Huffman Coding

- Proposed by Dr. David A. Huffman
- *"A Method for the Construction of Minimum Redundancy Codes"*
- Applicable to many forms of data transmission
  - example: text files

- Usage:
  - Fax Machines
  - ASCII

- Variations on ASCII
  - min number of bits needed
  - cost of savings

- Huffman coding is a form of statistical coding
- Not all characters occur with the same frequency!
- Yet, why all characters allocated the same amount of space -ASCII
  - 1 char = 1 byte, be it e or S or Z

- Therefore, any savings in tailoring codes to frequency of character?
- Code word lengths are no longer fixed like ASCII.

- Code word lengths vary and will be shorter for the more frequently used characters !!.

Intuition

- Consider the following short text:
  Oh Man What A Scene.

- Count up the occurrences of all characters in the text

- O→1 , h→2, M→1, a→2, n→2, W→1, t→1, A→1, S→1, c→1, SPACE→4

- Create binary tree nodes with character and frequency of each character
- Place nodes in a priority queue
  - The lower the occurrence, the higher the priority in the queue.

Huffman Encoding Algorithm

1. The source symbols are listed in order of decreasing probability (i.e. high to low).
2. The two source symbols of *lowest probability* are assigned a 0 and a 1.
3. Combine (add) these two symbols of low probability and generate a new probability. [as a new symbol !]
4. The probability of the new symbol is placed in the list in accordance with its value [*decreasing order as in step 1*]
5. Repeat until you are left with a final list of source statistics (symbols) of *only two* for which a 0 and a 1 are assigned. [*step 2~ step 4*]
6. Trace the sequence of 0s and 1s assigned to that symbol as well as its successors *backwards*.→ It's the codeword

Example:
We have 5 Symbols with probabilities 0.4, 0.2, 0.2, 0.1, and 0.1. Encode using Huffman coding scheme.

Symbol	Stage 1	Stage2	Stage 3	Stage 4
S1	**0.4**	**0.4**	**0.4** (top)	**0.6**
S2	**0.2**	**0.2** (top)	**0.4**	**0.4**
S3	**0.2**	**0.2**	**0.2**	
S4	**0.1**	**0.2**		
S5	**0.1**			

Symbol	Probability	codes
S1	0.4	00
S2	0.2	10
S3	0.2	11
S4	0.1	010
S5	0.1	011

- a) Find the avg. code word length, b) Entropy, c) variance of avg. code word length

- <u>Solution</u>: a) avg. code word length:
- $L' = 0.4\,(2) + 0.2\,(2) + 0.2\,(2) + 0.1\,(3) + 0.1\,(3) = 2.2$

- Entropy= H(S)=
- $0.4 \log (1/0.4) + 0.2 \log (1/0.2) + 0.2 \log (1/0.2) + 0.1 \log (1/0.1) + 0.1 \log (1/0.1) = 2.121$

Symbol	Probability	codes
S1	0.4	00
S2	0.2	10
S3	0.2	11
S4	0.1	010
S5	0.1	011

- Note:
- i) The entropy is lower than Avg. code word length. [as per theory]
- $$H(S) \leq L' < H(S)+1$$

- ii) The 0 and 1 *can be* assigned the other way too. Even, the combined symbol can be placed below the same probability symbol.

- *Now, code words may differ, their length too may differ !. But, the avg. codeword length remains the same !!*

- Thus, we measure the variance of code-word length.

- $\sigma^2 = \sum_{k=0}^{K-1} p_k(l_k - L')^2$

$= 0.4(2\text{-}2.2)^2 + 0.2(2\text{-}2.2)^2 + 0.2(2\text{-}2.2)^2 + 0.1(3\text{-}2.2)^2 + 0.1(3\text{-}2.2)^2$

$= 0.16$

Ex: [method 2: different choices]

We have 5 Symbols with probabilities 0.4, 0.2, 0.2, 0.1, and 0.1.
Encode using Huffman coding scheme.

Symbol	Stage 1	Stage2	Stage 3	Stage 4
S1	**0.4**	**0.4**	**0.4**	**0.6** 0
S2	**0.2**	**0.2**	**0.4** 0	**0.4** 1
S3	**0.2**	**0.2** 0	**0.2** 1	
S4	**0.1** 0	**0.2** 1		
S5	**0.1** 1			

Symbol	Probability	codes
S1	0.4	1
S2	0.2	01
S3	0.2	000
S4	0.1	0010
S5	0.1	0011

- Solution: a) avg. code word length:
- L’ = 0.4 (1) + 0.2 (2) + 0.2 (3) + 0.1 (4) + 0.1 (4) = 2.2

- $\sigma^2 = \sum_{k=0}^{K-1} p_k (l_k - L')^{2}$

$= 0.4(1-2.2)^{2} + 0.2(2-2.2)^{2} + 0.2(3-2.2)^{2} + 0.1(4-2.2)^{2} + 0.1(4-2.2)^{2}$

$= 1.36$

- Takeaway: [Should move the combined symbol up to get min. variance]

- **Are we better ?**
  - 5 Symbols : S1S2S3S4S5
  - Huffman method: 10100000100011 → 14 bit [from method 2]
  - ASCII : 8*5= 40 bit.

- A **boiling question**: Which is better:

  → Shannon-Fano, Huffman..? **Try at Home**