Theorem 10.4 $mH_s(\mathbf{p}) \leqslant n_s(\mathbf{p}^m) < mH_s(\mathbf{p}) + 1$. Theorem 10.5 follows by dividing by m and taking limits. ■

Theorem 10.5 is satisfying theoretically, since it tells us that the source $\mathbf{p}$ can indeed be represented faithfully using (perhaps slightly more than) $H_s(\mathbf{p})$ s-ary symbols per source symbol. It leaves something to be desired from a constructive viewpoint, however, since it relies on the weak construction of Theorem 10.4. In the next section we shall remedy this situation by presenting a technique which will enable us to construct the best possible codes for the sources $\mathbf{p}^m$.

10.4 The Construction of Optimal UD Codes (Huffman's Algorithm)

According to Theorem 10.4, $n_s(\mathbf{p})$ lies somewhere between $H_s(\mathbf{p})$ and $H_s(\mathbf{p}) + 1$, and this estimate is adequate for some purposes (e.g., the proof of Theorem 10.5). But it is natural to wonder about the exact value of $n_s(\mathbf{p})$ for a fixed s and $\mathbf{p}$. In this section we present an algorithm due to David Huffman that shows not only how to compute $n_s(\mathbf{p})$, but also how to construct a UD (indeed a prefix) code with average length $n_s(\mathbf{p})$.

Before giving a formal description of Huffman's algorithm, we shall work an example. Throughout this section we refer to a UD s-ary code for $\mathbf{p}$ whose average length is $n_s(\mathbf{p})$ as an *optimal* code for $\mathbf{p}$.

Example 10.5. Let $s=4, \mathbf{p}=(.24,.21,.17,.13,.10,.07,.04,.03,.01)$. The first step in Huffman's algorithm is to replace the probability vector $\mathbf{p}$ with a simpler one $\mathbf{p}'$, which is obtained from $\mathbf{p}$ by combining the three smallest probabilities in $\mathbf{p}$. Thus $\mathbf{p}'=(.24,.21,.17,.13,.10,.08,.07)$ after the components are rearranged into decreasing order. Since $\mathbf{p}'$ is still complicated, we reduce $\mathbf{p}'$ still further by combining the four smallest probabilities in $\mathbf{p}'$ and obtain $\mathbf{p}''=(.38,.24,.21,.17)$. Figure 10.1 gives a schematic diagram of these reductions. The reason why we combined three probabilities when going from $\mathbf{p}$ to $\mathbf{p}'$ and four when going from $\mathbf{p}'$ to $\mathbf{p}''$ will appear as the theory develops; for the moment just accept it.

It is of course clear how to construct an optimal code for $\mathbf{p}''$: the code $C''=\{0,1,2,3\}$ achieves $n_4(\mathbf{p}'')=1$. Starting from this trivial code, we now can work backward and construct optimal codes C' and C for $\mathbf{p}'$ and $\mathbf{p}$ by "expanding" the code C'' in a simple way.

First we construct an optimal code for $\mathbf{p}'$. Notice that three of the probabilities (viz.,.24,.21,.17) were not changed in the reduction from $\mathbf{p}'$ to $\mathbf{p}''$. The rule in this case is that in the expansion of C'' into C' the corresponding codewords do not change, either. However, probability .38

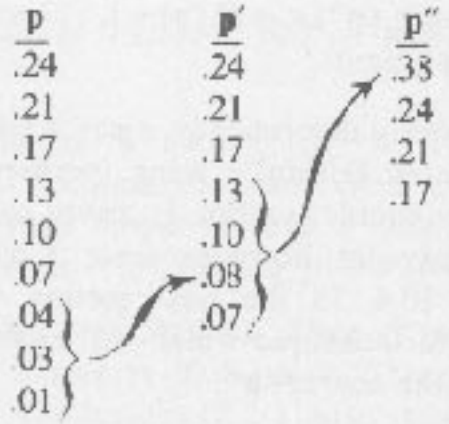

Figure 10.1. The successive reductions of **p**.

in $\mathbf{p}''$ expands into four probabilities (.13, .10, .08, .07) in $\mathbf{p}'$. Here the rule is that the codeword (0) for .38 in C'' expands into four codewords (00, 01, 02, 03) in C' (see Fig. 10.2). The resulting code, according to Theorem 10.7 below, is optimal for $\mathbf{p}'$.

The construction of C from C' proceeds similarly: every codeword but 02 corresponds directly to a single probability in $\mathbf{p}$, so the corresponding codewords do not change. However, 02 expands into 020, 021, 022 (again see Fig. 10.2). Thus allegedly $C=\{1,2,3,00,01,03,020,021,022\}$ is an optimal code for $\mathbf{p}$, and $n_4(\mathbf{p})=1\cdot(.24+.21+.17)+2(.13+.10+.07)+3(.04+.03+.01)=1.46$. ■

The preceding example is typical of the general Huffman algorithm. $\mathbf{p}$ is successively reduced to $\mathbf{p}', \mathbf{p}''$, etc., until a final reduction $\mathbf{p}^{(s)}$ with exactly s probabilities is reached. The obvious optimal code $\{0, 1, \ldots, s-1\}$ for $\mathbf{p}^{(s)}$ is then "expanded" in the above way until an optimal code for $\mathbf{p}$ is obtained. The only mysterious feature of the algorithm is the computation of the number of probabilities that must be combined in the reduction of $\mathbf{p}$ to $\mathbf{p}'$. If we denote this number by s', it turns out that s' is determined uniquely by the following two conditions:

$$s' \in \{2, 3, \ldots, s\}, \tag{10.2}$$

$$s' \equiv r \pmod{s-1}. \tag{10.3}$$

For example (cf. Example 10.5), if $s=4, r=9$, we get $s'=3$. If $s=2$, then $s'=2$ for all $r \geqslant 2$, and so this complication is absent for binary codes.[5] If $s=3$, then $s'=2$ if r is even and $s'=3$ if r is odd. Note that the number of probabilities in $\mathbf{p}'$ is $r'=r-s'+1$, which by Eq. (10.3) is congruent to 1 (mod $s-1$). Hence after the first reduction s' will always be equal to s, and so it is necessary to compute s' using Eqs. (10.2) and (10.3) only once, no matter how many reductions of $\mathbf{p}$ are required.

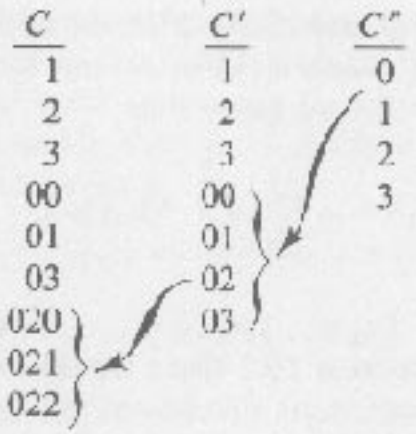

Figure 10.2. The synthesis of an optimal code for **p** (Cf. Fig. 10.1.)

Now we have described Huffman's algorithm; our remaining task—which is surprisingly tricky—is to show that it works, i.e. that it produces an s-ary prefix code for **p** whose average length is as small as possible. The following theorem is crucial; it guarantees that there is always an optimal prefix code for **p** in which the last s' words are of the same length, and agree except for their last component.

To fix our ideas, let us now assume that the probabilities $p_0, p_1, \ldots, p_{r-1}$ are arranged in decreasing order: $p_0 \geqslant p_1 \geqslant \cdots \geqslant p_{r-1}$. We consider prefix codes for **p** over the alphabet $\{0, 1, \ldots, s-1\}$ whose words are denoted by $\sigma_0, \sigma_1, \ldots, \sigma_{r-1}$, with lengths $n_i = |\sigma_i|, i = 0, 1, \ldots, r-1$.

THEOREM 10.6. *If $r \geqslant 2$, there exists an optimal s-ary prefix code for* **p** *with the following two properties:*

(a) $n_0 \leqslant n_1 \leqslant \cdots \leqslant n_{r-1}$.

(b) *The last s' (see Eqs. (10.2) and (10.3)) codewords are identical except for their last component, that is, there exists a string σ of length $n_{r-1} - 1$ such that:*

$$\sigma_{r-s'} = \sigma * 0,$$
$$\sigma_{r-s'+1} = \sigma * 1,$$
$$\vdots$$
$$\sigma_{r-1} = \sigma * (s'-1).$$

Proof. For a given code $\{\sigma_0, \sigma_1, \ldots, \sigma_{r-1}\}$ for **p**, it is clear that if $i < j$ and $|\sigma_i| > |\sigma_j|$, interchanging σ_i and σ_j cannot increase the average length $\sum p_i |\sigma_i|$. Hence there exist optimal s-ary codes for **p** such that (a) holds. In the rest of the proof, "optimal" code will mean a code of minimal average length that also satisfies (a).