数据结构 2022秋 Project: 基于哈夫曼编码的压缩工具

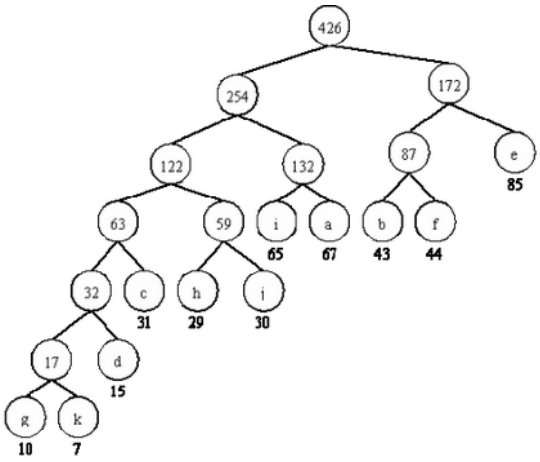
TA：张皓捷 19302010021　吴逸昕 19302010013

写 PJ 的过程中难免会遇到许多代码本身的逻辑问题，TA 可能难以在复杂的代码结构中直接找到存在的问题，此时你可以先使用 Debugger ~~或是无限打印~~ 进行本地调试，锁定问题的具体位置，善用谷歌、Stack Overflow、博客等搜索相关解决方案，并注意是否存在空引用、内存泄漏、类型错误、对象内容被不恰当地修改且未改回等问题。

背景介绍

文件压缩 在节省磁盘存储空间与减少传输时间上起着重要作用。通常而言，压缩机制可被分为有损压缩和无损压缩两类。顾名思义，有损压缩会在恢复压缩文件后导致内容的丢失，常用于媒体数据；而无损压缩可以完全恢复原始文件。我们常见的 `.zip` `.rar` 格式便属于无损压缩。

哈夫曼编码 (Huffman Coding)，又称霍夫曼编码，是一种编码方式，哈夫曼编码是可变字长编码 (VLC) 的一种。Huffman 于 1952 年提出一种编码方法，该方法完全依据字符出现概率来构造异字头的平均长度最短的码字，有时也称之为最佳编码。



在本项目中，你需要使用 **哈夫曼编码** 实现一个能够对文件与文件夹进行无损压缩的 **压缩/解压** 工具。共需提交 2 次（中期检查、最终提交）文档或代码。项目需使用 **C++** 语言完成，无初始代码。建议在开始前先思考“中期检查”中的问题，并做项目整体的构思与设计。

请注意：切勿抄袭！！ 代码提交后将做查重， **一经发现抄袭或雷同双方均作 0 分处理** 。简单改个变量名也是不行的！若先前已存在抄袭行为，第二次发现将按规定挂科处理！

具体要求

1. 中期检查 (10%)

所有同学需于 **2022年11月20日 23:59前** 提交 pdf 格式的 **中期文档** 到 elearning。

文件名格式：**学号-姓名-中期文档.pdf**。

中期文档中，需要思考并回答以下 6 个问题：

(1) 已知字母 a, b, c, d, e, f, g, h 的出现次数分别为 11, 6, 8, 6, 15, 2, 4, 3，且文件中仅存在这些字母。请据此绘制相应的哈夫曼树（电子版或手绘版并拍摄皆可），并给出每个字母对应的哈夫曼编码。

(2) 如何根据文件的字节流，构建哈夫曼树?

(3) 构建哈夫曼树的过程中，如何每次高效、便捷地选出出现频率最低的两个节点?

(4) 如何将哈夫曼树存储到文件?

(5) 如何完成文件夹的压缩并保留内部文件名等信息的一致性?

(6) 于文档中附上目前代码完成情况的主体部分，并做 **简略的** 说明。

此外，助教还会随机抽取大约 20% 的幸运观众，于课间或课后检查 PJ 代码与思路。届时请勿缺席。

若未提交中期文档，或是中期文档完成度较差，或是抽查到代码进度过慢，**将根据比例酌情扣分** 。

2. 核心需求 (60%)

(1) 文件的压缩与解压 (30%)

- 需要能够正常压缩/解压给定的 **一个** 非空文件。你需要确保文件在压缩/解压操作后的内容和原始文件是完全一致的，并确保在大多数情况下压缩后的文件大小应小于压缩前的文件大小。此外，文件可能会比较大 (size > 4GB) ，你需要小心 `int` 溢出。(20%)
- 需要能够正常压缩/解压 **一个** 空文件 (size = 0B) 。(5%)
- 压缩时，应当能够 **指定压缩包的名称** ；解压时，需 **还原出原本的文件名** ，即便压缩包名称与文件名不同，甚至之后又对压缩包进行了重命名。(5%)

(2) 文件夹的压缩与解压 (20%)

- 需要能够正常压缩/解压给定的 **一个** 非空文件夹。注意文件夹的 **深度** 是不确定的，即给定的文件夹中可能还有子文件夹。例如下面的 `Folder` 文件夹中还有 `SubFolder1` 和 `SubFolder2` 两个子文件夹， `SubFolder1` 下还有一个 `SubFolder3` 子文件夹。(10%)
- 需要能够正常压缩/解压 **一个** 空文件夹。(5%)
- 解压时，同样也应还原出原本的文件名、文件夹名。(5%)

```
Folder
├── SubFolder1
│   ├── SubFolder3
│   │   └── file1.txt
│   ├── file2.txt
│   └── file3.txt
├── SubFolder2
│   └── file4.txt
└── file5.txt
```

(3) 性能 (5%)

你的程序应该尽可能高效，包括时间上和空间上。

时间上：将根据运行时间排名给分。若你的电脑配置较低，可在演示时借用其他同学的电脑运行。

排名百分比	得分
1% ~ 20%	5
21% ~ 50%	4
51% ~ 80%	3
81% ~ 100%	2

空间上：如果相比其他同学，你的程序明显消耗过多内存，将会被扣去 2 分。

(4) 代码风格 (5%)

你的程序应保持良好的面向对象风格，良好的代码风格、注释习惯，具备较强的可读性，并符合标准命名规范。不宜出现过长的类或方法，过量的耦合，或是大篇幅的重复代码。最初写出的代码很有可能需要经过大规模耐心细致的重构。此部分将酌情给分。

3. 其他需求 (30%)

(1) 使用CLI与用户交互 (4%)

我们希望你的 PJ 可以编译成可执行程序，在命令行界面 (Command Line Interface) 以参数的形式指定输入输出。

可以参考 Linux 下 tar, zip 等工具的输入输出方式：

- 例如 `zip png.zip 1.png` 表示将当前目录下的 `1.png` 压缩成 `png.zip` 。
- 又例如 `unzip png.zip` 表示解压当前目录下的 `png.zip` 。

关于 tar 和 zip 等工具的具体用法，可参考 https://www.runoob.com/w3cnote/linux-tar-gz.html

请注意：以上只是提供一种思路，不一定要做得和 tar 或 zip 一样。合理即可。

(2) 检验压缩包来源是否是自己的压缩工具 (4%)

用户在使用我们的工具解压的时候可能会不小心输错参数，即尝试解压一个奇怪的，不是由我们的压缩工具创建的文件（例如尝试解压一个 `.mp4` 文件）。

对于这种情况，我们希望你的 PJ 可以给出类似于 **“这不是我创建的文件，无法解压”** 的提示，而不是任其崩溃报错，或给出一些用户看不懂的信息，或是放任它错误地运行。实现方式不限。

(3) 文件覆盖问题 (4%)

在压缩/解压的时候可能会遇到文件覆盖 (overwrite) 问题。

- 例子1：用户想将当前目录下的 `data_structures.txt` 压缩成 `ds.huffman` 。但是当前目录下 `ds.huffman` 文件已经存在。这时是否要覆盖掉原来的文件（丢失原有信息）？还是停止压缩?
- 例子2：用户想要将 `ds.huffman` 解压到当前目录。 `ds.huffman` 中包含了文件 `data_structures.txt` ，但当前目录下 `data_structures.txt` 文件已存在（可能与压缩包中的不同）。这时是否要覆盖掉原来的文件（丢失原有信息）？还是停止解压?

请设计一个方案，**防止用户在不知情的情况下发现自己的文件被覆盖，并且可以自由选择覆盖或是停止** 。实现方式不限。

(4) 压缩包预览 (8%)

用户可以在不解压的情况下，通过指令，预览压缩包内的文件结构。如果压缩的是文件夹，应在控制台输出文件/文件夹名的树形结构。如果压缩的是单个文件，按相似结构输出这单个文件的名称即可。

以下格式可供参考，不一定要做得和以下格式一致，只要体现出树形结构即可：

```
scrapy_shmeea
├── scrapy.cfg
├── scrapy_shmeea
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-39.pyc
│   │   ├── pipelines.cpython-39.pyc
│   │   └── settings.cpython-39.pyc
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       ├── ShmeeaNewsSpider.py
│       ├── __init__.py
│       └── __pycache__
│           ├── ShmeeaNewsSpider.cpython-39.pyc
│           └── __init__.cpython-39.pyc
└── shmeea_news.json
```

(5) 与其他压缩工具的压缩率和压缩时间比较 (4%)

将你的压缩工具在小文件、大文件（至少 3 个测试用例）上的 **压缩率** 和 **压缩时间** 与其他压缩工具 (WinRAR, 7Z, HaoZip 等，至少 2 种) 比较，绘制一张表格，并 **简要分析一下产生这些区别的原因**（少于 500 字）。此部分将酌情给分。

(6) 开发文档 (6%)

开发文档 (PDF 格式) 应至少包含以下内容：

- 代码结构概要说明；
- 项目“核心需求”与“其他需求”中每个评分项的设计、实现思路的大致描述；
- 开发环境/工具，以及如何编译/运行项目；
- 性能测试结果（表格记录每个测试用例的初始大小、压缩后大小、压缩率、压缩时间、解压时间）；
- 与其他压缩工具的压缩率和压缩时间比较；
- 遇到的问题和解决方案；
- 其他你想说明的问题（若有）。

评分汇总

评分项	该项总分
中期检查	10
文件的压缩与解压	30
文件夹的压缩与解压	20

评分项	该项总分
性能	5
代码风格	5
使用 CLI 与用户交互	4
检查压缩包来源	4
文件覆盖提示与选项	4
压缩包预览	8
与其他工具的比较	4
开发文档	6
总分	**100**

测试用例

123 云盘链接: https://www.123pan.com/s/mBRKVv-auzN3

提取码: 9Jyu

提示与建议

1. **使用任意你喜欢的工具开发：**在使用 **C++** 语言的前提下，你可以使用任意编辑器/IDE，在任意平台（Windows/macOS/Linux）上开发 PJ 。请在说明文档中注明你使用的开发环境/工具，并说明如何编译你的 PJ 。
2. **哈夫曼树的序列化与反序列化：**在压缩前，你需要考虑如何将内存中的哈夫曼树 **存储（序列化）**到硬盘上；并在解压前将硬盘上存储的哈夫曼树 **恢复（反序列化）** 到内存中。
3. **注意空文件和空文件夹：**如果设计不当，Corner Case 可能会使你的程序崩溃。
4. **使用带缓冲的输入输出**：使用不带缓冲的 IO 方式，逐字节读写文件是非常慢的。而 std::fstream, std::ifstream, std::ofstream, fopen, fread, fwrite 等 IO 方式默认带有用户态缓冲区。使用这些 IO 方式可以减少你的程序在 IO 上的时间开销。
5. **注意内存消耗**：在压缩/解压时，你不应该一次性将整个文件读取到内存中，以免内存消耗过大。
6. **如果你的编译器支持C++17**：使用 std::filesystem 中的类可以简化文件系统的操作。
7. **十六进制编辑器**：在调试PJ时经常需要以二进制/十六进制查看输入/输出文件内容。在 Visual Studio Code 中安装 Hex Editor 扩展，就可以在 Visual Studio Code 中以二进制/十六进制查看文件。
8. **检验压缩->解压后的文件是否与原始文件一致**：你应该确保你的 PJ 具有无损压缩/解压的能力。
9. **尽早动手！！**课程 PJ 工作量较大，建议不要在 DDL 前临时赶工，否则极有可能无法完成或是存在严重漏洞。

性能测试指南

Linux 或 macOS

打开 **终端**，执行命令 `time <your command>` 来测试压缩/解压消耗的时间。

例如：`time zip png.zip 1.png`。

你会得到类似于如下的输出。其中 `total`（或 `real`）所对应的时间，也就是 0.110s，就是你压缩/解压消耗的时间。

```
# 第一个样例
0.02s user 0.01s system 29% cpu 0.110 total

# 第二个样例
real 0.16
user 0.05
sys 0.01
```

在第一个样例中，我们发现除了 `total`，还有 `user` 和 `system` 的两个时间，分别为 0.02s 和 0.01s。有兴趣的同学可以思考以下两个问题（不计分，不用作答）：

- `user` 和 `system` 的时间分别代表了什么含义？
- 为什么有时会出现 `time[user] + time[system]` 明显小于 `time[total]` 的情况？以上面的情况为例，`0.02 + 0.01 < 0.110`，中间相差的 `0.08s` 去哪了？

Windows

打开 **Windows PowerShell**，执行命令 `Measure-Command { <your command> }` 来测试压缩/解压消耗的时间。

例如：`Measure-Command { zip png.zip 1.png }`

你会得到类似于如下的输出。以下的输出代表你的程序消耗了 3017.9706ms。

```
Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 3
Milliseconds      : 17
Ticks             : 30179706
TotalDays         : 3.49302152777778E-05
TotalHours        : 0.000838325166666667
TotalMinutes      : 0.05029951
TotalSeconds      : 3.0179706
TotalMilliseconds : 3017.9706
```

未尽事宜

本文档难免会有未尽事宜。如果在理解 PJ 需求或开发 PJ 过程中遇到问题可联系助教。

如果有多名同学提出了共性的问题，助教会在下面这个飞书文档中做出解释。所以小窗助教前请先看一下下面文档中的内容是否解决了你的疑惑。

飞书 Docs Link: https://ph8fmuv5sx.feishu.cn/docx/doxcnw85R1rNb75riq609HW3LEh
Password: uw8A

截止日期

中期检查

2022年11月20日 23:59前

提交 pdf 格式的 **中期文档** 到 elearning。文件名格式：**学号-姓名-中期文档.pdf**。

最终提交

2022年12月11日 23:59

请将完整的 **源代码** 和 **开发文档** 打包后上传到 elearning 。文件名格式：**学号-姓名-PJ.zip**。每逾期 1 日总分将扣除 10 分！

此外，还会组织全体同学面试，对项目进行现场演示。面试时间将另行通知。