**Diseño pruebas unitarias**

**Pruebas clase KruskalTest**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp1 | KruskalTest | Un objeto Kruskal y una matriz de enteros con los valores {0,2,0,6,0,2,0,3,8,5,0,3,0,0,7,6,8,0,0,9,0,5,7,9,0} |
| setUp2 | KruskalTest | Un objeto Kruskal y una matriz de enteros con los valores {0,3,20,3,0,3,0,0,0,0,20,0,0,2,3,3,0,2,0,0,0,0,3,0,0} |

* **Objetivo de la prueba:** Determinar si el método ***kruskalMST*** encuentra el árbol de recubrimiento mínimo correctamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| **KruskalTest** | **kruskalMSTTest1** | setUp1 |  | El método retorna correctamente el peso mínimo del árbol que se esperaba, en este caso 16 |

* **Objetivo de la prueba:** Determinar si el método ***kruskalMST*** encuentra el árbol de recubrimiento mínimo correctamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| **KruskalTest** | **kruskalMSTTest2** | setUp2 |  | El método retorna correctamente el peso mínimo del árbol que se esperaba, en este caso 11 |

**Pruebas clase GraphTest**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp1 | GraphTest | Un objeto Graph y 4 objetos User con los siguientes valores:  User a: “andrea”  User b: “danna”  User c: “escobar”  User d: “cordoba”  User e: “reyes”  donde :  a tiene conexión con d  a tiene conexión con c  d tiene conexión con c  c tiene conexión con e  a tiene conexión con b |

* **Objetivo de la prueba:** Determinar si el método ***dfs*** me devuelve el camino en profundidad desde un nodo de inicio hasta recorrer todo el grafo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| **GraphTest** | **dfsTest1** | setUp1 | String expected= "andreacordobaescobarreyesdanna"  Nodo inicial= andrea | El método retorna correctamente el camino en profundidad desde el nodo andrea hasta llegar a danna, recorriendo todos los nodos del grafo. |

* **Objetivo de la prueba:** Determinar si el método ***dfs*** me devuelve el camino en profundidad desde un nodo de inicio hasta recorrer todo el grafo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| **GraphTest** | **dfsTest2** | setUp1 | String expected= "dannaandreacordobaescobarreyes"  Nodo inicial= danna | El método retorna correctamente el camino en profundidad desde el nodo danna hasta llegar a reyes, recorriendo todos los nodos del grafo. |

* **Objetivo de la prueba:** Determinar si el método ***dfs*** me devuelve el camino en profundidad desde un nodo de inicio hasta recorrer todo el grafo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| **GraphTest** | **dfsTest3** | setUp1 | String expected= "escobarandreacordobadannareyes"  Nodo inicial= escobar | El método retorna correctamente el camino en profundidad desde el nodo escobar hasta llegar a reyes, recorriendo todos los nodos del grafo. |