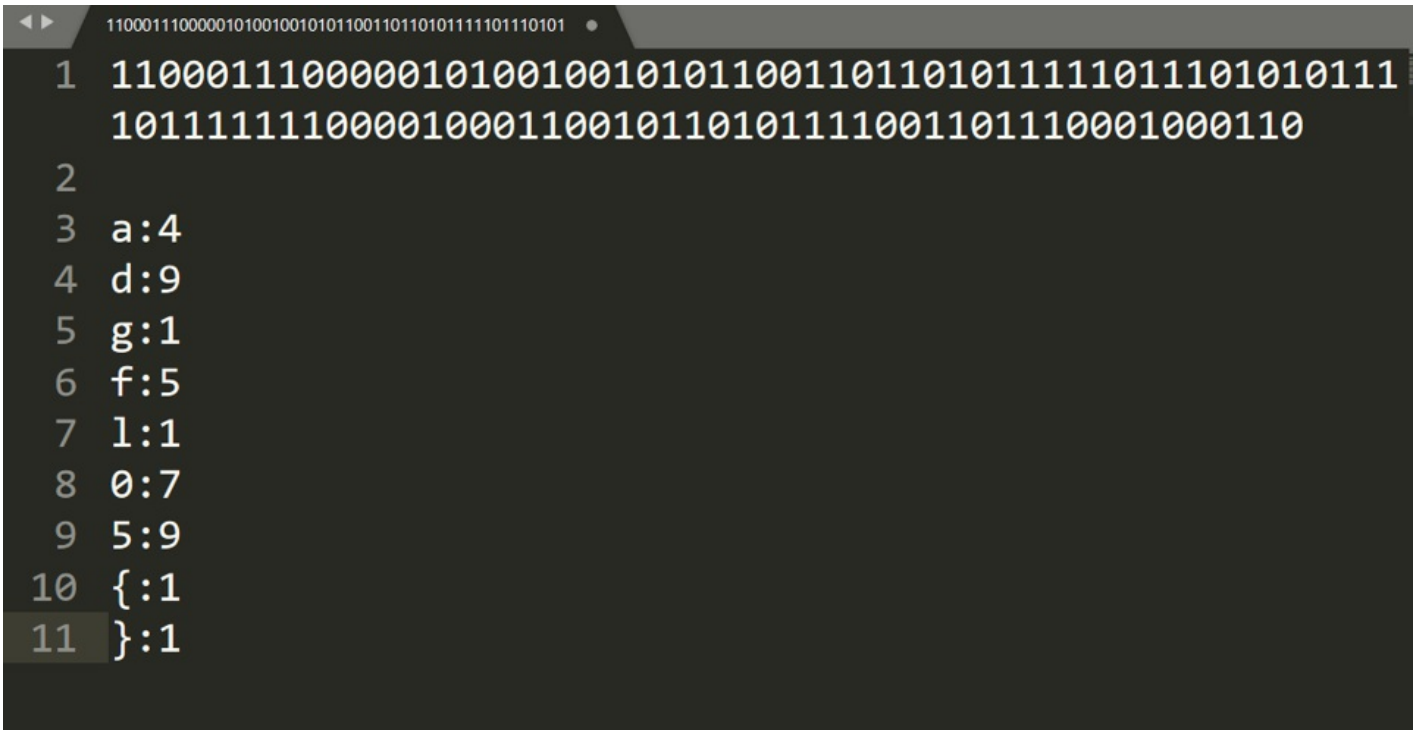
西湖论剑CTF2019

原创

a16511232 于 2019-04-11 14:38:00 发布 4221 收藏 1

Crypto

哈夫曼之谜



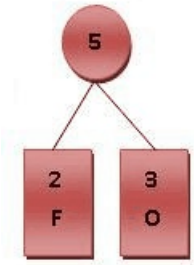
哈夫曼.png

下载下来就一个文件，里面全是0101和一些字符。

搜索哈夫曼，了解到哈夫曼压缩时用到的哈夫曼树。

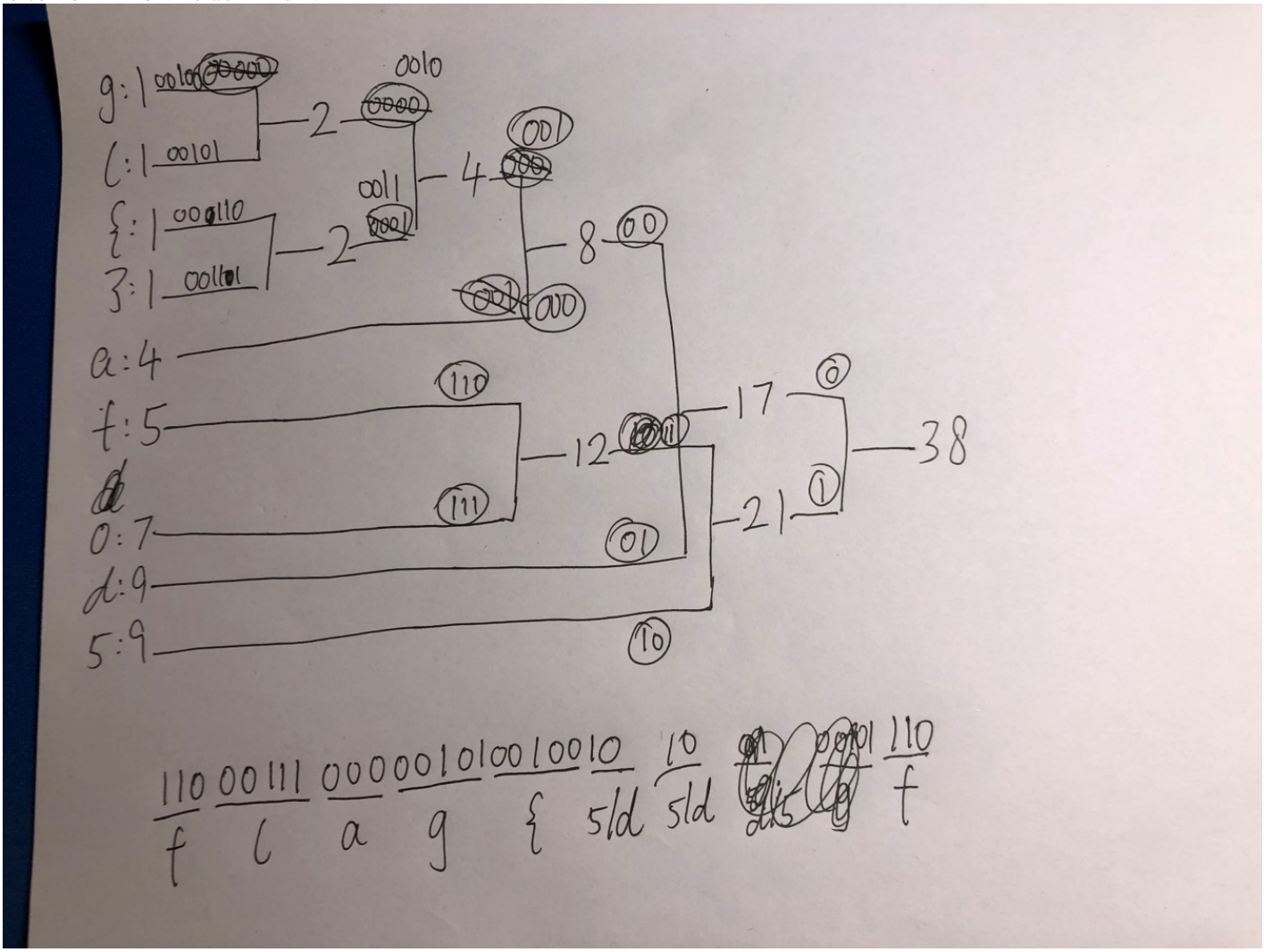
猜测下面的字符代表频率。

哈夫曼树建立过程如下



哈夫曼树建立.gif

字符不多，直接在草稿纸上手写。



哈夫曼手写.jpg

圆圈内就是构建好后对应字符的 哈夫曼编码。
在建立过程中，对于频率相同的字符在不同程序上排序的前后顺序可能不同，所以存在相同频率的字符哈夫曼编码不同。比如字符5和字符d的频率都是9，则字符d的哈夫曼编码可以是01，也可以是10。
然后对上面的01串进行解码，一般是以 `flag{`开头，所以前几个字符的哈夫曼编码可以确定下来。
但字符5和字符d的无法确定，所以得到两个flag。

`flag{ddf5dfd0f05550500a5af55dd0d5d0ad}`

flag{55fd5f50f0ddd0d00adafdd5505d50a5}
都提交上去就可以了

Hardgame

```python
flag1 = int('flag1{******************}'.encode('hex'),16)
flag2 = int('flag2{******************}'.encode('hex'),16)
flag = int('flag{******************}'.encode('hex'),16)
flag = flag1*flag2*flag
flag = libnum.n2s(flag)
p = number.getPrime(1024)
q = number.getPrime(1024)
n = p * q
e = number.getPrime(24)

def final(data):
    iv = os.urandom(256)
    fff = iv
    lll = binascii.hexlify(data)
    ooo = (len(lll) + 3) // 4
    kkk = lll.ljust(ooo * 4, b'f')
    for i in range(0, len(kkk), 4):
        qqq = number.bytes_to_long(kkk[i:i+4])
        www = pow(qqq, e, n)
        eee = binascii.unhexlify('%.512x' % www)
        fff += strxor(fff[-256:], eee)
    return base64.b64encode(fff)
f = open('enc','wb')
f.write('n=0x%x, e=0x%x' % (n, e))
f.write('\n')
f.write(final(flag))
f.close()
```

Hardgame.png

有三个flag，三段加密。

flag1

加密如下

```
n = 80367845335965467829490884979607087968395715079333476033191456456043634624631814697203734553219614484630239623156820812957822965842701613701581229476782976952618311036115521897353413263786862987869727688501062166779834099072247923220653973713463002267961138485665145767674250318155261257772388914848037181442700351991188782025999580913822488098597401924078736806386378567635290170632093300586632020997104756468726985566744912500603038857079174502560401299593296780884562686149406747757626955306318894286334582390440199311487506303917239894227392200691929791463607616576472431982097579035431617844707226402571062961279274462349383204346853605609294285080668223559278951675247267270289455229433207994433674284729765777111820207270697743634301201554591025634597867621179900279341273373266922969077784989097180467385359637467960548625099446206714449802133881666168001526050626012123346093003692879254719481099405181507957042513412777445437991143507101426673368439550927862534716776799293748523014679560356204884432499174326668666155331266289275676407763431338435327338882392875844836751881757226026820173509270505776422652436206224746614491397668247316050974170122064912373180229234431227192655345960223082314479121823620163578746247313,e = 2,c = 92179799413662202753778750958617109252070285517715206103872387348197592562230801756030321676580866698860321676580866698866613029629850463488651817405912513219666828485363315832435290
```

Hardgame-flag1.png

RSA加密，由于e只有2，相当于把明文m平方而已，得到的c也比n小很多。
尝试把c开根号看能否得到明文。
一般的python开根号方法精度较低，对大整数开出来的根号准确度低。
发现使用 `gmpy2`库可以对大整数开根号。

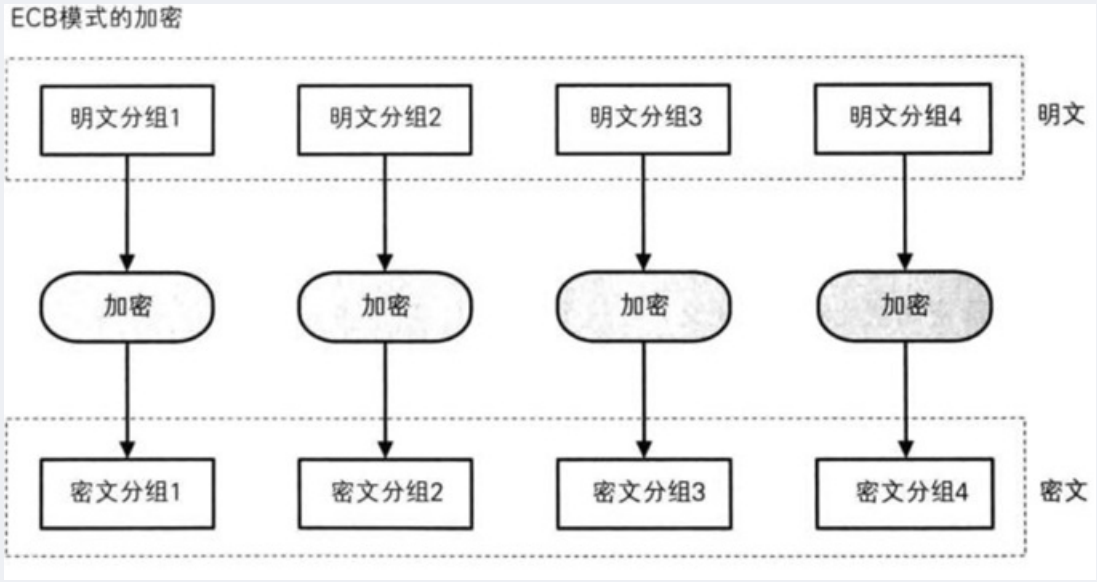
```
import gmpy2
import libnum
c = 92179799413662202753778750958617109252070285517715206103872387348197592562230801756030321676580866698860321676580866698866613029629850463488651817405912513219666828485363315832435290
m = gmpy2.isqrt(c)
m = int(m)
m_text = libnum.n2s(m)
print(m_text)
```

得到flag1`flag1{Th1s_i5_wHat_You_ne3d_FirsT}`

flag2

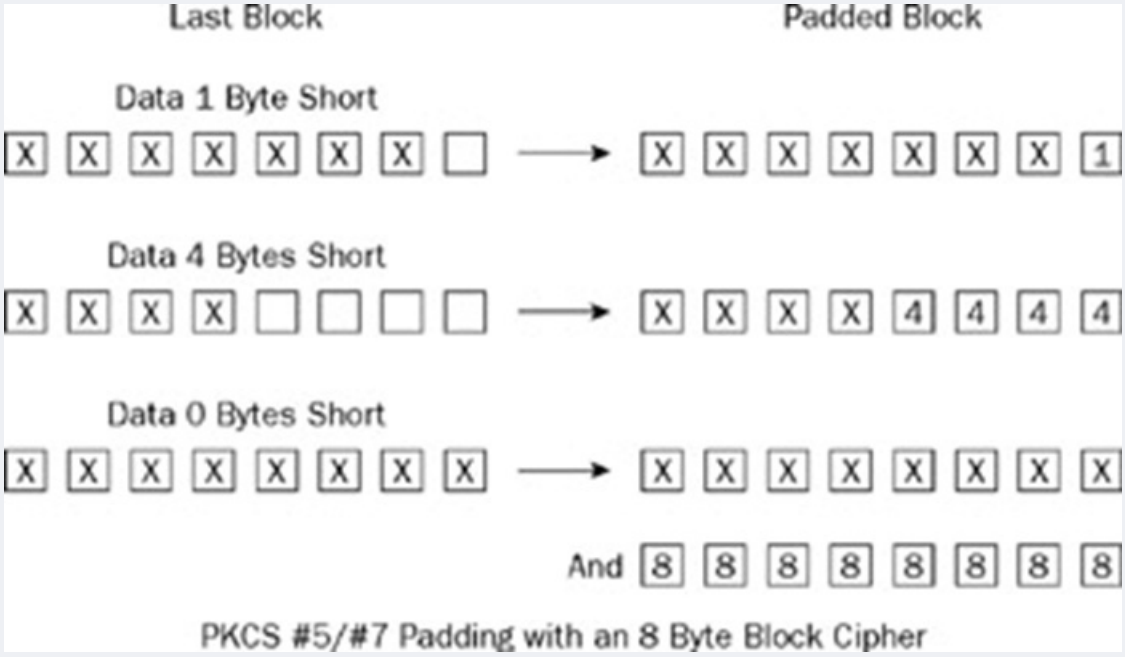
加密过程是DES-ECB-PKCS5，密钥是随机的8字节的大写字母

DES ECB（电子密本方式）其实非常简单，就是将数据按照8个字节一段进行DES加密或解密得到一段段的8个字节的密文或者明文，最后一段不足8个字节（一般补0或者F），按照需求补足8个字节进行计算（并行计算），之后按照顺序将计算所得的数据连在一起即可，各段数据之间互不影响。



ECB.png

填充使用的是PKCS5，就是全部填充到8字节，后面缺几位就填几
如：后面缺4个字节，就填04040404；
8个字节都是空，就填0808080808080808

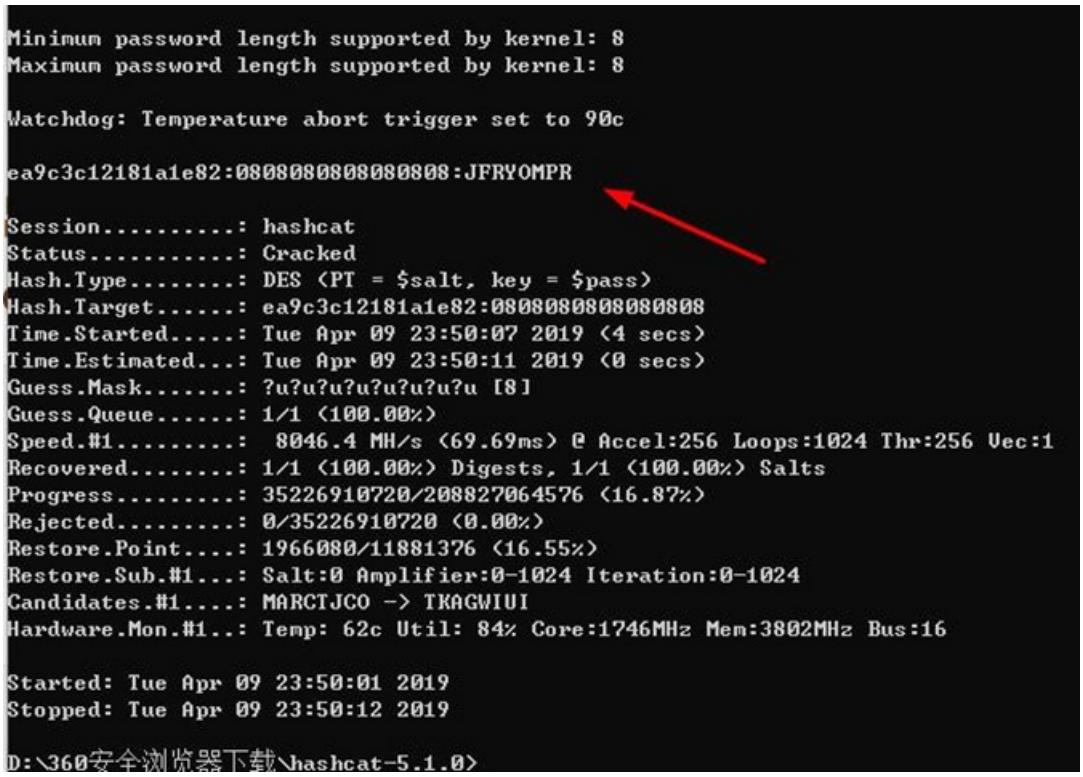


PKCS5.png

该算法对很多行明文按行分开加密，密钥是8字节，ECB加密是8字节一组，如果遇到空的8字节，就会出现填充后明文为0808080808080808与密文段进行异或。

截取各行密文的后8字节，统计发现`ea9c3c12181a1e82`和`16d0aa455a272fde`都出现过多次。
将密文放进hashcat进行爆破

```
hashcat64.exe -a 3 -m 14000 d33ad316eb246e5c:0808080808080808 -w 3 -O ?u?u?u?u?u?u?u?u
hashcat64.exe -a 3 -m 14000 ea9c3c12181a1e82:0808080808080808 -w 3 -O ?u?u?u?u?u?u?u?u
```



flag2.png

d33ad316eb246e5c解不出来
ea9c3c12181a1e82很快可得到密钥为 `JFRYOMPR`

然后解密脚本如下

```
from pyDes import *

Des_Key = "JFRYOMPR"
def DesDecrypt(str):
    k = des(Des_Key, ECB, pad=None, padmode=PAD_PKCS5)
    DecryptStr = k.decrypt(str.decode('hex'))
    return DecryptStr

with open('enc.txt','r') as fe:
    for line in fe.readlines():
        print DesDecrypt(line.strip('\n'))
```

得到flag2`flag2{Fuck_Y0u_cAn_Ge7_Se3ond}`

再看最终的flag
用的是rsa加密
n放到factordb无法解密
e也足够大
这里的加密用了类似于CBC的方式，区别在于先加密再异或
但分组长度过短，2字节一组
所以可以构建一个明文于密文的对应字典，通过查找的方式找回原文
且由于是先加密后异或，所以可以从后往前按顺序找出每个分组异或前的加密结果

解密脚本如下

```
#!usr/bin/env python
# -*- coding: utf-8 -*-
import base64
import binascii
import string
import libnum
from Crypto.Util import number
from Crypto.Util.strxor import strxor

n=0x834b44a67ea419e1c3e665cedf7790ebc5fb013e2304861b667232e7ec1cae53eb253639b348a6702561671a5c5c9105eacd5d4
e=0x9ae923
c="L+HK7Ksu+PoVJZBnIriWufZgKFH1APFFLFnKIf7OFjlBHRAkUtWaVp6y720Aux7f+ed+4rBIzsF7F+TwQAT+pOil825ftEJCTL+CUBgw

c_dec = base64.b64decode(c).strip()
#一开始iv有256位 每加密一个分组增加256位
length = len(c_dec)/256
#找出每组密文异或前的内容
c_list = []
for i in xrange(length,1,-1):
    c_list.insert( 0, strxor( c_dec[(i-1)*256:i*256], c_dec[(i-2)*256:(i-1)*256] ) )

#构建爆破字典
dic = []
for i in range(0,0xffff):
    m = str(hex(i)[2:]).rjust(4,'0')
    qqq = number.bytes_to_long(m)
    www = pow(qqq, e, n)
    eee = binascii.unhexlify('%.512x' % www)
    dic.append(eee)

#开始反查
m_hex = ""
for i in c_list:
    if i in dic:
        m_hex += hex(dic.index(i))[2:].rjust(4,'0')
    else:
        print "not found"

print m_hex

flag1 = int('flag1{Th1s_i5_wHat_You_ne3d_FirsT}'.encode('hex'),16)
flag2 = int('flag2{Fuck_Y0u_cAn_Ge7_Se3ond}'.encode('hex'),16)

m_num = libnum.s2n(binascii.unhexlify(m_hex))
m_num /= flag1*flag2
print libnum.n2s(m_num)
```

得到结果

```
10652cdf92fb9032a2e4c699448e3ca4ca266a667ccc5af2c95fae7f6de79fcd1fa52cfe72ee7fa3ab90a58c0c2310cfcc42dab372c
flag{64b60d7c2ddcf37f8d50358be1c35f45}
```

Misc

最短的路

题目.txt

```
#资深宅“flag{”在朋友邀请下，参加了一场聚会。
#在聚会上看到了美女“75D}”，一时心花荡漾、不能自己，坚信彼此就是天造地设的一双。
#想通过层层朋友的关系认识她，却无奈性格问题，不敢劳师动众。
#好在朋友帮忙搞到一张聚会人员关系图，如下：

[('FloraPrice','E11'),('FloraPrice','E9'),('FloraPrice','75D}'),('NoraFayette','E11'),('NoraFayette','E10'),('NoraFayette','E13'),('NoraFayette','E12'),('NoraFayette','E14'),('NoraFayette','E9'),('NoraFayette','E7'),('NoraFayette','E6'),('E10','SylviaAvondale'),('E10','MyraLiddel'),('E10','HelenLloyd'),('E10','KatherinaRogers'),('VerneSanderson','E7'),('VerneSanderson','E12'),('VerneSanderson','E9'),('VerneSanderson','E8'),('E12','HelenLloyd'),('E12','KatherinaRogers'),('E12','SylviaAvondale'),('E12','MyraLiddel'),('E14','SylviaAvondale'),('E14','75D}'),('E14','KatherinaRogers'),('FrancesAnderson','E5'),('FrancesAnderson','E6'),('FrancesAnderson','E8'),('FrancesAnderson','E3'),('DorothyMurchison','E9'),('DorothyMurchison','E8'),('EvelynJefferson','E9'),('EvelynJefferson','E8'),('EvelynJefferson','E5'),('EvelynJefferson','E4'),('EvelynJefferson','E6'),('EvelynJefferson','E1'),('EvelynJefferson','E3'),('EvelynJefferson','E2'),('RuthDeSand','E5'),('RuthDeSand','E7'),('RuthDeSand','E9'),('RuthDeSand','E8'),('HelenLloyd','E11'),('HelenLloyd','E7'),('HelenLloyd','E8'),('OliviaCarleton','E11'),('OliviaCarleton','E9'),('EleanorNye','E5'),('EleanorNye','E7'),('EleanorNye','E6'),('EleanorNye','E8'),('E9','TheresaAnderson'),('E9','PearlOglethorpe'),('E9','KatherinaRogers'),('E9','SylviaAvondale'),('E9','MyraLiddel'),('E8','TheresaAnderson'),('E8','PearlOglethorpe'),('E8','KatherinaRogers'),('E8','SylviaAvondale'),('E8','BrendaRogers'),('E8','LauraMandeville'),('E8','MyraLiddel'),('E5','TheresaAnderson'),('E5','BrendaRogers'),('E5','LauraMandeville'),('E5','CharlotteMcDowd'),('E4','CharlotteMcDowd'),('E4','TheresaAnderson'),('E4','BrendaRogers'),('E7','TheresaAnderson'),('E7','SylviaAvondale'),('E7','BrendaRogers'),('E7','LauraMandeville'),('E7','CharlotteMcDowd'),('E6','TheresaAnderson'),('E6','PearlOglethorpe'),('E6','BrendaRogers'),('E6','LauraMandeville'),('E1','LauraMandeville'),('E1','BrendaRogers'),('E3','TheresaAnderson'),('E3','BrendaRogers'),('E3','LauraMandeville'),('E3','CharlotteMcDowd'),('E3','flag{'),('E2','LauraMandeville'),('E2','TheresaAnderson'),('KatherinaRogers','E13'),('E13','SylviaAvondale')]

#你能在让最少人知道的情况下，帮助flag先生联系上75D小姐姐吗？
#求节点“flag{”到“75D”的最短路径，即为flag，比如：flag{E3AliceBobXXXXXXXXXXXXXXXX75D}
```

最短的路题目.png

简单来说就是找到最短路径，想起 迪杰斯特拉算法。
找了一下，发现python有自带的图论库 `networkx`，里面自带迪杰斯特拉算法。
代码如下：

```
import networkx as nx
import matplotlib.pyplot as plt

G=nx.Graph()
G.add_edge('FloraPrice','E11',weight=1)
G.add_edge('FloraPrice','E9',weight=1)
G.add_edge('FloraPrice','75D}',weight=1)
G.add_edge('NoraFayette','E11',weight=1)
G.add_edge('NoraFayette','E10',weight=1)
G.add_edge('NoraFayette','E13',weight=1)
G.add_edge('NoraFayette','E12',weight=1)
G.add_edge('NoraFayette','E14',weight=1)
G.add_edge('NoraFayette','E9',weight=1)
G.add_edge('NoraFayette','E7',weight=1)
G.add_edge('NoraFayette','E6',weight=1)
G.add_edge('E10','SylviaAvondale',weight=1)
G.add_edge('E10','MyraLiddel',weight=1)
G.add_edge('E10','HelenLloyd',weight=1)
G.add_edge('E10','KatherinaRogers',weight=1)
G.add_edge('VerneSanderson','E7',weight=1)
G.add_edge('VerneSanderson','E12',weight=1)
G.add_edge('VerneSanderson','E9',weight=1)
G.add_edge('VerneSanderson','E8',weight=1)
```

```
G.add_edge('E12','HelenLloyd',weight=1)
G.add_edge('E12','KatherinaRogers',weight=1)
G.add_edge('E12','SylviaAvondale',weight=1)
G.add_edge('E12','MyraLiddel',weight=1)
G.add_edge('E14','SylviaAvondale',weight=1)
G.add_edge('E14','75D}',weight=1)
G.add_edge('E14','KatherinaRogers',weight=1)
G.add_edge('FrancesAnderson','E5',weight=1)
G.add_edge('FrancesAnderson','E6',weight=1)
G.add_edge('FrancesAnderson','E8',weight=1)
G.add_edge('FrancesAnderson','E3',weight=1)
G.add_edge('DorothyMurchison','E9',weight=1)
G.add_edge('DorothyMurchison','E8',weight=1)
G.add_edge('EvelynJefferson','E9',weight=1)
G.add_edge('EvelynJefferson','E8',weight=1)
G.add_edge('EvelynJefferson','E5',weight=1)
G.add_edge('EvelynJefferson','E4',weight=1)
G.add_edge('EvelynJefferson','E6',weight=1)
G.add_edge('EvelynJefferson','E1',weight=1)
G.add_edge('EvelynJefferson','E3',weight=1)
G.add_edge('EvelynJefferson','E2',weight=1)
G.add_edge('RuthDeSand','E5',weight=1)
G.add_edge('RuthDeSand','E7',weight=1)
G.add_edge('RuthDeSand','E9',weight=1)
G.add_edge('RuthDeSand','E8',weight=1)
G.add_edge('HelenLloyd','E11',weight=1)
G.add_edge('HelenLloyd','E7',weight=1)
G.add_edge('HelenLloyd','E8',weight=1)
G.add_edge('OliviaCarleton','E11',weight=1)
G.add_edge('OliviaCarleton','E9',weight=1)
G.add_edge('EleanorNye','E5',weight=1)
G.add_edge('EleanorNye','E7',weight=1)
G.add_edge('EleanorNye','E6',weight=1)
G.add_edge('EleanorNye','E8',weight=1)
G.add_edge('E9','TheresaAnderson',weight=1)
G.add_edge('E9','PearlOglethorpe',weight=1)
G.add_edge('E9','KatherinaRogers',weight=1)
G.add_edge('E9','SylviaAvondale',weight=1)
G.add_edge('E9','MyraLiddel',weight=1)
G.add_edge('E8','TheresaAnderson',weight=1)
G.add_edge('E8','PearlOglethorpe',weight=1)
G.add_edge('E8','KatherinaRogers',weight=1)
G.add_edge('E8','SylviaAvondale',weight=1)
G.add_edge('E8','BrendaRogers',weight=1)
G.add_edge('E8','LauraMandeville',weight=1)
G.add_edge('E8','MyraLiddel',weight=1)
G.add_edge('E5','TheresaAnderson',weight=1)
G.add_edge('E5','BrendaRogers',weight=1)
G.add_edge('E5','LauraMandeville',weight=1)
G.add_edge('E5','CharlotteMcDowd',weight=1)
G.add_edge('E4','CharlotteMcDowd',weight=1)
G.add_edge('E4','TheresaAnderson',weight=1)
G.add_edge('E4','BrendaRogers',weight=1)
G.add_edge('E7','TheresaAnderson',weight=1)
G.add_edge('E7','SylviaAvondale',weight=1)
G.add_edge('E7','BrendaRogers',weight=1)
G.add_edge('E7','LauraMandeville',weight=1)
G.add_edge('E7','CharlotteMcDowd',weight=1)
G.add_edge('E6','TheresaAnderson',weight=1)
G.add_edge('E6','PearlOglethorpe',weight=1)
```

```
G.add_edge('E6','BrendaRogers',weight=1)
G.add_edge('E6','LauraMandeville',weight=1)
G.add_edge('E1','LauraMandeville',weight=1)
G.add_edge('E1','BrendaRogers',weight=1)
G.add_edge('E3','TheresaAnderson',weight=1)
G.add_edge('E3','BrendaRogers',weight=1)
G.add_edge('E3','LauraMandeville',weight=1)
G.add_edge('E3','CharlotteMcDowd',weight=1)
G.add_edge('E3','flag{',weight=1)
G.add_edge('E2','LauraMandeville',weight=1)
G.add_edge('E2','TheresaAnderson',weight=1)
G.add_edge('KatherinaRogers','E13',weight=1)
G.add_edge('E13','SylviaAvondale',weight=1)

nx.draw(G,pos = nx.random_layout(G),node_color = 'b',edge_color = 'r',with_labels = True,font_size =18,node
plt.savefig("wuxiangtu.png")
plt.show()
rs=nx.dijkstra_path(G,'flag{','75D}')
print(rs)
```

运行结果如下

```
(base) C:\Users\win10\Desktop>python dijkstra.py
['flag{', 'E3', 'EvelynJefferson', 'E9', 'FloraPrice', '75D}']
```

迪杰斯特拉算法.png

参考链接

西湖论剑2019 WriteUp -梅子酒的书札