DATA STRUCTURES

(DS) CSE23304

EXPERIENTIAL LEARNING

C.V. RAMAN GLOBAL UNIVERSITY,
BHUBANESWAR,

ODISHA, INDIA

CASE STUDY ON:

DATA STRUCTURES USED IN

GOOGLE MAPS

UNDER THE SUPERVISION OF

ASST. PROFESSOR

DR. PRIYANKA KUMARI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY: -

NAME	REG NO.	GROUP
SUBHASHREE PATRA	2201020794	CS & IT 8A
JYOTI PATRA	2201020813	
INDRANEEL MOHAPATRA	2201020822	
KADAMBINI BEHERA	2201020835	
SAI SRUJANI MISHRA	2201020879	
RAKESH KUMAR	2201020880	

ACKNOWLEDGEMENT

I would like to express my gratitude ana appreciation to **my DS Faculty Teacher, Dr. Priyanka Kumari mam**, from the **Department of Computer Science and Engineering**, who gave me the golden opportunity to do this case study of DS on **Data structures used in google maps** and also for providing constant guidance and support in completing my project. Through this project, my concept and understanding regarding Algorithm of DS used in google maps became clearer and more accurate. I would also like to thank my teammate friends who helped me a lot in finalizing this project within the limited time frame.

CONTENTS

INTRODUCTION

Google Maps is a popular navigation tool used by millions of people worldwide to find directions, explore new places, and plan their routes.

However, navigating through complex road networks and finding the shortest or fastest route can be challenging, especially in urban areas with high traffic congestion or frequent road closures. Efficient navigation and route planning are crucial for providing accurate and reliable directions to users in real-time.

Data structures play a critical role in addressing these challenges by providing efficient algorithms for representing and manipulating geographic data.

Google Maps uses various data structures like graphs, trees, and hash tables to store and process information about roads, intersections, landmarks, and other geographic features. These data structures enable fast and accurate route planning, real-time traffic updates, and personalized recommendations based on user preferences.

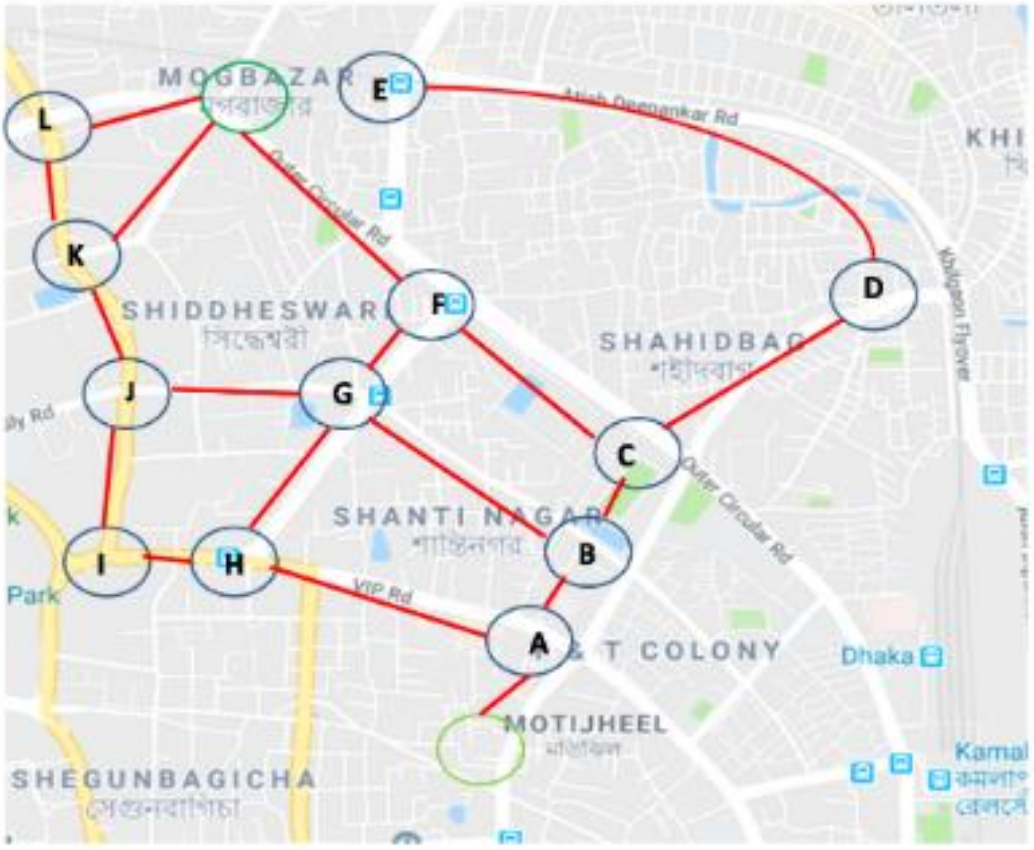
This report provides an overview of the role of data structures in Google Maps. It discusses the benefits and challenges of using different data structures for efficient navigation and route planning.

Finally, the report concludes with a discussion of future research directions and potential improvements to data structures in Google Maps.

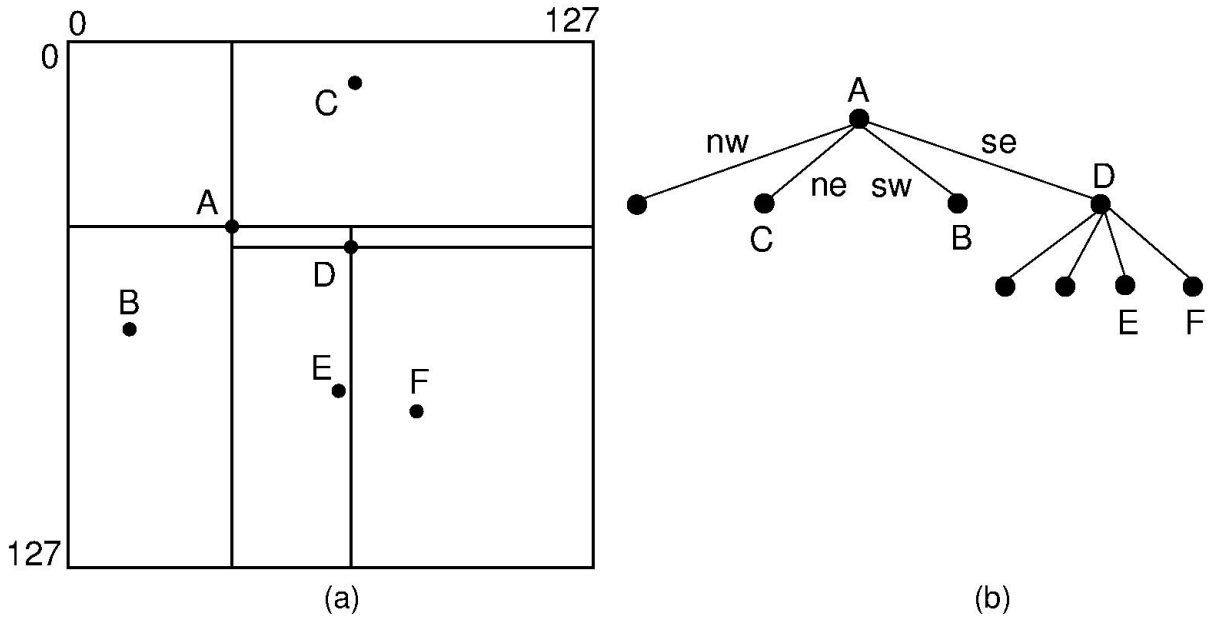
DATA STRUCTURES USED IN GOOGLE MAPS

Google Maps relies on a variety of data structures to efficiently store, manage, and process geographic and location-based data. Some of the key data structures used in Google Maps include Graphs, Hash Tables, Priority Queues/Heaps, Spartial Data Structures, Quad-Trees, Geohashes, Trie Data Structures.

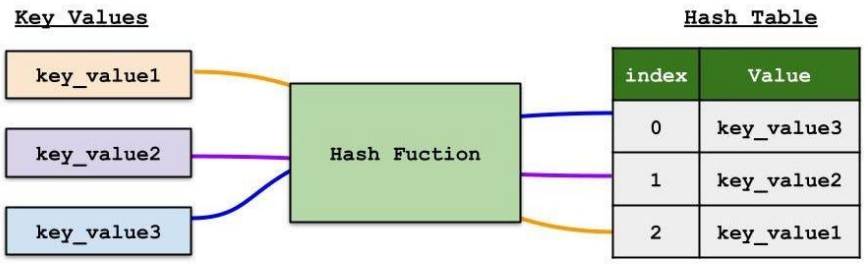
1. **Graphs:** -Graphs are used to represent road networks in Google Maps. Nodes in the graph represent intersections or locations, while edges represent the roads connecting them.



2. **Spartial Data Structures: -** Spatial data structures are employed to manage geographical data in Google Maps. These structures, such as quad trees or R-trees, organize spatial data based on their location coordinates. They enable efficient spatial indexing and querying, allowing for fast retrieval of nearby points of interest or road segments during route planning and navigation.

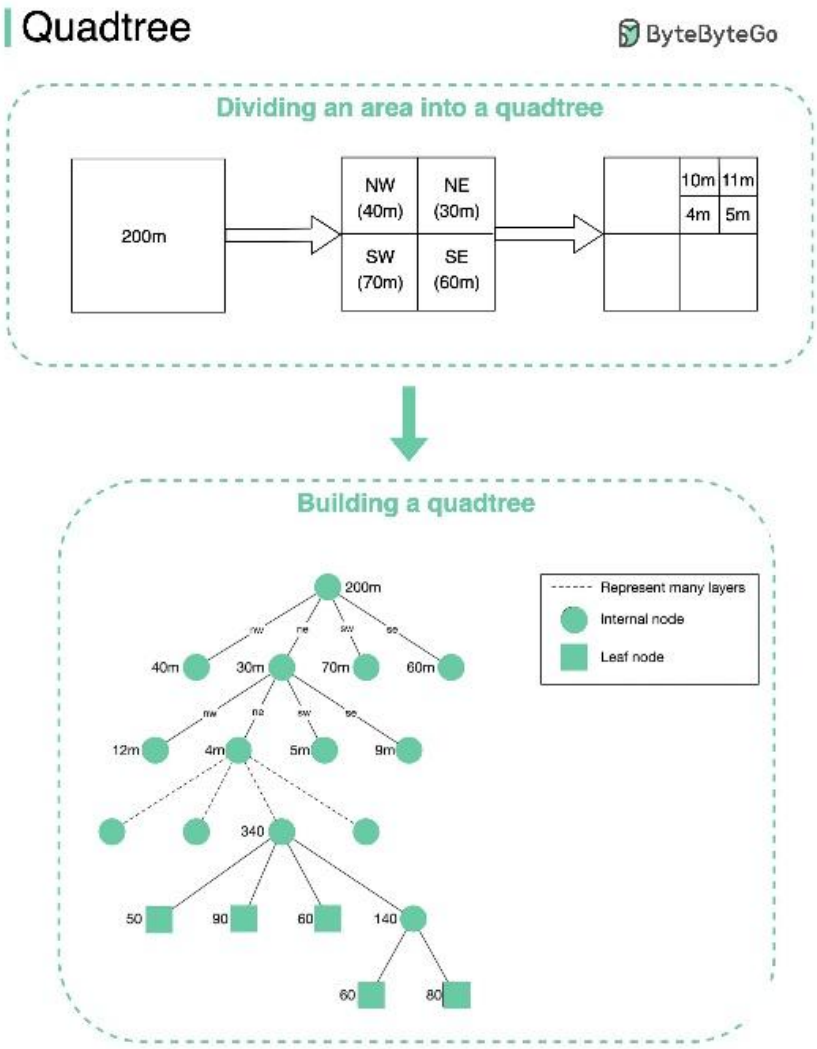


3. **Hash Tables: -** Hash tables are employed in Google Maps for efficient data storage and retrieval. They are used to store information about points of interest, addresses, and other relevant data. Hash tables provide fast access to data based on a unique key, enabling quick retrieval of information during route planning and navigation.



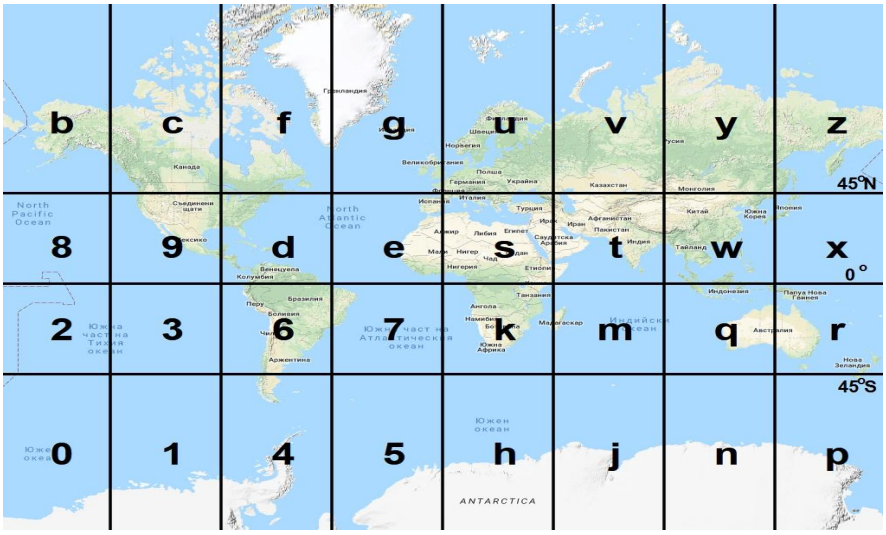
4. **Quad-Trees: -** Quad-trees are hierarchical spatial data structures that help divide geographic regions into smaller cells for efficient indexing and searching. Google Maps uses quad-trees for tasks like location-based search and map display.

   They enable efficient storage, indexing, and retrieval of map tiles and other geographic data, ensuring that users can explore maps at different zoom levels while minimizing data loading times and computational overhead.

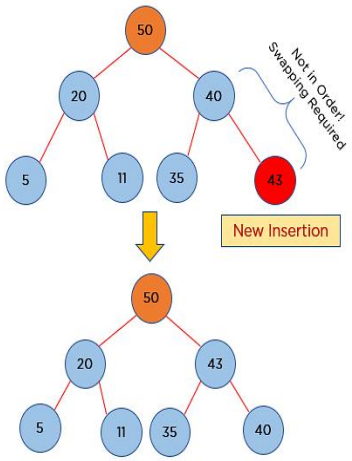


5. **Geo-Hashes: -** Geohash is a geocoding system that encodes geographic coordinates into a short string of letters and digits. It's used for indexing and efficient retrieval of location-based data.

   Google Maps, along with many other location-based services, relies on geohashes as a fundamental tool for organizing and processing geographic data efficiently and accurately. Geohashes offer a versatile and practical solution for location-based applications.



6.Priority Queues/Heaps: - Priority queues or heaps play a crucial role in optimal route calculation in Google Maps. These data structures prioritize the order in which nodes are visited during route planning based on factors such as distance, travel time, or traffic conditions. By using priority queues or heaps, Google Maps can determine the most optimal route efficiently.



The built in operation **Heapify** manages the shuffling of element in **Heap** data structure.

ALGORITHM and optimization

Dijkstra's algorithm is a fundamental graph search algorithm used in Google Maps for finding the shortest and fastest routes between locations.

Steps: -

- **Graph Representation:** Represents road networks as a weighted, directed graph.
- **Initial Setup:** Initializes with start and end points and assigns tentative distances.
- **Main Loop:** Repeatedly selects unvisited node with the smallest tentative distance.
- **Exploration of Neighbours:** Explores neighbours, calculates tentative distances, and updates values.
- **Marking as Visited:** Marks nodes as visited once explored.
- **Shortest Path Reconstruction:** Traces back from destination to source for the optimal route.
- **Real-Time Traffic:** Integrates real-time traffic data for fastest routes.
- **Alternate Routes**: Computes alternate routes for user choice.
- **Intermodal Routing:** Supports multiple modes of transportation.
- **Multi-Stop Directions**: Optimizes routes for multiple stops.

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define V 6 // Number of vertices in the graph

// Function to find the vertex with the minimum distance value
int minDistance(int dist[], int visited[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++) {
        if (visited[v] == 0 && dist[v] <= min) {
            min = dist[v];
            min_index = v;
        }
    }
    return min_index;
}

// Function to print the shortest path from source to the given vertex
void printPath(int parent[], int vertex) {
    if (parent[vertex] == -1) {
        printf("%d ", vertex);
        return;
    }
    printPath(parent, parent[vertex]);
    printf("%d ", vertex);
}

// Function to calculate and print the shortest paths using Dijkstra's Algorithm
void dijkstra(int graph[V][V], int source) {
    int dist[V]; // Array to store the distance from source to each vertex
    int visited[V]; // Array to keep track of visited vertices
    int parent[V]; // Array to store the parent of each vertex in the shortest path

    // Initialize distance, visited, and parent arrays
    for (int i = 0; i < V; i++) {
        dist[i] = INT_MAX;
        visited[i] = 0;
        parent[i] = -1;
    }

    // Distance of source vertex from itself is always 0
    dist[source] = 0;

    // Find shortest path for all vertices
    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, visited);
        visited[u] = 1;
```

```c
        // Update dist value of the adjacent vertices of the chosen vertex
        for (int v = 0; v < V; v++) {
            if (!visited[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
                parent[v] = u;
            }
        }
    }

    // Print the shortest paths from the source vertex to all other vertices
    for (int i = 0; i < V; i++) {
        printf("Shortest path from %d to %d: ", source, i);
        printPath(parent, i);
        printf("\n");
    }
}

// Main function
int main() {
    int graph[V][V] = {
        {0, 4, 0, 0, 0, 0},
        {4, 0, 8, 0, 0, 0},
        {0, 8, 0, 7, 0, 4},
        {0, 0, 7, 0, 9, 14},
        {0, 0, 0, 9, 0, 10},
        {0, 0, 4, 14, 10, 0}
    };

    int source = 0; // Source vertex

    dijkstra(graph, source);

    return 0;
}
```

Output

```
Shortest path from source 0 to source 0: 0
Shortest path from source 0 to source 1: 0 1
Shortest path from source 0 to source 2: 0 1 2
Shortest path from source 0 to source 3: 0 1 2 3
Shortest path from source 0 to source 4: 0 1 2 5 4
Shortest path from source 0 to source 5: 0 1 2 5

--------------------------------
Process exited after 0.03563 seconds with return value 0
Press any key to continue . . .
```

EXPERIMENTAL ANALYSIS

The performance and efficiency of the data structures and algorithms used in Google Maps have been extensively evaluated and optimized to provide fast and accurate route calculations. Here is an analysis of their performance:

1. **Speed and Efficiency**: Google Maps utilizes highly optimized data structures and algorithms to ensure efficient route calculations. The use of graphs allows for fast pathfinding algorithms such as Dijkstra's algorithm or A* algorithm to find the shortest or fastest routes between locations.

2. **Route Calculation Accuracy**: Google Maps is known for its accurate route calculations, taking into account factors like distance, travel time, traffic conditions, and real-time updates. The combination of data structures like graphs, priority queues, and spatial data structures allows for accurate representation and processing of geographical data.

3. **Handling Large-Scale Mapping Data**: To handle the vast amount of mapping data, Google Maps employs various optimizations. These include techniques like data partitioning, distributed computing, and caching mechanisms. By dividing the data into smaller partitions and distributing the processing across multiple servers, Google Maps can handle large-scale mapping data efficiently. Caching mechanisms also help reduce the computational load by storing frequently accessed data in memory for faster retrieval.

Continuous optimizations and improvements ensure that Google Maps can handle the ever-increasing volume of mapping data while maintaining a high level of performance and accuracy.

ADVANTAGES AND DISADVANTAGES OF PROPOSED APPROACH

The proposed approach of using data structures and algorithms in Google Maps offers several advantages, including:

1. **Faster Route Planning**: The use of optimized data structures and algorithms enables faster route planning, ensuring that users receive accurate directions in real-time.

2. **Real-time Updates**: The combination of data structures like graphs and priority queues enables real-time updates of traffic conditions, allowing Google Maps to suggest alternate routes to avoid congestion or road closures.

However, there are also potential disadvantages or limitations to this approach, such as:

1. **High Data Storage Requirements**: The vast amount of mapping data requires significant storage resources, which can be costly and challenging to manage.

2. **Occasional Inaccuracies in Traffic Prediction**: While Google Maps uses real-time traffic updates to provide accurate directions, occasional inaccuracies in traffic prediction can still occur, leading to suboptimal routes.

Overall, the advantages of using data structures and algorithms in Google Maps outweigh the potential disadvantages, making it a reliable and efficient navigation tool for users worldwide.

CONCLUSION

In conclusion, this report has highlighted the key findings and takeaways regarding the significance of data structures in enhancing Google Maps:

1. Data structures play a crucial role in addressing the challenges of efficient navigation and route planning in Google Maps. Graphs, hash tables, priority queues/heaps, and spatial data structures enable fast and accurate calculations, real-time updates, and efficient management of geographical data.

2. The use of these data structures in Google Maps leads to faster route planning, real-time updates on traffic conditions, and scalability for global usage. These advantages contribute to a seamless and reliable navigation experience for users.

3. However, there are areas for future research and improvements. One potential area is the exploration of advanced algorithms for route optimization, considering multiple factors like traffic congestion, road conditions, and user preferences. Additionally, enhancing the accuracy of traffic prediction algorithms can further improve the reliability of route recommendations.

In conclusion, the significance of data structures in enhancing Google Maps cannot be overstated. By leveraging efficient data structures and algorithms, Google Maps has transformed the way people navigate and plan their routes. With continuous research and improvements, Google Maps will continue to provide users with faster, more accurate, and personalized navigation experiences.