Universidad Icesi
Departamento de Computación y Sistemas Inteligentes

Scenario and Use Case Format

Configuración de los Scenarios

Name	Class	Scenario
setUpGraphDirectionalEmpty	GraphTest	adjacencyList = new HasMap<>();
setUpGraphDirectionalWithElements	GraphTest	adjacencyList != null
setUpGraphRoute	GraphTest	A directed graph is created with nodes: House, Hotel, Hospital, Bank, Casino, Prison. Prison has no outgoing edges.
setUpRepeatRoute	GraphTest	Prison connects to some of the other nodes.

Test Case Design

Objetive: Verify that Nodes are correctly added to the graph

Class	Method	Scenario	Input Values	Expected Result
GraphTest	addNodeTest	setUpGraphDirectionalEmpty	intersection != null	**true** A node is successfully added to the graph
GraphTest	addNodeTest	setUpGraphDirectionalEmpty	intersection == null	**false** Cannot add an empty node to the graph

GraphTest	addNodeTest	setUpGraphWithElements	intersection != null	**true** A node is successfully added to the graph
GraphTest	addNodeTest	setUpGraphWithElements	intersection == null	**false** Cannot add an empty node to the graph

Objetive: Verificar que se agregan Aristas correctamente al grafo				
Class	**Method**	**Scenario**	**Input Values**	**Expected Result**
GraphTest	addEdgeTest	setUpGraphDirectionalEmpty	source != null target != null	**true** Both nodes are successfully added to the graph (since they are not in the graph, the implementation of this Method allows it), and an edge is successfully added, with a source node and a destination node.
GraphTest	addEdgeTest	setUpGraphDirectionalWithElements	source != null target != null	**true** **An edge is successfully added, with a source node and a destination node.**

Objective: Verify that the graph is traversed correctly by BFS				
Class	**Method**	**Scenario**	**Input Values**	**Expected Result**
GraphTest	bfsTest	setUpGraphDirectionalEmpty	intersection != null	**false** The traversal cannot be started on an empty graph. There is nothing to traverse.

GraphTest	bfsTest	setUpGraphDirectionalWithElements	intersection != null	**true** The depth-first traversal begins from the chosen intersection, traversing all nodes correctly. A new BFS traversal is then performed, starting from the arrival node of the previous traversal.
GraphTest	bfsTest	setUpGraphRoute	start = Casa	The BFS traversal returns the nodes in expansion order: House, Hotel, Hospital, Bank, Casino, Prison
GraphTest	bfsTest	setUpGraphRoute	start = Prisión	Since Treasury has no outputs, BFS returns only ["Treasure"]
GraphTest	bfsTest	setUpGraphRoute setUpRepeatRoute	start = Prisión	The BFS traversal returns nodes in expansion order: Prison, Casino, Bank, Hospital, Hotel, House.
GraphTest	bfsTest	setUpGraphRoute setUpRepeatRoute	start = Casa	Upon arrival at the prison, the BFS route from the prison is repeated again.

Objective: Verify that the BFS algorithm does not traverse disconnected components of the graph.

Class	Method	Scenario	Input Values	Expected Result
GraphTest	bfsTest	setUpDisconnectedComponents	Nodo de inicio = "Casa"	The BFS traversal returns only the nodes of the connected component: ["House", "Hotel", "Hospital"]

Universidad Icesi
Departamento de Computación y Sistemas Inteligentes

Casos de Prueba – GraphUnweighted

Configuración de Scenarios

Name	Class	Scenario
setUpGraphEmpty	GraphUnweightedTest	Creates an empty graph with no nodes or edges.
setUpGraphWithOneNode	GraphUnweightedTest	Creates a graph with a single node ("A") with no connections.
setUpGraphWithTwoNodes	GraphUnweightedTest	Create a graph with two nodes ("A" and "B") connected to each other.
setUpGraphWithThreeNodes	GraphUnweightedTest	Create a graph with three nodes ("A", "B", "C") with no connections.
setUpGraphConnected	GraphUnweightedTest	Create a connected graph online: "A" – "B" – "C".
setUpGraphWithNodes	GraphUnweightedTest	Graph with several nodes but no edges.
setUpGraphWithDisconnectedComponents	GraphUnweightedTest	Create two disconnected components: ["X", "Y", "Z"] and ["A", "B"].
setUpGraphWithCycle	GraphUnweightedTest	Create a cyclic graph: "A" → "B" → "C" → "A".

Universidad Icesi
Departamento de Computación y Sistemas Inteligentes

Objetive	Class	Method	Scenario	Input Values	Expected Result
Verify that a node is added correctly in an unweighted graph.	GraphUnweightedTest	addNodeTest	setUpGraphEmpty	Nodo = "A"	true – Node added successfully
Verify that an edge is added correctly between two existing nodes.	GraphUnweightedTest	addEdgeTest	setUpGraphWithTwoNodes	Origen = "A", Destino = "B"	true – Edge added successfully
Verify that the adjacencies list is correctly obtained.	GraphUnweightedTest	getNeighborsTest	setUpGraphConnected	Nodo = "B"	List of neighbors: ["A","C"]
Verify that the set of nodes of the graph is correctly obtained.	GraphUnweightedTest	getAllNodesTest	setUpGraphWithThreeNodes	-	List: ["A", "B", "C"]
Verify that BFS correctly traverses the graph from an initial node.	GraphUnweightedTest	bfsTest	setUpGraphConnected	Nodo inicio = "A"	Route BFS: ["A", "B", "C"]

Objetive	Class	Method	Scenario	Input Values	Expected Result

Verify that a null node is not added.	GraphUnweightedTest	addNodeTest	setUpGraphEmpty	Nodo = null	false – Invalid node, should not be added
Check that an edge with a null origin node is not added.	GraphUnweightedTest	addEdgeTest	setUpGraphEmpty	Origen = null, Destino = "B"	false – Connection to null node is not allowed
Verify that you are not connecting to a node that does not exist.	GraphUnweightedTest	addEdgeTest	setUpGraphWithOneNode	Origen = "A", Destino = "Z"	false – Destination node does not exist in the graph
Verify that BFS in empty graph does not perform traversal.	GraphUnweightedTest	bfsTest	setUpGraphEmpty	Nodo inicio = "A"	Empty list – There are no nodes to traverse
Verify that get Neighbors returns empty for non-existent node.	GraphUnweightedTest	getNeighborsTest	setUpGraphWithNodes	Nodo = "Z"	Empty list

Objetive	**Class**	**Method**	**Scenario**	**Input Values**	**Expected Result**
Verify that BFS does not reach nodes of a disconnected component of the graph.	GraphUnweightedTest	bfsTest	setUpGraphWithDisconnectedComponents	Nodo inicio = "X"	Recorrido BFS: ["X", "Y", "Z"] – sin alcanzar ["A", "B"]
Verify that when creating a cycle in the graph, BFS does not enter an infinite loop.	GraphUnweightedTest	bfsTest	setUpGraphWithCycle	Nodo inicio = "A"	Recorrido BFS sin duplicados: ["A", "B", "C"]