实验：基于Huffman编码的文件压缩/解压工具

问题描述

在合适的情况下，利用 Huffman 编码对文件进行压缩可以减少其占用空间，同时在需要使用到文件的时候也可以根据压缩文件中所提供的信息来将其还原为原文件。本次实验中，我们将实现一个基于 Huffman 编码的文件压缩/解压缩工具。

基本要求

基于 Huffman 编码实现一个压缩器和解压缩器（其中 Huffman 编码以字节作为统计和编码的基本符号单元），使其可以对任意的文件进行压缩和解压缩操作。针对编译生成的程序，要求压缩和解压缩部分可以分别独立运行。具体要求为：

- 每次运行程序时，用戶可以指定只压缩/只解压缩指定路径的文件。实现的时候不限制与用戶的交互方式，可供参考的方式包括但不限于
    - 根据命令行参数指定功能（压缩/解压缩）和输入/输出文件路径
    - GUI 界面
    - 运行程序后由用戶交互输入指定功能和路径
- 【CAUTION!】不被允许的交互方式： 通过修改源代码指定功能和文件路径
- 压缩时不需要指定解压文件的目标路径，解压缩时不需要指定压缩前原文件的路径，压缩后的文件可以换到另一个位置再做解压缩

基本部分的验收方式

实验采用线下检查验收，验收时的测试数据不需要自己准备，这些测试数据包括：

- 一份 txt 文档
- 一张 png 格式图片
- 一份 wav 录音文件
- 一段 mp4 视频
- 一个 zip 压缩包，里面包含上述所有文件，并有一定的文件结构

具体检查流程与考察点如下：

- 展示原文件目录，打开 txt 文档并将其内容修改为随机文本
- 运行程序，对所有文件依次进行压缩，并展示各自的 Huffman 树，检查点包括除了 txt 文档外的 Huffman 树和压缩前后文件大小
- 将程序和压缩后的文件拷贝至另一处，同时原文件也移动到另外一个目录，具体路径由助教临时指定
- 对所有压缩后的文件进行解压缩，检查点包括 txt 文档内容是否与之前的随机文本一致，解压后的其他文件能否正常查看、播放、使用，将还原后的 zip 压缩包解压后内部文件结构和文件是否正常

基本部分的分数占比为：

- 代码实现 80%

选做内容

本次实验提供了两个可选的加分项，选做部分的加分占比为

- 指定符号单元大小：10%
- 指定多元 Huffman 压缩：10%

两个加分项如下：

- 【实现可指定的任意基本符号单元大小的 **Huffman** 压缩/解压缩算法】原先我们的 Huffman 编码是对原文件的每一个字节为基本的符号单位进行统计和编码，试修改你的压缩/解压缩器，使其可以指定基本符号单元的大小，以 0.5 个字节为其大小变化的粒度
  - 当基本符号单元的大小不再是字节的时候，可能会出现原文件本身大小并非基本符号单元大小整数倍的情况，你需要在不改动算法核心的前提下解决这个问题，一种可能的解决方案是将原文件填充到合适的大小，并通过为压缩文件的头部增添额外的信息来让你在解压时可以去掉填充的那部分
  - 验收时检查以 0.5 字节和 $x(x \in [1.5, 4] \wedge x = 0.5k, k \in N^*)$ 字节为基本符号单元大小的压缩/解压缩情况，二者分别占该加分项 35%、65% 的加分（并非完全严格的比例，根据实际情况将作一定调整）
  - 在实验报告中，你也可以尝试统计不同基本符号单元大小下的 Huffman 编码压缩/解压缩情况并试做出分析（不做强制要求）
- 【实现可指定的任意多元 **Huffman** 压缩/解压缩算法】 原先我们的 Huffman 编码是针对基本符号单元生成二叉树进行编码，试修改你的压缩/解压缩器，使其可以指定生成 n 叉树进行 Huffman 编码，要求展示修改后压缩/解压缩指定文件时的 n 叉树
  - 你可能需要在基本符号单元集合中增加权重为 0 的额外的占位符
  - 验收时检查指定 3 叉树和 $x(x \in [4, 16] \wedge x \in N^*)$ 叉树时的压缩/解压缩情况，要求展示生成的 n 叉 Huffman 树，二者分别占该加分项 35%、65% 的加分（并非完全严格的比例，根据实际情况将作一定调整）
  - 在实验报告中，你也可以尝试统计不同 n 值下 n 元 Huffman 编码压缩/解压缩情况并试做出分析（不做强制要求）