Hazelyn Cates
11/27/22
EN.605.620.81.FA22

README

This program was written in Python 3.8 in the IDE PyCharm, version 2021.2. This program can also be run on the command line given that the input file is in the same file as the program.

This program finds the longest common substring (LCS) between two DNA sequences using a dynamic programming approach. The input file of the program must be a text file that contains the DNA sequences to be compared. The file(s) should be formatted in the following way:

S1 = ATGGATGGCTAG

S2 = CCGGAATTCGGAAAAGTTCGGATCCTGGTTTAGGC

…etc.

Where there is a newline between sequences and a single space before and after the "=" sign of each sequence.

This program does not print to the terminal, instead writeing the results to an output file.

This program implements two functions:

LC_substring(s1, s2, s1_len, s2_len, string_count1, string_count2), which takes six arguments: two strings, the lengths of each string, and the position of the strings in the file (that is, S1, S2, etc.). This function builds the dynamic programming (DP) table and calls the **find_LC_suffix** function, which is described below:

find_LC_suffix(LC_suffix, s1, s2, s1_len, s2_len, string_count1, string_count2, start_time), takes eight arguments: the DP table built in LC_substring, the two strings and their lengths, the position of the strings in the input file, and for timing, the start time of the function call for each pair of strings being compared, which began in **LC_substring**. This function finds the longest common suffix between a pair of strings (i.e. starts from the ends of the strings and works back) using the values in the DP table to determine where the strings match. To do this, it tests three conditions, and by traveling along the diagonal of values generated in the DP table by LC_substring and comparing the strings index by index determines where they match, allowing gaps between matching bases. This function also records the number of comparisons needed to find the LCS between the strings and calculates all possible substring combinations between the two strings, which is theoretical and irrespective of if they match at all.

This function ends by writing the results to the file: the strings that were compared and their lengths, the LCS and its length, how many comparisons were done to find that LCS, all possible substring combinations between the two strings, and the execution time of that function call using those two strings as input.