1.1-1

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

- Sorting in organizing a list of names in alphabetical order. It could be used in a school or business setting, where a list of employees, students, or customers' needs to be sorted for easier access.
- Finding the shortest distance between two cities on a map like. For instance, travelling from Mukono to Kampala, a GPS system with the help of google map calculates the shortest driving route between your current location being Mukono and a destination being Kampala by considering different roads and traffic conditions.

1.1-2

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

- Memory usage: In some scenarios, such as embedded systems or mobile devices, how much memory an algorithm uses is crucial because these devices have limited resources.
- Energy consumption: In mobile or battery-powered devices, algorithms need to be efficient in terms of energy use to prolong battery life.
- Scalability: Some algorithms may perform well with small data sets but become inefficient with large ones.

1.1-3

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

- Linked List

  Strengths:

  - ✓ Dynamic Size: Linked lists can grow or shrink in size dynamically, which means you don't need to specify the size in advance. This makes them well-suited for applications where the number of elements is unpredictable.
  - ✓ Efficient Insertions/Deletions: Insertions and deletions are efficient, especially at the beginning or middle of the list, since they can be done in constant time, O(1), without the need to shift elements.

- ✓ Flexible Memory Use: Linked lists do not need contiguous blocks of memory, so they are ideal in environments with limited or fragmented memory.

Limitations:

- o Slow Lookups: Linked lists require traversal from the head to the desired element, resulting in O(n) time complexity for searching.
- o Extra Memory for Pointers: Each element (node) in a linked list requires additional memory to store a pointer/reference to the next node. This overhead can be significant when dealing with a large number of nodes.
- o Poor Cache Locality Due to non-contiguous memory allocation, linked lists suffer from poor cache locality, making them slower compared in terms of accessing elements, as they require more CPU cache misses.

1.1-4

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

Similarities:

- ✓ Both problems involve finding optimal paths in a graph or network. They are concerned with minimizing some metric, like distance or cost.
- ✓ Both problems typically use weighted graphs where edges have values representing distances or costs.

Differences:

- ✓ The Shortest Path Problem focuses on finding the least-cost path between two specific points in a graph.
- ✓ The Traveling Salesperson Problem (TSP) is about finding the shortest possible route that visits all points (nodes) exactly once and returns to the starting point, which is a more complex combinatorial optimization problem.

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:

  Airplane scheduling for airports requires the best solution because delays or overlaps in plane arrivals and departures could lead to serious safety risks and logistical problems.

- ✓ Approximate Solution is Good Enough:

  Online product recommendations, such as those used by Amazon or Netflix, do not require the absolute best match for a user's preferences. An approximate solution, where the recommendations are "good enough," typically satisfies the user's needs.

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.

Weather prediction is a problem where sometimes the entire input (historical weather data) is available, allowing for more accurate forecasts. However, real-time weather changes (e.g., a sudden storm) mean that the input may continue arriving over time, and forecasts must be adjusted dynamically as new data becomes available.