实验四: huffman编码

实验时间

实验	实验时间	补交时间(满分80%)
实验一:约瑟夫环	4.3晚——4.10晚	4.17晚
实验二: 停车场问题	4.17晚——4.24晚	4.26下午
实验三: 关键词检索	4.24晚——5.8晚	5.10下午
实验四:huffman编码	5.8晚——5.15晚	5.17下午
实验五: 管道铺设	5.15晚——5.22晚	5.24下午
实验六:统计程序相似性	5.22晚——5.29晚	5.31下午

实验内容

利用 Huffman 编码进行通信可以大大提高信道的利用率,缩短信息传输时间,降低传输成本。但是,这要求在发送端通过一个编码系统对待传数据进行预先编码,接收端进行解码。对于双工信道(即可以双向传输信息的信道),每端都需要一个完整的编/解码系统。

实验要求

使用 C 或 C++ 实现一个完整的huffman编解码系统,其应具有以下功能:

- 1. 初始化 [I]:Initialization:
 - 统计文件 ToBeTran.txt 中所有字符(包括空格、换行等其他符号)出现的次数,并将其按 ASCII 码顺序依次输出到控制台。随后根据每个字符的频度建立 Huffman 树,并存入 hfmTree.txt 中。(下图仅供参考,并不指定格式)

```
I:Initialization
I:Encoding
I:Print
I:Tree Printing
Quit
                                   id=134 weight=
                                                     4 ls=131 rs=132
                                   id=131 weight=
                                                     2 ls= 77 rs= 58
                                                     2 ls= 78 rs= 80
                                39 id=132 weight=
40 id=138 weight=
                                                     8 ls=135 rs=122
  id=135 weight=
                                                     4 ls= 33 rs= 67
                                42 id=144 weight=
                                                    18 ls=139 rs= 84
                                43 id=139 weight=
                                                    9 ls= 48 rs=106
                                   id=173 weight=1058 ls= 32 rs=169
                                45 id=169 weight= 551 ls=116 rs=163
                                46 id=163 weight= 300 ls=157 rs=158
                                   id=157 weight= 146 ls=102 rs=103
                                48 id=158 weight= 154 ls=153 rs=117
                                49 id=153 weight= 77 ls= 44 rs= 98
                                50 id=10 weight= 23 :\n:01010110
                                   id=32 weight= 507 : :110
                                52 id=33 weight=
                                                  2:!:10111110100
                                53 id=39 weight=
                                                    1 :':01010111
                                   id=44 weight= 38 :,:1111100
                                   id=45 weight=
                                                   6 :-:010101001
                                   id=46 weight= 29 :::0111101
                                   id=48 weight=
                                                   4:0:1011111100
                                   id=49 weight=
                                                    1 :1:101111100000
                                   id=53 weight=
                                                    1 :5:1011111100001
                                   id=58 weight=
                                                    1 :::1011111100101
                                   id=65 weight=
                                                    6 :A:011110010
                                   id=67 weight=
                                                    2 :C:10111110101
                                   id=68 weight=
                                                    2 :D:0101010000
                                   id=69 weight=
                                                   6 :E:011110011
                                   id=70 weight=
                                                   11 :F:01010101
```

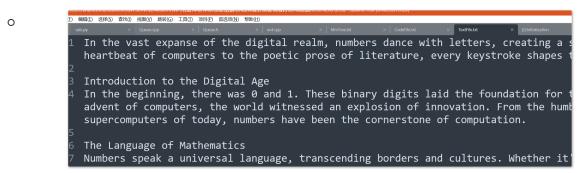
2. 编码 [E]:Encoding:

○ 利用已经建立好的 Huffman 树,对文件 ToBeTran.txt 中的正文进行编码,然后将结果存入文件 CodeFile.txt 中。所有字符都需要编码



3.解码 [D]:Decoding:

利用已经建立好的 Huffman 树,将文件 CodeFile.txt 中的代码进行解码,
 然后将结果存入文件 TextFile.txt 中。应当与 ToBeTran.txt 的内容和格式完全一致



4. 打印代码文件 [P]: Print :

○ 将文件 CodeFile.txt 以紧凑的格式显示在终端上,每行 50 个代码(不再有空格)。同时将此字符形式的编码文件写入文件 CodeFile50.txt 中。

5. 打印huffman树 [T]:Tree Printing:

○ 将建立好的 Huffman 树以直观的形式(树或者凹入的形式)显示在终端上,同时 将此字符形式的 Huffman 树写入文件 TreePrint.txt 中。

0

```
[I]:Initialization
[E]:Encoding
[D]:Decoding
[P]:Print
[T]:Tree Printing
[Q]:Quit
 -176
           -170
                    |-159
                 | -d:weight=79 00000
| |-m:weight=84 00001
|-s:weight=164 0001
-e:weight=334 001
                   |-r:weight=170 0100
|-160
                           -
-154
                              |-y:weight=46 010100
|-149
                                          -140
                                                -136
                                           | |-D:weight= 2 0101010000
| |-q:weight= 3 0101010001
|--:weight= 6 010101001
|F:weight=11 01010101
                         | |-\n:weight=23 01010110
|-\:weight= 1 01010111
|-\:weight=95 01011
                   |-a:weight=199 0110
|-161
                        |-c:weight=100 01110
|-155
                                -150
                                        |-I:weight=12 01111000
|-141
```

6. 退出 [Q]:Quit:

- 结束程序。
- 7. 对于上述的操作 2,3,5 ,若内存中没有 Huffman 树,则应从文件 hfmTree.txt 将树读取到内存中。

要求有交互,包含选择界面以及必要的提示语句

实验步骤

- 1. 下载群文件中的 lab4.zip 并解压到本地,仔细阅读本文档 实验四: huffman编码.pdf。
- 2. lab4.zip 中包含 ToBeTran.txt ,请自主编写符合要求的代码。

Bonus

- 将huffman树以其他更直观的方式画出来,并体现出其层级关系。*手工连线不算数*。
 - o hint: mermaid
- 手写堆加速huffman建树。直接用 priority_queue 或其他 STL 不算数。
- 二选一即可。