AUIS ENGR 244 (Engineering Computing), Assignment 3

Deadline:[1] March 28, 2018; Tutorial discussion: April 9, 2018

1) Write a program that asks the user for an integer number *n* and a real number *s*, generates *n* uniformly distributed **random numbers**[2] between 0.0 and *s*, and **writes them to a file.**

2) Write a program that solves the **travelling salesman problem** for small values of *n* as follows:

- The user is asked for an integer number *n*.
- The program reads *n* two-dimensional Cartesian coordinates (x_i, y_i) from a file.[3]
- The program determines the length of the shortest path connecting all coordinates.
- The program output is the length of the shortest path.

The path must begin and end at the same point.

Note that while this is a "hard problem" for a great values of *n*, it is feasible for small values of *n*.

Do experiments on the time that it takes your code to determine the shortest path, **measuring the time** elapsed between the end of the user input and the output of the result by hand, with a stopwatch. For some values of *n* (e.g., $n = 4$), the program will always be faster than you can measure, and from a certain problem size onward, it will always take too long to terminate.

Determine the range of values of *n* accessible to this method of time measurement, and for each value of *n* within this range, run five experiments (giving different coordinates each time), and plot the smallest, the greatest, and the average runtime of your code as a function of *n*.

One extra credit: Submission with the greatest value of n at an average runtime below one minute.[4]

3) **Write a program** that accomplishes the following task:

- The user is asked for an integer number *n* and a real number *d* (i.e., a sphere diameter).
- The program reads *n* coordinates in 3D space (i.e., the centers of *n* spheres) from a file.[3]
- The program determines if there is a pair of coordinates closer to each other than *d*. (Whenever such a pair exists, there is an overlap between at least two of the spheres.)

How do the time requirements scale? Discuss by **analysing your code** (not by measuring time).

By trial and error determine experimentally how great *n* can become for your code without taking more than a minute average runtime on your computer.

One extra credit: Submission with the greatest value of n at an average runtime below one minute.[4]

[1] *Submissions (paper only –* **print your code**, *do not submit handwritten code), as single work or done by groups of two people, can be handed in on* **Wednesday, March 28**, *at lecture time, or deposited in the mailbox (room B-F2-01) by Tuesday, Mar 27. Each problem contributes two credits. Total credits from assignments may not exceed 15.*

[2] *Make yourself familiar with the method Math.random( ), which returns a random value between 0.0 and 1.0.*

[3] *Use the program that you wrote for* **problem 1** *in order to generate the required file.*

[4] *Only if the result is correct and the performance can be successfully* **demonstrated at the tutorial session** *(April 9). Since the performance depends on the hardware, moderate deviations from the reported runtimes will be tolerated. In cases of doubt, the limit is two minutes on the presentation laptop used for the lectures (details can be provided).*