Programmer Manual

Digraph Class

A. Class Description

The Diagraph class is a templated implementation of a directed graph using an adjacency list. It provides member functions for adding and deleting vertices and edges, checking for their existence, and performing searches. The class also includes functions for reading graphs from input files and prompting the user for input.

B. Class standings

1.Protected Section:

- Int num_of_verticies: integer representing the number of vertices in the graph
- std::vector< vertex<V, W> > graph_array: graph being a vector

2. Public Section:

- Digraph() and ~Digraph(): constructor and destructor for the Digraph class.
- set_size(ifstream& inFile): sets the number of vertices in the graph by reading from an input file.
- getDigraph(ifstream& inFile): reads in the vertices and edges of the graph from an input file.
- get_size(): returns the number of vertices in the graph.
- isVertex(V& v): checks if a vertex with name v exists in the graph.

- isUniEdge(V& v1, V& v2): checks if there is a unidirectional edge from vertex v1 to vertex v2.
- isBiDirEdge(V& v1, V& v2): checks if there is a bidirectional edge between vertices v1 and v2.
- addVertex(V& v): adds a new vertex with name v to the graph.
- deleteVertex(V& v): removes the vertex with name v from the graph.
- addUniEdge(V& v1, V& v2, W& w): adds a new unidirectional edge from vertex v1 to vertex v2 with weight w.
- addBiDirEdge(V& v1, V& v2, W& w): adds a new bidirectional edge between vertices v1 and v2 with weight w.
- deleteUniEdge(V & v1, V& v2, W& w): removes the unidirectional edge from vertex v1 to vertex v2 with weight w.
- deleteBiDirEdge(V& v1, V& v2, W& w): removes the bidirectional edge between vertices v1 and v2 with weight w.
- printDigraph(): prints out the vertices and edges of the graph.
- breadth(V& start_vertex): performs a breadth-first search starting from the vertex start_vertex.
- depth(V& start_vertex): performs a depth-first search starting from the vertex start_vertex.
- getOneVertex(V& v1): reads in a single vertex name from the user.
- getTwoVerticies(V& v1, V& v2): reads in two vertex names from the user.

C. High Level Program Solution

D. Limitations and Suggestions

Limitations:

- The implementation assumes that an Adjacency List representation is being used for the graph.
- The class only supports weighted edges.
- The graph does not have a way to store data associated with each vertex or edge.

Suggestions:

- Add support for other types of graph representations (e.g., Adjacency Matrix).
- Add support for unweighted edges.
- Consider adding member functions to store and retrieve data associated with vertices and edges.