TAD < GRAPH >
< listaAyacencia, tiempo >
Graph (listaAyacencia ϵ Lista de nodos, tiempo ϵ Entero)
Operaciones Primitivas: ✩ addVertex → T → Void ✩ addEdge → T x T → Void ✩ removeEdge → T x T → Void ✩ removeVertex → T → Void ✩ searchNode → T → Nodo<T> ✩ DFS → () → Entero ✩ BFS → T → Booleano ✩ dijkstra → T → Matriz de Enteros ✩ floydWarshall → () → Matriz de Enteros ✩ isStronglyConnected → () → Booleano ✩ toString → () → Cadena ✩ toStringAsMatrix → () → T

Operaciones:

addVertex(elemento) “Agrega un vértice al grafo con el elemento especificado” {Pre: elemento ϵ T} {Pos: El vértice con el elemento especificado se agrega al grafo}

addEdge(elementoA, elementoB) “Agrega una arista entre los vértices con elementos especificados” {Pre: elementoA ϵ T ∧ elementoB ϵ T} {Pos: Se agrega una arista entre los vértices con elementos especificados}

removeEdge(elementoA, elementoB) “Elimina la arista entre los vértices con elementos especificados” {Pre: elementoA ϵ T ∧ elementoB ϵ T} {Pos: La arista entre los vértices con elementos especificados se elimina}

removeVertex(elemento) “Elimina un vértice del grafo con el elemento especificado” {Pre: elemento ϵ T} {Pos: El vértice con el elemento especificado se elimina del grafo}

searchNode(elemento)
“Busca y devuelve el nodo del grafo con el elemento especificado”

{Pre: elemento ϵ T}

{Pos: Se devuelve el nodo con el elemento especificado o null si no se encuentra}

DFS()
“Realiza un recorrido en profundidad (DFS) en el grafo y devuelve el número de árboles DFS encontrados”

{Pre: N/A}

{Pos: Se realiza un recorrido DFS en el grafo y se devuelve el número de árboles DFS}

BFS(elemento)
“Realiza un recorrido en amplitud (BFS) en el grafo desde el nodo con el elemento especificado”

{Pre: elemento ϵ T}

{Pos: Se realiza un recorrido BFS en el grafo desde el nodo especificado y se verifica si el grafo es conexo}

dijkstra(elemento)
“Calcula la matriz de distancias mínimas usando el algoritmo de Dijkstra desde el nodo con el elemento especificado”

{Pre: elemento ϵ T}

{Pos: Se calcula y devuelve una matriz de distancias mínimas o null si el grafo no es conexo}

floydWarshall()
“Calcula la matriz de distancias mínimas usando el algoritmo de Floyd-Warshall en el grafo”

{Pre: Ninguna}

{Pos: Se calcula y devuelve una matriz de distancias mínimas}

isStronglyConnected()
“Verifica si el grafo es fuertemente conexo”

{Pre: Ninguna}

{Pos: Se verifica si el grafo es fuertemente conexo y se devuelve un valor booleano}

toString()
“Devuelve una representación de cadena del grafo en forma de lista de adyacencia”

{Pre: Ninguna}

{Pos: Se devuelve una representación en cadena del grafo en forma de lista de adyacencia}

toStringAsMatrix()
“Devuelve una representación de cadena del grafo en forma de matriz de adyacencia”

{Pre: Ninguna}

{Pos: Se devuelve una representación en cadena del grafo en forma de matriz de adyacencia}