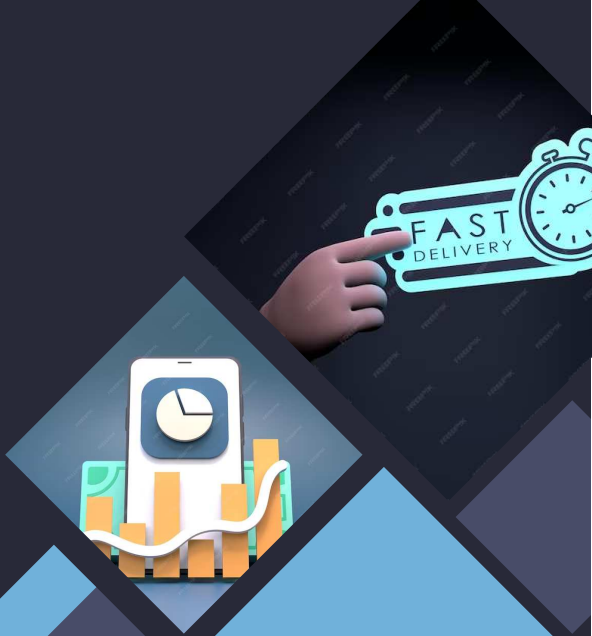
Travel Planner with Dijkstra's Algorithm

Utsah Shanker
210029011O178

Vansh Sharma
210029011O185



Introduction

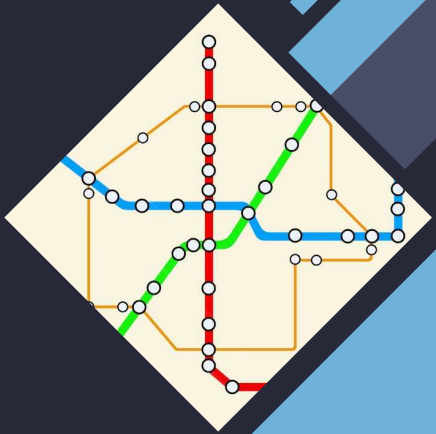
In this presentation, we will explore **Dijkstra's Algorithm** for finding the *shortest route* in a graph. We will understand the principles and applications of this powerful algorithm in various fields. Let's embark on the journey of optimal path discovery!

Time Complexity:- O((V+E)*logV)

Space Complexity:- O(V+E)

Understanding Dijkstra's Algorithm

Dijkstra's Algorithm, named after computer scientist **Edsger W. Dijkstra**, is a method for finding the *shortest path* between nodes in a graph. It uses a **greedy approach** to iteratively determine the shortest path from a single source vertex to all other vertices. This algorithm is widely used in network routing and transportation planning.



Key Components of Dijkstra's Algorithm

The key components of Dijkstra's Algorithm include the **priority queue** for efficient selection of the next vertex, a **distance array** to store the shortest distance from the source, and a **predecessor array** to reconstruct the shortest path. Understanding these components is essential for implementing and optimizing the algorithm.

Applications in Real-world Scenarios

Dijkstra's Algorithm has diverse applications in real-world scenarios, including *GPS navigation systems*, *network routing*, *logistics and supply chain management*, and *telecommunications*. By finding the shortest path, this algorithm plays a crucial role in optimizing transportation, communication, and resource allocation.

```
pair<int, vector<string>> dijkstra(string start, string end)
{
    map<string, int> distance;
    map<string, string> parent;
    set<pair<int, string>> pq;

    for (const auto &pair : locations)
    {
        string locName = pair.first;
        distance[locName] = numeric_limits<int>::max();
        pq.insert({numeric_limits<int>::max(), locName});
    }

    distance[start] = 0;
    pq.insert({0, start});

    while (!pq.empty())
    {
        string current = pq.begin()->second;
        pq.erase(pq.begin());

        for (const Edge &neighbor : locations[current].neighbors)
        {
            int alt = distance[current] + neighbor.distance;
            if (alt < distance[neighbor.destination])
            {
                pq.erase({distance[neighbor.destination], neighbor.destination});
                distance[neighbor.destination] = alt;
                parent[neighbor.destination] = current;
                pq.insert({alt, neighbor.destination});
            }
        }
    }
```

```
    int shortestDistance = distance[end];
    vector<string> path;
    string current = end;
    while (current != start)
    {
        path.push_back(current);
        current = parent[current];
    }
    path.push_back(start);
    reverse(path.begin(), path.end());

    return make_pair(shortestDistance, path);
}
```

Output

```
/tmp/DgN739oqCK.o
Enter start location: A
Enter end location: F
Shortest distance from A to F: 7
Shortest path: A -> C -> F
```

Conclusion

In conclusion, Dijkstra's Algorithm is a fundamental tool for *finding the shortest route* in various domains. Its impact on transportation, communication, and optimization is profound. By understanding the principles and applications of this algorithm, we can navigate complex networks and discover optimal paths in diverse real-world scenarios.

Thanks!