One solution that does not always work! (extended Huffman)

Letter	Probability	Codeword
s_1	0.95	0
s_2	0.02	11
s_3	0.03	10

Table 1: Huffman code for three-letter alphabet; H = 0.335 bits/symbol; l_{avg} = 1.05 bits/symbol; redundancy = 0.715 bits/symbol or 213% of entropy.

Letter	Probability	Code
s_1s_1	0.9025	0
s_1s_2	0.0190	111
s_1s_3	0.0285	100
s_2s_1	0.0190	1101
s_2s_2	0.0004	110011
s_2s_3	0.0006	110001
s_3s_1	0.0285	101
s_3s_2	0.0006	110010
s_3s_3	0.0009	110000

Table 2: The Huffman code for the extended alphabet; l_{avg} = 1.222 bits/new symbol or l_{avg} = 0.611 bits/original symbol; redundancy = 72% of entropy; redundancy drops to acceptable values for N=8 (alphabet size = 6561).

Comparision of Huffman and arithmetic coding

$$H(s) \leq l_{avg}^{H} \leq H(s) + \frac{1}{m}$$

$$H(s) \leq l_{avg}^{A} \leq H(S) + \frac{2}{m}$$

- **Huffman seems to have an advantage. However, it requires building the entire code for all possible sequences of length m (k=16, m=2 $\rightarrow$ codebook size $= 16^{20}$!)**
- **In practice, we can make m large for arithmetic but not for Huffman coder $\Rightarrow$ for most sources we can get rates closer to the entropy using arithmetic coding than by using Huffman coding (except when $p_i = 2^{-k}$)**
- **arithmetic coding best suited for sources with small alphabet (e.g., facsimile) and highly unbalanced probabilities**
- **easy to implement a system with multiple arithmetic codes ($\rightarrow$ JBIG)**
- **easier to adapt arithmetic codes to changing input statistics (local structure $\rightarrow$ JBIG)**