第一次作业

> 计64 翁家翌 2016011446

Enigma

源代码见 `enigma.cpp`，直接运行即可。输出答案如下：

```
change C <-> R
change D <-> I
swap_res_num: 4
find key: 1129 sel: 2 0 3 plugboard: I <-> D J <-> E Q <-> B R <-> C T <-> M W <-> K
FSAYUABIANSUKDNHBRQWWIZQCKIVSDCXP
FSAYUABIANSUKDNHBRQWWIZQCKIVSDCXP
```

可见选择了3号、1号和4号，插线板为 BQ / CR / DI / EJ / KW / MT，初始位置分别为 l / r / b

具体过程如下：

1. 首先从已知明文和密文中寻找环，为简单起见我只找了长度不超过5的环，输出结果共17行，如下：

   ```
    2 C 10 A  8 S 24 A
    2 C 10 A 13 S 21 D 30 I
    2 C 24 A
    2 C 24 A  2 C 24 A
    8 S 10 A
    8 S 10 A  8 S 10 A
    8 S 24 A  2 C 10 A
   10 A  8 S
   10 A  8 S 10 A  8 S
   10 A  8 S 24 A  2 C
   10 A 13 S 21 D 30 I  2 C
   13 S 21 D 30 I  2 C 10 A
   21 D 30 I  2 C 10 A 13 S
   24 A  2 C
   24 A  2 C 10 A  8 S
   24 A  2 C 24 A  2 C
   30 I  2 C 10 A 13 S 21 D
   ```

   其实画出来发现只有四个单环，而不是声称的八个：CA / SA / CASA / CASDI
2. 在空插线板的情况下枚举key和select rotor，一共需要$26*26*26*5*4*3$次，每次枚举都过一遍17个环，看看输入的第一个字母经过变换之后还是不是这个字母
3. 我运行了一下发现没有任何一种情况能够同时符合17个环，最多是12个同时满足，接下来是11，因此我设定了threshold为12/11的时候进行下一步暴力枚举plugboard
4. 从步骤3中满足情况的那些环中计算出一定有哪些字母是要交换的，大部分情况下交换两对，剩下4对暴力枚举，这样大概是C_{22}^8，算下来还是能接受的。

然后大概就等一分钟就能出结果了。

1.5

题中说明了是移位密码，编写如下python代码：

```python
s='BEEAKFYDJXUQYHYJIQRYHTYJIQFBQDUYJIIKFUHCQD'
def calc(offset):
    for i in s:
        print(chr((ord(i)-ord('A')+offset)%26+ord('A')),end='')
    print()
for i in range(26):
    calc(i)
```

即：枚举移位长度，人工查看结果是否合理，输出如下：

```
BEEAKFYDJXUQYHYJIQRYHTYJIQFBQDUYJIIKFUHCQD
CFFBLGZEKYVRZIZKJRSZIUZKJRGCREVZKJJLGVIDRE
DGGCMHAFLZWSAJALKSTAJVALKSHDSFWALKKMHWJESF
EHHDNIBGMAXTBKBMLTUBKWBMLTIETGXBMLLNIXKFTG
FIIEOJCHNBYUCLCNMUVCLXCNMUJFUHYCNMMOJYLGUH
GJJFPKDIOCZVDMDONVWDMYDONVKGVIZDONNPKZMHVI
HKKGQLEJPDAWENEPOWXENZEPOWLHWJAEPOOQLANIWJ
ILLHRMFKQEBXFOFQPXYFOAFQPXMIXKBFQPPRMBOJXK
JMMISNGLRFCYGPGRQYZGPBGRQYNJYLCGRQQSNCPKYL
KNNJTOHMSGDZHQHSRZAHQCHSRZOKZMDHSRRTODQLZM
LOOKUPINTHEAIRITSABIRDITSAPLANEITSSUPERMAN
MPPLVQJOUIFBJSJUTBCJSEJUTBQMBOFJUTTVQFSNBO
NQQMWRKPVJGCKTKVUCDKTFKVUCRNCPGKVUUWRGTOCP
ORRNXSLQWKHDLULWVDELUGLWVDSODQHLWVVXSHUPDQ
PSSOYTMRXLIEMVMXWEFMVHMXWETPERIMXWWYTIVQER
QTTPZUNSYMJFNWNYXFGNWINYXFUQFSJNYXXZUJWRFS
RUUQAVOTZNKGOXOZYGHOXJOZYGVRGTKOZYYAVKXSGT
SVVRBWPUAOLHPYPAZHIPYKPAZHWSHULPAZZBWLYTHU
TWWSCXQVBPMIQZQBAIJQZLQBAIXTIVMQBAACXMZUIV
UXXTDYRWCQNJRARCBJKRAMRCBJYUJWNRCBBDYNAVJW
VYYUEZSXDROKSBSDCKLSBNSDCKZVKXOSDCCEZOBWKX
WZZVFATYESPLTCTEDLMTCOTEDLAWLYPTEDDFAPCXLY
XAAWGBUZFTQMUDUFEMNUDPUFEMBXMZQUFEEGBQDYMZ
YBBXHCVAGURNVEVGFNOVEQVGFNCYNARVGFFHCREZNA
ZCCYIDWBHVSOWFWHGOPWFRWHGODZOBSWHGGIDSFAOB
ADDZJEXCIWTPXGXIHPQXGSXIHPEAPCTXIHHJETGBPC
```

注意到有一行还是挺合理的：`LOOKUPINTHEAIRITSABIRDITSAPLANEITSSUPERMAN`，正常应该是 `Look up in the air! It's a bird! It's a plane! It's superman!`

1.16

π^{-1} 如下所示

x	1	2	3	4	5	6	7	8
$\pi^{-1}(x)$	2	4	6	1	8	3	5	7

完整python代码如下：

```
import numpy as np
    pi_orig = np.array([4,1,6,2,7,3,8,5])-1
      pi_rev = np.zeros_like(pi_orig)
        for i in range(pi_orig.shape[0]):
    pi_rev[pi_orig[i]] = i
print(pi_rev+1)
  # should be [2 4 6 1 8 3 5 7]

def convert(s):
    for i in pi_orig:
        print(s[i], end='')
s='ETEGENLMDNTNEOORDAHATECOESAHLRMI'
for i in range(0, len(s), pi_rev.shape[0]):
    convert(s[i:i+pi_rev.shape[0]])
```

输出结果为 `GENTLEMENDONOTREADEACHOTHERSMAIL`，正常应该是 `Gentlemen do not read each other's mail.`

1.21 a

对原文进行词频统计，代码如下：

```
s='EMGLOSUDCGDNCUSWYSFHNSFCYKDPUMLWGYICOXYSIPJCKQPKUGKMGOLICGINCGACKSNISACYKZSCKXECJCKS
HYSXCGOIDPKZCNKSHICGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZUGFZCCNDGYYSFUSZCNXEOJNCGYEOWEUPXEZGA
CGNFGLKNSACIGOIYCKXCJUCIUZCFZCCNDGYYSFEUEKUZCSOCFZCCNCIACZEJNCSHFZEJZEGMXCYHCJUMGKUCY'
for i in range(26):
     c = chr(i + ord('A'))
     print('%.2f' % ( s.count(c)/len(s)))
```

结果如下：

A	B	C	D	E	F	G	H	I	J	K	L	M
0.02	0.00	0.14	0.03	0.05	0.04	0.09	0.02	0.06	0.03	0.07	0.03	0.02
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0.05	0.04	0.02	0.00	0.00	0.08	0.00	0.05	0.00	0.02	0.03	0.06	0.05

大概率推测 C->E，而题目中给出提示 F->w，则字符串为 `EMGLOSUDeGDNeUSWYSwHNSweYKDPUMLWGYIeOXYSIPJeKQPKUGKMGOLIeGINeGAeKSNISAeYKZSeKXEeJeKSHYSXeGOIDPKZeNKSHIeGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZUGwZeeNDGYYSwUSZeNXEOJNeGYEOWEUPXEZGAeGNwGLKNSAeIGOIYeKXeJUeIUZewZeeNDGYYSwEUEKUZeSOewZeeNeIAeZEJNeSHwZEJZEGMXeYHeJUMGKUeY`

接下来考虑双字出现频率，代码如下：

```python
before = np.zeros(26)
after = np.zeros(26)
for i in range(len(s)):
    if s[i] == 'e':
        prev = ord(s[i-1]) - ord('A')
        next = ord(s[i+1]) - ord('A')
        before[prev] += 1
        after[next] += 1
for i in range(26):
    c = chr(i + ord('A'))
    print(c, end=' ')
print()
for i in range(26):
    c = chr(i + ord('A'))
    print('%d ' % before[i], end='')
print('# _e')
for i in range(26):
    c = chr(i + ord('A'))
    print('%d ' % after[i], end='')
print('# e_')
```

输出如下:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
5 0 3 1 1 1 0 1 3 2 0 0 0 5 1 0 0 0 1 0 2 0 0 3 1 7 # _e
0 0 3 0 0 2 7 0 3 3 5 0 0 5 1 0 0 0 2 0 1 0 0 0 4 1 # e_
```

其中 N 出现次数最多，但是经过试验发现 N 并不对应 r，感觉很诡异

注意到在s中出现三次 `wZeeN`，如果是一个单词的话大概率是 `wheel`，猜测 Z->h, N->l，此时字符串为 `EMGLOSUDeGDleUSWYSwHlSweYKDPUMLWGYIeOXYSIPJeKQPKUGKMGOLIeGIleGAeKSlISAeYKhSeKXEeJeKSHYSXeGOIDPKhelKSHIeGIWYGKKGKGOLDSILKGOIUSIGLEDSPWhUGwheelDGYYSwUShelXEOJleGYEOWEUPXEhGAeGlwGLKlSAeIGOIYeKXeJUeIUhewheelDGYYSwEUEKUheSOewheeleIAehEJleSHwhEJhEGMXeYHeJUMGKUeY`

此时确定一下the对应的密文，因为h和e已经确定出来了，只需找到t对应的密文即可。除whe外，有一次Khe、一次She和两次Uhe，猜测 U->t，字符串为 `EMGLOStDeGDletSWYSwHlSweYKDPtMLWGYIeOXYSIPJeKQPKtGKMGOLIeGIleGAeKSlISAeYKhSeKXEeJeKSHYSXeGOIDPKhelKSHIeGIWYGKKGKGOLDSILKGOItSIGLEDSPWhtGwheelDGYYSwtShelXEOJleGYEOWEtPXEhGAeGlwGLKlSAeIGOIYeKXeJteIthewheelDGYYSwEtEKtheSOewheeleIAehEJleSHwhEJhEGMXeYHeJtMGKteY`

最后有一个 `theSOewheel`，查了查发现只可能是 `one`，于是 S->o, O->n，字符串为`EMGL not DeGDle to WYowHloweYKDPtMLWGYIenXYoIPJeKQPKtGKMGnLIeGIleGAeKolIoAeYKhoeKXEeJeKoHYoXeGnIDPKhelKoHIeGIWYGKKGKGnLDoILKGnItoIGLEDoPWhtG wheel DGYYow to helXEnJleGYEnWEtPXEhGAeGlwGLKloAeIGnIYeKXeJteI the wheel DGYYowEtEK the one wheel eIAehEJleoHwhEJhEGMXeYHeJtMGKteY`

首字母为E，出现频率挺高，有 `EMGL` 和 `EtEK`，感觉上 E->i，而 `itiK` 中 K最有可能是 s，猜 K->s，字符串为`iMGL not DeGDle to WYowHloweYsDPtMLWGYIenXYoIPJesQPstGsMGnLIeGIleGAesolIoAeY shoes XieJesoHYoXeGnIDPshel so HIeGIWYGssGsGnLDoILsGnI to IGLiDoPWhtG wheel DGYYow to helXinJleGYinWitPXihGAeGlwGLsloAeIGnIYesXeJteI the wheel DGYYow it is the one wheel eIAehiJleoHwhiJhiGMXeYHeJtMGsteY`

下面只能乱猜了......看到有个 `loweY` 猜 Y->r，看到 `whiJh` 猜 J->c，有个`GssGsG` 查两个连续字母表，发现只剩下a能对应，猜测 G->a，此时字符串为 `i MaL not DeaDle to Wrow Hlowers DPtMLWarIenXroIPcesQPstasManLIeaIleaAesolIoAer shoes Xiece so HroXeanIDPshelsoHIeaI Wrass as an LDoILsanItoIaLiDoPWhta wheel Darrow to helX in clearinW it PXihaAealwaLsloAeIanIresXecteI the wheel Darrow it is the one wheel eIAehicleoH which i aM XerHect Master`

很明显推测 W->g, H->f, M->m，为`i maL not DeaDle to grow flowers DPtmLgarIenXroIPcesQPst as manLIeaIleaAesolIoAer shoes Xieces of roXeanIDPshels of IeaI grass as an LDoILsanI to IaLiDoPghta wheel Darrow to helX in clearing it PXihaAealwaLsloAeIanIresXecteI the wheel Darrow it is the one wheel eIAehicle of which i am Xerfect master`

推测 X->p, L->y, 查了下 `Darrow`有什么词对应，有 `barrow`、`darrow`、`farrow`、`harrow`、`marrow`、`narrow`，考虑到 f、h、m、n 已经被使用，只可能是b和d，而Darrow只有一个人名的意思，因此很大概率是 D->b。此时字符串：`i may not be able to grow flowers bPt my garIen proIPces QPst as many IeaIleaAesolIoAershoes pieces of rope anI bPshels of IeaI grass as anyboIy sanI toIay i boPght a wheel barrow to help in clearing it PpihaAe always loAeIanI respect eI the wheel barrow it is the one wheel eIAehicle of which i am perfect master`

推测 I->d, P->u：`i may not be able to grow flowers but my garden produces Qust as many dead leaAes old oAershoes pieces of rope and bushels of dead grass as anybody sand today i bought a wheel barrow to help in clearing it up i haAe always loAed and respected the wheel barrow it is the one wheeled Aehicle of which i am perfect master`

剩下直接能猜出来：Q->j, A->v，最终如下：

```
I may not be able to grow flowers, but my garden produces just as many dead leaves, old
overshoes pieces of rope, and bushels of dead grass as anybody's. And today I bought a
wheel barrow to help in clearing it up. I have always loved and respected the wheel
barrow. It is the one wheeled vehicle of which I am perfect master.
```

1.21 b

对字符串进行不同间隔的频率分析，python代码如下：

```python
s='KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFDETDGILTXRGUDDKOTFMBPVGEGLTGCKQRACQCWDNAWCRXIZAKFTLE
WRPTYCQKYVXCHKFTPONCQQRHJVAJUWETMCMSPKQDYHJVDAHCTRLSVSKCGCZQQDZXGSFRLSWCWSJTBHAFSIASPRJ
AHKJRJUMVGKMITZHFPDISPZLVLGWTFPLKKEBDPGCEBSHCTJRWXBAFSPEZQNRWXCVYCGAONWDDKACKAWBBIKFTIO
VKCGGHJVLNHIFFSQESVYCLACNVRWBBIREPBBVFEXOSCDYGZWPFDTKFQIYCWHJVLNHIQIBTKHJVNPIST'
def test(s):
    p = np.zeros(26)
    for i in s:
        p[ord(i) - ord('A')] += 1
    return (p * (p-1)).sum() / (p.sum() * (p.sum() - 1))
for m in range(1, 10):
    print(m, end=': ')
    for i in range(m):
        print('%.6f' % test(s[i::m]), end=' ')
    print()
```

输出结果如下：

```
1: 0.040872
2: 0.038462 0.047120
3: 0.055942 0.048102 0.048263
4: 0.037255 0.042742 0.037579 0.049053
5: 0.042581 0.043020 0.032564 0.035278 0.042967
6: 0.062657 0.083766 0.049351 0.064935 0.042857 0.073377
7: 0.030612 0.044326 0.043440 0.040780 0.044326 0.044326 0.040780
8: 0.033223 0.040650 0.033682 0.040650 0.039489 0.045296 0.040650 0.054588
9: 0.051209 0.042674 0.064011 0.075391 0.040541 0.034535 0.043544 0.048048 0.042042
```

推测密钥长度为6，接下来找最多出现的字母，根据前一题的经验很可能并不是最多就对应着e，我尝试设置threshold，按照真实频率顺序依次枚举，最终枚举到 N 的时候有了结果，代码如下：

```python
for i in range(6):
    print(i, chr(np.array([s[i::6].count(chr(ord('A')+x)) for x in range(26)]).argmax()
+ ord('A')))
# 0 Q
# 1 K
# 2 L
# 3 T
# 4 H
# 5 C
# ETAOIN
password = [0, 0, 0, 0, 0, 0]
probable = [ord(x) - ord('A') for x in ['E', 'T', 'A', 'O', 'I', 'N']]
max_6 = [ord(x) - ord('A') for x in ['Q', 'K', 'L', 'T', 'H', 'C']]
tmp = np.zeros_like(t)
def dfs(depth):
    if depth == 6:
        test_s = ''
        for i in tmp:
            test_s += chr(i+ord('A'))
        if test(test_s) < 0.065:
            return
        for i in password:
            print('%c'%(chr(i+ord('A'))), end='')
        print(' : ', end='')
        for i in tmp:
            print('%c'%(chr(i+ord('A'))), end='')
        print()
        return
    for i in probable:
        password[depth] = (26 - i + max_6[depth]) % 26
        tmp[depth::6] = (t[depth::6] - max_6[depth] + i + 26) % 26
        dfs(depth + 1)
dfs(0)
```

使用之前的0.065来判断一个字符串是否是英文字符串，输出结果如下：

```
CRYPTO :
ILEARNEDHOWTOCALCULATETHEAMOUNTOFPAPERNEEDEDFORAROOMWHENIWASATSCHOOLYOUMULTIPLYTHESQUAR
EFOOTAGEOFTHEWALLSBYTHECUBICCONTENTSOFTHEFLOORANDCEILINGCOMBINEDANDDOUBLEITYOUTHENALLOW
HALFTHETOTALFOROPENINGSSUCHASWINDOWSANDDOORSTHENYOUALLOWTHEOTHERHALFFORMATCHINGTHEPATTE
RNTHENYOUDOUBLETHEWHOLETHINGAGAINTOGIVEAMARGINOFERRORANDTHENYOUORDERTHEPAPER
```

正常应该是 `I learned how to calculate the amount of paper needed for a room when I was at school. You multiply the square footage of the walls by the cubic contents of the floor and ceiling combined and double it. You then allow half the total for openings such as windows and doors. Then you allow the other half for matching the pattern. Then you double the whole thing again to give a margin of error and then you order the paper.`

1.21 c

进行字母频率统计，发现C最多，B第二多，猜测 C->e, B->t，解得明文=(密文×11+8) mod 26

算出来明文为 `ocanadaterredenosaieuxtonfrontestceintdefleuronsglorieuxcartonbrassaitporterlepeeilsaitporterlacroixtonhistoireestuneepopeedesplusbrillantsexploitsettavaleurdefoitrempeeprotegeranosfoyersetnosdroits`，找了找发现是加拿大国歌

1.21 d

词频统计看起来不像是直接置换，因此考虑维吉尼亚密码

```
s='BNVSNSIHQCEELSSKKYERIFJKXUMBGYKAMQLJTYAVFBKVTDVBPVVRJYYLAOKYMPQSCGDLFSRLLPROYGESEBUU
ALRWXMMASAZLGLEDFJBZAVVPXWICGJXASCBYEHOSNMULKCEAHTQOKMFLEBKFXLRRFDTZXCIWBJSICBGAWDVYDHA
VFJXZIBKCGJIWEAHTTOEWTUHKRQVVRGZBXYIREMMASCSPBHLHJMBLRFFJELHWEYLWISTFVVYEJCMHYUYRUFSFMG
ESIGRLWALSWMNUHSIMYYITCCQPZSICEHBCCMZFEGVJYOCDEMMPGHVAAUMELCMOEHVLTIPSUYILVGFLMVWDVYDBT
HFRAYISYSGKVSUUHYHGGCKTMBLRX'
for m in range(1, 10):
    print(m, end=': ')
    for i in range(m):
        print('%.6f' % test(s[i::m]), end=' ')
    print()
```

结果如下:

```
1: 0.041613
2: 0.044333 0.046382
3: 0.044645 0.048125 0.048387
4: 0.043468 0.057504 0.046517 0.047452
5: 0.045045 0.042523 0.041441 0.044058 0.037023
6: 0.051203 0.063458 0.054997 0.069804 0.055526 0.069804
7: 0.040531 0.044724 0.044267 0.037736 0.046444 0.031930 0.046444
8: 0.058279 0.059204 0.050879 0.048104 0.037928 0.064734 0.046377 0.050242
9: 0.048780 0.045296 0.041812 0.036005 0.040244 0.040244 0.042683 0.045122 0.062195
```

猜想还是密码长度为6，直接跑第二题代码，发现没有结果；修改threshold为0.6得到如下结果:

```
IHEORY :
TGREWUAAMONGDLOWTAWKERSMPNINPACTICULLRWHODCOPPEDHORDSAQEWATAEIMELIVEBEANDINAHIWLANDWSEN
IGOETOMINYEAPOLTSWHERPPEOPLPTOOKAWAKEWOMEGONCZMMATOXEANTHPENDOFLSTORYTCOULDYTSPEAVAWHOL
PSENTEYCEINCZMPANYLNDWASNONSIDPREDNOETOOBRTAHTSOTENROLWEDINADPEECHNOUQSEEAUGHTMYORVIWLE
SANOTHEFOFNDEROQREFLEIIVEREWAXOLORYASELQHYPNOEICTECSNIQUEEHATENLBLEDAAERSONEOSPEAVUPTOT
SREEHUYDREDWZRDSPECMINUTP
THEORY :
IGREWUPAMONGSLOWTALKERSMENINPARTICULARWHODROPPEDWORDSAFEWATATIMELIKEBEANSINAHILLANDWHEN
IGOTTOMINNEAPOLISWHEREPEOPLETOOKALAKEWOBEGONCOMMATOMEANTHEENDOFASTORYICOULDNTSPEAKAWHOL
ESENTENCEINCOMPANYANDWASCONSIDEREDNOTTOOBRIAHTSOIENROLLEDINASPEECHCOUQSETAUGHTBYORVILLE
SANDTHEFOUNDEROFREFLEXIVERELAXOLOGYASELFHYPNOTICTECHNIQUETHATENABLEDAPERSONTOSPEAKUPTOT
HREEHUNDREDWORDSPERMINUTE
MHEORY :
PGREWUWAMONGZLOWTASKERSMLNINPAYTICULHRWHODYOPPEDDORDSAMEWATAAIMELIREBEANZINAHISLANDWOEN
IGOATOMINUEAPOLPSWHERLPEOPLLTOOKASAKEWOIEGONCVMMATOTEANTHLENDOFHSTORYPCOULDUTSPEARAWHOL
LSENTEUCEINCVMPANYHNDWASJONSIDLREDNOATOOBRPAHTSOPENROLSEDINAZPEECHJOUQSEAAUGHTIYORVISLE
SANKTHEFOBNDEROMREFLEEIVERESAXOLONYASELMHYPNOAICTECONIQUEAHATENHBLEDAWERSONAOSPEARUPTOT
OREEHUUDREDWVRDSPEYMINUTL
```

感觉就是THEORY那行了，正常应该是

```
I grew up among slow talkers men in particular who dropped words a few at a time like
beans in a hill and when I got to Minneapolis where people took a lake wobegon comma to
mean the end of a story. I couldn't speak a whole sentence in company and was
considered not to obriaht so I enrolled in a speech couqse taught by orvilles and the
founder of reflexive relaxology a self hypnotic technique that enabled a person to
speak up to three hundred words per minute.
```

1.26

python解密代码如下：

```python
def decrypto(s, m, n):
    t=''
    for j in range(0, len(s), m*n):
        for i in range(n):
            t+=s[j:j+m*n][i::n]
    return t
decrypto('ctaropyghpry', 4, 3)
# 'cryptography'
```

经过寻找合适的m和n之后，破解结果如下：

```python
s='myamraruyiqtenctorahroywdsoyeouarrgdernogw'
decrypto(s, 3, 2)
# marymaryquitecontraryhowdoesyourgardengrow
```

正常应该是 `Mary, Mary, quite contrary. How does your garden grow?`