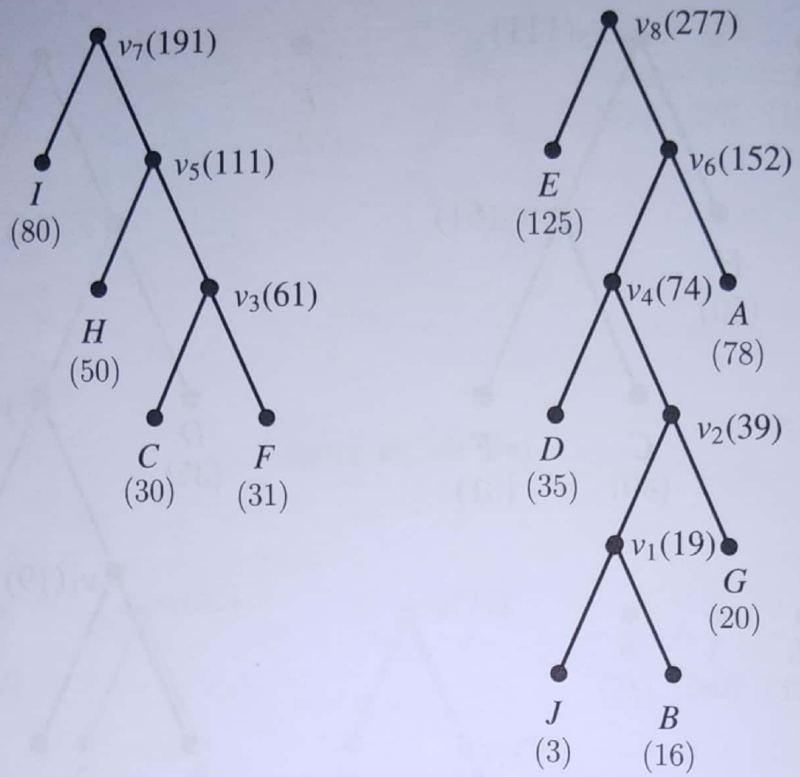


Figure 10.30: (ix)

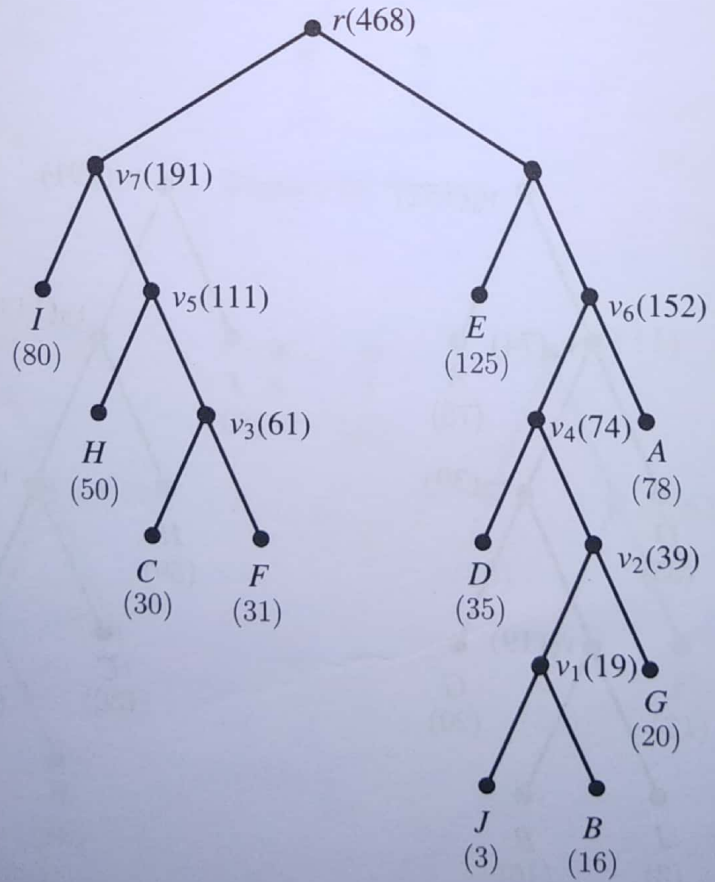


Figure 10.30: (x)

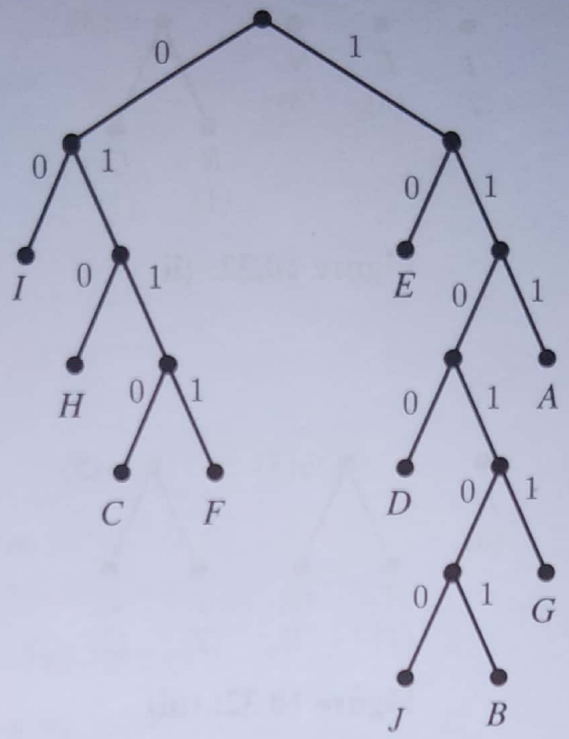


Figure 10.31

The tree in Figure 10.31 is an Huffmann tree (optimal tree) for the given data. From this tree, we obtain the following optimal prefix code for the given symbols.

A: 111,	*B*: 110101,	*C*: 0110,	*D*: 1100,	*E*: 10
F: 0111,	*G*: 11011,	*H*: 010,	*I*: 00,	*J*: 110100.

■

Example 8 *Construct an optimal prefix code for the letters of the word* ENGINEERING. *Hence deduce the code for this word.*

▶ The given word consists of the letters E, N, G, I, R, with frequencies 3, 3, 2, 2, 1 respectively. First, we arrange these letters in the non-decreasing order of their frequencies (which may be regarded as weights). Their representation as isolated verties is shown below.

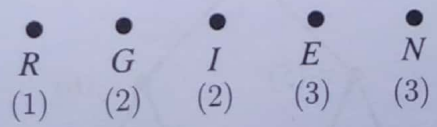


Figure 10.32: (i)

We now construct an optimal tree having these letters as leaves by using the Huffman's procedure. The graphs obtained in successive steps of the procedure are shown below in Figures 10.32(ii)-(v) in the order of their occurrence. The labeled version of the final tree is shown in Figure 10.33.

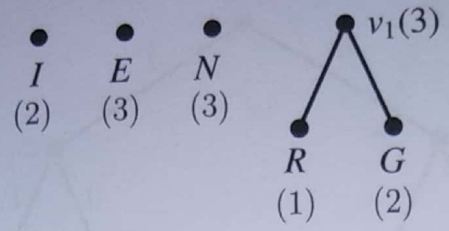


Figure 10.32: (ii)

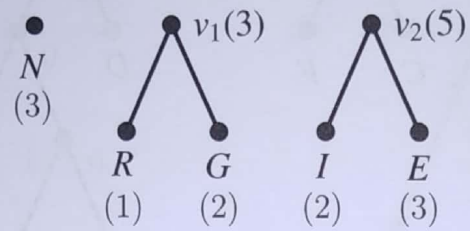


Figure 10.32: (iii)

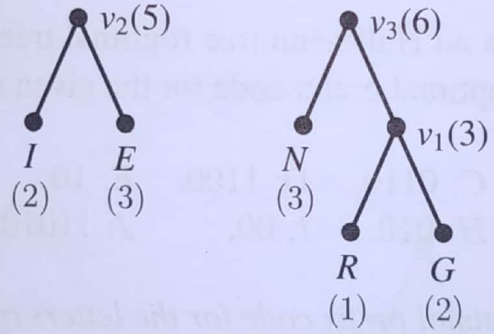


Figure 10.32: (iv)

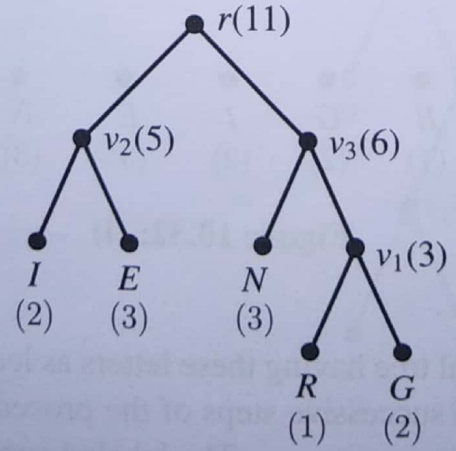


Figure 10.32: (v)

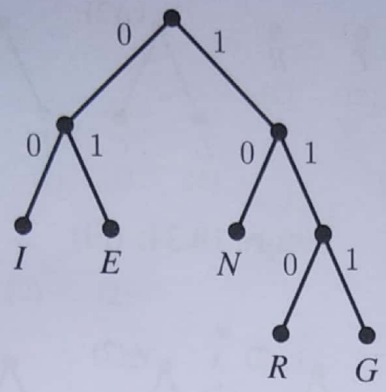


Figure 10.33

The tree shown in Figure 10.33, is the optimal tree that we sought. From this tree, we obtain the optimal prefix codes shown below for the letters with which we started:

$$R : 110 \quad G : 111 \quad I : 00 \quad E : 01 \quad N : 10$$

Accordingly, the code for the given word ENGINEERING is

$$01101110010010111000 10111$$ ■

Example 9 *Obtain an optimal prefix code for the message* LETTER RECEIVED. *Indicate the code.*

► The given message consists of letters L, E, T, R, C, I, V, D with frequencies 1, 5, 2, 2, 1, 1, 1, 1 respectively. Further, there is one blank space (□) between the two words of the message.

First, we arrange the letters and □ in the non-decreasing order of their weights (frequencies). Their representation as isolated vertices is shown below

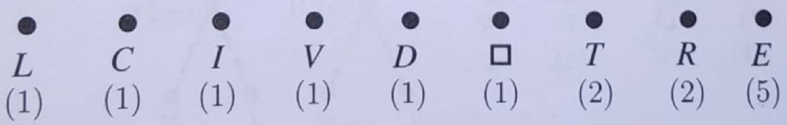


Figure 10.34: (i)

We now construct an optimal tree having these symbols as leaves by using the Huffman's procedure. The graphs obtained in successive steps of the procedure are shown below in Figures 10.34(ii)-(ix) in the order of their occurrence. The labeled version of the final graph is shown in Figure 10.35.

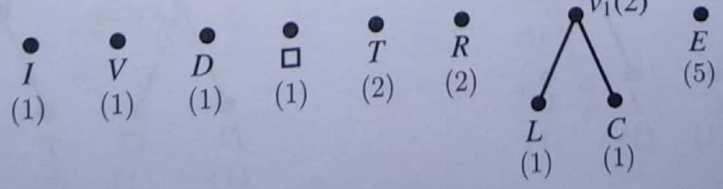


Figure 10.34: (ii)

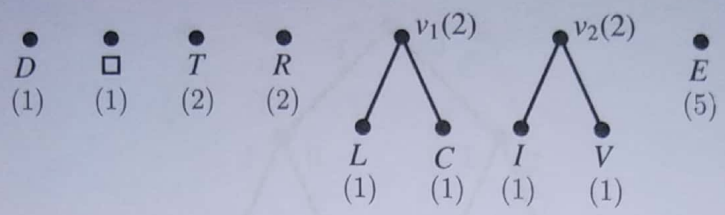


Figure 10.34: (iii)

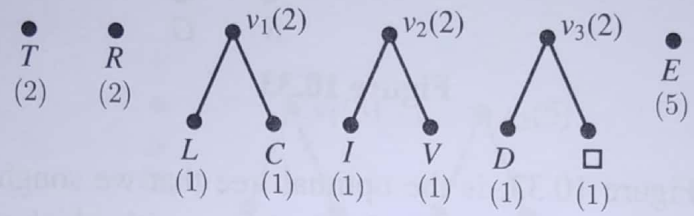


Figure 10.34: (iv)

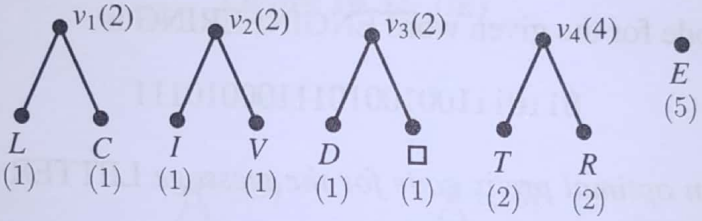


Figure 10.34: (v)

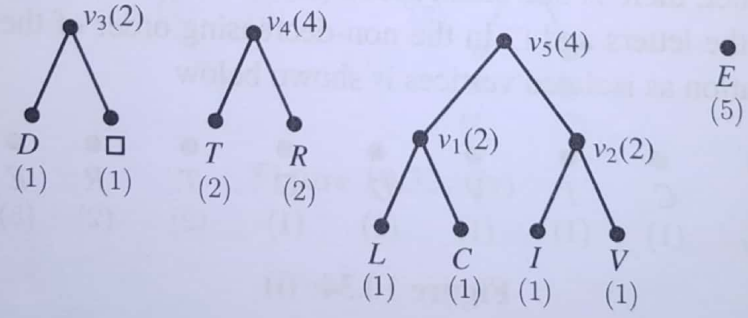


Figure 10.34: (vi)

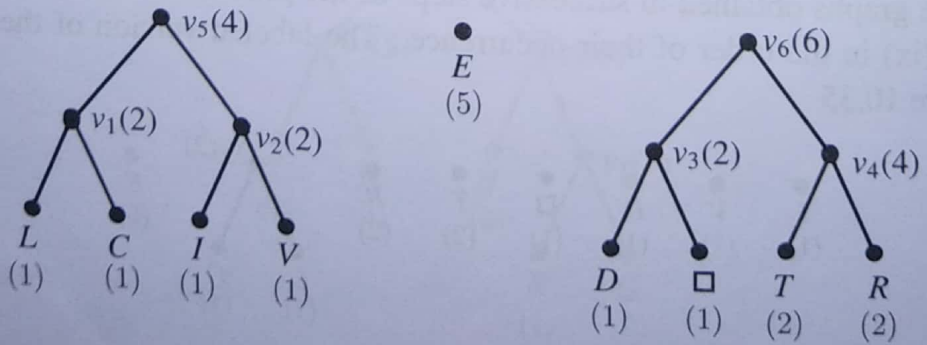


Figure 10.34: (vii)

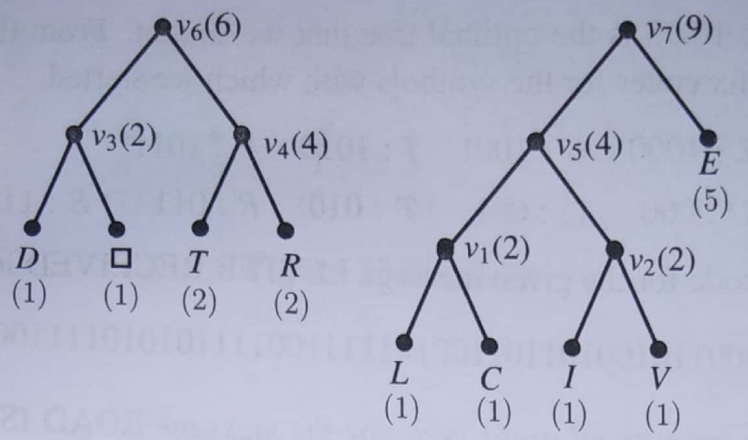


Figure 10.34: (viii)

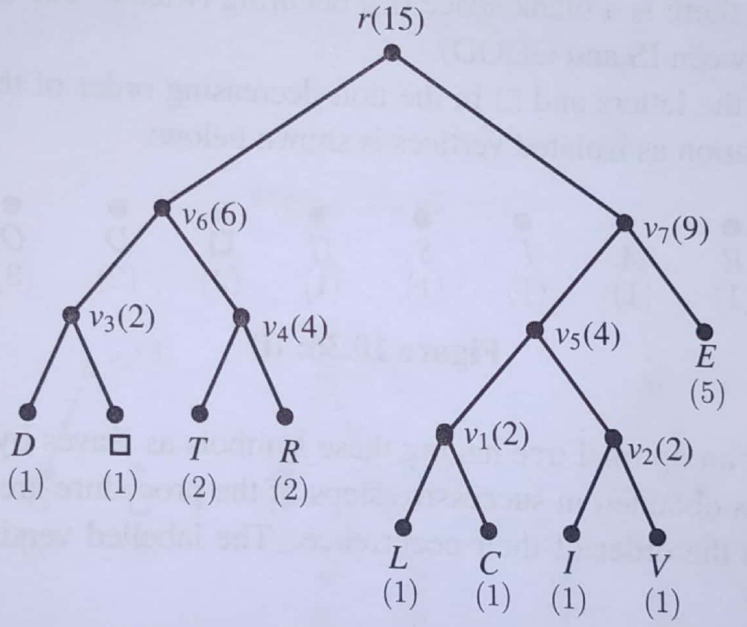


Figure 10.34: (ix)

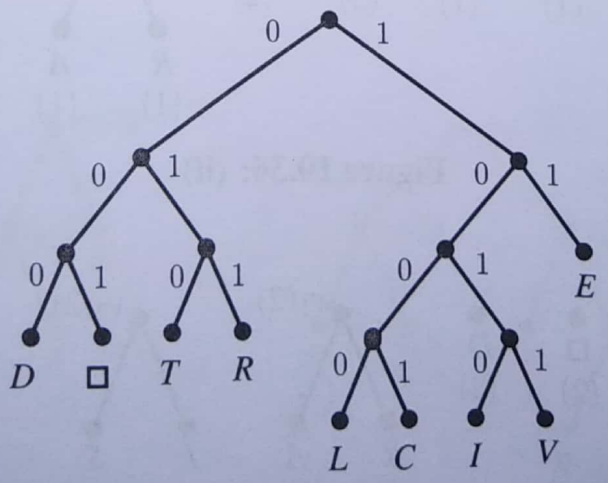


Figure 10.35

The tree in Figure 10.35 is the optimal tree that we sought. From this tree we obtain the following optimal prefix codes for the symbols with which we started.

$L : 1000$ $C : 1001$ $I : 1010$ $V : 1011$

$D : 000$ $\square : 001$ $T : 010$ $R : 011$ $E : 11$

Accordingly, the code for the given message LETTER RECEIVED is:

10001101001011011001011111001111010101111000 ■

Example 10 *Obtain an optimal prefix code for the message* ROAD IS GOOD. *Indicate the code.*

► The given message consists of letters R, O, A, D, I, S, G with frequencies 1, 3, 1, 2, 1, 1, 1, respectively. Further, there is a blank space (□) occuring twice (– one □ between ROAD and IS, and another □ between IS and GOOD).

First, we arrange the letters and □ in the non-decreasing order of their weights (frequencies). Their representation as isolated vertices is shown below:

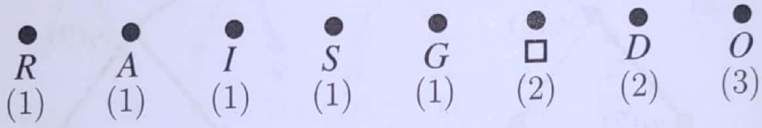


Figure 10.36: (i)

We now construct an optimal tree having these symbols as leaves by using the Huffman's procedure. The graphs obtained in successive steps of the procedure are shown below in Figures 10.36(ii)-(viii) in the order of their occurrence. The labelled version of the final tree is shown in Figure 10.37.

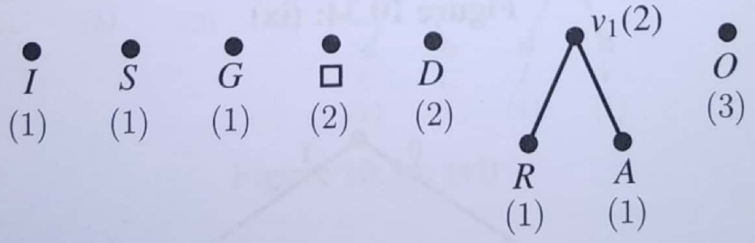


Figure 10.36: (ii)

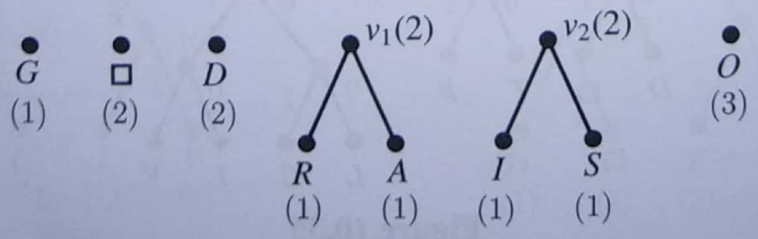


Figure 10.36: (iii)

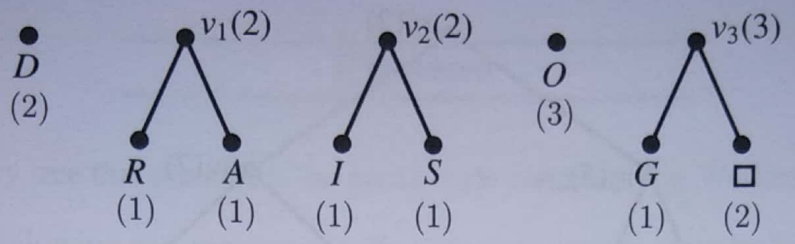


Figure 10.36: (iv)

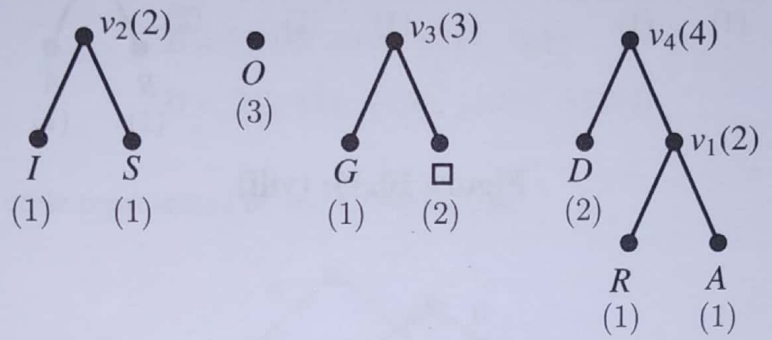


Figure 10.36: (v)

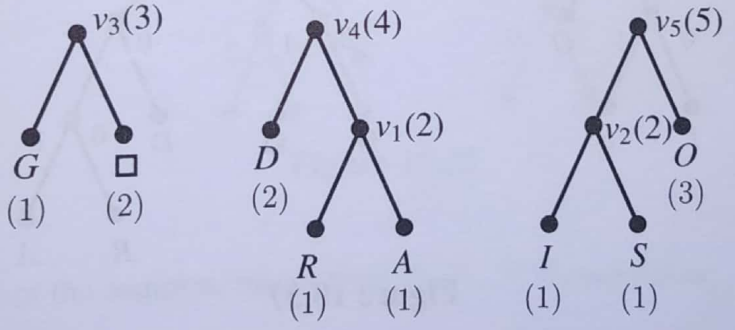


Figure 10.36: (vi)

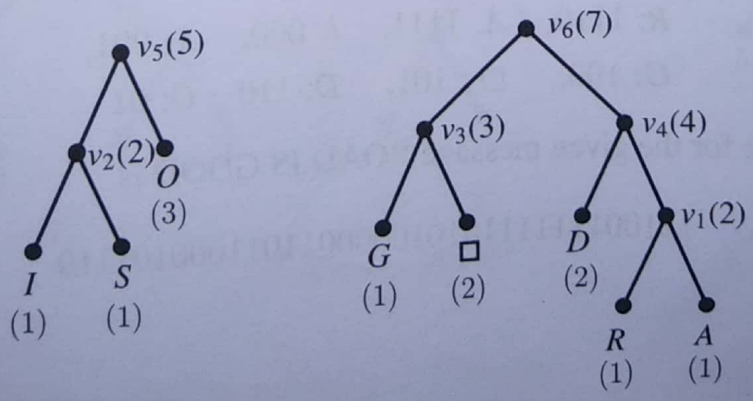


Figure 10.36: (vii)

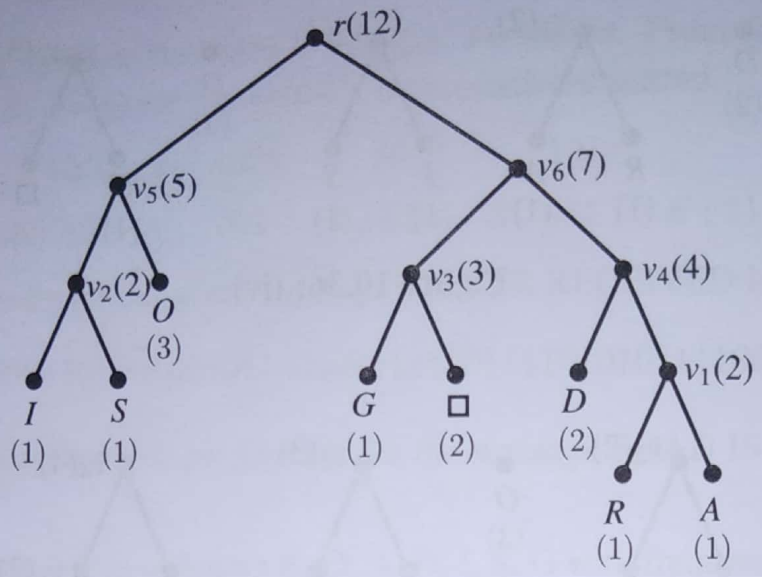


Figure 10.36: (viii)

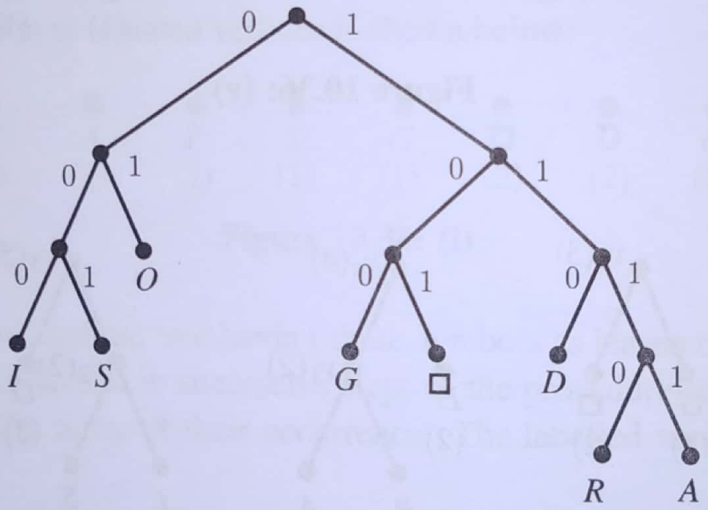


Figure 10.37

The tree shown in Figure 10.37 is the optimal tree that we sought. From this tree, we obtain the following optional prefix code for the symbols with which we started.

R: 1110, A: 1111, I: 000, S: 001,
G: 100, □ : 101, D: 110 O: 01

Accordingly, the code for the given message ROAD IS GOOD is

11100111111101010000011011000101110 ■

Exercises

1. Draw the binary tree that represents the prefix code considered in Worked Example 1.
2. Which of the following sets represent prefix codes? Represent them by binary trees.

$$C = \{00, 010, 011, 10, 111, 1100, 1101\}$$
$$A = \{00, 011, 111, 11011\}$$
$$B = \{1100, 1010, 1111, 101\}$$
$$D = \{10, 111, 1100, 11010, 11011\}$$

3. Find the prefix code represented by the following tree:

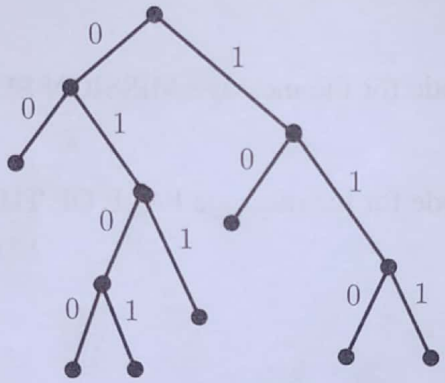
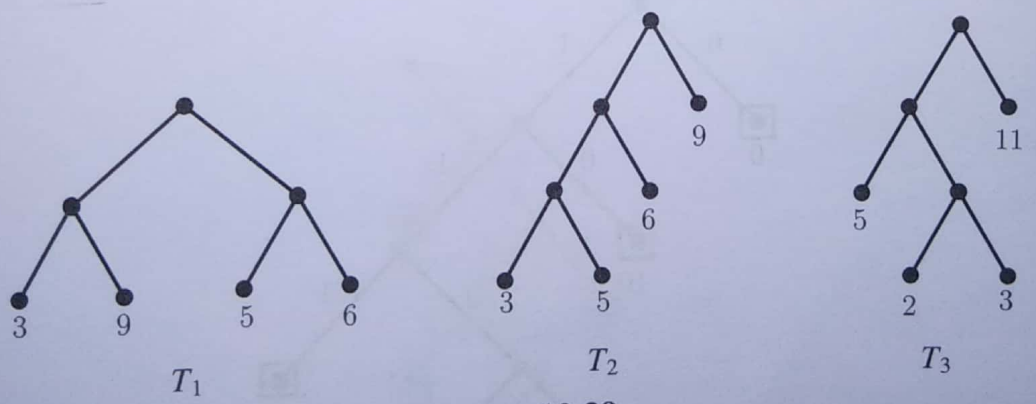


Figure 10.38

4. Find the weights of the complete binary trees T_1, T_2, T_3 shown below:



T_1 T_2 T_3

Figure 10.39

5. Construct an optimal tree for the weights 2, 3, 5, 10, 10.
6. Construct an optimal tree for the weights 11, 9, 12, 14, 10, 24 and 16.

7. Construct an optimal prefix code for the symbols with given weights in each of the following cases:

(1)

symbol:	a	b	c	d	e	f	g
weight:	10	30	5	15	20	15	5

(2)

symbol:	A	B	C	D	E	F	G	H
weight:	22	5	11	19	2	11	25	5

8. Obtain an optimal prefix code for the letters of the word CALCULUS. Indicate the code for the word.

9. Obtain an optimal prefix code for the message TAKE CARE. Indicate the code of the message.

10. Obtain an optimal prefix code for the message PROPOSAL ACCEPTED. Indicate the code for the message.

11. Obtain an optimal prefix code for the message MISSION SUCCESSFUL. Indicate the code for the message.

12. Obtain an optimal prefix code for the message FALL OF THE WALL. Indicate the code for the message.

Answers

1.

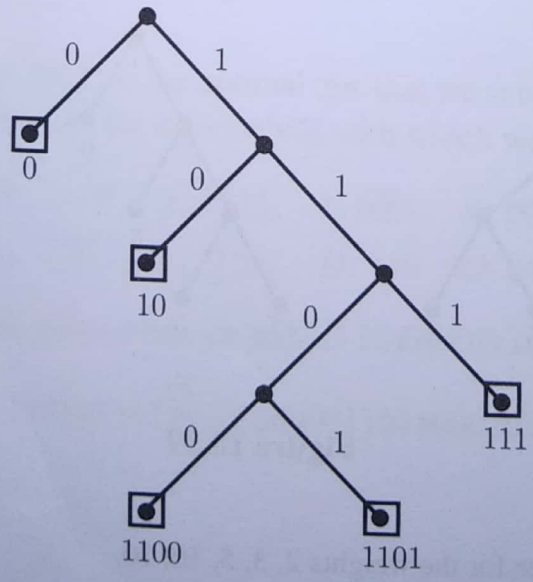
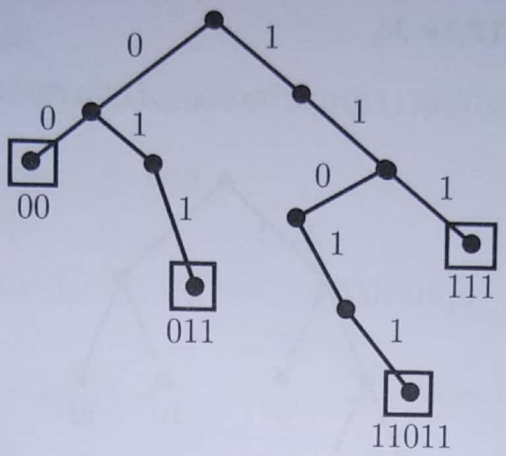


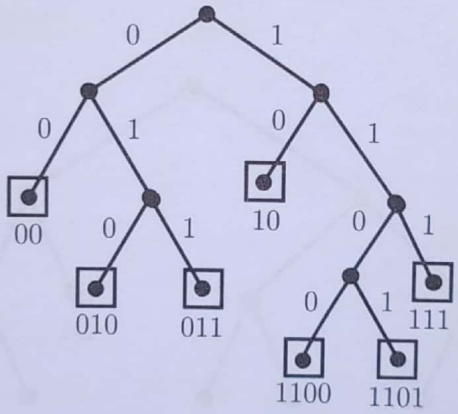
Figure 10.40

2. Sets A, C, D are prefix codes. Their binary trees are as shown below.



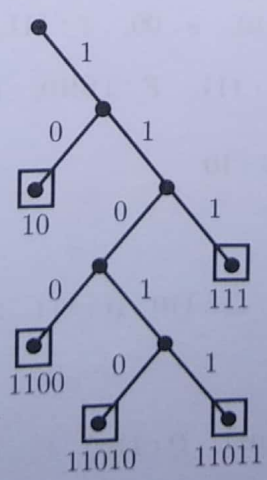
(A)

Figure 10.41



(C)

Figure 10.42



(D)

Figure 10.43

3. $P = \{00, 0100, 0101, 011, 10, 110, 111\}$

4. $W(T_1) = 46,\ W(T_2) = 45,\ W(T_3) = 36,$

5.

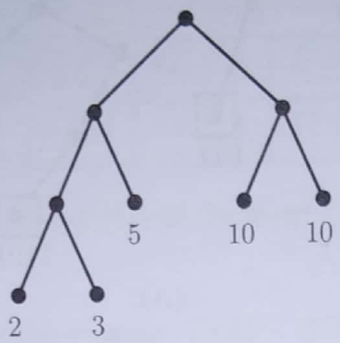


Figure 10.44

6.

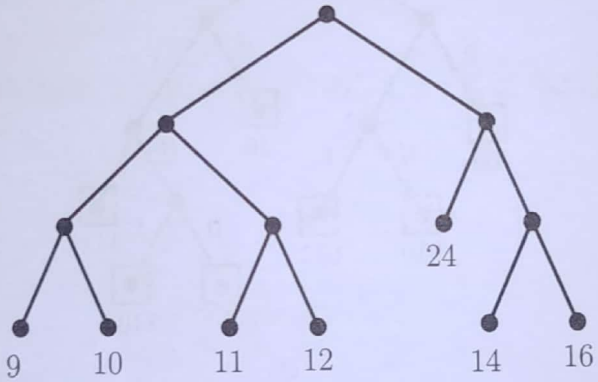


Figure 10.45

7. (1) a : 010, b : 10, c : 0110, d : 110, e : 00, f : 111, g : 0111

   (2) A : 00, B : 11011, C : 010, D : 111, E : 11010, F : 011, G : 10, H : 1100

8. A : 110 S : 111 C : 00 L : 01 U : 10

   code : 001100100100110111

9. T : 010 K : 011 C : 100 R : 101 □ : 110 A : 111 E : 00

   code : 010111011001101001111010 0

10. R : 1000 S : 1001 L : 1010 T : 1011 D : 1100 □ : 1101

    O : 000 A : 001 C : 010 E : 011 P : 111

    code : 1111000000111000100100110101101001010010011111101101111 00

11. *M*: 0100 *O*: 0101 *N*: 0110 *E*: 0111
F: 1100 *L*: 1101 □: 1110 *I*: 1111
U: 000 *C*: 001 *S*: 10

code : 0100111110101111010101101110100000010010111101011000001101

12. *O*: 0110 *T*: 0111 *H*: 1100 *E*: 1101
W: 000 *F*: 001 *A*: 010 □: 111 *L*: 10

code: 0010101010111011000111101111001101111000010101 0

VTU Syllabus

(2019-20)

Paper 18CS36: DISCRETE MATHEMATICAL STRUCTURES

Module - 1

Fundamentals of Logic: Basic Connectives – Truth Tables. Logical Equivalence. The Laws of Logic. Logical Implication. Rules of Inference. Quantifiers. Proofs of Theorems.

Module - 2

Properties of Integers: The Well Ordering Principle. Mathematical Induction.

Fundamental Principles of counting; Rules of Sum and Product, Permutations, Combinations, Binomial Theorem, Combinations with repetitions.

Module - 3

Cartesian Products and Relations. Functions – plain, one-to-one and onto functions. The Pigeon-hole Principle. Function composition and Inverse functions.

Properties of Relations. Computer Recognition – Zero-One Matrices and Directed Graphs. Partial Orders – Hasse Diagrams. Equivalence Relations and Partitions.

Module - 4

The Principle of Inclusion-exclusion, Generalization of the Principle, Derangements, Rook Polynomials.

First-order linear recurrence relations. The Second-order linear homogeneous recurrence relations with constant coefficients.

Module - 5

Graphs, Definitions and Examples. Subgraphs, Complements. Graph Isomorphism.

Trees, Definitions, Properties and Examples, Rooted Trees, Weighted Trees and Prefix Codes.

Examination

The Question Paper will have a total of 10 full questions, each carrying 20 marks. There will be Two full questions from each Module of the Syllabus. Each full question will have a maximum of four subquestions covering all the topics under the respective Module.

For the maximum of 100 marks, *five* full questions choosing *one* full question from *each Module* are to be answered. Marks scored will be proportionately reduced to a maximum of 60 marks.

Distribution of Chapters in this Book

In this Book, the distribution of Chapters among the Modules in the Syllabus is as given below.

Module 1. Chapters 1 and 2

Module 2. Chapters 3 and 4

Module 3. Chapter 5 and 6

Module 4. Chapters 7 and 8

Module 5. Chapters 9 and 10