Implementation of Huffman Source Coding
Information Theory : Project 1

Name : Dinakar P Roll No. : **EE04S053** Date : Feb 02, 2005

Aim:

To design a source code for an *iid* binary source that outputs a 0 with probability 0.95 and a 1 with probability 0.05 using Huffman coding technique.

Algorithm:

With a source emitting bits with probabilities mentioned above, the expected length per bits is 1 when we take one bit per symbol. To achieve better average length and closer to the source's entropy, we should take symbols with more bits. Let N be the number of bits per symbol. Using the source's probabilities, the steps to do source coding using Huffman coding technique is given below.

Step 1: Calculate the probabilities of each symbol and sort it in descending order.

The sorted array is of length 2^N.

Step 2: Calculate the sum of last two elements in the sorted array. Replace the sum in the last but one location and NULL the last location. Sort the new array in descending order and note the location of the calculated sum in the sorted array. Perform this operation for 2^N-2 times. Finally, we have an array size of two.

Step 3: Assign code '0' and '1' for the final probabilities.

Step 4: Now we trace back and assign code word, whenever there is a node split, to the computed Code string, we append a '0' and '1' respectively to the two branches. Finally, the symbol's code is obtained.

Calculation:

Source Entropy:

$E(X) = -(p0 * \log_2(p0) + p1 * \log_2(p1))$

$= -(0.95 * \log_2(0.95) + 0.01 * \log_2(0.05))$

$= 0.2864$

Source Coding:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Let N (No. of bits per symbol) = 3;

Symbol	Probability	Code Word
"000"	0.85737	"0"
"001"	0.045125	"100"
"010"	0.045125	"101"
"011"	0.002375	"11100"
"100"	0.045125	"110"
"101"	0.002375	"11101"
"110"	0.002375	"11110"
"111"	0.000125	"11111"

Expected Length per bit

$= (p_0*\text{length}(C_0) + p_1*\text{length}(C_1) + ...+p_{2N-1}*\text{length}(C_{2N-1}))/N$

$= 0.4333$

Let N (No. of bits per symbol) = 6

Symbol	Probability	Code Word
"000000"	0.73509	"0"
"000001"	0.038689	"111"
"000010"	0.038689	"1000"
"000011"	0.0020363	"11010001"
"000100"	0.038689	"1001"
"000101"	0.0020363	"11010010"
"000110"	0.0020363	"11010011"
"000111"	0.00010717	"110100000101"
"001000"	0.038689	"1010"
"001001"	0.0020363	"11010100"
"001010"	0.0020363	"11010101"
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
"001011"   0.00010717  "1101000000110"
"001100"   0.0020363   "11010110"
"001101"   0.00010717  "1101000000111"
"001110"   0.00010717  "1101000001000"
"001111"   5.6406e-006 "11010000010010000"
"010000"   0.038689    "1011"
"010001"   0.0020363   "11010111"
"010010"   0.0020363   "11011000"
"010011"   0.00010717  "1101000001001"
"010100"   0.0020363   "11011001"
"010101"   0.00010717  "1101000001010"
"010110"   0.00010717  "1101000001011"
"010111"   5.6406e-006 "11010000010010001"
"011000"   0.0020363   "11011010"
"011001"   0.00010717  "1101000001100"
"011010"   0.00010717  "1101000001101"
"011011"   5.6406e-006 "11010000010010010"
"011100"   0.00010717  "1101000001110"
"011101"   5.6406e-006 "11010000010010011"
"011110"   5.6406e-006 "11010000010010100"
"011111" 2.9688e-007 `11010000010011111111"
"100000"   0.038689    "1100"
"100001"   0.0020363   "11011011"
"100010"   0.0020363   "11011100"
"100011"   0.00010717  "1101000001111"
"100100"   0.0020363   "11011101"
"100101"   0.00010717  "1101000000000"
"100110"   0.00010717  "1101000000001"
"100111"   5.6406e-006 "11010000010010101"
"101000"   0.0020363   "11011110"
"101001"   0.00010717  "1101000000010"
"101010"   0.00010717  "1101000000011"
"101011"   5.6406e-006 "11010000010010110"
"101100"   0.00010717  "1101000000100"
"101101"   5.6406e-006 "11010000010010111"
"101110"   5.6406e-006 "11010000010011000"
"101111" 2.9688e-007 "11010000010011111000
"110000"   0.0020363    "11011111"
"110001"   0.00010717   "1101000000101"
"110010"   0.00010717   "1101000000110"
"110011"   5.6406e-006 "11010000010011001"
"110100"   0.00010717  "1101000000111"
"110101"   5.6406e-006 "11010000010011010"
"110110"   5.6406e-006 "11010000010011011"
"110111"   2.9688e-007
                  "11010000010011111001"
"111000"    0.00010717
                  "1101000001000"
"111001"        5.6406e-006
                  "11010000010011100"
"111010"        5.6406e-006
                  "11010000010011101"
"111011"        2.9688e-007
                  "11010000010011111010"
"111100"        5.6406e-006
                  "11010000010011110"
  "111101"        2.9688e-007
                  "11010000010011111011"
  "111110"        2.9688e-007
                  "11010000010011111100"
  "111111"        1.5625e-008
                  "11010000010011111101"
```

Expected Length per bit

$= (p_0 * \text{length}(C_0) + p_1 * \text{length}(C_1) +$

$.....+ p_{2N-1} * \text{length}(C_{2N-1}))/N$

$= 0.3162$

Expected Length per bit Vs No. of bits per symbol:

In the figure below, the expected length per bit is plotted against the No. of bits per symbol. It is clear that the expected length per bit tends to the source Entropy as the bits per symbol increases.