**BCB 567 Fall 2017**

# Assignment 2: Construction of A Superword Array

**Due at 11:50 pm on Wednesday, October 18**

## Requirements

This assignment gives you an opportunity to implement an algorithm for constructing a superword array for a DNA sequence in Fasta format. The algorithm is described in some of the tutorials in the Week 2-3 folders on Blackboard Learn (Bb). 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: seq_file word_model wlcut where seq_file is a file of a DNA sequence of length $n$ 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 sequence of 1's and 0's specifies a word model with each 1 bit denoting a checked position and each 0 bit denoting an unchecked position. Consider a word model of length 15 with 12 checked positions and 3 unchecked positions: 110110111110111. Two words of length 15 form a match if they have the same base at each of the checked positions. The code of a word of length 15 is the same as that of the string of length 12 produced by taking bases at each checked position from the word. If the string contains a non-regular base (other than A, C, G or T), then the code of the string is -1. Let $m$ be the length of the word model with $c$ checked positions and $u$ unchecked positions. Then $m = c + u$.

Your program reads the file of a DNA sequence in Fasta format and the other parameters, constructs a superword array $SW$ of the $n$ 1-based positions of the DNA sequence that is sorted based on a lexicographic order of the superwords at word level $wlcut$ starting at the positions, finds a first longest block of $SW$ such that the superwords starting at

the positions in the block have the same code with nonnegative component code at each rank, and produces the required output (see below).

Your program reports, in this order on the stdout, the sequence of the word model (on a line), the value for the $wlcut$ parameter (on a line), the number of positions in the block (on a line), each position in the block along with the superword at the position (on a line). The positions in the block should be reported according to their order in the block. Note that the superwords starting at each of these positions form matches with base mismatch allowed at the unchecked positions.

The algorithm discussed in lecture can be extended to handle superword matches with base mismatches. A word code array of length $n$ can be created to keep the codes of words of length $m$ starting at each position in the DNA sequence. The $SW$ array is constructed by using a look-up table. The look-up table consists of a string array of length $4^c$ (for all strings of length $c$) and a location array of length $n$. No additional arrays are needed by the algorithm.

## 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_hw2.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.