|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenary** |
| setupKruskal1 | GraphTest | A no directed and weighted graph implemented with adjacency matrix with the next vertexes:  “A”, “B”, “C”, “D”, “E”, “Z”.  And the next edges, (From, to, weight):   * ("A", "B", 4.0) * ("A", "C", 2.0) * ("B", "D", 5.0) * ("B", "C", 1.0) * ("C", "D", 8.0) * ("C", "E", 10.0) * ("D", "E", 2.0) * ("D", "Z", 6.0) * ("E", "Z", 3.0) |
| setUpWeightedGraphWithCycle | GraphTest | A no directed and weighted graph implemented with adjacency list with the next vertexes:  “A”, “B”, “C”, “D”, “E”, “Z”.  And the next edges, (From, to, weight):   * ("A","B",4); * ("A","C",2); * ("B","D",5); * ("B","C",1); * ("C","D",8); * ("C","E",10); * ("E","D",2); * ("E","Z",3); * ("D","Z",6); |
| setUpStage7Directed | GraphTest | A directed and no weighted graph implemented with adjacency list with the next vertexes:  “V”, “R”, “S”, “H”, “L”  And the next edges, (From, to):   * (“V”, “R”) * (“S”, “R”) * (“H”, “L”) |
| setUpStage9NoDirectedAndWeight | GraphTest | A no directed and weighted graph implemented with adjacency list with the next vertexes:  “A”, “B”, “C”  And the next edges, (From, to, weight):  ("A","B",10)  ("C","A",15) |
| setUpStage9DirectedAndWeight | GraphTest | A directed and weighted graph implemented with adjacency list with the next vertexes:  “A”, “B”, “C”  And the next edges, (From, to, weight):  ("A","B",10)  ("C","A",15) |
| setUpStage2NoDirected | GraphTest | A no directed and no weighted graph implemented with adjacency list with the next vertexes:  “A”, “B” |
| setUpStage3Directed | GraphTest | A directed and no weighted graph implemented with adjacency list with the next vertexes:  “A”, “B” |
| setupFloyd | GraphTest | A no directed and weighted graph implemented with adjacency matrix with the next vertexes:  “A”, “B”, “C”, “D”, “E”, “F”, “G”, “H”  And the next edges, (From, to, weight):  ("A","B",3)  ("C","A",5)  ("B","D",1)  ("B","E",8)  ("C","F",12)  ("C","G",5) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los métodos InsertVertex e insertEdge de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente, recorriendo el grafo diseñado por anchura. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | insertEdge | setUpStage3Directed | From = “A” To = “B” | Graph size = 2  Edge: From = “A” To = “B” |
| Igraph | insertEdge | setUpStage2NoDirected | From = “A” To = “B” | Graph size = 2  Edge:  From = “A” To = “B”  From = “B” To = “A” |
| Igraph | insertVertex  insertEdge | setUpStage9DirectedAndWeight |  | The next graph should be created  vertexes:  “A”, “B”, “C”  Edges:  (From, to, weight):  ("A","B",10)  ("C","A",15) |
| Igraph | insertVertex  deleteVertex | An empty graph. (0 vertexes) |  | Graph size = 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los métodos deleteVertex e deleteEdge de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente, recorriendo el grafo diseñado por anchura. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | deleteVertex  deleteEdge | setUpStage2NoDirected | Insert Edge  From = “A” To = “B”  Delete edge  From = “A” To = “B” | Adjacency list of vertex “A” size = 0  Adjacency list of vertex “B” size = 0 |
| Igraph | deleteVertex  deleteEdge | setUpStage3Directed | Insert Edge  From = “A” To = “B”  Delete edge  From = “A” To = “B” | Adjacency list of vertex “A” size = 0  Adjacency list of vertex “B” size = 0 |
| Igraph | deleteVertex  deleteEdge | A graph with a vertex v =”A” | Delete vertex v = “A” | List of vertexes size = 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los métodos BFS de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente, recorriendo el grafo diseñado por anchura. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | bfs | setUpStage7Directed | From = “V” | “V R” |
| Igraph | bfs | setUpStage7Directed | From = “R” | “R” |
| Igraph | bfs | setUpStage7Directed | From = “H” | “H L” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los métodos DFS de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente, recorriendo el grafo diseñado por profundidad. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | dfs | setUpStage7Directed |  | 3 arboles  Arbol #1:  “V R” |
| Igraph | dfs | setUpStage7Directed |  | 3 arboles  Arbol #2:  “S” |
| Igraph | dfs | setUpStage7Directed |  | 3 arboles  Arbol #3:  “H L” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los método kruskal de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | kruskal | setupKruskal1 |  | “B C A C D E E Z B D” |
| Igraph | kruskal | setUpWeightedGraphWithCycle |  | "B C A C E D E Z B D" |
| Igraph | kruskal | setUpStage9NoDirectedAndWeight |  | “A B C A” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los método prim de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | prim | setupKruskal1 |  | "A","C","B","D","E","Z" |
| Igraph | prim | setUpWeightedGraphWithCycle | sValue = “A” | "A","C","B","D","E","Z" |
| Igraph | prim | setUpWeightedGraphWithCycle | sValue = “D” | "D","B","C","A","E","Z" |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los método Dijkstra de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | Dijkstra | setUpWeightedGraphWithCycle | source “A” | Distance Map:  (“Z”, 13)  (“A”, 0)  (“B”, 3)  (“C”, 2)  (“D”, 8)  (“E”, 10)  Prev path:  "A C B D E Z" |
| Igraph | Dijkstra | setUpWeightedGraphWithCycle | source “D” | Distance Map:  (“Z”, 5)  (“A”, 8)  (“B”, 5)  (“C”, 6)  (“D”, 0)  (“E”, 2)  Prev path:  "A C B D E Z" |
| Igraph | Dijkstra | setUpWeightedGraphWithCycle | source “Z” | Distance Map:  (“Z”, 0)  (“A”, 13)  (“B”, 10)  (“C”, 11)  (“D”, 5)  (“E”, 3)  Prev path:  "A C B D E Z" |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que los método floydWarshall de la intefraz Igraph, que implementan las clases AdjacencyListGraph y AdjacencyMatrixGraph funcionen correctamente. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Igraph | floydWarshall | setupFloyd |  | Distance matrix:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 3 | 5 | 4 | 11 | 17 | 10 | ∞ | | 3 | 0 | 8 | 1 | 8 | 20 | 13 | ∞ | | 5 | 8 | 0 | 9 | 16 | 12 | 5 | ∞ | | 4 | 1 | 9 | 0 | 9 | 21 | 14 | ∞ | | 11 | 8 | 16 | 9 | 0 | 28 | 21 | ∞ | | 17 | 20 | 12 | 21 | 28 | 0 | 17 | ∞ | | 10 | 13 | 5 | 14 | 21 | 17 | 0 | ∞ | | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | |
| Igraph | floydWarshall | setupKruskal1 |  | Distance matrix:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 3 | 2 | 8 | 10 | 13 | | 3 | 0 | 1 | 5 | 7 | 10 | | 2 | 1 | 0 | 6 | 8 | 11 | | 8 | 5 | 6 | 0 | 2 | 5 | | 10 | 7 | 8 | 2 | 0 | 3 | | 13 | 10 | 11 | 5 | 3 | 0 | |
| Igraph | floydWarshall | setUpStage9NoDirectedAndWeight |  | Distance matrix:   |  |  |  | | --- | --- | --- | | 0 | 10 | 15 | | 10 | 0 | 25 | | 15 | 25 | 0 | |