|  |  |  |
| --- | --- | --- |
| Nombre | Clase | Escenario |
| setUpScenery1 | AdjacencyListTest |  |
| setUpScenery1 | AdjacencyMatrixTest |  |
| setUpScenery1 | Dijkstra |  |
| setUpScenery1 | FloydW |  |
| setUpScenery1 | Kruskal |  |
| setUpScenery1 | Prim |  |
| setUpScenery1 | BFS |  |
| setUpScenery1 | DFS |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Metodo | Escenario | Valores de entrada | Resultado |
| Dijkstra | testShortestPath() | setUpScenery1 | int[][] matrix ={{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}, }; int nodes=3; int source=1; | Verifys that the algorithm travels the minimum distance of each vertex passing through all the edges |
| Dijkstra | testShortestPath1() | setUpScenery1 | int[][] matrix ={ {1,2,3},  {4,5,6},  {7,8,9}}; int nodes=3; int source=1; | Verifys that the algorithm travels the minimum distance of each vertex passing through all the edges |
| FloydW | testCalculate() | setUpScenery1 | dis = new int[6][6]; int[][] validate = { {0 , 5 , 3 , 4 , 18 , 19}, {5 , 0 , 2 , 1 , 13 , 14}, {3 , 2 , 0 , 1 , 15 , 16}, {4 , 1 , 1 , 0 , 14 , 15}, {18 , 13 , 15 , 14 , 0 , 1}, {19 , 14 , 16 , 15 , 1 , 0} }; | Calculate the minimum distance |
| Kruskal | kruskalAlgorithmTest1() | setUpScenery1 | matrix = new int[][] {  { 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 },  }; | Finds a minimum spanning forest of an undirected edge-weighted graph. |
| Kruskal | kruskalAlgorithmTest2() | setUpScenery2 | graph = new int[][] {  {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},  }; | Finds a minimum spanning forest of an undirected edge-weighted graph. |
| Prim | primAlgorithmTest1() | setUpScenery1 | graph = new int[][] {  { 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 },  }; | Finds a minimum spanning tree for a weighted undirected graph |