New **Y**ork **U**niversity
Computer Science Department
Data Structures
Dr. Anasse Bari

Homework Three: Practical Application to Sorting

Deadline: *See NYUclasses for the deadline, 15% off per day after the deadline (4 days maximum).*

Learning Objectives:

- Observing and analyzing the runtime difference of various sorting algorithms.
- Determining the effect of various pivoting strategies.

Read the guidelines bellow carefully to avoid receiving a zero grade on the HW:

- Attach the Java source files and include them into the HW's zip file. The file name should be **YourLastName_HW3.zip**
- Make an archive (zip file or compressed file) with all the **java files (the .java files NOT the .class files)** and post it on NYU Classes
- You must comment you code (basic comments explaining the role of a class, a method or variables used in your submission)
- Compile and run the program before you submit.
- It is your responsibility to make sure the Zip files has your actual latest files. You may send the file to yourself by email to double check that it is the actual file before you upload on NYU classes.
- **If the graders cannot open the file, you will receive a grade of zero.**
- **If you send the .class files instead of the .java files (source files) you will receive a zero.**
- Any act of cheating will be severely addressed with an immediate zero on the homework and a report to the academic advisor and the administration.
- You will automatically lose 50% of the points for an exercise if the program does not compile and run correctly.
- **Plagiarized assignments will get a ZERO grade**. You cannot change the variable names of other student's solution and submit it as yours. The program structure of other students must not match yours. Every student must come up with his/her own solution. Any cheating (e.g. copying from internet without citing sources) is a serious violation of the University student code.
- **Homeworks sent by *email* to the instructor or to the graders will NOT be reviewed and will not be accepted.**

Instructions: Submit your code (*.java code not the *.class files) and include a word document that has your answers to the exercises that do not require programming. Please organize your answers very clearly.

Maximum Number of Points: 100pts

Consider the attached public dataset about the US Census Bureau Population and Housing Unit Estimates.

You are hired as a *software engineer* to build a simple program that will do the following:

(1) **calculate the percent increase or decrease in population size per state and per region (United States, Northeast, Midwest, South and West) and per state.**

(2) to sort the percent increase (percentage of population change per state and per region) from the highest to the lowest.

This could be a repetitive process for the end user, as the population size could change every day depending on the input/data error. Therefore, you will need to design the program to use the fastest sorting algorithm possible to sort the percent increase in population sizes.

In this homework you should empirically experiment sorting algorithms: Merge Sort, Bubble sort, Plain QuickSort and "your" designed version of an enhanced QuickSort (refer to the discussion we had in class on the last slides from chapter 5. You should also refer to the book on the sorting chapter).

Your program shall be able to do the following:

- Display the sorted percent change of all states between year X and year Y using a sorted algorithm, where the years X and Y are given as user input form the user and the name of the sorting algorithm is given as input from the user (see 3)
- Display the most similar state to state X in terms of percent change in population for a given year
- (3) Calculate the running time for your algorithm using System and the time function in Java for
    - Merge Sort
    - Bubble Sort
    - Plain QuickSort
    - **Improved QuickSort** (comment the QuickSort algorithm and how you have improved it. I recommend looking at the official implementation found in Java for some inspiration)
- Ignore the grey portion in the Excel file and feel free to make changes to the Excel sheet as you see best to make it easy to read using Java and populate an array of State objects.
- Report on a word document the running time for one example of two years of your choice using the different sorting algorithms. Your improved quicksort should have the lowest running time using the *Java.lang.System.currentTimeMillis()*.