# Vigenere cipher

## Task

The content of this file is encrypted with a *Vigenere cipher*. (The key length is limited to 20 letters)

Your task is to decrypt the content. Please note that you do not know the length of the key in this task!

Your solution will only be accepted if you find a better approach than simply trying all possible combinations for every possible key length (w.r.t. brute force) paired with a *simple* frequency analysis.

*A more intelligent appraoch will solve this task in roughtly quadratic time* $O=n^2$, *whereas trying all possible combinations, i.e. brute force, needs exponential time* $O=c^n \leftarrow c \gt 1$.

## Help

| | |
|---|---|
| **NOTE** | The values used in this example are artifical and not the real values. For a very short text, the solution is not accurate enough! |

1. you need to guess the key length, thus you know it is not longer than 20 letters, try every possible key length $2 \leq length \leq 20$
2. for every key length analyze the ciphertext statistically to derive the best plaintext
   a. you have to crack the key two letters at a time
   if the length is 3

```
VHEEMCRD (<- note that this is the plaintext for the example)
ABCABCAB
```

you observe that the letters VH, EM and RD are encrypted with the first two letters of the key

a. cut the ciphertext to strings with the size of the key length. each column is then encoded with one letter
   example:
   Suppose the key is **ABC**. The *ciphertext* **VIGENERE** should be cut into

```
VIG
ENE
RE
```

1. combining the first string with the second, the second with the third and so on. the last string should be combined with the first

this gives us:

```
01 01 01 (key_index)
--------
VI EN RE # letters enrypted with key[0] and key[1]
```

```
12 12    (key_index)
-----
IG NE    # letters enrypted with key[1] and key[2]
```

```
20 20    (key_index)
-----
GV EE    # letters enrypted with key[2] and key[0]
```

2. for each sequence of two consecutive letters try all possible two letters keys (AA, AB, AC, ..., ZY, ZZ)

```
VI EN RE decrypted with AA gives VI EN RE
VI EN RE decrypted with AB gives VH EM RD
...
VI EN RE decrypted with ZZ gives WJ FO SF
```

3. using the Bigram Frequencies (the frequencies are sorted alphabetically) give each key a score by multiplying the frequencies of the bigrams in the plaintext

```
 CIPHER  | KEY | PLAIN
--------------------------
VI EN RE | AB  | VH EM RD
--------------------------
score(AB) = frequency(VH) + frequency(EM) + frequency(RD)
```

**TIP**

- Normally, you would calculate the score by multiplication: `score(AB) = frequency(VH) * frequency(EM) * frequency(RD)`. However, with the given bigram frequencies, which are provided as logarithmic values, you can use an addition instead
- This means, the calculation using the numbers from the provided Bigram Frequencies is as follows: `score(AB) = frequency(VH) + frequency(EM) + frequency(RD)`
  - The higher the calculated value, the better is the score for that particular bigram pair

1. select for each string the key with the best score

```
 PLAIN   | KEY | K_index | SCORE
-----------------------------------
VH EM RD | AB  | 01      |151125528
DA IY    | FG  | 12      |119303088
EI CR    | CN  | 20      |122328401
```

2. form the final key from these keys

```
key[0] is either A or N, since score(AB) > score(CN), take A
key[1] is either B or F, since score(AB) > score(FG), take B
key[2] is either G or C, since score(CN) > score(FG), take C
=> the best key of length 3 is ABC
```

3. do this process for key lengths 2 to 20
4. using frequency analysis find the best fit of the 19 keys