

CSE204: Data Structures and Algorithms I Sessional

Assignment 7 on Sorting Algorithms

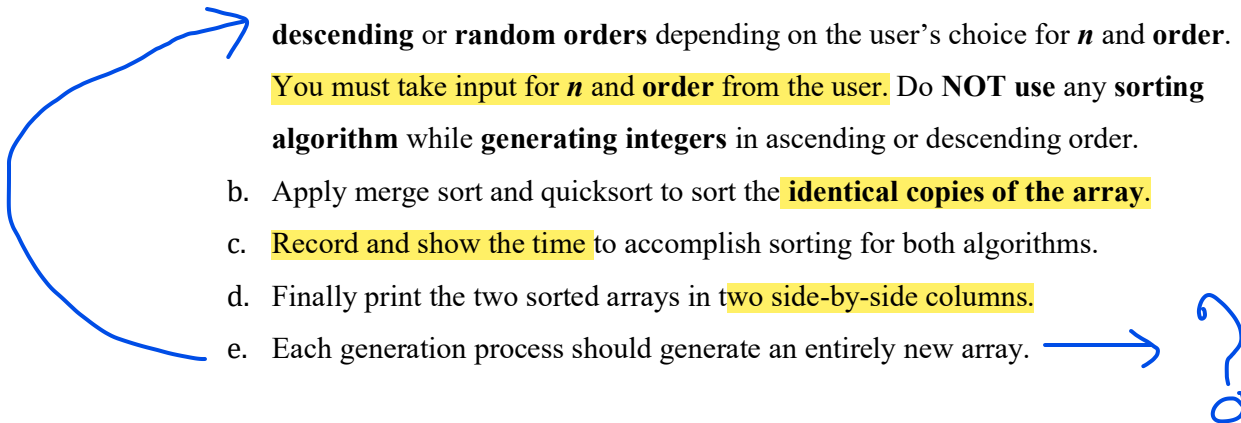
In this assignment you will implement **Quicksort and Merge Sort algorithms** and **compare** their **performance**. The assignment includes **demonstration** of your implementation to the instructor **during evaluation** as well as a **report submission** using pre-generated data. Please carefully read, understand and follow the steps outlined below.

A. Implementation of Sorting Algorithms

1. Implement **Quicksort** and **Merge sort** algorithms for sorting a number of **integers** in **ascending order**.

B. During Evaluation

2. **Repeatedly do the following steps** until the user quits.
 - a. **Generate automatically** an array of **n integers** in anyone of **ascending, descending or random orders** depending on the user's choice for **n** and **order**.
You must take input for n and order from the user. Do **NOT** use any **sorting algorithm** while **generating integers** in ascending or descending order.
 - b. Apply merge sort and quicksort to sort the **identical copies of the array**.
 - c. **Record and show the time** to accomplish sorting for both algorithms.
 - d. Finally print the two sorted arrays in **two side-by-side columns**.
 - e. Each generation process should generate an entirely new array.



C. Report Generation

3. For each value of $n = 10^1, 10^2, 10^3, \dots, 10^6$, generate statistics as follows.
 - a. Generate **three** arrays of **n integers** in **ascending, descending and random orders** (one in each order). These arrays must be generated multiple times for each value of n as explained in **Step c**. Do **NOT** use any **sorting algorithm** while **generating integers** in ascending or descending order.
 - b. Apply merge sort and quicksort to sort the arrays. Use **identical copies of arrays** as input to both algorithms.

- c. Record the time to accomplish sorting in the following table. For example: You want to get timing for $n=10$ to sort an array in ascending order with merge sorting. Generate the scenario multiple times (**say, 10 or 20 times**) and take the **average sorting time**. Record **ONLY** the **average sorting time** into the cell. **Please note exactly the same data must be used for the other sorting algorithm.**

Table: *Average time* for sorting n integers in different **input orders**

Input Order	$n =$	10	100	1000	10000	100000	1000000
	Sorting Algorithm						
Ascending	Merge						
	Quick						
Descending	Merge						
	Quick						
Random	Merge						
	Quick						

4. **Plot average** running **time** of both sorting algorithms against the input array size n for all input orders (e.g., a **single** plot of time vs. n using all statistics). Use appropriate legend, colors, line width, line type, etc. to improve the readability your plot.
5. **Generate the report** containing **complexity analysis** of both algorithms for all input orders, **machine configuration, table** (as explained in Step 3) and **plot** ((as explained in Step 4)). The report must be in **pdf format**.

Special Instructions:

Write **readable, re-usable, well-structured, quality** code. This includes but is not limited to writing appropriate functions for implementation of the required algorithms, **meaningful naming** of the variables, **suitable comments** where required, proper indentation etc.

Please **DO NOT COPY** solutions from anywhere (your friends, seniors, internet etc.). Any form of plagiarism (irrespective of source or destination), will result in getting -100% marks in

the offline. Also, be informed that for repeated offence of plagiarism, the departmental policies suggest stricter measures.

Submission

1. Create an empty folder named to your student_id (e.g. 1805001)
2. Put all the source code files and reports in that folder
3. Zip that folder. It should give you student_id.zip
4. Submit the zip file to moodle

Submission Deadline: June 11, 2021 11:55 PM