CPSC 2120/2121

Lab 12

DNA Sequencing
Due 11:59 Fri, Dec 7

Learning Objectives

- To improve our ability to implement algorithms and data structures in C++
- To use simple data structures to implement algorithms
- To become proficient in fundamental data structures used throughout computer science
- To improve our analytical skills while gaining familiarity with mathematical tools used in the analysis of algorithms
- To implement an algorithm and understand its intricacies

Problem/Exercise

For this lab, you will implement a technique to compare DNA sequences. You will be provided a file with DNA sequences in the format given below.

You are allowed to use any data structures in the C++ STL for this assignment. Your final output must be accessible through the interface provided in lab12.cpp, and must compile without errors using a makefile named Makefile (**not** provided). In order to thoroughly test your program, you will also submit a single input file named lab12.txt which has at least 3 test cases you have written to test your program.

NOTE - This should be your work. Copying existing code will be considered plagiarism and will result in academic integrity action being taken against you.

Additional Requirements

- You must include planning documents. While not required to be formal software engineering documents/diagrams, they must at minimum cover a description of the basic problem and discussion of what algorithm(s) you have chosen to use and why.
- In addition, there are several decisions which need to be made concerning things like insertion/deletion penalties. Your planning documents must give what values you are using and explain why.

DNA String Specification

The DNA strings used to test your program will be in the format below. The length of the sequences will not be given, and you may **not** assume sequences have the same length.

--human
GAAATCCAGAGGGCAGGGGTGGCCGGCACAGCAGACGTACCCTCCCTCGCTGCCTGCCTG
CGGCCTGCCCTGCATGCAGGATGGCCCTGAGGAAAGCGGGTGACCTCGAGCCTCAGTTCA
CACCTGAGCGCCGCTTCCGGCTGTGCTGGTATCAGGCTCACTCGGGCAGAGCCCTCCTCG
GGCCGCCTCCCCAAGCCTCTCCCCCCGCAGGAGGCCTGGCCCCGGCGCTGCTGCTGCTGT
CCTGGGTGGCACTGGGCCCCCGCAGCCTGGAGGGAGCAGACCCCGGAACGCCGGGGGAAG
CCGAGGGCCCAGCGTGCCCGGCCGCCTGTGTCTGCAGCTACGATGACGACGCGGATGAGC
TCAGCGTCTTCTGCAGCTCCAGGAACCTCACGCGCCTGCCTGACGGAGTCCCGGGCGGCA
CCCAAGCCCTGTGGCTGGACGGCAACAACCTCTCGTCCGTCCCCCCGGCAGCCTTCCAGA
ACCTCTCCAGCCTGGGCTTTCTCAACCTGCAGGGCGGCCAGCTGGGCAGCCTGGAGCCAC
AGGCGCTGCTGGGCCTAGAGAACCTGTGCCACCTGCACCTGGAGCGGAACCAGCTGCGCA
GCCTGGCACTCGGCACGTTTGCACACACGCCCGCGCTGGCCTCGCTCGGCCTCAGCAACA
ACCGTCTGAGCAGGCTGGAGGACGGGCTCTTCGAGGGCCTCGGCAGCCTCTGGGACCTCA
ACCTCGGCTGGAATAGCCTGGCGGTGCTCCCCGATGCGGCGTTCCGCGGCCTGGGCAGCC
TGCGCGAGCTGGTGCTGGCGGGCAACAGGCTGGCCTACCTGCAGCCCGCGCTCTTCAGCG
GCCTGGCCGAGCTCCGGGAGCTGGACCTGAGCAGGAACGCGCTGCGGGCCATCAAGGCAA
ACGTGTTCGTGCAGCTGCCCCGGCTCCAGAAACTCTACCTGGACCGCAACCTCATCGCTG
CCGTGGCCCCGGGCGCCTTCCTGGGCCTGAAGGCGCTGCGATGGCTGGACCTGTCCCACA
ACCGCGTGGCTGGCCTCCTGGAGGACACGTTCCCCGGTCTGCTGGGCCTGCGTGTGCTGC
GGCTGTCCCACAACGCCATCGCCAGCCTGCGGCCCCGCACCTTCAAGGACCTGCACTTCC
TGGAGGAGCTGCAGCTGGGCCACAACCGCATCCGGCAGCTGGCTGAGCGCAGCTTTGAGG
--monkey
GCCTGGGGCAGCTTGAGGTGCTCACGCTAGACCACAACCAGCTCCAGGAGGTCAAGGCGG
GCGCTTTCCTCGGCCTCACCAACGTGGCGGTCATGAACCTCTCTGGGAACTGTCTCCGGA
ACCTTCCGGAGCAGGTGTTCCGGGGCCTGGGCAAGCTGCACAGCCTGCACCTGGAGGGCA
GCTGCCTGGGACGCATCCGCCCGCACACCTTCACCGGCCTCTCGGGGCTCCGCCGACTCT
TCCTCAAGGACAACGGCCTCGTGGGCATTGAGGAGCAGAGCCTGTGGGGGCTGGCGGAGC
TGCTGGAGCTCGACCTGACCTCCAACCAGCTCACGCACCTGCCCCACCGCCTCTTCCAGG
GCCTGGGCAAGCTGGAGTACCTGCTGCTCTCCCGCAACCGCCTGGCAGAGCTGCCGGCGG
ACGCCCTGGGCCCCCTGCAGCGGGCCTTCTGGCTGGACGTCTCGCACAACCGCCTGGAGG
CATTGCCCAACAGCCTCTTGGCACCACTGGGGCGGCTGCGCTACCTCAGCCTCAGGAACA
ACTCACTGCGGACCTTCACGCCGCAGCCCCCGGGCCTGGAGCGCCTGTGGCTGGAGGGTA
ACCCCTGGGACTGTGGCTGCCCTCTCAAGGCGCTGCGGGACTTCGCCCTGCAGAACCCCA
GTGCTGTGCCCCGCTTCGTCCAGGCCATCTGTGAGGGGGACGATTGCCAGCCGCCCGCGT
ACACCTACAACAACATCACCTGTGCCAGCCCGCCCGAGGTCGTGGGGCTCGACCTGCGGG

Source Code Requirements

- Put a comment at the top of your source code file with your name (first and last), the date of your submission, your lab section, and the assignment's name.
- All functions should be commented. Use inline documentation as needed to explain ambiguous, tricky, or important parts of your code.
- All source code must follow good programming style standards such as properly indenting source code; and all variables, functions, and classes should be well-named.
- Your class must use dynamic memory allocation and deallocation properly, which means your class cannot contain a memory leak. You may use valgrind to check for memory leaks. Refer to the Unix manual and valgrind's online documentation for more information on valgrind.
- Sequences to be compared will be provided in a file called sequences.txt. Your program must read the input from the file.
- Your program's output should be the names of the species with the closest DNA sequences. Also output the "distance" between one of the closest pair and every other species. For example
    - INPUT: As above. Suppose for this problem there are human, monkey, dog, cat, fish, and fly *(these are the input names, DNA sequences not written here to save space)*
    - OUTPUT: human-monkey *(this line gives the closest pair. After this is the distance from all species to human--one of the closest pair. Could have been distance from monkey to all)*
    - human-monkey: 1.8
    - human-dog: 3.2
    - human-cat: 3.6
    - human-fish: 5.5
    - human-fly: 20.1
- Since you are choosing how to calculate the distance yourself, obviously your numbers may be different. Still, the pair listed as closest should have the smallest "distance".

Submission

Before the date/time stated on the CPSC 2121 Canvas webpage, you need to submit your code to handin under the correct lab assignment. Make sure to submit all of the following.

1. All source files required for this lab
2. Your testing file (lab12.txt)

After you submit, always double check that the file(s) you submitted were the correct version. To double check, download the submitted file(s), put them on one of our Unix machines, and make sure they compile and run correctly.

Grading

If your class does not compile on our Unix machines or your assignment was not submitted on time, you will receive a grade of 0 on this assignment. Otherwise, your class will be graded using the criteria below. This assignment will be graded on:

1. Your writeup, and the correctness/coherence of your justifications for each decision
2. The appropriateness of your algorithm/data structures given the decisions you made
3. The correctness of the implementation of the algorithm/data structure you chose
4. The length of time required for execution

You must test, test, and retest your code to make sure it compiles and runs correctly and efficiently on our Unix machines with any given set of valid inputs. We will only test your program with valid sets of inputs.