1. **题目及要求**

题目 D-1 哈夫曼编码算法的仿真和比较

目的：掌握基本哈夫曼编码的原理和 Matlab 的实现。  
要求：

（1） 产生[0 255]范围内均匀分布、正态分布、拉普拉斯分布的三组离散随机整数。每组数据个数为 1920\*1080，并对这三组数据分别进行哈夫曼编码和解码。计算熵和编码平均码长。概率分布的参数自行选定。

（2）选取 3-5 幅典型的彩色图像， 将像素的 R、 G、 B 值作为独立符号，进行哈夫曼编码和解码，计算熵和符号平均码长。

（3）选取 3-5 幅典型的彩色图像，将像素的 R、 G、 B 值作为联合符号，进行哈夫曼编码和解码，计算熵和符号的平均码长。

试比较（2）、（3）情况的编码效率。

1. **原理**

（一）哈夫曼编码

[哈夫曼](http://baike.baidu.com/view/1436260.htm)编码(Huffman Coding)，又称霍夫曼编码，是一种编码方式，哈夫曼编码是可变[字长](http://baike.baidu.com/view/731.htm)编码(VLC)的一种。Huffman于1952年提出一种编码方法，该方法完全依据[字符](http://baike.baidu.com/view/263416.htm)出现概率来构造异字头的平均长度最短的码字。Huffman编码将给字母分配编码。每个字母的编码的长度取决于在被压缩文件中对应字母的出现频率，我们称之为权重(weight)。每个字母的Huffman编码是从称为Huffman编码树的满二叉树(所有节点要么有左右两个子孩子，要么就没有子孩子)中得到的。Huffman编码树的每一个叶节点对应于一个字母，叶节点的权重 （weight）就是它对应的字母出现的频率。

（二）构建Huffman编码树的过程

1. 创建n个初始化的Huffman树，每个树只包含单一的叶节点，叶节点纪录对应的字母和该字母出现的频率(weight)；
2. 按照weight从小到大对其进行所有的Huffman树进行排序，取出其中weight最小的两棵树，构造一个新的Huffman树，新的Huffman树的weight等于两棵子树的weight之和，然后再加入到原来的Huffman树数组当中；
3. 反复上面的2中的操作，直到该数组当中只剩下一棵Huffman树，那么最后剩下来的那棵Huffman树就是我们构造好的Huffman编码树；

（三）哈夫曼编码及解码的过程

编码过程：

1. 获取要压缩的数据

2. 统计每个值出现的频率

3. 开始构造原始的huffman编码树数组，用于构造Huffman编码树

4．构造huffman编码树，并且获取到每个字符对应的 编码

5．对数据进行压缩

解码过程：

1. 读取压缩文件中保存的树的叶节点的个数

2. 读取压缩文件中 ［0-255］出现的频率的信息，用于构造一个字典 char\_freq 一遍重建 Huffman编码树

3. 重建huffman 编码树，和压缩文件中建立Huffman编码树的方法一致

4. 使用步骤3中重建的huffman编码树，对压缩文件进行解压缩

（四）参数评价

1. 信息量、图像的熵与编码平均长度

令图像像素灰度级和为，其对应的概率分别为，根据香农信息论，其信息量为：

图像的熵定义为：

平均码长：

1. 编码效率

编码效率定义为

1. **实验和结果分析**
2. 问题1

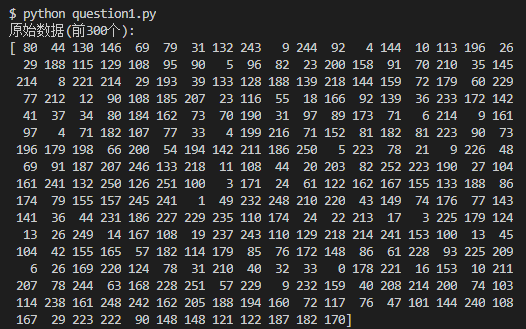
先使用numpy产生[0 255]范围内均匀分布、正态分布、拉普拉斯分布的三组离散随机整数，每组数据个数为 1920\*1080，并对这三组数据分别进行哈夫曼编码和解码。计算熵和编码平均码长。

具体过程如下：

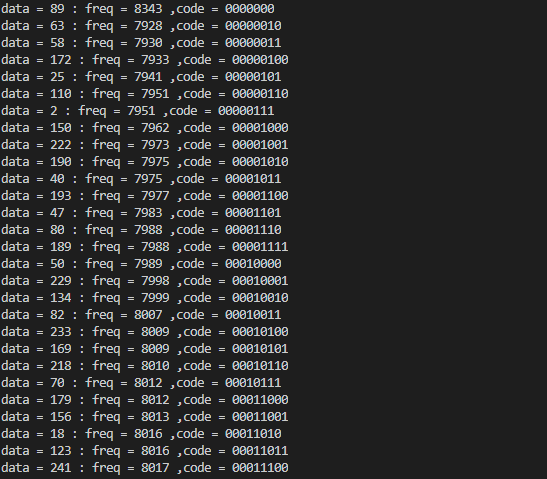
运行：question1.py

（1）均匀分布：

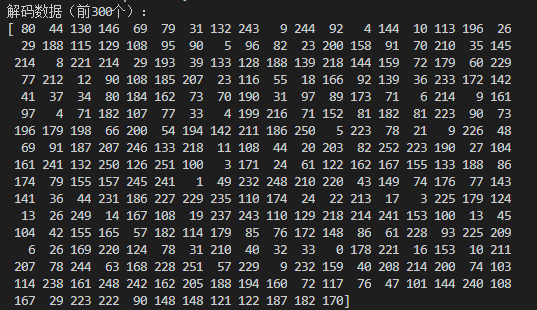
生成数据：



编码过程（部分）：



解码数据（前300个）

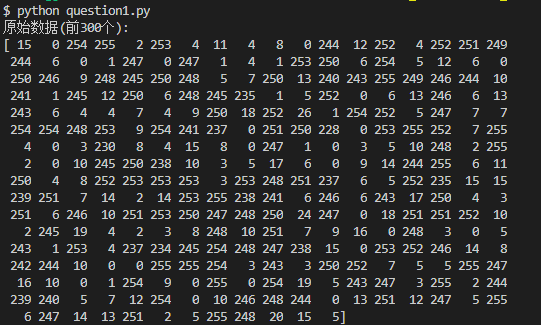


熵、编码平均码长及编码效率

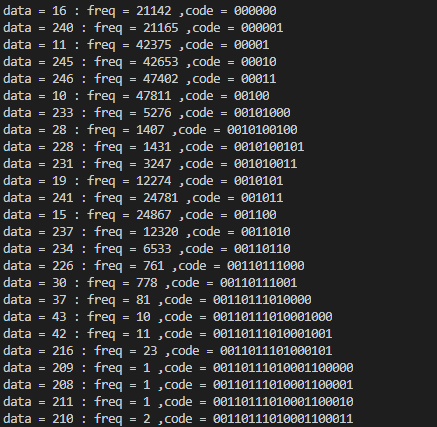


（2）正态分布

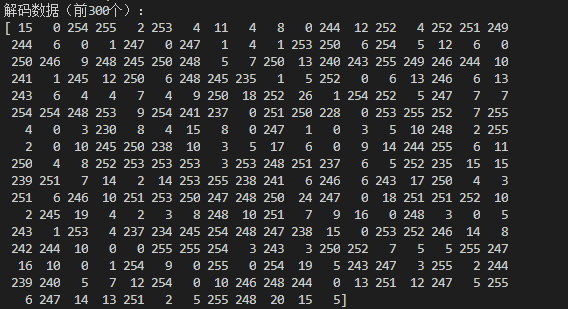
生成数据（前300个）：



编码过程（部分）：



解码数据（前300个）：

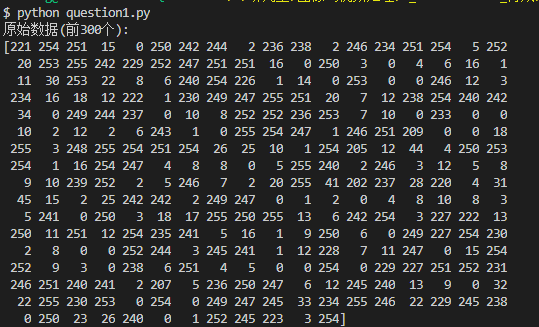


熵、编码平均码长及编码效率：

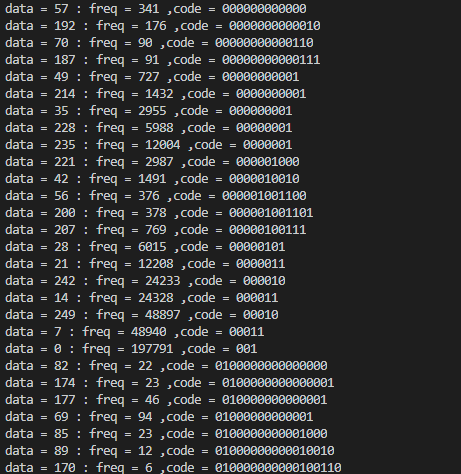


（3）拉普拉斯分布

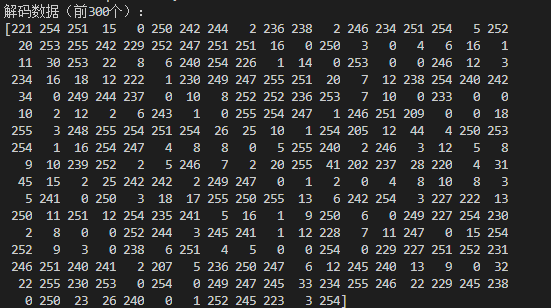
生成数据（前300个）：



编码过程：



解码数据（前300个）：



熵、编码平均码长及编码效率：



问题2：

选取 3-幅典型的彩色图像， 将像素的 R、 G、 B 值作为独立符号，进行哈夫曼编码和解码，计算熵和符号平均码长。

具体过程如下：

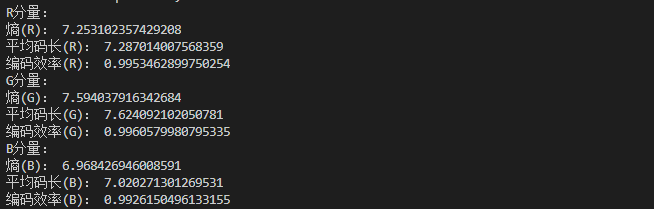
运行：question2.py

（1）测试图片1：lena-512.png

原图和解压解码后图像：



熵、编码平均码长及编码效率：

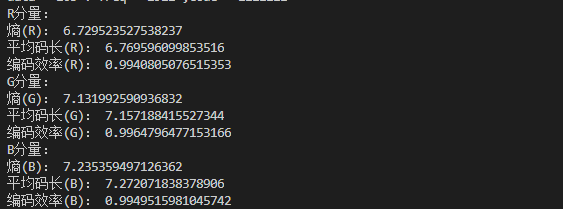


（2）测试图片：cat-512.png

原图和解压解码后图像：



熵、编码平均码长及编码效率：

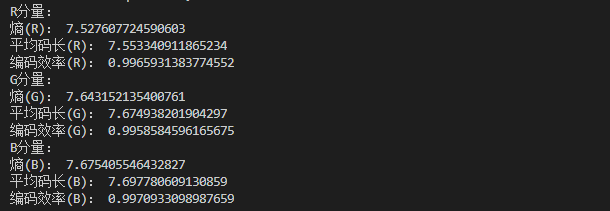


（3）测试图片：Sandhill-Crane-512.png

原图和解压解码后图像：



熵、编码平均码长及编码效率：



问题3：

选取 3幅典型的彩色图像，将像素的 R、 G、 B 值作为联合符号，进行哈夫曼编码和解码，计算熵和符号的平均码长。

具体过程如下：

运行：question3.py

（1）测试图片：lena-512.png

原图和解压解码后图像：

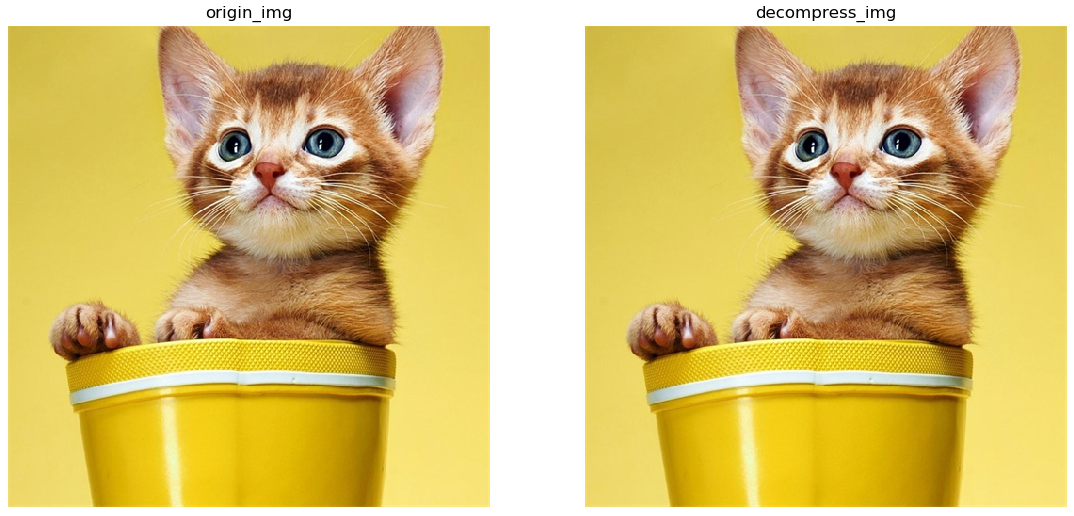


熵、编码平均码长及编码效率：



（2）测试图片：cat-512.png

原图和解压解码后图像：



熵、编码平均码长及编码效率：

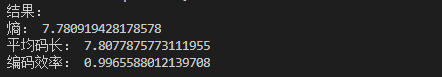


（3）测试图片：Sandhill-Crane-512.png

原图和解压解码后图像：



熵、编码平均码长及编码效率：



最后，通过比较，（2）的编码效率更高。