CSDS 233 Assignment #5

Due November 25, 2024, before 11:59 pm. 100 points

Submission Instructions

- The submissions will be evaluated on completeness, correctness, and clarity.
- Please provide sufficient comments in your source code to help the TAs read it.
- Please generate a single zip file containing all your *.java files needed for this assignment (not .class) and, optionally a README.txt file with an explanation about added classes and extra changes you may have done.
- Name your file P5_YourCaseID_YourLastName.zip for your **coding exercises**. Submit your zip file electronically to Canvas.
- Please submit a **PDF** for your answers to the written exercise **separate from the zip file**.

Office Hours for This Assignment

- Written Assignment: November 16th , 3-4pm.. Please use my office hour link
- Programming Assignment: November 20th, 6-7pm Zoom Link Passcode: 000050

Contact Information

If you have questions outside of the scheduled office hours:

- For written questions, email: Vedant Gupta
- For coding problems, email: Michael Boelens

Written Exercise [50 points]

Sorting Algorithm (38 points)

Q1 Merge, Quick and Selection Sort (12 points)

Given the following list {12,5,9,3,15,7,2,10,6,8} and sorting functions below, write out each step of the sorting and comparison process until the list is fully sorted.

a) Merge Sort (4pts)
b) Quick Sort (4pts)
c) Selection Sort (4pts)

Q2 Bubble Sort (12 points)

a) Implement the **bubble sort** algorithm to sort the following list of integers in ascending order {34, 7, 23, 32, 5, 62}. Provide a step-by-step explanation of each pass through the list, detailing the comparisons and swaps made. (6pts)

b) Analyze the time complexity of bubble sort in the best case, worst case and average case scenarios. Discuss how the initial order of elements affects the performance of the algorithm. (6pts, 2pts for each scenario : Total 6pts)

Q3 Bucket Sort (14 points)

a) You are given the following list of floating-point numbers: {0.78, 0.17, 0.39, 0.26, 0.72, 0.94, 0.21, 0.12, 0.23, 0.68}. Use the bucket sort algorithm to sort the list. Please show each step for full credit, including the contexts of each bucket before and after sorting. Use 5 equal-sized buckets for this. (4pts)

b) How does the number of buckets affect the performance and accuracy of the bucket sort algorithm? (3pts)

c) Given the non-uniform distribution of the dataset, how does this impact the effectiveness of the Bucket sort algorithm? How can we modify the algorithm to handle the uneven distribution? (3pts)

d) Compare the time complexity of Bucket Sort with that of Quick Sort and Merge Sort. Under what circumstances would Bucket Sort be more efficient as compared to the other two sorting algorithms? List two.(4pts)

Insertion Sort on LinkedList (12 points)

Q4

Implement the insertion sort algorithm to sort a singly linked list of integers in ascending order. Please provide a write-up of your approach, including pseudocode and analyze the time complexities of your solution.

Programming Exercise [50 points]

In this assignment, you will implement several sorting algorithms and compare their performance on the same array. Each sorting method must be implemented manually without using built-in sorting libraries. After implementing each sorting algorithm, analyze the number of steps each algorithm takes to complete, to better understand their efficiency and where they might be most effective.

You will implement the following methods in a class named Sort.java:

1. **Insertion Sort** [10 pts]
    - **public void insertionSort(int[] array):**
        - Sorts an array into descending order using the Insertion Sort algorithm. The function should print the state of the array after each swap.
2. **Merge Sort** [10 pts]
    - **public void mergeSort(int[] array):**
        - Sorts an array into descending order using the Merge Sort algorithm. The function should print the state of the array after each merge operation.

3. **Quick Sort** [10 pts]
    - **public void quickSort(int[] array, int low, int high):**
        - Sorts an array into descending order using the Quick Sort algorithm. The function should print the state of the array after each partitioning operation.
4. **Bucket Sort** [10 pts]
    - **public void bucketSort(int[] array, int size):**
        - Sorts an array into descending order using the Bucket Sort algorithm. The function should print the state of the buckets and the array after distributing elements into buckets and after each bucket is sorted and combined.

- **Testing** [5 pts]
    - Additionally, you are requested to test each function in the **main** method using the array [26, 13, 72, 3, 17, 37, 0, 17, 73, 45] for testing each sorting algorithm

- **Analysis** [5 pts]
    - After testing each sorting algorithm with the same array, compare the number of steps (comparisons, swaps, and accesses) for each algorithm. Does the algorithm with the fewest steps seem the most efficient? (you can put this in a comment at the end of the file)