ADT < GRAPH >
< adjacencyList, time >
Graph (adjacencyList ϵ List of nodes, time ϵ Integer)
* Primitive Operations: * addVertex → T → Void * addEdge → T x T → Void * removeEdge → T x T → Void * removeVertex → T → Void * searchNode → T → Node<T> * DFS → ()→ Integer * BFS → T → Boolean * dijkstra → T → Integer[][] * floydWarshall → () → Integer [ ] [ ] * isStronglyConnected → ()→ Boolean * prim → () → Void * kruskal → () → Void * toString → () → String * toStringAsMatrix → () → TtoStringAsMatrix → () → T

Operaciones:

addVertex(element)
"Adds a vertex with the specified element to the graph"

{Pre: element ϵ T}

{Post: Adds a vertex with the specified element to the graph}

addEdge(elementA, elementB)
"Adds an edge between the vertices with elements elementA and elementB"

{Pre: elementA, elementB ∈ T}

{Post: Adds an edge between the vertices with elements elementA and elementB}

removeEdge(elementA, elementB)
"Removes the edge between the vertices with elements elementA and elementB"

{Pre: elementA, elementB ∈ T}

{Post: Removes the edge between the vertices with elements elementA and elementB}

removeVertex(element)
"Removes the vertex with the specified element from the graph"

{Pre: element ∈ T}

{Post: Removes the vertex with the specified element from the graph}

searchNode(element)
"Searches and returns the graph node with the specified element"

{Pre: element ∈ T}

{Post: Returns the node with the specified element or null if not found}

DFS()
"Makes a Depth-First Search (DFS) on the graph and returns the number of DFS trees found"

{Pre: N/A}

{Post: Performs a DFS traversal on the graph and returns the number of DFS trees}

BFS(element)
"Makes a Breadth-First Search (BFS) on the graph starting from the node with the specified element"

{Pre: element ∈ T}

{Post: Performs a BFS traversal on the graph from the specified node and checks if the graph is connected}

dijkstra(element)
"Calculates the matrix of minimum distances using Dijkstra's algorithm from the node with the specified element"

{Pre: element ∈ T}

{Post: Calculates and returns a matrix of minimum distances or null if the graph is not connected}

floydWarshall()
"Calculates the matrix of minimum distances using the Floyd-Warshall algorithm on the graph"

{Pre: None}

{Post: Calculates and returns a matrix of minimum distances}

isStronglyConnected()
"Checks if the graph is strongly connected"

{Pre: None}

{Post: Checks if the graph is strongly connected and returns a boolean value}

prim()
"Applies the Prim's algorithm to find the minimum spanning tree"

{Pre: None}

{Post: Applies the Prim's algorithm on the graph}

kruskal()
"Applies the Kruskal's algorithm to find the minimum spanning tree"

{Pre: None}

{Post: Applies the Kruskal's algorithm on the graph}

toString()
"Returns a string representation of the graph"

{Pre: None}

{Post: Returns a string representing the graph}

toStringAsMatrix()
"Returns a string representation of the graph as an adjacency matrix"

{Pre: None}

{Post: Returns a string representing the graph as an adjacency matrix}