**实习一 动态规划算法及应用**

**【题目一】BMP灰度图像压缩**

**一、实习目的与要求**

**【问题描述】**

灰度图像的像素值范围在[0，255]之间，如果采用一个像素一个字节的存储方式，势必会造成空间的浪费。如果采用一定的无损压缩算法，可以大大提高减小文件大小，减少存储空间。本课题要求针对提供的256色（8位）位图数据，采用教材上第15章动态规划中图像压缩算法（图像分段合并的思想），设计一个类，实现灰度位图数据的压缩和解压过程。

**【基本要求】**

一个完整的灰度图像类应具有以下功能：

（1）对8位位图数据的读功能，提供ReadBitmap方法。

ReadBitmap方法有一个参数为输入位图文件名(\*.bmp)，它能解析8位位图文件格式，获取位图BITMAPINFOHEADER信息和每个像素的数据信息，放入内存中。

（2）对8位位图数据的写功能，提供WriteBitmap方法。

WriteBitmap方法有一个参数为输出位图文件名(\*.bmp)，它能将内存中的位图文件信息，按照位图格式，写到位图文件中保存。

（3）灰度图像压缩功能，提供Compress方法。

Compress方法有一个参数为输出压缩文件名(\*.img) ，它能将已经装入到内存中的8位位图信息，进行压缩，形成段标题和以变长格式存储的像素的二进制串，写入到文件中（注意：Img文件格式自行定义）。

（4）灰度图像解压功能，提供UnCompress方法。

UnCompress方法有一个参数为输入压缩文件名(\*.img)，它能解析Img文件格式，将其在内存中解压缩为8位位图信息，以便输出为位图文件。

（5）以上是该灰度图像类基本的四个方法，在实现时可根据需要扩充其他方法。在设计时，要使用面向对象的思想，考虑各个成员的访问权限。

**【提高要求】**

（1）基于Windows对话框界面，可选择输入/输出文件名，有压缩进度条显示。

（2）采用不同的数据集，比较其压缩比，采用最有效的压缩方式。

**【测试数据】**

lena.bmp，512\*512\*8

**【测试用例】**

类的测试用例如下：

CCompressImage Test;

Test. ReadBitmap(“lena.bmp”); 读原始位图

Test. Compress(“Out.img”); 压缩

Test. UnCompress(“Out.img”); 解压

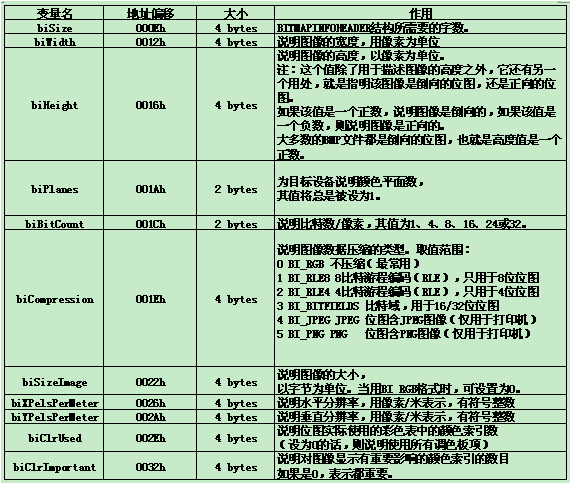
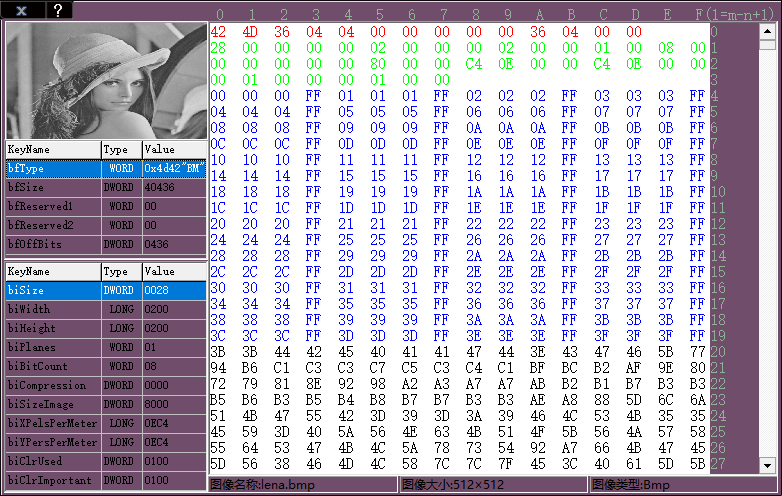
Test. WriteBitmap(“Out.bmp”); 还原位图信息

【测试结果】

可以使用MD5比较解压后的图与原图是否一样，验证你所实现的灰度图像类是否做到了无损压缩。

**【实现提示】**

有关8位的位图格式可以参考MSDN中BITMAPINFOHEADER结构的说明文档，注意其中biBitCount=8的说明。



**二、分析与设计**

1. **需求分析与类设计**

**需求分析：**

1. 压缩灰度图
2. 解压灰度图

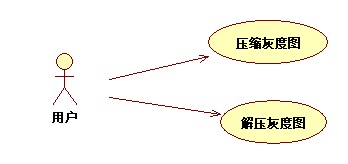


图1-用例图

**类设计：**

1. 位图类CCompressImage：

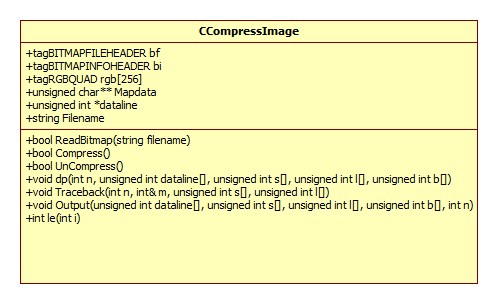


图2-类图

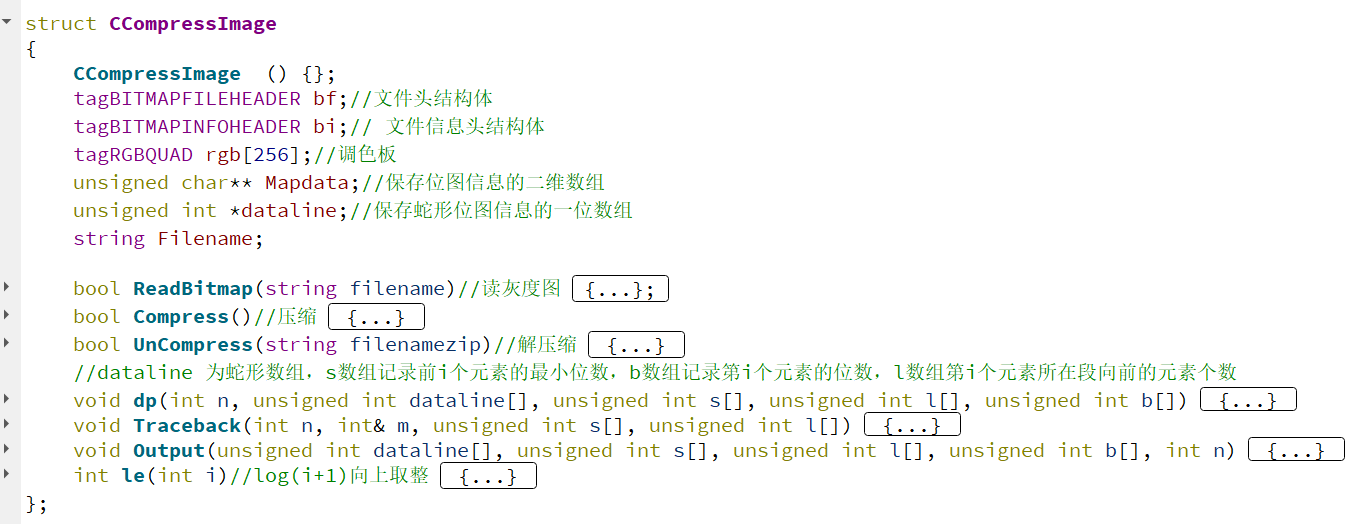


图3-类的结构显示

采用结构体实现封装；

设置了6个成员变量，分别用来保存位图头结构体、位图信息头结构体、位图颜色表结构、位图的数据信息、经过蛇形变换之后的数据信息、文件路径；

写了7个函数：其中ReadBitMap函数是用来读取位图文件的、Compress函数是用来输出经过压缩之后的文件、UnCompress函数是用来解压缩的、dp、Traceback、Output、le函数是对数组进行动态规划分段的函数；

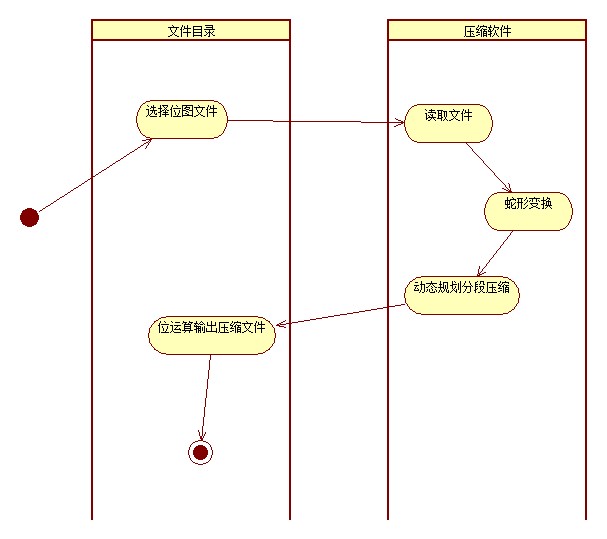


图4-压缩活动图

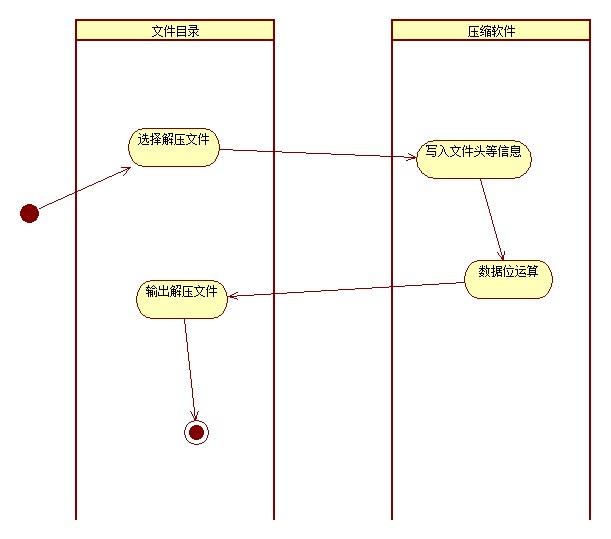


图5-解压活动图

1. **算法设计与分析**

将像素点的序列{p1,p2,p3…pn}分为m段，每段是所占位数相近的像素点，得到新序列为{s1,s2,s3…sm},每一段中包含一个或者多个像素点,取这一段中像素占用最大的位数作为当前段所有的像素点所占的位数(如s1中的最大像素是128，则取7位为s1中所有像素的所占的位数)，用L[t]记录第t段中所包含的像素点的个数，每个占用b[t]位（如前面举例的s1是取7位作为此段的像素点所占的位数，若还有其他的需要1位，2位。。。8位，所以b[t]需要3位要标识当前段所占的位数，7位就表示成110，1位就表示成001）。所以我们要引入一个段头来记录当前段的L[t]和b[t]，段头的位数就是L[t]的位数8加上b[t]的位数3,一共是11位。

1. 读位图：

首先了解到位图的结构是有头文件结构、头文件信息结构、颜色表结构、位图数据，所以使用二进制方式打开文件之后，读取14bit的头文件结构、再读取40bit的文件头信息结构、再读取4\*256的颜色表结构，之后就是位图数据；然后使用一个二维数组装位图的数据。



图6-读位图代码

[了解位图结构参考的博客](https://lyy289065406.github.io/exp-blog/gitbook/book/markdown/technical/deeplearn/%E5%9B%BE%E5%83%8F%E8%AF%86%E5%88%AB_Cpp%E8%AF%BB%E5%8F%96BMP%E4%BD%8D%E5%9B%BE%E5%85%A5%E9%97%A8.html)

1. 蛇形变换：

得到了二维数组数据后，进行蛇形变换将数据写入一维数组，通过判断奇数行偶数行进行蛇形变换，这里需要注意的是，从一维数组的下标1开始。



图7-蛇形变换代码

1. 动态规划：

获得蛇形变换之后的数组数据之后，参考提供的动态规划算法进行压缩数据。

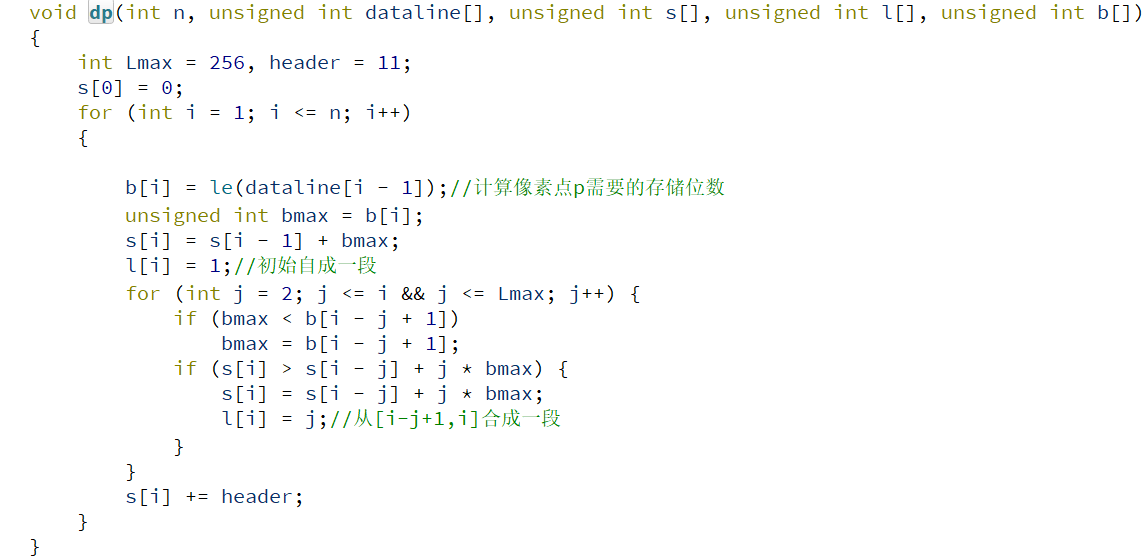


图8-动态规划代码

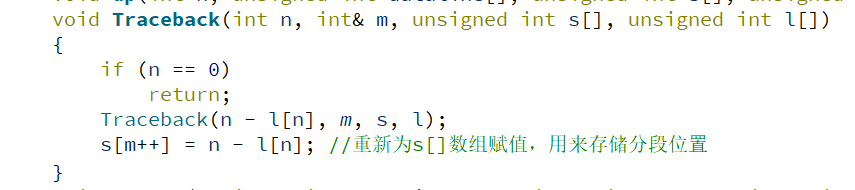


图9-动态规划

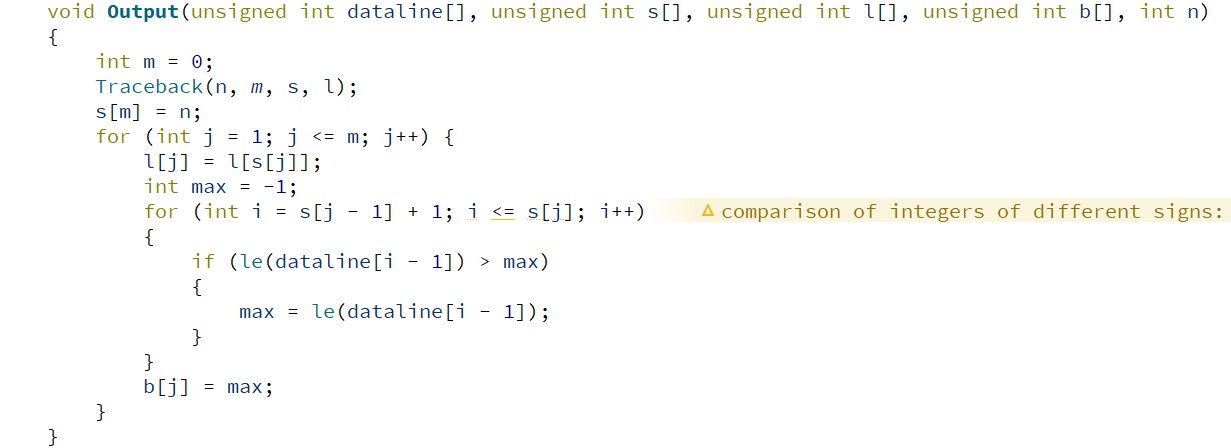


图10-动态规划

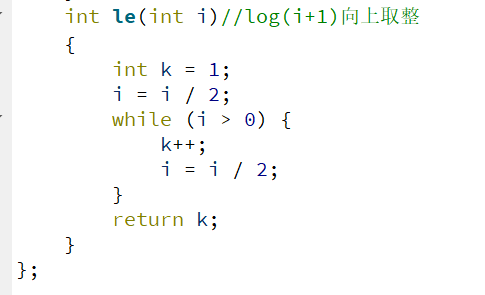


图12-动态规划

1. 位运算：

首先向文件中写入文件头、信息头、调色板信息，然后对数据进行处理。先将数据保存在一个vector中，对每一段都写入8位的l[i]，段内位数 3位的b[i]，每一段的像素值 maxbit 位的dataline[i]。写好了所有数据之后，对0进行清除，进行移位操作，每8位进行写入文件，当扫描到改位为0时，将temp的7-count位置0，否则将temp的7-count位置1；最后，如果最后一个字节不足8位，则进行补0；最后输出文件。



图13-写入头文件、信息头、调色板信息

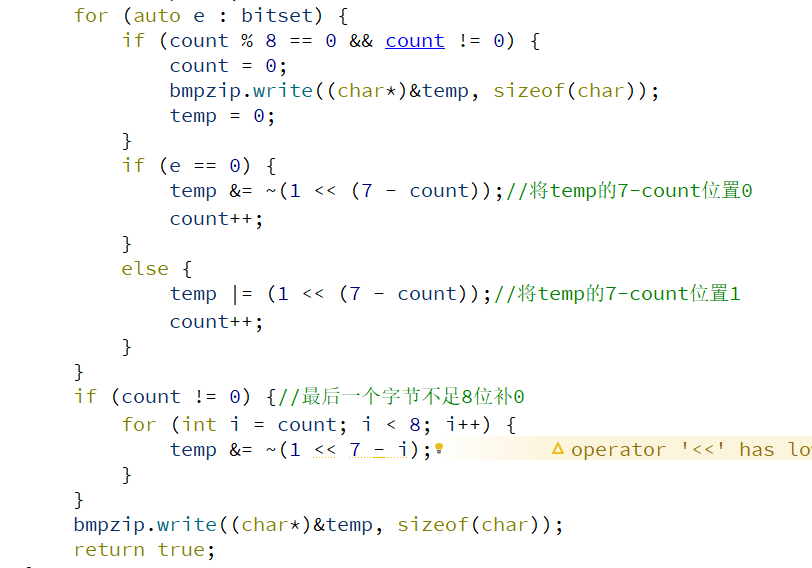


图14-位运算的核心代码

1. 解压：

首先写入头文件、头信息、颜色表信息，然后对压缩数据进行逆的位操作，从而得到原始的数据数组，然后再经过逆的蛇形变换，得到原始的数据；



图15-解压缩核心代码

**3、功能测试与改进**

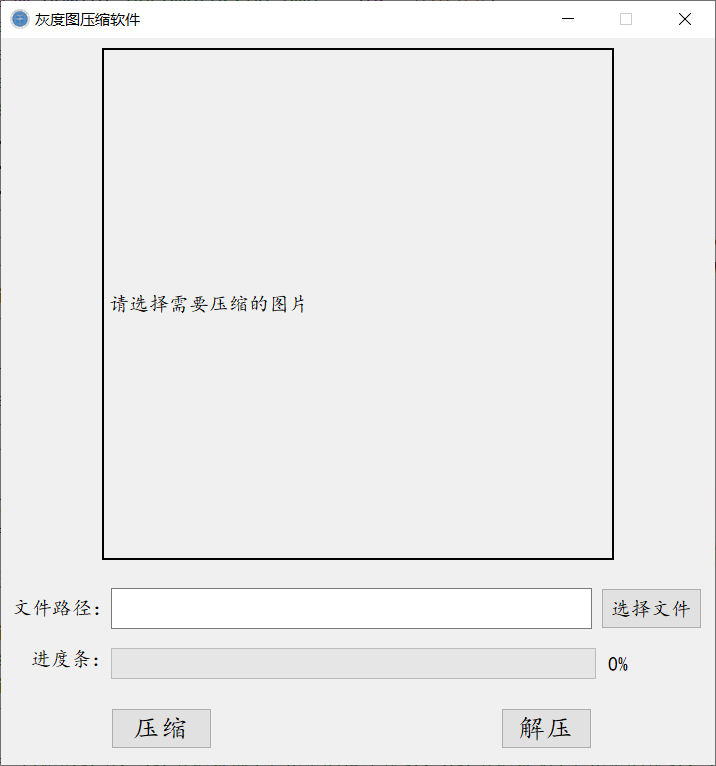


图16-软件界面

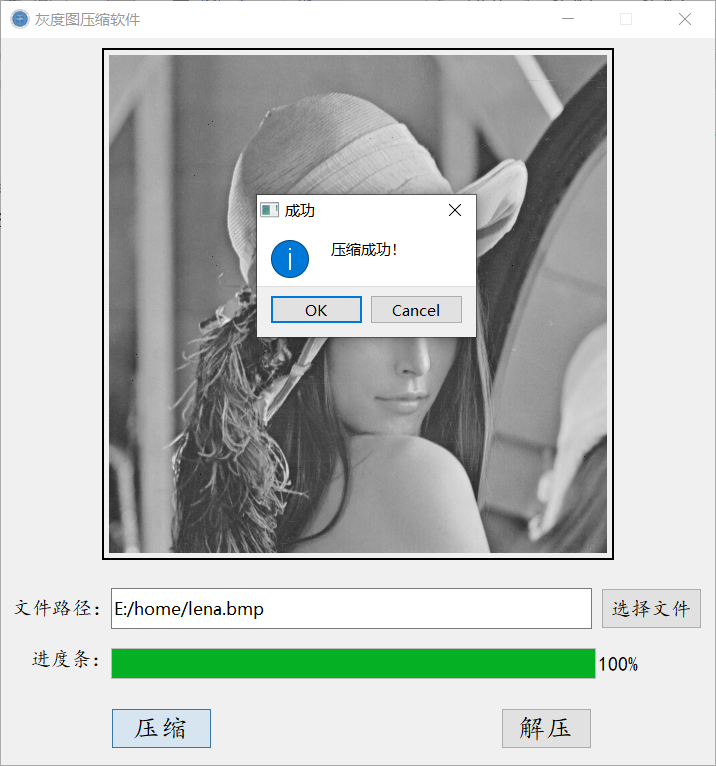


图17-压缩



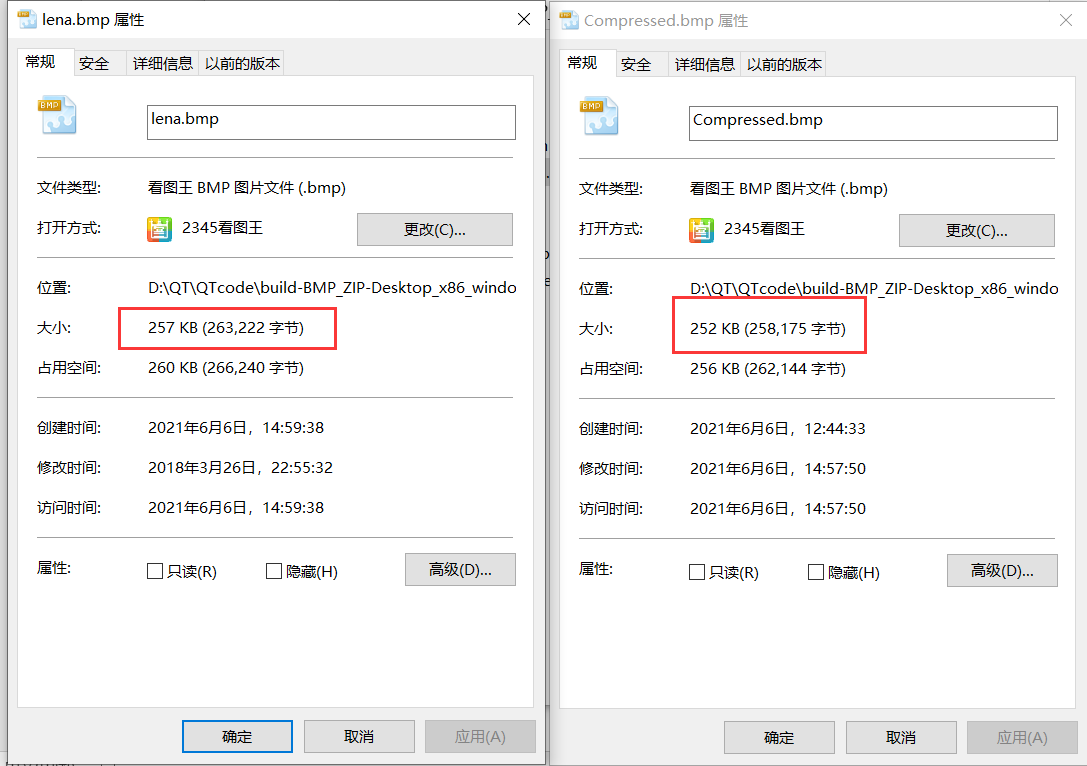


图18-压缩结果对比

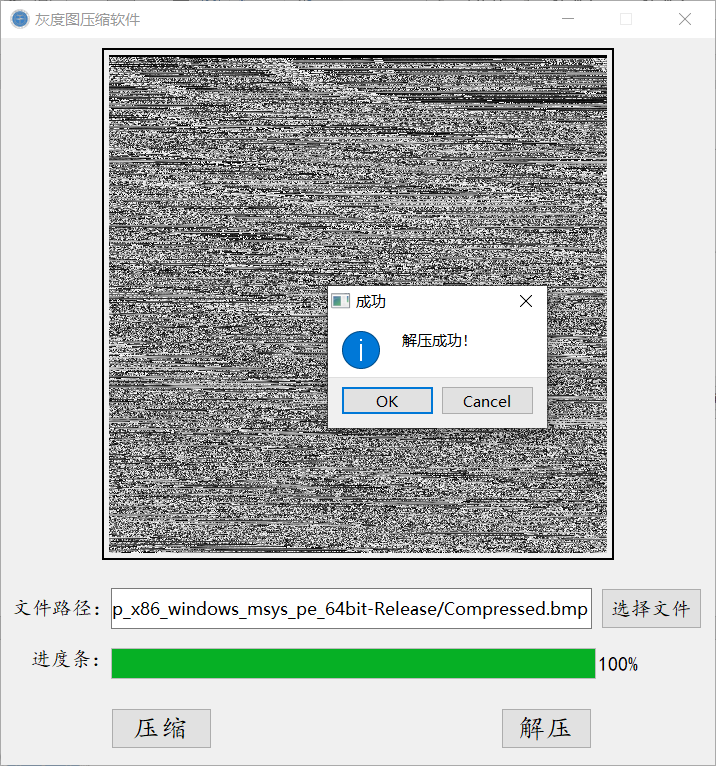
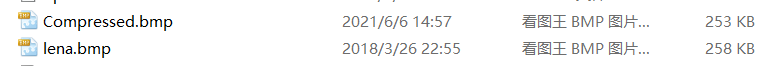


图19-解压





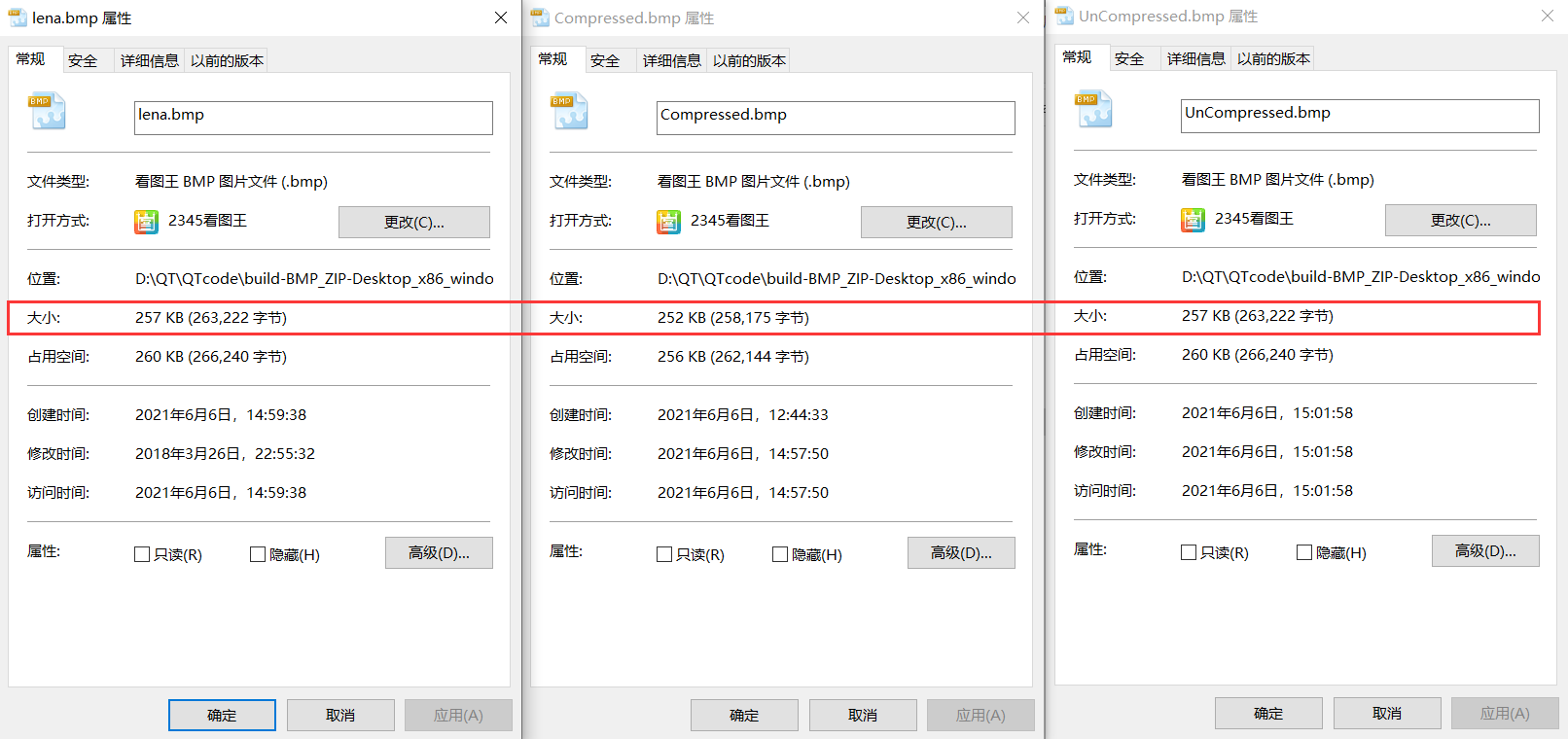


图20-解压图片对比

**三、实习小结**

通过本次实习，我对位图的结构、动态规划分段压缩、位运算等知识更加的熟练和了然了。

同时还需要额外加强的就是对进度条的正确使用，本次采用的是串行方法，所以不能对压缩进度的实时更新，正确的做法应该是采用多线程进行；

-------------------------------

成绩评定：

教师签名：

批改日期：