CSE 5370: Bioinformatics
Homework 2

Due Thursday, February 24th, 2022 at 4:59PM CST

In this homework you will write basic genome assemblers using two methods: greedy SCS and De Bruijn Graphs.

StackOverflow.com & Similar Sites

Use of stackoverflow.com and other sites is explicitly allowed (industry reserachers and academic labs use these sites frequently). However, for this course you must include a comment in your code with the link to the page you referenced whenever these sites influence your own code writing. For example, when writing this homework assignment I forgot how to insert code into LaTeXdocuments and recalled how to after visiting stackoverflow.com. If I were submitting this as an assignment, I would want to include a comment like the below example in my code submission:

```
%When writing this homework assignment, I did not recall how to
%insert code in a nice looking way into LaTeX documents,
%so I referred to this page on stackoverflow for help:
%https://stackoverflow.com/questions/3175105
\usepackage{minted}
\begin{minted}[mathescape, linenos]{python}
Code To Insert in \LaTeX...
```

Group Work

You are allowed to work together in groups of up to 4, but the assignment is structured so that every person must generate their own synthetic read pools with a random number generator (such that every submission will be unique). You may collaborate on discussing logic in the code. The automated grading script automatically renames all functions: function1, function2, function3 etc. and all variables: var1, var2, var3 etc. then standardizes all white space. Code from different groups should not be identical after this process and every person should have a different random set of reads and assemblies per the assignment specification.

1 Simulated Reads (25 points)

1. Write python code to generate a synthetic genome (a string) consisting of random nucleotide sequence of length 10,000.
2. Randomly generate a pool of synthetic "reads" (Pool A) of length 250 such that no reads overlap on the genome from 1.
3. Randomly generate a pool of synthetic "reads" (Pool B) such that every nucleotide in the genome from 1 is in a minimum of 4 reads and each read is guaranteed to overlap with another read by at least 7 nucleotides.
4. Randomly generate a pool of synthetic "reads" (Pool C) such that every nucleotide in the genome from 1 is in a minimum of 30 reads and each read is guaranteed to overlap with another read by at least 14 nucleotides.

Use a random number generate to produce these read pools and submit them as text files with the assignment. Also include in your submission a text files containing your synthetic genome. Include the code to generate these in your single code file.

2. so each read(sub
should be 250? 1000
or
total number of read
should be 250? 25

3. so sub-string of len-4??
or
total sub-string of 4??
and
sub-string overlap is 7 char

4. so sub-string of len
or
total sub-string of 4
and
sub-string overlap is 1

2 Greedy SCS (35 points)

Modify the below code such that it is greedy and run on your 3 files with min_length=9. Include three separate text files containing the output for running your greedy SCS implementation on each of the three read pools from section 1. Also include your code.

```
import itertools

def overlap(read_a, read_b, min_length=1):
    start = 0  # start all the way at the left
    while True:
        start = read_a.find(read_b[:min_length], start)
        if start == -1:
            return 0
        if read_b.startswith(a[start:]):
            return len(read_a)-start
        start += 1

def shortestCommonSuperstring(string_set):
    shortest_sup = None
    for perm in itertools.permutations(string_set):
        sup = perm[0]
        for i in range(len(string_set)-1):
            olen = overlap(perm[i], perm[i+1], min_length=1)
            sup += perm[i+1][olen:]
```

```
20          if shortest_sup is None or len(sup) < len(shortest_sup):
21              shortest_sup = sup
22      return shortest_sup
```

3 De Bruijn Graphs (40 points)

Write a de bruijn graph and run it on each of your three read pools from section 1 three times (9 total output files) with

$$k = 1, 9, 18$$

for each run. Include a text file with the assembly output in your submission as well as your code.