NAME: NABASA ISAAC

M23B23/043

B22448

1.1-1 Describe your own real-world example that requires sorting. Describe one that requires finding the shortest distance between two points.

My real world problem is sorting files sent on whatsapp. When I download them, they end up missed up with other files.

- Finding the fastest route between two locations on a map using GPS navigation

1.1-2 Other than speed, what other measures of efficiency might you need to consider in a real-world setting?

 Memory usage**:** How much memory is required to execute the algorithm, especially for large datasets.

 Energy consumption**:** The amount of power used by the algorithm, particularly relevant for battery-powered devices or high-performance computing.

 Scalability**:** How well the algorithm performs as the input size increases, ensuring it can handle large datasets efficiently.

 Responsiveness**:** The time it takes for the algorithm to produce intermediate results or updates, especially in interactive applications.

1.1-3 Select a data structure that you have seen, and discuss its strengths and limitations.

Queues

Strengths of Queues

- **FIFO order:** Queues ensure that elements are processed in the order they were added, which is essential in many applications where fairness or priority is not a factor.
- **Efficient operations:** Basic operations like enqueue (adding an element) and dequeue (removing an element) are typically efficient, often with a time complexity of O(1).
- **Versatility:** Queues can be used in various applications, including task scheduling, print spooling, and breadth-first search algorithms.

Limitations of Queues

- **Fixed order:** Queues strictly adhere to FIFO order, which may not be suitable in situations where priority or specific ordering is required.
- **Limited random access:** Elements cannot be accessed or modified directly at arbitrary positions within the queue, except for the front and rear.
- **Potential for memory issues:** Implementing queues using arrays can lead to memory inefficiencies if the queue size is not managed properly.

1.1-4 How are the shortest-path and traveling-salesperson problems given above similar? How are they different?

The traveling salesman problem (TSP) is a classic optimization problem in computer science. It involves finding the shortest possible route that visits a given set of cities exactly once and returns to the starting city.

Similarities:

- Both involve finding optimal paths in a graph.
- Both can be represented as optimization problems.

Differences:

- The shortest-path problem finds the shortest path between two specific nodes, while the traveling-salesperson problem finds the shortest path that visits all nodes exactly once.
- The traveling-salesperson problem is generally NP-hard, making it more challenging to solve efficiently.

1.1-5 Suggest a real-world problem in which only the best solution will do. Then come up with one in which "approximately" the best solution is enough

Problem Requiring the Best Solution:

- Medical Diagnosis: When diagnosing a critical illness, an exact diagnosis is crucial to ensure proper treatment. An incorrect diagnosis could lead to severe health consequences or even death. For example, misdiagnosing a heart attack could have fatal results.

Problem Requiring an Approximate Solution:

- Product Optimization: In product design, finding the *absolute* optimal solution can be computationally expensive or impractical. Instead, an approximate solution that is close to optimal can be sufficient. For instance, when designing a new car, finding the exact configuration of components that minimizes fuel consumption and maximizes performance might be challenging. An approximate solution that is within a reasonable margin of error can still be highly effective.

1.1-6 Describe a real-world problem in which sometimes the entire input is available before you need to solve the problem, but other times the input is not entirely available in advance and arrives over time.

Imagine you're playing an online multiplayer game where you need to build a base to defend against attacks.

- Complete Input: Before the game starts, you might have a map of the entire game world, showing the locations of resources, potential enemy spawn points, and safe zones. This would allow you to plan your base construction strategy in advance.

- Partial Input: However, as the game progresses, unexpected events can occur, such as enemy attacks, new resources becoming available, or changes to the game environment. You'll need to adapt your base defence strategy based on this new information, which arrives over time.