AIML-BETA
DAA CASE STUDY

2211CS020136

Q)

1.) An e-commerce platform is implementing a feature where products need to be sorted by various attributes (e.g., price, rating, and name). The product list contains millions of items, and the sorting operation needs to be efficient and scalable. 1. What are the time and space complexities of the commonly used sorting algorithms (Quick Sort, Merge Sort)?

2.) How do the characteristics of the data (e.g., range of prices, product name lengths) impact the choice of sorting algorithm?Sorting Algorithms for E-commerce Platform

1. Time and Space Complexities of Common Sorting Algorithms

Quick Sort

- **Time Complexity**:
  - Best Case: **O(n log n)** (when pivot divides the list evenly).
  - Average Case: **O(n log n)**.
  - Worst Case: **O(n^2)** (when pivot selection is poor, e.g., smallest or largest element repeatedly).
- **Space Complexity**:
  - **O(log n)** (due to recursive stack usage in the best and average cases).
  - **O(n)** in the worst case if the recursion depth becomes the size of the list.
- **Scalability**: Quick Sort is generally faster in practice due to low constant factors, but performance degrades with poor pivot selection.

Merge Sort

- **Time Complexity**:
  - Best Case: **O(n log n)**.
  - Average Case: **O(n log n)**.
  - Worst Case: **O(n log n)** (merge sort always divides the list evenly).
- **Space Complexity**:
  - **O(n)** (for temporary arrays used during merging).
- **Scalability**: Merge Sort is stable and consistent in terms of performance, making it suitable for scenarios requiring guaranteed O(n log n).

2. Impact of Data Characteristics on Sorting Algorithm Choice

Range of Prices

- If the range of prices is small and prices are integers, **counting sort** or **radix sort** can outperform comparison-based algorithms with time complexity O(n). These algorithms exploit the small range of values for efficiency.
- For larger ranges or floating-point prices, comparison-based algorithms like Merge Sort or Quick Sort are preferable. **Product Name Lengths**
- Sorting by product names involves comparing strings. The time complexity depends on both the number of strings (n) and the average string length (m):
    - ✦ Quick Sort and Merge Sort work in **O(n log n)** comparisons but may incur additional costs due to string comparison overhead.
    - ✦ For uniform and short string lengths, radix sort can perform in **O(n · m)**.

Additional Considerations

- **Stability**: Merge Sort is stable, which means it preserves the relative order of items with equal keys. This is important when secondary sorting criteria are involved.
- **Parallelism**: Merge Sort is easier to parallelize, making it suitable for distributed systems.
- **Memory Constraints**: Quick Sort is more memory-efficient (O(log n) space) than Merge Sort but may be less stable.
- **Distribution of Data**: Quick Sort performs better with evenly distributed data, while Merge Sort is not affected by data distribution.

Recommendations for E-commerce Platform

1. **Sorting by Price**:
    - Use Quick Sort for general use cases.
    - Use Counting Sort or Radix Sort for integer prices with a small range.
2. **Sorting by Ratings**:
    - Use Radix Sort if ratings are bounded integers.
    - Use Quick Sort or Merge Sort otherwise.

3. Sorting by Product Names:

- Use Merge Sort for stability if secondary sorting is needed.
- Use Quick Sort for faster performance if stability is not required.