## 一：Arithmetic Coding简介

早在1948年，香农就提出将信源符号依其出现的概率降序排序，用符号序列累计概率的二进值作为对芯源的编码，并从理论上论证了它的优越性。1960年， Peter Elias发现无需排序，只要编、解码端使用相同的符号顺序即可，提出了算术编码的概念。Elias没有公布他的发现，因为他知道算术编码在数学上虽然成 立，但不可能在实际中实现。1976年，R. Pasco和J. Rissanen分别用定长的寄存器实现了有限精度的算术编码。1979年Rissanen和G. G. Langdon一起将算术编码系统化，并于1981年实现了二进制编码。1987年Witten等人发表了一个实用的算术编码程序，即CACM87（后用 于ITU-T的H.263视频压缩标准）。同期，IBM公司发表了著名的Q-编码器（后用于JPEG和JBIG图像压缩标准）。从此，算术编码迅速得到了 广泛的注意。算术编码是图像压缩的主要算法之一，是一种无损数据压缩方法，也是一种熵编码的方法，和其他熵编码方法不同的地方在于，其他的熵编码方法通常把输入的消息分割为符号，然后对每个符号进行编码，而算术编码是直接把整个输入的消息编码为一个数，一个满足(0.0≤n<1.0)的小数n。

## 二：工作原理：

在给定符号集和符号概率的情况下，算术编码可以给出接近最优的编码结果。使用算术编码的压缩算法通常先要对输入符号的概率进行估计，然后再编码。这个估计越准，编码结果就越接近最优的结果。

算术编码的基本原理是将编码的消息表示成实数0和1之间的一个间隔(Interval),消息越长，编码表示它的间隔就越小，表示这一间隔所需要的二进制位数就越多。算术编码用到两个基本的参数：符号的概率和它的编码间隔。信源符号的概率决定压缩编码的效率，也决定编码过程中信源符号的间隔，而这些间隔包含在0到1之间，编码过程中的间隔决定了符号压缩后的输出。

给定事件序列的算术编码的步骤如下：

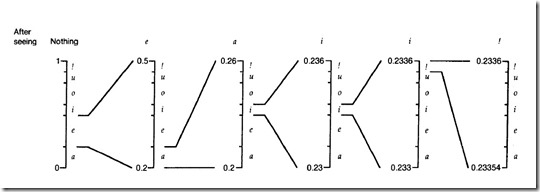
1. 编码器在开始时将“当前间隔”[L,H)设置为[0,1)
2. 对每一事件，编码器按照步骤(a)和(b)进行处理
3. 编码器将“当前间隔”分为子间隔，每个事件一个。
4. 一个子间隔的大小与下一个将出现的事件的概率成比例，编码器选择子间隔对应于 下一个确切发生的事件相对应，并使它称为新的“当前间隔”.
5. 最后输出的“当前子间隔”的下边界就是该给定事件序列的算术编码。

设low和high分别表示当前间隔的下边界和上边界，coderange为编码间隔的长度，lowrange(symbol)和highrange(symbol)分别代表为了事件symbol分配的初始间隔下边界和上边界。

算术编码是用符号的概率和它的编码间隔两俩个基本参数来描述的（见下文教程）。算术编码可以是静态的或是自适应的。在静态算术编码中，信源符号的概率是固定的。在自适应算术编码中，信源符号的概率根据编码时符号出现的频繁程度动态地进行修改。

在编码期间估算信源符号概率的过程叫建模。需要开发动态算术编码的原因，是因为事先知道精确的信源符号概率是很难的，而且是不切实际的。动态建模是确定编码器压缩效率的关键。

对于算术编码的译码过程就是一系列的比较过程，每一步比较与的大小，为前面已经译出的字符串序列似的分布函数，即对应区间的下界，是此区间内下一个输入符号所占区间的宽度，对于K元信源，译每一个码元最多需要K-1次比较以确定是信源编码符号集中的哪一个。



伪代码如下：

set Low to 0

set High to 1

while there are input symbols do

take a symbol

CodeRange = High – Low

High = Low + CodeRange \*HighRange(symbol)

Low = Low + CodeRange \* LowRange(symbol)

end of while

output Low

算术码解码过程用伪代码描述如下：

get encoded number

do

find symbol whose range straddles the encoded number

output the symbol

range = symbo.LowValue – symbol.HighValue

substracti symbol.LowValue from encoded number

divide encoded number by range

until no more symbols

在算术编码中有几个问题需要注意：

1.由于实际的计算机的精度不可能无限长，一个明显的问题是运算中出现溢出，但多数机器都有16、32或者64位的精度，因此这个问题可使用比例缩放方法解决。

2.算术编码器对整个消息只产生一个码字，这个码字是在间隔[0，1]中的一个实数，因此译码器在接受到表示这个实数的所有位之前不能进行译码。

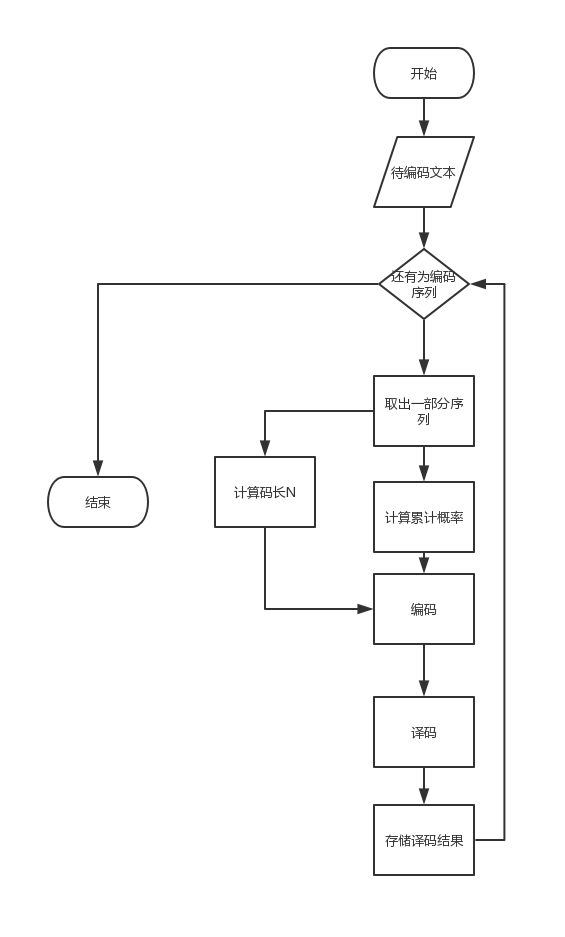
算术编码也是一种对错误很敏感的编码方法，如果有一位发生错误就会导致整个消息译错。

3.算术编码可以是静态的或者自适应的。在静态算术编码中，信源符号的概率是固定的。在自适应算术编码中，信源符号的概率根据编码时符号出现的频繁程度动态地 进行修改，在编码期间估算信源符号概率的过程叫做建模。需要开发动态算术编码的原因是因为事先知道精确的信源概率是很难的，而且是不切实际的。当压缩消息 时，我们不能期待一个算术编码器获得最大的效率，所能做的最有效的方法是在编码过程中估算概率。因此动态建模就成为确定编码器压缩效率的关键。

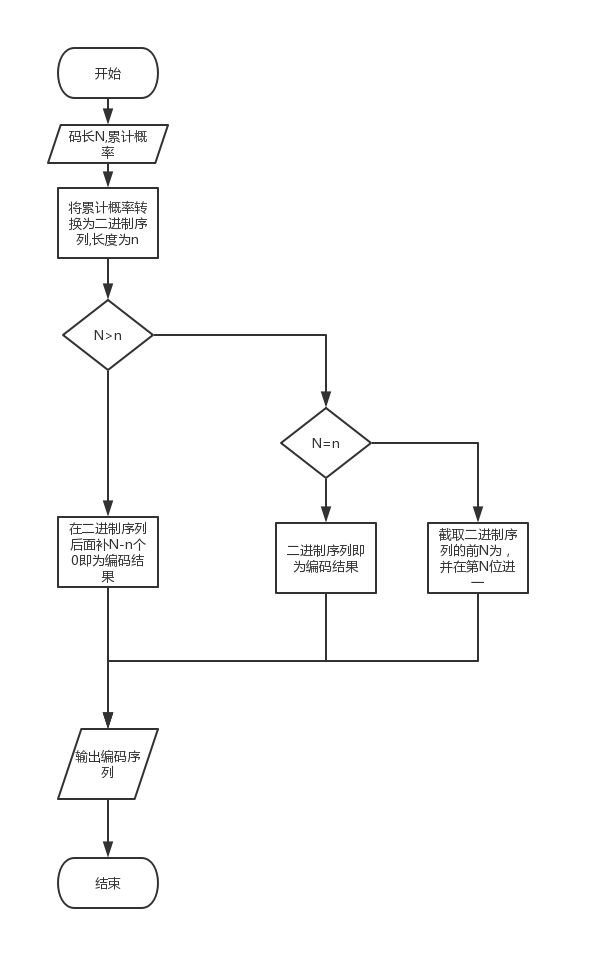
4.此外，在算术编码的使用中还存在版权问题。JPEG标准说明的算术编码的一些变体方案属于IBM， AT&T和Mitsubishi拥有的专利。要合法地使用JPEG算术编码必须得到这些公司的许可。

## 三．算法流程图：

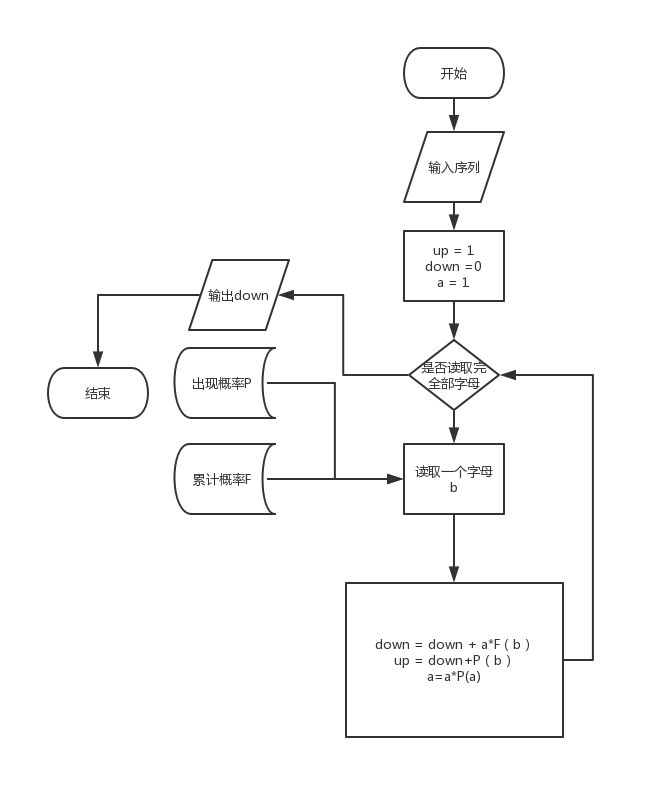
编解码总流程图：



编码流程图：



解码流程图：



## 四：算法实现（python):

*# -\*-coding:utf-8-\*-***import** time**import** numpy **as** np**import** pprint**import** matplotlib.pyplot **as** plt**import** random

alpha\_dict **=** {

'a': 0.0575,

'b': 0.0128,

'c': 0.0263,

'd': 0.0285,

'e': 0.0913,

'f': 0.0173,

'g': 0.0133,

'h': 0.0313,

'i': 0.0599,

'j': 0.0006,

'k': 0.0084,

'l': 0.0335,

'm': 0.0235,

'n': 0.0596,

'o': 0.0689,

'p': 0.0192,

'q': 0.0008,

'r': 0.0508,

's': 0.0567,

't': 0.0706,

'u': 0.0334,

'v': 0.0069,

'w': 0.0119,

'x': 0.0073,

'y': 0.0164,

'z': 0.0007,

' ': 0.1928,}

color **=** ['#dc2624', '#2b4750', '#45a0a2',

'#e87a59', '#7dcaa9', '#649E7D',

'#dc8018', '#C89F91', '#6c6d6c',

'#4f6268', '#c7cccf'

]

*# 计算信源熵***def** **calc\_entropy**(alpha\_dict):

entropy **=** 0

gailv\_jihe **=** list(alpha\_dict**.**values())

**for** i **in** range(len(alpha\_dict)):

entropy **=** entropy**+**gailv\_jihe[i]**\***np**.**log(gailv\_jihe[i])

**return** **-**entropy

*# 计算累乘概率***def** **mul\_pos**(input\_str**=**''):

pre\_possibility **=** 1

input\_list **=** list(input\_str)

**for** i **in** input\_list:

pre\_possibility **=** pre\_possibility**\***alpha\_dict**.**get(i)

**return** pre\_possibility

*# 计算码长***def** **calc\_machang**(leic\_possibility):

ma\_length **=** np**.**ceil(**-**np**.**log(leic\_possibility)**/**np**.**log(2))

**return** ma\_length

*# 十进制小数转换为二进制小数***def** **dec2bin**(x):

x **-=** int(x)

bins **=** []

**while** x:

x **\*=** 2

bins**.**append(1 **if** x **>=** 1. **else** 0)

x **-=** int(x)

**return** bins

*# 二进制小数转换为十进制小数***def** **bin2dec**(b):

d **=** 0

**for** i, x **in** enumerate(b):

d **+=** 2**\*\***(**-**i**-**1)**\***x

**return** d

*# 二进制小数进位***def** **bin\_jinwei**(input\_bin**=**[]):

**for** i **in** range(len(input\_bin)):

**if** input\_bin[len(input\_bin) **-** 1 **-** i] **==** 0:

input\_bin[len(input\_bin) **-** 1 **-** i] **=** 1

**break**

**else**:

input\_bin[len(input\_bin) **-** i **-** 1] **=** 0

**return** input\_bin

**def** **accu\_pos**(para\_dict):

accu\_pos\_dict **=** {}

*# print(len(alpha\_dict))*

**for** i **in** range(len(para\_dict)):

pre\_pos **=** 0

pre\_alpha **=** list(para\_dict**.**keys())[i]

**if** i **==** 0:

pre\_pos **=** 0

**else**:

**for** j **in** range(i):

pre\_pos **=** list(para\_dict**.**values())[j]**+**pre\_pos

accu\_pos\_dict[pre\_alpha] **=** pre\_pos

**return** accu\_pos\_dict

*# 计算待编码序列的累计概率***def** **calc\_xulie\_pos**(xulie, accu\_pos\_dict**=**{}):

xulie **=** list(xulie)

alpha\_list **=** list(alpha\_dict**.**keys())

pos\_down **=** 0

pos\_up **=** 0

accu **=** 1

plt**.**ion()

xianshi **=** ''

**for** i **in** range(len(xulie)):

xianshi **=** xianshi**+**xulie[i]

pos\_down **=** pos\_down **+** accu**\***accu\_dict**.**get(xulie[i])

pos\_up **=** pos\_down **+** alpha\_dict**.**get(xulie[i])**\***accu

accu **=** accu **\*** alpha\_dict**.**get(xulie[i])

leiji\_pos **=** (pos\_down **+** pos\_up) **/** 2

leiji\_pos **=** pos\_down

**return** leiji\_pos

*# # 解码过程***def** **decode\_proceing**(rec\_pos, accu\_pos\_dict, machang**=**0):

pre\_pos **=** rec\_pos

alpha\_list **=** list(alpha\_dict**.**keys())

*# print(alpha\_list)*

jiema\_result **=** ''

**for** i **in** range(machang):

**for** j **in** range(1, len(alpha\_list)):

leijigailv **=** accu\_pos\_dict**.**get(alpha\_list[j])

**if** pre\_pos **>** 0.8074:

jiema\_result **=** jiema\_result**+**' '

pre\_pos **=** (pre\_pos **-** accu\_dict**.**get(' ')) **/** alpha\_dict**.**get(' ')

**break**

**elif** pre\_pos **<** leijigailv:

jiema\_result **=** jiema\_result**+**alpha\_list[j**-**1]

pre\_pos **=** (pre\_pos**-**accu\_dict**.**get(alpha\_list[j**-**1])) **/** alpha\_dict**.**get(alpha\_list[j**-**1])

**break**

**return** jiema\_result

*# 将累计概率由十进制小数转换为二进制小数***def** **pos\_change**(leijigailv):

bin\_pos **=** dec2bin(leijigailv)

**return** bin\_pos

*# 从小数后截取N位，如果后还有尾数则进位，如果不足则补零***def** **bianma**(bin\_pos, pre\_length**=**30):

pre\_result **=** []

**if** len(bin\_pos) **==** pre\_length:

pre\_result **=** bin\_pos

**elif** len(bin\_pos) **>=** pre\_length:

''' 如果N小于二进制概率位数，则将二进制小数的第N位进1，然后截取N位 '''

pre\_result **=** bin\_pos[:pre\_length]

pre\_result **=** bin\_jinwei(pre\_result)

**else**:

''' 如果N大于二进制概率位数，则将二进制小数位后面补零至N位，然后截取N位 '''

**for** i **in** range(pre\_length**-**len(bin\_pos)):

bin\_pos**.**append(0)

pre\_result **=** bin\_pos

**return** pre\_result

**if** \_\_name\_\_ **==** "\_\_main\_\_":

resouse\_entropy **=** calc\_entropy(alpha\_dict)

print("信源熵为：{}"**.**format(resouse\_entropy))

accu\_dict **=** accu\_pos(alpha\_dict)

pprint**.**pprint(accu\_dict)

f\_test **=** open('test1.txt', 'r')

text **=** f\_test**.**read()

a **=** ''

total **=** len(text)**\***8

total\_ **=** 0

count **=** 0

count\_wrong **=** 0

all\_unit **=** 0

start\_time **=** time**.**time()

**with** open('result1.txt', 'a') **as** f:

**for** i **in** range(len(text)):

alp **=** text[i]**.**lower()

**if** alp **in** alpha\_dict**.**keys():

a **+=** alp

**if** alp **==** ' ':

all\_unit **+=** 1

print(a)

r **=** calc\_xulie\_pos(a, accu\_pos\_dict**=**accu\_dict)

lei\_possibility **=** mul\_pos(a)

N **=** calc\_machang(leic\_possibility**=**lei\_possibility)

total\_ **=** total\_**+**N

print(N)

print(r)

bin\_pos **=** pos\_change(r)

bin\_pos **=** bianma(bin\_pos**=**bin\_pos, pre\_length**=**int(N))

print(bin\_pos)

back\_pos **=** bin2dec(bin\_pos)

print(back\_pos)

result **=** decode\_proceing(back\_pos, accu\_pos\_dict**=**accu\_dict, machang**=**len(a))

print(result)

count **+=** 1

f**.**write(result)

original\_len **=** len(a)**\***8

print("压缩比为： {:.2f}%"**.**format(**-**N**/**original\_len**\***100))

**if** a **!=** result:

count\_wrong **+=** 1

**if** count **==** 12:

f**.**write('\n')

count **=** 0

a **=** ''

**else**:

a **+=** ' '

f\_test**.**close()

print("译码错误占比为： {:.2f}%"**.**format(count\_wrong**/**all\_unit**\***100))

stop\_time **=** time**.**time()

run\_time **=** stop\_time **-** start\_time

print("运行时间为{}"**.**format(run\_time))

print("总压缩比为：{:.2f}%"**.**format(total\_**/**total**\***100))*# 测试序列 baacdaba*

## 五：测试

### 5.1测试文档：(text.txt)

Financial regulators in Britain have imposed a rather unusual rule on the bosses of big banks. Starting next year, any guaranteed bonus of top executives could be delayed 10 years if their banks are under investigation for wrongdoing. The main purpose of this “clawback” rule is to hold bankers accountable for harmful risk-taking and to restore public trust in financial institutions. Yet officials also hope for a much larger benefit: more long-term decision making, not only by banks but by all corporations, to build a stronger economy for future generations.

### 5.2 实验结果：

##### 5.2.1译码后的文档：(result.txt)

financial regulators in britain have imposed a rather unusual rule on the

bosses of big banks starting next year any guaranteed bonus of top

executives could be delayed years if their banks are under investigatin m

for wrongdoing the main purpose of this clawback yrule is to hold

bankers accountable for harmful risk takingsand to restore public trust in financial

institutiontabyet officials also hope for a much larger benefit more long term

decision making not only by banks but by all corporationtaato build

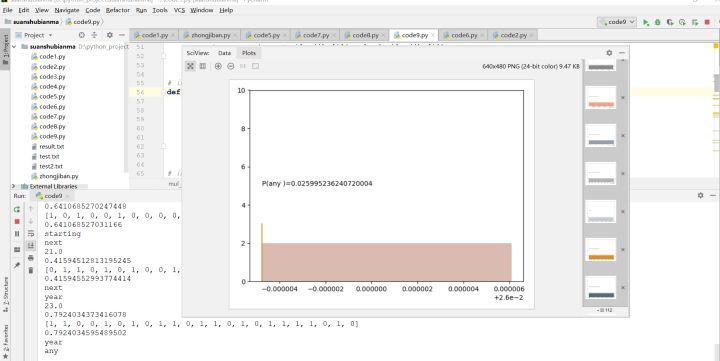
a stronger economy for future

##### 5.2.2结果分析：

1.由于所给信源符号集中只有小写字母及空格，所以在编码过程中我将大写转换为小写，将除空格外的符号全部转换为空格。

2.由于Python 的浮点数精度只有小数点后17位，所以每次编码的序列长度不能过长，当字符串序列长度过长时，序列区间越来越小，计算机不能精确识别，导致译码错误，由于我是以空格来作为字符串的结束标志，所以每次长度不一样，比如原文中的“investigation”错误的译成了“investigatin m”。

3.由于ASCII码在计算机中占7个二进制位，所以我们可以用编码后的二进制码长度和序列长度与7的乘积来计算编码效率。





## 六：总结：

算术编码是上课讲的很详细的方法，所以当老师说选择一个算法写报告时，我脑子里就只有它了，所以就选择了它，python是一门非常“自由”的语言，所以我就选择了用python来实现，编码采用的概率表，是再往上找了个编的，测试文档是随便找了段外文新闻报道。从运行结果来看，每个单词的压缩比达到35%+，运行时间还是可观的，此外译码错误占比基本在5%左右。基本实验结果图示等如上所示。