2023秋季学期-数据结构 Project: 基于哈夫曼编码的压缩工具

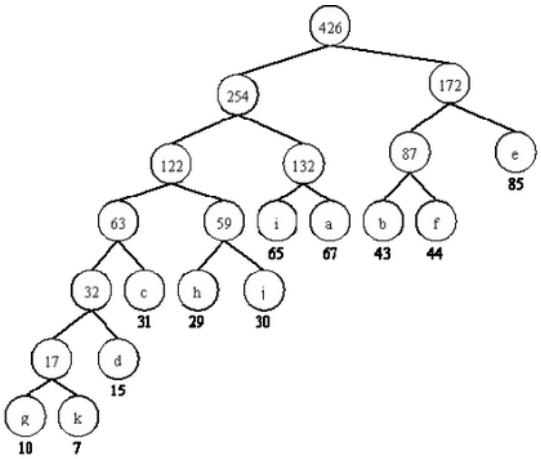
助教：吴逸昕 23210240335　　袁乐天 23210240093

> 写 PJ 的过程中若遇到许多代码本身的逻辑问题，请先使用 Debugger 进行本地调试，锁定问题的具体位置，并搜索相关解决方案，同时注意是否存在空引用、类型错误、对象内容被不恰当地修改等问题。

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 **Java/C++ 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！！ 代码提交后将进行查重， **一经发现抄袭或雷同双方均作 0 分处理** 。先前已发现个别抄袭情况，请务必引起重视，切勿抄袭。

截止日期

中期检查

2023年11月19日 23:59

提交 pdf 格式的 **中期文档** 并将 **目前已完成的源代码** 打包成 zip 格式提交到 elearning。两个提交窗口分别提交，文件名格式：文档 `学号_姓名_中期文档.pdf` ，代码 `学号_姓名.zip` 。

最终提交

2023年12月10日 23:59

提交 pdf 格式的 **开发文档** 并将 **源代码** 打包成 zip 格式提交到 elearning。两个提交窗口分别提交，文件名格式：文档 `学号_姓名_开发文档.pdf` ，代码 `学号_姓名.zip` 。每逾期 1 日将从最终得分上扣除 10 分。

此外，还会组织全体同学面试，于同学自己的电脑上对项目进行现场运行演示。面试时间另行通知。

具体要求

1. 中期检查 (10%)

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。**

若中期文档完成度较差，或是抽查到代码进度过慢，**将视情况酌情扣分** 。

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。若电脑配置较低，可在演示时借用其他同学的电脑运行。

注：鉴于 C/C++ 代码执行效率高于其他任何编程语言，我们会将 C/C++ 的代码执行时间乘以 `2` 之后再与其他编程语言的进行比较。

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以至于使得整个运行体验相当卡顿的，得 -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。最初写出的代码很可能需要经过大规模耐心细致的重构。此部分按完成情况酌情给分。

3. 其他需求 (30%)

(1) 用户交互 (4%)

你的 PJ 需要在控制台 **以参数的形式指定输入输出**，**不断等待用户的新的指令**，并显示 **压缩时间**、**压缩率** 等相关信息。

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 **压缩时间** 和 **压缩率** 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** （< 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
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-oq5N3.html

提取码:WXgu

提示与建议

1. **使用任意你喜欢的工具开发：**你可以使用任意语言，任意编辑器/IDE，在任意平台（Windows/macOS/Linux）上进行开发。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。你应减少你的程序在 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. **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。
7. **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。
8. **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漏洞。

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 过程中遇到非调试性问题可联系助教。

对于多名同学提出的共性的问题，助教会在 OJ 公告上做出解释。所以小窗助教前请先看一下 OJ 上的问题是否解决了你的疑惑。