AdjacencyList and AdjacencyMatrix Test Cases

Tests for addVertex method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpAddVertex1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: addVertexTestStandard** | | | | |
| **Objective of test:** Add a single vertex and check if it is present. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | addVertex | setUpAddVertex1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: addVertexTestLimit** | | | | |
| **Objective of test:** Add several new vertices and check if they exist in the vertex list. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | addVertex | setUpAddVertex1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: addVertexTestInteresting** | | | | |
| **Objective of test:** Add vertices with duplicate values ​​and check if they are handled correctly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | addVertex | setUpAddVertex1() | 1 | 1 |

Tests for addEdge method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpAddEdge1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: addEdgeTestStandard** | | | | |
| **Objective of test:** Add multiple edges between existing vertices and check if the edges are created correctly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | addEdge | setUpAddEdge1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: addEdgeTestLimit** | | | | |
| **Objective of test:** Try to add an edge to a vertex that does not exist and check if it is handled properly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | addEdge | setUpAddEdge1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: addEdgeTestInteresting** | | | | |
| **Objective of test:** Add an edge with minimum weight 0 and check if it is added correctly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | addEdge | setUpAddEdge1() | 1 | 1 |

Tests for getNeighbors method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpGetNeighbors1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: getNeighborsTestStandard** | | | | |
| **Objective of test:** Get the neighbors of a vertex that has multiple neighbors and check if the list is correct. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | getNeighbors | setUpGetNeighbors1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: getNeighborsTestLimit** | | | | |
| **Objective of test:** Get the neighbors of a vertex that has no neighbors and check if the list is empty. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | getNeighbors | setUpGetNeighbors1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: getNeighborsTestInteresting** | | | | |
| **Objective of test:** Get the neighbors of a vertex in a directed graph and check if the direction is handled correctly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | getNeighbors | setUpGetNeighbors1() | 1 | 1 |

Tests for bfs method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpBfs1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: bfsTestStandard** | | | | |
| **Objective of test:** Perform a BFS traversal on a simple graph and verify the order of visits of the nodes. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | bfs | setUpBfs1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: bfsTestLimit** | | | | |
| **Objective of test:** Run BFS on an empty graph and check if no errors occur. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | bfs | setUpBfs1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: bfsTestInteresting** | | | | |
| **Objective of test:** Perform BFS on a disconnected graph and check if all components are visited. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | bfs | setUpBfs1() | 1 | 1 |

Tests for dfs method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpDfs1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: dfsTestStandard** | | | | |
| **Objective of test:** Perform a DFS traversal on a simple graph and check the visiting order of the nodes. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | dfs | setUpDfs1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: dfsTestLimit** | | | | |
| **Objective of test:** Run DFS on an empty graph and check if no errors occur. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | dfs | setUpDfs1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: dfsTestInteresting** | | | | |
| **Objective of test:** Perform DFS on a graph with cycles and check if they are detected correctly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | dfs | setUpDfs1() | 1 | 1 |

Tests for getVertices method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpGetVertices1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: getVerticesTestStandard** | | | | |
| **Objective of test:** Get the list of vertices of a non-empty graph and check if it matches the expected number. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | getVertices | setUpGetVertices1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: getVerticesTestLimit** | | | | |
| **Objective of test:** Get the list of vertices of an empty graph and check if the list is empty. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | getVertices | setUpGetVertices1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: getVerticesTestInteresting** | | | | |
| **Objective of test:** Delete a vertex and check if the associated edges are also deleted. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | getVertices | setUpGetVertices1() | 1 | 1 |

Tests for removeVertex method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpRemoveVertex1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: removeVertexTestStandard** | | | | |
| **Objective of test:** Delete a vertex from a graph and check if it is no longer present. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | removeVertex | setUpRemoveVertex1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: removeVertexTestLimit** | | | | |
| **Objective of test:** **Try to delete a vertex that does not exist and check if it is handled properly.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | removeVertex | setUpRemoveVertex1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: removeVertexTestInteresting** | | | | |
| **Objective of test:** Delete an edge and check if the associated vertices are still present. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | removeVertex | setUpRemoveVertex1() | 1 | 1 |

Tests for clear method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpClear1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: clearTestStandard** | | | | |
| **Objective of test:** Delete all the vertices and edges of the graph and check if it is empty. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | clear | setUpClear1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: clearTestLimit** | | | | |
| **Objective of test:** **Try to delete an empty graph and check if no errors occur.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | clear | setUpClear1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: clearTestInteresting** | | | | |
| **Objective of test:** Add vertices and edges, delete the graph and check if it is empty. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | clear | setUpClear1() | 1 | 1 |

Tests for dijkstra method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpDijkstra1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: dijkstraTestStandard** | | | | |
| **Objective of test:** Run the Dijkstra algorithm on a simple graph and check the shortest paths. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | dijkstra | setUpDijkstra1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: dijkstraTestLimit** | | | | |
| **Objective of test:** **Run Dijkstra on a graph with negative weights and check if it is handled correctly.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | dijkstra | setUpDijkstra1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: dijkstraTestInteresting** | | | | |
| **Objective of test:** Ejecutar Dijkstra en un grafo con múltiples caminos y verificar si se obtienen los resultados correctos. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | dijkstra | setUpDijkstra1() | 1 | 1 |

Tests for floydWarshall method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpFloydWarshall1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: floydWarshallTestStandard** | | | | |
| **Objective of test:** Run the Floyd-Warshall algorithm on a simple graph and check the distances between all pairs of vertices. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | floydWarshall | setUpFloydWarshall1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: floydWarshallTestLimit** | | | | |
| **Objective of test:** **Run Floyd-Warshall on a graph with negative cycles and check if they are detected correctly.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | floydWarshall | setUpFloydWarshall1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: floydWarshallTestInteresting** | | | | |
| **Objective of test:** Run Floyd-Warshall on a graph with unreachable vertices and check if they are handled properly. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | floydWarshall | setUpFloydWarshall1() | 1 | 1 |

Tests for primMST method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpPrimMST1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: primMSTTestStandard** | | | | |
| **Objective of test:** Run Prim's algorithm on a simple graph and verify if a minimum spanning tree is obtained. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | primMST | setUpPrimMST1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of testprimMSTTestLimit** | | | | |
| **Objective of test:** **Run Prim on an empty graph and check if no errors occur.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | primMST | setUpPrimMST1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: primMSTTestInteresting** | | | | |
| **Objective of test:** Ejecutar Prim en un grafo no conexo y verificar si se maneja correctamente. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | primMST | setUpPrimMST1() | 1 | 1 |

Tests for kruskalMST method

**Scenery setup**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpKruskalMST1() | **GraphTest** | Queue<Integer> queue = new Queue<>(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: primMSTTestStandard** | | | | |
| **Objective of test:** Ejecutar el algoritmo de Kruskal en un grafo simple y verificar si se obtiene un árbol de expansión mínima. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | kruskalMST | setUpKruskalMST1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of testprimMSTTestLimit** | | | | |
| **Objective of test:** **Ejecutar Kruskal en un grafo vacío y verificar si no se produce ningún error.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | kruskalMST | setUpKruskalMST1() | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: primMSTTestInteresting** | | | | |
| **Objective of test:** Ejecutar Kruskal en un grafo con aristas de pesos iguales y verificar si se manejan adecuadamente. | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| AdjacencyList & AdjacencyMatrix | kruskalMST | setUpKruskalMST1() | 1 | 1 |