实验报告

功能实现

模式选择与读入参数功能

由于使用命令行读入参数较为麻烦，选择改为运行exe，根据提示来进行输入，同时设置了默认参数。

主要函数

```
//建树
node *buildTree();
//阅读权重文件，每一行格式为：字母:权重。唯一例外是空格用<space>标识。
void read_weight(string path);
//保存模型
void save_model(string path);
//加载模型
void load_model(string path);
//解码
void decode(string path,string outpath);
//编码
void encode(string path,string outpath);
//根据权重得到编码
void getcode(node *Node, string pre_code);
```

数据结构

```
//优先队列，用于生成树，这里重写了比较方法来实现最小堆。
priority_queue<struct node, vector<struct node>, CompareNode> q;
//用来保存得到的码表。
map<string, string> code;
//用于生成树所用的数据结构。
struct node
{
    int weight;
    string name;
    struct node *left = nullptr, *right = nullptr;
};
```

使用方法以及运行结果

首先根据用户输入参数进行模式选择，然后根据输入文件参数运行文件。

c模式

输入模型文件，默认选择`model.txt`。

运行`read_weight()`函数，读取输入，生成哈夫曼树，再得到编码表，输出编码表。

输出结果如下：

```
please input mode(c/e/d):
c
please input the weight file(default: weight.txt)
weight.txt
PWL:115
<space>:100
A:00100
B:01
C:0011
D:0001
E:101
F:11
G:00101
H:0000
```

e模式

同样是先读入参数，然后根据不同模式选择不同参数。

编码模式下，首先会加载模型文件，得到编码表。

这个模式比较简单，有了编码表之后，依次读入字符串文件的每个字符，转为编码，输出到屏幕并写入文件即可。

运行截图：

```
please input mode(c/e/d):
e
please input the model file(defualt:model.txt)
model.txt
please input the string file(defualt:string.txt)
string.txt
please input the output file path(defualt:01.txt)
01.txt
the result of encoding:
00100010011000100010001100101100000010010010010010010010010010
01001001001001001001001001000010000100001000010000100001000010
00101010101
```

d模式

同样是先读入参数，然后根据不同模式选择不同参数。

编码模式下，首先会加载模型文件，得到编码表。

由于输入文件是一个巨大的01串，解码要稍微复杂一点。

同时由于map是一个单射，每次判断值是否有对应的键进行判断时都要遍历整个map。

解决方案是先从1个01串开始读，每次匹配不成功就增加1个01字符，成功了就找到对应的键，那就是对应的解码结果，同时注意要对已经匹配的01串进行删除操作。

基于此，在权重表以及字符串均在表中出现时，可以正确运行解码，并输出到屏幕，结果如下：

```
please input mode(c/e/d):
d
please input the model file(defualt:model.txt)
model.txt
please input the string file(defualt:01.txt)
01.txt
please input the output file path(defualt:string.txt)
string.txt
the result of encoding:
ABCDDD E H                    AAAAAAABBBBB
```