605.621 Foundations of Algorithms Fall 2022
Programming Assignment #1
Assigned with Module 1, Due at the end of Module 3 (September 19)

The goals of Programming Assignment 1 are: (1) to have you prepare your programming tools to support algorithm implementation, trace runs, and test runs to measure asymptotic behavior of your algorithm implementation, and (2) to exercise the algorithm analysis techniques you'll study in Modules 1 and 2. A firm foundation in these two goals will set you on towards successful assignment work this semester. Consult Course Modules |General Algorithm Links and/or |Advanced Algorithm Links and our text if you find notation you are unfamiliar with in this assignment or in the CLRS book.

This assignment requires you to "construct an algorithm". There are, of course, many algorithms solving this problem. Some algorithms are asymptotically more efficient than others. Your objectives are to (a) complete the assignment, and (b) submit an asymptotically efficient algorithm.

In this course, critical thinking and problem analysis results in discovering appropriate and better, the best, algorithm for a problem; defining the algorithm in pseudocode; demonstrating (we don't usually prove) that the algorithm is correct; and determining the asymptotic runtime for the algorithm using one or more of the tools for asymptotic analysis you have studied this semester.

This programming problem is not a collaborative assignment. You are required to follow the **Programming Assignment Guidelines** and **Pseudocode Restrictions** (Canvas page *Pseudocode Restrictions & Programming Guidelines*) when preparing your solutions to these problems.

Exercising the algorithm you design using one or more example data set(s) removes any doubt from the grader that the algorithm is structurally correct and all computations are correct. When a problem supplies data you are expected to use it to demonstrate that your algorithm is correct.

You are permitted to use Internet resources while solving problems, however complete references must be provided. Please follow the Sheridan Libraries' citation guidance at "*Citing Other Things* - HOW DO YOU CITE AN INTERVIEW, A TWEET, OR A PUBLIC WEB PAGE?" and continue to the APA Academic Writer *Sample References*. Additional example citations are provided at the Purdue University, Purdue Online Writing Lab (OWL), College of Liberal Arts *Reference List: Electronic Sources*. You can also use training provided by Victoria University Library, Melbourne Australia on *Harvard Referencing: Internet/websites*

Assignment continues on the next page ...

1. **Conditions**

   (a) Consider a set, P, of n points, $(x_1, y_1), ..., (x_n, y_n)$, in a two dimensional plane.

   (b) A metric for the distance between two points (x_i, y_i) and (x_j, y_j) in this plane is the Euclidean distance $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

2. **Closest Pairs** [85 points]

   (a) [40 points] Construct an algorithm for finding the $m \leq \binom{n}{2}$ closest pairs of points in P. Your algorithm inputs are P and m. Return the distances between the m closest pairs of points, including their x and y coordinates.

      i. [25 points] Define your algorithm using pseudocode. [See Programming Assignment Guidelines sections 4 and 5]

      ii. [15 points] Determine the worst-case running time (page 25) of your algorithm (call this the algorithm's worst-case running time).

   (b) [20 points] Implement your algorithm. Your code must have a reasonable, consistent, style and documentation. It must have appropriate data structures, modularity, and error checking. [See Programming Assignment Guidelines section 6]

   (c) [10 points] Perform and submit trace runs demonstrating the proper functioning of your code. [See Programming Assignment Guidelines section 7]

   (d) [10 points] Perform tests to measure the asymptotic behavior of your program (call this the code's worst-case running time). [See Programming Assignment Guidelines section 8]

   (e) [5 points] Analysis comparing your algorithm's worst-case running time to your code's worst-cast running time. Suggest you prepare a graph with *work* (running time) on the y-axis and n on the x-axis on which you plot your algorithm's worst-case running time and your code's worst-case running time.

3. **Retrospection** [10 points]

   (a) [10 points] Now that you have designed, implemented, and tested your algorithm, what aspects of your algorithm and/or code could change and reduce the worst-case running time of your algorithm? Be specific in your response to this question.

4. **Package your materials for submission** [5 points]

   Consult Programming Assignment Guidelines section 1. Any material uploaded to Canvas outside of a <**lastname**><**assignment#**>**.zip** or <**lastname**><**assignment#**>**.tar.gz** file will not be graded. This strict rule is enforced because the Instructor bulk downloads all student submissions at one time. The Instructor cannot guarantee correct receipt of multiple individual files using Canvas' bulk assignment download.