CHAPTER 7

# Trees

## 7.1. Trees

**7.1.1. Terminology.** Let $T$ be a graph with $n$ vertices. The following properties are equivalent:

1. $T$ is connected and acyclic (has no cycles).
2. $T$ is connected and has $n-1$ edges.
3. $T$ is acyclic and has $n-1$ edges.
4. If $v$ and $w$ are vertices in $T$, there is a unique simple path from $v$ to $w$.

A graph having any of the above equivalent properties is called a *free tree* or simply *tree*. A union of trees, or equivalently a simple graph with no cycles, is called *forest*.

A *rooted tree* is a tree in which a particular vertex is designated as the root.

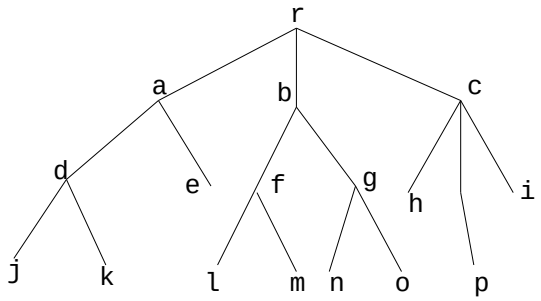


FIGURE 7.1. A rooted tree.

The *level* of a vertex $v$ is the length of the simple path from the root to $v$. The *height* of a rooted tree is the maximum level of its vertices.

Let $T$ be a tree with root $v_0$. Suppose that $x$, $y$ and $z$ are vertices in $T$ and that $(v_0, v_1, \ldots, v_n)$ is a simple path in $T$. Then:

1. $v_{n-1}$ is the *parent* of $v_n$.
2. $v_0, v_1, \ldots, v_{n-1}$ are *ancestors* of $v_n$.
3. $v_n$ is a *child* of $v_{n-1}$.
4. If $x$ is an ancestor of $y$, $y$ is a *descendant* of $x$.
5. If $x$ and $y$ are children of $z$, $x$ and $y$ are *siblings*.
6. If $x$ has no children, it is called a *terminal vertex* or *leaf*.
7. If $x$ is not a terminal vertex, it is an *internal* or *branch vertex*.
8. The *subtree of $T$ rooted at $x$* is the graph $(V, E)$, where $V$ is $x$ together with its descendants and $E$ = edges of simple paths from $x$ to some vertex in $E$.

**7.1.2. Huffman Codes.** Usually characters are represented in a computer with fix length bit strings. *Huffman codes* provide an alternative representation with variable length bit strings, so that shorter strings are used for the most frequently used characters. As an example assume that we have an alphabet with four symbols: $A = \{a, b, c, d\}$. Two bits are enough for representing them, for instance $a = 11$, $b = 10$, $c = 01$, $d = 00$ would be one such representation. With this encoding $n$-character words will have $2n$ bits. However assume that they do not appear with the same frequency, instead some are more frequent that others, say $a$ appears with a frequency of 50%, $b$ 30%, $c$ 15% and $d$ 5%. Then the following enconding would be more efficient than the fix length encoding: $a = 1$, $b = 01$, $c = 001$, $d = 000$. Now in average an $n$-character word will have $0.5n$ $a$'s, $0.3n$ $b$'s, $0.15n$ $c$'s and $0.05n$ $d$'s, hence its length will be $0.5n{\cdot}1{+}0.3n{\cdot}2{+}0.15n{\cdot}3{+}0.05n{\cdot}3 = 1.7n$, which is shorter than $2n$. In general the length per character of a given encoding with characters $a_1, a_2, \ldots, a_n$ whose frequencies are $f_1, f_2, \ldots, f_n$ is

$$\frac{1}{F}\sum_{k=1}^{n} f_k\, l(a_k)\,,$$

where $l(a_k)$ = length of $a_k$ and $F = \sum_{k=1}^{n} f_k$. The problem now is, given an alphabet and the frequencies of its characters, find an optimal encoding that provides minimum average length for words.

Fix length and Huffman codes can be represented by trees like in figure 7.2. The code of each symbol consists of the sequence of labels of the edges in the path from the root to the leaf with the desired symbol.

**7.1.3. Constructing an Optimal Huffman Code.** An optimal Huffman code is a Huffman code in which the average length of the symbols is minimum. In general an optimal Huffman code can be made

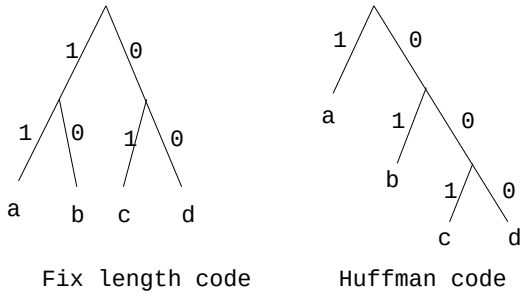


FIGURE 7.2. Fix length code and Huffman code.

as follows. First we list the frequencies of all the codes and represent the symbols as vertices (which at the end will be leaves of a tree). Then we replace the two smallest frequencies $f_1$ and $f_2$ with their sum $f_1 + f_2$, and join the corresponding two symbols to a common vertex above them by two edges, one labeled 0 and the other one labeled 1. Than common vertex plays the role of a new symbol with a frequency equal to $f_1 + f_2$. Then we repeat the same operation with the resulting shorter list of frequencies until the list is reduced to one element and the graph obtained becomes a tree.

*Example*: Find the optimal Huffman code for the following table of symbols:

| character | frequency |
|---|---|
| $a$ | 2 |
| $b$ | 3 |
| $c$ | 7 |
| $d$ | 8 |
| $e$ | 12 |

*Answer*: : The successive reductions of the list of frequencies are as follows:

$$\underbrace{2,3}_{5},7,8,12 \to \underbrace{5,7}_{12},8,12 \to 12,8,12$$

Here we have a choice, we can choose to add the first 12 and 8, or 8 and the second 12. Let's choose the former:

$$\underbrace{12,8}_{20},12 \to \underbrace{20,12}_{32} \to 32$$

The tree obtained is the following:

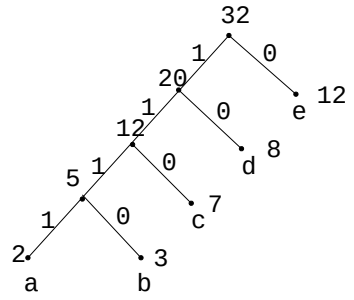


FIGURE 7.3. Optimal Huffman code 1.

The resulting code is as follows:

| character | code |
|---|---|
| $a$ | 1111 |
| $b$ | 1110 |
| $c$ | 110 |
| $d$ | 10 |
| $e$ | 0 |

The other choice yields the following:

$$12, \underbrace{8, 12}_{20} \rightarrow \underbrace{20, 12}_{32} \rightarrow 32$$

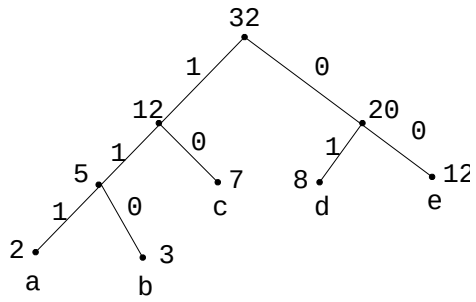


FIGURE 7.4. Optimal Huffman code 2.

| character | code |
|---|---|
| $a$ | 111 |
| $b$ | 110 |
| $c$ | 10 |
| $d$ | 01 |
| $e$ | 00 |