**Configuración de los escenarios.**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setup1 | Grafo | Hay un grafo sin ningún vértice ni arista. |
| setup2 | Grafo | Hay un grafo que contiene 4 vértices y 4 aristas. |
| setup3 | Grafo | Hay un grafo que contiene un solo vértice. |
| Setup4 | Grafo | Hay un grafo con varios vértices y aristas. |

**Diseño de las pruebas.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar el correcto funcionamiento de método agregarArista() de la clase Grafo, cuando:   1. El grafo no tenga vértices. 2. El grafo tenga vértices y aristas. 3. El grafo ya contenga la arista a agregar. | | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Grafo | agregarArista () | setup1 | Un String=”A” que representa el vértice de origen y uno de destino “B”, un integer=30 Sque representa el costo de esa arista . | Nada porque no hay vértices para unirlos mediante una arista. |
| Grafo | agregarArista () | setup2 | Un String=”A” que representa el vértice de origen y uno de destino “C”, un integer=7 que representa el costo de esa arista . | Se agrega correctamente la arista entre A y C. |
| Grafo | agregarArista() | setup2 | Un String=”A” que representa el vértice de origen y uno de destino “B”, un integer=2 que representa el costo de esa arista . | No se agrega la arista. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar el correcto funcionamiento de método eliminarArista() de la clase Grafo, cuando:   1. El grafo no tenga vértices. 2. El grafo tenga vértices y aristas. 3. El grafo no tenga la arista a eliminar. | | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Grafo | eliminarArista () | setup1 | Un String=”A” que representa el vértice de origen y uno de destino “B”. | Nada porque no hay vértices para unirlos mediante una arista. |
| Grafo | eliminarArista () | setup2 | Un String=”A” que representa el vértice de origen y uno de destino “B”. | Se elimina correctamente la arista entre A y B. |
| Grafo | eliminarArista () | setup2 | Un String=”A” que representa el vértice de origen y uno de destino “H”. | No se elimina la arista. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar el correcto funcionamiento de método agregarVertice() de la clase Grafo, cuando:   1. El grafo no tenga vertices. 2. El grafo tenga 1 vertice. 3. El grafo ya tenga ese vertice. | | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Grafo | agregarVertice () | setup1 | Un String=”B” que representa el vértice que se va a agregar al grafo vacío. | Cantidad de vertices en el grafo aumentado a 1. |
| Grafo | agregarVertice () | Setup2 | Un String=”B” que representa el vértice a agregar a un grafo con 1 solo vertice. | Se agrega correctamente el vertice y la cantidad de vertices en el grafo aumenta en 1. |
| Grafo | agregarVertice () | setup2 | Un String=”A” que representa un vertice que ya estaba agregado en el grafo | No se agrega el vertice. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar el correcto funcionamiento de método eliminarVertice() de la clase Grafo, cuando:   1. El grafo no tenga vertices. 2. El grafo tenga 1 vertice. 3. El grafo no tenga ese vertice. | | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Grafo | eliminarVertice () | setup1 | Un String=”B” que representa el vértice que se va quiere eliminar del grafo vacío. | No se elimina ningun vertice poque no los hay. |
| Grafo | eliminarVertice () | Setup4 | Un String=”B” que representa el vértice a eliminar. | Se elimina correctamente el vertice y la cantidad de vertices en el grafo disminuye en 1. |
| Grafo | eliminarVertice () | setup2 | Un String=”A” que representa un vertice que ya estaba agregado en el grafo | No se elimina el vertice, porque no se encuentra. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar el correcto funcionamiento de método djkstra () de la clase Grafo el cual usa el algoritmo de djkstra, cuando:   1. El grafo no tenga vertices. 2. El grafo tenga 2 caminos entre destinos 3. El grafo tenga mas de 2 posibles caminos | | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Grafo | djkstra () | setup1 | Un String=”A” que representa el vértice de origen y uno de destino “B”. | Nos dirá que la distancia será 0, pues no hay distancia que calcular. |
| Grafo | djkstra () | Setup2 | Un String=”A” que representa el vértice de origen y uno de destino “C. | Nos dirá que el camino más corto cuesta 5, mientras hay otro que cuesta 6. |
| Grafo | djkstra () | Setup4 | Un String=”A” que representa el vértice de origen y uno de destino “G. | Nos dirá que el camino más corto cuesta 6, mientras hay otros caminos. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar el correcto funcionamiento de método bellman () de la clase Grafo el cual usa el algoritmo de bellman, cuando:   1. El grafo no tenga vertices. 2. El grafo tenga 2 caminos entre destinos 3. El grafo tenga mas de 2 posibles caminos | | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Grafo | bellman () | setup1 | Un String=”A” que representa el vértice de origen y uno de destino “B”. | Nos dirá que la distancia será 0, pues no hay distancia que calcular. |
| Grafo | bellman () | Setup2 | Un String=”A” que representa el vértice de origen y uno de destino “C. | Nos dirá que el camino más corto cuesta 5, mientras hay otro que cuesta 6. |
| Grafo | bellman () | Setup4 | Un String=”A” que representa el vértice de origen y uno de destino “G. | Nos dirá que el camino más corto cuesta 6, mientras hay otros caminos. |