17 Huffman Codes

The most common way to represent letters and numbers in computing is by using the ASCII code,

American **S**tandard **C**ode for **I**nformation **I**nterchange

This is based on a string of seven 'bits' where each bit can be either '1' or '0'. For example,

Character	*ASCII Code*
A	100 0001
B	100 0010
C	100 0011
D	100 0100
...	...
...	...
1	011 0001
2	011 0010
3	011 0011
...	...

The system can also be used for punctuation marks and symbols; the full list is given in the Appendix.

Exercise 1

Using a seven bit construction, how many characters can be coded?

This system is not particularly efficient for situations where, as with calculators, memory is limited, or when we are trying to condense data (as in the many compression programs that are vitally important for internet transmission of files).

However, when transmitting, for example, a message, it is more efficient to use a code which has shorter lengths (i.e. number of bits used) for the most frequently used letters and longer lengths for the least used letters. This is the basis for the Huffman coding system, which was developed by David A Huffman as PhD student at Massachusetts Institute of Technology in 1952.

We illustrate the method with an example that uses just five letters.

Example 1

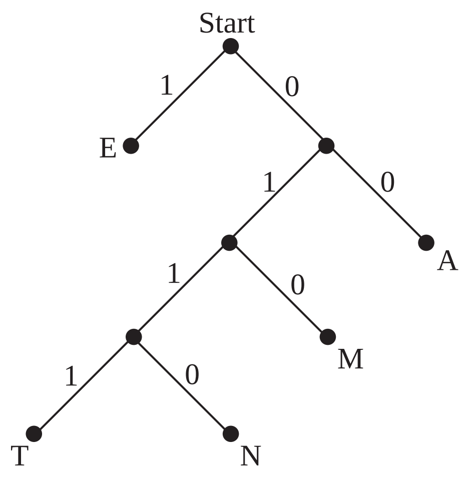
Construct a Huffman code for the five letters

E A M N T

which are listed in decreasing order of frequency of use.

Solution

You need to allocate the shortest codes to the letters with the highest frequency, so E is allocated to the code 1, A to 00, etc. This is best illustrated using a tree diagram.



Branching to the left is coded '1' and to the right, '0'. This gives the code

Letter	*Code*	*Length*
E	1	1
A	00	2
M	010	3
N	0110	4
T	0111	4

and is particularly suitable if the frequencies of N and T are about the same.

Example 2

Use the diagram or list to decode

0 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 1 1 0

Solution

Using the tree diagram, follow the code until a letter is reached and then start again.

0 1 1 1	0 0	0 1 0	1	0 1 0	0 0	0 1 1 0
T	A	M	E	M	A	N

Exercise 2

Decode the following using the Huffman codes above.

a) 01000011000011110111100

b) 0101101110111100010000111011110110

With this restricted alphabet, the question remains as to whether it is more efficient than a normal binary code. With just 5 letters, we would require 3 bits from a binary code, for example:

Letter	*Code*	*Length*
E	001	3
A	010	3
M	011	3
N	100	3
T	101	3

So the average length of letter transmitted using the Huffman code is 3.

Suppose, though, that the frequency of use of the letter is given by

E - 40%; A - 30%; M - 20%; N - 5%; T - 5%

We now calculate the average word length using these frequencies.

Example 3

Find the average word length using the Huffman code.

Solution

Letter	*Code*	*Length*	*Frequency*	*Length $\times$ frequency proportion*
E	1	1	0.4	0.4
A	00	2	0.3	0.6
M	010	3	0.2	0.6
N	0110	4	0.05	0.2
T	0111	4	0.05	0.2
			Total	2.0

So this example shows that the average length is now 2.0, which is a significant reduction.

Activity 1

Can you design other Huffman codes for 5 letters? When would you use them?

So there are a number of possibilities and this number increases as the number of words increases.

Activity 2

a) Design four different Huffman codes if exactly 8 letters are used.

b) Design a Huffman code if the only codewords used are as shown opposite and all words are used equally frequently.

BUS	CUPS	MUSH	PUSS
SIP	PUSH	CUSS	HIP
PUP	PUPS	HIPS	

Appendix

ASCII codes

Character	***Code***	
Space	0 1 0	0 0 0 0
0	0 1 1	0 0 0 0
1	0 1 1	0 0 0 1
2	0 1 1	0 0 1 0
3	0 1 1	0 0 1 1
...	...	...
...	...	...
9	0 1 1	1 0 0 1
+	0 1 0	1 0 1 1
−	0 1 0	1 1 0 1
=	0 1 1	1 1 0 1
A	1 0 0	0 0 0 1
B	1 0 0	0 0 1 0
...	...	...
O	1 0 0	1 1 1 1
P	1 0 1	0 0 0 0
Q	1 0 1	0 0 0 1
...	...	...
...	...	...
Z	1 0 1	1 0 1 0

ASCII code