CSCI 3320: Data Structures
HW 3: Implementing and Analyzing Sorting Algorithms
Due: 11/6 (Mon) 11:59 pm (8 points)

1. Project Details

In this project, you will implement diverse sorting algorithms and measure the performance of each sorting algorithm using different input data. Your program will generate random, sorted, and reverse-order numbers and sort them using each sorting algorithm. Your program receives 3 parameters from the users via the **command line**.

1. `input`: The first parameter indicates how to generate number as follow:
   1 – Totally random numbers
   2 – Sorted numbers (0 ~ `cnt`)
   3 – Reverse-order numbers (`cnt` ~ 0)
2. `cnt`: The second parameter indicates how many numbers are generated. The range of generated numbers is 0 ~ `cnt`.
3. `sorting`: The third parameter indicates a sorting algorithm to sort the generated input data. You can use an integer value to determine a sorting algorithm as follows:
   1 – Bubble sort
   2 – Selection sort
   3 – Insertion sort
   4 – Shell sort
   5 – Merge sort
   6 – Quick sort

Your program **must** take these parameters from **command line arguments** as follows:

- > python3 SortingArray.py **1 500 1** or java SortingArray **1 500 1**
  > 500 random numbers are generated and sorted using a Bubble sort algorithm.
- > python3 SortingArray.py **3 100000 6** or java SortingArray **3 100000 6**
  > 100000 reverse-order numbers are generated and sorted using a Quick sort algorithm.
- >python3 SortingArray.py **2 200000 4** or java SortingArray **2 200000 4**
  > 200000 sorted numbers are generated and sorted using a Shell sort algorithm.

I will test your program using a command line as shown above. Thus, you must not make a menu interface to get parameters.

2. Implementation Details

A skeleton code (**SortingArray.py** and **SortingArray.java**) will be provided. Please add comments to your code for the instructor to understand your code. You can use the sorting algorithm code in the slides. You must handle all the exceptions. You must not use any built-in sort functions in this project. Please put your information in the code.

To compare diverse sorting algorithms' performance, **you will measure elapsed time for each execution** using the Python function `time.time()` or Java function `System.nanoTime()`. You need to measure execution time for <u>**sorting algorithms only**</u>, i.e., execution time for generating input data must not be included in elapsed time. Finally, you need to draw graphs (plots) to show results for each sorting algorithm using various large input data sizes, e.g., 10000, 500000, and 2000000 (or any different size). For each input size, you must draw a graph with varying types of input, i.e., **random**, **sorted**, and **reverse order,** for each sorting algorithm. For the performance comparisons, **you must analyze the results and write your observations (performance) for each sorting algorithm.**

You can discuss this project with your: classmates to get some hints. However, **it is not allowed to 1) share the solution with your classmates, 2) post this project to websites e.g., stackoverflow.com and chegg.com to get solutions, and 3) just copy and paste results (performance comparisons) you found on the Internet or other references.** In short, <u>**please complete the project by yourself**</u>.

3. Deliverables

Please submit your code and report for performance measurement on Canvas by due. Please put your information in the code.

4. Grading

The following factors will be considered while grading your submission:

- Code correctness: 50%
- Performance measurement report: 50%