Experiment topic: **Design and Implementation of Huffman Coding and Decoding System**

1. Experimental content

Design and Implementation of Huffman Encoding / Decoding System (16 credit hours)

1. Problem description: Using Huffman coding for communication can greatly improve channel utilization, shorten information transmission time, and reduce transmission costs. However, this requires pre-coding the data to be transmitted by a coding system at the sending end, and decoding (decoding) the incoming data at the receiving end. For a duplex channel (that is, a channel that can transmit information in both directions), each end requires a complete encoding / decoding system. Try to design a Huffman encoding and decoding system for such a messaging station.

2. basic requirements

(1) Initialization (Initialization). Read characters and the weight of each character from the data file DataFile.data , and build a Huffman tree HuffTree ;

(2) Encoding (EnCoding). Use the built Huffman tree to encode the text in the file ToBeTran.data to form a message, and write the message in the file Code.txt ;

(3) Decoding (Decoding). Use the built Huffman tree to decode the code in the file CodeFile.data to form the original text, and store the result in the file Textfile.txt ;

(4) Output (Output). Output the characters that appear in DataFile.data and the frequency (or probability) that each character appears; Output ToBeTran.data and its message Code.txt ; Output CodeFile.data and its original text Textfile.txt ;

(5) Design independently according to the requirements of the topic, and write a design report as required after the design is completed .

2. Design idea:

1、When performing initialization

(1) Initialize the Huffman tree first, and set all values of the tree nodes to 0.

(2) First call the ReadFile function to read the letters and weights in DataFile.data. DataFile.data is designed to store characters and weight numbers into tree nodes respectively, and especially add a char letter for storage when defining struct tree nodes, and store weights into int weight.

(3) According to the content read, further create a Huffman tree

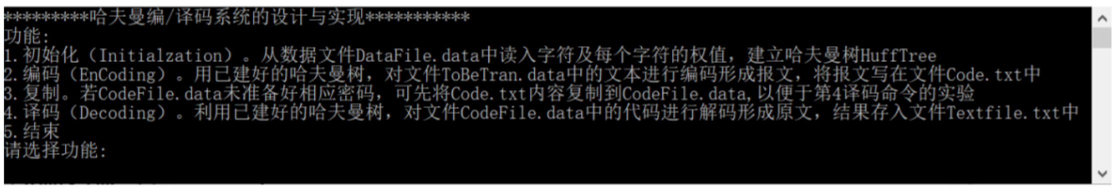
(4) According to the existing Huffman tree, the Huffman code of each character is reversely calculated from the leaf to the heel, and stored in the code table HC.

2. When coding

(1) Read all the characters in ToBeTran.data and store them in a temporary array. Count the length of the array, which is the total number of characters, and print the original content.

(2) Call the built Huffman tree HT, check the character and Huffman code information, and generate the corresponding Huffman code. Store these codes in another temporary array, print and display the coded results, that is, the requested message.

(3) Save the message (temporary array content) into Code.txt.
3. When decoding
(1) The program I designed includes a copy function. If CodeFile.data has no usable ciphertext based on the newly established Huffman code, you can use the copy function to copy the message in Code.txt to CodeFile.data In order to facilitate the next experiment, it also simulates the process of sending and receiving information.
(2) Read the ciphertext in CodeFile.data, store it in a temporary array, and count the length of the ciphertext.
(3) Check the ciphertext content character by character with the existing Huffman code, use the Huffman tree to find the corresponding characters, and store the characters into another temporary array.
(4) Print the original content and translation.
(5) End the program.
4. Design a friendly user interface to display each operation step.



```
*********哈夫曼编/译码系统的设计与实现***********
功能:
1.初始化（Initialzation）。从数据文件DataFile.data中读入字符及每个字符的权值，建立哈夫曼树HuffTree
2.编码（EnCoding）。用已建好的哈夫曼树，对文件ToBeTran.data中的文本进行编码形成报文，将报文写在文件Code.txt中
3.复制。若CodeFile.data未准备好相应密码，可先将Code.txt内容复制到CodeFile.data,以便于第4译码命令的实验
4.译码（Decoding）。利用已建好的哈夫曼树，对文件CodeFile.data中的代码进行解码形成原文，结果存入文件Textfile.txt中
5.结束
请选择功能:
```

3. Code:

The following code gives two versions: (single-file version and multi-file version)

Code (single-file version):

```
#include <stdio.h>
#include <string.h>
#include <iostream>

using namespace std;

typedef struct
{
    char letter;
    int weight; //The weight of the node
    int parent,lchild,rchild; //The parent of the node, the left child, the subscript of the right
child
}HTNode,*HuffmanTree; //Dynamically allocate an array to store the Huffman tree

typedef char ** HuffmanCode; //Dynamically allocate an array to store the Huffman code table
int ReadFile(HuffmanTree &HT,int n); //read file
void Select(HuffmanTree &HT, int n, int &a, int &b);
int CreateHuffmanTree(HuffmanTree &HT,int n); //construct Huffman tree HT
void CreatHuffmanCode(HuffmanTree HT,HuffmanCode &HC,int n); // Find the Huffman code of
each character from the leaf to the heel, and store it in the code table HC
```