**NAME**: MOKILI PROMISE PIERRE

**REGISTRATION** **NUMBER**: M23B23/032

**ACCESS NUMBER**: B20848

**Exercises**

**1.1 - 1**

The urban food delivery route optimisation is a good example of a real-world scenario that requires sorting while finding shortest distance between two points, especially during peak hours where many orders are made in close succession in busy cities.

To save time and fuel, the company needs to find the shortest path for its deliverers to move between various delivery points. Each deliverer may have to pick up food from various restaurants and deliver it to several customers in a short period of time. Given the complexity of city streets, one-way roads, and traffic, determining the shortest route between all points becomes crucial.

**1.1 - 2**

**Other than speed, other measures you might need to consider in a real-world setting include:**

* Efficient use of available resources, such as raw materials, energy, labor, and capital, to achieve the desired result with the lowest possible cost, without sacrificing quality or performance.
* Using less energy to perform the same task or achieve the same result.
* The ability of a system or process to handle increased workload or growth without a significant drop in performance or efficiency.
* The consistency and dependability of a system or process in delivering results over time.
* Minimizing negative environmental impact while maintaining operational efficiency.
* The ability to meet or exceed customer or industry standards without defects or rework, which contributes to overall operational efficiency.
* The ability to adapt quickly and efficiently to changing conditions, such as new technologies, regulations, or market demands.
* The amount of work or products that can be processed or produced in a given period.
* The frequency of errors or defects in a process. Fewer errors indicate higher efficiency.
* The extent to which a service or product meets or exceeds customer expectations, often linked to efficiency in terms of service delivery, reliability, and quality.
* Managing and processing data in a way that minimizes storage space, bandwidth, or computational power.

**1.1 - 3**

**HASH TABLES**

Hash tables are a data structure used to offer fast data retrieval and storage based on key-value pairs kinda like python dicts.

**Strengths**

**Efficient for Large Datasets**

For large collections of data, the efficiency of constant-time operations (on average) becomes crucial. Hash tables can store large datasets efficiently, particularly when there are few hash collisions.

**Flexible Keys**

Hash tables can store and retrieve data based on various key types (strings, numbers, objects), offering more flexibility than array-based structures.

**Fast Access (O(1) Average Time Complexity)**

Hash tables provide constant-time average complexity for insertion, deletion, and lookup operations. This is because a hash function computes an index where data can be stored or retrieved, making access extremely fast for large datasets.

**Dynamic Sizing**

Many hash table implementations dynamically resize as the number of entries increases. This resizing process ensures that the load factor (ratio of elements to available slots) remains low, which helps maintain fast access times.

**Weaknesses**

**Collisions**

A hash table's performance is highly dependent on the hash function. If multiple keys map to the same index (a collision), the table may degrade in performance as it needs to resolve the collision (e.g., through chaining or open addressing). This can increase access time from O(1) to O(n) in the worst case.

**Memory Overhead**

Hash tables may require more memory than other data structures due to the need for an underlying array and space to handle potential collisions. The table often has more space than needed to maintain fast access times and avoid a high load factor.

**Poor Performance for Range Queries**

Hash tables are inefficient for operations where sorted data or range queries are required (e.g., retrieving all entries with keys between a certain range). Since the keys are distributed non-sequentially based on the hash function, range queries are not feasible without first collecting and sorting the keys.

**Hash Function Complexity**

The choice and design of the hash function are critical. A poorly designed hash function can result in excessive collisions, reducing the efficiency of the hash table. Additionally, complex hash functions can slow down insertions and lookups due to the time it takes to compute the hash.

**No Order Guarantee**

Hash tables do not preserve the order of keys. Unlike structures like balanced trees or linked lists, which maintain a sorted or insertion order, hash tables scatter keys based on the hash function. This can be problematic in scenarios where ordered data is necessary.

**1.1-4**

**SIMILARITIES BETWEEN SHORTEST-PATH AND TRAVELLING-SALESPERSON PROBLEMS**

*The shortest path problem and the traveling salesperson problem are similar in several ways:*

* Both problems are represented using graphs, where locations (e.g., cities, delivery points) are nodes, and the paths or routes between them are edges. These edges are often weighted with distances or costs.
* Both problems aim to minimise a cost (e.g., distance, time, or fuel consumption). In the shortest path problem, the goal is to minimise the total distance between two specific points, while in TSP, the goal is to minimise the total distance required to visit all points and return to the starting point.
* Both problems are commonly applied in routing and logistics contexts. They help optimise travel time, reduce fuel consumption, and improve overall efficiency in transportation networks.
* Both problems can be tackled using graph traversal algorithms like greedy approaches or dynamic programming. While the shortest path problem often uses specific algorithms like Dijkstra’s, TSP requires more complex methods.
* Both problems experience a significant increase in complexity as the graph becomes larger (more nodes and edges). In large graphs, finding the shortest path between two nodes can be computationally expensive, while solving the TSP is NP-hard and grows exponentially more difficult as the number of nodes increases.

**DIFFERENCES BETWEEN SHORTEST-PATH AND TRAVELLING-SALESPERSON PROBLEMS**

The Shortest Path Problem only requires finding the best route between two points while the traveling Salesperson Problem Requires finding the shortest tour that visits all points and returns to the starting point.

**1.1-5**

**Air Traffic Control Routing**

This is a real world scenario that requires only the best solution with minimal risk and maximum safety because routing planes safely to their destinations involves complex planning to avoid mid-air collisions and ensure efficient travel times. Here, the best possible solution is critical, as any miscalculation could result in catastrophic consequences like flight delays, collisions, or near-misses. Air traffic controllers need to determine the optimal routes and altitudes for multiple planes in real time, taking into account weather, airspace restrictions, and flight schedules.

**Delivery Route Optimisation**

For delivery companies like Amazon or UPS, determining the most efficient route to deliver packages to multiple locations can save time and fuel. However, due to changing traffic conditions, customer availability, and other factors, an approximate solution that is nearly optimal is often good enough.

In logistics, a slightly suboptimal route may add only a minor delay, but the difference is often insignificant in the broader scope of business operations.