**Test Tables:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1 | DijkstraTest | Crea un grafo vacio de tipo Double |
| Setup2 | DijkstraTest | Crea un grafo vacio de tipo String |
| Setup3 | DijkstraTest | Crea un grafo vacio de tipo Character |

Prueba 1: testAddVertex

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| DijkstraTest | testAddVertex | Setup1 | Entrada de valor 4.0 al grafo | Se espera que después de agregar el elemento, al buscarlo, se encuentre en el grafo. |

Prueba 2: testAddVertexWithContains

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| DijkstraTest | testAddVertexWithContains | Setup1 | Entrada de valor 4.0 al grafo | Se espera que después de agregar el elemento, al buscar en el mapa de adjacencia si contiene el elemento, lo encuentre. |

Prueba 3: testSearchANonExistentVertex

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| DijkstraTest | testSearchANonExistentVertex | Setup1 | Entrada de un grafo vacío | Se espera que después de buscar el número 1.0 el cual nunca se ha ingresado, se reciba un NULL. |

Prueba 4: testAddEdge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| DijkstraTest | testAddEdge | setup2 | Entrada de dos vértices “A” y “B” con una arista conectando desde “A” hasta “B” | Se espera que después de agregar la arista a los dos vértices. Se identifique que el vértice “B” está conectado al vertice ”A”: |

## Test 5 to 9

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Expected Result** |
| testAddEdgeBiDerection | Verify if an edge is added bidirectionally | Edge exists from "B" to "A" |
| testAddEdgeVerifyWeight | Verify if the weight of the added edge is correct | Weight of the edge "B" to "A" is 5 |
| testDijkstra | Verify Dijkstra's algorithm for finding shortest paths | Distances from "A" to other vertices: "A" - 0, "B" - 5, "C" - 2, "D" - 3, "E" - 6 |
| testDijkstra2 | Verify Dijkstra's algorithm for another set of vertices | Distances from "A" to other vertices: "A" - 0, "B" - 2, "C" - 4 |
| testBfs | Verify breadth-first search (BFS) algorithm | Path from "A" to "E": "A" -> "C" -> "E" |

Prueba 10: testBfs2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** | |
| DijkstraTest | testBfs2 | Setup3(). Se realiza una búsqueda en amplitud en un grafo de cadenas de texto. | Grafo inicial con vértices {"A", "B", "C"} | Se espera obtener el camino desde el vértice "A" hasta el vértice "C" utilizando el algoritmo de búsqueda en amplitud. El camino esperado es ["A", "C"]. | |

Prueba 11: testGetShortestPathDj\_Double

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** | |
| DijkstraTest | testGetShortestPathDj\_Double | Setup1().Se obtiene el camino más corto entre dos vértices en un grafo de números decimales. | Grafo inicial con vértices {1.0, 2.0, 3.0, 4.0, 5.0} | Se espera obtener el camino más corto desde el vértice 1.0 hasta el vértice 5.0. El camino esperado es [1.0, 3.0, 5.0] | |

Prueba 12: testGetShortestPathDj\_Character

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** | |
| DijkstraTest | testGetShortestPathDj\_Character | Setup3(). Se obtiene el camino más corto entre dos vértices en un grafo de caracteres. | Grafo inicial con vértices {'A', 'B', 'C', 'D', 'E'} | Se espera obtener el camino más corto desde el vértice 'A' hasta el vértice 'E'. El camino esperado es ['A', 'C', 'D', 'E']. | |

Prueba 13: testRemove

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** | |
| DijkstraTest | testRemove | Setup1(). Se intenta eliminar un vértice que existe en el grafo. | Se llama al metodo agregar y se le pasa que el numero 4 | Fallo en la eliminación del vértice 4.0. Se espera que el método devuelva false. | |
|  |  |  |  |  | |

Prueba 14: testIncorrectRemove

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** | |
| DijkstraTest | testIncorrectRemove | Setup1(). Se intenta eliminar un vértice que no existe en el grafo. | Se llama al metodo eliminar y se le pasa que elimine el numero 4 | Fallo en la eliminación del vértice 4.0. Se espera que el método devuelva false. | |
|  |  |  |  |  | |