CSCI 665 Foundations of Algorithms
Lab assignment 2
Algorithm

Srushti Kotak
sk7735

Step 1: Find the frequency of each character in the input string.

Create a list which has (frequency of character 1, character 1, frequency of character 2, character 2, frequency of character 3, character 3, ...) : *freq_characters*
And a list which has all the characters in the input string : *list_ch*

Step 2: Generate nodes for the huffman tree

nodes is a list within list. Parse the *freq_characters* list and add the first two entries as one entry in the *nodes* list, then chop the first two elements of the *freq_characters* list.

Step 3: Generate huffman tree

Append the *nodes* list to the *huffman_tree* list.
Sort the *nodes* list and then append a 0 at index 0 and 1 at index 1.
Then we start joining the nodes, i.e, we add the frequencies at 0th index and concat the characters at 1st index.
Update the *nodes* list.

Step 4: Continue step 3 till we reach the root of huffman tree.

Step 5: Encode

5a. Create binary codes for each node in huffman tree.

If the length of character in *list_ch* is 1 , then it's binary code is 0.
Else, for each character in the *list_ch* we generate a chcode, by parsing each node in the huffman list, add the 0 or 1 at the 2nd index of the node.
This way we get the binary codes for each node and the encoded string will be *bitstring* in 0, 1 format.

5b. Convert the bitstring to actual binary data type to find the length of compressed file.

Step 6: Decode

6a. Decode

For each character in the *bitstring,* append it to the *code* and then for each 1st index of the node in the *binary_codes,* we check if the code is same as that node. If it is same, the we decode that with the string at 0th index of the *binary_codes,* else we increment the index and check for all binary_codes.

6b. Repeat till we have decoded the entire bitstring.