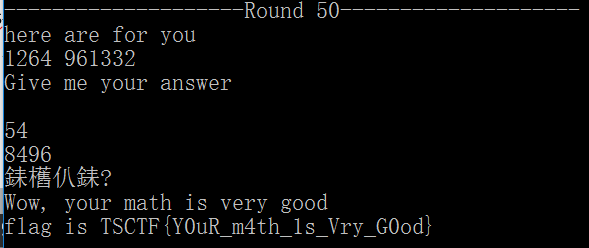
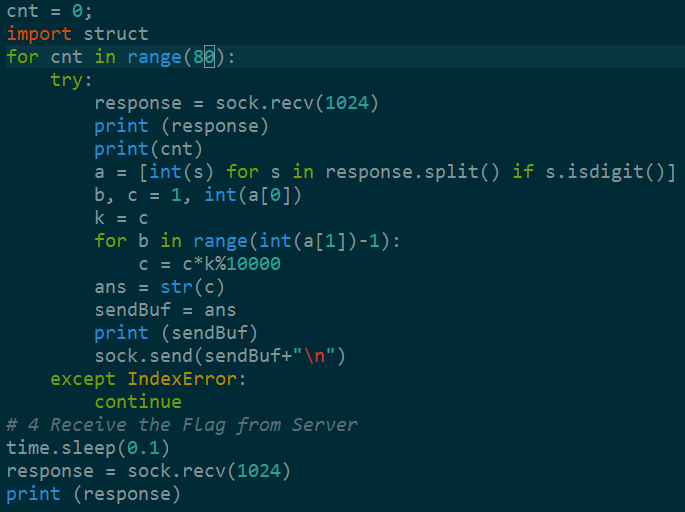
**Coding**

泽哥的算数：

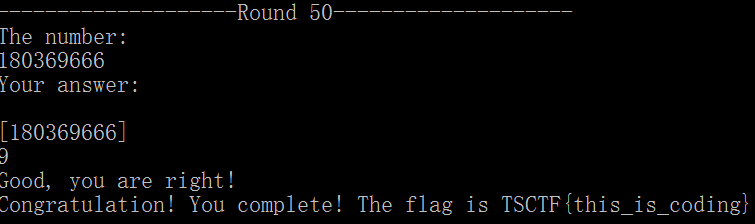
每次取余防止溢出或超时



小明二进制：

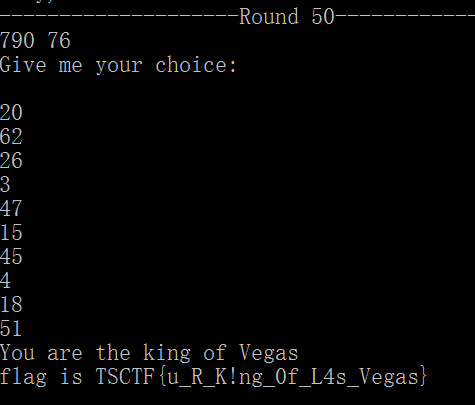
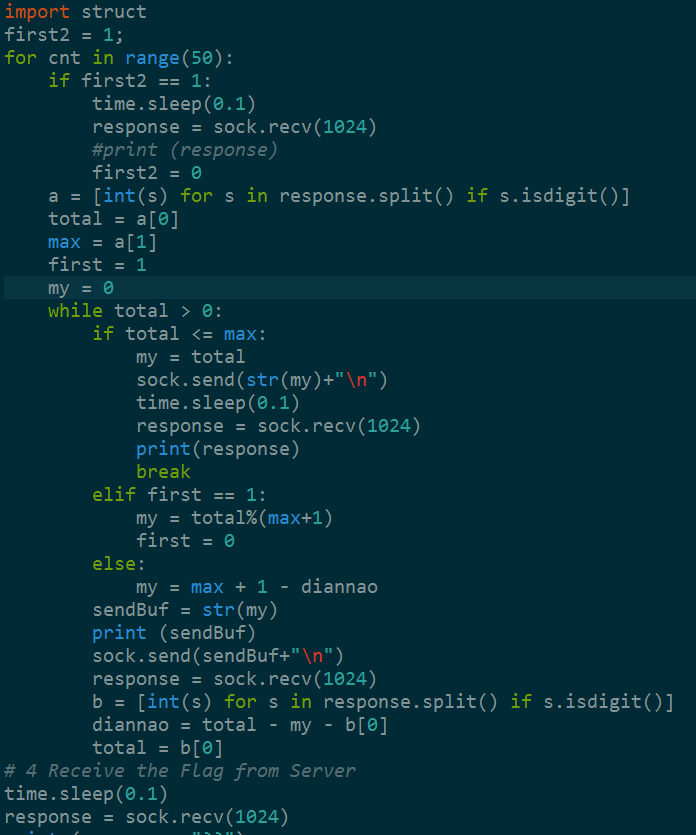
求一个数最少由几个小明二进制数组成，也就是让一个数一次减去一个小明二进制数，求最少减几次能把这个数变成0。每次每个位数上只能减0或1，所以最终的最少用的次数就是取这个数字中最大的一个数

感谢队友提供的思路



Las vegas：

好像叫做巴什博弈，也是搜到的，初始值不是max+1整数倍的话，先手一定赢，每次根据电脑取的数量，只要保证接下来剩下的是max+1的整数倍就可以。

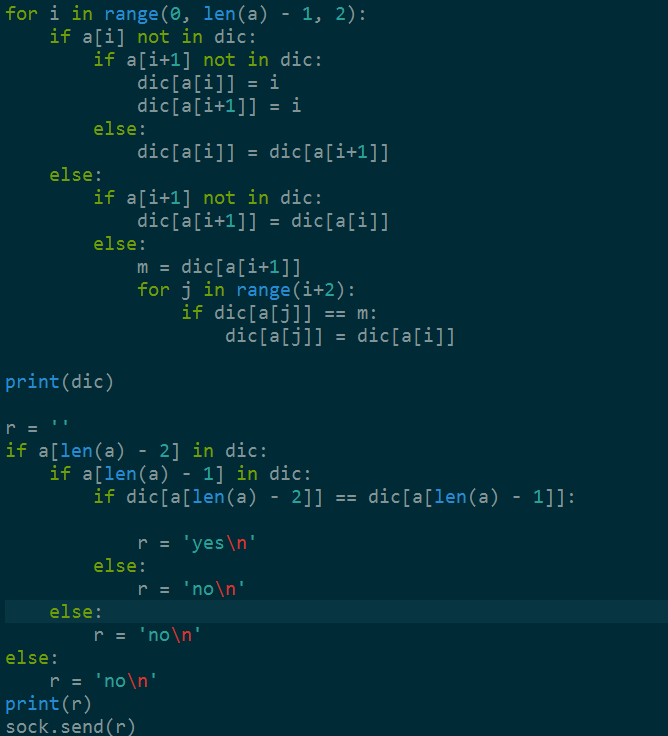


修路：

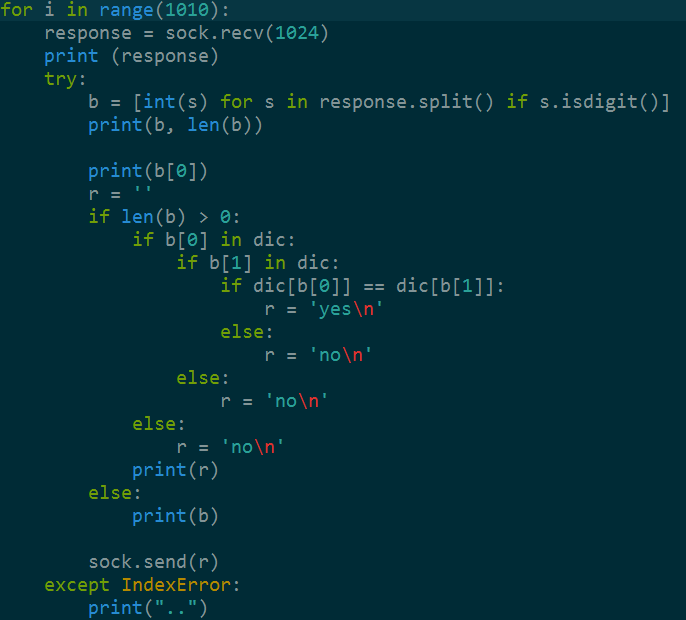
求无向图的连通子图，算法是并查集，用dict实现。接收一组连通信息，若dict不存在则加入或创建一个子集，若都存在则把两个子集连接起来。

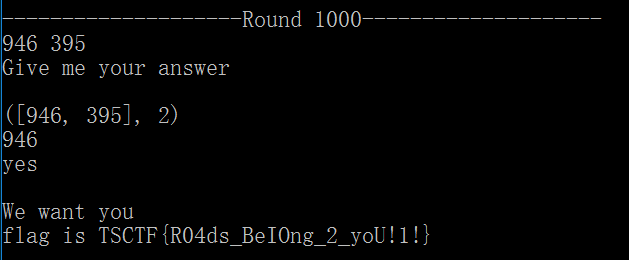
本来这算法没学过，搜了下发现思路挺简单，感觉python好好用 : )

先做并查集字典，因为一次取的数据过多，把round1的数也一次接收了，所以单独算round1（谢谢学长指点）



剩下继续判断：

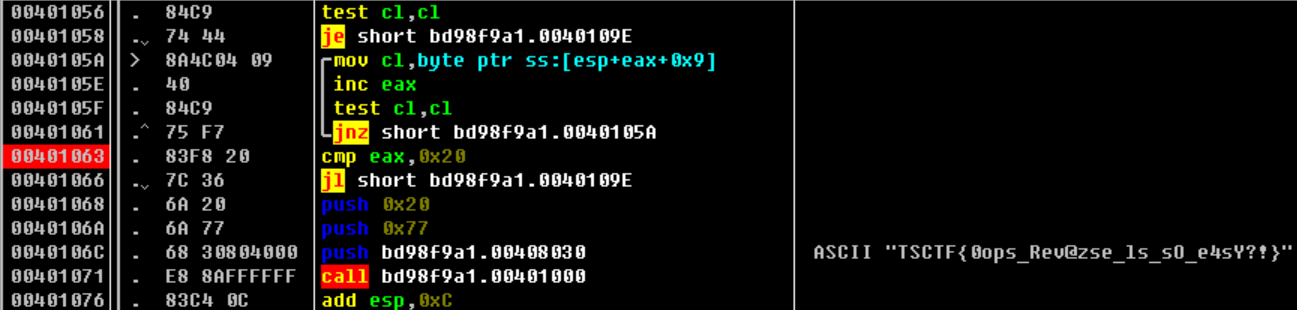




**Rev/apk：**

Check in:

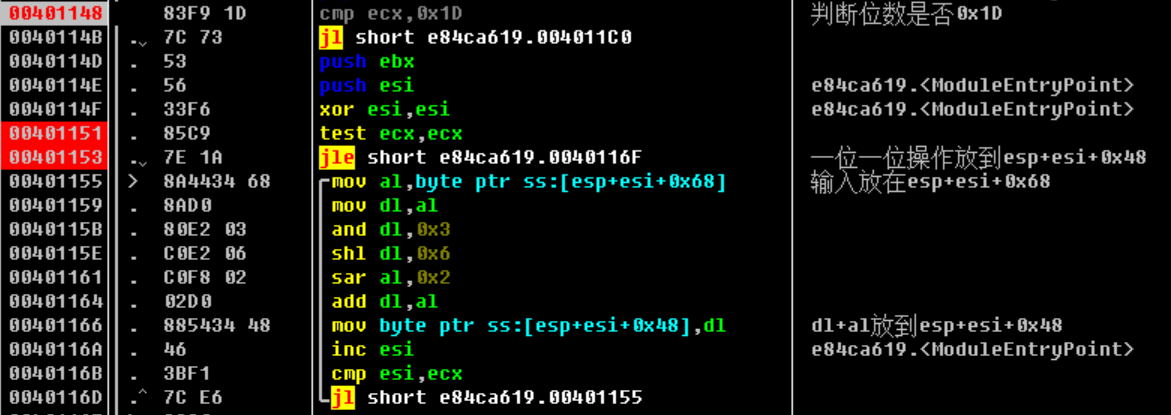
单步执行，发现key长度是0x20，否则直接跳转到incorrect，nop掉跳转句，继续单步执行发现key直接出来了。

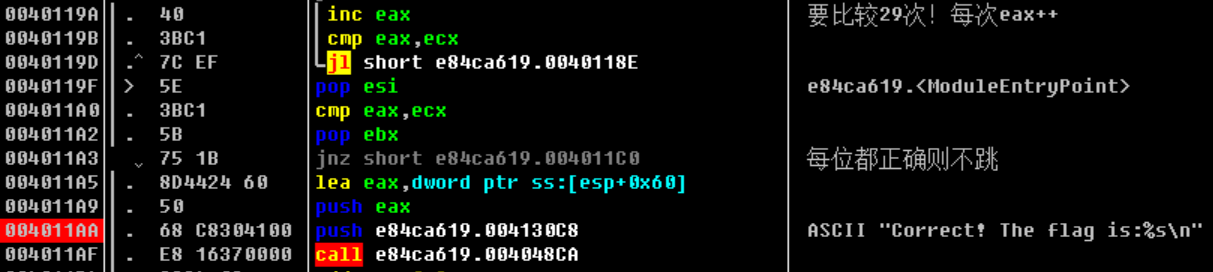


Take it easy：

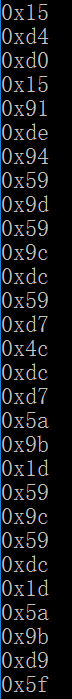
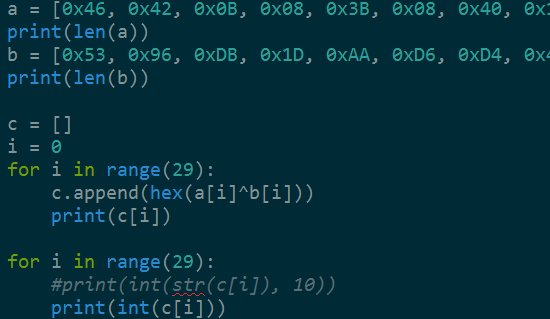
初学汇编。。。啃了好久，不知道有没有更直接的办法：

还是先发现他先判断输入长度，29位，继续单步执行，发现他拿我的输入，经过两次循环的位移啊异或啊什么的操作后，和一个内置的字符串进行比较判断，所以在OD里面把内置的字符串（在ss:[esp+eax+0x28]）一个一个抄出来，还有一个内置的串在（ss:[esp+eax+0x8] ），

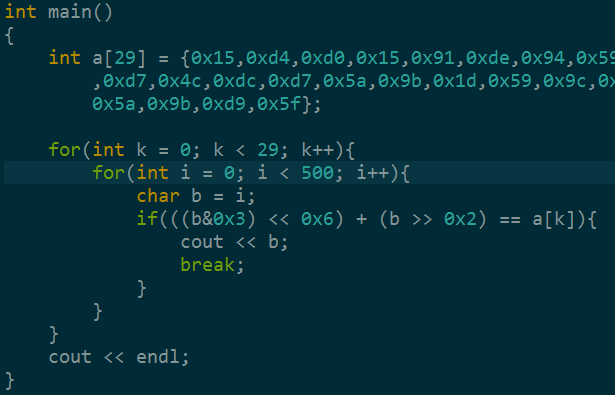


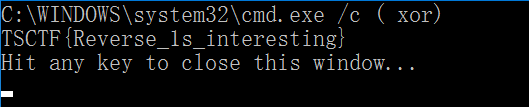


把两个内置字符串一个一个从OD里拿出来后写个程序，根据汇编的算法逆运算回去，算出正确输入，



再写个程序把十六进制的flag转换成字符：



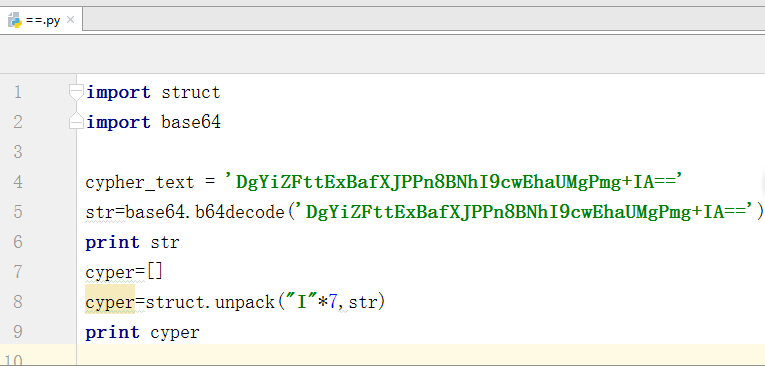


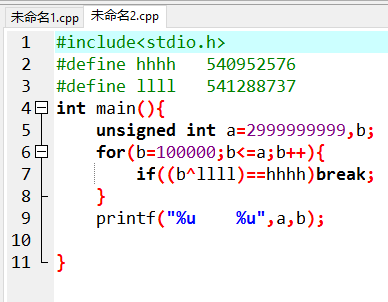
。。。。。。想看官方writeup。。。。。。

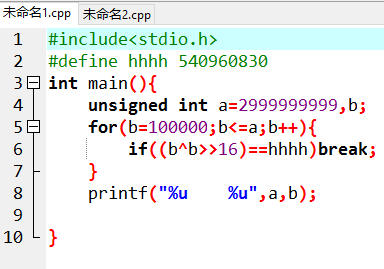
esayCrypto

拿到题目，发现是python，我并不会这一门语言，不过看起来除了个别语法之外和c语言并没有多大区别，百度了不懂的struck.pack和struct.unpack.明白了题目就是逆运算就可以了。

对于给出的密文，先base64解码struck.unpack得到一个7个unsigeed integer的数组



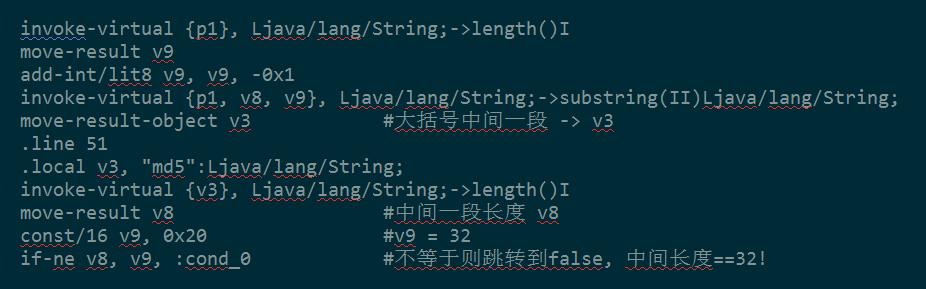
，之后根据题目给的位运算逆向利用了随手写的两个c程序一个一个算出了datas数组，之后用struct.pack获得了flag。这儿反思一下，其实两个程序在我写writeup时想到了可以写个循环程序一次性出来的，而比赛中用了大量时间手动操作，浪费了时间。



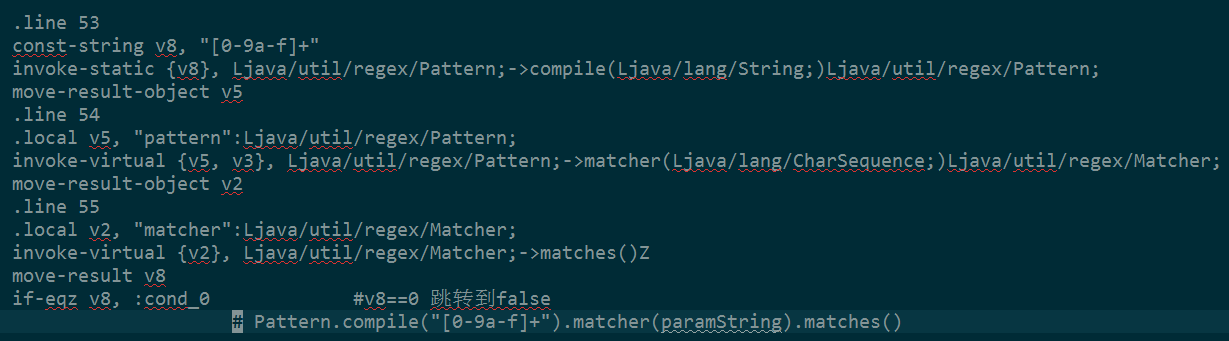
Baby\_android：

现学的安卓逆向各种工具和smali语法。。在jd-gui里的java代码逻辑看不懂，后来用apktool，直接看smali文件，发现和上面逆向的题差不多，把我的输入经过操作后和一串东西比较。

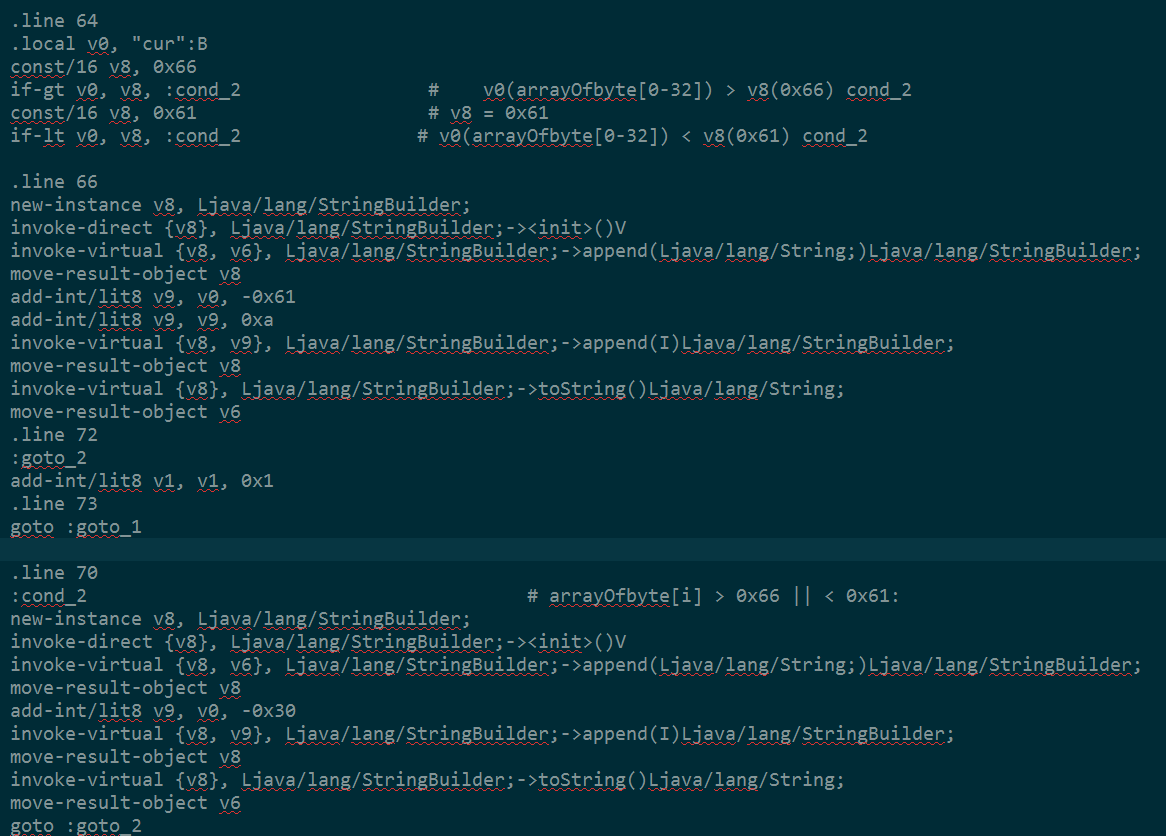
这一段把flag大括号中间一段拿出来，32位：



这一段好像是正则表达式？发现中间那段只有0-9a-f：



调用getByte换成后，如果是>=97或<=102说明是a-f，把它减去87加到字符串里，如果<=97说明是0-9,把它减去47加到字符串里：

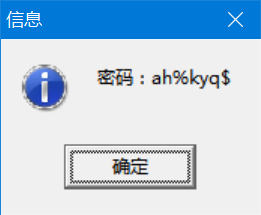
把最后那段1192811610815159146852912439081023130161513分割成32位在换回0-9a-f得出flag

TSCTF{b928b6a8ff9e68529c43908a23d016fd}

这长得也太不像flag了。。。。搜了好久拿到类似md5的串后该怎么办。。。

**Misc**

神秘的文件：

在notepad里能看到zip，还有后来的字典，用的网上找的ziperello：base64解码后就是flag

logo：

打开图片发现是天枢的logo，winhex打开最下面发现有一段字符串，看起来像是base64加密，复制用base64解密器解密，得出flag

签到：

扫二维码复制黏贴……