|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup(1) | AdjacencyListGraph | Grafo vacío al inicio. |
| Setup(2) | AdjacencyListGraph | Grafo con dos vértices (A, B) y una arista entre ellos con peso 5. |
| Setup(3) | AdjacencyListGraph | Grafo con múltiples vértices (A, B, C) y aristas: A-B (peso 5), B-C (peso 3), A-C (peso 7). |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Agregar vértices | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| AdjacencyListGraph | addVertex() | setUp1() | * A | El vértice A está presente en el grafo. |
| AdjacencyListGraph | addVertex() | setUp1() | * B, C | Los vértices B y C están presentes en el grafo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: Agregar aristas** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| AdjacencyListGraph | addEdge() | setUp1() | * addEdge(A, B, 5) | La arista entre A y B existe, y su peso es 5. |
| AdjacencyListGraph | addEdge() | setUp2() | * addEdge(B, C, 3) | La arista entre B y C existe, y su peso es 3. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: Eliminar vertices** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| AdjacencyListGraph | removeVertex() | setUp2() | |  | | --- | |  |  |  | | --- | | * removeVertex(B) | | El vértice B y sus aristas asociadas ya no están en el grafo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: consultar vértices y aristas** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| AdjacencyListGraph | hasVertex() | setUp1() | |  | | --- | |  |  |  | | --- | | * A | | Retorna true porque A fue agregado. |
| AdjacencyListGraph | hasVertex() | setUp1() | * Z | Retorna false porque Z no está en el grafo. |
| AdjacencyListGraph | hasEdge() | setUp2() | * A, B | Retorna true porque hay una arista entre A y B. |
| AdjacencyListGraph | getWeight() | setUp2() | * A, B | Retorna 5 porque el peso de la arista es 5. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nombre** | | **Clase** | | **Escenario** | | |
| Setup(1) | | |  | | --- | |  |  |  | | --- | | PrimAndKruskalTest | | | Grafo con 4 vértices (A, B, C, D) y las siguientes aristas: A-B (1), B-C (2), C-D (3), D-A (4) | | |
| Setup(2) | | |  | | --- | |  |  |  | | --- | | PrimAndKruskalTest | | | |  | | --- | |  |  |  | | --- | | Grafo con 6 vértices (A, B, C, D, E, F) y las siguientes aristas: A-B (3), A-C (1), B-C (3), B-D (6), C-E (5), D-E (2), E-F (4) | | | |
| Setup(3) | | |  | | --- | |  |  |  | | --- | | PrimAndKruskalTest | | | |  | | --- | |  |  |  | | --- | | Grafo disconexo con dos componentes: Componente 1: A-B (2), B-C (3), Componente 2: D-E (1) | | | |
| **Objetivo de la Prueba: algoritmos kruskal** | | | | | | | | |
| **Clase** | | **Método** | | **Escenario** | | **Valores de Entrada** | **Resultado** | |
| PrimAndKruskalTest | | kruskal() | | setUp1() | | * Ninguno (procesa todo el grafo) | El árbol generador mínimo incluye las aristas: A-B (1), B-C (2), C-D (3) | |
| PrimAndKruskalTest | | kruskal() | | setUp2() | | * Ninguno (procesa todo el grafo) | El árbol generador mínimo incluye las aristas: A-C (1), D-E (2), A-B (3), E-F (4), C-E (5) | |
| PrimAndKruskalTest | | kruskal() | | setUp3() | | * Ninguno (procesa todo el grafo) | Retorna true porque hay una arista entre A y B. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: algoritmos de prim** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| PrimAndKruskalTest | prim() | setUp1() | * Nodo inicial: A | El árbol generador mínimo incluye las aristas: A-B (1), B-C (2), C-D (3) |
| PrimAndKruskalTest | prim() | setUp2() | * Nodo inicial: A | El árbol generador mínimo incluye las aristas: A-C (1), D-E (2), A-B (3), E-F (4), C-E (5) |
| PrimAndKruskalTest | prim() | setUp3() | * Nodo inicial: A | El algoritmo no puede procesar grafos disconexos. Devuelve un error o resultado parcial. |