[CS302-Data Structures]

Homework 3: Sorting Algorithms

Instructor: Erin Keith

Teaching Assistants: Mustafa Solmaz, Huan Nguyen

Spring 2020 Semester

The goal of this programming assignment is twofold. In particular, it relates to you comparing the relative performance of different sorting algorithms on the same dataset and the same algorithm on different datasets.

First choose **TWO** sorting algorithms from the following list:

* selection sort
* insertion sort
* bubble sort

and **ONE** algorithm from the following list:

* merge sort
* quick sort

Case 1: Randomly generate [1,000, 10,000, and 100,000] integer values (in the range of 0 to 106) to be inserted into the data structures. **NOTE** that this list (i.e., the input values) itself should not be sorted AND all algorithms should use the same input file when the input size is the same.

Case 2: Also, test the speed of algorithms when the input array is already sorted (for the same input data).

The following output should be provided for an average of 10 sorts of each algorithm and input size:

* the CPU time (use the same machine)
* the number of comparisons
* the number of swaps

Note: You can only use (i.e., copy) code from the book and slides. You need to develop the rest of the code yourself.

Deliverables: (as source code or a pdf file)

* Source code for your classes (one for each sort algorithm for a total of 3)
* A main file that tests the classes with randomly generated data.
* The data input files (there should be 6 of them – best and worst cases for each input size)
* Test output of your algorithms (with a tabular of each run and their averages for each of the metrics listed above)
  + Tables
    - Include data for individual runs
    - 6 tables per algorithm (so 18 total)
    - Algorithm – metric – scenario
    - Example:
      * Algorithm 1 - CPU time - Best Case
      * Algorithm 1 - CPU time - Worst Case
      * Algorithm 1 - Comparisons - Best Case
      * Algorithm 1 - Comparisons - Worst Case
      * Algorithm 1 - Swaps - Best Case
      * Algorithm 1 - Swaps - Worst Case
    - Include data for each run and input size
      * 3 columns: 1,000; 10,000; 100,000
      * 10 rows: one for each run
  + Graphs
    - Include data for averages
    - 6 graphs
    - Metric – size – scenario
    - Example:
      * CPU time – 1000 – Best Case
      * Graph averages for each algorithm
        + Algorithm 1
        + Algorithm 2
        + Algorithm 3

What to turn in: A softcopy of your sourcecode and reports should be uploaded to WebCampus. Combine the files in a single tar file.