# Programming Assignment Report 4

*Huffman Code*

(CSCI B505 Fall 19)

**Manisha Suresh Kumar**

**UID:2000529881**

11.08.2019

# INTRODUCTION

Huffman Code is a compression technique having varied lengths of bits for characters based on the frequency of their occurrences. It is an efficient method to save bits when compared to fixed length encoding. In this assignment, we will create Huffman Code for a book in text file and compare the total bits with that of fixed length encoding.

# IMPLEMENTATION

The Huffman Code was implemented for characters in the book 'Women of Belgium: Turning Tragedy to Triumph by Charlotte Kellogg' obtained from http://www.gutenberg.org/ebooks/60599.

**Note**:

- Encoding is done for characters having ASCII values between and including 32 and 127.

## Creating Initial List

First, we add all characters in the book and their frequencies in a Priority Queue. This is done in the function `find_frequency()`. The frequencies are the priorities and **only the characters in the in the book are considered**. So if a character, say 'z' doesn't occur anywhere in the book, then encoding is not done for it.

## Creating Huffman Code Tree

Now we can start creating the tree. For this we use the function `huffman_code_tree()`. Since we are using a priority queue, every time we pop, we get the characters with least two priorities and **in case of a tie, the Priority Queue in python selects according to the order in the queue**. A new node is created which is an object of type `Character` that has a left child, right child and a frequency that is the sum of frequencies of its children. This is done iteratively until only one element is left in the PQ that is the root. The characters are the leaves in the tree. Asymptotic Time Complexity is $O(nlogn)$.

## Generating Code

The last step is to generate codes for each character. This is done by `generate_code()`. We start from the root(the only element in priority queue) and travel recursively till the

leaves. Any traversal on the right side of the tree adds a '1' to the code and any on the left adds a '0'.

## ANALYSIS AND CONCLUSION

For the book, the most and least frequent character is ' ' (blank space) and '%' and their Huffman Codes are:

```
' ': 111
%: 100001010001110111
```

The following are the total number of bits for every character, each time they occur in the book for Huffman Code and fixed length coding and the number of bits saved:

```
Total number of bits in huffman code: 884349
Total number of bits in fixed code: 1382591
Bits saved: 498242
```

Hence, by assigning a shorter code for characters that occur frequently and a longer code for characters that occur much less frequently, we save a substantially large number of bits when compared to the standard 7-bit fixed length encoding that does not take into consideration frequencies.