合肥工業大學

计算机与信息学院

《多媒体信息处理基础》实验报告 1

专　业　班　级　智能科学与技术 19-1 班

学生姓名及学号　武梓龙 2019212300

指　导　教　师　李佳　胡珍珍

2022 年 6 月 29 日

一、理论部分

1、简述 LZ78 和 LZW 算法的基本原理，并比较其异同

`LZ78 算法的基本原理：

在压缩时维护一个动态词典 Dictionary，其包括了历史字符串的 index 与内容；压缩情况分为三种：

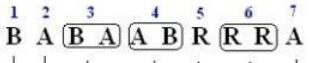
1、若当前字符 c 未出现在词典中，则编码为(0, c)；

2、若当前字符 c 出现在词典中，则与词典做最长匹配，然后编码为(prefixIndex,lastChar)，其中，prefixIndex 为最长匹配的前缀字符串，lastChar 为最长匹配后的第一个字符；

3、为对最后一个字符的特殊处理，编码为(prefixIndex,)。

下面给出例子。例子一，对于字符串“ABBCBCABABCAABCAAB”压缩编码过程如下：

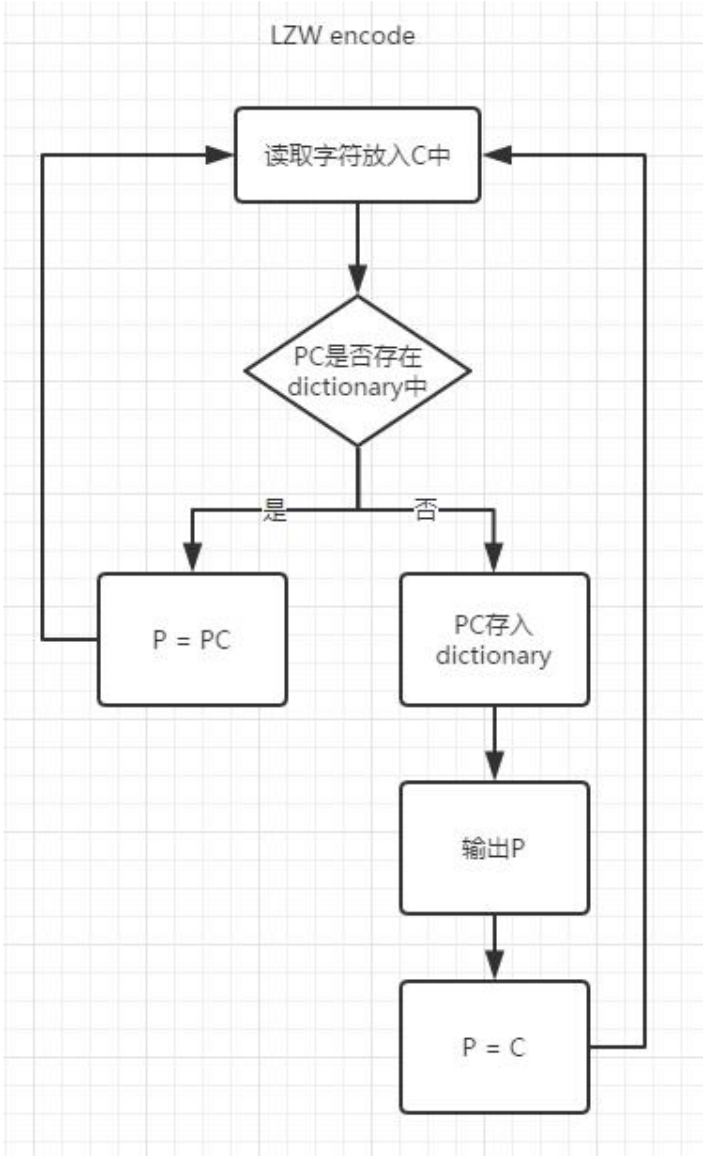
	Dictionary	
output	index	string
(0, A)	1	A
(0, B)	2	B
(2, C)	3	BC
(3, A)	4	BCA
(2, A)	5	BA
(4, A)	6	BCAA
(6, B)	7	BCAAB



	Dictionary	
output	index	string
(0, B)	1	B
(0, A)	2	A
(1, A)	3	BA
(2, B)	4	AB
(0, R)	5	R
(5, R)	6	RR
(2,)		

• **LZW 算法的基本原理：**

在正文流中词汇和短语很可能会重复出现。当出现一个重复时，重复的序列可以用一个短的编码来代替。压缩程序重复扫描这样的重复，同时生成编码来代替重复序列。随着时间过去，编码可以用来捕获新的序列。算法必须设计成压缩程序能够在编码和原始数据序列推导出当前的映射。



然后我们以字符串 abcbcabcabcd 为例，来模拟这个过程

首先我们先建立初始码表 a= 1，b=2，c=3，d=4

step	P	C	is PC in dic	output P	dictionary	description
1	NULL	a		初始化，不处理		
2	a	b	no	1	ab:5	ab不存在dictionary中，扩充
3	b	c	no	2	bc:6	
4	c	b	no	3	cb:7	
5	b	c	yes			bc在dictionary中，P = PC
6	bc	a	no	6	bca:8	
7	a	b	yes			
8	ab	c	no	5	abc:9	
9	c	a	no	3	ca:10	
10	a	b	yes			
11	ab	c	yes			
12	abc	d	no	9		
13	d	NULL		4		结束

LZ78 与 LZW 比较：

• LZW 只输出代表词典中的字符串的码字。这就意味在开始时词典不能是空的，它必须包含可能在字符流出现中的所有单个字符，即前缀根（Root）。

• 由于所有可能出现的单个字符都事先包含在词典中，每个编码步骤开始时都使用一字符前缀，因此至少可以在词典中找到长度为 1 的匹配串。

• LZ78 算法和 LZW 算法都通过预先扫描输入缓存中的数据并与其字典中的数据进行匹配来实现压缩。

• 对于大型的数据而言，LZ78 算法往往能得到较小的压缩率。作为其实用版本的 LZW 算法，它设计逻辑简单，有利于简单高效快速的实现。

• LZ78 算法和 LZW 算法的自适应速度比较慢

2、根据算法原理，实现两种编码算法

LZ78:

```
1.  import sys
2.  def encode(text):
3.      #传进字符串 abracadabra
4.      mydict = dict()
5.      i = 0
6.      index = 1
7.      lstNumbers = []
8.      lstLetters = []
9.      while i < len(text):
```

```
        stringToBeSaved = text[i]
        index = 0
        while stringToBeSaved in mydict:
            index = mydict[stringToBeSaved] # 找该字符在字典中的下标
            if (i == len(text) - 1):
                stringToBeSaved = " "
                break
            i = i + 1
            stringToBeSaved = stringToBeSaved + text[i]
            #print(stringToBeSaved)   ac   ad  ab  ra
        #print ("<{0}, {1}>".format(index, stringToBeSaved[len(stringToBeSaved) - 1]))
        lstNumbers.append(index)
        lstLetters.append(stringToBeSaved[len(stringToBeSaved) - 1]) # 加入最后一个字符
        if (stringToBeSaved not in mydict):
            mydict[stringToBeSaved] = index  #以（index，string）形式存入字典中
            index = index + 1
        i = i + 1
    return lstNumbers, lstLetters, mydict


def decode(lstNumbers, lstLetters, mydict):
    i = 0
    while i < len(lstNumbers):
        if (lstNumbers[i] != 0):
            print(list(mydict.keys())[list(mydict.values()).index(lstNumbers[i])], end="")
        print(lstLetters[i], end="")
        i = i + 1
    print('\n')


if __name__ == "__main__":
    stringToEncode=input()
    [lstNumbers, lstLetters, mydict] = encode(stringToEncode)
    print("Encoded string: ", end="")
    i = 0
    while i < len(lstNumbers):
        print("<{0}, {1}>".format(lstNumbers[i], lstLetters[i]), end=" ")
        i = i + 1
    print('\n')
    print("Decoded string: ", end="")
    decode(lstNumbers, lstLetters, mydict)
```

LZW:

```
import sys
```

```
def encode(text):

    mydict = dict()
    i = 0
    index = 1
    while i < len(text):
        if text[i] in mydict:
            i = i + 1
        else:
            mydict[text[i]] = index
            index = index + 1

    # encode the text
    i = 0


    output = []
    while i < len(text):
        j = 0
        stringToBeSaved = text[i]
        # while the stringToBeSaved is in the mydict and we are not in the end of the string
        while stringToBeSaved in mydict and i + j < len(text):
            # save longest mydict occurence's index
            indexInDictionary = mydict[stringToBeSaved]
            length = len(stringToBeSaved)
            if (i + j == len(text) - 1):
                break
            j = j + 1
            stringToBeSaved = stringToBeSaved + text[i + j]
        i = i + length
        # print ("<{0}>".format(indexInDictionary))
        output.append(indexInDictionary)
        if (stringToBeSaved not in mydict):
            mydict[stringToBeSaved] = index
        index = index + 1
    return output, mydict

def decode(output, mydict):
    i = 0
    while i < len(output):
        print(list(mydict.keys())[list(mydict.values()).index(output[i])], end="")
        i = i+1
    print('\n')

```

```
46.
47.  if __name__ == "__main__":
48.      text=input()
49.      [output, mydict] = encode(text)
50.      print("output string: ", output)
51.      print("Decoded string: ", end="")
52.      decode(output, mydict)
```

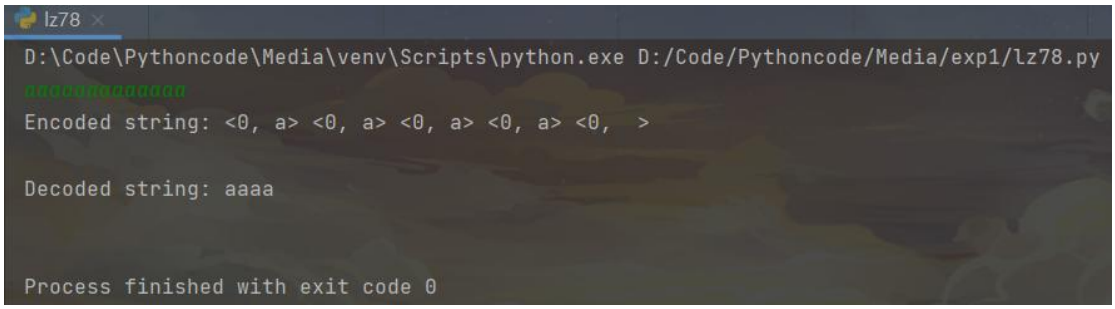
3、运行代码，并对以下字符串采用两种算法进行编码

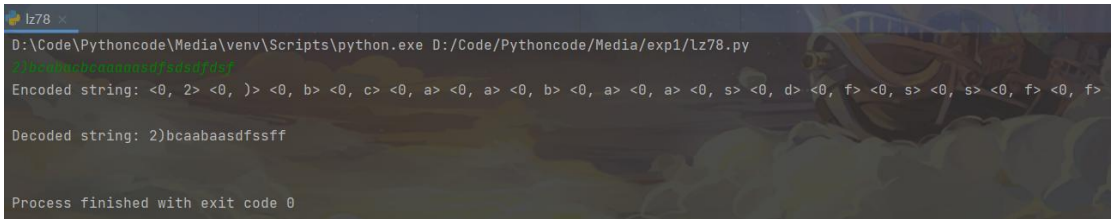
1) aaaaaaaaaaaa

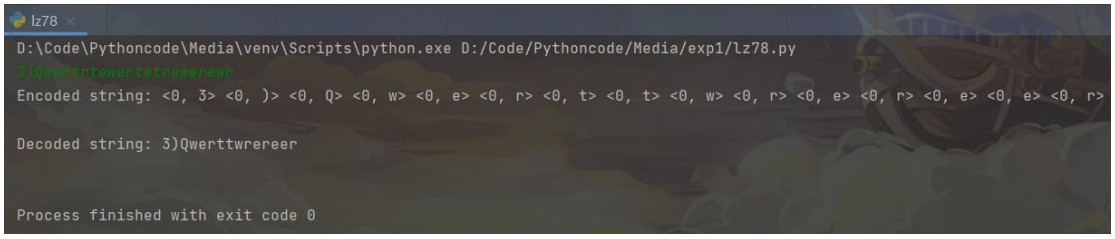
2) bcabacbcaaaaasdfsdsdfdsf

3) Qwertrtewertetrewerewr

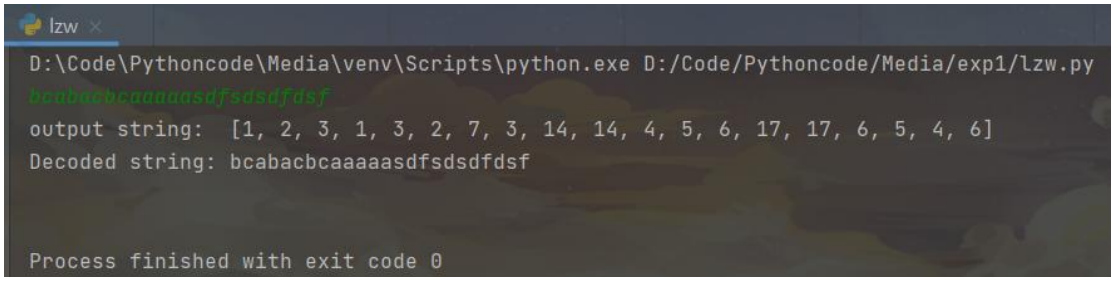
LZ78：

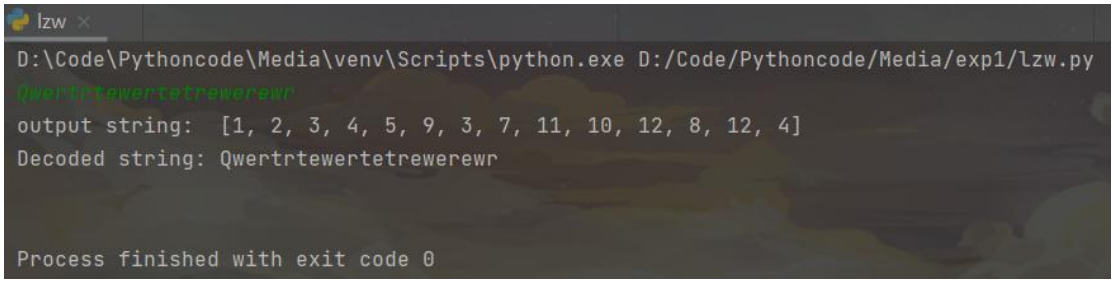






LZW：





二、 应用部分

用摄像头录制一段视频（5-10 秒即可），用 MoviePy 工具进行编辑，实现添加字幕、配音和水印（学号）等功能（也可自定义其他功能）。记录过程，和相关的截图。参考：https://zulko.github.io/moviepy/

参考文档：MoviePy 中文手册

https://moviepy-cn.readthedocs.io/zh/latest/

主要过程：

- 导入编辑视频剪辑所需的所有内容

```
# Import everything needed to edit video clips
from moviepy.editor import *
```

- 裁剪视频：0~6 秒

```
video = VideoFileClip("test.mp4").subclip(0,6)
```

- 添加字幕

```
# 添加字幕
string=["字幕：合肥工业大学","计算机与信息学院","智能科学与技术"]
```

```
    #起始时间
    time=[0.5, 1.7, 3]
    # 持续时间
    dur=[1,1,1]
    for a in range(3):
        txt_clip = TextClip(string[a],font='msyhbd.ttc',fontsize=70,color='white').set_pos('bottom').set_duration(dur[a]).set_start(time[a])
        video = CompositeVideoClip([clip, txt_clip])
```

- 添加配音

```
# 添加配音
peiyin=AudioFileClip("peiyin.aac")
```

- 添加背景音乐

```
#背景音乐
audio_clip = AudioFileClip(r'D:/Code/Pythoncode/Media/exp1_2/bgm.mp3').subclip(0,6).volumex(0.8)
#设置背景音乐循环，时间与视频时间一致
audio = afx.audio_loop(audio_clip, duration=clip.duration)
#配音和背景音乐，音频叠加
audio_clip_add = CompositeAudioClip([peiyin,audio])
#视频写入背景音
video = video.set_audio(audio_clip_add)
```

- 添加图片水印和文字水印

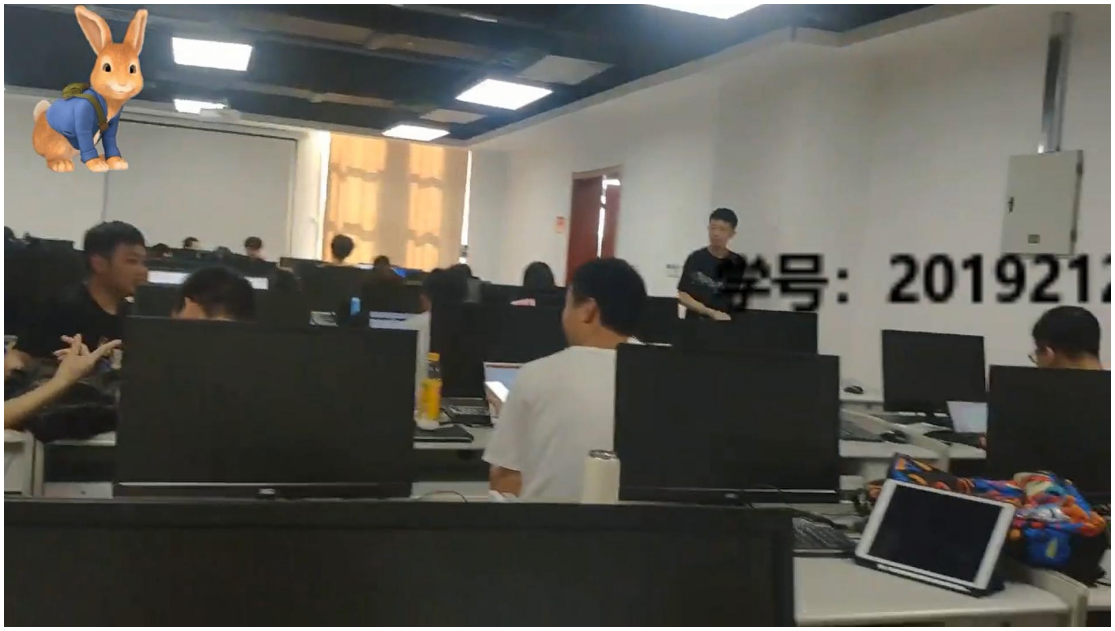
```
import moviepy.editor as mp
# 图片水印
logo = (mp.ImageClip("sy2.png")
        # 水印持续时间
        .set_duration(video.duration)
        # 水印高度，等比缩放
        .set_position(lambda t: (150*t, 50*t))  # 随着时间移动
        .resize(height=200)
        # 水印的位置
        .set_pos(('left', 'top')))

# 制作文字，指定文字大小和颜色
txt_clip = (TextClip("学号：2019212300",font = 'msyhbd.ttc')
            .set_position(lambda t: (150*t, 50*t))  # 随着时间移动
            .resize(height=100)
            .set_duration(video.duration))  # 水印持续时间

output = mp.CompositeVideoClip([video, logo, txt_clip])
# # 加上 aac 才有声音
```

- 输出成新的视频

```
output.write_videofile("output.mp4", audio_codec="aac", codec="libx264")
```

效果展示：

播放过程（含有配音，水印随时间移动）：



成绩

评语(学生工作评价):   成绩： 指导教师签字：

年　月　日