**实验三 哈夫曼编码和解码**

1. 实验题目：给出一篇英文文章输出: 1)每个字符的哈夫曼编码 2)压缩比 3)压缩后的文章 4)解码
2. 解题思路：

1）申请2n个空间，构造五元组（字符，权值，左树，右树，父亲）全部置零。（n为字符种数）。

2）统计字符出现的频率(考虑大小写,空格,逗号,句号,引号)

0号元素空置，遍历，找到权值最小的两个（父亲为零的，并把父亲改为新节点下标 ），权值相加得到新的节点权值，新节点儿子设置为原来的两个结点下标以此往复直至得到2n个节点。

3）得到每个字母的编码

从叶子节点遍历放入栈中，依次出栈将0/1序列读入二维数组中。

4）对文章进行编码

扫描文章字符串并将编码依次存入数组。

5）计算压缩比

编码数组长度/原文件字符串长度/8

6）解码

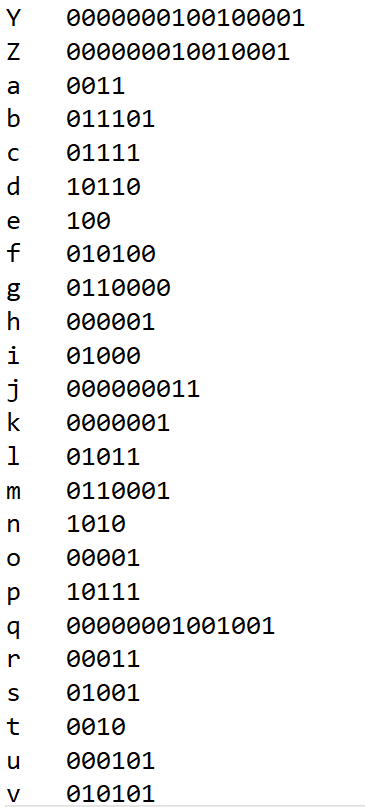
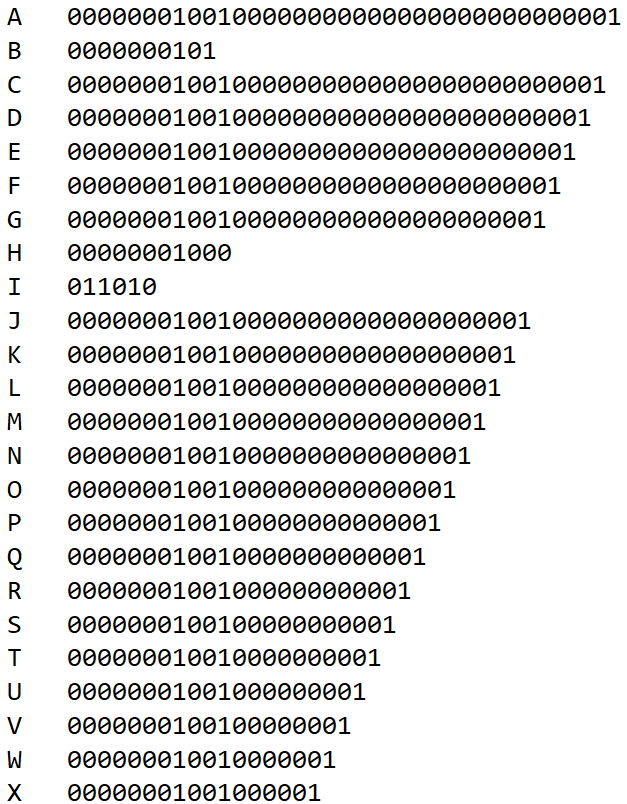
flag变量首先置0，然后读入编码，按编码从根节点遍历树，找到叶子节点时flag置1，然后寻找下一个字符实现解码。

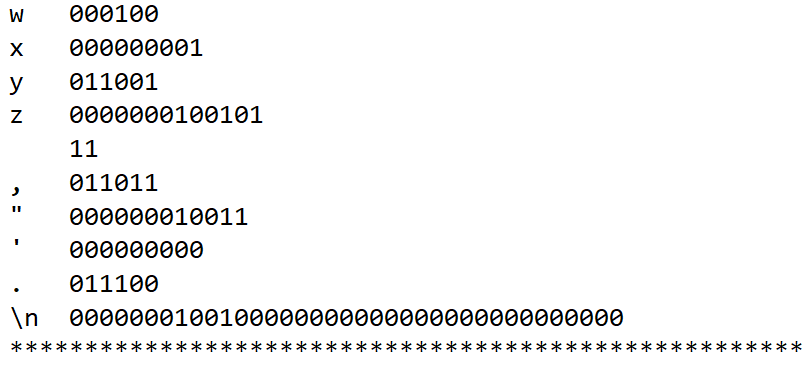
1. 源代码

#include **<iostream>**#include **<stdio.h>**#include **<string.h>**#define **n** 58  
**typedef struct**{  
 **char** C;  
 **int** parent,lchild,rchlid;  
 **int** weight;  
}HuffmanTree;  
**typedef struct**{  
 **char** s[60];  
 **int** top;  
}Stack;  
**void** CreateHT(HuffmanTree \*&HT){  
 **int** i;  
 **for**(i = 0;i < 2\***n**;i ++){  
 HT[i].C = **'0'**;  
 HT[i].parent = HT[i].lchild = HT[i].rchlid = HT[i].weight = 0;  
 }  
 **for**(i = 1;i < 27;i ++){  
 HT[i].C = 64+i;  
 }  
 **for**(i = 27;i < 53;i ++){  
 HT[i].C = 70+i;  
 }  
 HT[53].C = **' '**;  
 HT[54].C = **','**;  
 HT[55].C = **'"'**;  
 HT[56].C = 39;  
 HT[57].C = **'.'**;  
 HT[58].C = **'\n'**;  
}  
  
**void** InitHT(HuffmanTree \*&HT){  
 FILE \*fp;  
 **char** ch;  
 **if**((fp = fopen(**"test.txt"**,**"r"**))==**NULL**){  
 printf(**"cannot open the file\n"**);  
 exit(0);  
 }  
 **while**((ch = fgetc(fp))!=**EOF**){  
 **if**(ch>=65&&ch<=90)  
 HT[ch-64].weight++;  
 **else if**(ch>=97&&ch<=122)  
 HT[ch-70].weight++;  
 **else if**(ch == **' '**)  
 HT[53].weight++;  
 **else if**(ch == **','**)  
 HT[54].weight++;  
 **else if**(ch == **'"'**)  
 HT[55].weight++;  
 **else if**(ch == 39)  
 HT[56].weight++;  
 **else if**(ch == **'.'**)  
 HT[57].weight++;  
 **else if**(ch == **'\n'**)  
 HT[58].weight++;  
 **else**{  
 printf(**"The list is not enough\n"**);  
 exit(1);  
 }  
 }  
 fclose(fp);  
}  
  
**void** ChooseMinWeight(HuffmanTree \*HT, **int** i, **int** &s1, **int** &s2){  
 **int** j, temp2, temp1,temp;  
 **for**(j = 1; ;j ++){  
 **if**(HT[j].parent == 0)  
 **break**;  
 }  
 temp1 = HT[j].weight;  
 s1 = j;  
 **for**(j = s1 + 1; ;j++){  
 **if**(HT[j].parent == 0)  
 **break**;  
 }  
 temp2 = HT[j].weight;  
 s2 = j;  
 **if**(temp1 < temp2){  
 temp = temp1;  
 temp1 = temp2;  
 temp2 = temp;  
 temp = s1;  
 s1 = s2;  
 s2 = temp;  
 }  
 **for**(j = s2 + 1;j < i;j ++){  
 **if**(HT[j].parent == 0){  
 **if**(HT[j].weight <= temp1 && HT[j].weight >= temp2){  
 temp1 = HT[j].weight;  
 s1 = j;  
 }  
 **if**(HT[j].weight < temp2){  
 temp1 = temp2;  
 s1 = s2;  
 temp2 = HT[j].weight;  
 s2 = j;  
 }  
 }  
 }  
}  
  
**void** GetHT(HuffmanTree \*&HT){  
 **int** i,j;  
 **int** s1 = 0,s2 = 0;  
 **for**(i = **n**+1; i<2\***n** ;i ++){  
 ChooseMinWeight(HT,i,s1,s2);  
 HT[i].weight = HT[s1].weight + HT[s2].weight;  
 HT[i].lchild = s1;  
 HT[i].rchlid = s2;  
 HT[s1].parent = HT[s2].parent = i;  
 }  
}  
  
**void** GetHCode(HuffmanTree \*HT,**char** Code[**n**+1][60]){  
 Stack S;  
 **int** i,j,k;  
 **for**(i = 1;i <= **n**;i++){  
 S.top = 0;  
 j = i;  
 **while**(HT[j].parent != 0){  
 **if**(HT[HT[j].parent].lchild == j){  
 S.s[S.top] = **'0'**;  
 S.top++;  
 }  
 **if**(HT[HT[j].parent].rchlid == j){  
 S.s[S.top] = **'1'**;  
 S.top++;  
 }  
 j = HT[j].parent;  
 }  
 **for**(k = S.top;k > 0;k--)  
 Code[i][S.top-k] = S.s[k-1];  
 Code[i][S.top] = **'\0'**;  
 }  
}  
  
**float** Value(**char** Code[**n**+1][60],**int** &count1,**int** &count2){  
 FILE \*fp;  
 **char** ch;  
 count1=0,count2=0;  
 **if**((fp = fopen(**"test.txt"**,**"r"**))==**NULL**){  
 printf(**"cannot open the file\n"**);  
 exit(0);  
 }  
 **while**((ch = fgetc(fp))!=**EOF**){  
 **if**(ch>=65&&ch<=90)  
 count1 = count1 + strlen(Code[ch-64]);  
 **else if**(ch>=97&&ch<=122)  
 count1 = count1 + strlen(Code[ch-70]);  
 **else if**(ch == **' '**)  
 count1 = count1 + strlen(Code[53]);  
 **else if**(ch == **','**)  
 count1 = count1 + strlen(Code[54]);  
 **else if**(ch == **'"'**)  
 count1 = count1 + strlen(Code[55]);  
 **else if**(ch == 39)  
 count1 = count1 + strlen(Code[56]);  
 **else if**(ch == **'.'**)  
 count1 = count1 + strlen(Code[57]);  
 **else if**(ch == **'\n'**)  
 count1 = count1 + strlen(Code[58]);  
 **else**{  
 printf(**"The list is not enough\n"**);  
 exit(1);  
 }  
 count2++;  
 }  
 fclose(fp);  
}  
  
**void** ArticleAfter(**char** Code[**n**+1][60], **char** \*&s){  
 s[0]=**'\0'**;  
 FILE \*fp;  
 **char** ch;  
 **if**((fp = fopen(**"test.txt"**,**"r"**))==**NULL**){  
 printf(**"cannot open the file\n"**);  
 exit(0);  
 }  
 **while**((ch = fgetc(fp))!=**EOF**){  
 **if**(ch>=65&&ch<=90)  
 strcat(s,Code[ch-64]);  
 **else if**(ch>=97&&ch<=122)  
 strcat(s,Code[ch-70]);  
 **else if**(ch == **' '**)  
 strcat(s,Code[53]);  
 **else if**(ch == **','**)  
 strcat(s,Code[54]);  
 **else if**(ch == **'"'**)  
 strcat(s,Code[55]);  
 **else if**(ch == 39)  
 strcat(s,Code[56]);  
 **else if**(ch == **'.'**)  
 strcat(s,Code[57]);  
 **else if**(ch == **'\n'**)  
 strcat(s,Code[58]);  
 **else**{  
 printf(**"The list is not enough\n"**);  
 exit(1);  
 }  
 }  
 fclose(fp);  
}  
  
**void** Translate(HuffmanTree \*HT, **char** \*s, **int** count1){  
 **int** i,flag = 0,next;  
 **for**(i = 0; i < count1; i ++){  
 **if**(flag == 0){  
 **if**(s[i] == **'0'**){  
 next = HT[2\***n**-1].lchild;  
 }  
 **if**(s[i] == **'1'**)  
 next = HT[2\***n**-1].rchlid;  
 flag = 1;  
 **continue**;  
 }  
 **if**(flag == 1){  
 **if**(s[i] == **'0'**){  
 **if**(i == count1-1)  
 printf(**"%c"**,HT[HT[next].lchild].C);  
 **if**(HT[next].lchild==0){  
 printf(**"%c"**,HT[next].C);  
 i--;  
 flag = 0;  
 **continue**;  
 }  
 next = HT[next].lchild;  
 }  
 **if**(s[i] == **'1'**){  
 **if**(i == count1-1)  
 printf(**"%c"**,HT[HT[next].rchlid].C);  
 **if**(HT[next].rchlid==0){  
 printf(**"%c"**,HT[next].C);  
 i--;  
 flag = 0;  
 **continue**;  
 }  
 next = HT[next].rchlid;  
 }  
 }  
 }  
}  
**int** main() {  
 **int** j,count1,count2;  
 HuffmanTree \*HT; *//四元组表示Huffman树* **char** Code[**n**+1][60]; *//存放编码* HT = (HuffmanTree \*)malloc(2\***n**\***sizeof**(HuffmanTree));  
 CreateHT(HT); *//初始化树* InitHT(HT); *//存入频度信息* GetHT(HT); *//将Huffman树完善化* GetHCode(HT,Code); *//生成编码* **for**(j = 1;j <= **n**-1;j ++){  
 printf(**"%c %s\n"**,HT[j].C,Code[j]);  
 }  
 printf(**"\\n %s\n"**,Code[j]);  
 Value(Code,count1,count2); *//计算压缩比* printf(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);  
 printf(**"%lf\n"**,**float**(count1)/**float**(count2\*8));  
 **char** \*s = (**char** \*)malloc(**sizeof**(**char**)\*count1); *//存放压缩后的文件* ArticleAfter(Code, s); *//生成压缩文件* printf(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);  
 printf(**"%s\n"**,s);  
 printf(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);  
 Translate(HT,s,count1); *//解码* printf(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);  
 **for**(j = 1;j < 116;j ++){  
 printf(**"%d %c %d %d %d\n"**,j,HT[j].C,HT[j].lchild,HT[j].rchlid,HT[j].parent);  
 }  
 **return** 0;  
}

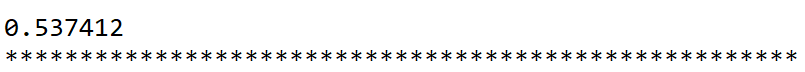
1. 运行结果

1）字符编码

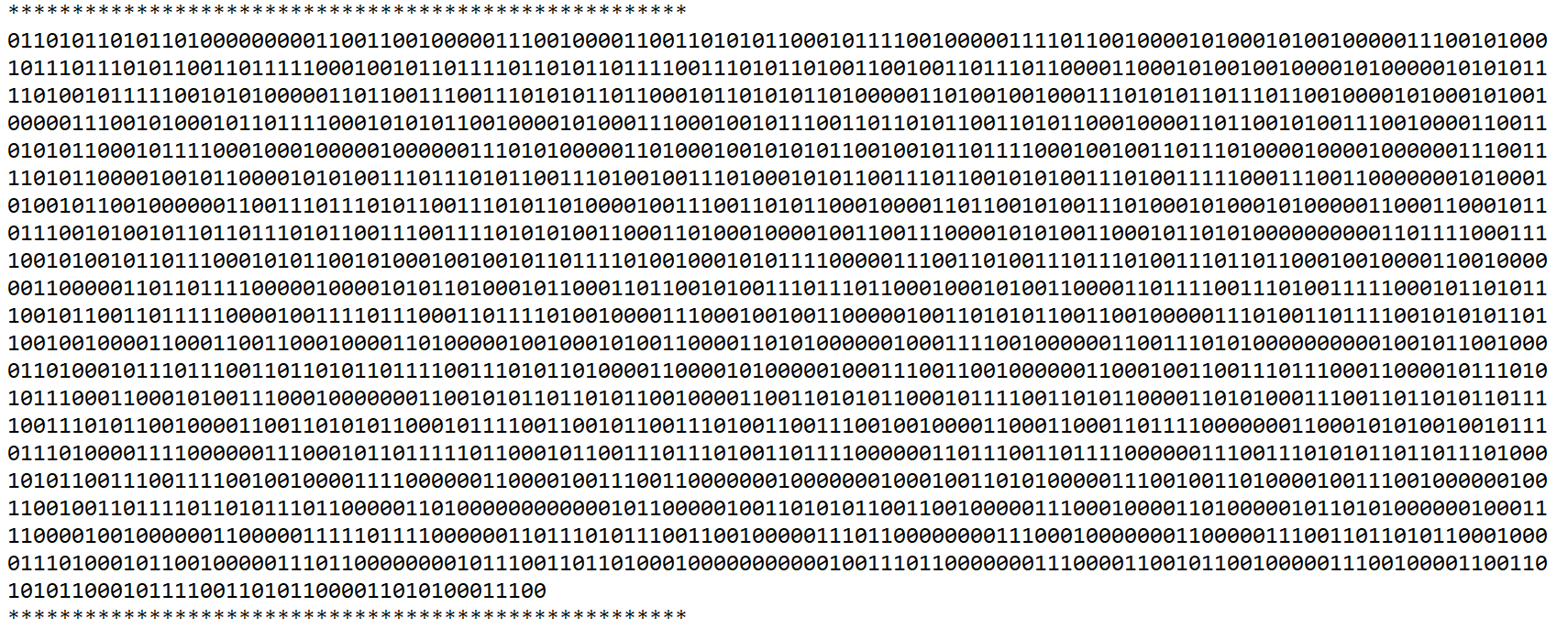




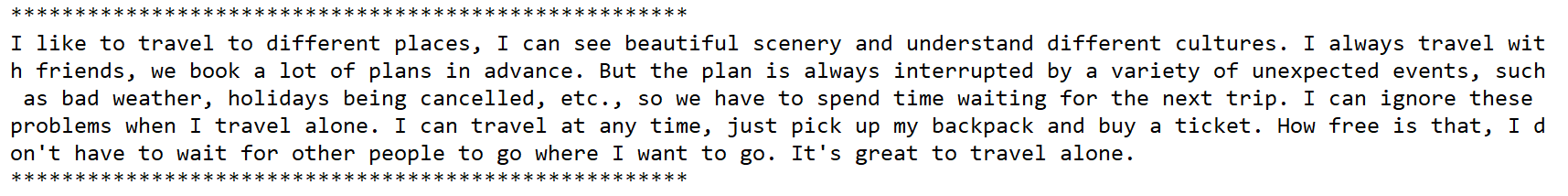
2）压缩比



3）压缩后的文件



4）解码效果



1. 心得体会
2. 将字符存入四元组更方便编码 解码，虽然牺牲了空间但是时间复杂度降低。
3. 用五元组来代替树结构反而更加方便快捷，所以要根据实际需要转换数据结构。