1

1.1

(1)有 6 位，因为图像一共有 64 个灰度级，2^6=64

(2)平面 5，也即最上层的平面，因为它影响的是像素的最高位，最高位对整个数值的影响也是最大的

(3)1024*2048*6=12582912

1.2

令 p 点坐标为(3,0)，q 点坐标为(1,4)

(1)不存在 4 邻接通路，因为 q 点的三个四邻接像素是 4 或 0，都不在集合{1,2,3}中

(2)最短的 8 邻接通路距离为 4，路径为(3,0)->(2,1)->(2,2)->(2,3)->(4,1)

(3)最短的 m 邻接通路距离为 5，路径为(3,0)->(2,0)->(2,1)->(2,2)->(2,3)->(4,1)，与 8 邻接最短路径的区别是(3,0)和(2,1)两点不是 m 邻接

2

2.1

(1)依次是 92×128，96×64，48×32，24×16 和 12×8

(2)300*200

(3)450*300

(4)500*200

(5)

缩放方法的关键是找到一个映射关系，从原图像到目标图像的映射，当然，在双线性插值的算法中，目标图像的一个像素要与原图像中的 4 个像素对应，通过这 4 个像素的值来计算出目标图像中的像素，在找像素的时候应该做到的一点是让目标图像中的像素与原图像中的像素实现中心对齐，而不是在某个角上对齐，否则的话就会变成最近邻，原图像的边缘没有得到计算，造成图像边缘失真

关于坐标映射关系如下

```
s_x = (j + 0.5) * (s_width / width) - 0.5
s_y = (i + 0.5) * (s_height / height) - 0.5
```

s_x：目标像素(j，i)所对应的原图像中代表宽度的像素值

s_y：目标像素(j，i)所对应的原图像中代表高度的像素值

i：目标图像中代表宽度的像素值

j：目标图像中代表高度的像素值

s_width：原图像宽度

s_height：原图像高度

width：目标图像宽度

height：目标图像高度

在这里说一下这里有一个坑

我们平时描述图像的时候一般是宽度*高度，比如 500*200 的图像表示宽度为 500，高度为 200 的图像，但是在 python 中，一个图像是用一个三元组来表示的(height，width，level)其中 level 的值为 0，1，2，但是前两个值第一个值表示高度，第二个值表示宽度，这和我们平常的习惯不相符

求出来的这两个值可能带小数，也可能不带，我们直接把它取整，同时再加 1，这样又会得到 4 个数字，

```
n1 = int(s_x)
n2 = int(s_y)
n3 = n1 + 1
n4 = n2 + 1
```

所以四个点的坐标分别就是(n2,n1),(n2,n3),(n4,n1),(n4,n3)(在这里采用了 python 里面表示图像的方法，高度在前，宽度在后)
这样的话我们就把需要的数据全部就位，然后使用双线性插值法进行计算
在双线性插值的计算公式上面，我参考了一些网上的资料
下面的公式中，目标图像的像素是(x,y),
原图中 4 个对应的像素分别是(x1,y1),(x1,y2),(x2,y1),(x2,y2),
在 X 轴方向进行插值

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{Where} \quad R_1 = (x, y_1),$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{Where} \quad R_2 = (x, y_2).$$

在 Y 方向进行插值

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

然后将上面的两个式子带入下面，得到

$$f(x,y) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y)$$

$$+ \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1).$$

代码实现如下

```
v1 = (n3 - s_x) * src[n2, n1, n] + (s_x - n1) * src[n2, n3, n]
v2 = (n3 - s_x) * src[n4, n1, n] + (s_x - n1) * src[n4, n3, n]
res[i, j, n] = int((n4 - s_y) * v1 + (s_y - n2) * v2)
```

然后用我们计算出来的矩阵构造新图像，算法基本完成

2.2

(1)

层次级别从上到下依次是 2，8，32，64，128

(2)这道题主要是让我们把图像分级，原来有 256 级，我们要做的是把它重新分为 2，8，32，64，128 级，举例来说，如果分为 2 级，那么像素就只有 0 和 256 的区别，
所以大体思路就是把一系列的像素值归为一个值，在这里我使用的是先整除再乘的方法

```
dis=256/level
for k in range(3):
    for i in range(height):
        for j in range(weight):
            res[i][j][k]=src[i][j][k]//dis*dis
```

这里面 level 是我们分出来的级别数
dis 是每个区间的宽度
在这里遍历像素，然后对像素进行先整除然后再乘区间宽度的做法来把像素进行归类，通过对 level 的调节得到上面的结果