HW2 - Substitution Cipher Solver

105031212 吳紹齊

1. 程式執行環境

我使用 Python 3，以 Jupyter Notebook 開發，最後包成 solve.py，
經過測試，在 Python 3.6 下可以順利執行 python solve.py。

2. 程式流程說明

(1) 從 nc 140.114.77.172 60003 得到加密的 flag 與 本文。
(2) 將加密的本文與 flag 填入 essay_raw 與 flag。
(3) 執行 python solve.py。
(4) 螢幕依序輸出每個 trial 的 Score、解密的flag、解密的本文。

3. 解題過程（以 Jupyter Notebook 展示）

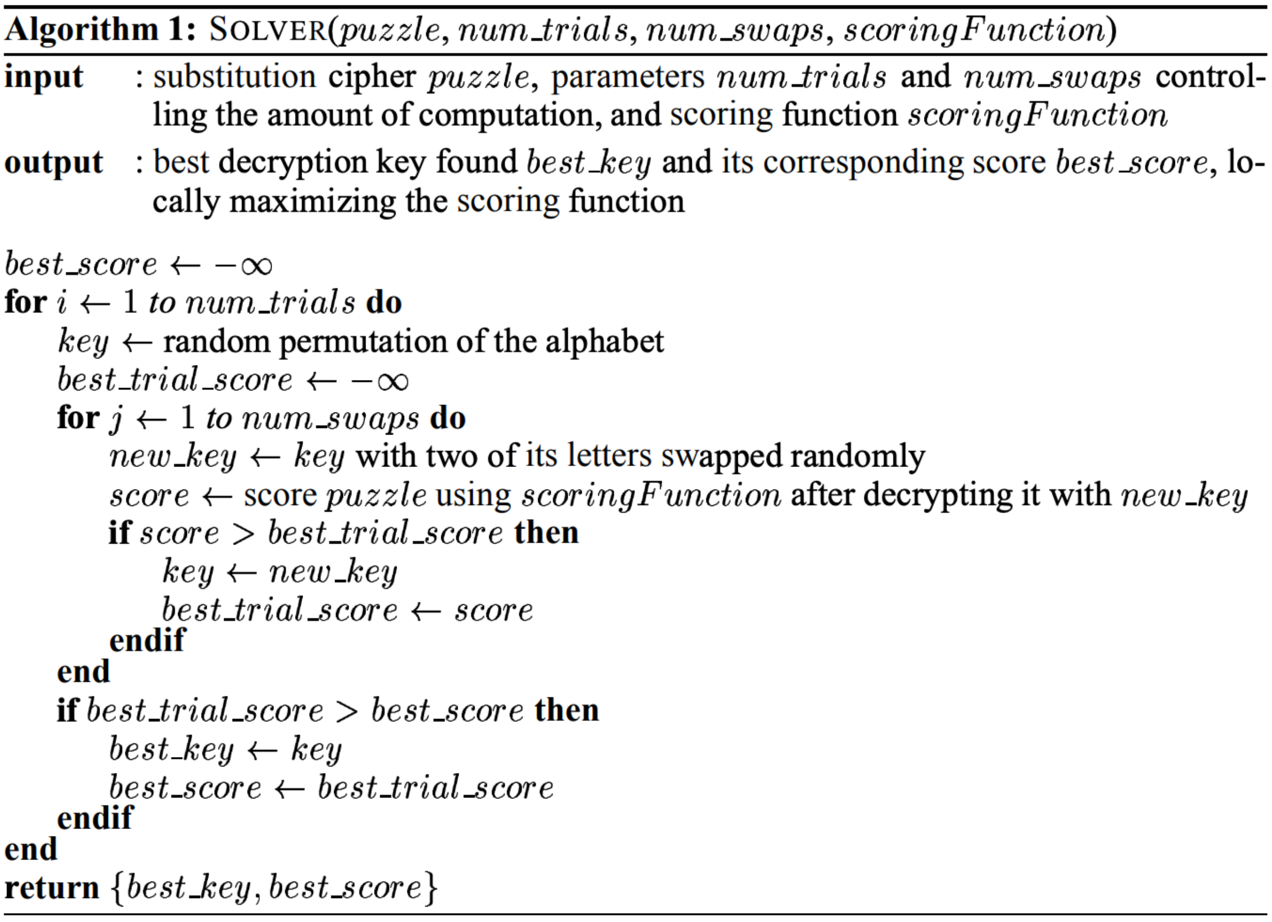
首先準備 replace 函數，用於最後把 A ~ Z 與對應的 KEY 置換，取得解密後的文字。

In [1]:
```
def replace(input1, input2, essay):
    output = ""
    for c in essay:
        for i in input1:
            if i == c:
                output = output + input2[input1.index(i)]
    #print(len(output))
    return output
```

接著準備 swap 函數，此函數可以隨機在KEY裡面置換兩個字母。

In [2]:
```
import random
def swap(key):
    new = ""
    a = 0
    b = 0
    while(a==b):
        a = random.randint(0,25)
        b = random.randint(0,25)
    for x in key:
        if key.index(x) == a:
            new += key[b]
        elif key.index(x) == b:
            new += key[a]
        else:
            new += x
    return new
```

再來是主要演算法，我參考一篇 Paper: Solving Substitution Ciphers。
https://people.csail.mit.edu/hasinoff/pubs/hasinoff-quipster-2003.pdf



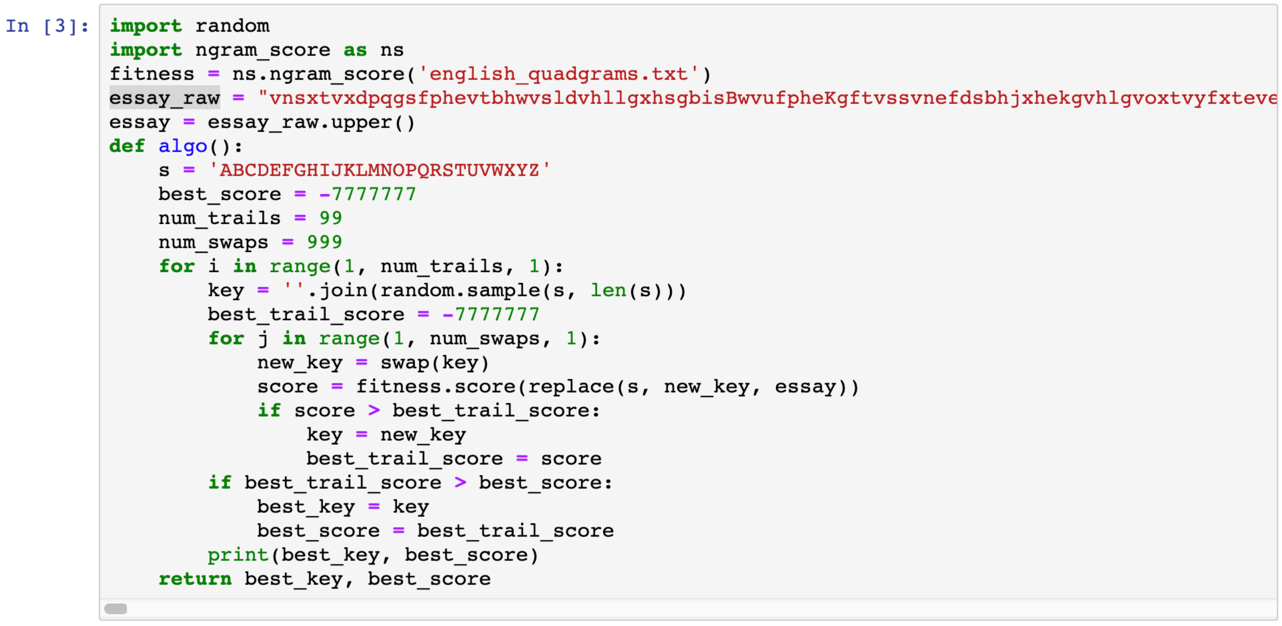
Algorithm 1: SOLVER($puzzle, num\_trials, num\_swaps, scoringFunction$)

input : substitution cipher $puzzle$, parameters num_trials and num_swaps controlling the amount of computation, and scoring function $scoringFunction$

output : best decryption key found $best\_key$ and its corresponding score $best\_score$, locally maximizing the scoring function

```
$best\_score \leftarrow -\infty$
for $i \leftarrow 1$ to $num\_trials$ do
    $key \leftarrow$ random permutation of the alphabet
    $best\_trial\_score \leftarrow -\infty$
    for $j \leftarrow 1$ to $num\_swaps$ do
        $new\_key \leftarrow key$ with two of its letters swapped randomly
        $score \leftarrow$ score $puzzle$ using $scoringFunction$ after decrypting it with $new\_key$
        if $score > best\_trial\_score$ then
            $key \leftarrow new\_key$
            $best\_trial\_score \leftarrow score$
        endif
    end
    if $best\_trial\_score > best\_score$ then
        $best\_key \leftarrow key$
        $best\_score \leftarrow best\_trial\_score$
    endif
end
return $\{best\_key, best\_score\}$
```

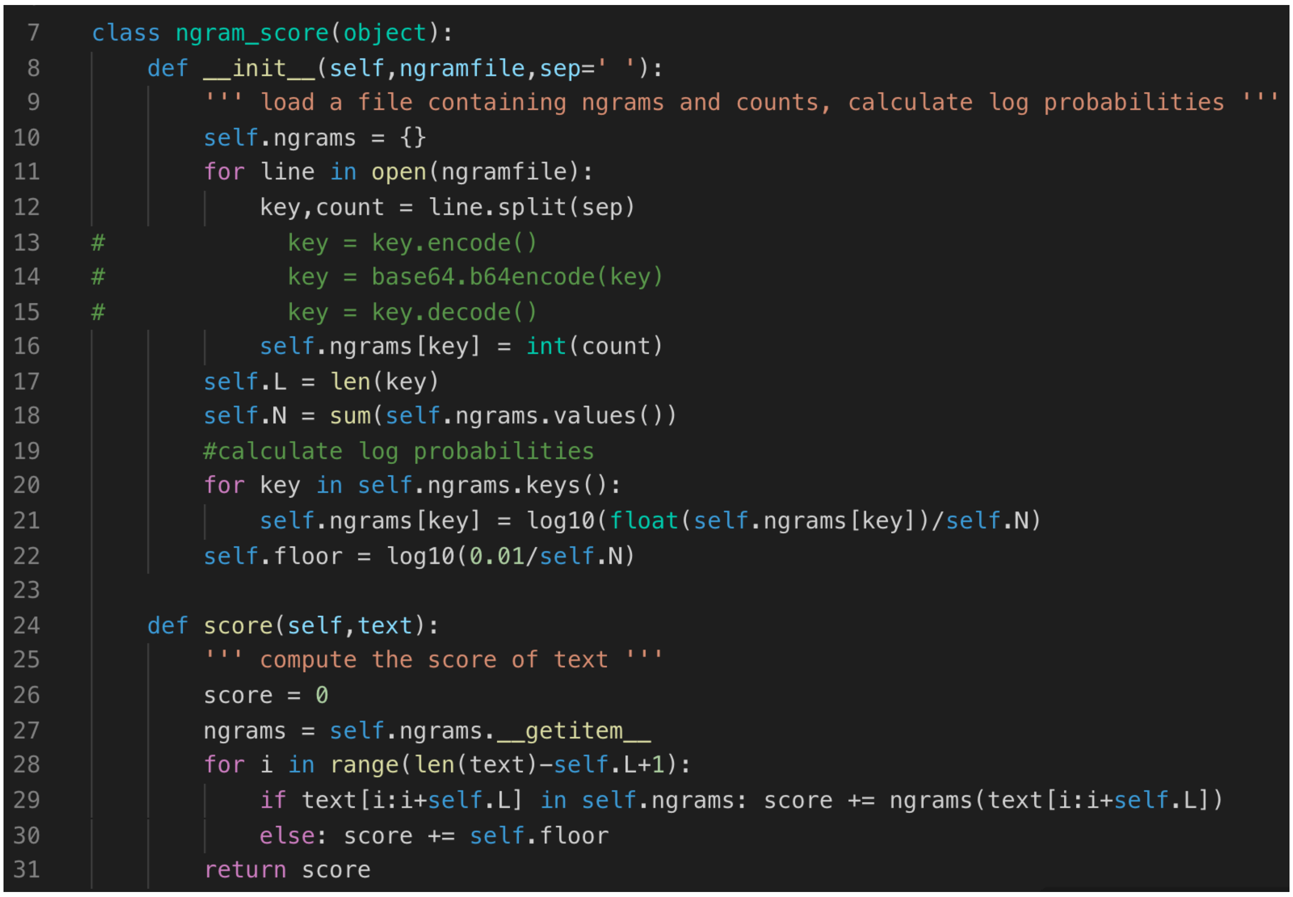
我仿照這個演算法刻出我的 Algorithm。



```
In [3]: import random
        import ngram_score as ns
        fitness = ns.ngram_score('english_quadgrams.txt')
        essay_raw = "vnsxtvxdpqgsfphevtbhwvsldvhllgxhsgbisBwvufpheKgftvssvnefdsbhjxhekgvhlgvoxtvyfxteve
        essay = essay_raw.upper()
        def algo():
            s = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
            best_score = -7777777
            num_trails = 99
            num_swaps = 999
            for i in range(1, num_trails, 1):
                key = ''.join(random.sample(s, len(s)))
                best_trail_score = -7777777
                for j in range(1, num_swaps, 1):
                    new_key = swap(key)
                    score = fitness.score(replace(s, new_key, essay))
                    if score > best_trail_score:
                        key = new_key
                        best_trail_score = score
                if best_trail_score > best_score:
                    best_key = key
                    best_score = best_trail_score
                print(best_key, best_score)
            return best_key, best_score
```

（演算法特色：多跑幾個 trail 可以避免陷入 local maxima。）

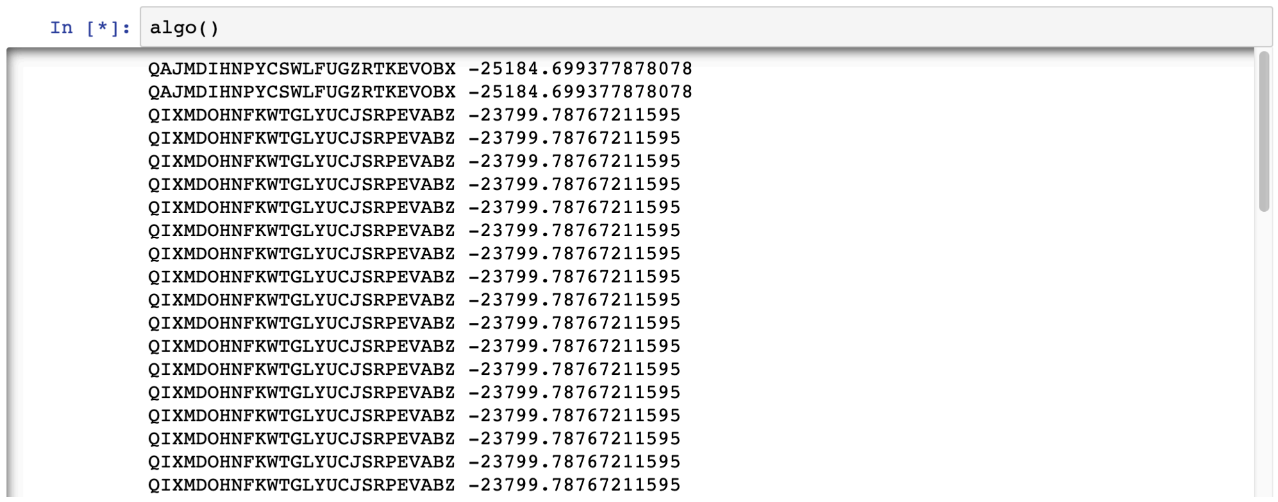
另外，score 的設計我參考一篇網頁文章: Quadgram Statistics as a Fitness Measure
http://practicalcryptography.com/cryptanalysis/text-characterisation/quadgrams/



```python
class ngram_score(object):
    def __init__(self,ngramfile,sep=' '):
        ''' load a file containing ngrams and counts, calculate log probabilities '''
        self.ngrams = {}
        for line in open(ngramfile):
            key,count = line.split(sep)
#             key = key.encode()
#             key = base64.b64encode(key)
#             key = key.decode()
            self.ngrams[key] = int(count)
        self.L = len(key)
        self.N = sum(self.ngrams.values())
        #calculate log probabilities
        for key in self.ngrams.keys():
            self.ngrams[key] = log10(float(self.ngrams[key])/self.N)
        self.floor = log10(0.01/self.N)

    def score(self,text):
        ''' compute the score of text '''
        score = 0
        ngrams = self.ngrams.__getitem__
        for i in range(len(text)-self.L+1):
            if text[i:i+self.L] in self.ngrams: score += ngrams(text[i:i+self.L])
            else: score += self.floor
        return score
```

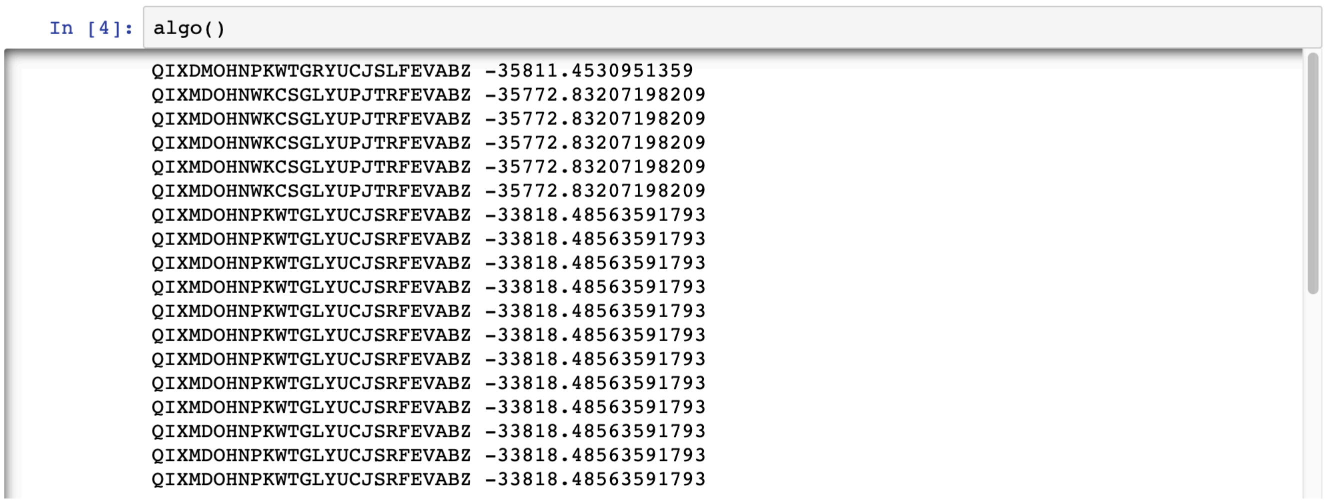
我分別用三個連續字母（trigrams）、四個連續字母(quadgrams)做完整實驗，
可以發現這次作業的 task 用 Trigrams 就可以很快猜到，不一定要 Quadgrams。

用 Trigrams 跑



```
In [*]: algo()

QAJMDIHNPYCSWLFUGZRTKEVOBX -25184.699377878078
QAJMDIHNPYCSWLFUGZRTKEVOBX -25184.699377878078
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
QIXMDOHNFKWTGLYUCJSRPEVABZ -23799.78767211595
```

用 Quadgrams 跑

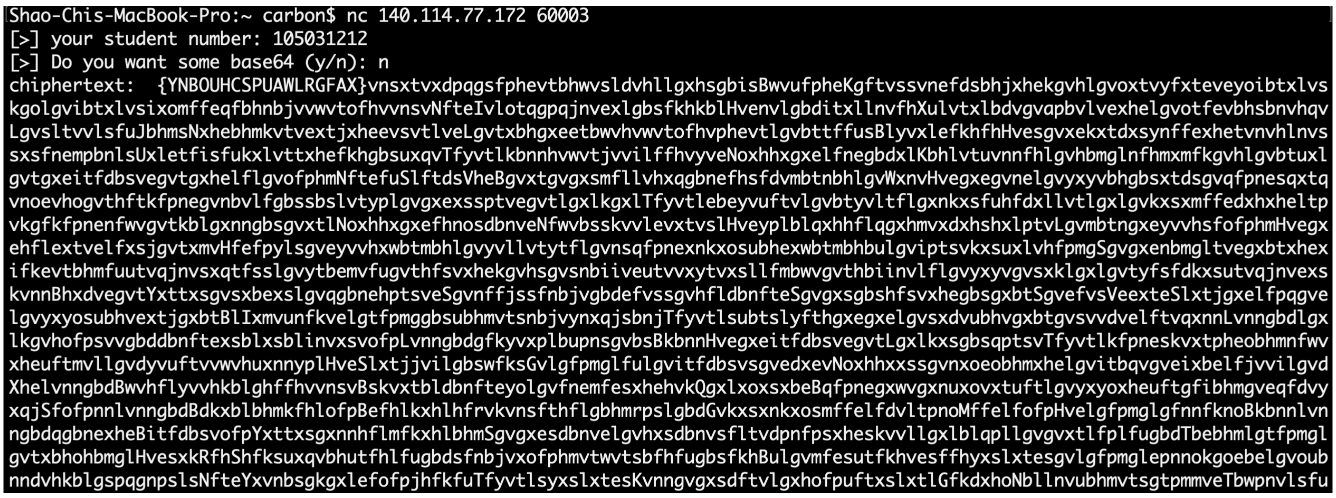


```
In [4]: algo()
```

```
QIXDMOHNPKWTGRYUCJSLFEVABZ -35811.4530951359
QIXMDOHNWKCSGLYUPJTRFEVABZ -35772.83207198209
QIXMDOHNWKCSGLYUPJTRFEVABZ -35772.83207198209
QIXMDOHNWKCSGLYUPJTRFEVABZ -35772.83207198209
QIXMDOHNWKCSGLYUPJTRFEVABZ -35772.83207198209
QIXMDOHNWKCSGLYUPJTRFEVABZ -35772.83207198209
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
QIXMDOHNPKWTGLYUCJSRFEVABZ -33818.48563591793
```

（備註：過程中不需要任何人工介入，分數會逐漸穩定，經過反覆測試都能在50個trials內完成。）

4. KEY、FLAG （以一組為例）



```
Shao-Chis-MacBook-Pro:~ carbon$ nc 140.114.77.172 60003
[>] your student number: 105031212
[>] Do you want some base64 (y/n): n
chiphertext:  {YNBOUHCSPUAWLRGFAX}vnsxtvxdpqgsfphevtbhwvsldvhllgxhsgbisBwvufpheKgftvssvnefdsbhjxhekgvhlgvoxtvyfxteveyoibtxlvs
kgolgvibtxlvsixomffeqfbhnbjvvwvtofhvvnsvNfteIvlotqgpqjnvexlgbsfkhkblHvenvlgbditxllnvfhXulvtxlbdvgvapbvlvexhelgvotfevbhsbnvhqv
LgvsltvvlsfuJbhmsNxhebhmkvtvextjxheevsvtlveLgvtxbhgxeetbwvhvwvtofhvphevtlgvbttffusBlyvxlefkhfhHvesgvxekxtdxsynffexhetvnvhlnvs
sxsfnempbnlsUxletfisfukxlvttxhefkhgbsuxqvTfyvtlkbnnhvwvtjvvilffhvyveNoxhhxgxelfnegbdxlKbhlvtuvnnfhlgvhbmglnfhmxmfkgvhlgvbtuxl
gvtgxeitfdbsvegvtgxhelflgvofphmNftefuSlftdsVheBgvxtgvgxsmfllvhxqgbnefhsfdvmbtnbhlgvWxnvHvegxegvnelgvyxyvbhgbsxtdsgvqfpnesqxtq
vnoevhogvthftkfpnegvnbvlfgbssbslvtyplgvgxexssptvegvtlgxlkgxlTfyvtlebeyvuftvlgvbtyvltflgxnkxsfuhfdxllvtlgxlgvkxsxmffedxhxheltp
vkgfkfpnenfwvgvtkblgxnngbsgvxtlNoxhhxgxefhnosdbnveNfwvbsskvvlevxtvslHveyplblqxhhflqgxhmvxdxhshxlptvLgvmbtngxeyvvhsfofphmHvegx
ehflextvelfxsjgvtxmvHfefpylsgveyvvhxwbtmbhlgvyvllvtytflgvnsqfpnexnkxosubhexwbtmbhbulgviptsvkxsuxlvhfpmgSgvgxenbmgltvegxbtxhex
ifkevtbhmfuutvqjnvsxqtfsslgvytbemvfugvthfsvxhekgvhsgvsnbiiveutvvxytvxsllfmbwvgvthbiinvlflgvyxyvgvsxklgxlgvtyfsfdkxsutvqjnvexs
kvnnBhxdvegvtYxttxsgvsxbexslgvqgbnehptsveSgvnffjssfnbjvgbdefvssgvhfldbnfteSgvgxsgbshfsvxhegbsgxbtSgvefvsVeexteSlxtjgxelfpqgve
lgvyxyosubhvextjgxbtBlIxmvunfkvelgtfpmggbsubhmvtsnbjvynxqjsbnjTfyvtlsubtslyfthgxegxelgvsxdvubhvgxbtgvsvvdvelftvqxnnLvnngbdlgx
lkgvhofpsvvgbddbnftexsblxsblinvxsvofpLvnngbdgfkyvxplbupnsgvbsBkbnnHvegxeitfdbsvegvtLgxlkxsgbsqptsvTfyvtlkfpneskvxtpheobhmnfwv
xheuftmvllgvdyvuftvvwvhuxnnyplHveSlxtjjvilgbswfksGvlgfpmglfulgvitfdbsvsgvedxevNoxhhxxssgvnxoeobhmxhelgvitbqvgveixbelfjvvilgvd
XhelvnngbdBwvhflyvvhkblghffhvvnsvBskvxtbldbnfteyolgvfnemfesxhehvkQgxlxoxsxbeBqfpnegxwvgxnuxovxtuftlgvyxyoxheuftgfibhmgveqfdvy
xqjSfofpnnlvnngbdBdkxblbhmkfhlofpBefhlkxhlhfrvkvnsfthflgbhmrpslgbdGvkxsxnkxosmffelfdvltpnoMffelfofpHvelgfpmglgfnnfknoBkbnnlvn
ngbdqgbnexheBitfdbsvofpYxttxsgxnnhflmfkxhlbhmSgvgxesdbnvelgvhxsdbnvsfltvdpnfpsxheskvvllgxlblqpllgvgvxtlfplfugbdTbebhmlgtfpmgl
gvtxbhohbmglHvesxkRfhShfksuxqvbhutfhlfugbdsfnbjvxofphmvtwvtsbfhfugbsfkhBulgvmfesutfkhvesffhyxslxtesgvlgfpmglepnnokgoebelgvoub
nndvhkblgspqgnpslsNfteYxvnbsgkgxlefofpjhfkfuTfyvtlsyxslxtesKvnngvgxsdftvlgxhofpuftxslxtlGfkdxhoNbllnvubhmvtsgtpmmveTbwpnvlsfu
```

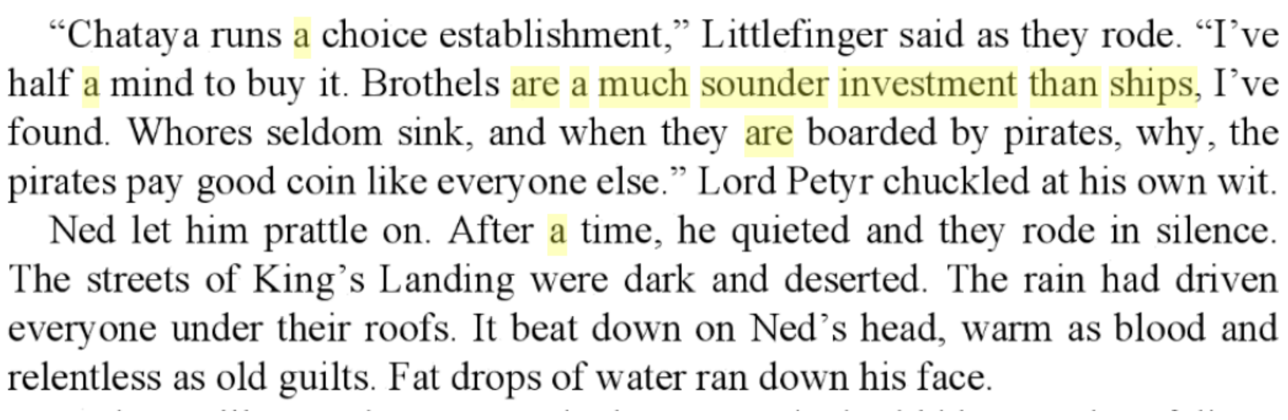
KEY: QIXMDOHNPKWTGLYUCJSRFEVABZ
FLAG: {BLIYFNXSUFQVTJHOQA}

```
In [5]: flag = "YNBOUHCSPUAWLRGFAX"
        print(replace("ABCDEFGHIJKLMNOPQRSTUVWXYZ", "QIXMDOHNPKWTGLYUCJSRFEVABZ", flag))

        BLIYFNXSUFQVTJHOQA
```

Content

```
In [6]: print(replace("ABCDEFGHIJKLMNOPQRSTUVWXYZ", "QIXMDOHNPKWTGLYUCJSRFEVABZ", essay))

        ELSAREAMUCHSOUNDERINVESTMENTTHANSHIPSIVEFOUNDWHORESSELDOMSINKANDWHENTHEYAREBOARDEDBYPIRATES
        WHYTHEPIRATESPAYGOODCOINLIKEEVERYONEELSELORDPETYRCHUCKLEDATHISOWNWITNEDLETHIMPRATTLEONAFTER
        ATIMEHEQUIETEDANDTHEYRODEINSILENCETHESTREETSOFKINGSLANDINGWEREDARKA[illegible]ESERTEDTHERAINHADDRIV
        ENEVERYONEUNDERTHEIRROOFSITBEATDOWNONNEDSHEADWARMASBLOODANDRELENTLE[illegible]SOLDGUILTSFATDROPSOFW
        ATERRANDOWNHISFACEROBERTWILLNEVERKEEPTOONEBEDLYANNAHADTOLDHIMATWINTERFELLONTHENIGHTLONGAGOW
        HENTHEIRFATHERHADPROMISEDHERHANDTOTHEYOUNGLORDOFSTORMSENDIHEARHEHASGOTTENACHILDONSOMEGIRLIN
        THEVALENEDHADHELDTHEBABEINHISARMSHECOULDSCARCELYDENYHERNORWOULDHELIETOHISSISTERBUTHEHADASSU
        REDHERTHATWHATROBERTDIDBEFORETHEIRBETROTHALWASOFNOMATTERTHATHEWASAGOODMANANDTRUEWHOWOULDLOV
        EHERWITHALLHISHEARTLYANNAHADONLYSMILEDLOVEISSWEETDEARESTNEDBUTITCANNOTCHANGEAMANSNATURETHEG
        IRLHADBEENSOYOUNGNEDHADNOTDAREDTOASKHERAGENODOUBTSHEDBEENAVIRGINTHEBETTERBROTHELSCOULDALWAY
        SFINDAVIRGINIFTHEPURSEWASFATENOUGHSHEHADLIGHTREDHAIRANDAPOWDERINGOFFRECKLESACROSSTHEBRIDGEO
        FHERNOSEANDWHENSHESLIPPEDFREEABREASTTOGIVEHERNIPPLETOTHEBABEHESAWTHATHERBOSOMWASFRECKLEDASW
        ELLINAMEDHERBARRASHESAIDASTHECHILDNURSEDSHELOOKSSOLIKEHIMDOESSHENOTMILORDSHEHASHISNOSEANDHI
        SHAIRSHEDOESEDDARDSTARKHADTOUCHEDTHEBABYSFINEDARKHAIRITPAGEFLOWEDTHROUGHHISFINGERSLIKEBLACK
        SILKROBERTSFIRSTBORNHADHADTHESAMEFINEHAIRHESEEMEDTORECALLTELLHIMTHATWHENYOUSEEHIMMILORDASIT
        ASITPLEASEYOUTELLHIMHOWBEAUTIFULSHEISIWILLNEDHADPROMISEDHERTHATWASHISCURSEROBERTWOULDSWEARU
        NDYINGLOVEANDFORGETTHEMBEFOREEVENFALLBUTNEDSTARKKEPTHISVOWSHETHOUGHTOFTHEPROMISESHEDMADELYA
        NNAASSHELAYDYINGANDTHEPRICEHEDPAIDTOKEEPTHEMANDTELLHIMIVENOTBEENWITHNOONEELSEISWEARITMILORD
        BYTHEOLDGODSANDNEWCHATAYASAIDICOULDHAVEHALFAYEARFORTHEBABYANDFORHOPINGHEDCOMEBACKSOYOULLTEL
        LHIMIMWAITINGWONTYOUIDONTWANTNOJEWELSORNOTHINGJUSTHIMHEWASALWAYSGOODTOMETRULYGOODTOYOUNEDTH
```

最後，我把內文丟到 Google ，發現是冰與火之歌的文本，因此可以驗證自己沒有錯。



"Chataya runs a choice establishment," Littlefinger said as they rode. "I've half a mind to buy it. Brothels are a much sounder investment than ships, I've found. Whores seldom sink, and when they are boarded by pirates, why, the pirates pay good coin like everyone else." Lord Petyr chuckled at his own wit.

Ned let him prattle on. After a time, he quieted and they rode in silence. The streets of King's Landing were dark and deserted. The rain had driven everyone under their roofs. It beat down on Ned's head, warm as blood and relentless as old guilts. Fat drops of water ran down his face.