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

Real-world example that requires sorting;

A grocery store wants to organize its products on shelves in alphabetical order to make them easy for customers to find.

Real world example that requires finding the shortest distance between two points;

Navigation apps like Google Maps or Waze need to calculate the shortest distance between two locations on a map, considering factors like road conditions and traffic.

* + 1. **Other than speed, what other measures of efficiency might you need to consider in a real-world setting?**
* **Memory Usage:** The memory the algorithm requires to run is especially important for devices with limited memory, like smartphones or embedded systems.
* **Energy Consumption:** For battery-powered devices, energy efficiency is crucial. Algorithms that require less energy can extend battery life.
* **Scalability:** How well does the algorithm handle large datasets? A scalable algorithm can efficiently process increasing amounts of data.
  + 1. **Select a data structure that you have seen, and discuss its strengths and limitations.**

My data structure of choice is Stacks.

Strengths:

* **Simple Implementation:** Stacks can be implemented using arrays or linked lists, making them relatively straightforward to code.
* **LIFO (Last-In, First-Out) Order:** Stacks naturally follow a LIFO order, which is useful in applications where the most recently added element needs to be accessed or removed first.
* **Efficient Operations:** Basic operations like push (adding an element) and pop (removing the top element) are typically efficient, especially when implemented using arrays.

Limitations:

* **Fixed Size (for Array-Based Stacks):** Array-based stacks have a fixed size, which can be a limitation if the number of elements is unpredictable. If the stack overflows, you might need to resize the array, which can be inefficient for frequent resizing.
* **Inefficient Random Access:** If you need to access an element at a specific index in the stack, you'll need to traverse the entire stack from the top, which can be inefficient for large stacks.
  + 1. **How are the shortest-path and traveling-salesperson problems given above similar? How are they different?**

Similarities:

* Both problems involve finding the optimal path between multiple points.
* Both problems can be represented as graphs, where nodes are points and edges represent connections between them.

Differences;

|  |  |
| --- | --- |
| Shortest-path problem | Traveling-salesperson problem |
| The goal is to find the shortest path between two specific points. | The goal is to find the shortest path that visits all nodes exactly once and returns to the starting node. |
| can be solved efficiently using algorithms like Dijkstra's algorithm | more complex, being NP-hard |

* + 1. **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 in which only the best solution will do;

**Medical Diagnosis:** A medical diagnosis system must provide the most accurate diagnosis to ensure proper treatment.

Real-world problem in which “approximately” the best solution is good enough;

**Route Optimization:** A delivery service can use approximation algorithms to find near-optimal routes, balancing efficiency with cost.

* + 1. **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.**

In weather forecasting**,** meteorologists use historical weather data to predict future conditions. However, real-time weather data (like temperature, wind speed, and atmospheric pressure) is continually updated. Forecasting models must adjust dynamically to new information as it arrives, balancing predictions based on past data with real-time updates.