**Project No.5: Huffman Coding (Font: Calibri, 14pt)**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student No. | Email | Responsibility |
| 付新宇 | 14346005 | fuxy203@163.com |  |
| 陈嘉民 | 14346002 | Chenjm69@mail2.sysu.edu.cn |  |

1. **Introduction (1 point)**

What are the problems? Briefly describe how you solve them. (Font: Calibri, 12pt)

This project is to design compression and decompression programs based on Huffman Coding. The idea of Huffman Coding is to minimize the weighted expected length of the code by means of assigning shorter codes to frequently-used characters and longer codes to seldom-used code.

The programs we design can compress and decompress text file consisting of 128 ASCII characters.

The compression procedure is as follows:

1. Open file, count and store the frequencies of different characters.
2. Construct Priority Queue and then Huffman Tree.
3. Calculate Huffman Encode Table.
4. Encode the file, store Huffman Encode Table and encoded to the desired file.

The decompression procedure is as follows:

1. Open file, recreate Huffman Tree based on Huffman Encode Table.
2. Decode the file based on the tree and store it to the desired file.

The encoded text form:



1. **Analysis and Design (3 points)**

Data structures you use, the decision behind selecting them;

Describe the most important algorithms, again with a justification of your decision;

Time complexity of your algorithms.

Important data structures

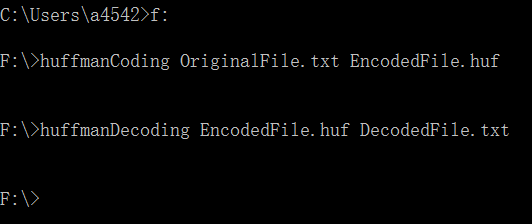
1. Priority Queue:
   1. Convenient for saving characters’ information in frequency order
   2. Convenient for building Huffman Tree
2. Huffman Tree: Basic structure for Huffman coding
3. String
   1. Rather large container of char
   2. Using for binary input or output
   3. Convenient for setting fixed form need in coding

Important algorithms

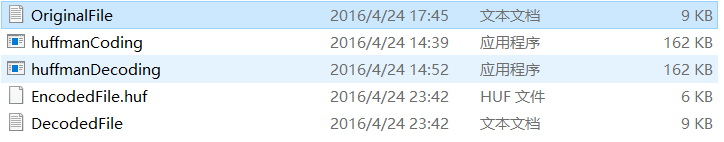
1. Create Huffman Tree
2. Tree Traversal
3. Build Huffman Tree
4. Coding
5. Decoding
6. **Test (1 points)**

Describe the test plan -- how the program/system was verified. Show your test cases and results within screenshots.

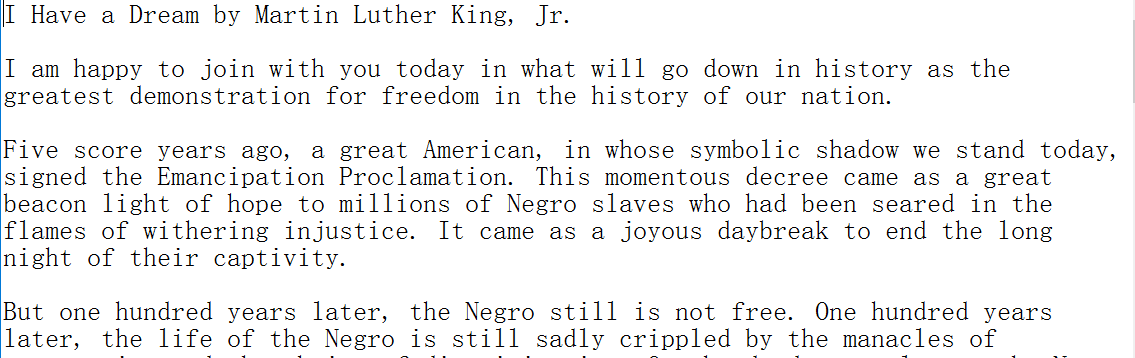
The commands:



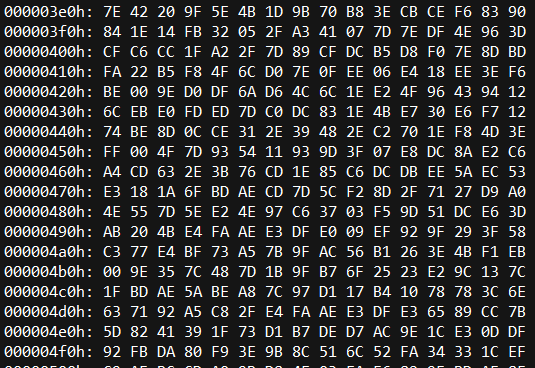
Execution results:



The original text:



Part of the encoded text (represented in hexadecimal):



1. **Conclusion and Discussion (1 point)**

What have you achieved and have not achieved?

Any other highlights about your design and implementation (you may get some bonus credits)

Achieved：Coding & decoding of file

(coding information in encoded file &

decoding file according to coding information in encoded file)

Highlights：

Saving Huffman coding table in the encoded file instead of frequency so there is no need to know the coding rule when decoding.

1. **Appendices (4 points)**

The code listing (Your code should be well commented.)

1. **References**

Books and websites

[1] Y.Daniel Liang. Introduction to Programming with C++ (Third Edition) [M］．机械工业出版社

[2] Unknown. “huffman encoding”:

http://www.cplusplus.com/forum/general/112141/

[3] Unknown. “std::priority\_queue”:

http://www.cplusplus.com/reference/queue/priority\_queue/