Discrete Mathematics

DIRECTED GRAPH

Perez Molina, Tomás

Contents

1 Assignment

1. Specify a directed non-weighted graph.
2. Implement and test a directed non-weighted graph with an edges linked list. Calculate the order of each operation.
3. To test the correct functionality of the previous class:
   (i) Implement a function to load data to a graph (for both vertices and edges).
   (ii) Implement a function to generate a random graph.
   (iii) Implement a function to show the graph on screen (either a list with vertices and edges or a graph drawing).
4. Implement all three traversal algorithms (Plain search, BFS and DFS)
5. Modifying previous functions, write and test algorithms that:
   (i) Calculate the amount of sources and sinks.
   (ii) Verify if the graph is weakly connected.
   (iii) Given two vertices, verify if a path of length 2 exists between them.
   (iv) Implement Warshall's algorithm.

2 Specification

2.1 DirectedGraph

Description: Represents a directed non-weighted graph.

DirectedGraph: IdMaterial x Type x Title x Author x Year → Material
Description: Creates a new Material with the given data.
Precondition:
o Type is either 1 or 2.
o IdMAterial ≥ 1
Postcondition: A Material is created.
Classification: Constructor

destroyMaterial: Material → void
Description: Deallocates all memory assigned to the Material.
Precondition: Material must exist.
Postcondition: Memory freed.
Classification: Destructor

2.2 Graph Related Algorithms

Description: Algorithms related to graph theory

getSources: Graph → Number
Description: Given a graph returns the number of source vertices.
Precondition: Graph must exist
Postcondition: Return the amount of source vertices in the graph
Classification: Analyzer

getSinks: Graph → Number
Description: Given a graph returns the number of sink vertices.
Precondition: Graph must exist
Postcondition: Return the amount of sink vertices in the graph
Classification: Analyzer

isWeaklyConnected: Graph → Boolean
Description: Given a graph, checks whether it is weakly connected.
Precondition: Graph must exist
Postcondition:

- True if the graph is weakly connected.
- False if the graph is not weakly connected.

Classification: Analyzer

pathOfLengthTwoExists: Graph x Vertex x Vertex → Boolean
Description: Given a graph and two vertices checks if a path of length 2 exists between them in the graph.
Precondition: Graph and both vertices must exist
Postcondition:

- True if a path of length two between the two vertces exists.
- False if a path of length two between the two vertces does not exist.

Classification: Analyzer

warshall: Graph → Transitive Closure
Description: Given a graph calculates its transitive closure using Warshall's.
Precondition: Graph must exist
Postcondition: The transitive closure must have the capability of calculating whether there is a path between two vertices in O(1) time.
Classification: Analyzer