GraphM and GraphL Documentation

Overview

The GraphM class is a C++ implementation of a weighted directed graph. It utilizes matrices to represent the graph's cost matrix, distance matrix, and visited matrix. The class provides methods for building the graph from an input file, inserting and removing edges, finding the shortest paths between nodes, and displaying relevant information about the graph. The GraphL class is a C++ implementation of an adjacency list representation of a directed graph. It uses an array of GraphNode structures, where each structure contains information about a node and a linked list of adjacent nodes. The class provides methods for building the graph from an input file, displaying the graph, performing depth-first search, and clearing the graph.

GraphM Class Members

- Data Members:
  - size: Represents the number of nodes in the graph.
  - C[MAXNODES][MAXNODES]: Cost matrix representing edge weights between nodes.
  - T[MAXNODES][MAXNODES]: Structure matrix storing distance, visited status, and path information.
- Methods:
  - Constructor (GraphM):
    - Initializes the graph with a size of 0.
    - Initializes cost matrix, distance matrix, and visited matrix.
  - buildGraph(ifstream& infile):
    - Reads the size of the graph from the input file.
    - Reads and stores locations in the 'data' array.
    - Reads edges and weights to insert into the graph.
  - insertEdge(int source, int destination, int weight):
    - Inserts an edge into the graph with a given source, destination, and weight.
  - removeEdge(int source, int destination):
    - Removes an edge from the graph with a given source and destination.
  - checkEdge(int source, int destination, int weight):
    - Checks if an edge exists between source and destination with a given weight.
  - findShortestPath():
    - Finds the shortest paths from each node to every other node in the graph using Dijkstra's algorithm.
  - displayAll():
    - Displays all shortest paths, distances, and paths between nodes.

display(int source, int destination):
- Displays the shortest path, distance, and path data between a given source and destination.

printData(int source, int destination):
- Recursively prints data along the shortest path between source and destination.

printPath(int source, int destination):
- Recursively prints the shortest path between source and destination.

displayAdjacencyMatrix():
- Displays the adjacency matrix representing the edges and their weights.

displayLocations():
- Displays the locations stored in the 'data' array.

GraphL Class Members

- Data Members:
  size: Represents the number of nodes in the graph.
  nodeArray: An array of GraphNode structures, where each structure contains information about a node and its adjacency list.
- Methods:
  Constructor (GraphL):
  - Initializes the graph with a size of 0 and a null nodeArray.

  Destructor (~GraphL):
  - Deallocates memory used by the graph, calling clearGraph().

  buildGraph(ifstream& infile):
  - Reads the size of the graph from the input file.
  - Allocates memory for the array of GraphNode structures.
  - Reads data for each node and builds the adjacency list.

  displayGraph():
  - Displays the graph information, including nodes and their adjacency lists.

  depthFirstSearch():
  - Initiates a depth-first search on the graph, outputting the depth-first ordering of nodes.

  depthFirstSearchHelper(int node):
  - Recursively performs depth-first search starting from the given node.

  clearGraph():
  - Deallocates memory used by the graph, including the adjacency lists.

Testing

I thoroughly tested the code using different scenarios, trying various sizes of graphs with different edge weights and configurations. I specifically checked if the code correctly printed specific paths, displayed all shortest paths, and performed depth-first search. In all tests, the code worked as expected, and I didn't find any issues with memory leaks.