BCB 567 Fall 2017

Assignment 3: Construction of a Multiple Sequence Alignment

Due at 11:50 pm on Wednesday, November 29

Requirements

This assignment gives you an opportunity to produce a multiple sequence alignment without gaps from a file of DNA sequences in Fasta format by modifying the algorithm in Assignment 2 for constructing a superword array. You can write your program in any of the following programming languages: C, C++, Java, Perl and Python. You should include a README file in your submission that describes the components and structure of your program and shows the instructions for compiling and running your program at command line.

Your program should take the following arguments at command line: seqs_file word_model $wlcut$, where seqs_file is a file of multiple DNA sequences of any lengths in Fasta format, word_model is a file with a sequence of 1's and 0's on a single line, and wlcut is the maximum number of words in any superword (a positive integer). The word_model file and $wlcut$ are the same as in Assignment 2.

Your program reads the file of multiple DNA sequences in Fasta format and the other parameters, and saves the DNA sequences and their lengths in arrays as well as the other parameters in arrays and scalar variables. The program should be able to handle any positive number (k) of input DNA sequences of any lengths in Fasta format. The program processes the sequences as follows.

Concatenate all input DNA sequences into a combined sequence C with a pound sign (#) between adjacent sequences. Let n be the total length of the combined sequence including all occurrences of the pound sign. For $1 \leq i \leq k$, let $len[i]$ be the length of input DNA sequence i and let $st[i]$ be the start position of input DNA sequence i in the combined sequence. And for $1 \leq j \leq n$, let $id[j]$ be the index i if $st[i] \leq j \leq st[i] + len[i] - 1$ and be 0 otherwise. Note that all sequences positions and array indexes are 1-based.

Build a superword array SW for the combined sequence C with the value of -1 given to the code of the pound sign # and to the code of any non-regular base. For a superword at word level $wlcut$ starting at position j in sequence C with no component word code of -1, the superword comes from input sequence $i = id[j]$ and starts at position $j - st[i] + 1$ in sequence i.

Find each largest block of SW with the same superword code having no component code of -1 such that the block contains exact one superword from each input DNA sequence. Every one of these superword blocks denotes a local multiple sequence alignment of the length of the superwords in the block. Each superword block is represented by its superword in sequence 1.

Sort the superword blocks in an increasing order of the positions of their representatives. Quick Sort or Index Sort can be used here.

Build a longest chain of superword blocks with the following property using dynamic programming. For each input sequence, all superwords from the sequence in all the blocks of the chain are in an increasing order of their start positions. For any two adjacent superword blocks in the chain and for each sequence, the distance between the superwords from the sequence depends on those from sequence 1. Specifically, let h be the number of superword blocks in the chain. Let $j_1, j_2, ..., j_h$ be the starting indexes of these orderd superword blocks in the superword array SW. The start positions of the superwords from sequence 1 in those blocks are in an increasing order, as given in the following formula: for each g with $1 \leq g \leq h - 1$, we have $SW[j_g] < SW[j_{g+1}]$.

In addition, for each i with $2 \leq i \leq k$, the start positions of the superwords from sequence i in those blocks are also in an increasing order. And if the superwords from sequence 1 in adjacent superword blocks overlap with respect to their positions in sequence 1, then the superwords from sequence i in adjacent superword blocks have the same amount of overlaps. Otherwise, the superwords from sequence i in adjacent superword blocks have no overlap with respect to their positions in sequence i. Specifically, for each g with $1 \leq g \leq h - 1$, we have

$$SW[j_{g+1} + i - 1] - SW[j_g + i - 1] = SW[j_{g+1}] - SW[j_g]$$

if $SW[j_{g+1}] - SW[j_g] < m * wlcut$, or

$SW[j_{g+1} + i - 1] - SW[j_g + i - 1] \geq m * wlcut$ otherwise.

The length of a chain of superword blocks with the above property is the total number of base positions in sequence 1 covered by all superwords from sequence 1 in these superword blocks. In other words, the length of such a chain is the length of multiple sequence alignments constructed from this chain of superword blocks.

Your program reports, in this order on the stdout, the sequence of the word model (on a separate line), the value for the $wlcut$ parameter (on a separate line), the length of a longest chain of superword blocks (on a separate line), the number of superword blocks in the chain (on a separate line), each superword block (in the chain) in the sorted order. The output for each superword block consists of a blank line and k lines of superwords in their order in the block. The line for each superword includes the name of the input sequence (to which the superword belongs) in the first section of 10 characters (left adjusted), a space, the start position of the superword (in the input sequence) in the next section of 9 characters (left adjusted), and the sequence of the superword in the remaining section.

Submission

You are required to include, in your submission, each source code file. You should make your code efficient and non-redundant and include as many checks as possible to catch errors and avoid segmentation faults in execution. Note that the specification is subject to change. Please check on Bb for the latest clarifications.

Be sure to put down your name after the @author tag in each source file. Your zip file should be named Firstname_Lastname_hw3.zip, You may submit a draft version of your code early to see if you have any submission problem with Blackboard Learn. We will grade only your latest submission.