

K dešifrování zprávy využijí způsob popsany v první přednášce. Uděláme-li frekvenční analýzu bigramů podle <https://www.dcode.fr/bigrams>, zjistíme, že nejčastější bigram není ani zdaleka tak častý jako TH v anglické jazyce a tedy je to nejspíše mylné.

Uděláme-li však frekvenční analýzu trigramů podle <https://www.dcode.fr/trigrams>, uvidíme zajímavý výsledek, že frekvence **KPY** v šifrovaném textu velmi převyšuje ostatní pod ním a tedy je s velkou pravděpodobností využita k šifrování matice 3x3.

Nejčastější trigramy jsou **THE**, **AND** a **ING**, tedy v maticovém zápise to vypadá takto.

$$\left| \mathbf{A} \begin{pmatrix} 19 & 0 & 8 \\ 7 & 13 & 13 \\ 4 & 3 & 6 \end{pmatrix} \right|_{26} = \left| \begin{pmatrix} c_{10} & c_{20} & c_{30} \\ c_{11} & c_{21} & c_{31} \\ c_{12} & c_{22} & c_{32} \end{pmatrix} \right|_{26}$$

Pomocí tohoto vztahu spočítáme

$$\left| \mathbf{A} \right|_{26} = \left| \begin{pmatrix} c_{10} & c_{20} & c_{30} \\ c_{11} & c_{21} & c_{31} \\ c_{12} & c_{22} & c_{32} \end{pmatrix} \begin{pmatrix} 13 & 2 & 0 \\ 16 & 22 & 9 \\ 5 & 5 & 13 \end{pmatrix} \right|_{26}$$

Uděláme frekvenční analýzu bigramů šifrovaného textu, kdy předpokládáme, že $THE \rightarrow c_1$, $AND \rightarrow c_2$, $ING \rightarrow c_3$

Najdeme si nejčastější trojice např. pomocí této stránky <https://www.dcode.fr/trigrams>.

Následně už následuje rozumný bruteforce, přes všechny tyto trojice napsaný v jazyce Python.

```
from sympy import Matrix

most_frequent = ('KPY', 'BTI', 'LUV', 'QGO', 'DJX', 'SUC', 'ZZR', 'IHK', 'RPW', 'JCP',
                 'VHD', 'YVX', 'EQL', 'VCM', 'HAV', 'UEB', 'BBY', 'ZIX', 'QNN', 'VLA',
                 'FYJ', 'YEP', 'GKY', 'UKN', 'RZI', 'UUH', 'IDB', 'HYP', 'MLT', 'QQN')

text = 'UFQDNQCWJPXLNFNDSTSWHYSHDJXDJXHAVCMAIHDMIYHQNYOHEBCRZILJKLKZ'

P = Matrix([[19, 0, 8], [7, 13, 13], [4, 3, 6]]) # THE, AND, ING
P_inv = P.inv_mod(26)

def get_num(letter: str) -> int:
    return ord(letter) - ord('A')
def to_char(p: int) -> str:
    return chr(p + 65)

def decipher(c_matrix):
    A = c_matrix * P_inv
    A_inv = A.inv_mod(26)
    OT = ''
    for i in range(0, len(text), 3):
        c_vector = Matrix([get_num(text[i]), get_num(text[i+1]), get_num(text[i+2])])
        p_vector = (A_inv * c_vector) % 26
        OT += to_char(p_vector[0])
        OT += to_char(p_vector[1])
        OT += to_char(p_vector[2])
    if OT.count('E') > 5:
        print(OT, end = ' ')
    return
```

```

    raise ValueError

def hill():
    for c1 in most_frequent:
        for c2 in most_frequent:
            for c3 in most_frequent:
                if c1 == c2 or c1 == c3 or c2 == c3:
                    continue
                if c1 != 'KPY':
                    continue
                c_matrix = Matrix([ [get_num(c1[0]), get_num(c2[0]), get_num(c3[0])],
                                    [get_num(c1[1]), get_num(c2[1]), get_num(c3[1])],
                                    [get_num(c1[2]), get_num(c2[2]), get_num(c3[2])]
                                    ])

                try:
                    decipher(c_matrix)
                    print(f'THE -> {c1}, AND -> {c2}, EST -> {c3}')
                except ValueError:
                    pass

```

V rámci optimalizací zkusíme, že **c1** je vždy **KPY**, neboť je to z frekvenční analýzy téměř jisté. Též po dešifrování kusu textu spočítáme počet výskytu E, který by měl být nad nějakou mez (v našem případě jsem vybral 5 pro úsek dlouhý 60 znaků)

```

NPGWVFLXLHYOPWEINOPFVZRRBEEBEEWXCKAWQYXAOWNMXNXRDSWBPEVVAJY THE -> KPY, AND -> BTI,
    ING -> JCP
JRSOMDFEYNEBTUFUNDRAUTUWBIZBIZONLKIMGFSEEUWPDLEQTLEUEZKEBSRR THE -> KPY, AND -> LUV,
    ING -> YVX
JRSOMDFEUNEFTUFUNXRAYTUWBIVBIVONZKIEGFQEECWPLESTLIUEXKEPSRN THE -> KPY, AND -> LUV,
    ING -> MLT
LZQSCZUVMYNNEZHTQPEREWTGANDANDPSTISURICEMOPYFYTITPOOENEUDVMR THE -> KPY, AND -> DJX,
    ING -> YEP
PPKAVALXLHYBNWNENTRFZPRXLELLELEXFEAKAYREOEONJLNZTDWCBOVKOJN THE -> KPY, AND -> JCP,
    ING -> BTI
PREAMBNEBFYUNGUNQPADPUBNIKNIKNQIIIOFVAEMCPCXEBRLAQERYEDQRA THE -> KPY, AND -> YVX,
    ING -> LUV
HDGKKFZMLTWOVIECBOVEVDMRLYELYEOHCEOWGRXMMWOXMGXXPSYSPEIVFYF THE -> KPY, AND -> YVX,
    ING -> JCP
TRYIMPHEXLEJUMQNWZABVUFBICBICINGKISSFPUEIWPYPENBLYMEXIEBMRG THE -> KPY, AND -> YVX,
    ING -> HAV
BPEYVOVUBXBLBJTICDTIDHEBDBXDBXEODOUIADVUMKSPZIBVTAETECZQEGN THE -> KPY, AND -> YVX,
    ING -> VLA
RPCEIXODFEFUYJILWAQZXSRLYKEYXEVCUEMWFONCCABQMEXLFXUQENCABFCQ THE -> KPY, AND -> EQL,
    ING -> ZZR
LRISMJPEADEZRUPSNNJAIJUUVIDVIDANDYIUIFIOEWGPFHEKLLSSEJCEPQRB THE -> KPY, AND -> HAV,
    ING -> YVX
BRYYMPLEWHEDBUZGNBDACHUSTIBTIBANDUIQIFICEKSPREUFLMMEDYELURF THE -> KPY, AND -> HAV,
    ING -> MLT
HHGXFFMHLTOOVEECFOINVQFRLAELAEVCSWTCXMEWOLMARXXLSLVPRVVYPY THE -> KPY, AND -> UEB,
    ING -> JCP
RXMEYRUZVYJYBYHAEKLRSNDNEHSEHSNWEQGYVATQAORIGODVZJOGAZUSPLCY THE -> KPY, AND -> UEB,
    ING -> VHD
NFUWBHIKTKLGCTQXUYSIBMCZYTEYTENWEEEWNTGUIBOMSINHXYOXJIDNKEA THE -> KPY, AND -> VLA,
    ING -> VHD

```

```
RXKRLNAHPFOKLOAQZMEDVSDXXCCXCCSHECWSLETEKWUZYZZXTBSJRDZDJEXE THE -> KPY, AND -> YEP,  
    ING -> HYP  
PREAMBLETHEGNUGENERALPUBLICLICENSEISAFREECOPYLEFTLICENSEFORS THE -> KPY, AND -> MLT,  
    ING -> LUV  
VRMMMRNENFEMHUYINSVAZLUNRIKRIKINGQIISFPMEEEPCNEZXLWIEVOERGRQ THE -> KPY, AND -> MLT,  
    ING -> HAV
```

Takto vypadá výstup a vykoukáme, že správné mapování je $THE \rightarrow KPY$, $AND \rightarrow MLT$, $ING \rightarrow LUV$.

A zbytek textu už dešifrujeme obdobně, neboť mapování jsme už našli.

```
def get_text():  
    c_matrix = Matrix([ [get_num('K'), get_num('M'), get_num('L')],  
                        [get_num('P'), get_num('L'), get_num('U')],  
                        [get_num('Y'), get_num('T'), get_num('V')]  
                        ])  
  
    A = c_matrix * P_inv  
    A_inv = A.inv_mod(26)  
    with open('a.txt') as f:  
        while True:  
            c = f.read(3)  
            if not c:  
                break  
            c_vector = Matrix([get_num(c[0]), get_num(c[1]), get_num(c[2])])  
            p_vector = (A_inv * c_vector) % 26  
            print(to_char(p_vector[0]) + to_char(p_vector[1]) + to_char(p_vector[2]), end  
                  = '')
```
