**CASOS DE PRUEBA**

**Configuración**:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupScenary1 | ButterflyTest |  |
| setupScenary2 | ButterflyTest |  |
| setupScenary3 | ButterflyTest |  |

**Diseño de casos de prueba para el grafo representado con matrices de adyacencia:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se agrega un vértice correctamente al grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | addVertice | setupScenary1 | Vértice = ”Japón” |  |
| MatrixGraph | addVertice | setupScenary1 | Vértice = ”Portugal” |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se agrega una arista correctamente al grafo de peso no negativo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | addEdge | setupScenary2 | Vértice inicial = “Portugal”  Vértice final = “Colombia”  Costo = 1600 |  |
| MatrixGraph | addEdge | setupScenary2 | Vértice inicial = ”Madagascar”  Vértice final = “Colombia”  Costo = -200 | No se pudo añadir la arista al grafo, ya que su costo es un número negativo. |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se elimina un vértice correctamente del grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | deleteVertice | setupScenary2 | Vértice = ”Dubai” |  |
| MatrixGraph | deleteVertice | setupScenary1 | Vértice = ”Colombia” | Arraylist de vértices vacía. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba: Verificar que el algoritmo bfs funciona correctamente** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | bfs | setupScenary2 | Vértice inicial =” Colombia” |  |
| MatrixGraph | bfs | setupScenary3 | Vértice inicial =” Colombia” |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Objetivo de la prueba: Verificar que el algoritmo dfs funciona correctamente** | | | | | | | | |
| **Clase** | | **Método** | | **Escenario** | | **Valores de entrada** | | **Resultado** |
| MatrixGraph | | dfs | | setupScenary3 | |  | | Size=10 |
| **Objetivo de la prueba:**Verificar que se elimina una arista correctamente del grafo. | | | | | | | | |
| **Clase** | **Método** | | **Escenario** | | **Valores de entrada** | | **Resultado** | |
| MatrixGraph | deleteEdge | | setupScenary2 | | Vértice inicial = ”Colombia”  Vértice final = “Dubai”  Costo = 1200 | |  | |
| MatrixGraph | deleteEdge | | setupScenary2 | | Vértice inicial = ”Dubai”  Vértice final = “Portugal”  Costo = 3000 | |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Validar que se modificó una adyacencia dentro del grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | setEdge | setupScenary2 | Vértice inicial = ”Dubai”  Vértice final = “Portugal”  Costo = 2130 |  |
| MatrixGraph | setEdge | setupScenary2 | Vértice inicial = ”Colombia”  Vértice final = “Dubai”  Costo = 3000 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Dijkstra funciona correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | makeDijsktra | setupScenary3 | Vértice = ” Portugal”  Vértice = “Japón” | $6100 – El costo mínimo que hay en el vuelo de Colombia a Japón. |
| MatrixGraph | makeDijsktra | setupScenary3 | Vértice = ”Portugal”  Vértice = “null” | [∞, ∞, $6100, $9100, $2100, $5100, $4600, $0, $3350, $8350]  Donde los precios corresponden a los siguientes países respectivamente:  Colombia, España, Japón, EEUU, Rusia, Australia, Nigeria, Portugal, Dubai y Madagascar. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Dijkstra NO encuentra valores mínimos cuando no hay vuelos entre dos países. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | makeDijsktra | setupScenary3 | Vértice = ”Madagascar”  Vértice = “Dubai” | No se encontró un costo mínimo en el vuelo de Madagascar a Dubái. |
| MatrixGraph | makeDijsktra | setupScenary3 | Vértice = ”España”  Vértice = “Andorra” | No se encontró un costo mínimo en el vuelo de España a Andorra. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Prim funciona correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | prim | setupScenary3 | Vértice = ” Colombia” | $28370 – El costo mínimo que hay en el vuelo de Colombia a todos los demás países. |
| MatrixGraph | prim | setupScenary2 | Vértice = ”Colombia” | $6950 – El costo mínimo de ir desde Colombia a todos los demás países. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Prim NO encuentra el costo mínimo de viajar por todos los países si hay países a los que no se puede viajar. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | prim | setupScenary3 | Vértice = ”Madagascar” | $1930 – Solo se encontró el valor de ir de Madagascar a Japón, ya que no se podía viajar hacia más países. |
| MatrixGraph | prim | setupScenary3 | Vértice = ”Rusia” | No se encontró un costo mínimo teniendo como país de origen a Rusia. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba: Verificar que el algoritmo bfs funciona correctamente** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | bfs | setupScenary2 | Vértice inicial =” Colombia” |  |
| ListGraph | bfs | setupScenary3 | Vértice inicial =” Colombia” |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba: Verificar que el algoritmo dfs funciona correctamente** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | dfs | setupScenary3 |  | Size=10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar que el algoritmo de floyd-warshall acomoda de forma correcta los costos de cada par de vértices. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | floyd | setupScenary3 |  | Se organizó correctamente la matriz con los costos de cada par de vértices obteniendo el siguiente resultado. |
| ListGraph | floyd | setupScenary1 |  | La matriz utilizada por el algoritmo no se encuentra inicializada ya que no hay un grafo en el cual trabajar. |
| ListGraph | floyd | setupScenary2 |  | Se organizó correctamente la matriz con los costos de cada par de vértices obteniendo el siguiente resultado. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Validar que el algoritmo de kruskal calcula el menor recorrido entre todos los vértices | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | kruskal | setupScenary3 |  | Retorna el valor de mínimo de recorrer todos los países  int = 17.400 |
| ListGraph | kruskal | setupScenary1 |  | No hay un grafo en el cuál trabajar  int = 0 |
| ListGraph | kruskal | setupScenary2 |  | Retorna el valor mínimo de recorrer todos los países.  int = 4.540 |

**Diseño de casos de prueba para el grafo representado con listas de adyacencia:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se agrega un vértice correctamente al grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | addVertice | setupScenary1 | Vértice = ”Nigeria” |  |
| ListGraph | addVertice | setupScenary1 | Vértice = ”Madagascar” |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se agrega una arista correctamente al grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | addEdge | setupScenary2 | Vértice inicial = ” Colombia”  Vértice final = “Portugal”  Costo = 1600 |  |
| ListGraph | addEdge | setupScenary2 | Vértice inicial = ” Dubai”  Vértice final = “Madagascar”  Costo = 780 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se elimina un vértice correctamente del grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | deleteVertice | setupScenary2 | Vértice = ”Dubai” |  |
| ListGraph | deleteVertice | setupScenary1 | Vértice = ”Colombia” | El arraylist de vértices está vacio. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que se elimina una arista correctamente del grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | deleteEdge | setupScenary2 | Vértice inicial = ”Colombia”  Vértice final = “Dubai”  Costo = 1200 |  |
| ListGraph | deleteEdge | setupScenary2 | Vértice inicial = ”Dubai”  Vértice final = “Portugal”  Costo = 3000 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Validar que se modificó una adyacencia dentro del grafo. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MatrixGraph | setEdge | setupScenary2 | Vértice inicial = ”Portugal”  Vértice final = “Madagascar”  Costo = 1320 |  |
| MatrixGraph | setEdge | setupScenary2 | Vértice inicial = ”Madagascar”  Vértice final = “Dubai”  Costo = 3000 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Dijkstra funciona correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | makeDijsktra | setupScenary3 | Vértice = ” Portugal”  Vértice = “Japón” | $6100 – El costo mínimo que hay en el vuelo de Colombia a Japón. |
| ListGraph | dijsktra | setupScenary3 | Vértice = ”Portugal”  Vértice = “null” | [∞, ∞, $6100, $9100, $4600, $2100, $5100, $3350, $0, $8350]  Donde los precios corresponden a los siguientes países respectivamente:  Colombia, España, Japón, EEUU, Rusia, Australia, Nigeria, Portugal, Dubai y Madagascar. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Dijkstra NO encuentra valores mínimos cuando no hay vuelos entre dos países. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | makeDijsktra | setupScenary3 | Vértice = ”Madagascar”  Vértice = “Dubai” | No se encontró un costo mínimo en el vuelo de Madagascar a Dubái. |
| ListGraph | makeDijsktra | setupScenary3 | Vértice = ”España”  Vértice = “Andorra” | No se encontró un costo mínimo en el vuelo de España a Andorra. |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Prim funciona correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | prim | setupScenary3 | Vértice = ” Colombia” | $28370 – El costo mínimo que hay en el vuelo de Colombia a todos los demás países. |
| ListGraph | prim | setupScenary2 | Vértice = ”Colombia” | $6950 – El costo mínimo de ir desde Colombia a todos los demás países. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Verificar que el algoritmo Prim NO encuentra el costo mínimo de viajar por todos los países si hay países a los que no se puede viajar. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | prim | setupScenary3 | Vértice = ”Madagascar” | $1930 – Solo se encontró el valor de ir de Madagascar a Japón, ya que no se podía viajar hacia más países. |
| ListGraph | prim | setupScenary3 | Vértice = ”Rusia” | No se encontró un costo mínimo teniendo como país de origen a Rusia. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Verificar que el algoritmo de floyd-warshall acomoda de forma correcta los costos de cada par de vértices. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | floyd | setupScenary3 |  | Se organizó correctamente la matriz con los costos de cada par de vértices obteniendo el siguiente resultado. |
| ListGraph | floyd | setupScenary1 |  | La matriz utilizada por el algoritmo no se encuentra inicializada ya que no hay un grafo en el cual trabajar. |
| ListGraph | floyd | setupScenary2 |  | Se organizó correctamente la matriz con los costos de cada par de vértices obteniendo el siguiente resultado. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:**Validar que el algoritmo de kruskal calcula el menor recorrido entre todos los vértices | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ListGraph | kruskal | setupScenary3 |  | Retorna el valor de mínimo de recorrer todos los países  int = 17.400 |
| ListGraph | kruskal | setupScenary1 |  | No hay un grafo en el cuál trabajar  int = 0 |
| ListGraph | kruskal | setupScenary2 |  | Retorna el valor mínimo de recorrer todos los países.  int = 4.540 |