MODULE 2 (B17)

Route map navigation using Floyd Warshall algorithm

# Description of the module :

1. Basic Idea: Module 2 of our project focuses on the implementation of the Floyd-Warshall algorithm in Java. The Floyd-Warshall algorithm is a well-known graph algorithm used for finding the shortest path between all pairs of vertices in a weighted graph. In the context of your project, this algorithm is essential for calculating the shortest distances between various locations within your campus.
2. Need for the Module: The implementation of the Floyd-Warshall algorithm in Module 2 is necessary to address the following requirements:
3. Efficient Distance Calculation: Your route map navigation system aims to find the shortest distance between any two points within your campus. The Floyd-Warshall algorithm is a suitable choice for this task as it efficiently calculates the shortest distances for all pairs of vertices in a graph. By implementing this algorithm, you ensure that your navigation system can provide accurate and optimal route suggestions to users.
4. All-Pairs Shortest Path: The Floyd-Warshall algorithm's unique feature is its ability to calculate the shortest path between all pairs of vertices in a graph. This means that for any given location within your campus, the algorithm can determine the shortest distances to all other locations. This comprehensive approach allows users of your navigation system to receive information on the shortest routes from their current location to any desired destination, considering all possible paths.
5. By following this development approach, you can implement Module 2 to efficiently calculate the shortest distances between locations within your campus using the Floyd-Warshall algorithm. This module forms the foundation of your route map navigation system, providing the core data necessary for accurate and optimal routing in Module 3.

# Motivation for the module :

Module 2, which focuses on the implementation of the Floyd-Warshall algorithm in Java, serves as a fundamental component of your route map navigation project. The motivation behind Module 2 can be understood in the following ways:

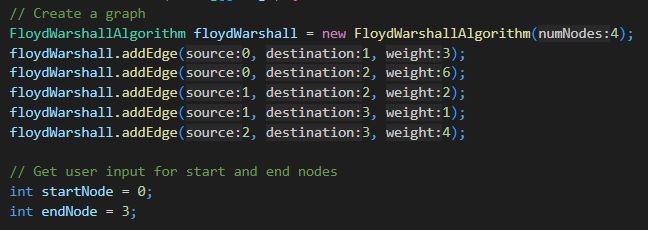
1. Efficient Shortest Distance Calculation: The core functionality of your navigation system relies on finding the shortest distance between locations within your campus. The Floyd-Warshall algorithm is well-known for its ability to efficiently calculate the shortest distances between all pairs of vertices in a weighted graph. By implementing this algorithm in Module 2, you can ensure that your navigation system can provide accurate and optimal route suggestions to users, saving them time and effort.
2. All-Pairs Shortest Path Calculation: The Floyd-Warshall algorithm's unique feature lies in its ability to calculate the shortest path between all pairs of vertices in a graph. In the context of your project, this means that users of your navigation system can receive information on the shortest routes from any given location to all other locations within your campus. This comprehensive approach provides users with a broader perspective of the possible routes, enabling them to make informed decisions about their navigation choices.
3. In summary, the motivation for Module 2 revolves around efficient shortest distance calculation, all-pairs shortest path calculation, flexibility in handling weighted graphs, scalability for campus-scale navigation, and integration with Module 3. By implementing the Floyd-Warshall algorithm in Module 2, you lay the groundwork for the core functionality of your route map navigation system, enabling accurate and optimal routing within your campus environment.

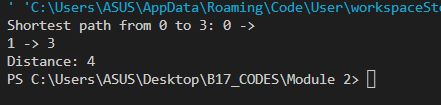
# Relevance of the module in entire system :

Module 1, which focuses on the implementation of the Floyd-Warshall algorithm in Java, holds significant relevance in the overall route map navigation system. Its relevance can be understood within the context of the entire system in the following ways:

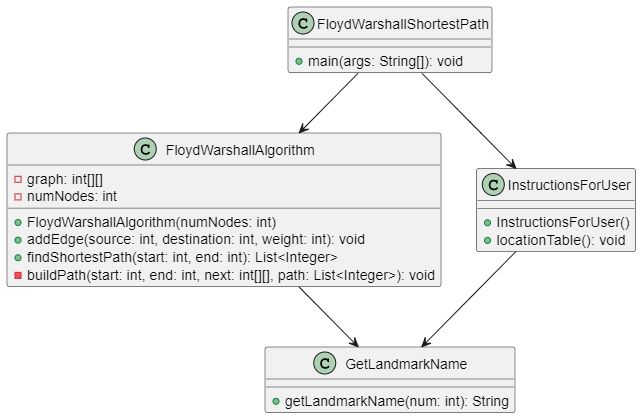
1. Core Functionality: Module 1 provides the core functionality of the route map navigation system by implementing the Floyd-Warshall algorithm. This algorithm is essential for calculating the shortest distance between any two locations within your campus. By accurately determining the shortest distances, Module 1 forms the foundation for the entire navigation system, ensuring the system's primary objective of finding optimal routes.
2. Efficient Distance Calculation: The Floyd-Warshall algorithm's efficiency lies in its ability to calculate the shortest distances between all pairs of vertices in a graph. In the context of your project, this translates to efficient distance calculation between various locations within your campus. By implementing this algorithm in Module 1, you ensure that the navigation system can provide users with the most efficient routes, saving time and effort in their navigation endeavors.
3. In conclusion, Module 2 relevance in the entire route map navigation system is significant. Its implementation of the Floyd-Warshall algorithm provides the core functionality, efficient distance calculation, all-pairs shortest path generation, essential data generation, and seamless integration with Module 3. Module 2 forms the foundation of the navigation system, driving accurate and optimal routing within your campus environment.

# Input and Output :





# Appendix :



*1. FloydWarshallAlgorithm -> FloydWarshallAlgorithm :*

This is the constructor of the FloydWarshallAlgorithm class. It initializes the class with the number of nodes in the graph. It creates a 2D array graph to represent the weighted graph, where numNodes defines the size of the array.

*2.addEdge -> addEdge :*

This function is used to add an edge to the graph. It takes the source vertex, destination vertex, and weight of the edge as parameters. It sets the weight of the edge between the source and destination vertices in the graph array.

*3.findShortestPath -> findShortestPath :*

This function finds the shortest path between two nodes using the Floyd-Warshall algorithm. It takes the starting node (start) and the ending node (end) as parameters. The function initializes the dist and next arrays, and then performs the Floyd-Warshall algorithm to calculate the shortest distances and next vertex information. Finally, it prints the shortest path from the start node to the end node, along with the distance of the shortest path.

*4. printPath -> printPath :*

This is a helper function used by findShortestPath to print the path from the start node to the end node. It takes the start node, end node, and the next array as parameters. It recursively prints the path by following the next array to determine the intermediate vertices along the shortest path.

*5.FloydWarshall Main ->FloydWarshallMain:*

This is the entry point of the program. It creates an instance of the FloydWarshallAlgorithm class, adds edges to the graph, and specifies the start and end nodes for finding the shortest path. It then calls the findShortestPath function to calculate and print the shortest path and distance.

MODULE PRESENTED BY : PRANAY [CB.EN.U4AIE22143]