哈尔滨工业大学计算学部

实验报告

课程名称：数据结构与算法

课程类型：专业基础（必修）

实验项目：树形结构及其应用

实验题目：哈夫曼编码与译码方法

实验日期：2022 年 10 月 20 日

班级：2103601

学号：2021112845

姓名：张智雄

设计成绩	报告成绩	指导老师
		李秀坤

一、实验目的

哈夫曼编码是一种以哈夫曼树（最优二叉树，带权路径长度最小的二叉树）为基础变长编码方法。其基本思想是：将使用次数多的代码转换成长度较短的编码，而使用次数少的采用较长的编码，并且保持编码的唯一可解性。在计算机信息处理中，经常应用于数据压缩。是一种一致性编码法（又称"熵编码法"），用于数据的无损压缩。要求实现一个完整的哈夫曼编码与译码系统。

二、实验要求及实验环境

实验要求：

1. 从文件中读入任意一篇英文文本文件，分别统计英文文本文件中各字符（包括标点符号和空格）的使用频率；
2. 根据已统计的字符使用频率构造哈夫曼编码树，并给出每个字符的哈夫曼编码（字符集的哈夫曼编码表）；
3. 将文本文件利用哈夫曼树进行编码，存储成压缩文件（哈夫曼编码文件）；
4. 计算哈夫曼编码文件的压缩率；
5. 将哈夫曼编码文件译码为文本文件，并与原文件进行比较。
6. 利用堆结构，优化的哈夫曼编码算法

实验环境：Windows11 操作系统+ VS Code 编译器

三、设计思想

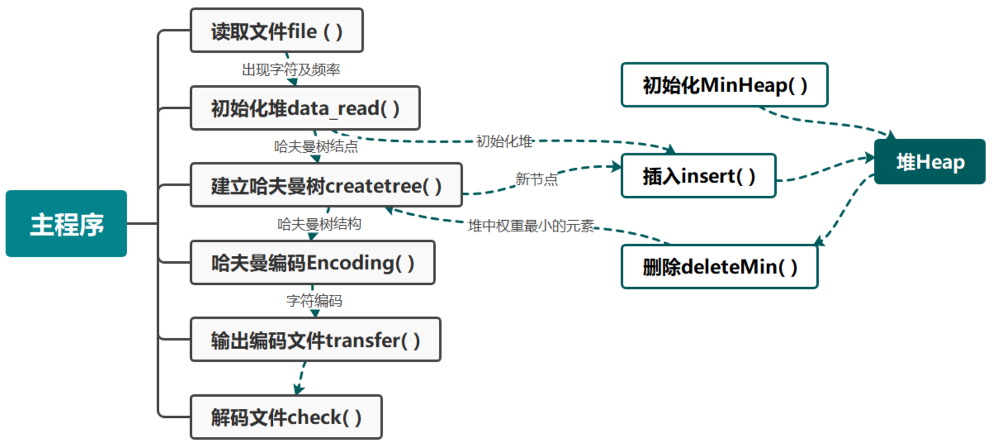
数据结构：

本实验中的堆结构以及哈夫曼树均采用顺序结构，即逻辑与物理结构相统一的结构，通过**结构体数组**的形式储存节点信息。具体使用数据类型及功能如下表：

序号	名称	功能	内含参数
1	**struct data**	**读取数据临时储存**	**文本出现字符char alphabet，该字符频率double frequency**
2	**struct node(HTNODE)**	**哈夫曼树节点**	**自身编号int self，左儿子int lchild ，右儿子int rchild， 父节点int parent， 储存字符char alphabet，字符权重double weight**
3	**struct code**	**编码结构**	**字符char ch，字符对应编码char bits[MAX+1];**
4	**struct heap(HEAP)**	**堆结构**	**顺序储存堆结构HTNODE heap[2*MAX];堆内数据个数int n;**

函数以及程序主要流程：

函数间调用关系以及程序主要流程图如下图：



各函数具体参数、功能以及返回值如下表：

序号	函数名	函数功能	函数参数	函数返回值
1	MinHeap( )	最小堆结构初始化	HEAP *minheap	无
2	insert( )	向堆结构中插入新节点	HEAP *minheap, HTNODE x	无
3	deleteMin( )	删除堆结构最小节点	HEAP *minheap	堆最小节点HTNODE
4	file ( )	读取文件信息	字符计数数组int x[], 信息储存数组struct data Youth[]	等长编码文件长度pre
5	data_read( )	文件信息转移到树节点，并初始化堆	信息储存数组struct data Youth[], HTNODE T[], HEAP *minheap	无
6	createtree( )	建立哈夫曼树	HTNODE T[], HEAP *minheap	无
7	Encoding( )	哈夫曼编码	(HTNODE T[], 编码信息数组struct code HAff[]	无
8	transfer( )	输出编码文件，返回压缩后的长度	编码信息数组struct code HAff[], 输出路径char path[]	压缩后文件长度now
9	check( )	解码文件	HTNODE T[], 输出路径char path[]	无

核心算法的主要步骤：

A. 堆结构的插入以及删除

插入：

1. 每次插入都是将先将新数据放在当前树的结尾位置，

2.　随后将其与其父节点数据相比较，若其权值小于其父节点，则交换两节点位置，重复此过程，直至小于当前父节点的权值停止。

3.　在插入过程中，最大交换次数为树的高度（log n）

删除：

堆中每次都只能删除堆顶元素。删除后需要重建堆，实际的操作是从根节点开始遍历（无需回溯），依次将左右儿子中的较小值赋值给其父节点，最后将未删除前的结尾节点赋给当前遍历到的尾节点。

B.　哈夫曼树的建立

每次从堆中弹出两个最小的哈夫曼树节点，作为新哈夫曼树节点的左右儿子（默认左儿子权值较小），新哈夫曼树节点的权值等于其左右儿子的权值之和，随后将新节点加入堆中，直至新节点权值为 1 时停止，此时哈夫曼树建立完成。

C.　哈夫曼编码以及解码

编码：编码过程，自哈夫曼树的叶节点由下而上遍历，通过 start 指针记录当前编码在数组中位置（初始为数组结尾），当为其父节点左儿子时，在编码数组里输入 0；反之为 1。随后 start 前移一位，当遍历至根节点时，将编码数组从 start 所指位置开始复制到编码节点中。

解码：自根节点由上而下开始遍历，读入‘0’则移动至其左儿子；‘1’为其右儿子，直至遍历到叶节点时输出字符，并将直至重置到根节点，直至文件读取结束。

D.　压缩率的计算

设文本内使用的字符种类为$n(n \leq 256)$，则使用等长编码长度$e = [log_2 n] + 1$（**取对后向上取整**）；设文本包含字符个数为X，则压缩率：

$$\eta = \frac{\text{H(实际哈夫曼译码文件长度)}}{eX}$$

四、测试结果

测试用例：

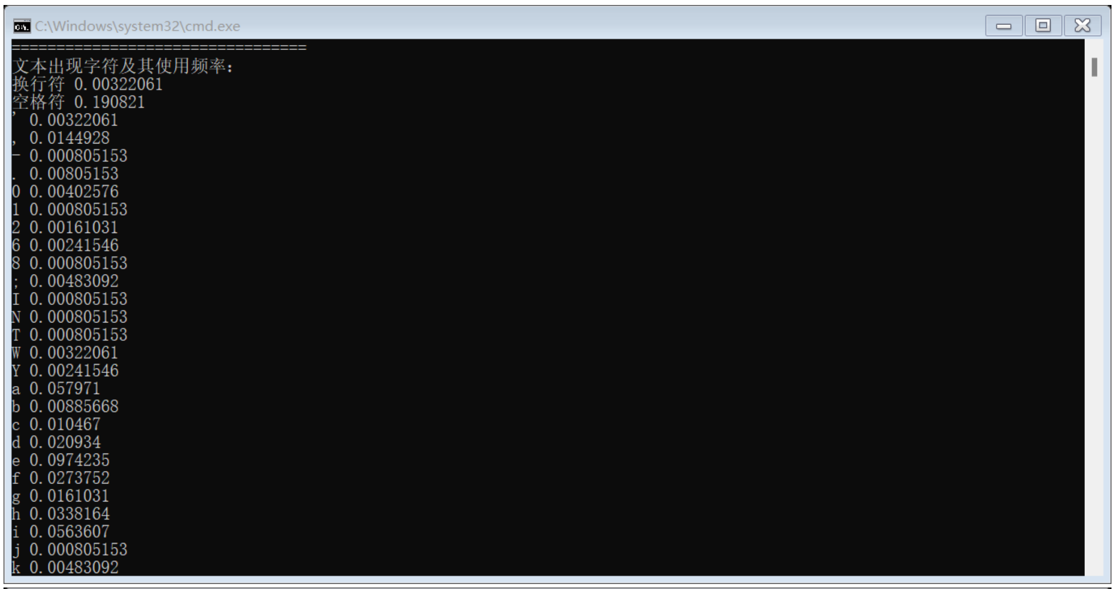
Youth is not a time of life; it is a state of mind; it is not a matter of rosy cheeks, red lips and supple knees; it is a matter of the will, a quality of the imagination, a vigor of the emotions; it is the freshness of the deep springs of life. Youth means a temperamental predominance of courage over timidity, of the appetite for adventure over the love of ease. This often exists in a man of 60 more than a boy of 20. Nobody grows old merely by a number of years. We grow old by deserting our ideals.

Years may wrinkle the skin, but to give up enthusiasm wrinkles the soul. Worry, fear, self-distrust bows the heart and turns the spirit back to dust.

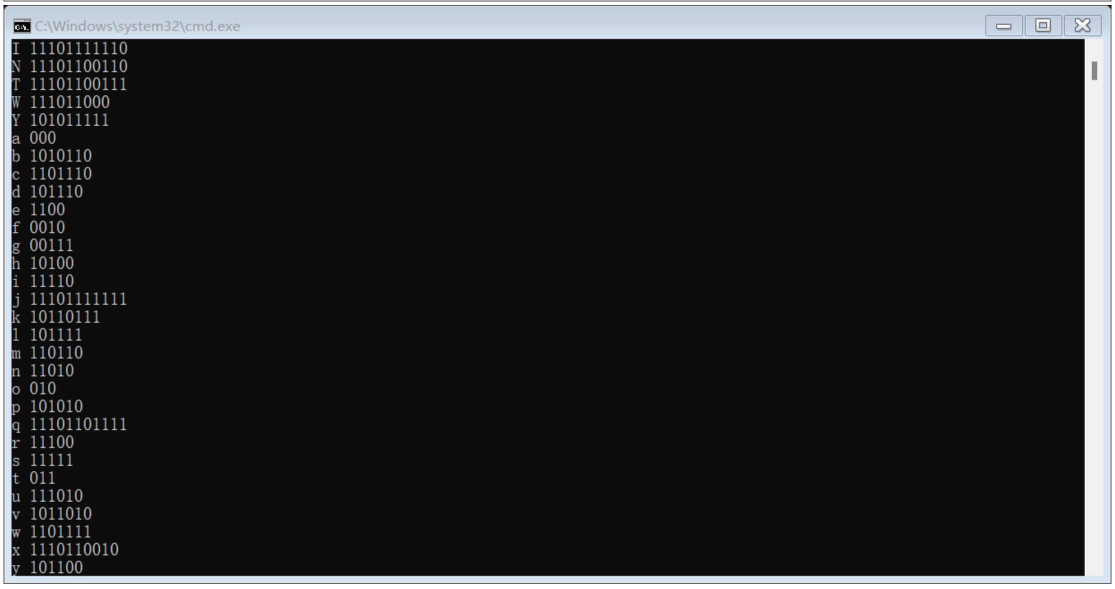
Whether 60 or 16, there is in every human being's heart the lure of wonders, the unfailing appetite for what's next and the joy of the game of living. In the center of your heart and my heart, there is a wireless station; so long as it receives messages of beauty, hope, courage and power from man and from the infinite, so long as you are young. When your aerials are down, and your spirit is covered with snows of cynicism and the ice of pessimism, then you've grown old, even at 20; but as long as your aerials are up, to catch waves of optimism, there's hope you may die young at 80.

编码结果：（均为同次程序运行结果）

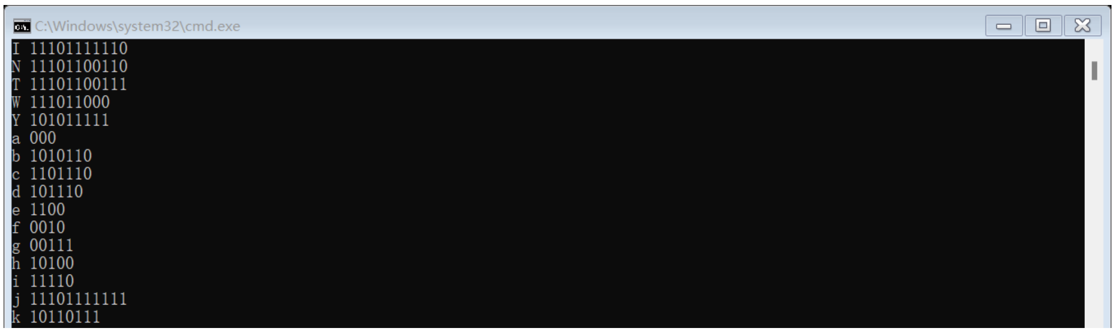
1．运行结果：



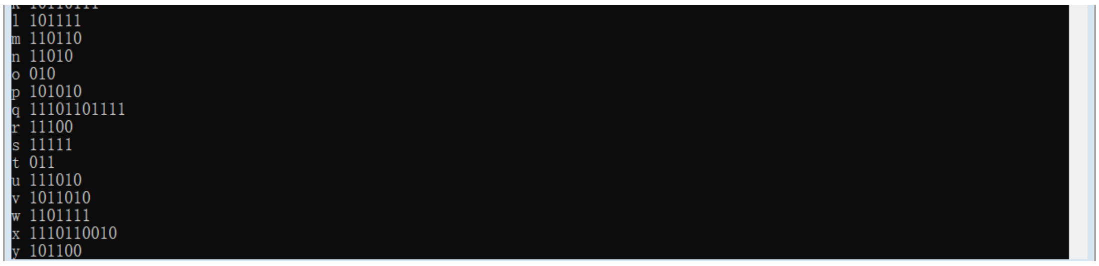
```
C:\Windows\system32\cmd.exe
================================
文本出现字符及其使用频率:
换行符 0.00322061
空格符 0.190821
' 0.00322061
, 0.0144928
- 0.000805153
. 0.00805153
0 0.00402576
1 0.000805153
2 0.00161031
6 0.00241546
8 0.000805153
; 0.00483092
I 0.000805153
N 0.000805153
T 0.000805153
W 0.00322061
Y 0.00241546
a 0.057971
b 0.00885668
c 0.010467
d 0.020934
e 0.0974235
f 0.0273752
g 0.0161031
h 0.0338164
i 0.0563607
j 0.000805153
k 0.00483092
```



```
C:\Windows\system32\cmd.exe
I 11101111110
N 11101100110
T 11101100111
W 111011000
Y 101011111
a 000
b 1010110
c 1101110
d 101110
e 1100
f 0010
g 00111
h 10100
i 11110
j 11101111111
k 10110111
l 101111
m 110110
n 11010
o 010
p 101010
q 11101101111
r 11100
s 11111
t 011
u 111010
v 1011010
w 1101111
x 1110110010
y 101100
```



```
C:\Windows\system32\cmd.exe
I 11101111110
N 11101100110
T 11101100111
W 111011000
Y 101011111
a 000
b 1010110
c 1101110
d 101110
e 1100
f 0010
g 00111
h 10100
i 11110
j 11101111111
k 10110111
```



```
l 101111
m 110110
n 11010
o 010
p 101010
q 11101101111
r 11100
s 11111
t 011
u 111010
v 1011010
w 1101111
x 1110110010
y 101100
```

2. 整理结果：

字符	概率	哈夫曼编码
换行符	0.00322061	111011010
空格符	0.190821	100
’	0.00322061	111011110
,	0.0144928	00110
-	0.000805153	11101101100
.	0.00805153	11101110
0	0.00402576	10101110
1	0.000805153	11101101101
2	0.00161031	1110111110
6	0.00241546	101011110
8	0.000805153	11101101110
;	0.00483092	10110110
I	0.000805153	11101111110
N	0.000805153	11101100110
T	0.000805153	11101100111
W	0.00322061	111011000
Y	0.00241546	101011111
a	0.057971	000
b	0.00885668	1010110
c	0.010467	1101110
d	0.020934	101110
e	0.0974235	1100
f	0.0273752	0010
g	0.0161031	00111
h	0.0338164	10100
i	0.0563607	11110
j	0.000805153	11101111111
k	0.00483092	10110111
l	0.0217391	101111
m	0.0241546	110110
n	0.047504	11010
o	0.0676329	010
p	0.0161031	101010
q	0.000805153	11101101111
r	0.0507246	11100
s	0.0563607	11111
t	0.0668277	011
u	0.0249597	111010
v	0.010467	1011010
w	0.0120773	1101111
x	0.00161031	1110110010
y	0.020934	101100

3. 编码文件（节选）：

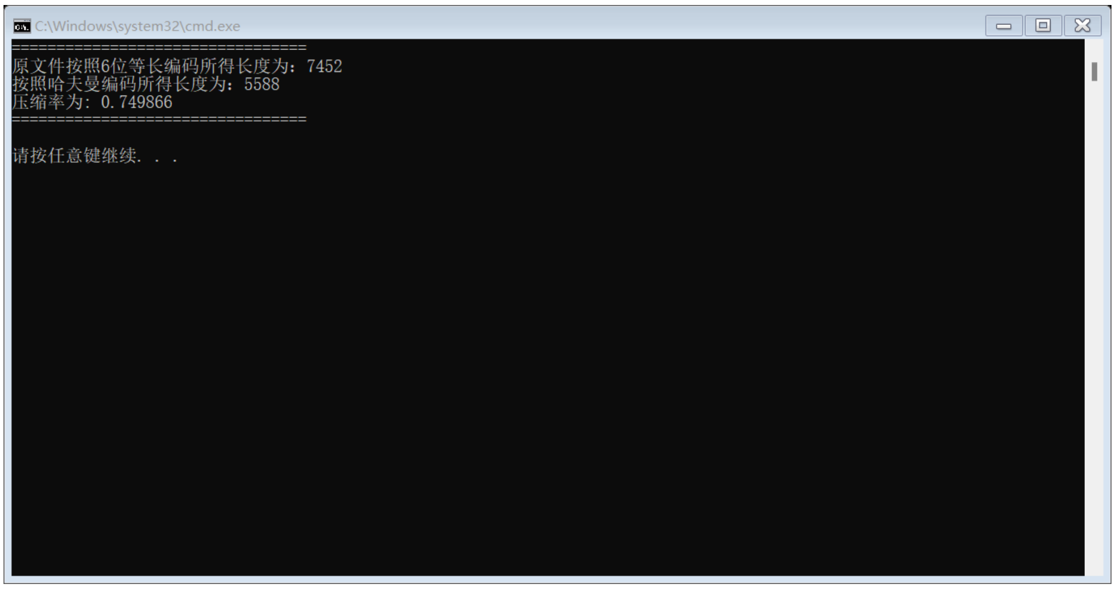
1010111110101110100111010010011110111111001101001001110000010001111110110110110010001000101001011111111000
1011001011011010011110011100111101111110000010011111011000011110010001000101001101101111011010101110101101
1010011110011100111101111110011010010011100000100110110000011011110011100100010001010011100010111111011001
0011011101010011001100101101111111100110100111001100101110100101111111101010101111111000001101010111010011 1
1111101010101010101010111111001001011011111010110011001111110110110100111100111001111011111100000100110110
0000110111100111001000100010100011101001100100110111111111010111110111100110100000100111011011111111010000 10
1111111001110110010001000101000111010011001001111011011000000111111101101000001111110010110100011010000 01
0010110101111000111010111001000100010100011101001100100110011011001001111110010110101111110110110100111100
1110011110111111000111010011001000010111001100111111010011010110011111111111000100010100011101001100100101
1101100110010101010011111101010111001111011010001111111110001000101001011111111000101100111011101110110101
0101111101011101001110100100110110110000011010111111100000100011110011011010101011001110000011011011001101
0011000101111100101010111001100101110010110110111101101000011010110111011001000100010100110111001011101 01

4. 译码结果：

Youth is not a time of life; it is a state of mind; it is not a matter of rosy cheeks, red lips and supple knees; it is a matter of the will, a quality of the imagination, a vigor of the emotions; it is the freshness of the deep springs of life. Youth means a temperamental predominance of courage over timidity, of the appetite for adventure over the love of ease. This often exists in a man of 60 more than a boy of 20. Nobody grows old merely by a number of years. We grow old by deserting our ideals.

Years may wrinkle the skin, but to give up enthusiasm wrinkles the soul. Worry, fear, self-distrust bows the heart and turns the spirit back to dust.
Whether 60 or 16, there is in every human being's heart the lure of wonders, the unfailing appetite for what's next and the joy of the game of living. In the center of your heart and my heart, there is a wireless station; so long as it receives messages of beauty, hope, courage and power from man and from the infinite, so long as you are young.
When your aerials are down, and your spirit is covered with snows of cynicism and the ice of pessimism, then you've grown old, even at 20; but as long as your aerials are up, to catch waves of optimism, there's hope you may die young at 80.

5. 比对结果：正确率 100%，压缩率 74.9866%



```
================================
原文件按照6位等长编码所得长度为：7452
按照哈夫曼编码所得长度为：5588
压缩率为：0.749866
================================

请按任意键继续. . .
```

五、经验体会与不足

经验体会：

通过本实验，加深了对哈夫曼树以及堆结构的理解，在实践中体会了哈夫曼编码的前缀性，以及编码的冗余较小的特点，对其中蕴含的贪心思想有了一定了解；而对于堆，使用堆排序可以大大减小时间复杂度，由原来的 O（n）减小到 O（log n），对于本问题的选择优化效果较为明显，但在建立堆以及维护堆的过程中也具有较大的时间复杂度，而且结构本身也增加了空间复杂度。在元素比较多的情况下，还是不错的一个选择。尤其是在解决诸如“前 n 大的数”一类问题时，几乎是首选算法。

存在不足：

1. 可以考虑基于单词的压缩，提高压缩效果
2. 可以采用 K 叉的哈夫曼树完成上述工作，实现“K 进制”的编码和译码
3. 在文件读入，信息统计过程中程序还较为冗杂，值得优化

六、附录：源代码（带注释）

```cpp
#include <iostream>
#include <fstream>
#include <cstring>
#include <algorithm>
#define SIZE 200
#define MAX 42 //文本所用字符个数

using namespace std;
struct data
{
    char alphabet;
    double frequency = 0;
};//读取数据临时储存

typedef struct node
{
    int self;
    int lchild = -1;
    int rchild = -1;
    int parent = -1;
    char alphabet;
    double weight = 0;
}HTNODE;//哈夫曼树节点

struct code
{
    char ch;
    char bits[MAX+1];
};//编码结构

typedef struct heap
{
    HTNODE heap[2*MAX]; //顺序储存堆结构
    int n; //堆内数据个数
}HEAP; //堆结构

//最小堆结构初始化
void MinHeap(HEAP *minheap)
{
    minheap->n = 0;
}

```

```c
//插入新节点
void insert(HEAP *minheap, HTNODE x)
{
    int i;
    if((minheap->n < MAX) && (minheap->n >= 0))
    {
        i = minheap->n + 1;
        while(((i != 1) && (x.weight < minheap->heap[i/2].weight)) || ((i != 1) && (x.weight == minheap->heap[i/2].weight) && (x.self < minheap->heap[i/2].self)))
        {
            minheap->heap[i] = minheap->heap[i/2];
            i = i / 2;
        }
        minheap->heap[i] = x;
        minheap->n++;
    }
}

//删除最小节点
HTNODE deleteMin(HEAP *minheap)
{
    int par = 1, child = 2;
    HTNODE ele, tmp;
    if(minheap->n > 0)
    {
        ele = minheap->heap[1];
        tmp = minheap->heap[minheap->n--];
        while(child < minheap->n)
        {
            if((child < minheap->n) && (minheap->heap[child].weight > minheap->heap[child + 1].weight))
                child++;
            if(tmp.weight <= minheap->heap[child].weight) break;
            minheap->heap[par] = minheap->heap[child];
            par = child;
            child *= 2;
        }
        minheap->heap[par] = tmp;
        return ele;
    }
}

//读取文件信息，返回等长编码文件长度
```

```
int file(int x[], struct data Youth[]);
//文件信息转移到树节点，并初始化堆
void data_read(struct data Youth[], HTNODE T[], HEAP *minheap);
//寻找最小的两个节点（不使用堆）
void Select(HTNODE T[], int i, int &p1, int &p2);
//建立哈夫曼树
void createtree(HTNODE T[], HEAP *minheap);
//哈夫曼编码
void Encoding(HTNODE T[], struct code HAff[]);
//输出编码文件，返回压缩后的长度
int transfer(struct code HAff[], char path[]);
//解码文件
void check(HTNODE T[], char path[]);

int main()
{
          struct data Youth[MAX];
          struct code HAFF[MAX];
          HTNODE T[2*MAX - 1];
          HEAP min;
          MinHeap(&min);
          int x[SIZE] = {0};
          int pre = file(x,Youth);
          cout<<"=================================="<<endl;
          cout<<"文本出现字符及其使用频率：  "<<endl;
          data_read(Youth, T, &min);
          cout<<"=================================="<<endl;
          createtree(T, &min);
          cout<<'\n';
          cout<<"=================================="<<endl;
          cout<<"各字符哈夫曼编码：   "<<endl;
          Encoding(T, HAFF);
          cout<<"=================================="<<endl;
          int now = transfer(HAFF,"result.txt");
          cout<<"原文件按照 6 位等长编码所得长度为："<<pre<<endl;
          cout<<"按照哈夫曼编码所得长度为："<<now<<endl;
          cout<<"压缩率为："<<double(now)/double(pre)<<endl;
          cout<<"=================================="<<endl;
          check(T, "check.txt");
      }

      int file(int x[], struct data Youth[])
      {
          char data;
```

```cpp
        ifstream infile;
        infile.open("Youth.txt");
        int sum = 0;
        while(infile.get(data))
        {
            x[data]++;
            sum++;
        }
        int i = 0;
        for(int j = 0; j < SIZE; j++)
        {
            if(x[j] != 0)
            {
                Youth[i].alphabet = char(j);
                Youth[i].frequency = double(x[j]) / double(sum);
                i++;
            }
        }
        infile.close();
        return sum*6;
    }

    void data_read(struct data Youth[], HTNODE T[], HEAP *minheap)
    {
        //节点初始化
        for(int i = 0; i < MAX; i++)
        {
            if(Youth[i].alphabet == '\n') cout <<"换行符"<<" "<< Youth[i].frequency<<endl;
            else if(Youth[i].alphabet == ' ')  cout <<"空格符"<<" "<< Youth[i].frequency<<endl;
            else  cout<<Youth[i].alphabet <<" "<< Youth[i].frequency<<endl;
            T[i].alphabet = Youth[i].alphabet;
            T[i].weight = Youth[i].frequency;
            T[i].self = i;
            insert(minheap, T[i]);
        }
    }

    void Select(HTNODE T[], int n, int &p1, int &p2)
    {
        int i, j;
        for(i = 0; i < n; i++)
```

```
            if(T[i].parent == -1) {p1 = i; break;}
        for(j = i + 1; j < n; j++)
            if(T[j].parent == -1) {p2 = j; break;}
        for(i = 0; i < n; i++)
            if((T[p1].weight > T[i].weight) && (T[i].parent == -1) && ((i-p2) != 0))  p1 = i;
        for(j = 0; j < n; j++)
            if((T[p2].weight > T[j].weight) && (T[j].parent == -1) && (j != p1))  p2 = j;
        //cout<<p1<<p2;
    }

    void createtree(HTNODE T[], HEAP *minheap)
    {
        int i, p1, p2,j,k;
        for(i = MAX; i < (2*MAX-1); i++)
        {
            j = deleteMin(minheap).self;
            k = deleteMin(minheap).self;
            p1 = min(j,k);
            p2 = max(j,k);
            T[p1].parent = T[p2].parent = i;
            T[i].lchild = p1;
            T[i].rchild = p2;
            T[i].weight = T[p1].weight + T[p2].weight;
            T[i].self   = i;
            insert(minheap, T[i]);

        }
        cout << "树建立完毕！";
    }

    void Encoding(HTNODE T[], struct code HAff[])
    {
        for(int i = 0; i < MAX; i++)
        {
            HAff[i].ch = T[i].alphabet;
            int start = MAX;
            char x[MAX + 1];
            x[MAX] = '\0';
            int p, q = i;
            while((p = T[q].parent) != -1)
            {
                if(T[p].lchild == q) x[--start] = '0';
                else if(T[p].rchild == q) x[--start] = '1';
                q = p;
            }
            strcpy(HAff[i].bits, &x[start]);
            if(HAff[i].ch == '\n') cout <<"换行符"<<" "<<HAff[i].bits<<endl;
            else if(HAff[i].ch == ' ') cout<<"空格符"<<" "<<HAff[i].bits<<endl;
            else cout << HAff[i].ch <<" " <<HAff[i].bits<<endl;
        }
    }

    int transfer(struct code HAff[], char path[])
    {
        ofstream fout(path);
        char x;
        int number = 0;
        if(fout)
        {
            ifstream infile;
            infile.open("Youth.txt");
            while(infile.get(x))
                for(int i = 0; i < MAX; i++)
                    if(HAff[i].ch == x)
                    {
                        fout<<HAff[i].bits;
                        number += strlen(HAff[i].bits);
                    }
        }
        fout.close();
        return number;
    }

    void check(HTNODE T[], char path[])
    {
        char x;
        ifstream infile;
        ofstream fout(path);
        if(fout)
        {
            int p,q;
            infile.open("result.txt");
            for(int i = 0; i < 2*MAX-1;i++)
                if(T[i].weight == 1)  {p = i;q = p;}
            while(infile.get(x))
            {
                if(x == '0') p = T[p].lchild;
                else if(x == '1') p = T[p].rchild;

                if((T[p].lchild == -1) && (T[p].rchild == -1))
                {
                    fout<<T[p].alphabet;
                    p = q;
                }
            }
        }
        fout.close();
        infile.close();
    }
```