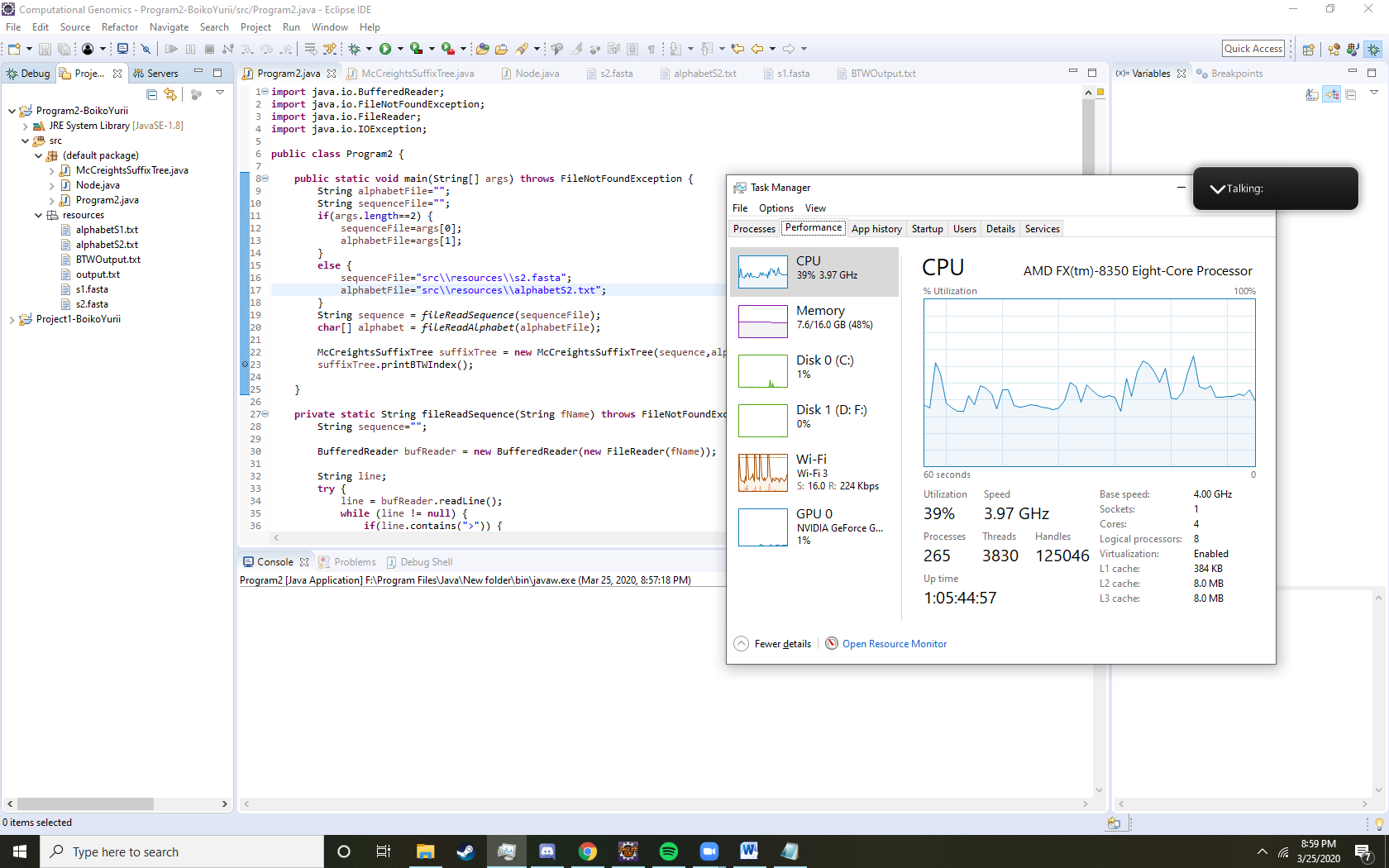
Yurii Boiko

Program 2 Report

1. System configuration:      CPU used, Clock rate, RAM, Cache size (if you know it).

Processor: AMD FX(tm)-8350 Eight-Core Processor (8 CPUs), ~4.0GHz

Memory: 16384MB RAM

1. Construction performance:        For all the inputs successfully tested, report the *running times for ST construction*(do not include other times such as I/O read time or print times). Time can be reported in seconds or milliseconds or microseconds. If you want to report any other performance statistics you are welcome to do so in your report.

string s1 (BANANA) Execution time in milliseconds: 5

string s2 (MISSISSIPPI) Execution time in milliseconds: 3

  Opsin gene in human (3,302 bp) Execution time in milliseconds: 121

Opsin gene in mouse (3,843 bp) Execution time in milliseconds: 135

Human BRCA2 gene (11,382 bp) Execution time in milliseconds: 1042

  Tomato's chloroplast genome (155,461 bp) Execution time in seconds: 109.48

1. Justification:    Do the performance observations made above, meet your expectations? Please explain.

No because I expected that as a sequence get larger than the runtime would be logarithmic while it looks like the time is exponentially growing.

Let’s compare Opsin gene in human that has 3302 characters and it took only 121 milliseconds to complete and the Human BRCA2 gene which is 11382 characters and it took 1042 milliseconds to run. The change in rate of characters 11382/3843=2.97 and change of rate in time is 1042/135=7.71 is exponential increased compare the Human BRCA2 gene to Tomato's chloroplast genome 155461/11,382=13.65 and 109480/1042=105.0. Because change in rate of characters from first to second increased by 13.65/2.97=4.60 and change of rate in time increased by 105.0/7.71=13.62

genome  with 155461bp which is

1. Implementation constant: Report your implementation's space constant for your code - i.e., how many bytes does your code consume for *every input byte?*  This can obviously be a roughly estimate (need not be exact) intended to give the end user an idea of your code's peak memory usage behavior.

Each node has 4 ints variable which are 4 bytes each and (2+alphabet.length+1) Node pointer variables which are 8 bytes and alphabet.length+1 char variables which are 2 bytes each. Best case scenario. A node with no parent and no children will be 16 bytes. Worst case scenario a node will be 16+8\*7+5\*2=82 bytes for alphabet of ACGT. This means that it has a parent pointer and 5 children pointer with suffix link node pointer and 5 char. For English alphabet worst case node would be 302 bytes. On average a node will be 16+82=49 bytes for alphabet of size 4 so let’s say this is the internal node. Leaf node will be closer to best case scenario because it doesn’t have any children but has parent pointer 16+8=24 bytes.

The Suffix tree itself has 3 int variables and 2 nodes internal nodes variables and the sequence variable and char array of alphabet variable. All this combine will come to 110+2(N+A) bytes where N is the length of the sequence and A length of alphabet.

For a BANANA string there were 4 internal nodes and 7 leaf nodes. The whole tree memory cost comes to be 4\*49+7\*24+110+2(6+26)=538 bytes. If we were to add one more letter to the sequence than we would add a new internal node and new leaf node and a char to the sequence therefore the memory cost would be 5\*49+8\*24+110+2(7+26)=613 bytes. This increases by 78 bytes

1. BWT index: . For this component, please output the BWT index for all the inputs, following [the example shown here for the string banana$](https://eecs.wsu.edu/~ananth/CptS571/Programs/Program2/BWT_from_ST_example.pdf) . Name the BWT file should denote which input - e.g., yeast\_BWT.txt for yeast. )

All the outputs are located in the output folder

1. Exact matching repeat:    For each of the input sequence, what is the *length*and the *<start> coordinates*of the longest exact matching repeat? The repeat occurrences may or may not be overlapping as long as they don't share the exact same start and end coordinates. For example, for sequence "banana" it should be "ana" starting at positions 2 and 4. In your report, also comment on how you found this repeat - i.e., your algorithm to find this repeat.

Didn’t implement this part of the project