SIT221 –DATA STRUCTURES AND ALGORITHMS

PROJECT2 DUE 11:59PM, SEPTEMBER 24TH

SUBMISSION INSTRUCTIONS

Please review submission instructions available in the same folder

PROJECT TASKS

HUFFMAN CODING – 15 MARKS

Huffman coding is a lossless data compression algorithm. The core idea is to assign codes of different lengths to characters based on the frequencies of the characters. The most frequent character gets the shortest code length and the least frequent character gets the longest code length. However, Huffman coding guarantees that that there is no ambiguity when decoding a generated bit stream, i.e. there are no characters, e.g. x and y such that the code of x is a prefix of the code of y.

Your tasks will include

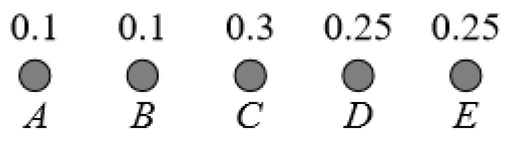
1) Building a Huffman tree from input data
2) Coding, i.e. assigning codes to characters in the input data
3) Decoding, i.e. given a code, identifying the corresponding character.

Following is an example about Huffman coding

Assume that we have an input data including characters A, B, C, D, and E. Assume that the frequencies of those characters are as f(A)=f(B)=0.1, f(C)=0.3, f(D)=f(E)=0.25.

Step 1

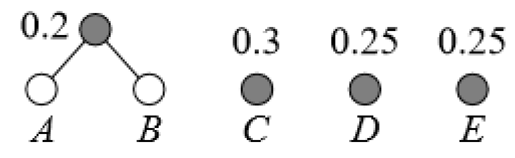
Create a list of nodes in which each node corresponds to a character and is weighted by its frequency.

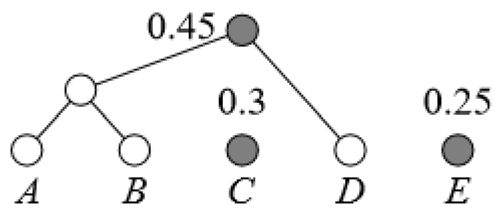


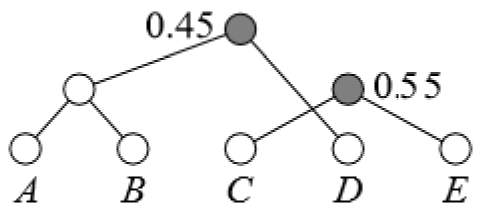
Step 2

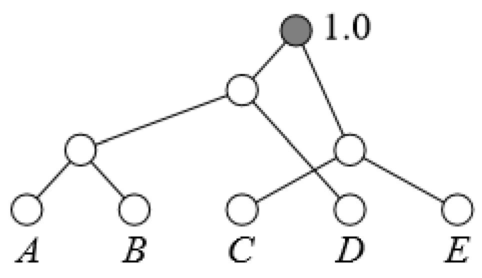
From the list, group 2 nodes with lowest frequencies and create an internal node weighted by the sum of the weights of the two nodes. These two nodes are then deleted from the list and the new internal node is added in the list.

This step is repeated until the list is empty.



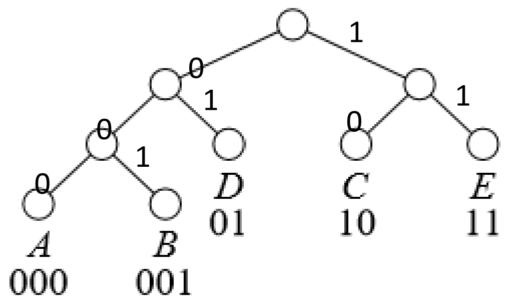






Step 3

Assigning codes to characters: All the characters should be the leave nodes in the tree. The code of a leave node can be obtained by traversing the tree from the root to that leave node. Edges on the left of a node are coded by 0s and edges on the right of a node are coded by 1s (see figure below).



Input of your program includes three 3 file names: **input.txt**, **encoded_input.txt**, and **decoded_input.txt**. **Note that all the file names must be given as the arguments of your program.**

The input data is given in **input.txt** file, each line in the file contains one letter (varying from A, B,…, Z and in uppercase). The example above can be represented in **input.txt** file as follows,

```
D
A
E
C
B
D
C
A
B
C
C
C
D
D
C
D
E
E
E
E
```

You are to read **input.txt** file, build the Huffman tree, encode the data, and save the encoded data into **encoded_input.txt**. In **encoded_input.txt** file, each character is represented by its Huffman code and on a text line. For example, the **encoded_input.txt** file for the data in the **input.txt** file above should look like

```
01
000
11
10
001
01
10
000
001
10
10
10
01
01
10
01
11
11
11
11
```

You are then to read the **encoded_input.txt** file, use the Huffman tree built from the **input.txt** file, decode the encoded data in the **encoded_input.txt** file, and write the decoded result to **decoded_input.txt**. Note that the files: **decoded_input.txt** and **input.txt** must contain identical content.

Bonus

You would have 10 marks as bonus if you could develop a compression data application using Huffman coding. Your developed program takes 3 file names (input data file, encoded data file, and decoded data file) as arguments. These files contain input data, encoded data, and decode data respectively. Note that your program should not limit to any specific file types (e.g. the input file can be image files, video files, etc.).