Course Project Guidelines

Please pay attention to the following notes.

- The course project report and the related source codes must be submitted before the deadline. No late work will be accepted.
- In order to submit your document effectively, you are required to compress your course report and related documents, and name the compressed file as **StudentID_Name_Course_Project_B.xxx**.
- Replace StudentID_Name with your own personal details.
- The course project is conducted on the individual basis. If any plagiarism is found, your project report will not be graded at all.

1 Implemenation of Huffman Coding and Shannon Coding

This programming project is designed for the memento to Steve Jobs, D.A. Huffman and C. E. Shannon. The objective of this programming assignment is to deeply understand the Huffman coding and Shannon coding method. In your source code, please write down proper comment on each line which you think is important. You may discuss ideas with others in the class, but you need to implement your own program. **Do not copy from other's source code.** When you finish, create a ZIP file with your student ID as the file name. In the ZIP file, please include the source files and the corresponding executable files. Submit your ZIP file to the course email address before the due date.

In the above zipped file, please provide a **readme.txt** file which explains each files in your zipped file and how to run your program.

Detailed requirements:

1. Read Steve Jobs' 2005 Stanford Commencement Address "You've got to find what you love" as in the file "Steve_Jobs_Speech.txt".

2. Collect the statistics of the letters, punctuation, space in "Steve_Jobs_Speech.doc".

3. Compute the entropy of "Steve_Jobs_Speech.doc".

4. Apply the Huffman coding method and Shannon coding method for "Steve_Jobs_Speech.doc". Output the letters/punctuation/space and their Huffman codewords and Shannon codes, respectively.

5. Compute the average code length of "Steve_Jobs_Speech.doc" using your Huffman codes and Shannon codes, respectively.

6. Note: For the Huffman code, please develop your code to generate a Q**-ary Huffman code, where** Q **is an input variable which can be random chosen.**

Hint:

Input: r: the number of the source symbols; P: the probability distribution of source symbols.

Output: Output the Huffman codewords w_i corresponding to the source symbols s_i.

initialization;

if $r == 2$ **then**

  return $s_0 \longmapsto 0$, $s_1 \longmapsto 1$

else

  sort $\{p_i\}$ in Descending order;

  reduce source: create a new symbol s' to replace s_{r-1}, s_{r-2} with the probability $p' = p_{r-1} + p_{r-2}$;

  Call the Huffman algorithm recursively to obtain the codes of $s_0, \ldots, s_{r-3}, s'$ as $w_0, \ldots, w_{r-3}, w'$ with the corresponding probability distribution $p_0, \ldots, p_{r-3}, p'$;

  return $s_0 \longmapsto w_0$, $s_1 \longmapsto w_1$, ..., $s_{r-3} \longmapsto w_{r-3}$, $s_{r-2} \longmapsto w'_0$, $s_{r-1} \longmapsto w'_1$.

end

Algorithm 1: The Binary Huffman Algorithm

Input: r: the number of the source symbols;

$P = \{p(s_i)\}, i = 1, \ldots, r$: the probability distribution of source symbols.

Output: Output the Shannon codewords w_i corresponding to the source symbols s_i.

initialization;

sort $\{p(s_i)\}$ in Descending order; **for** $i = 1 \to r$ **do**

  $F(s_i) \leftarrow \sum_{k=1}^{i-1} p(s_k)$;

  $l_i \leftarrow \left\lceil \log \frac{1}{p(s_i)} \right\rceil$;

  Code $F(s_i)$ using binary;

  Take l_i digits after the dot as the codeword for the source symbols s_i.

end

Algorithm 2: The Shannon Coding Algorithm