**International Journal of Advanced Research and Development**
**ISSN: 2455-4030**
**Impact Factor: RJIF 5.24**
**www.advancedjournal.com**
**Volume 2; Issue 5; September 2017; Page No. 361-365**

# Source coding based on binary tree

**Shashi Kant Pandey, Vijay Dahiya**
Department of Business Administration, Maharaja Surajmal Institute, Janakpuri, Delhi**,** India

**Abstract**
Information data can be stored in many ways and scientists are always looking for new and better ways to store strings of data without any loss of information with as little space as possible. Optimization of length of information bits or code is playing a key role in design of any encoding scheme. Huffman coding scheme provide an optimal length code for any given probability distribution of information, which is discussed here in this article.

**Keywords:** graph theory, tree, entropy, efficiency, uncertainty, lossy encoding

## 1. Introduction
In 1948 Shannon produce a problem of information theory in his famous paper on information theory. Problem was that to find an algorithm to produce an optimal symbol code for any given probability distribution on a finite set. Here optimal means minimal expected length of a code. In his paper Shannon produce Shannon- Fano coding scheme, which does not produce an optimal code always. It was a challenging problem to develop such scheme which provides always an optimal length code.
It is interesting to say that in 1951 David Huffman a student of Fano produce an algorithm which is an optimal coding scheme and it is known as Huffman coding scheme. This scheme is based on a property of tree in Graph theory. Using a basic property of tree structure he proposed this scheme. In graph theory a tree, one and only one path between any two vertices, Huffman used this concept and produces this optimal scheme. It provides this scheme efficiency and accuracy. In other words we can say that it encodes the text in unique and concise patterns. An important characteristic of this encoding technique is its less storage capability. There are various schemes in which the methods of compression techniques are in use. One of the techniques, use for storage more optimally is to compress the files by taking advantage of redundancy or patterns, it may be able to "abbreviate" the contents in such a way to take up less space yet maintain the ability to reconstruct a full version of the original when needed. Such compression could be useful when trying to cram more information on a disk or to shorten the time needed to copy/send a file over a network. There are some known compression algorithms, which give compression formats, such as JPEG, MPEG, or MP3, are specifically designed to handle a particular type of data file. Some of the compression algorithms (e.g. JPEG, MPEG) are loss-decompressing; the compressed result doesn't recreate a perfect copy of the original. Such an algorithm compresses by "summarizing" the data. The summary retains the general structure while discarding the more minute details. Sound and video data may be acceptable for lossy encoding schemes because there is huge amount of data and some missing pixels does not affect in retrieving the original information. But for text data these lossy encoding schemes are not appropriate.

## 2. Prilliminaries
**Graph:** A graph G is representation of relation between two sets called vertices and edges. If E and V be the set of edges and vertices respectively then we denote the graph as G = (V, E).
**Path:** Sequence of continuous vertices in a graph is called path.
Connected Graph: A graph is connected graph, if there is a path between any two vertices of that graph.
**Walk:** A walk in a graph is a path or a sequence of vertices with no repetition.
**Cycle:** A closed walk is called a cycle in a graph. A graph which has no any cycle then it is called acyclic.
**Tree:** A connected acyclic graph is a tree. In a tree we always get a path from any two vertices and it is unique for any two vertices.

## 3. Definitions
### 3.1 Perfect secrecy
Achievement of perfect secrecy is always required in design of any cryptosystem. In practical scenario perfect secrecy means that Oscar can obtain no information about plaintext by observing the cipher text. This terminology can be described in terms of probability distribution also. It can be define as

Definition 3.1 (a): A cryptosystem has perfect secrecy if $Pr[x \mid y] = Pr[x]$ for all $x \in P, y \in C$. That is, the a posteriori probability that the plaintext is *x*, given that the cipher text y is observed, is identical to the a priori probability that the plaintext is *x*

### 3.2Entropy

The idea of entropy is introduced by Shannon at first time in his famous paper named "A Mathematical Theory of communication" in 1948. Entropy can be thought of as a mathematical measure of information or uncertainty, and is computed as a function of probability distribution.

Suppose we have probability distribution of some events. The amount of information we gain from occurrence of these particular events under given probability distribution or we say what is the uncertainty about the outcomes of events which are not yet occurred. So from definition it is clear that entropy is a function of probability distribution. If X is a random variable then entropy of X is denoted by H(X).

**Definition 3.2(a):** suppose ***X*** is a discrete random variable which takes on values from set X Then the entropy of the random variable ***X*** is defined to be the quantity

$$H(\boldsymbol{X}) = -\sum_{x \in X} pr[x] log_2 pr[x]$$

### 4. Compression of data & Source Coding Theorem

The primary objective is the compression of data by efficient representation of the symbols suppose a discrete memory less source (DMS) outputs a symbol every t seconds and each symbol is selected from a finite set of symbols $x_i$, i=1,2,.....l, occurring with probabilities P ($x_i$), i= 1,2,.....L, the entropy of this DMS in bits per source symbol is

$$H(X) = \sum_{j=1}^{L} P(x_i) log_2 P(x_i) \leq log_2 L$$

When each of information is coded or represented by some fixed length code then it is called a fixed length code. When this length is variable then it's called variable length code. Suppose a DMS outputs a symbol selected from a finite set of symbols $x_i$ i=1, 2... L. the number of bits R required for unique coding when L is a power of 2 is

$$R = \log_2 L$$

And, when L is not a power of 2, it is

$$R = [log_2 L] + 1$$

The fixed length code (FLC) for the English alphabet suggests that every letter in the alphabet is equally important (probable) and hence each one require same number of bits for representation. It appears that allotting equal number of bits to both the most frequent and less probable used letters is not an efficient way of representation. For resolving this problem we can represent more frequent letter by less numbers of bits and less frequent letters by more number of bits. In this manner encoded text have less overall bits. And this method is called variable length coding.

Prefix condition: when any one of codeword not forming the prefix of any other codeword then this condition is called prefix condition for that particular coding scheme. Such code is uniquely decidable.

**Theorem 4.1: (Source coding theorem)**: Let X be the set of letters from a DMS with finite entropy H(X) and $x_k$, k= 1,2,...L the output symbols, occurring with probabilities P($x_k$), k= 1,2,... L.

Given these parameters, it is possible to construct a code that satisfies the prefix condition and has an average length R that satisfies the inequality

$$H(X) \leq \bar{R} \leq H(X) + 1$$

**Proof:** Consider the lower bound of above inequality. The codeword of length $n_k$ $1 \leq k \leq L$, the difference $\bar{R} - H(X)$ can be expressed as

$$H(X) - \bar{R} = \sum_{k=1}^{L} p_k log \frac{1}{2^{p_k}} - \sum_{k=1}^{L} p_k n_k = \sum_{k=1}^{L} p_k log \frac{2^{-n_k}}{p_k}$$

We now make use of inequality $\ln x \leq x - 1$ to get

And by Craft inequality

$$H(X) - \bar{R} \leq log_{2^e} \sum_{k=1}^{L} p_k \left( \frac{2^{-n_k}}{p_k} - 1 \right) \leq log_{2^e} \sum_{k=1}^{L} p_k \left( 2^{-2_k} - 1 \right) \leq 0$$

There is equality holds if and only if $p_k = 2^{-n_k}$ for $1 \leq k \leq L$
Hence the lower bound is proved.

Now let us suppose that the codeword length $n_k$ such that $2^{-n_k} \leq p_k \leq 2^{-n_k+1}$. First consider $2^{-n_k} \leq p_k$ summing both sides over $1 \leq k \leq L$ gives us

$$\sum_{k=1}^{L} 2^{-n_k} \leq \sum_{k=1}^{L} p_k = 1$$

Which is the Kraft inequality for which there exists a code satisfying the prefix condition? Next consider $p_k \leq 2^{-n_k+1}$. Take logarithm on both sides to get

$$log_{2^{p_k}} < -n_k + 1,$$

Or $n_k < 1 - log_{2^{p_k}}$

On multiplying both sides by $p_k$ and summing over $1 \leq k \leq L$ we obtain

$$\sum_{k=1}^{L} p_k n_k < \sum_{k=1}^{L} p_k + \left( -\sum_{k=1}^{L} p_k log_{2^{p_k}} \right)$$

Or $\bar{R} < H(X) + 1$

Thus upper bound is proved. The source coding theorem tells us that for any prefix code used to represent the symbols from a source, the minimum number of bits required to represent the source symbols on an average must be at least equal to the entropy of the source. If we have found a prefix code that satisfies $\bar{R} = H(X)$ for certain source X, we must abandon further search because we cannot do any better. The theorem also tells us that a source with higher entropy (uncertainty) requires, on an average, more number of bits to represent the source symbols in terms of prefix code.

**Definition: 4.2** Efficiency: The efficiency of a prefix code is defined as η = $\frac{H(X)}{\bar{R}}$, so it is clear from source coding theorem that the efficiency of a prefix code η $\leq$ 1. Efficient representation of symbols leads to compression of data. Source coding is primarily used for compression of data and image.

**5. Huffman encoding algorithm**

Let given probability distribution is P = $\{p_i : 1 \leq i \leq n\}$

**Step1:** Short the $p_i's$ in decreasing order such that $p_1 \leq p_2 \leq \cdots \leq p_n$ and assume they are vertices of tree. These vertices are called as leaf of this tree and we start the extension of that tree from its leaf to its root.

**Step2:** Choose two minimum of $\{p_i : 1 \leq i \leq n\}$, these are $p_1$ & $p_2$

**Step3:** Find new vertices with entry are sum of {$p_1$ & $p_2$ } let it is $q_1$ & makes edge between$p_1 to\ q_1$ & $p_2\ to\ q_1$. In this way we can extend this tree.

**Step 4:** Make new set of probability distribution with removing two selected $p_i$ and include the new vertices $q_i$ in P.
**Step 5:** Do step2& step3 till the vertices end or when we rich at the root of the tree.
**Step 6:** For encoding start labeling the tree starting from root as left edge with 0 and right edge with 1.
**Step 7:** Choose path from the root to the pendent vertices to encode the character whose probability distribution is given. And in this way we can provide a unique code to every information very efficiently.

## 6. Huffman coding scheme example and Algorithm

Huffman coding uses 'variable length coding' which means that symbols in the data which we want to encode are converted to a binary symbol based on how often that symbol is used.

### 6.1. Example

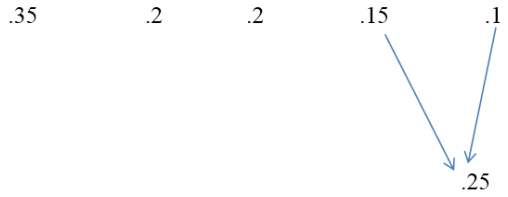
Let the probability distribution of five information or characters are as follows:

P = {.35,. 2,. 2,. 15,. 1}

**First step:** short P in descending order

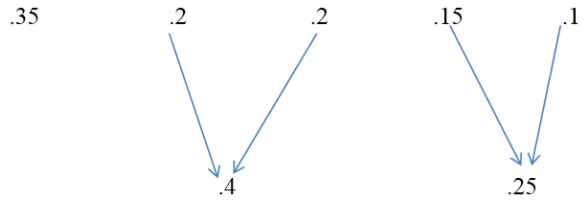
P($x$) = {.35,. 2,. 2,. 15,. 1}

**Second step:** Take smallest two of them and starting to build edge between both of them and sum of both probabilities:

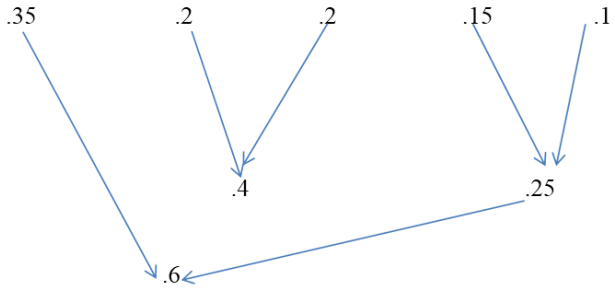



Again recursively do this process till all probability does not assign

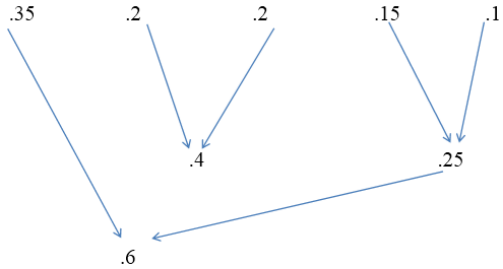
Choose again minimum of them and add them for next vertices of the tree, we can see the minimum of them is. 2 &. 2 so next vertices is attached with them and its entry is. 2+.2=.4




Again do recursively we get these steps

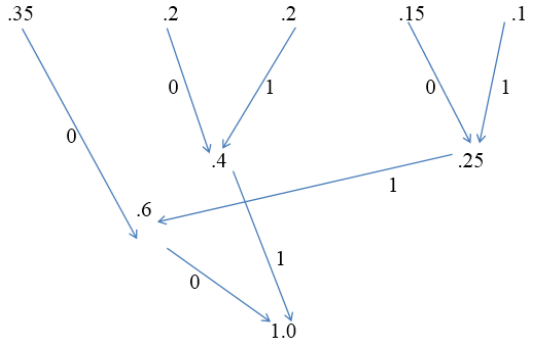



and




Recursive process of these steps gives a tree. Give level to all edges of this tree which are code or alphabets. After leveling all the edges of this tree we get the required optimal code for this probability distribution of information.

Give level of this tree with 0 or 1 in the way such that all left edge are leveled with 0 and all right edges corresponding to each vertices are leveled with 1,we get this figure




To produce code words for a character x follow the path from root of this tree to pendent vertices which is the probability corresponding to particular *x*.
We get code words for each one as follows

Cod words: 00 10 11 010 011
P(*x*):. 35. 2. 2. 15. 1

Since from each pair pendent vertices and root we have one and only one path so we get a unique code words for any " x"
Now we see that expected length of each code world is
L (*x*): 2 2 2 3 3
P (*x*): 35. 2. 2. 15. 1
P (*x*)*l(*x*). 7. 4. 4. 45. 3

L =expected length of this code is = $\sum P(x)*l(x)$ = 2.25 bits and we know that in binary value system the entropy of this P is $H_2(P)$ = 2.2016 bits

And we can see that H $\leq$ L ≤ H+1
So it satisfy source coding inequality and we can say that this scheme work like that the most probable information or character are represented by less bits and less probable character or information are represented by more bits so this algorithm can optimize the expected length and produce an optimal code.

**7. Conclusion**

Every coding scheme is efficient on the basis of its secrecy and low computation complexity. Encoding scheme which provide minimal length of code with no loss of data which improve the efficiency of encoding scheme. Huffman's algorithm is an example of a greedy algorithm and it is a method of storing strings of data as binary code in an efficient manner without losing the data. It work well in text and fax transmission.

**8. References**

1. Julie Zelenski. with minor edits by Keith Schwarz "Huffman Encoding and Data Compression"CS106B, spring, 2012.
2. Douglas R. Stinson Cryptography Theory and Practice.
3. Ranjan Bose. Information Theory Coding and Cryptography, T.M.H publication, 1-45.
4. Wikipedia the encyclopedia.
5. Harary. Graph Theory