1a 哈夫曼编码需要有关信息源的先验统计信息，而这样的信息通常很难获得。在多媒体应用中表现得尤其突出，数据在到达之前是未知的，而且符号表的传输本身也是很大的开销。而自适应的哈夫曼编码则可以解决这个问题。在这种算法中，统计数字是随着数据流的到达而动态地收集和更新的。概率不再是基于先验知识而是基于到目前为止收到的数据。

1b1 bacc

推导过程：

1.接收到 0 无操作

2.接收到 1 得到 01==b，执行下图中第一步的变换

3.接收到 0 无操作

4.接收到 1 得到 01==a，执行下图中第二步的变换

5.接收到 0 无操作

6.接收到 0 无操作，判定输入的是一个新的字符

7.接收到 1 无操作

8.接收到 0 得到新的字符为 c，并执行第三步的变换

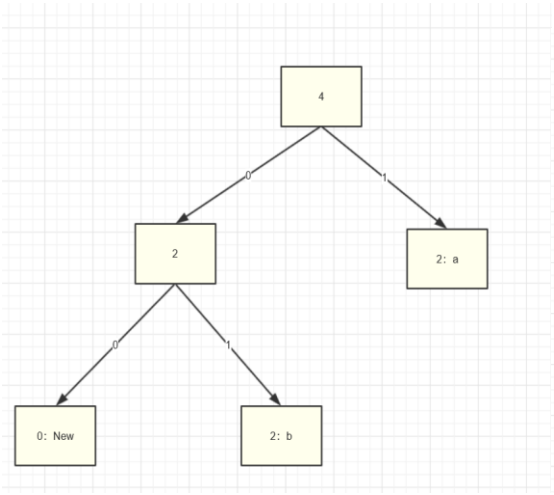
9.接收到 1 无操作

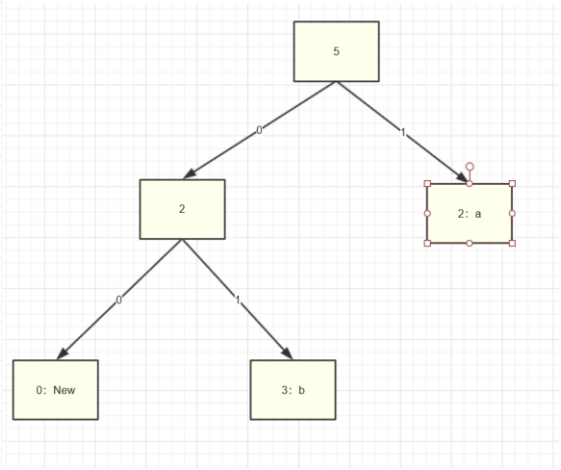
10.接收到 0 无操作

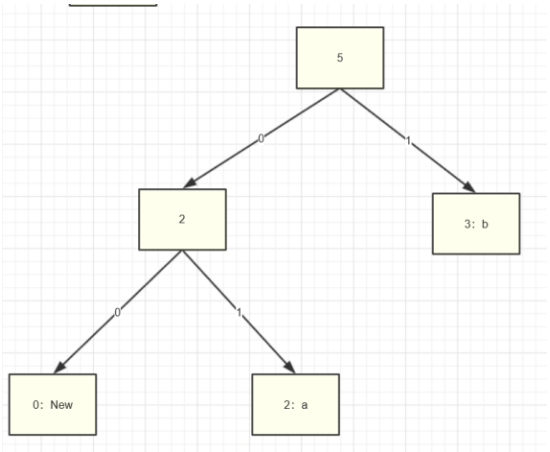
11.接收到 1 得到 101==c，执行第四步的变换

1b2 如图

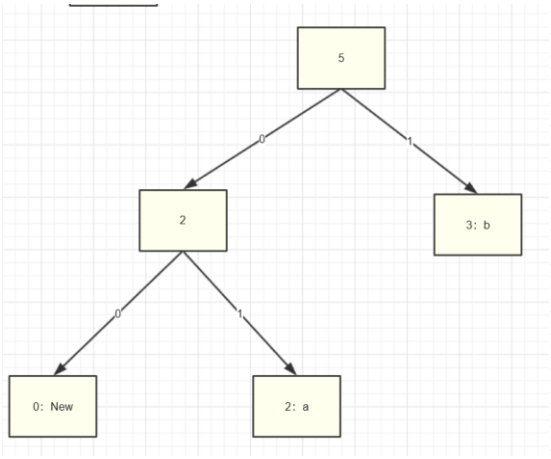
第一步，接收到 b

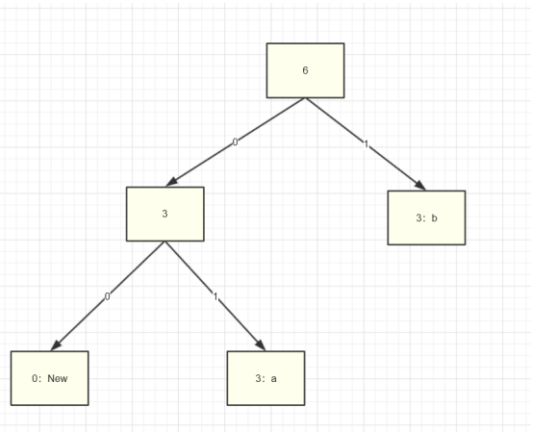


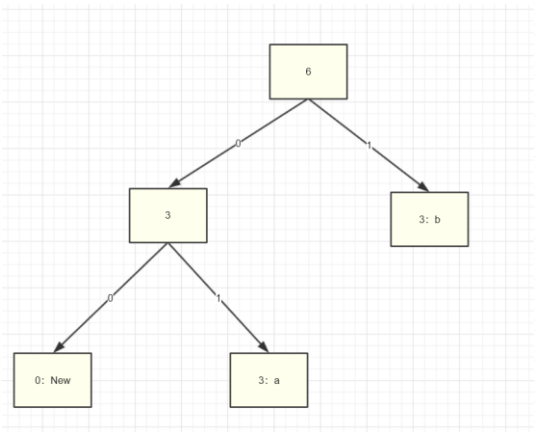



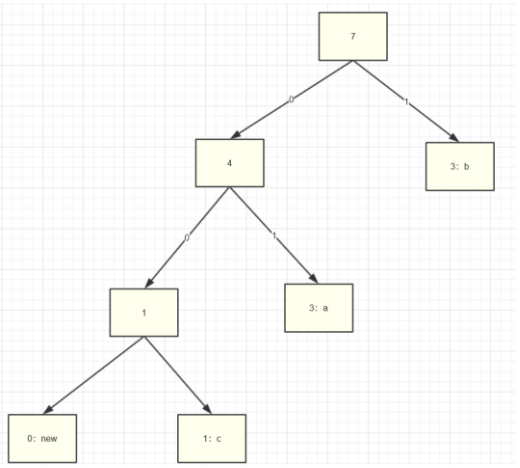

第二步，接收到 a

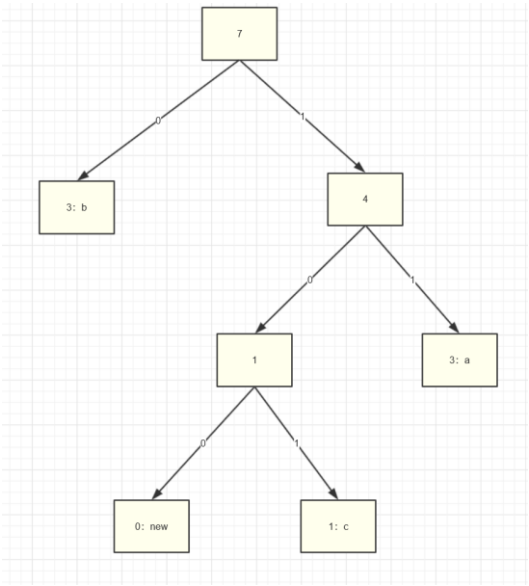


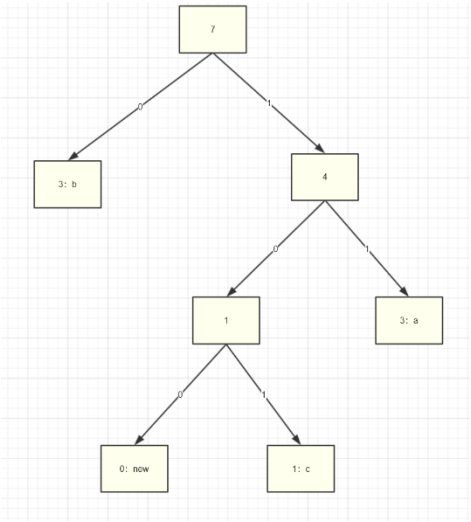

第三步，接收到 c

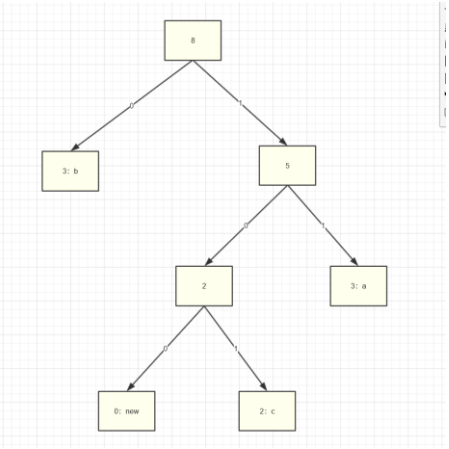



第四步，接收到 c




Part2

## 理论分析：对于色彩丰富的动物图片进行压缩应该选用 JPEG，而对卡通图片进行压缩应当选用 GIF。由于 GIF 选用了调色板机制，最多只能存储 256 种颜色，所以不太擅长处理照片，容易出现大量色块。

GIF 加载时可以支持模糊加载（通过隔行载入），使图像更快的出现在屏幕上，而且支持动画，可以把多张图片合为一张成为帧动画。

Jpeg 可以设置压缩系数来调整压缩的质量，其压缩率更加优秀。JPEG 采用了二次采样和 DCT 变换来丢弃高频信息，并通过熵编码的方法来压缩图像数据，所以 Jpeg 的压缩能力可以达到非常高。

## 代码实现部分：代码见 JpegSimulator.m 和 testhuffman.m

根据题目要求，我实现了

1. 颜色转换

   这一个模块把原图的色彩空间从 RGB 空间转移到了 YCrCb 上，此处使用了内建的库函数，如果手动实现的话可以使用下面的公式

$$Y' = 0.299*R' + 0.587*G' + 0.114*B'$$

$$U' = -0.147*R' - 0.289*G' + 0.436*B' = 0.492*(B' - Y')$$

$$V' = 0.615*R' - 0.515*G' - 0.100*B' = 0.877*(R' - Y')$$

2. 二次采样

   在二次采样之前，必须对原图进行格式化，使得行数和列数都能被 16 整除。格式化可以补 0，此处采用的则是全部补边缘列。然后按照 4:2:0 的比例进行采样。在这一步骤种，信息出现了损失。4:2:0 指的是它表示奇数行中，每连续四个点， 采样两个 CrCb，

在紧邻的第二行中不采样 CrCb，而并非是 Cb 采样为 0。

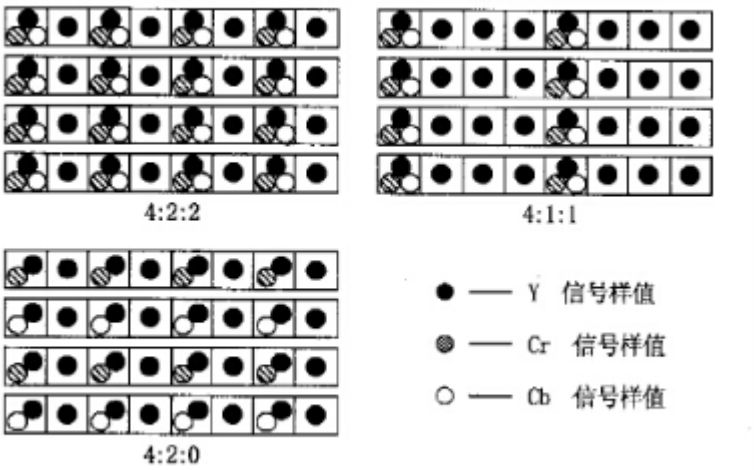
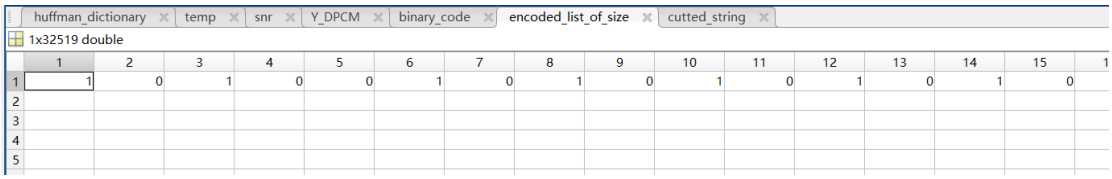


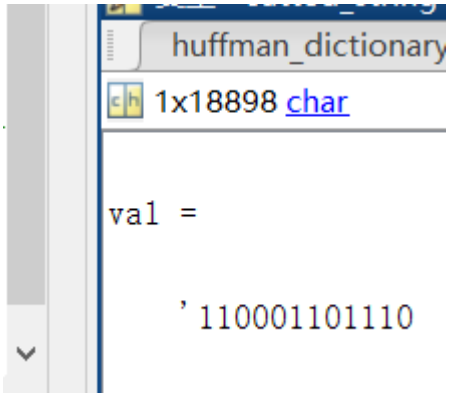

图1 3种取样格式的比较

3. DCT 变换　见 Dct_Quantize.m
   每个图像被划分成了 8*8 的块，公

$$F(u,v)=c(u)c(v)\sum_{i=0}^{N-1}\sum_{i=0}^{N-1}f(i,j)\cos\left[\frac{(i+0.5)\pi}{N}u\right]\cos\left[\frac{(i+0.5)\pi}{N}v\right]$$

   如此，将高频信息和低频信息分离开来。
4. 量化　见 Dct_Quantize.m
   将 DCT 得到的每个频率组成的 DCT 系数矩阵除以量化矩阵，而后取整。如此，可以将能量集中于低频区域，从而尽可能不损失视觉感觉地减少信息量。量化系数矩阵手动给出，还要再乘上一个质量因子（用来调控图像压缩质量）。接着把所有的量化过后的 DCT 系数-128。这样做的目的是方便进行游长编码（因为游长编码的压缩通过幅度来进行压缩，这个工序可以使得使用反码可以表示相反数）
5. DPCM 编码　见 DPCM_Encoding.m
   根据课本，这里的 DPCM 并非有损压缩，而是无损压缩。DPCM 的原理是提取每个 8*8 的块的第一个数据，然后通过与上一个系数的差来表示本数据。这样一来，数据将由第一个数和接下来的分布在 0 附近的差值组成。由于集中分布在 0 附近，可以取得更好的熵编码效果
6. 游长编码　见 Run_Length_Encoding.m
   这一部分是无损压缩中减少数据量最大的一部分。原理是经过量化过的块将体现出能量集中于左上角的特性。我们先进行一次 Z 扫描（Z 扫描通过打表得到），把二维数组改编成一维数组。一般来讲，这个一维数组的波动呈现出前端大后端小的特点，而且越往后 0 出现的越多。接下来把数组改写成{（本非 0 数之前的 0 的个数，本非 0 数）}这样的集合。注意到第一个数已经存在于 DPCM 编码中，所以这个集合可以从第二个开始。如果某个块全为 0，那么用（0，0）表示它。
7. 熵编码　见 testhuffman
   这一部分我只完成了一半，对 DPCM 的编码的结果是一个幅度的位宽的哈夫曼编码，和一个剪切过的幅度值（每个幅度值只保留非 0 的位宽部分长度）的串。




8. 游长编码解码 见 DPCM_RLE_Decoding
   游长编码的逆过程。通过恢复 0 的个数把每个串补全 64 个元素的块（第一个元素是 DPCM 解码得到的）
9. DPCM 编码解码 DPCM_RLE_Decoding
   DPCM 编码的逆过程，比较简单。
10. DCT 逆变换

DCT 过程的逆过程

对着公式打，一个字母都不能少

11. 恢复原图

恢复到二次采样之前的数据。

## 结果对比：可以看到，GIF 图片色彩过渡十分生硬，而且有明显的类似喷墨打印机的小点。，在视图中部的草甸上，可以看到明显的色块

和预测不同的是，jpeg 在两张图的测试中，无论在图像质量、还是在压缩率上都明显优于 gif，并不存在 jpeg 只在照片情况下优于 gif。


gif 版本


Jpeg 版本


Gif 的卡通图


Jpeg 的卡通图

压缩率：计算压缩率的时候，不应该以原图作为基准，而应该以原图生成的 bmp 图像作为原始的数据量，因为原图是 jpg 格式，本身已经压缩过了。

1. Animal 信噪比通过 RGB 比较

JPEG（Matlab 库函数）：

压缩率：230703Byte /2095158Byte=0.1101

```
val(:,:,1) =

   38.7904


val(:,:,2) =

   39.3579


val(:,:,3) =

   38.1334
```

信噪比：

GIF：（来自网站转换器）

压缩率：612336Byte/2095158Byte=0.2923

```
val(:,:,1) =

   19.0967


val(:,:,2) =

   21.8190


val(:,:,3) =

   16.6108
```

信噪比：

JPEG（个人实现版）：

```
val(:,:,1) =

   36.5657


val(:,:,2) =

   37.3111


val(:,:,3) =

   35.4376
```

信噪比：

2. Cartoon 信噪比通过 RGB 通道比较

JPEG（Matlab 库函数）:

压缩率：119251Byte /2145054Byte=0.0556

信噪比:

```
val(:,:,1) =

   47.2195


val(:,:,2) =

   49.7554


val(:,:,3) =

   46.4501
```

GIF:（来自网站转换器）

压缩率：408312Byte/2145054Byte=0.1904

```
val(:,:,1) =

   14.8227


val(:,:,2) =

   16.3820


val(:,:,3) =

   22.2730
```

信噪比:

JPEG（个人实现版）:

```
val(:,:,1) =

   34.1437


val(:,:,2) =

   36.0315


val(:,:,3) =

   33.8300
```

信噪比:

代码见附件：运行说明：请运行 JpegSimulator，然后才可以运行 testhuffman