Muti-media assignment 2

|  |  |
| --- | --- |
| 课程 | 多媒体程序设计 |
| 姓名 | 许若敏 |
| 学号 | 15336212 |
| 日期 | 2018/11/20 |

# 第一题：

a)

解：

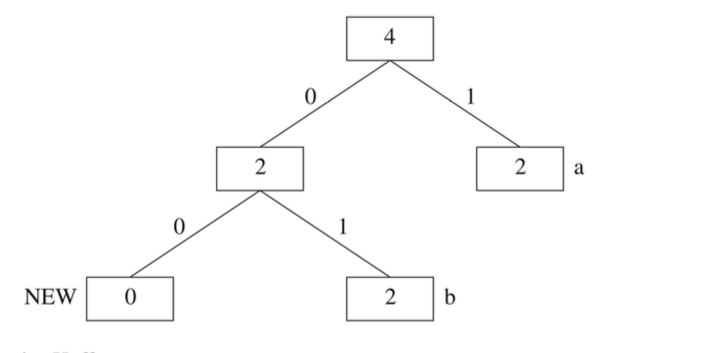
1. Adaptive Huffman Coding不需要信息源先验的统计信息。当编码多媒体信息源时，下一个符号常常是未知的，无法使用普通的哈夫曼编码，自适应哈夫曼编码这时就很占优势。

2. Adaptive Huffman Coding在编码后的传输过程中无需传输字符编码表。

b)

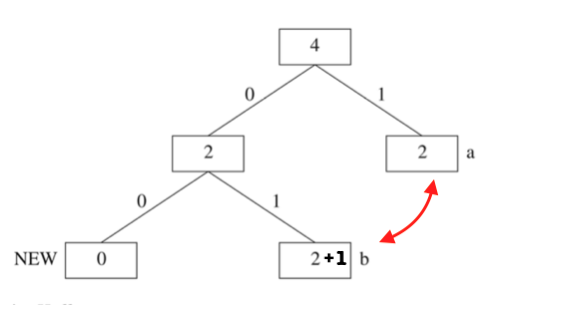
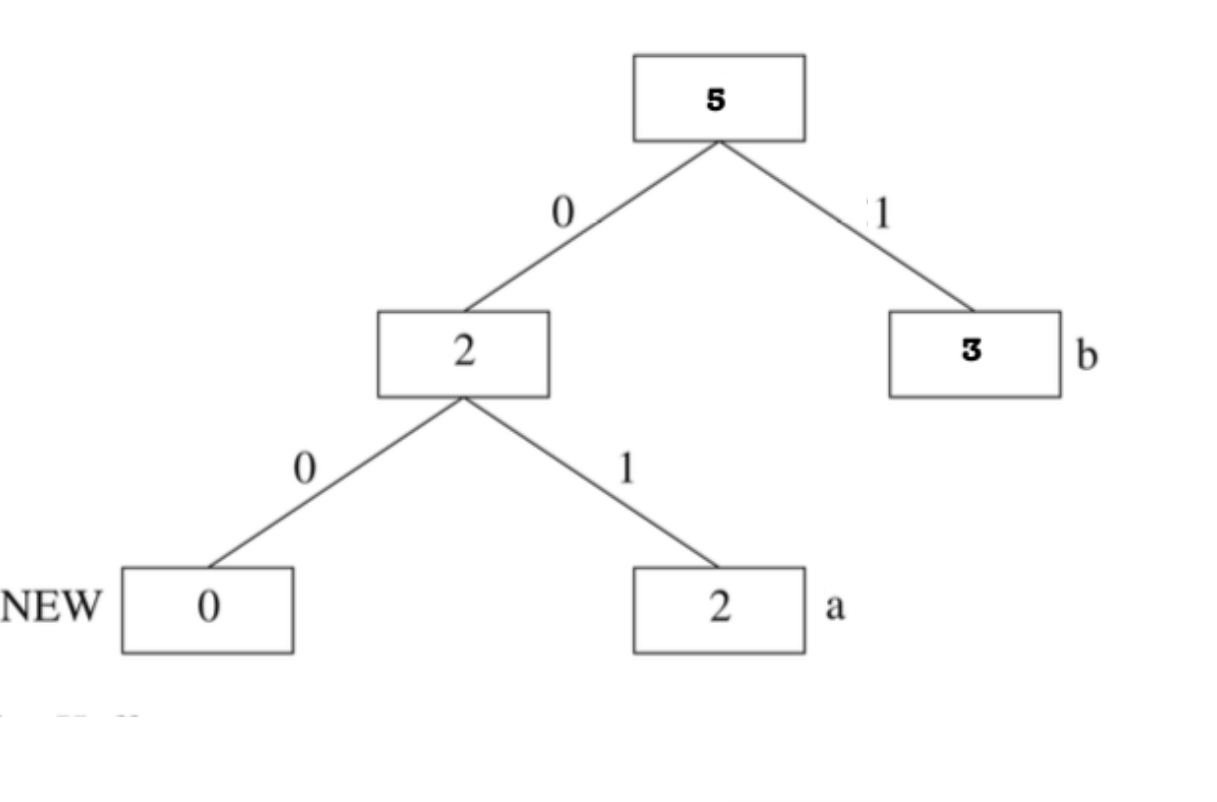
解：

i, ii

这是decoder原始的哈夫曼编码树

**step1**:

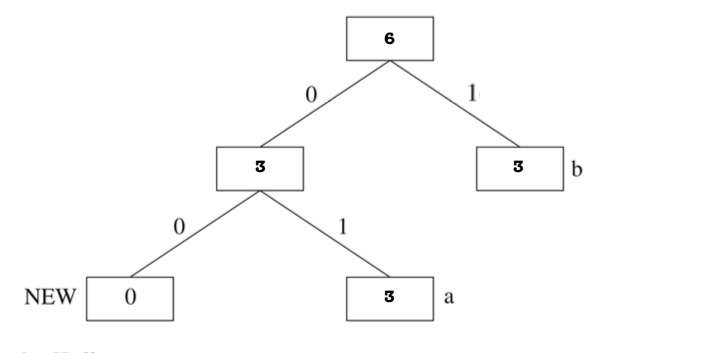
输入流：01010010101，

先解码得到第一个字符：01 -> b，c0 = b，更新哈夫曼树b的位置+1，则a和b交换位置得到：

**step2**:

输入流：010010101

解码：01 -> a，c1 = a，更新哈夫曼树a的位置+1：



**step3**:

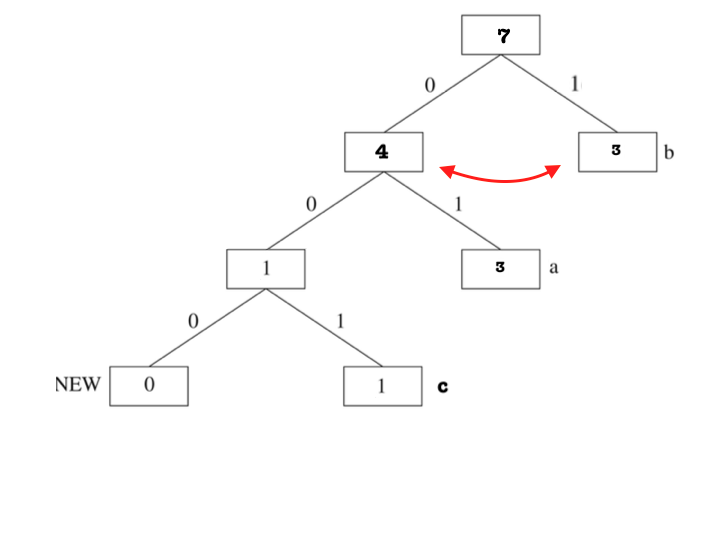
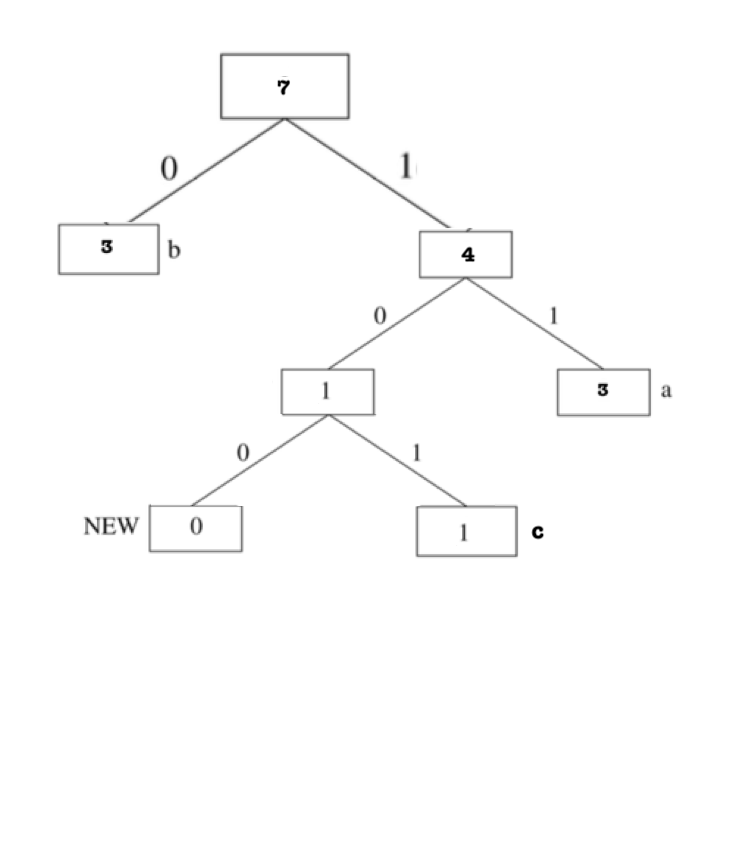
输入流：0010101

解码：00 -> NEW

**step4**:

输入流：10101

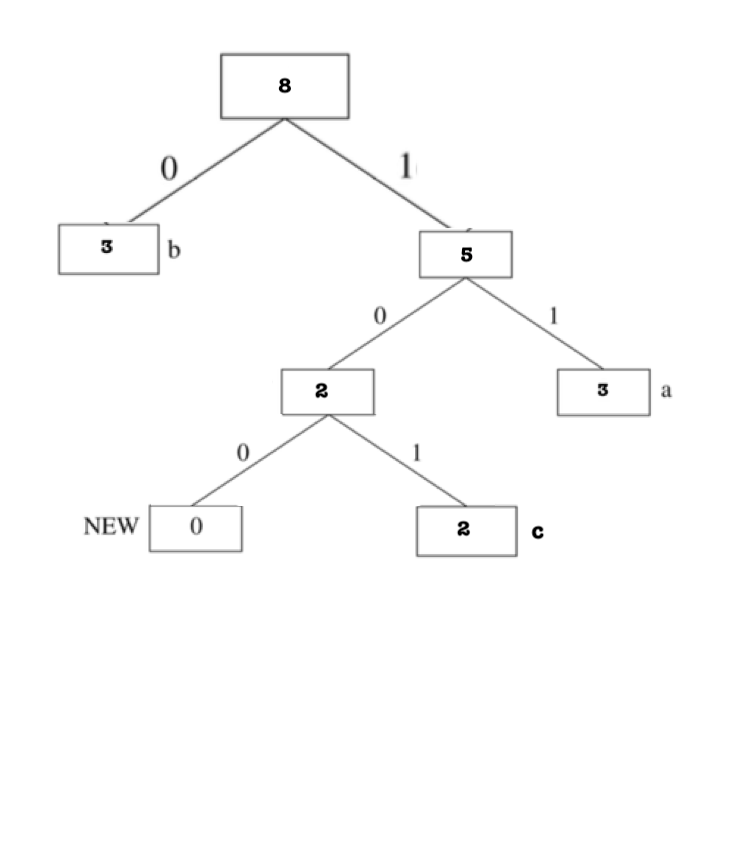
解码：由于上一个step解码是NEW，所以说明有新的符号进入，要根据初始定义的编码表进行翻译，即 10 -> c，c2 = c。更新哈夫曼树加入c的节点，交换第二层节点：



**step4**:

输入流：101

解码：101 -> c，c3 = c。更新哈夫曼树：



最终可解码接收序列：c0c1c2c3 = bacc

# 第二题：

为了比较Jpeg标准压缩效果，这里我使用c++编程实现了基本的Jpeg压缩标准，包括：颜色转换+色度二次采样, 二维 DCT 变换，量化， DPCM 和 游长编码， 熵编码等。其中熵编码采用 <https://www.w3.org/Graphics/JPEG/itu-t81.pdf> 中的标准哈希表K:

主要Jpeg压缩编码核心实现于mjpeg.cpp文件中。

最后压缩结果保存于output文件夹中（output/animal\_out.jpeg, output/animal\_cartoon\_out.jpeg）。其中.dat后缀文件是压缩后的二进制文件。

结合python脚本计算GIF图像的失真率，我进行实验得出如下结果（下面是程序输出图）

**c++**实现结果（输出**gif**，**jpeg**压缩率，**jpeg**失真率）：

**python**脚本计算**gif**失真率结果：

GIF vs JPEG 压缩效果比较

1. animal图片

|  |  |  |  |
| --- | --- | --- | --- |
|  | 原图 | GIF | JPEG |
| 图像细节 |  |  |  |
| 失真率（MSE） | - | 48.305 | 98.577 |
| 失真率（SNR) | - | 3.319 | 2.638 |
| 压缩率 | - | 0.276 | 0.123 |

2. animal cartoon 图片

|  |  |  |  |
| --- | --- | --- | --- |
|  | 原图 | GIF | JPEG |
| 图像细节 |  |  |  |
| 失真率（MSE） | - | 56.707 | 98.392 |
| 失真率（SNR) | - | 2.638 | 24.825 |
| 压缩率 | - | 0.133 | 0.120 |

总结，GIF和JPEG对卡通图片的压缩效果几乎一致，但是JPEG的失真率比GIF高。自然图片上使用JPEG压缩效果明显比GIF好，JPEG失真率比较GIF略高，但是肉眼上看图片细节差别不大。

所以对卡通图片进行压缩时，建议使用GIF格式，对自然图片压缩时建议使用JPEG格式。