**数字图像处理hw1**

15331416 赵寒旭

**Exercises**

**1.1 Storage (3 ∗ 3 = 9 Points)**

If we consider an N-bit gray image as being composed of N 1-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image and plane N all the highest-order bits, then given a 1024 × 2048, 128-level gray-scale image:

1. How many bit planes are there for this image? 

**Ans:**

7 bit planes

1. Which panel is the most visually significant one?

**Ans:** plane 7 contains all the highest-order bits, which is the most visually significant one.

3. How many bytes are required for storing this image? (Don’t consider image headers and

compression.)

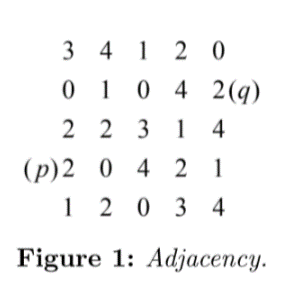
**Ans:** 1024\*2048个像素，每个像素7bits

1B = 8bits, 1Kb = 1024B, 1MB = 1024KB



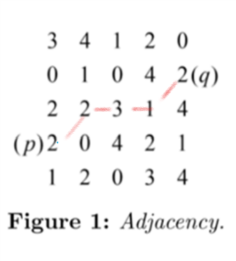
**1.2 Adjacency (3 ∗ 3 = 9 Points)**

Figure 1 is a 5 × 5 image. Let V = {1, 2, 3} be the set of pixels used to define adjacency. Please report the lengths of the shortest 4-, 8-, and m-path between p and q. If a particular path does not exist, explain why.

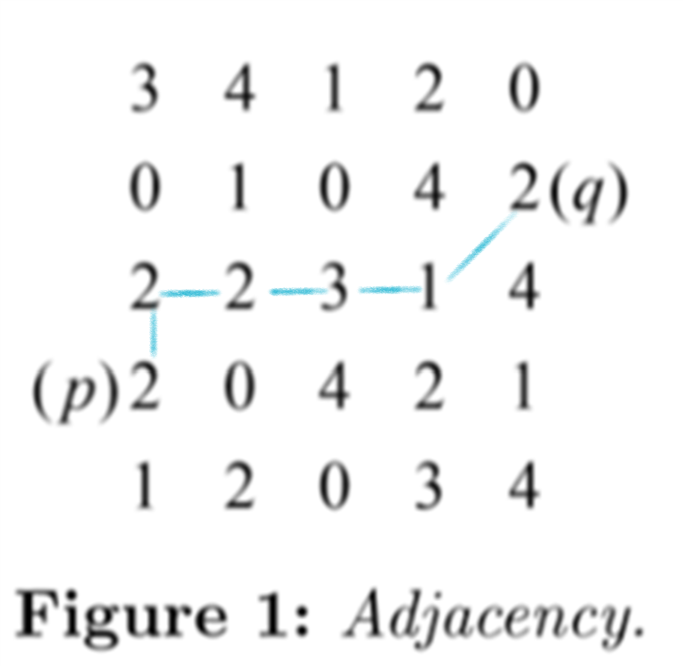


**Ans:** 4-path 不存在，p, q之间不存在这样一条路径使得路径上所有点互为4-neighbors

8-path 最短为4



m-path 最短为5



**1.3 Logical Operations (3 ∗ 3 = 9 Points)**

Figure 2 are three different results of applying logical operations on sets A,B and C. For each of the result, please write down one logical expression for generating the shaded area. That is, you need to write down three expressions in total.

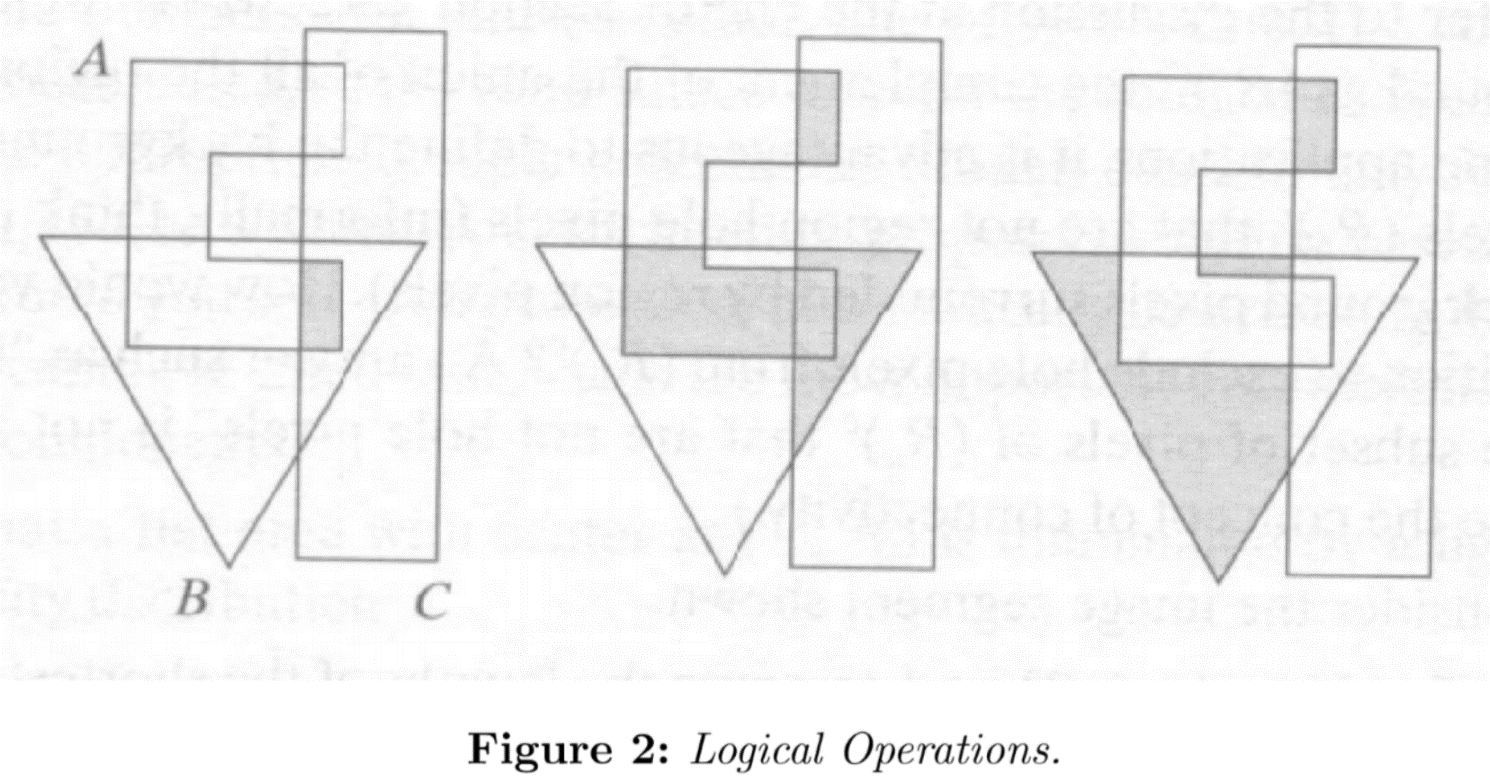
**Ans:** 图1：

图2：

图3：

**Programming Tasks**

**2.2 Scaling (45 Points)**

1. Down-scale to 192 × 128 (width: 192, height: 128), 96 × 64, 48 × 32, 24 × 16 and 12 × 8, then manually paste your results on the report. (10 Points)

192 × 128



96 × 64



48 × 32

C:\Users\ZhaoHanxu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\48X32_scaled.png

24 × 16



12 × 8

C:\Users\ZhaoHanxu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\12X8_scaled.png

2. Down-scale to 300 × 200, then paste your result. (5 Points)



3. Up-scale to 450 × 300, then paste your result. (5 Points)



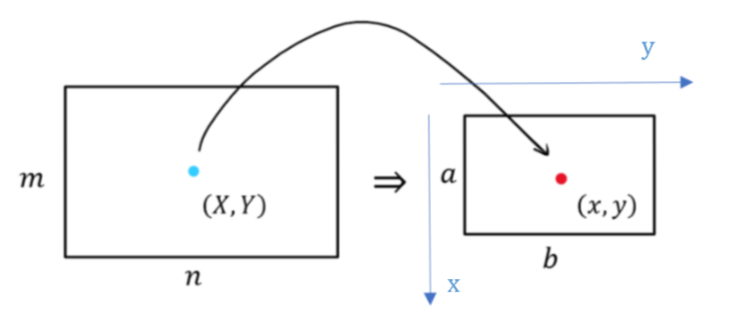
4. Scale to 500 × 200, then paste your result. (5 Points)



5. Detailedly discuss how you implement the scaling operation, i.e., the “scale” function, in less than 2 pages. Please focus on the algorithm part. If you have found interesting phenomenons in your scaling results, analyses and discussions on them are strongly welcomed and may bring you bonuses. But please don’t widely copy/paste your codes in the report, since your codes are also submitted. (20 Points)

Attention: you should not rescale your scaling results in your report, unless they exceed the page height/width.

算法理论：



已知原图像大小为，缩放后图像大小为,要通过双线性插值法求出放缩后图像x行y列像素点 的像素值。

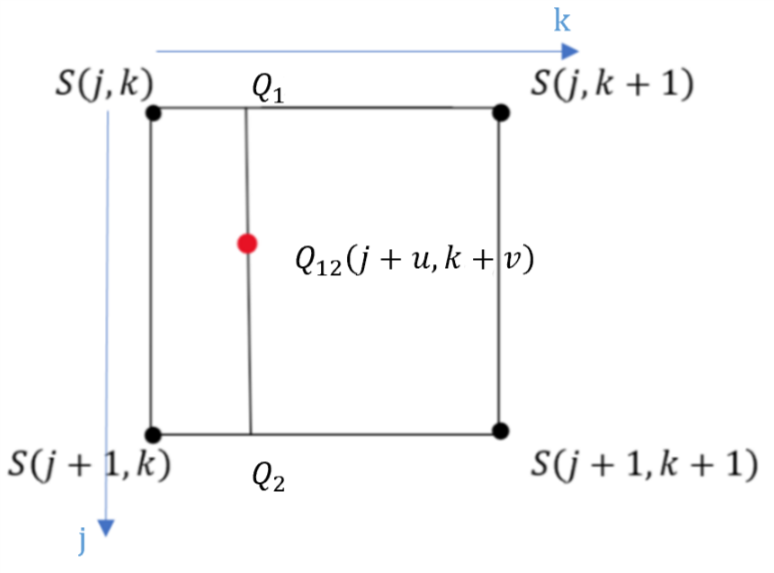
1. 找到原图中对应点



j, k分别为X，Y的整数部分，u, v分别为X，Y的小数部分。

1. 在原图中找到待插值的四个点



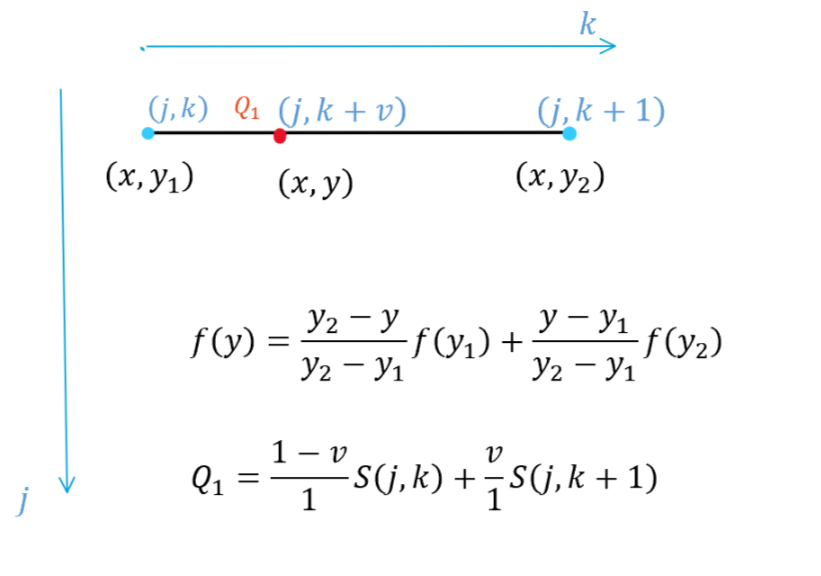


1. y方向插值分别得到， 的像素值

处像素值由和处像素值插值得到（权值大小由决定）。

容易知道,值越小则位置越接近，的像素值在插值中占比例越大，，故有：

注：考虑一般情况：





同理： 

1. 在x方向上对，插值得到，同理有：



的值即为插值所得对应转换图的像素值。

实现方法：

我们只需要根据缩放后图像的width，length，即可遍历输出图像的每一个像素点，通过放缩比例找到对应原图的像素点位置（可能为小数），找到该位置最近的待插值的四个点，由双线性插值求出输出图像在该点的像素值，遍历完成后就得到缩放后的图像。



**2.3 Quantization (28 Points)**

Write a function that takes a gray image and a target number of gray levels as input, and generates the quantized image as output. The function prototype is “quantize(input img, level) → output img”, where “level” is an integer in [1, 256] defining the number of gray levels of output. You can modify the prototype if necessary. For the report, please load your input image and use your “quantize” function to:

1. Reduce gray level resolution to 128, 32, 8, 4 and 2 levels, then paste your results respectively. Note that, in practice computers always represent “white” via the pixel value of 255, so you should also follow this rule. For example, when the gray level resolution is reduced to 4 levels, the resulting image should contain pixels of 0, 85, 170, 255, instead of 0, 1, 2, 3. (8 Points)

128 levels:



32 levels:



8 levels:



4 levels:



2 levels:



2. Detailedly discuss how you implement the quantization operation, i.e., the “quantize” function, in less than 2 pages. Again, please focus on the algorithm part. Analyzing and discussing interesting experiment results are also welcomed, but please don’t widely

copy/paste your codes in the report (20 Points).

算法理论：

对灰度图像进行量化处理，根据量化等级level确定原灰度值和量化后灰度值对应关系，为输出图像的各个像素点赋值，从而得到量化后图像。

分级样例：

2-level： 0，255

4-level：0，85，170，255

8-level：0，36，72，108，144，180，216，252

每一级灰度值范围长度

1. 确定原图灰度值对应量化后的等级

L向下取整得出浮点数的整数部分a

量化后等级应取最接近等级：

若，量化后等级newLevel = L+1

若，量化后等级newLevel = L

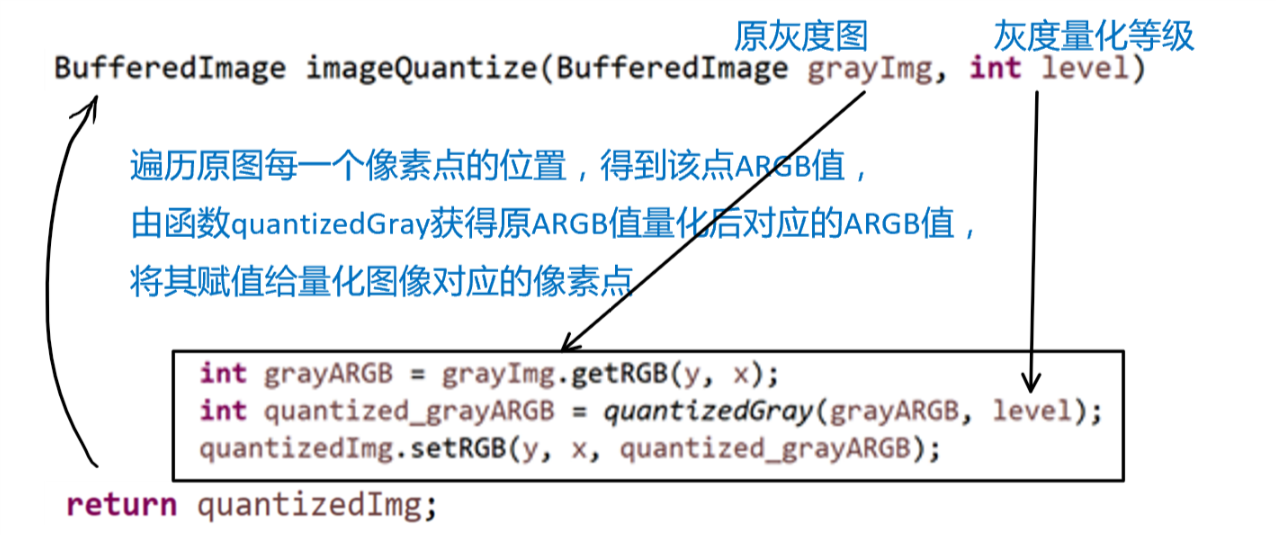
1. 由量化后等级确定像素值

易得：量化后灰度值 = length \* newLevel

实现方法：

输入BufferedImage 对象（待处理灰度图）和量化等级level，遍历灰度图像每一个像素点，获取其ARGB值，由上述算法量化后把新的ARGB值赋给此像素点，最后得到量化后图像。

注：灰度图RGB三个分量值均相同，故计算灰度值时可选取一个分量计算，具体实现时选取B分量代表此像素灰度值。



注意：

1. 在本文中坐标系下，若处理目标图像i行j列的像素点，调用setRGB(j, i), getRGB(j, i )才能得到正确的处理结果，否则可能会发生越界。
2. 作业提供的16.png输入图像是灰度图像，类型：TYPE\_BYTE\_GRAY，位深度：8

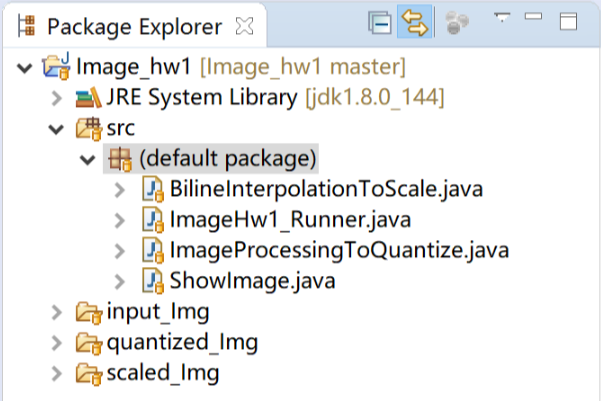
若处理图像时新声明BufferedImage对象使用TYPE\_INT\_ARGB类型，位深度为32，和原图不符合，会使输出图像效果不佳。

常量值：2

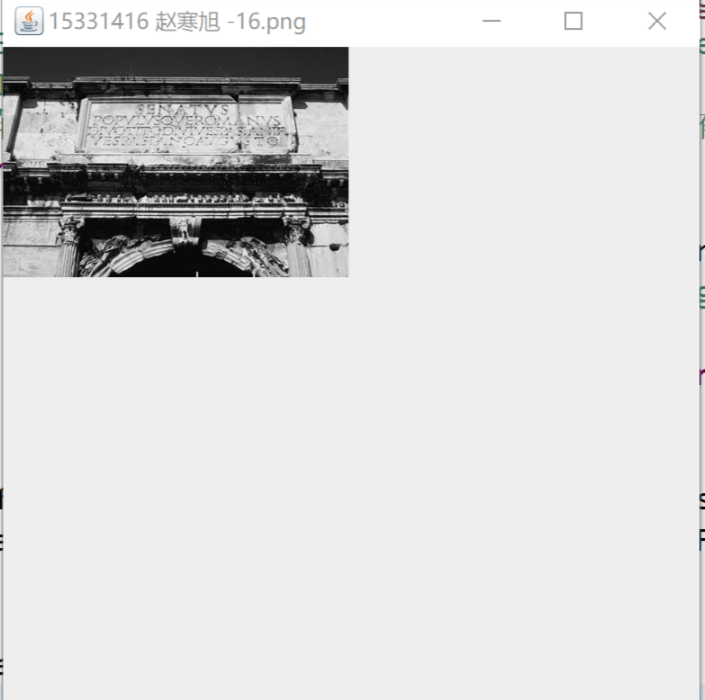
常量值：10

程序运行指南：

1. 程序结构如图

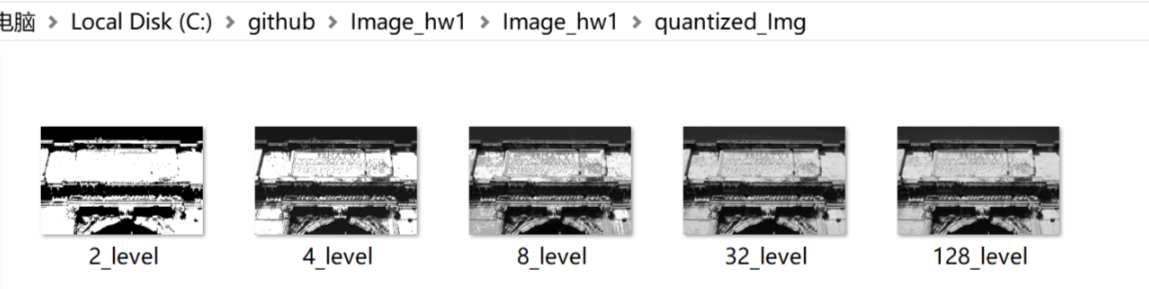


1. 运行Runner后，显示输入图像 16.png



并且在项目根目录下生成scaled\_Img和quantized\_Img两个文件夹，运行scale和quantize函数，同时把处理结果保存在相应文件夹中。





可以直接在文件夹中查看结果。

1. 通过jar包直接运行，生成两个文件夹，结果可直接查看

