KIISA ANGELA, B24258, S23B23/027

1.1-1

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

Example that Requires Sorting:

- Organizing a Playlist: With a large playlist of songs on Spotify, I decide to sort the songs alphabetically by title or artist, or by release date. This makes it easier to find specific songs when I want something particular.

Example that Requires Finding the Shortest Distance Between Two Points:

- Navigating to a Friend's House: When using google maps to drive to a friend's house, the app calculates the shortest distance based on current traffic conditions. It considers various routes and factors like road closures or traffic jams to determine the quickest way to reach my destination.

1.1-2

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

- Memory Usage: The amount of memory an algorithm or process consumes is important, especially when working with limited resources or large datasets.

- Scalability: The ability of an algorithm or system to handle increasing amounts of data without significant decrease in performance.

- Accuracy: The correctness of the output is essential, especially in critical systems where errors can have significant consequences.

- Maintainability: How easily the code or system can be understood, modified, and extended by developers over time.

- Robustness: The ability of the system to handle unexpected inputs or situations without crashing.

- Security: Ensuring that the algorithm or system protects against unauthorized access and vulnerabilities.

1.1-3

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

Array

Strengths:

- Simplicity: Arrays are one of the simplest data structures consisting of a fixed-size sequence of elements stored in neighbouring memory locations.

- Constant-Time Access: Arrays allow for O(1) time complexity for accessing an element by its index, which is extremely fast and efficient.

- Efficient Iteration: Iterating through an array is straightforward and can be done in linear time, O(n), where n is the number of elements.

Limitations:

- Fixed Size: The size of an array is determined at the time of creation and cannot be changed. This makes it difficult to handle scenarios where the amount of data is dynamic or unknown.

- Difficult Insertions and Deletions: Inserting or deleting elements in an array can be difficult because it may require shifting other elements.

- Memory Inefficiency: If the array is underutilized (i.e many of its elements are unused), it can waste memory. Conversely, if it becomes full, creating a new, larger array and copying elements can be costly.

1.1-4

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

Similarities:

- Both problems involve finding an optimal path in a graph.

- They require understanding the structure of the graph, the weights on the edges, and ensuring that the chosen path is either minimal in distance or cost.

Differences:

-Shortest-Path Problem: This problem focuses on finding the shortest possible path between two specific nodes in a graph. It does not require visiting all nodes, only the most efficient route from the start to the destination.

- Traveling Salesperson Problem: This requires finding the shortest possible route that visits every node (city) exactly once and returns to the starting node.

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 good enough.

Best Solution Needed:

Medical Treatment: In certain medical cases, only the best solution is acceptable. The exact dosage and targeting must be precise to maximize treatment effectiveness and minimize harm to healthy areas.

Approximately Best Solution is Good Enough:

Route Planning for Delivery Services: In many delivery scenarios, finding the exact best route is difficult due to the large number of possible routes. However, an approximately best solution that minimizes travel time and fuel consumption is generally sufficient to ensure efficiency.

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.

Planning a Vacation:

- Entire Input Available: If you have all your travel details like flight times, hotel bookings, and activities sorted out before your trip, you can create a complete schedule and plan your days in advance.

- Input Arrives Over Time: Sometimes, you might not have all the information upfront, such as weather conditions, local events, or specific activity availability. You may need to adjust your plans as new information comes in, like changes in weather forecasts.