# 实验三：哈夫曼压缩/解压器

## 问题描述

你需要利用 Huffman 编码原理实现一个支持压缩和解压的程序。针对一个输入的文件（例如 `image.png`），它能将其压缩为 `image.png.huff`；将 `image.png.huff` 解压后，能获得和原始文件一样的文件。

## 实现要求

1.统计元素频次（以 8 bit 为基本单位），构建哈夫曼树。

2.使用三叉链表("三叉"指*parent, *lchild 和 *rchild) 实现哈夫曼树。

3.利用 Huffman 编码表"翻译"原文件，写入到目标文件中。注意，你必须同时将 Huffman 编码表也写入到目标文件中，否则将无法解压。为此，你需要自己设计合理的方式来存储这两项内容到一个文件中。

4.压缩和解压要分成两步来实现，必须通过已压缩的文件来实现解压。具体实现是，输入原始文件，得到压缩文件（压缩文件应包含huffman编码表），结束程序。输入压缩文件，得到解压文件，结束程序。不可以在一个进程中同时实现压缩和解压。

## 测试数据

```
testcase
├── 1.MIT-license.txt
├── 2.hamlet.txt
├── 3.USTC-logo.bmp
├── 4.USTC-logo.png
├── 5.data-structure.pdf
└── 6.MV.mp4
```

针对 `testcase` 目录中的每一个测试样例，将其压缩存至 `compressed` 目录，再解压至 `decompressed` 目录。`decompressed` 目录中的每个文件应能正常打开，且和 `testcase` 中对应文件的内容一致。

## 选做内容

### 针对英文自然语言纯文本文件的压缩率优化

在基本要求中，我们以 1 byte / 8 bit 为基本单位，对于英文纯文本文件，"以 1 byte / 8 bit 为基本单位"等价于"以单个字符作为基本单位"。考虑到在英文自然语言中，同一个单词可出现多次，且不同单词的频次差异很大，当我们将单词作为基本单位时，有可能可以实现更高的压缩率。

但另一方面，以单词作为基本单位可能导致元素种类过多，这一方面会让存储的 Huffman 编码表过大，另一方面会导致很低频的单词的编码过长，对提高压缩率无益。

为此，我们采用这样的优化策略：当某单词出现频次超过某一个阈值时，我们取单词作为基本单位；否则，取组成它的各个字符作为基本单位。其中，阈值的取值由你自行设定（实际上，当这个阈值无限大时，这种策略实际上退化成了原始的以单个字符作为基本单位的情况）。

请实现此优化策略，针对 `2.hamlet.txt` （《哈姆雷特》全书）文件，计算采用这种优化策略的压缩率，对比没有采用优化策略时的压缩率。

如果你的实现中，该优化策略没能起效，可能是因为存储编码表的方式不够高效。你可以自行找一个更大的英文自然语言纯文本文件（当然，直接把 `2.hamlet.txt` 自身复制很多遍也是可以的），只要你能找到一个 testcase 让你的优化方案实际起效，该选做就视为通过）

### 构造哈夫曼树过程中寻找子节点比较次数优化

哈夫曼树在构建过程中，需先进行两轮比较，选取最小的两棵树作为左、右子树。直接进行两轮比较，比较次数为$2n-3$次（选出最小的节点，需要n-1次比较，选出次小的节点需要n-2次比较），我们可以对这个比较次数进行优化，使得选出最小的两个子树只需要$n+\lceil \lg n \rceil -2$次比较，请使用这种方法寻找最小的两个子树。

## 得分统计

6个测试样例，前三个测试样例每个一分，后三个测试样例，每个两分。

助教对代码实现提问一分。

附加分两分。每个附加选项一分。

## 检查时间

11月3号，11月10号，一共两周的时间。