**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

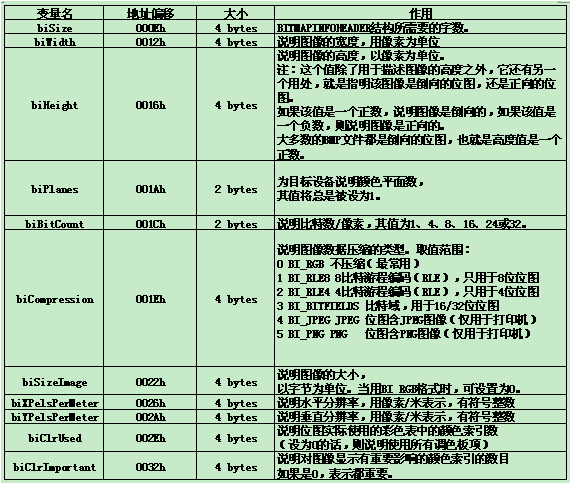
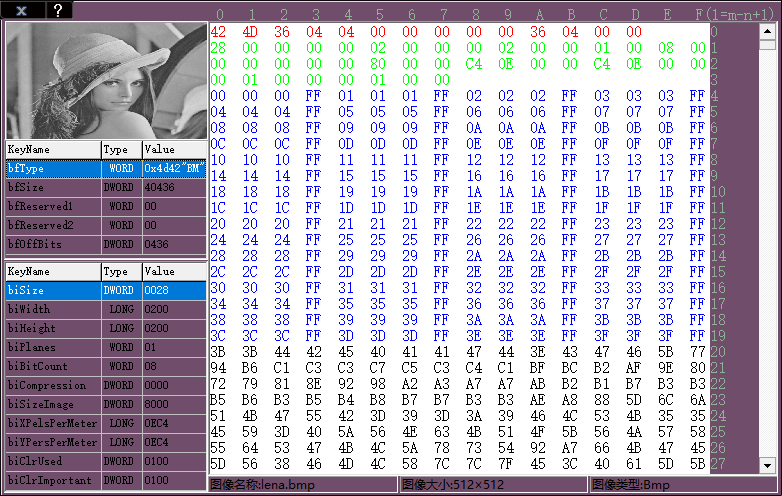
Test. WriteBitmap(“Out.bmp”); 还原位图信息

【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**
2. **需求分析：**

需要对深度为8的bmp文件进行压缩和解压。压缩过程先将位图数据进行蛇形存储，然后再进行动态规划分段存储，最后利用这个一维数据进行分段写入.img文件。

1. **类设计：**

class **compress** : public QWidget

{

Q\_OBJECT

public:

explicit **compress**(QWidget \*parent = nullptr);

//解析.bmp文件信息并压缩

void **CompressBmp**(string filename);

//解析.bmp文件信息并进行解压

void **UncompressBmp**(string filename);

private:

//存放文件信息头 位图信息头 颜色序列 数据

BITMAPFILEHEADER FileHeader;

BITMAPINFOHEADER InfoHeader;

RGBQUAD ipRGB[256];

//像素信息矩阵

unsigned char\*\*imgData;

//位图长宽高

int Height;

int Width;

//s矩阵

unsigned int\*snake;

signals:

void **Signal1**(int value);

void **Signal2**(int value);

**CompressBmp**(string filename) 根据传入的文件路径读取数据进行压缩。

**UncompressBmp**(string filename)根据传入的文件路径读取数据进行解压。

**Signal1**(int value)根据压缩任务进行进度修改进度条进度。

**Signal2**(int value)根据解压任务进行进度修改进度条进度。

另外使用到的辅助函数是：

int **length**(int i) 计算位长

void **dp**(int n, unsigned int \*p, unsigned int \*s, unsigned int \*l, unsigned int \*b) 动态规划进行分段

void **Traceback**(int n, int& i, unsigned int s[], unsigned int l[])利用回溯更新蛇形一维数组的数

int **Output**(unsigned int s[], unsigned int l[], unsigned int b[], int n) 重新存取位数和个数

1. **算法设计与分析**
2. **压缩：**

a)进行压缩操作时首先进行bmp文件读取。分别读取文件的文件信息头，位图信息头，RBG颜色序列，图形信息。用到的文件操作主要是：

ifstream f(&a,b); a为操作文件名，b为打开方式，这里使用二进制。

f.read(c,d);c代表存取读取到的信息变量，d代表存取信息位数。这里d必须严格按照每段不同数据长度进行存储，不然就出错。

这里需要注意的是：

实际位图数据的偏移字节数 = 文件头字节数+位图信息头字节数+RBG颜色序列字节数，读取是注意读取的字节数。

读取图形信息也不能一次性读完，必须分行读。按照这个公式算出行总占字节数：

(InfoHeader.biWidth\*InfoHeader.biBitCount / 8 + 3 ) / 4 \* 4;

b)对文件数据读取结束后，就对图形信息进行操作，也就是将图形信息进行蛇形存储操作，然后进行动态规划分段重新取值操作。

蛇形存储操作：

for (int i = 0; i < InfoHeader.biHeight; i++)//蛇形

{

if (i % 2 == 0)//奇数行

{

for (int j = 0; j < lineNum; j++)

{

snake[flag] = imgData[i][j];

flag++;

}

}

else if (i % 2 != 0)//偶数行

{

for (int j = lineNum - 1; j >= 0; j--)

{

snake[flag] = imgData[i][j];

flag++;

}

}

}

c)将文件信息，位图信息头，RBG颜色序列存入设定好的.img文件后，最后再分段存取图形信息。

由于每段数据灰度值最多有8位，用即3位数字来表示当前组的像素位数，剩下的才是数据，且至32位存入一次，所以必须分段存取。主要进行的操作拿分八位举例：

for(int i=1;i<=pointNum;i++){

if(track+8<32){

flag<<=8;

flag = flag|(l[i]-1);

track+=8;

}

else if(track+8==0){

flag<<=8;

flag = flag|(l[i]-1);

n\_f.write((char\*)&flag,sizeof(flag));

flag=0;

track=0;

}

else{ //track+8>32 就分段存32的存

int t =32-track;

flag<<=t;

flag= flag|(l[i]-1)>>(8-t);//先存前t位

n\_f.write((char\*)&flag,sizeof(flag));

flag=0;

flag = flag|(((l[i]-1)) << (32-8+t))>>(32-8+t); //再存后8-t位

track=8-t;

}//8位

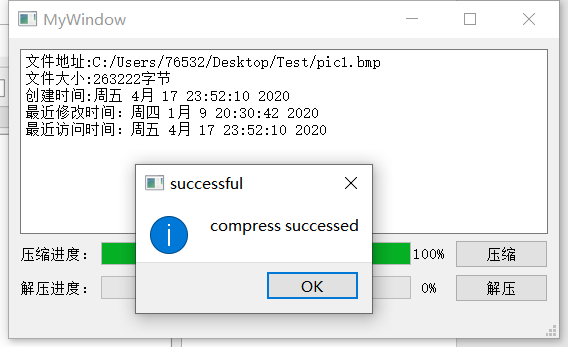
可以看到当前存储量+8如果超过32之后，先存储没有超过32位部分，然后中间变量存储后8-t位，下一个32位分段存储。

1. **解压**

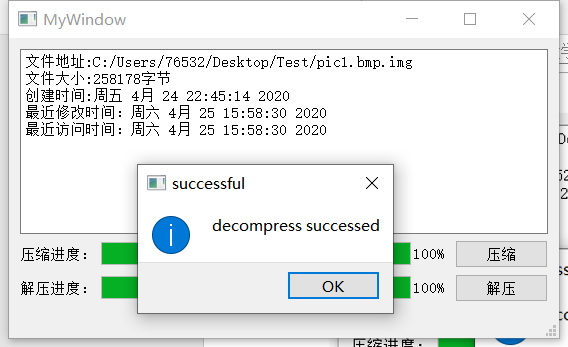
解压操作与压缩操作差不多，只不过在读取图像信息时，先将为8位的段长数据和3位数据最大位取出，再根据段长和位数将数据存入文件中。

**3、功能测试与改进**

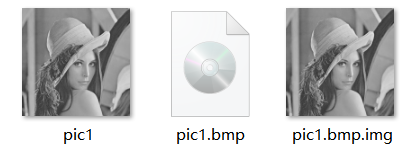
压缩：

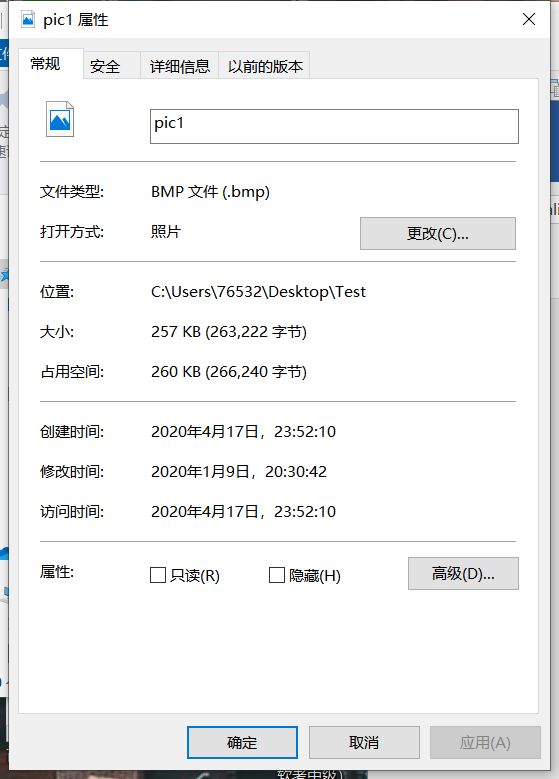
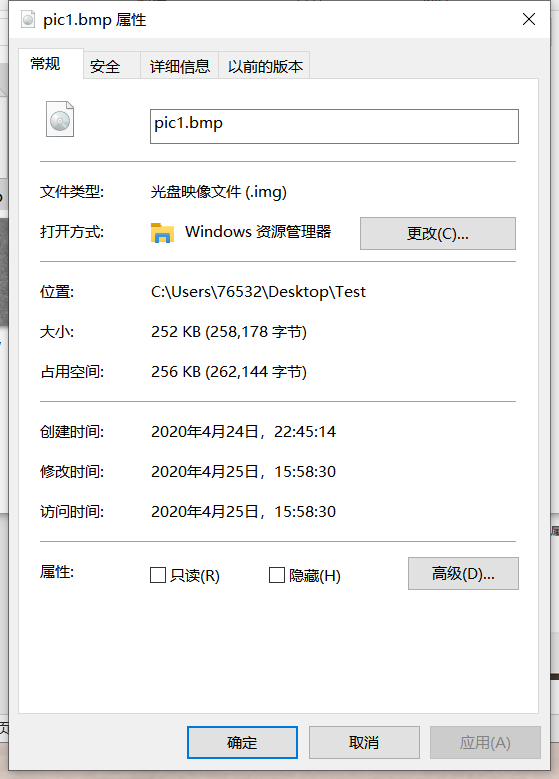


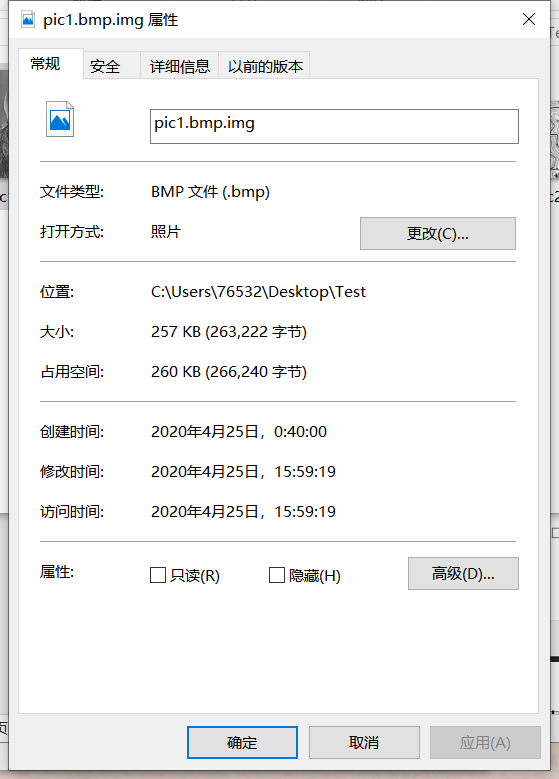
解压：



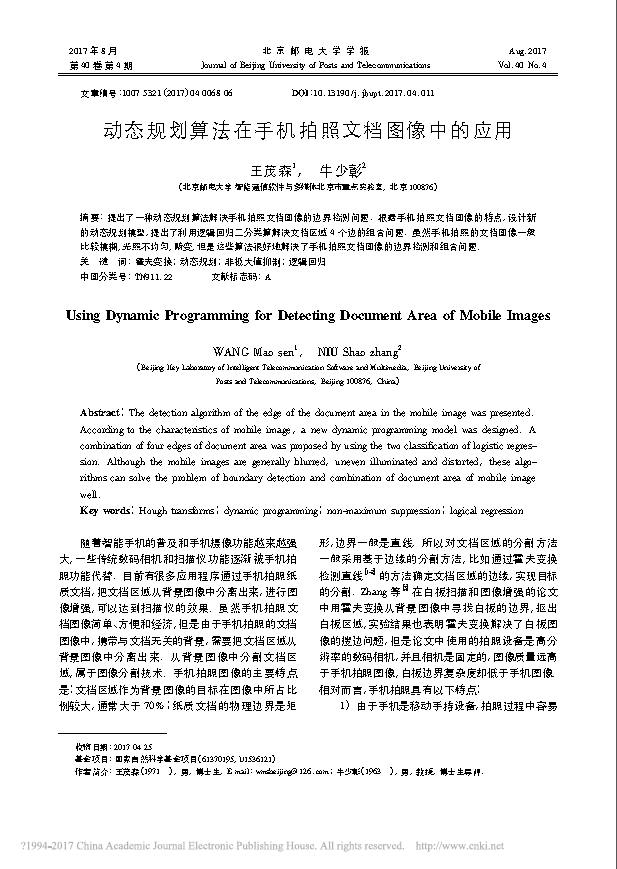
文件信息：





**【题目二】 动态规划算法在手机拍照文档图像中的应用**



**三、实习小结**

通过本次实习，让我了解了bmp文件的具体结构，也明白了动态规划压缩算法的原理。但是之前很少进行文件操作，也不明白文件各项结构，位数，大小等的转化问题，所以在理解方面花了很大的时间。另外在进行存取图形数据分段的操作时也非常困难，一旦一位操作出现错误之后，就会导致后面的读取出错，就会出现图片丢失数据，无法显示完整的图片的情况。总的来说，这次图片压缩的题目，还是收获了一些新的知识。

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

成绩评定：

教师签名：

批改日期：