BLAST

5.1 (E) BLAST
BLAST uses heuristics to speed up the search against databases. One of them is the usage of words, short sequences that have a high substitution score with the query sequence. Only sequences containing those words are considered for the alignment step. Additional improvements over the years include the two-hit method, gapped BLAST and finally PSI-BLAST.

5.2 (H) BLAST database
In the first task you will implement a python class called BlastDB that can contain many protein sequences. Besides the tested methods, you can implement any number of methods that you want to use during the BLAST search tasks. For the purpose of this exercise, all words are considered to be 3-letter words (i.e. $w = 3$).

(a) Complete the add_sequence() method for the BlastDB class in blast.py. The method takes a single string as input (a protein sequence) and adds it to the database structure. Note: The protein sequences might not be unique.

(b) Complete the get_sequences() method for the BlastDB class in blast.py. The method takes a single string as input (a 3-letter word) and returns a list of all sequences in the database that contain this word.

(c) Complete the get_db_stats() method for the BlastDB class in blast.py. The method takes no parameters and returns a tuple of four integer values. Each of those represents (in the following order) the number of sequences in the database, the number of different words within all sequences in the database, the average number of different words each sequence contains and the average number of sequences containing each word in the database (round the averages to the nearest integer).
5.3 (H) BLAST search

In the second task you will implement a python class called Blast that implements the original one-hit method from the 1990 BLAST version and the two-hit method from 1997 (neither are using gaps during the extension step). Both search methods will be provided with a BlastDB object as well as a query sequence or PSSM. In case of a query sequence, use the provided (during initialization) substitution matrix. Otherwise use the PSSM to calculate the scores for aligned residues. In the PSSM, each row represents one position in the query and each column one of the 20 amino acids (see exercise 4 for more details). HSP positions are 0-based (unlike the example in the slides).

(a) Complete the __init__() method for the Blast class in blast.py. The method takes a single matrix as input (a 20x20 substitution matrix).

(b) Complete the get_words() method for the Blast class in blast.py. The method takes as input either a query sequence or PSSM, as well as the word-score threshold $T$. It returns a list with all 3-letter words that have a minimum score of $T$ when aligned to the query. Each word may only be once in the list.

(c) Complete the search_one_hit() method for the Blast class in blast.py. It implements the original one-hit method from the 1990 BLAST. The method takes as input either a query sequence or PSSM, as well as a BlastDB containing the potential target sequences. In addition, the word-score threshold $T$, extension drop-off threshold $X$ and minimum HSP score $S$ can be provided. It returns a dictionary where the keys are the target sequences for which HSPs have been found and the corresponding values are lists of tuples. Each tuple is a HSP with the following elements (and order): the start position of the HSP in the query, the start position of the HSP in the target, the length of the HSP and the total score of the HSP. If multiple words generated the same HSP for a specific target, return only one of them.

(d) Complete the search_two_hit() method for the Blast class in blast.py. It implements the two-hit method from the 1997 BLAST. The method takes the same parameters as the search_one_hit() method, but it also includes the maximum distance $A$ for two hits on the same diagonal. The return type is identical to the previous search_one_hit().