

WeirdSound的解题思路

题目信息

| 题目名 | 类型 | 难度 |
|------------|------|----|
| WeirdSound | MISC | 困难 |

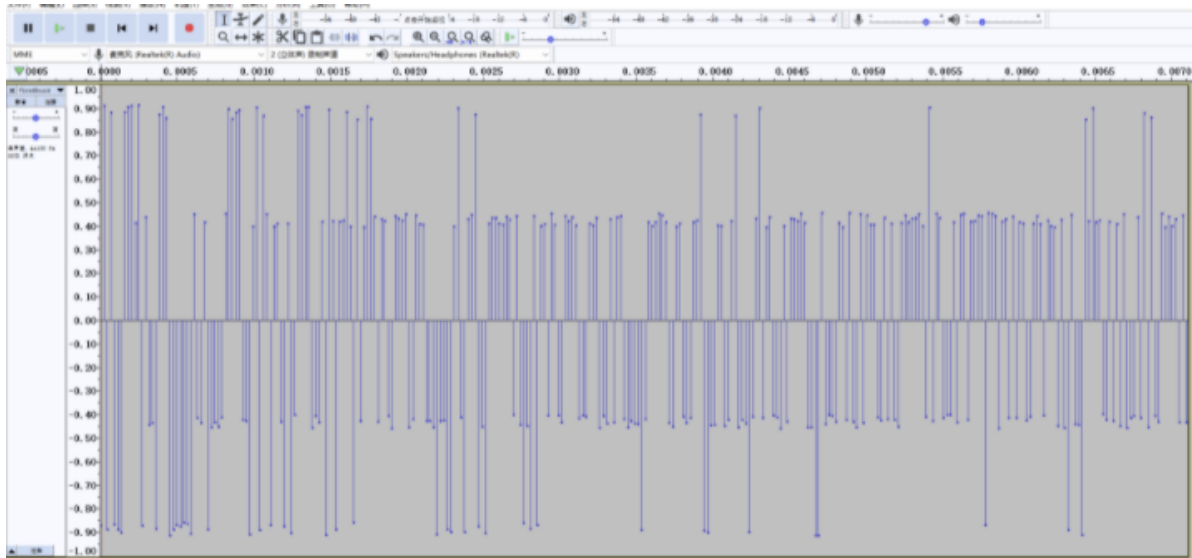
知识点

- Wav振幅高低振幅转换二进制数据

解题步骤

首先wi redsound 很明显是 wav 文件，添加后缀 .wav

原声听起来非常嘈杂，使用 Audacity 打开分析



整体看下来，可以发现一个比较明显的特征就是 波形的振幅有高有低

且高低振幅差距都比较明显，即可猜测振幅高即代表 1，振幅低代表 0，从而转换成二进制数据

转换之前先取一些数据做个分析

wav每一帧大小为 两字节，且存储方式为 小端存储，所以转换的时候注意高字节位和低字节位换一下

```

import wave

wav = wave.open('wiredSound.wav', 'r')
frames = wav.getnframes()
print("All Frames: {}".format(frames))
framehexdata = wav.readframes(20).hex() # 取前20帧数据做分析
for i in range(0, len(framehexdata), 4):
    data = framehexdata[i:i+4]
    real_data = int(data[2:] + data[:2], 16)
    data1 = data[2:] + data[:2]
    print("第{:<2}帧: {} => {} 数值: {}".format(int((i+4)/4), data, data1,
real_data))

```

运行之后

```

wiredSound\附件1> python .\analyse.py
All Frames: 2554344
第1 帧: 7190 => 9071 数值: 36977
第2 帧: f874 => 74f8 数值: 29944
第3 帧: 688e => 8e68 数值: 36456
第4 帧: 1b71 => 711b 数值: 28955
第5 帧: 1f91 => 911f 数值: 37151
第6 帧: 758e => 8e75 数值: 36469
第7 帧: cb8c => 8ccb 数值: 36043
第8 帧: 5371 => 7153 数值: 29011
第9 帧: 2574 => 7425 数值: 29733
第10帧: da74 => 74da 数值: 29914
第11帧: 2835 => 3528 数值: 13608
第12帧: 0b75 => 750b 数值: 29963
第13帧: 7c90 => 907c 数值: 36988
第14帧: 0b38 => 380b 数值: 14347
第15帧: 73c7 => c773 数值: 51059
第16帧: 80c8 => c880 数值: 51328
第17帧: f78e => 8ef7 数值: 36599
第18帧: f86f => 6ff8 数值: 28664
第19帧: 2f74 => 742f 数值: 29743
第20帧: 106e => 6e10 数值: 28176

```

或者使用 `O10 Editor` 加载wav结构分析插件，也可以看到正确的振幅值

```

起始页  WiredSound.wav X
编辑方式: 十六进制(0) 运行脚本: 运行模板: WAV.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 52 49 46 46 F4 F3 4D 00 57 41 56 45 66 6D 74 20 RIFFóóM.WAVEfmt
0010h: 10 00 00 00 01 00 01 00 44 AC 00 00 88 58 01 00 .....D~...^X..
0020h: 02 00 10 00 64 61 74 61 D0 F3 4D 00 71 90 F8 74 ....dataóóM.q.øt
0030h: 68 8E 1B 71 1F 91 75 8E CB 8C 53 71 25 74 DA 74 hž.q.'užĚŒSg%tÚt
0040h: 28 35 0B 75 7C 90 0B 38 73 C7 80 C8 F7 8E F8 6F (5.u|..8sÇĚÈ÷žø0
0050h: 2F 74 10 6E 47 8B 78 8E 15 91 34 90 6A 92 91 91 /t.nG<xž.'4.j'\`
0060h: 54 8C 02 3A 37 CB 55 C8 68 35 91 8E F5 C5 BD C8 TĚ.:7ĚUĚh5'žóĀ%è
0070h: 8F C6 8D CB 2D 3A 4C 73 DF 6D 3A 71 82 72 01 CA .Ě.Ě-:Lsβm:q,r.Ě
0080h: 82 C9 CF 8B 33 33 CF 73 25 8E 6D 6F BD 39 ED 90 ,ĚĪ<33Īs%žmo%9i.
0090h: 1B 33 C1 34 14 C9 57 90 D8 34 7A 8C CC CC 33 72 .3Ā4.ĚW.ø4zĚĪĪ3r
00A0h: CC 6F 29 74 09 74 EA C5 26 CC BA C8 F0 35 93 8B Īo)t.tĚĀ&Ī°Ěø5"<
00B0h: 7F 70 20 20 0F 0F 03 25 20 20 43 71 10 20 24 00 T~0C žžF ÇT~ 24/

```

```

模板结果 - WAV.bt
名称
> struct WAVRIFFHEADER header
> struct FORMATCHUNK format
> struct DATACHUNK data
  > ID chunkID[4]
    data
    long chunkSize
    5108688
  > short samples[2554344]
    short samples[0] -28559
    short samples[1] 29944
    short samples[2] -29080
    short samples[3] 28955
    short samples[4] -28385
    short samples[5] -29067
    short samples[6] -29493
    short samples[7] 29011
    short samples[8] 29733
    short samples[9] 29914
    short samples[10] 13608
    short samples[11] 29963
    short samples[12] -28548
    short samples[13] 14347
    short samples[14] -14477
    short samples[15] -14208
    short samples[16] -28937
    short samples[17] 28664
    short samples[18] 29743
    short samples[19] 28176
    short samples[20] -29881
    short samples[21] -29064
    short samples[22] -28395
    short samples[23] -28620

```

从正振幅上看，对应的数值看起来都是正确的，主要是负振幅的值，每一帧的大小是2字节，而这里负振幅的数据很明显都超过了2字节的最大正数值 $2^{15}-1=327687$ ，这里输出并不能表示负数，所以发生了溢出，转换一下就好了

```

>>> from math import *
>>> pow(2,15)-(36977-pow(2,15))
28559.0 # 第一帧
>>> pow(2,15)-(36456-pow(2,15))
29080.0 # 第三帧
>>> pow(2,15)-(37151-pow(2,15))
28385.0 # 第五帧

```

然后就是解决高低振幅范围的问题，这个其实很简单，多读几帧看看就能确定高振幅范围为：28000-30000 or (-30000)-(-28000)，低振幅范围为：13000-15000 or (-15000)-(-13000)

或者利用 010 Editor 还可以直接通过分析结构中的值直观看出来

确定读取正确数值方法以及判定范围，即可编写Python脚本将所有帧数据转换成二进制数据，然后再转换成文件

```

import wave, math, struct
from binascii import *

obj = wave.open('wiredSound.wav', 'r')

```

```

frames = obj.getnframes()
frames_data = obj.readframes(frames).hex()

bin_data = ''
for idx in range(0, len(frames_data), 4):
    data = frames_data[idx:idx+4]
    data = data[2:] + data[:2]
    if int(data, 16) <= 15000:
        bin_data += '0'
    elif int(data, 16) > 15000 and int(data, 16) <= 30000:
        bin_data += '1'
    elif int(data, 16) > math.pow(2, 15):
        overflow_data = math.pow(2, 15) - (int(data, 16) - math.pow(2, 15))
        if overflow_data > 15000 and overflow_data <= 30000:
            bin_data += '1'
        elif overflow_data <= 15000:
            bin_data += '0'

hex_data = ''
for idx in range(0, len(bin_data), 8):
    hex_data += '{:02x}'.format(int(bin_data[idx:idx+8], 2))

with open('flag.jpg', 'wb') as f1:
    f1.write(unhexlify(hex_data))

```

得到的图片并没有直观看出什么，使用 stegsolve 打开，发现隐藏的flag