压缩大师

题目背景

罗恩一直奇怪，赫敏的串珠小包为什么能放下那么多东西。

这是！？计算机科学家的敏锐直觉告诉你，是~~无痕伸展咒~~压缩算法！（飞来咒大概是语音识别……）

任务描述

现在给你一些**图片**处理后的数据`{id}.txt`（编号对应的JPEG为对应图片，供探索数据性质），现请你实现一种图片压缩算法，补全hpp中对应的编码、解码函数，将上述图片数据进行压缩。

任务的目标分为三类：

- 压缩率：使编码后文件更小
- 效率：使整个编、解码过程尽可能快
- 准确率：使经过编解码后的文件所表示的图片与原图差异尽可能小（即可实现**有损压缩**）

分数评定

注意：本次题目**并未设置**各种形式的捷径，请勿进行各类奇怪尝试，任何

1. 故意尝试绕过原程序逻辑存储、发送数据的
2. 故意修改原程序评测逻辑的
3. 其他编写**恶意**代码的
4. 故意**短时间内多次提交**卡评测干扰他人的（本次程序因为包含计时，所以可能有**小幅度变化**但不至于逆天改命）

无论分数如何变化按分数锁定为1处理（即该次机考直接垫底：D）

分数的计算公式如下：

- 总分数 = $\Sigma_{i\in\text{数据点}}\text{单点分数}_i$
- 单点分数 = $\frac{\text{准确率}^5}{0.4\times\text{压缩率}+0.2}+\text{效率评定值}$
- 压缩率 = $\frac{\text{压缩后文件大小}}{\text{原文件大小}}$
  - 注意：这里的原文件大小指**并非**txt大小，而是$\text{长}\times\text{高}\times\text{颜色}(3)\times+2\times 2^4$ 字节(即直接存储成二进制所用的空间)
- 准确率通过SSIM(structural similarity)来反映
  - $SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$
  - μ：平均值 σ：标准差/协方差
  - $c_1=(k_1L)^2$, $c_2=(k_2L)^2$, $L=2^{\#\text{bits per pixel}}-1$
  - $k_1=0.01$, $k_2=0.03$
  - 准确率 = $max(SSIM,0)$

    - 对于多颜色，SSIM为对三种颜色分别单独求SSIM后取平均
    - **注意**：如果还原的图片尺寸与原图不同，**准确率记为0**，如果使用压缩后尺寸不同的算法，请在解码阶段自行进行拉伸
- 效率评定值 = $\frac{1}{\text{你的程序用时}\times 8+0.2}$
    - 注意：你的单点用时不应超过**0.1**秒（否则按0分计算）

输入格式

`{id}.txt`

注意：其实如果不看数据可以无视该文件的格式

第一行为两个整数，H，W分别表示高和宽

接下来的$H\times W$行，每行三个$0-255$的数字，分别表示该像素点的RGB值

注意：迭代顺序**行优先**，即像素坐标依次为$(0,0),(0,1),\cdots,(0,W-1),(1,0),\cdots,(H-1,W-1)$

程序模板说明

`coder.h`

无意外不会改动，提供给大家进行本地测试。**不需要通篇阅读**（当然如果你对下方某条叙述不清楚，也欢迎阅读具体实现）。

类型简述：

- Raw
    - 一个用于存储压缩后数据的数据类型，可以方便的进行和文件的操作。
    - **不可拷贝**
    - `resize`函数**会清空当前内容**并分配好新大小的空间
    - 可使用下标运算符来进行内容访问（以char计）
    - `show()` 可用于调试时展示内容（如果使用注意int的倒置，具体参见实现）
    - `getSize()` 可以得到当前大小
- Pic
    - 用于表示一张图片的RGB原始数据
    - **不可拷贝**
    - `h, w`为长高，请勿直接进行更改（虽然是`public`），而应使用`resize(new h, new w)`，**会清空当前内容**并分配好新大小的空间
    - 图片RGB信息以`uint_8`形式存储在`data[color][height][width]`内
    - 涉及指针操作，请勿频繁调用原数据
- Timer
    - 一个计时类

注意事项：

- 涉及到精确计时、编译**请使用c++11及以上版本**（如不能达到请自行尝试通过删除解决报错）
- 开头的`#define DEBUG`用于输出可视的反馈，请勿删除
- 编译后调用时请使用：

    - `./[your_program] -e [src] [des]`用于压缩src到des
    - `./[your_program] -d [src][des]`用于解压src到des
    - `./[your_program] -t [src]`用于使用src的图片测试你的压缩性能
- Windows系统在输出时**可能**出现少量乱码是**正常现象**，无视即可（加颜色的部分）
- 模拟测试的分数可能与服务器上测试分数产生不同，这主要是由于你机器与服务器运算能力不同导致的

`coder.cpp`

编码器和解码器的实现，即你需要填写并上交的部分。

如果你觉得助教提供的类**不符合你的编程习惯或设计风格**，请**自行编写接口**进行转换。

```
#include <algorithm>
#include "coder.h"
// Notice: Raw and Pic object are not copiable
void encode(const Pic & pic, Raw & raw)
{
    // TODO: you need to resize Raw to fit your encoder
}
void decode(const Raw & raw, Pic & pic)
{
    // TODO: you need to resize Pic to load the result of your decoder
}
```

由于Pic对象和Raw对象均不可拷贝，故请直接原址resize（再次提醒注意resize会清空内容）后修改。

一些其他的附件

`test.sh`

参考的本地测试脚本`test.sh`可提供较为舒适的调优环境

- **不必须**使用
- 脚本在linux中运行
- 使用了一些依赖，请自行安装（特例：依赖安装**允许联网**）
- 如无法使用请**自行修改脚本或研究替代方式**

`display.py`

- **不必须**使用
- 调用格式`python3 display.py [src]`
- 可用于展示本次输入的txt格式的图片，并将图片以test.JPEG的格式保存于工作目录

测试数据集

- 998张涉及工具、动植物、乐器、人像、衣服、风景等等各门类的照片
- **尺寸均不大于255×255**
- 附件中的是按比例采样的其中100张图片以及其可打开的图片版本
- 正式测试时会**在整个数据集上**进行测试

知识背景

参考阅读：

https://en.wikipedia.org/wiki/Image_compression

https://zh.wikipedia.org/wiki/%E5%9B%BE%E5%83%8F%E5%8E%8B%E7%BC%A9

允许：访问上述链接和其他维基百科词条的内容

严禁：访问其他网站、获取与实际算法实现有关的代码