Algorithm Design and Analysis  
Project  
Code Documentation  
I210874 Muhammad Mobin

I210827 Minhaj Khalil  
I210895 Muhammad Shahnawaz

I200418 Raja Behzad

Edge Struct

Purpose: Represents an edge between nodes in the graph.

Attributes:

src: Source node index of the edge.

dest: Destination node index of the edge.

weight: Weight/cost of the edge.

Node Class

Purpose: Represents a node in the graph.

Attributes:

data: String data stored in the node.

next: Pointer to the next node in a linked list.

Relationship Class

Purpose: Represents a relationship between nodes in the graph.

Attributes:

type: Type of relationship.

weight: Weight/cost associated with the relationship.

next: Pointer to the next relationship in a linked list.

Event Class

Purpose: Represents an event associated with a node.

Attributes:

eventName: Name of the event.

attendance: Number of participants attending the event.

Graph Class

Purpose: Represents a graph data structure.

Attributes:

nodes: Array of nodes in the graph.

relationships: 2D array representing relationships between nodes.

events: Array of events associated with nodes.

allEdges: Array to store all edges in the graph.

parent: Array to store parent nodes in disjoint set operations.

numEdges: Counter to keep track of the number of edges in the graph.

Methods:

Graph(int numNodes): Constructor to initialize the graph with a specified number of nodes.

~Graph(): Destructor to free allocated memory.

addNode(int index, const string& nodeName): Method to add a node to the graph.

addRelationship(int index1, int index2, const string& relationshipType, int edgeWeight): Method to add a relationship between nodes.

addEvent(int index, const string& eventName): Method to add an event associated with a node.

attendEvent(int index): Method to simulate attendance at an event.

displayEventAttendance(int index): Method to display event attendance for a specific node.

displayRelationships(int index): Method to display relationships of a node.

prim(): Method to find and print Minimum Spanning Tree (MST) using Prim's algorithm.

dijkstra(int src): Method to find and print shortest paths using Dijkstra's algorithm.

DFS(int startNode, bool visited[]): Method to perform Depth-First Search starting from a given node.

detectCommunities(): Method to detect communities using DFS.

addEdgeToList(int src, int dest, int weight): Method to add an edge to the list for Kruskal's algorithm.

kruskal(): Method to find and print Minimum Spanning Tree (MST) using Kruskal's algorithm.

main() Function

Purpose: Entry point of the program where the graph functionalities are demonstrated.

Comments

Comments are added throughout the code to explain the purpose and functionality of methods, loops, and key operations.