1.1-1

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

- Organizing a Playlist by Song Duration: while creating a playlist for a party, one may want to arrange songs by their length. For example, you might want the shortest songs to play first and the longest ones later. Sorting the songs by duration in ascending order would allow you to structure the playlist from shortest to longest.

- Optimizing a Delivery Route**:** A delivery company with multiple packages to deliver in a city would need to figure out the most efficient path for the driver to take. The goal is to minimize the total driving distance while visiting each delivery point. This is an alternative of the "Traveling Salesperson Problem," where you need to find the shortest path that connects all points (delivery locations) and returns to the starting point.

1.1-2

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

- Cost: The total amount of money required to complete a task or project, including materials, labor, and overhead expenses.
- Energy: The quantity of power or fuel consumed during the process, which can impact both the environment and operating costs.
- Resource Usage: The amount of materials, supplies, or tools needed to achieve the desired outcome, influencing waste and sustainability.
- Time: The overall duration it takes to finish a task, which can affect deadlines and productivity levels.
- Precision: How accurate or correct the results are, ensuring that the final product meets quality standards and expectations.
- Effort: The degree of physical or mental work required, which can impact employee morale and efficiency.

- Space: The physical area or storage capacity needed to carry out a task or hold materials, which can affect logistics and organization.

1.1-3

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

Strengths of an Array:

- **Fast Access (O(1))**: Arrays provide constant time access to elements when using their index. If you know the index of an element, you can retrieve it instantly.
- **Simple Structure**: Arrays are straightforward to understand and implement. They store elements of the same type in contiguous memory, which keeps things organized.
- **Efficient Memory Usage**: Since arrays are stored in contiguous memory, they have low overhead and are memory-efficient compared to other data structures like linked lists.
- **Good for Fixed-Size Data**: If the number of elements is known ahead of time and won't change, arrays are a very efficient structure.

Limitations of an Array:

- **Fixed Size**: Once an array is created, its size cannot be changed. If you need more space, you would have to create a new array and copy the elements over, which can be time-consuming.
- **Insertion and Deletion (O(n))**: Adding or removing elements in the middle of an array is inefficient, as it requires shifting elements to accommodate the change. This takes linear time.
- **Limited Flexibility**: Arrays are not dynamic in nature, which means they can’t easily adapt to growing or shrinking data unless you manually manage resizing.
- **Inefficient for Search (O(n))**: If the array is unsorted, finding an element requires checking each one sequentially, making search operations slow.

1.1-4

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

Similarities:

- **Both involve finding optimal paths**: In both the shortest-path problem and the traveling salesperson problem, you're trying to find the best route between locations.
- **Deal with distances**: They both focus on minimizing distances, either between two points (shortest-path) or across multiple points (traveling salesperson).

Differences:

- **Shortest-path problem**: Only finds the best route between two specific points.
- **Traveling salesperson problem**: Finds the shortest route to visit all locations and return to the starting point.
- **Complexity**: The traveling salesperson problem is more complicated because it involves many locations, while the shortest-path problem only focuses on two points.

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.

Real-World Problem Needing the Best Solution:

Airplane Safety System: When designing an airplane's safety system, only the best solution will work because people's lives are at stake. There is no room for errors.

Real-World Problem Where an Approximate Solution is Good Enough:

Packing a Suitcase for a Trip: When packing for a trip, you don't need the *perfect* arrangement of clothes to fit everything. An arrangement that fits most of what you need is good enough, even if it's not perfectly optimized.

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.

Online Shopping Orders:

- **Entire Input Available:** In some cases, all customer orders are placed by a cutoff time, such as 9:00 AM. The delivery service has all the information upfront and can plan the most efficient delivery routes for the day. Since all the input is available at once, it allows the delivery team to optimize for time and cost, making the process smoother**.**

- **Input Arrives Over Time**: Other times, orders keep coming in throughout the day. For example, if orders are placed at different times, the delivery service can’t plan everything upfront. Instead, they have to adjust routes on the fly, adding new stops and reworking the schedule to accommodate the new information. This is trickier and less efficient, as adjustments need to be made constantly.