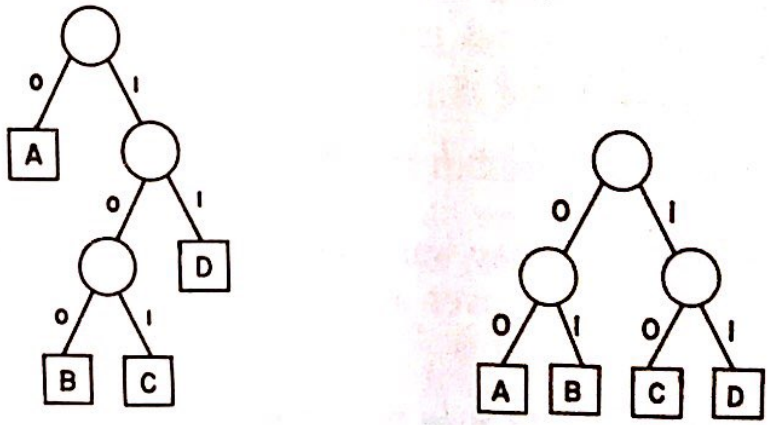
§5.2 HUFFMAN'S TREE

Consider the binary trees in Figure 5.4 where the four leaves are labeled A, B, C, D and the left arcs are labeled zero and the right arcs are labeled one.



(a) (b)

Figure 5.4

We can regard the trees in Figure 5.4 as representations of the letters A, B, C, D by binary sequences. Such a representation is called an encoding. A letter is represented by the binary digits associated with the arcs which form the path to the letter. Thus in Figure 5.4(a) the letters and their corresponding binary sequences are

A	0
B	100
C	101
D	11

A sequence such as 100011 of zero's and one's can be uniquely decoded as follows: Starting from the left, we delete the subsequence which corresponds to a letter. In other words, we mark off the left part as soon as its correspondence to a letter is known. Thus, the sequence

1 0 0 0 1 1

corresponds to

100 0 11

B A D

Note that in the encoding of the letters A, B, C, D, no code of one letter is a prefix of another. This property is called the prefix property and it is why we use the extended binary tree and represent the letters by the leaves instead of the internal nodes. When an encoding has the prefix property, we do not need spaces between binary digits in decoding.

Different binary trees correspond to different ways of encoding. Thus the tree in Figure 5.4(b) represents the following encoding:

A	00
B	01
C	10
D	11

The same message "BAD" would be represented by

01 00 11

B A D

Here every letter is represented by exactly two digits while previously some letters were represented by one digit and others by three digits.

We want to have a system such that we need the minimum number of digits, on the average, to transmit messages. Thus letters used frequently should be represented by short sequences and letters seldom used should be represented by long sequences.

Thus we associate a positive weight w_i with every letter V_i which indicates the relative frequency, and define the weighted path length of a tree to be

$$\Sigma w_i l_i$$

If the letters A, B, C, D have weights 4, 1, 2, 3 respectively, then the weighted path length of the tree in Figure 5.4(a) is

$$\begin{aligned} & w_A \cdot 1 + w_B \cdot 3 + w_C \cdot 3 + w_D \cdot 2 \\ & = 4 \cdot 1 + 1 \cdot 3 + 2 \cdot 3 + 3 \cdot 2 \\ & = 19. \end{aligned}$$

While the weighted path length of the tree in Figure 5.4(b) is

$$\begin{aligned} & w_A \cdot 2 + w_B \cdot 2 + w_C \cdot 2 + w_D \cdot 2 \\ & = 4 \cdot 2 + 1 \cdot 2 + 2 \cdot 2 + 3 \cdot 2 \\ & = 20 . \end{aligned}$$

We shall refer to the "weighted path length" as the cost, since the weighted path length is, in a sense, the cost of the encoding.

For given weights of the leaves, a binary tree with minimum cost is called an *optimum binary tree.* Th algorithm which constructed an optimum binary tree is due to Huffman [9] and people usually refer to the optimum binary tree as *Huffman's tree.*

Now we shall describe the Huffman algorithm. Let the weights of the n leaves be $w_1, w_2, \ldots, w_n$ where

$$w_1 \leqslant w_2 \leqslant \ldots \leqslant w_n \ .$$

Replace the two smallest nodes by a node with weight $w_1 + w_2$ and do this recursively for the n−1 weights

$$(w_1 + w_2), w_3, \ldots, w_n \ .$$

The final single node with weight $(w_1 + w_2 + \ldots + w_n)$ is the root of the binary tree.

If we apply the algorithm to the set of weights 4, 1, 2, 3, we first combine the two smallest weights 1 and 2 and obtain a set of three weights

$$4, 3, 3 .$$

Then the two smallest weights 3 and 3 are combined, and we have two weights

$$4, 6$$

and so the final tree looks like Figure 5.5.

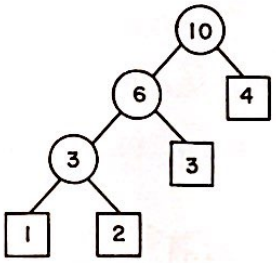


Figure 5.5

Note that we have written the weight of a node inside the node and the sum of the weights of the n−1 internal nodes

$$(3) + (6) + (10) = 19$$

is the weighted path length of the tree, a fact which can be proved in general.

Lemma 1 The sum of the weights of the n−1 internal nodes is equal to the cost of the binary tree.

Proof. By induction.

A larger numerical example with combined weights underlined is shown on page 168.

The resulting Huffman tree is shown in Figure 5.6.

We can easily prove the validity of Huffman's algorithm as follows. Given any binary tree, there must be an internal node of maximum path length. If this internal node V_i has two sons V_a and V_b which are not the smallest nodes, we could interchange the positions of w_1 and w_2 with the current two sons of V_i without increasing the cost of the tree. In the resulting tree, we have essentially n−1 leaves with weights

$$(w_1 + w_2), w_3, \ldots, w_n$$

2,	2,	2,	3,	4,	6,	6,	7,	9,	12,	13
	4,	2,	3,	4,	6,	6,	7,	9,	12,	13
	4,		5,	4,	6,	6,	7,	9,	12,	13
			5,	8,	6,	6,	7,	9,	12,	13
				8,	11,	6,	7,	9,	12,	13
				8,	11,		13,	17,	12,	13
					11,		13,	17,	23,	13
							13,	17,	23,	13
								17,	23,	13
									40,	13
										26
										26
										66

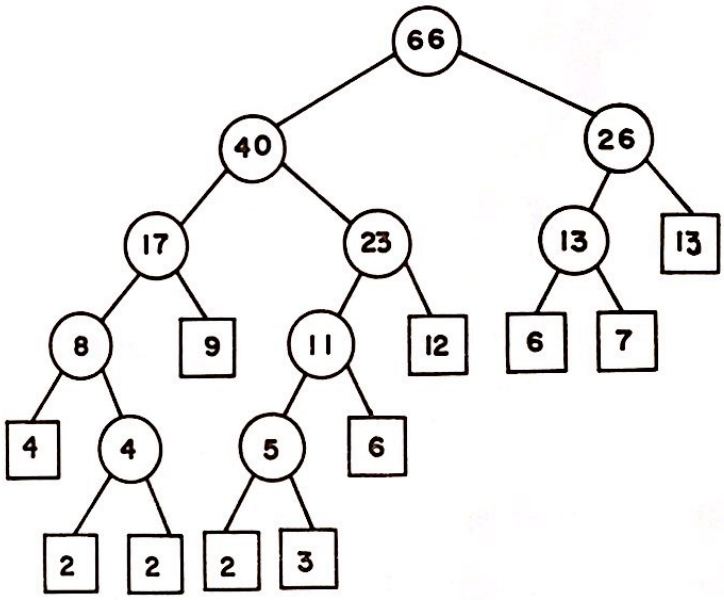


Figure 5.6

The proof is therefore completed by induction.

If we stop the Huffman construction after creating m internal nodes ($m < n-1$), then the result is a *forest.* (A forest is just a set of trees.) We shall call the forest an m-sum forest if it is obtained after creating m internal nodes.

We can define the cost of a forest as $\Sigma w_i l_i$ where l_i is the path length of every node in its trees. For example, the 4-sum forest of the leaves in Figure 5.6 is shown in Figure 5.7.

The cost of the 4-sum forest is

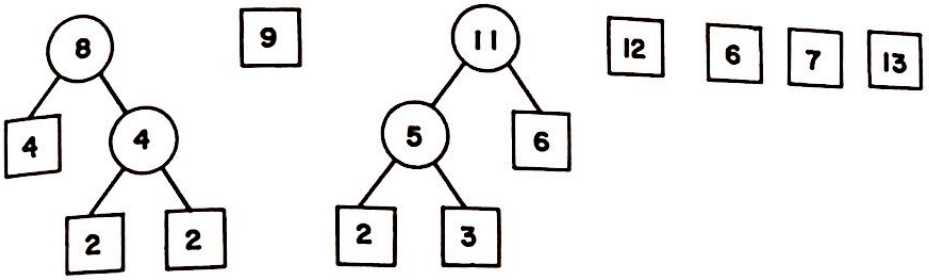


Figure 5.7

$$2 \cdot 2 + 2 \cdot 2 + 4 \cdot 1 + 2 \cdot 2 + 3 \cdot 2 + 6 \cdot 1 = 28$$

which is also equal to the sum of the weights of circular nodes created so far

$$④ + ⑧ + ⑤ + ⑪ = 28$$

Lemma 2 The Huffman algorithm gives the minimum cost m-sum forest for $1 \leqslant m \leqslant n-1$.

Proof. The proof is by induction on m. It is clear that Huffman's algorithm gives an optimum 1-sum forest. Assume that the lemma is true for k-sum forests and we now try to construct an optimum (k+1)-sum forest.

In this (k+1)-sum forest, there must be an internal node V_i of maximum path length. If the two sons of V_i do not have the smallest weights w_1 and w_2, then we can interchange the two sons of V_i with the nodes w_1 and w_2 without increasing the cost of the forest. Hence this new (k+1)-sum forest, which combines w_1 and w_2, is also optimum. Clearly, this (k+1)-sum forest can be optimum if and only if the k sum forest on the weights

$$(w_1 + w_2), w_3, \ldots, w_t$$

is optimum. However, by induction, the Huffman algorithm generates the optimum k-sum forest. Q.E.D.

We can define t-ary trees analogously to binary trees, where every internal node has exactly t sons and every square node has no sons.

It is easy to see that the Huffman construction of combining the t smallest nodes will give an optimum t-ary tree.

There is no explicit formula which gives the cost (of an optimum binary tree) as a function of the weights of the leaves. Consider two sets of nodes, both sets have the same number of nodes and the same total weight. For example, the sets (A) and (B) both have four nodes and total weight 20.

(A) 1, 3, 6, 10

(B) 2, 5, 5, 8

If we build Huffman's tree on (A), denoted by T_A, and Huffman's tree on (B), denoted by T_B, which tree has the higher cost? The following lemma decides this issue. We shall use $|T_A|$ to denote the cost of the tree T_A.

Lemma 3 (Extreme Set Lemma). Let T denote the optimum tree built on a set of nodes with weight w_i and T_u be the optimum tree built on a set of nodes with weight w_i'. If the following conditions are satisfied, then the cost of T is less than or equal to the cost of T_u.

(i) $\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} w_i'$

(ii) $w_1 \leqslant w_2 \leqslant \ldots \leqslant w_n$ and $w_1' \leqslant w_2' \leqslant \ldots \leqslant w_n'$

(iii) $w_i + \delta_i = w_i'$ $(i = 1,\ldots,k-1)$ and $w_i - \delta_i = w_i'$ $(i = k,\ldots,n)$.

$(\delta_j \geqslant 0,\ j = 1,\ldots,n)$.

Roughly speaking this lemma says that if the total weight of two sets are the same, the set with more extreme weights will give a tree of less or equal cost.

Proof. Assume that in the minimum cost tree, the path length for w_j' is l_j' $(j = 1,\ldots,n)$ then it follows from (ii) that

$$l_1' \geqslant l_2' \geqslant \ldots \geqslant l_n'. \tag{1}$$

It follows from (i) and (iii) that

$$\sum_{j=1}^{k-1} \delta_j - \sum_{j=k}^{n} \delta_j = 0$$

and

$$\sum_{j=1}^{k-1} \delta_j - \sum_{j=k}^{m} \delta_j > 0 \qquad (m < n) \tag{2}$$

Now
$$\begin{aligned}
|T_u| &= w_1' l_1' + w_2' l_2' + \ldots + w_k' l_k' + \ldots + w_n' l_n' \\
&= (w_1 + \delta_1) l_1' + \ldots + (w_k - \delta_k) l_k' + \ldots + (w_n - \delta_n) l_n' \\
&= \sum_{j=1}^{n} w_j l_j' + [\delta_1 l_1' + \delta_2 l_2' + \ldots + \delta_{k-1} l_{k-1}' - \delta_k l_k' - \ldots - \delta_n l_n'] \\
&= \sum_{j=1}^{n} w_j l_j' + l_n' [\delta_1 + \delta_2 + \ldots + \delta_{k-1} - \delta_k - \ldots - \delta_n] \\
&\quad + (l_{n-1}' - l_n') [\delta_1 + \delta_2 + \ldots + \delta_{k-1} - \delta_k - \ldots - \delta_{n-1}] \\
&\quad \vdots
\end{aligned}$$

$$
\begin{aligned}
&\quad + (l_1' - l_2')\,\delta_1 \\
&= \sum_{j=1}^{n} w_j l_j' + s' \\
&= |T''| + s' \qquad \text{where } s' \geqslant 0 . \\
&\geqslant |T|
\end{aligned}
$$

Intuitively, the proof of this lemma is to remove the leaves w_i' from the tree T_u and attach the leaves w_i to the corresponding places. This will create a tree T" cheaper than T_u. Since T is the optimum tree for w_i, $|T| \leqslant |T''| \leqslant |T_u|$.