Datastructures Lab

Lab 9

Date: 11.9.25

Observation questions

1. What is the time complexity of Insertion Sort, Quick Sort, and Heap Sort in average and worst cases?
2. Explain how a Binary Search Tree (BST) helps in sorting data and performing searches.
3. Describe the difference between internal sorting and external sorting techniques. Give examples of when each is used.
4. Explain the principle and time complexity of Counting Sort. Why is Counting Sort preferred over comparison-based sorting algorithms in certain scenarios?
5. What is External Sorting? Describe situations where External Sorting is necessary. How does it differ from internal sorting techniques?
6. Explain Multiway Merge Sort and how it improves the efficiency of merging large datasets. How is it used in external sorting applications such as database management systems?

Execution questions

7. An e-commerce application stores a list of product prices. Implement Heap Sort to sort the product prices in ascending order. After sorting, implement Binary Search to find if a product of a particular price is available.

Input Format:

- First line: `n` (number of products)
- Second line: `n` integers representing product prices
- Third line: `target_price` (price to search)

Output Format:

- Sorted product prices in ascending order
- "Found" or "Not Found" based on binary search result

8. A university wants to process student marks. Implement Quick Sort to sort the student marks in descending order.

Input Format:

- First line: `n` (number of students)
- Second line: `n` integers representing marks of students

Output Format:

- Sorted marks in descending order

9. An Operating System uses Insertion Sort to prioritize tasks based on their priority numbers (lower number → higher priority).

Input Format:

- First line: `n` (number of tasks)
- Second line: `n` integers representing task priority numbers

Output Format:

- Sorted priority numbers in ascending order

10. In a file system directory structure represented as a tree, use **Counting Sort** to sort files at each level of the directory by their file sizes.

- **Input Format** (Sample directory files with sizes in KB):

```
Level 1: 50 20 70 10
```

- **Expected Output Format**:

```
Sorted File Sizes: 10 20 50 70
```

11. Simulate **External Sorting** by sorting a large file of numbers using a chunk-based approach and merging sorted chunks.

- **Input Format**:

A file with numbers:

```
45
12
89
23
56
78
34
90
```

- **Expected Output Format**:

```
Sorted Numbers in File:
12
23
34
45
56
78
89
90
```