CS 253 Design and Analysis of Algorithms

Course Project
Due Date: 03/03/2025

Part1: Comparison of Sorting Algorithms

The objective of the first part of the assignment is to study how the theoretical analysis of a variety of sorting algorithms compares with their actual performance. The sorting algorithms you will study are:

- **Bubble Sort**
- **Insertion Sort**
- **Merge Sort**
- **Quick Sort**
- **Heap Sort**
- **Radix Sort**

The major emphasis of this assignment is on analyzing the performance of the algorithms, NOT on coding the algorithms. Most of your time should be spent on designing careful test cases and analyzing your results in order to draw useful conclusions regarding the performance of the various algorithms.

The program can read the input from a file, or to create three types of input lists: (i) random generated elements, (ii) elements sorted in increasing order, (iii) elements sorted in decreasing order. If needed, you can add additional functionality to it. The format of the input files is as follows:

```
[n] /*number of positive integers to sort*/
[element 1]
[element 2]
:
[element n]
```

The times returned are microseconds and seconds *on my machine*. However, the value returned is system dependent.

Coding

The version of QuickSort program uses the first element in the array as the pivot element. You will study different strategies for selecting the pivot:

- Pivot Choice 1: The first element in the list
- Pivot Choice 2: A random element in the array.
- Pivot Choice 3: The median of the first, middle, and last elements in the array.

You should implement the Pivot Choices 2 and 3 listed above. You will then have three different versions of QuickSort (you would need to add them to the program as additional menu options).

2.Data generation and experimental setup.

The choice of test data is up to you (i.e., for each sorting algorithm, which input sizes should be tested, how many different inputs of the same size, which particular inputs of a given size.) Note that you will need to run your experiments several times in order to get stable measurements (i.e., times will vary depending upon system load, input, etc.). Your experimental setup must be described in terms of the following:

- What kind of machine did you use?
- What timing mechanism?
- How many times did you repeat each experiment?
- What times are reported?
- How did you select the inputs?
- Did you use the same inputs for all sorting algorithms?

3. Which of the three versions of Quick sort seems to perform the best?
   - Graph the best case running time as a function of input size n for all three versions (use the best case input you determined in each case in part 1).
   - Graph the worst case running time as a function of input size n for all three versions (use the worst case input you determined in each case in part 1).
   - Graph the average case running time as a function of input size n for all three versions.

4. Which of the six sorts seems to perform the best (consider the best version of Quicksort)?
   - Graph the best case running time as a function of input size n for the six sorts
   - Graph the worst case running time as a function of input size n for the six sorts
   - Graph the average case running time as a function of input size n for the six sorts.

5. Analyze your data to see if the number of comparisons is correlated with execution time. Plot (time / #comparisons) vs. n and refer to these plots in your answer.

Deliverables

• A **HARDCOPY REPORT.** It should address the points mentioned above. Your write-up must include a coherent discussion of which experiments you ran, how many times you ran them, etc.
Write your report carefully; explain things as clearly as possible, check for spelling errors. Answer the questions in the order presented. Use meaningful titles for each subsection and figure captions to explain the graphs. Also, graphs should be numbered and must be in the same section where they are discussed.

• AN ELECTRONIC COPY OF YOUR CODE. If you have several files to turn in, then please submit them in a zip/rar file. Provide instructions of how to run your codes.

****