TAD <Grafo>
Graph = {nodes, edges, directed}
Primitive Operations: - CreateGraph(): -> Graph - AddNode(node): -> void - Add Edge(Origin, Destination, Weight): -> void - Get Nodes(): -> nodes - Get Edges(): -> edges - IsTargeted(): -> targeted - ExistsArista (origin, destination): -> bool - GetNeighbors(node): -> neighbors

TAD <Matriz de Adyacencia>
MatrizAdyacencia = {nodos, aristas, matriz}
Inv: matriz[i][j] = 1 si hay arista entre nodo i y nodo j, 0 en caso contrario
Primitive Operations: - CrearMatrizAdyacencia(n): -> MatrizAdyacencia - AgregarArista(origen, destino): -> void - ObtenerAristas(): -> aristas - ObtenerMatriz(): -> matriz

TAD <Lista de Adyacencia>
ListaAdyacencia = {nodos, lista}
Inv: lista[i] contiene los nodos adyacentes al nodo i
Primitive Operations: - CrearListaAdyacencia(n): -> ListaAdyacencia - AgregarArista(origen, destino): -> void - ObtenerVecinos(nodo): -> lista

TAD <BFS>
BFS = {grafo, visitados, cola}
Primitive Operations: - IniciarBFS(grafo): -> BFS - RealizarBFS(inicio): -> void - ObtenerRecorrido(): -> recorrido

TAD <DFS>
DFS = {grafo, visitados}

Primitive Operations: - IniciarDFS(grafo): -> DFS - RealizarDFS(inicio): -> void - ObtenerRecorrido(): -> recorrido

TAD <Dijkstra>
Dijkstra = {grafo, distancias, visitados}
- IniciarDijkstra(grafo): -> Dijkstra - CalcularDistancias(origen): -> void - ObtenerCamino(destino): -> camino - ObtenerDistancia(destino): -> distancia

TAD <Floyd-Warshall>
FloydWarshall = {grafo, distancias}
Primitive Operations: - IniciarFloydWarshall(grafo): -> FloydWarshall - CalcularDistancias(): -> void - ObtenerDistancia(origen, destino): -> distancia

Max Grafo

CreateGraph() "Create a new empty graph." {Pre: None} {Pos: Created Graph}

AddNode(Node)

"Add a node to the graph."

{Pre: node}

{Pos: Node added to graph}

Add Edge(Origin, Destination, Weight)

"Add an edge between the source and destination nodes with the given weight."

{Pre: Origin, Destination, Weight}

{Pos: Edge added to graph}

GetNodes()

"Returns the list of nodes in the graph."

{Pre: None}

{Pos: Node List}

GetEdges()

"Returns the list of edges in the graph."

{Pre: None}

{Pos: List of edges}

IsDirected()

"Indicates whether the graph is directed or not."

{Pre: None}

{Pos: Boolean value

There isArista (origin, destination)

"Check if there is an edge between the source and destination nodes."

{Pre: Origin, Destination}

{Pos: Boolean value}

GetNeighbors(node)

"Returns the list of neighboring nodes to the given node."

{Pre: node}

{Pos: Neighbor Node List}

Max Matriz de Adyacencia

CrearMatrizAdyacencia(n)

“Crea una matriz de adyacencia con n nodos.”

{Pre: n}

{Pos: Matriz de adyacencia creada}

AgregarArista(origen, destino)

“Agrega una arista entre los nodos origen y destino en la matriz#.

{Pre: origen, destino}

{Pos: Matriz actualizada con la arista}

ObtenerAristas()

“Retorna el número total de aristas en el grafo.”

{Pre: Ninguno}

{Pos: Número total de aristas}

ObtenerMatriz()

“Retorna la matriz de adyacencia.”

{Pre: Ninguno}

{Pos: Matriz de adyacencia}

Max Lista de adyacencia

CrearListaAdyacencia(n)

“Crea una lista de adyacencia con n nodos.”

{Pre: n}

{Pos: Lista de adyacencia creada}

AgregarArista(origen, destino)

“Agrega una arista entre los nodos origen y destino en la lista.”

{Pre: origen, destino}

{Pos: Lista actualizada con la arista}

ObtenerVecinos(nodo)

“Retorna la lista de nodos adyacentes al nodo dado.”

{Pre: nodo}

{Pos: Lista de nodos adyacentes}

Max BFS

IniciarBFS(grafo)

“Inicializa el algoritmo BFS con el grafo dado.”

{Pre: grafo}

{Pos: Algoritmo BFS inicializado}

RealizarBFS(inicio)

“Realiza el recorrido BFS empezando desde el nodo de inicio.”

{Pre: inicio}

{Pos: Recorrido BFS realizado}

ObtenerRecorrido()

“Retorna el recorrido BFS realizado.”

{Pre: Ninguno}

{Pos: Recorrido BFS}

Max DFS

IniciarDFS(grafo)

“Inicializa el algoritmo DFS con el grafo dado.”

{Pre: grafo}

{Pos: Algoritmo DFS inicializado}

RealizarDFS(inicio)

“Realiza el recorrido DFS empezando desde el nodo de inicio.”

{Pre: inicio}

{Pos: Recorrido DFS realizado}

ObtenerRecorrido()

“Retorna el recorrido DFS realizado.”

{Pre: Ninguno}

{Pos: Recorrido DFS}

Max Dijkstra

IniciarDijkstra(grafo)

“Inicializa el algoritmo Dijkstra con el grafo dado.”

{Pre: grafo}

{Pos: Algoritmo Dijkstra inicializado}

CalcularDistancias(origen)

“Calcula las distancias más cortas desde el nodo de origen.”

{Pre: origen}

{Pos: Distancias calculadas}

ObtenerCamino(destino)
“Retorna el camino más corto hasta el nodo destino.”
{Pre: destino}
{Pos: Camino más corto}

ObtenerDistancia(destino)
“Retorna la distancia más corta hasta el nodo destino.”
{Pre: destino}
{Pos: Distancia más corta}

Max Floyd-Warshall

IniciarFloydWarshall(grafo)
“Inicializa el algoritmo Floyd-Warshall con el grafo dado.”
{Pre: grafo}

{Pos: Algoritmo Floyd-Warshall inicializado}

CalcularDistancias()

“Calcula todas las distancias entre cada par de nodos.”

{Pre: Ninguno}

{Pos: Distancias calculadas}

ObtenerDistancia(origen, destino)

“Retorna la distancia entre el nodo de origen y el nodo destino.”

{Pre: origen, destino}

{Pos: Distancia entre nodos}