Lab 4: Sorting algorithm

Figure 1: Good luck!

1 Exercise

1. Implement each sorting algorithm to sort a list containing 1K random integers. Measure the execution time and compare the performance of the algorithms.
2. Implement each sorting algorithm and benchmark their performance on arrays of varying sizes (e.g., 100, 1,000, 10,000 elements). Record the time taken for each sort and plot the results to visualize the differences in efficiency.
3. Modify each sorting algorithm to handle not just integers but also strings and floating-point numbers. Test the algorithms with different types of data and analyze how they perform with varying data types.
4. Implement each sorting algorithm to sort a linked list, queue and stack. Compare their performance.
5. Enhance the Insertion Sort algorithm by using Binary Search to find the insertion position. Compare its performance with standard Insertion Sort.
6. Implement Heap Sort to find the maximum value in an array without fully sorting the array. Explain how this algorithm works.
7. Design a hybrid algorithm that uses Merge Sort for large arrays (e.g., size greater than 100). and Insertion Sort for smaller arrays (e.g., size less than or equal to 10). Compare its performance with standard Merge Sort.
8. Design a hybrid sorting algorithm that uses Quick Sort for large datasets (100 records) and switches to Insertion Sort for smaller subarrays (e.g., size less than or equal to 10). Analyze its performance compared to using Quick Sort alone.

PREPARED BY Tran Vinh Khiem

9. Extend the sorting algorithms to handle multi-dimensional data (e.g., sorting a list of points in a 2D space based on their distance from the origin). Analyze how each algorithm adapts to this new challenge

Note

- Develop a visualization tool that graphically represents different searching algorithms (e.g., binary search, linear search) in action. Use animations to show how elements are compared and swapped during the sorting process. This can help in understanding the mechanics behind each algorithm. Allow users to input data and visualize the steps taken during the search process.
- Each sorting algorithm including buble sort, selection sort, insertion sort, binary insertion sort, quick sort, merge sort, heap sort

2 Homework

1. An e-commerce website needs to display products based on various criteria, such as price, rating, or popularity. Implement best sorting algorithms to allow users to sort product listings by different attributes. Analyze the performance of each algorithm when dealing with large datasets of products with varying attributes. Link of dataset is here `https://www.kaggle.com/datasets/arashnic/e-product-pricing`.
2. A social media platform needs to sort user posts based on engagement metrics (likes, shares, comments). Simulate a social media feed where posts are sorted based on multiple criteria (e.g., time posted, number of likes). Implement sorting algorithms and analyze how they affect the user experience in terms of responsiveness and accuracy. Students can use this datatset: `https://www.kaggle.com/datasets/kashishparmar02/social-media-sentiments-analysis-dataset`
3. A school management system needs to sort students based on their grades or names for reporting purposes.Create a program that sorts student records (name, grade, age) using different sorting algorithms. Provide options for sorting by different fields and evaluate which algorithm performs best with large datasets. Students can use this datatset: `https://www.kaggle.com/datasets/spscientist/students-performance-in-exams`
4. The application of sorting algorithms significantly transforms the organization and accessibility of music playlists, thereby improving user experience by facilitating song selection according to diverse criteria such as genre, artist, or mood. This digital sorting mechanism enables personalized playlists and effective classification of genres, ultimately enhancing the enjoyment and customization of music listening experiences. Students can use this datatset: `https://www.kaggle.com/datasets/viktoriiashkurenko/278k-spotify-songs`
5. * Exlore the new world of sorting algorithm. Acess to this link `https://stanford-cs161.github.io/winter2023/assets/files/lecture6-notes.pdf` to find out more about Counting Sort and Radix Sort. Implement these algorithms and insert them into exercise 4.

Notice

- Use C++ for practice.
- In the programming file, the student should include the following complete information:

```
//STT: 39 (Example)
//Full Name: X, With X is you, don't need to find X anywhere else.
//Session 01 - Exercise 01
//Notes or Remarks: ......
```

- Develop a visualization tool that graphically represents different exercises in action. Allow users to input data and visualize the steps taken during the search process.
- Students can use Google colab for homework.

References :

[1]. Skiena, S. S. (1998). The algorithm design manual (Vol. 2). New York: springer.
[2]. Pai, G. V. (2023). A Textbook of Data Structures and Algorithms, Volume 3: Mastering Advanced Data Structures and Algorithm Design Strategies. John Wiley & Sons.
[3]. Anggoro, W. (2018). C++ Data Structures and Algorithms: Learn how to write efficient code to build scalable and robust applications in C++. Packt Publishing Ltd.
[3].Leetcode
[4].Codeforce
[5]. `https://www.youtube.com/watch?v=RfXt_qHDEPw`