Exercise 2.3: Cryptographic Analysis

In the following part of work we approach the n-gram distributions of English and German to crack the substitution cipher. The algorithm provides deciphering based on bigrams and trigrams distributions that were produced in *frequency_analysis.py*.

The substitution key: **nqjcgxyszuhvkboamitpefwlrd**

Explanation of the algorithm:

1. The algorithm starts with generating a random key, and incrementally improve it changing two chars in the step. Then the algorithm restarts. The number of restarts and steps in each restart is preliminarily defined.
2. **Improving the key.** The keys is judged based on the decryption. A new key is chosen closely to the previous key based on bigrams distribution by *neighboring_keys()*. Decrypted text is converted to 2-grams and yields keys by the following: one letter in each 2-gram is swapped to random letter so that a new bigram had higher probability.
3. **Optimization.** The local maximum of the fitness function refers to the most appropriate key among those produced in each restart. Local maximums in each restart are searching in the following way: if the value of the fitness function is better than in the previous step, the previous memorizes key is changed to the current one. This optimization is provided by *steepest_ascent()*.
4. **Decryption evaluation.** A decryption is evaluated based on trigrams distribution by *trigram_string_prob()*. The decryption is converted to 3-grams and the method calculates a value - sum of 3-grams' probabilities (more precisely, logarithms' of the probabilities). The larger the value, the better decryption. These values are the fitness values for *steepest_ascent()*.

5. **Final evaluation.** In *crack_ciphertext()* the local maximums are compared to each other, and the global maximum refers to the key which is returned by the method and used to decipher text.
6. Finally, decryptions for both languages are compared and the best one is chosen according to larger value of the fitness function.