# CS319 Algorithm Analysis
# Programming assignment 4 – Huffman Coding
# 40 points

Huffman coding is a scheme that assigns variable-length bit-codes to characters, such that the lengths of the codes depend on the frequencies of the characters in a typical message. As a result, encoded messages take less space (as compared to fixed-length encoding) since the letters that appear more frequently are assigned shorter codes. This is performed by first building a Huffman coding tree based on a given set of frequencies. From the tree, bit-codes for each character are determined and then used to encode a message. The tree is also used to decode an encoded message as it provides a way to determine which bit sequences translate back to a character.

Write a program to implement Huffman coding. It should do the following:
Accept a text file, possibly of more than one line.
Create a Huffman tree for this text file.
Create a Huffman code table.
Encode the message into binary.

You may assume that the messages are written in lower-case letters. The frequency table has 30-lines, where each line contains a letter (or a special character) followed by a space and a positive integer (string of digits). For the simplicity purposes, the only special characters are: `-' for space, `.' for period, `!' for new line, and `+' for end-of-message.

Sample input file, together with expected outputs can be found on course website.

**This project contains three parts, each of which has a different due date:**

Part1 (10 points): Read an input text file, create the frequency table. (Easy)

Part II (20 points): Create the Huffman tree based upon frequency table, the create the Huffman code table based upon the tree. (Difficult)

Part III (10 points): Using the Huffman code table to encode the input text file. (Easy)

**What needs to be submitted:**

For each part of the project, please submit the source code(s), the readme document file explaining how to run your code, and the sample input/output file that you used to test your code.

Here are the **sample outputs** for each phase/part:

**Part I sample output:**

Frequency table

c 4
d 2
- 15
a 11
b 7
e 20
! 4
+ 1
f 5
. 0
z 0
w 0
...
(all other letters have frequency 0)

**Part II sample output:**

Huffman Codes

a 101
b 001
c 0001
d 00001
e 11
f 1001
g ...
h ...
...
z ...
- 01
. ...
! 1000
+ 000001

**Part III sample output:**

**Encode example**

Message:

a bad

face

encoded message

10101001 10100001 10001001 10100011 11000000 00100000