**BIOMI 609 Computational Genomics and Bioinformatics**

**Spring 2022**

**Dr. Arun Sethuraman**

**Midterm Exam 1 - Key**

Q1. I have a few assemblies here that I compared using QUAST, including SPaDes, Velvet, and Unicycler, all compared against the reference genome of SARS-CoV2. Clearly Unicycler has produced the best assembly in this case.

|  |  |  |  |
| --- | --- | --- | --- |
| Genome Fraction | Velvet | SPaDEs | Unicycler |
| Genome fraction | 37.32% | 98.62% | 99.69% |
| Largest Alignment | 1416 | 7890 | 29809 |
| Total Alignment Length | 11268 | 29952 | 29809 |
| NGA50 (largest aligned contig length beyond which all contigs cover > 50% of the reference genome) | - | 7256 | 29809 |

Q2.

Burrow Wheeler Transform: CAAAAAAAAAAAAAATAACGGAATTAAGGGGCAAGACACAGAGGGGGCACGGCAGTCAAATAGGACGACCAGAACTGACGCTACGCGCGCCCGCTGGAGGCCGAGATTGCCCCACGCTAGTGGACCTACCAGAGATATATTATACCTGCAGGCCCGGACCATGGCTACTTCGTACGTACCACCCTATCCCCCGCCCCGCCACCAAGCCTACGTTACGCACCCGCGGGTCCAATCCAAAGACGTACCCGTCGCCATGCGCTCACGTCAGGACACCAAATACCGGCAAGCAACCGAAGTGCCTCCCCCCGGGATAACAGGCGAGCCCCCCGCGCGGGGCCCCATGGTCACCGGCTCACCCCAAGTCAGGAGGAGGCGGCCCCCGGCACATCACAGGATTCGTCGGATGGGCCGACCAAGGGGGCTCCCACGTTCAACGACTTCTGACACGGCCTTGGCTCAAACGCCGCTCCGCACGGCTCAGTGCAGTAGTGCACCCGCGGAGACCGAGCGCGTGCCCCTGACTGAGCACTGTCCACCCCCTCCTATAAACCGTCCACGATTTCAAGAGAATCCTGTTCTGGCCTCGGATCCCGTTCAGATCATGCCCCCAGCGCTCTCTACGGTCCACATACTTCACACTTTCACTCCCATCCATTGGGCCGGCATCTCCCCCCGCTCAGGGTCAGAACAACTCCCCCCTAAGTAGACTCCCGAATGGAAAGGGCGACCGTGGTGGGACGTCAGTGACGCCGTATAATTCACTCCGGCTACCGTAATACCCCACAGCTGCAACCGCAGAGTGCTTCTCCGTTTGCAGGGCAGCGCTCATTCACGGAAGGATCTTTGCTCGGTTGCCTCACCGCTGGTTCCCTCCCCCATCAGTATACCATTTTGCCGTCCCATCAACCTAGTTATAATCACAGTCTGGGGCAGCCCTCCATAGGCCGTCCATCCTTTCGACCCACCCGTGCCCGGACTACTGTTACAGTGCGCCGCCCCCGGAGTACCACCGGCCCCCATACAGACGGCGGAGCGACATAATCTACCTGTATCTCCTCACTCCTGCAGCTCGGAGTAACGCTGCCACTTTTCGCGCACGTTCTCGCTTCTCAACGTAACTCTTCCCACTGACGGCCCAGGTAGCGTACGAGTCTGTGACGCTAGAGACATGGTCCCTTACGACGGCTTAACACCGCAGCTGACACGATTCCCTTTTGAGTTGCAGGTCAACTTTTAAGTAGATGTAGTTGTTTGAATGGATGGGAGTAATATTTTAGTGCTGATAAAGGTCTAAAATGTAGAATGCAGCGCGCTGCGTGTTACAGGGGGTTAATTTTTGGGGTAACGGGACGCAACTTAGTGGTGGACAAGAGTGGTACTTACAATTAGTTATATGAGTAACATAGTTTTTGTTAATAGAAATAGTGACGTCTGGTTTGGTTAAGGGGCTTTAATATAATAGCGTTGGAAGAGCGAAGGTCAGGCTAGTGTGATAAGGATCACGAGTTGATGATAAGACCGTTTATCATAACTTAGTGTGAGCGATGGGGAAGTGATCCCGTTTGTTTATAGGTACGGCTTTTTGTAATTCTGTTAACAGCGTGGGCTAAAATGATTTTTTAGGGGAATATGATTATGTGATTACGATTAACAGGTTAAATTTTAATGGGGATGTGGTGTCATGCTATGTGTAGAGACGATTGTTACCTGAAGGTAGTGAGCAAGTATGCCTTTTTCATCGAGGGTGATGCAGTGGTTCTTTCTGAGTTAGCATACACTTGAGTTGCAAGAGGGTGGTTTACTTGTTAGAGTAGTAGATTTTGTTGAACGCCGCTTGTACCTTTCCGTCTGACGCCCGGTCTCCGGTACGGGTCATATGTTGATAACATCTAAGTCCGTCTGGACGCCCTCTTCTGCGTCCTGCACCGCCGATGTCGGTAATACCGCGCTGACTCCATCGCGCTCCCGGCTTCGCTACGCTTACAACCTTGCGCCTTCGACGCATGAGTGGAGTCCCTAGTAGAAGCCTACAGATGTTAGACATTACCTGTATTTCTAGCTTAGCACCTCGACCTACCGTGAAGAGGTGTCCAGATTACTCCTTCCGTCGTTGCCTAACCTATCCGCCTCTGCGTCACTACGATGGTATCGCAACGCGCTACCCCTAACCGCTCGTTCCCCCTCTAGCCGAAACTGGTGTTCATGGCCAGAACTTTCCCCTCCCCCCCTTCTCTGGTCGAATTAACTGCTAATAAGGTGTTAATCTCGCGTTCTTCGCTTCGTCTGATCCGCGCGCTCGTCTAACTAAGCGCCTTTTCATGCTCTTGCTCCTCTGGCTTTCTCTAGGCTGTTGGCTCGTAGATGCCTTCTTGTTTTTTTATACGTGTTTATCCCAATTCACAAGTTTCGCTAACTTTGCAACATTTCGGAGGGCACTTTCTTTTTTTTTTC

Suffix Array: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

Here’s the code I used to produce this (also find this as exam1.py in your Midterm1 folder).

#Midterm Exam 1 - BIOMI 609 - Spring 2022

#Solution - Dr. Arun Sethuraman

#Q2 - write a piece of code to read a reference genome FASTA, print its BWT, and SA

import numpy as np

import math

from Bio import SeqIO

#Read the FASTA file using Bio.SeqIO

for record in SeqIO.parse("referencegenome.fasta","fasta"):

ref = record.seq

#Creating an empty vector for all the rotations

t = []

t.append(ref)

x = ref

indices = []

#Looping all rotations

for i in range(len(ref)-1):

y = x[-1] + x[:-1]

x = y

t.append(x)

indices.append(i)

indices.append(i+1)

# Sorting all rotations

sortedt = sorted(t)

#Empty arrays for the suffix array, BWT

sa=''

bwt=''

#Constructing SA, BWT

for i in range(len(ref)):

x = sortedt[i]

sa+=x[0]

bwt+=x[-1]

#Printing them

print("Burrow Wheeler Transform:",bwt)

print("Suffix Array:",sa)