Python 大作业：Text Compressor

施晓钰, 516030910567

2017 年 12 月 25 日

1 Text Compressor 简介

此压缩器可通过重新编码对英文文本文档进行压缩和解压操作。

压缩操作：点击“选择文件”可在本地选择一文本，压缩成一个二进制文件和一个对应的字典存入通过“选择输出路径”指定的文件夹，并输出压缩率。其中二进制文件命名为 compressed.out，字典命名为 dict.txt。

解压操作：点击“选择文件”选择本地的二进制文件，点击“选择字典”选择与该二进制文件对应的字典，将此二进制文件解压后存入通过“选择输出路径”指定的文件夹。此过程可能较慢。

若使用 command line 版本则直接运行程序，按提示进行输入即可。

注意：此文本压缩器只适用于英文文档。

2 Text Compressor 实现

压缩部分：

- 生成字典：首先按字符遍历该文档，对每个字符在文档中的出现次数进行记数，并以此构建哈夫曼树。这里采用了 sorted 函数，将字符由小到大排成一个队列，每次取出最小的两个字符作为新节点的儿子节点，并将它们的值之和作为新节点的值，在队列中删除这两个字符并加入新的节点，重新排序，重复以上过程直到队列中只剩下一个节点。再由根对哈夫曼树进行遍历，并对每个节点进行赋值作为它的新编码。对于每一个节点而言，它的左儿子的编码是他的编码末端加上‘0’，右儿子则加上‘1’。
- 生成二进制文件：遍历整个文档，将每一个字符换成对应的哈夫曼编码，再将这个长长的 01 串进行以八位为一单位的切割并存成一个 char，将这个转化后的文件存为二进制文件。

解压部分：

- 字典导入：通过上传的字典文件生成相对应的哈夫曼树。对于每一个字符，从根开始建立新的节点，最终将每一个字符都加到哈夫曼树的叶子节点上。

- 文件解码：将每个字符再转化为八位二进制编码，利用这个 01 串对建立好的字典树进行查询。由于构建了哈夫曼树，所有字符对应的节点都是叶子节点，没有字符的编码是其他字符编码的前几位，因此只需直接依据该 01 串进行查询直到找到一个字符，再返回树的根节点进行下一轮查询即可。翻译出的文件即为原来未压缩的文件。

压缩原理：利用了哈夫曼树，可以将使用频率较大的字符存成最低至三位的编码，使用频率小的字符可能会对应相对较长的编码，但总的来说相比于原来的八位占用了更小的空间。

3 一些经历

作者是计算机系的学生，但之前没有接触过 python，所以对 python 的一些函数不是很了解，所以一开始想利用 heapq 的堆进行字符计数后的排序，但最后发现它只是一个堆没有进行排序。于是就改用了 sorted 函数。

对于存成二进制文件这件事之前一直不是很理解，把文本变成了 01 串之后就进行了储存，结果发现比原来的文件还大，花了一段时间弄明白切割和转码的操作。

因为更倾向于用 commandline 操作，所以 ui 画面没有进行美化。（但 commandline 需要手动输入文件路径也挺麻烦的。）