Vigenere tores

Egy elképzelt nyelvben csak az angol ábécé nagybetűi szerepelnek. A nyelv minden szövege úgy épül fel, hogy blokkokból áll: minden blokk egymástól függetlenül

80% valószínűséggel egy olyan 5 hosszú blokk, melyben csak magánhangzók szerepelnek (A, E, I, O, U)

15% valószínűséggel egy olyan 7 hosszú blokk, melyben csak mássalhangzók szerepelnek,

5% valószínűséggel a következő sztring: "KORE".

Valósítsunk meg egy olyan törő algoritmust, mely a fenti nyelvnek egy legalább 200 hosszú szövegének Vigenere-titkosítását vissza tudja fejteni (jó eséllyel). Használhatjuk a 2-3. órán tekintett módszert, ahol a titkos szöveg ismétléseit kellett detekálni. A Vigenere-kulcs egy 3 és 15 közti hosszúságú lista véletlen shiftekkel.

In [1]:
```
import random
import math
import copy
```

In [2]:
```
# karakterek kodolasa
def indexOfLetter(c):
    return ord(c) - 65
def letterWithIndex(i):
    return chr(i+65)
def shiftChar(c, shift):
    i = indexOfLetter(c)
    new = (i + shift) % 26
    return letterWithIndex(new)
```

In [3]:
```
# kulcs eloallitasa
def generateKey():
    L = []
    random_number = random.randint(3, 15)
    for i in range (0,random_number):
        shift_number = random.randint(0, 26)
        L.append(shift_number)
    return L
```

In [4]:
```
# Vigenere titkosítás -> orai kod
def vigenere(s, shiftList):
    ret = ""
    # Current position in the shiftList
    pos = 0
    n = len(shiftList)
    for c in s:
        new = shiftChar(c, shiftList[pos])
        ret += new
        # Add one, restart if list is over
        pos = (pos + 1) % n
    return ret
```

In [5]:
```
# Valoszinuseghez szamgenerator
# Egy random szam generatorral szimulalom a nyelvben valo elofordulasi val
# 1 es 100 kozott generalok random szamot, es a hatarok reprezentaljak az
def blockProbability():
    L = ""
    rand_prob = random.randint(1, 100)
    if ((rand_prob > -1) and (rand_prob < 81)):
        for i in range (0,5):
            rand_char = random.randint(0,4)
            m = ['A', 'E', 'I', 'O', 'U']
            L += m[rand_char]
    elif ((rand_prob > 80) and (rand_prob < 96)):
        for i in range (0,7):
            m = ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O
            rand_char = random.randint(0, len(m)-1)
            L += m[rand_char]
    else:
        L += "KORE"
    return L
```

In [6]:
```
# Titkos szoveg eloallitasa
def encodingText (s, L):
    for i in range (0, len(s)):
        s[i] = vigenere(s[i], L)
        print(s[i])
```

In [7]:
```
# Ismetlodesek kiszurese
def repetitionsInString(s, length = 3):
    n = len(s)
    ret = [(i,j) for i in range(0, n-length) for j in range(i+1, n-length+1
    return ret
```

In [8]:
```
s = []
for i in range (0,100):
    s.append(blockProbability())
L = generateKey()
```

In [9]:
```
# egy hosszu stringet keszitek a szoveg blokkokbol
complete = ""
for i in range (0, len(s)):
    complete += s[i]
print(complete)
```

```
AEAEOUIOOOMWQCBHHAUAEUIOOIAIOIOUIOUEAEIAIOVFDOXMVUEUUOAOOEOUAEAOEOUAOOIEOIU
AUUUEIAOOOAOOUIEUUOUOUOIIUOIOMPTMLOWUUIEAIOOIAEIOIAIIEAOOIUAIAIIUOUEOUIIOOU
OOEEOIIOOIOEIAEOEEUEUAOOAEOUUUOKOREEUUAAWSKQTPNIOIUAKOREOIAIAAEEIAIEUEIOAII
OUAEEUOAUEUAIUAEAEEOIEUIIATRJTOMLPDWFNXRIEEIOOOIEOZGJNMPLAIOOUUUOOOIIUAETXH
XLMNUAAIUOAUIOUIIOOUIUUUEUUUUEAAIEAUAIAJXWYXYTKOREAUOUEAIAEIEOOAAOEOAIUUOAA
UOEUEEIOAUIAAEAOIUAIZSLLZGZOOUUEAEOEEOOUUOUUEEIEAEEIUUAOEIUAOEQZQFDNHAIAAEU
UEIAOOAUOEAUUUOIOUUAAOAOKOREIOEOIKOREIUUEOOOEOOVFOTMJMAIUAUIAOAIUAIEU
```

In [10]:
```
# titkositas ket fele modon

# 1 - blokkonkent titkositom a szoveget -> minden blokk karaktereihez adot
r = copy.deepcopy(s)
encodingText(r, L)
```

```
ULPJM
OPDTM
GDFHZDM
UBPJS
CVDNY
CVXTS
CVJJY
YPPNM
PMSTVIA
OLJZM
UVDJM
OHTFM
YVJFM
IPTTG
OHJZS
YPPTM
IHDTS
CLJZM
OVJTG
CBDNM
GWIRJKB
OBXJY
CVDNY
YPDNY
CPTFM
IPJFG
UPXZM
OLDZG
CVDZM
ILTTG
CVDNM
YPPJM
YLJJS
UVDFC
IBJZM
EVGJ
YBJFY
QZZVRLS
CVXZY
EVGJ
```

```
IPPNY
ULTNY
CLJJG
IHXNM
OHTJS
IHJJS
UPJFC
ULTTG
YBXNY
NYYYMIQ
JKLKLTW
CLTNM
IVXJM
TNYSKLQ
UPDTS
OBDTM
CPJFC
NEWCJIS
OHPNS
IHJNM
OPXTM
OPJZS
YBJZS
YHPNC
UBPNY
DELDVUY
EVGJ
UBDZC
UPPJG
YVDFY
ILDFG
OBDFY
OVTZC
YPDFS
CHPJY
IPJFG
TZAQXCE
IVJZC
ULDJC
IVJZM
OBTJG
YHTJG
OBPTC
CBPTC
KGFKBJM
UPPFC
OBTNY
IVPZM
YHJZS
IPDZS
UHDFM
EVGJ
CVTTG
EVGJ
CBJJM
IVTTM
PMDYKFR
UPJFS
CHDFG
OHXJS
```

In [11]:
```
# 2 - az egesz szovegen egyben vegig megyek, es karakterenkent kodolok
complete_text = copy.deepcopy(complete)
vigenere(complete_text, L)
```

Out[11]:
```
'ULPJMQNVXOGDFHZDMHDAYBXTMEFPXIIBXTSAFLRACVKKBKCTEUYBJTYKTLXUULPTCKZHXOCLDN
SWZBDECHDTMWTVDIYBJTSKZVRIOVXTKLYTUOQBJNCWNVXIULXTGWNPNAIVXZYEFPRUIBTTSENVX
UIVTJMENVXIILXFCKJLDEOHDTYATBDUIRDWCAZBJAQZZVRLSPXIOHZTPATPJIUHTJGWNLDECVPN
GKZHNEOVPZCQFPDAYHTJMEJBRIUAGORKRSYDQMCCPEJLROIVXJMVLQWMJSPNMKZBDOIVXNSWJAG
HRSBSSWFPDOUBXTSENVXUCBJZCQZBDEUHXJYQFPJJRDNCWPPVAEUBDZCWNHNIYVDFYKJVJIOBDF
YQTLDEYPDFSEFHNAIPJFGVXSUZAGDTSQJHNOYLDTSQTBDEYPTFCANBDAILXZYKJXIQZKCMYEFHN
UOLXFMKFBXEUBJZMETBDAUVPTIKWLROYVXPMNJPDUYVDTCKTCOONTYRYEZHDIUVPNSWNLD'
```

In [12]:
```
# ismerem a nyelvet, igy tudom, hogy vannak benne 4 hosszu KORE stringek, 
# a 4 hosszu stringeknel pontosan vissza tudom fejteni, hogy az elso 4 kar

four_char = repetitionsInString(complete_text, 7)
print(four_char)
```

```
[(77, 449), (143, 313)]
```

In [13]:
```
def guessedKeyLength(ciphertext, length = 3):
    differences = [j-i for (i,j) in repetitionsInString(ciphertext, length
    return gcd(differences)
```

In [14]:
```
# megprobalom a 4 hosszu ismetlodesekbol az eltolasokat meghatarozni
def KeyLengths(ciphertext, length = 3):
    differences = [j-i for (i,j) in repetitionsInString(ciphertext, length
    return differences
```

In [15]:
```
differences = KeyLengths(complete_text, 5)
print(differences)

def gcdOfKeys(array):
    if len(array) > 0:
        b=array[0]
        for i in range (0, len(array)):
            s=math.gcd(b,array[i])
            b=s
        return b
print(gcdOfKeys(differences))
```

```
[94, 7, 202, 372, 372, 372, 259, 170, 170, 170, 244, 109, 293, 236, 9]
1
```

In [ ]:
```
# az alabbi levezetesen lathato, hogy a kulcsmeret meghatarozasa sikeres v
# ha a 4-es ismetlodeseket keressuk a szovegben, ugy tobb esetet is talalu
# a 10-es periodicitas pedig ebben az esetben a kulcsunk hossza
```

In [31]:
```
m = ['A', 'E', 'I', 'O', 'U']
L2 = [3,2,1]
complete2 = copy.deepcopy(complete)
print(complete2)
complete2
complete2 = vigenere(complete2, L)

print(L)
print(complete2)
print(repetitionsInString(complete2, 4))
diff = KeyLengths(complete2, 4)
gcd_needed = gcdOfKeys(diff)

print(gcd_needed)
print(complete2[155:159])
print(complete2[315:319])
print(complete[155:159])
print(complete[315:319])

print()

print(complete2[76:80])
print(complete2[326:330])
print(complete[76:80])
print(complete[326:330])
```

```
AEAEOUIOOOMWQCBHHAUAEUIOOIAIOIOUIOUEAEIAIOVFDOXMVUEUUOAOOEOUAEAOEOUAOOIEOIU
AUUUEIAOOOAOOUIEUUOUOUOIIUOIOMPTMLOWUUIEAIOOIAEIOIAIIEAOOIUAIAIIUOUEOUIIOOU
OOEEOIIOOIOEIAEOEEUEUAOOAEOUUUOKOREEUUAAWSKQTPNIOIUAKOREOIAIAAEEIAIEUEIOAII
OUAEEUOAUEUAIUAEAEEOIEUIIATRJTOMLPDWFNXRIEEIOOOIEOZGJNMPLAIOOUUUOOOIIUAETXH
XLMNUAAIUOAUIOUIIOOUIUUUEUUUUEAAIEAUAIAJXWYXYTKOREAUOUEAIAEIEOOAAOEOAIUUOAA
UOEUEEIOAUIAAEAOIUAIZSLLZGZOOUUEAEOEEOOUUOUUEEIEAEEIUUAOEIUAOEQZQFDNHAIAAEU
UEIAOOAUOEAUUUOIOUUAAOAOKOREIOEOIKOREIUUEOOOEOOVFOTMJMAIUAUIAOAIUAIEU
[20, 7, 15, 5, 24, 22, 5, 7, 9, 0]
ULPJMQNVXOGDFHZDMHDAYBXTMEFPXIIBXTSAFLRACVKKBKCTEUYBJTYKTLXUULPTCKZHXOCLDNS
WZBDECHDTMWTVDIYBJTSKZVRIOVXTKLYTUOQBJNCWNVXIULXTGWNPNAIVXZYEFPRUIBTTSENVXU
IVTJMENVXIILXFCKJLDEOHDTYATBDUIRDWCAZBJAQZZVRLSPXIOHZTPATPJIUHTJGWNLDECVPNG
KZHNEOVPZCQFPDAYHTJMEJBRIUAGORKRSYDQMCCPEJLROIVXJMVLQWMJSPNMKZBDOIVXNSWJAGH
RSBSSWFPDOUBXTSENVXUCBJZCQZBDEUHXJYQFPJJRDNCWPPVAEUBDZCWNHNIYVDFYKJVJIOBDFY
QTLDEYPDFSEFHNAIPJFGVXSUZAGDTSQJHNOYLDTSQTBDEYPTFCANBDAILXZYKJXIQZKCMYEFHNU
OLXFMKFBXEUBJZMETBDAUVPTIKWLROYVXPMNJPDUYVDTCKTCOONTYRYEZHDIUVPNSWNLD
[(31, 311), (50, 90), (76, 326), (116, 156), (143, 313), (144, 314), (145,
155), (145, 315), (146, 316), (152, 242), (155, 315), (215, 515), (269, 289
), (378, 418), (385, 445), (402, 412)]
10
ENVX
ENVX
IIOO
IIOO

ZBDE
ZBDE
UUUE
UUUE
```

In [17]:
```
# statisztikai elofordulasok
LP = [0.8, 0.15, 0.05]
```

In [18]: