

**COMSATS UNIVERSITY ISLAMABAD (CUI)**

***FINAL PROJECT***

Data Structures and

Algorithms

**Submitted By**

Rabeea Chughtai SP22-BCS-079

Sadia Imran SP22-BCS-087

Sadia Javed SP22-BCS-088

**Submitted To**

Dr. Inayat Ur Rehman

**Submitted On**

28th June 2023

# **`GROUP MEMBERS AND PERFORMED TASKS**

|  |  |  |
| --- | --- | --- |
| **Reg. No** | **Student Name** | **Tasks** |
| SP22-BCS-079 | Rabeea Chughtai | * Dijkstra Algorithm * Breadth First Search (BFS) |
| SP22-BCS-087 | Sadia Imran | * Heap * Hashing by Chaining Method |
| SP22-BCS-088 | Sadia Javed | * AVL * Prims Algorithm |

# **Dataset Description:**

* The Amazon product review dataset offers a rich source of information for analyzing customer feedback and extracting valuable insights.
* The Amazon product review dataset comprises data members such as Product ID, User ID, Profile Name, Helpfulness Numerator, Helpfulness Denominator, Score, Price, Summary, and Text.
* The dataset serves as the basis for implementing various data structures and exploring their applications.

## **AVL:**

* Because AVL trees self-balance, these operations remain efficient even as the dataset evolves over time.
* In context of my dataset an AVL tree based on the user ID, allowing you to search for specific products or users quickly.
* AVL trees can support range queries efficiently, allowing you to find products or reviews within a certain price range or based on a specific score.
* AVL trees keep their elements sorted, you can retrieve products in either ascending or descending order based on any parameter.
* With a time complexity of O(log n), AVL trees provide efficient searching. You can quickly find specific products, users, or ratings using their IDs or other attributes. This can be useful for analyzing specific data subsets.

## **Heap:**

* As a Priority Queue, a heap data structure might be useful in prioritizing customers with a 5 rating in Amazon product reviews.
* The highest-ranked customer is always at the root when utilizing a max-heap, allowing constant-time access.
* This enables quick retrieval of top-ranked customers, as well as efficient access to satisfied customers and their feedback.
* Prioritizing customers based on their ranking to products is critical for finding product strengths, improving beneficial evaluations, and addressing potential customer complaints.

## **Dijkstra’s Algorithm:**

* In my case, I have represented my dataset as a graph, where each product is a node and the cost difference between products is the weight assigned to the edges, then applying Dijkstra's algorithm can help find the shortest path (lowest cost) between any two products in terms of their cost difference.
* You can improve product recommendations by utilizing Dijkstra's algorithm to locate products with similar price differences. Users who are interested in a specific product can be recommended related products based on their price range. Customers looking for alternatives or products in each price range may find this technique useful.
* Dijkstra's algorithm identifies products with minimal cost differences, enabling shortest paths for market research, product recommendations, and price optimization.

## **PRIMS Algorithm:**

* Each client in this scenario can be visualized as a vertex in the graph. The price difference for shipping products from one customer to another is represented by the weighted edges between customers.
* You have discovered the minimum spanning tree (MST), which is the best route for providing goods to customers at the lowest cost, once all customers are considered.
* Applying Prim's algorithm allows you to connect all the customers in the graph while minimizing the overall delivery cost based on the price differences between them.
* In addition to this, applying this approach can help maximize product deliveries based on Amazon customer reviews.
* The resulting MST will give the delivery person an effective path to visit each customer while keeping the cost as low as possible.

## **Breadth-First Search (BFS) Algorithm:**

* The dataset is transformed into a graph with reviewers as nodes, with edges between nodes indicating similarity in product reviews.
* The algorithm starts with a reviewer and explores the graph by visiting neighbors at each level, marking visited nodes to avoid revisiting, and continuing until all nodes are visited.
* BFS identifies clusters of reviewers with similar reviews, indicating similar patterns or preferences. Influential reviewers have a high number of outgoing edges, while connectivity reveals overall network connectivity. It can identify isolated reviewers or groups with poor connections.

## **Hashing:**

* In my dataset I prefer to choose the chaining method to implement hashing technique. Each product in the context of product reviews can be allocated a unique key, such as Products Id, which is transformed to its specific ascii value.
* Hashing allows you to quickly find and obtain a certain review by entering its key into the hash function.
* Hashing can successfully handle huge datasets. You can rapidly discover duplicates and avoid keeping redundant information by hashing the reviews and comparing their hash values.
* If you want to search all reviews linked to a specific product, you can use the product's ID as a key to quickly retrieve the accompanying reviews. This enables rapid filtering and searching for specific attributes.

# **Conclusion:**

The Amazon product review dataset project analyzed various data structures, revealing their diverse applications and effectiveness in solving specific problems. These structures, like heaps, AVL trees, Dijkstra's algorithm, BFS, sentiment analysis, and hashing, were used for top-K recommendations, efficient searching, and sentiment analysis. Understanding their strengths and limitations is crucial for analyzing large-scale datasets and extracting valuable insights.