Ceng 124
Discrete Structures

2018-2019 Spring Semester

by Sibel T. Özyer, Spring 2019

Topics

- 11.1 Introduction to Trees
  - Rooted Trees
  - Terminology of Trees
  - M-ary Trees
  - Binary Trees
- 11.2 Applications of Trees
  - Binary Search Trees
  - Code Prefix
  - Huffman Coding

11.1 Introduction to Trees

- A connected graph that contains no simple circuits is called a tree.
- particularly useful in computer science.
- used to construct efficient algorithms for locating items in a list.
- such as Huffman coding, that construct efficient codes saving costs in data transmission and storage.
- games such as checkers and chess.
- sorting and searching algorithms.

Definition

A *tree* is a connected undirected graph with no simple circuits.

Because a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops. Therefore any tree must be a simple graph.

Question

Which of the graphs shown in Figure 1 are trees?

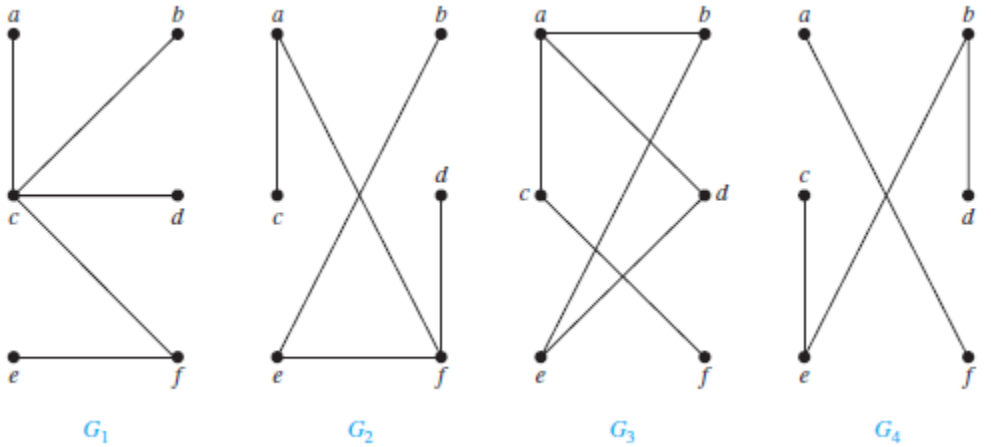


Figure 1: Examples of Trees and Graphs That Are Not Trees.

Solution

- $G1$ and $G2$ are trees, because both are connected graphs with no simple circuits.
- $G3$ is not a tree because e, b, a, d, e is a simple circuit in this graph.
- Finally, $G4$ is not a tree because it is not connected.

Theorem

An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

What about graphs containing no simple circuits that are not necessarily connected? These graphs are called forests and have the property that each of their connected components is a tree.

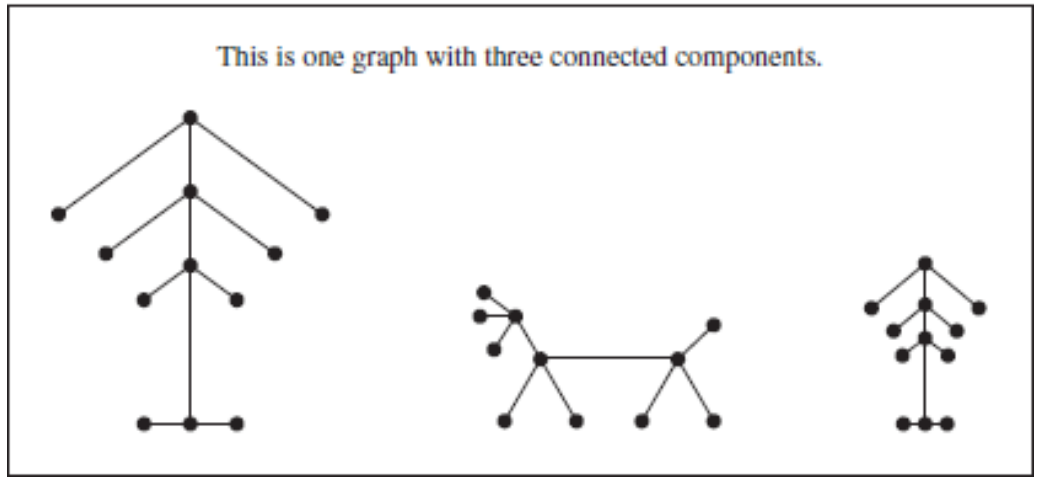


Figure 2: Example of a Forest.

Rooted Trees

- In many applications of trees, a particular vertex of a tree is designated as the root.
- We can assign a direction to each edge. Because there is a unique path from the root to each vertex of the graph, we direct each edge away from the root. Thus, a tree together with its root produces a directed graph called a rooted tree.

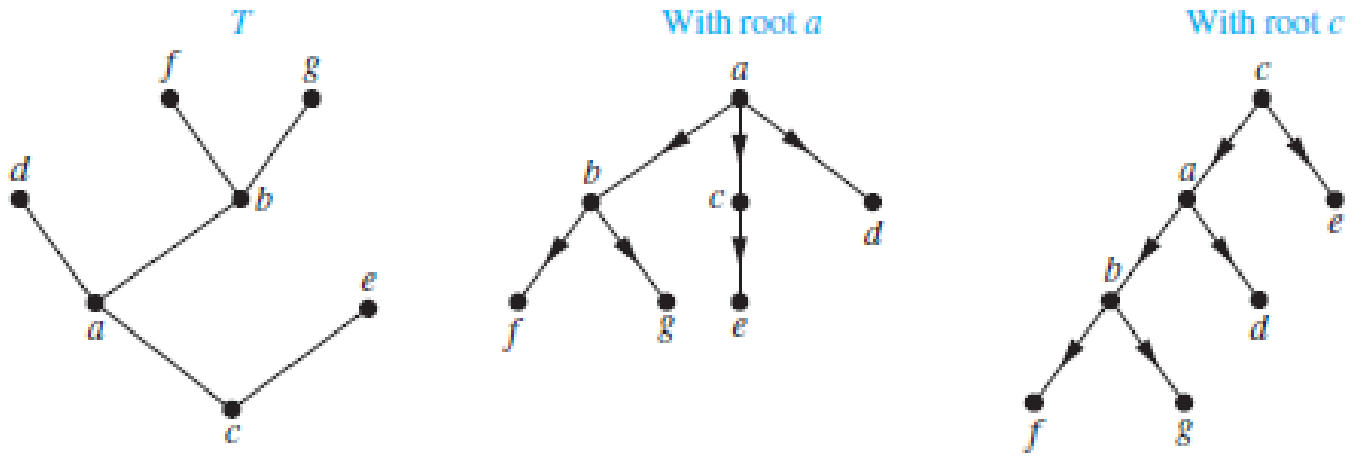
Definition

A *rooted tree* is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

Terminology for Trees

- Suppose that *T* is a rooted tree. If *v* is a vertex in *T* other than the root, the **parent** of *v* is the unique vertex *u* such that there is a directed edge from *u* to *v*.
- When *u* is the parent of *v*, *v* is called a **child** of *u*. Vertices with the same parent are called **siblings**.
- The **ancestors** of a vertex other than the root are the vertices in the path from the root to this vertex,excluding the vertex itself and including the root.
- The **descendants** of a vertex *v* are those vertices that have *v* as an ancestor. A vertex of a rooted tree is called a **leaf** if it has no children.

Samples



Figuere 3: A Tree and Rooted Trees Formed by Designating Two Different Roots.

Example

- In the rooted tree T (with root a) shown in Figure 4, find the parent of c, the children of g, the siblings of h, all ancestors of e, all descendants of b, all internal vertices, and all leaves. What is the subtree rooted at g?

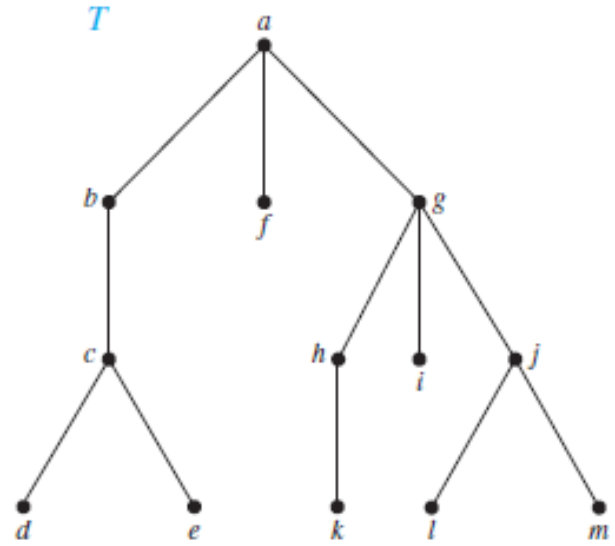


Figure 4: A Rooted Tree T

Solution

- The parent of *c* is *b*. The children of *g* are *h*, *i*, and *j* .
- The siblings of *h* are *i* and *j* . The ancestors of *e* are *c*, *b*, and *a*.
- The descendants of *b* are *c*, *d*, and *e*. The internal vertices are *a*, *b*, *c*, *g*, *h*, and *j* .
- The leaves are *d*, *e*, *f* , *i*, *k*, *l*, and *m*. The subtree rooted at *g* is shown in Figure 5.

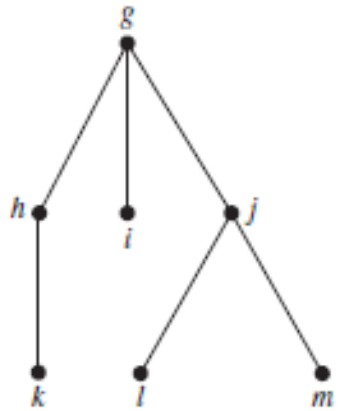


Figure 5: The Subtree Rooted at *g*.

M-ary Tree

A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children. The tree is called a *full m-ary tree* if every internal vertex has exactly m children. An m-ary tree with $m = 2$ is called a *binary tree*.

Question

- Are the rooted trees in Figure 6 full *m*-ary trees for some positive integer *m*?

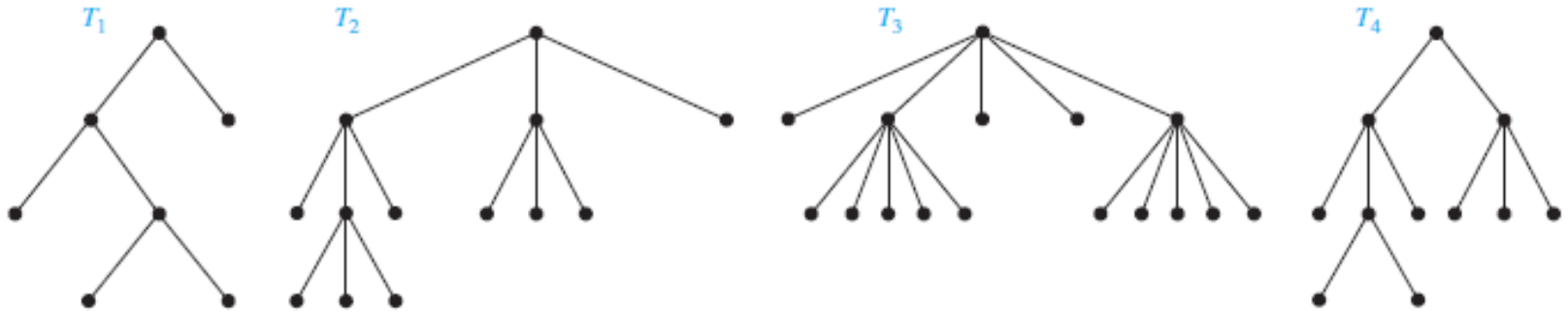


Figure 6: Four Rooted Trees

Solution

- $T1$ is a full binary tree because each of its internal vertices has two children.
- $T2$ is a full 3-ary tree because each of its internal vertices has three children.
- In $T3$ each internal vertex has five children, so $T3$ is a full 5-ary tree.
- $T4$ is not a full m-ary tree for any m because some of its internal vertices have two children and others have three children.

Ordered Rooted Trees

- is a rooted tree where the children of each internal vertex are ordered. Ordered rooted trees are drawn so that the children of each internal vertex are shown in order from left to right.
- In an ordered binary tree (usually called just a **binary tree**), if an internal vertex has two children, the first child is called the **left child** and the second child is called the **right child**.

Question

- What are the left and right children of *d* in the binary tree *T* shown in Figure 7(a)? What are the left and right subtrees of *c*?

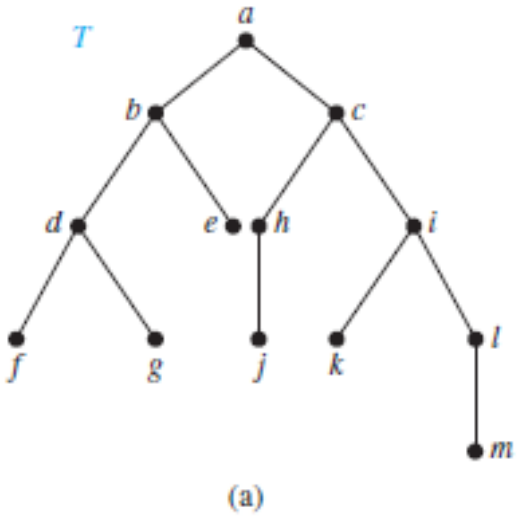


Figure 7 (a): A Binary Tree *T*

Solution

- The left child of *d* is *f* and the right child is *g*.
- Left and Right Subtrees of the Vertex *c*:

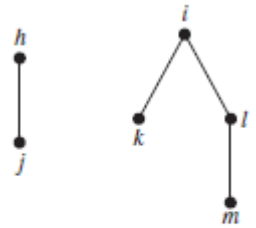


Figure 7 (b) (c): Left and Right Subtrees of the Vertex *c*.

Trees as Models

Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, and psychology.

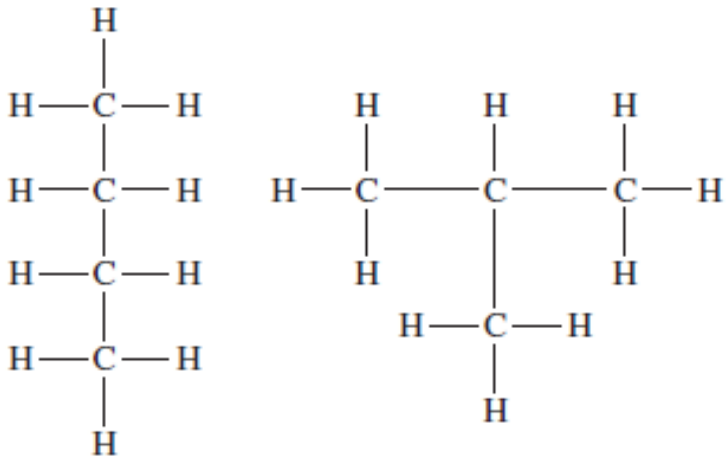


Figure 8: The Two Isomers of Butane

Trees as Models (cont.)

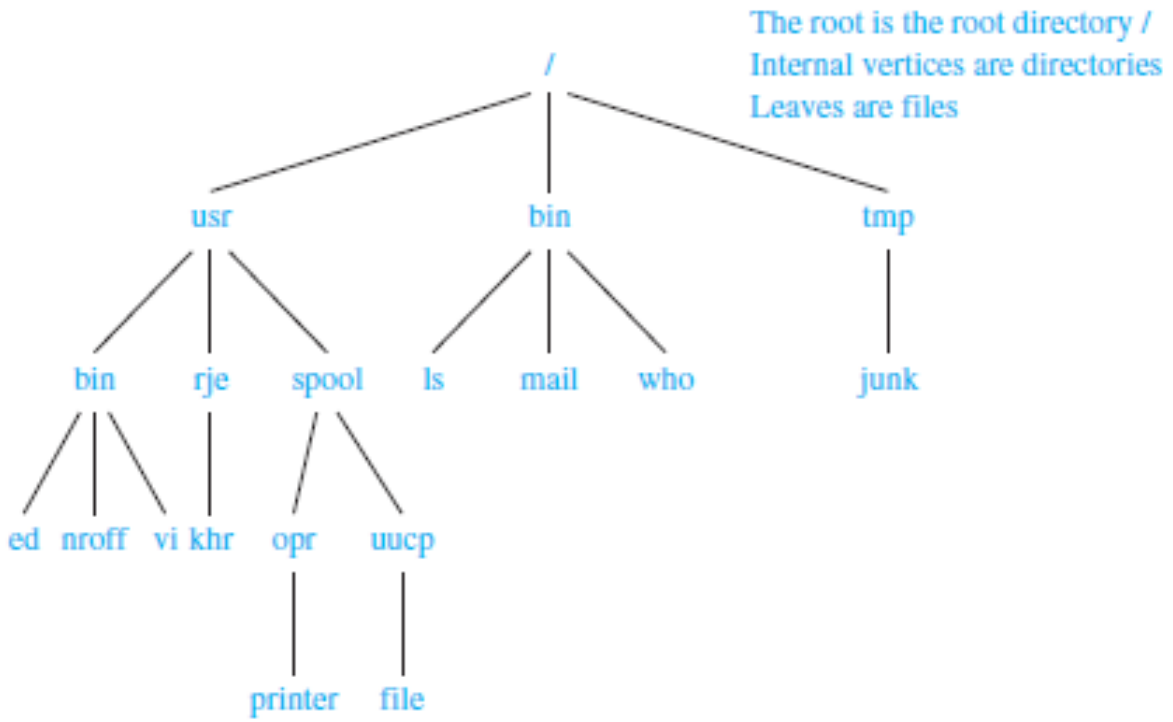


Figure 9: A computer File Sytem

11.2 Applications of Trees

- How should items in a list be stored so that an item can be easily located?
- What series of decisions should be made to find an object with a certain property in a collection of objects of a certain type?
- How should a set of characters be efficiently coded by bit strings?

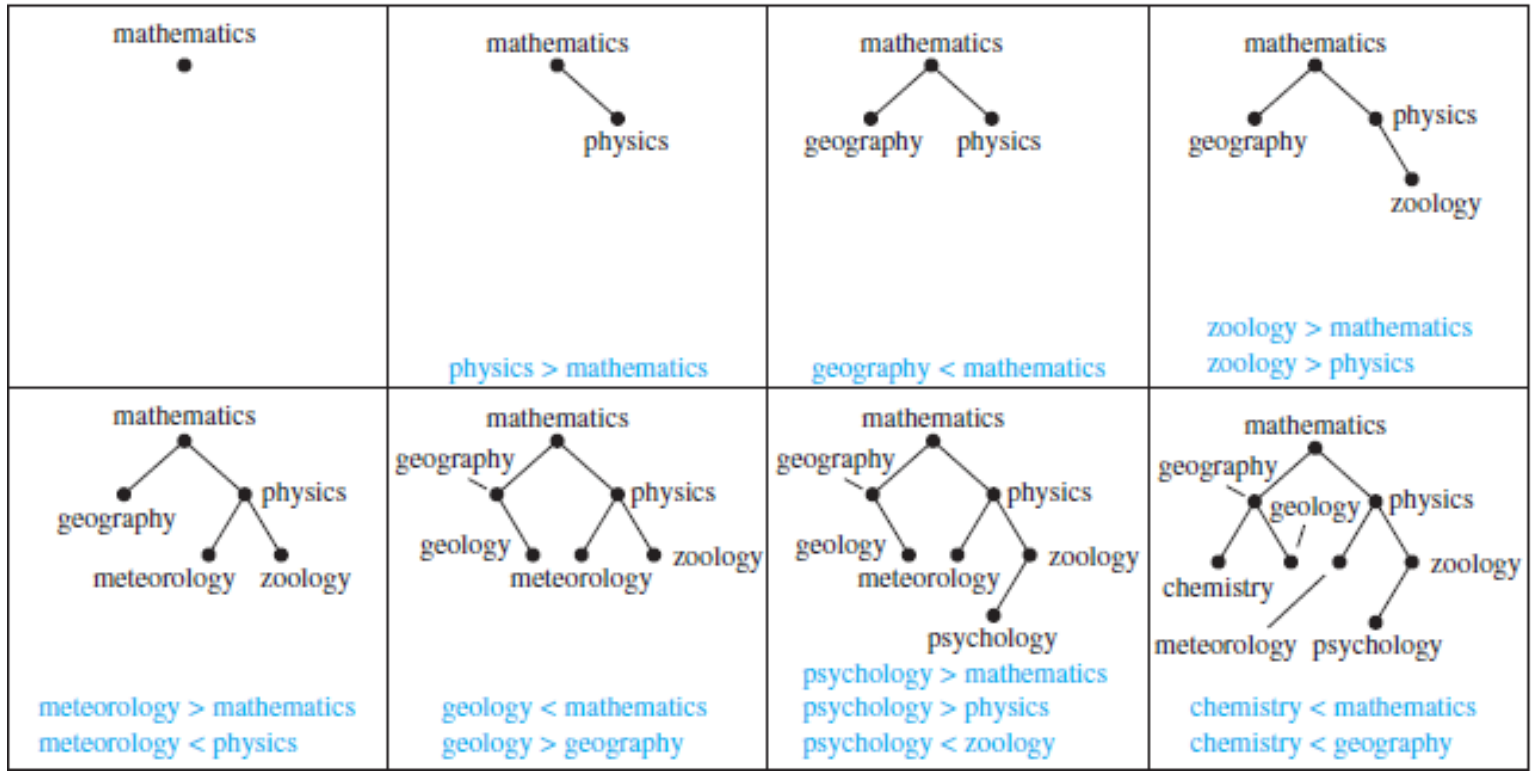
Binary Search Trees

- start with a tree containing just one vertex, namely, the root.
- to add a new item, first compare it with the keys at the root.
- moving to the left if the item is less than the key of the respective vertex if this vertex has a left child,
- or moving to the right if the item is greater than the key of the respective vertex if this vertex has a right child.

Sample

- Form a binary search tree for the words *mathematics*, *physics*, *geography*, *zoology*, *meteorology*, *geology*, *psychology*, and *chemistry* (using alphabetical order).

Constructing a Binary Search Tree



Prefix Codes

- When letters are encoded using varying numbers of bits, some method must be used to determine where the bits for each character start and end.
- For instance, if *e* were encoded with 0, *a* with 1, and *t* with 01, then the bit string 0101 could correspond to *eat*, *tea*, *eaea*, or *tt*.
- One way to ensure that no bit string corresponds to more than one sequence of letters is to encode letters so that the bit string for a letter never occurs as the first part of the bit string for another letter. Codes with this property are called **prefix codes**.

Prefix Codes Sample

- The encoding of *e* as 0, *a* as 10, and *t* as 11 is a prefix code.
- The string 10110 is the encoding of *ate*.
- note that the initial 1 does not represent a character, but 10 does represent *a* (and could not be the first part of the bit string of another letter).
- Then, the next 1 does not represent a character,
- but 11 does represent *t*. The final bit, 0, represents *e*.

Prefix Codes with Binary Tree

- A prefix code can be represented using a binary tree.
- The edges of the tree are labeled so that an edge leading to a left child is assigned a 0 and an edge leading to a right child is assigned a 1.

Prefix Codes Sample

The tree in Figure 10 represents the encoding of *e* by 0, *a* by 10, *t* by 110, *n* by 1110, and *s* by 1111.

For instance, consider the word encoded by 11111011100 using the code in Figure 10. The original word is *sane*.

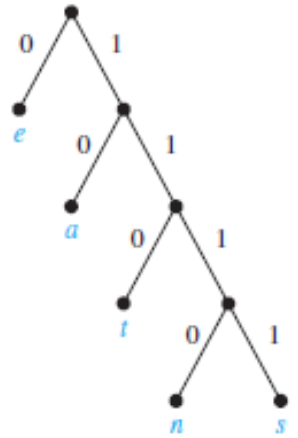


Figure 10: A Binary Tree with a Prefix Code.

Huffman Coding

- An algorithm that takes as input the frequencies of symbols in a string and produces as output a
- prefix code that encodes the string using the fewest possible bits, among all possible binary prefix codes for these symbols.
- This algorithm, known as Huffman coding, was developed by
- David Huffman in a term paper he wrote in 1951 while a graduate student at MIT.
- Huffman coding is a fundamental algorithm in *data compression*,
- the subject devoted to reducing the number of bits required to represent information.

Question

- Use Huffman coding to encode the following symbols with the frequencies listed: A: 0.08, B: 0.10, C: 0.12, D: 0.15, E: 0.20, F: 0.35. What is the average number of bits used to encode a character?
- Solution: Figure displays the steps used to encode these symbols. The encoding produced encodes A by 111, B by 110, C by 011, D by 010, E by 10, and F by 00.
- The average number of bits used to encode a symbol using this encoding is

  $3 \cdot 0.08 + 3 \cdot 0.10 + 3 \cdot 0.12 + 3 \cdot 0.15 + 2 \cdot 0.20 + 2 \cdot 0.35 = 2.45.$

Huffman Coding of Symbols - Solution of Question