**Graph tests**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| initSetup | GraphTest | Grafo vacío |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test cases** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Graph | addVertex | initSetup | vertices: [“A”,”B”,”C”] | El grafo contiene 3 vértices. Contiene “A”, contiene “B” y contiene “C”. |
| Graph | addEdge | initSetup | vertices: [“A”,”B”,”C”]  edges:[[“A”,”B”,10],[ “A”,”C”,5]] | El vértice “A” tiene 2 vecinos: “B” y “C”. |
| Graph | removeVertex | initSetup | vertices: ["A","B","C"]  vertexToRemove: "B" | El grafo contiene 2 vértices: "A" y "C". No contiene "B". |
| Graph | removeEdge | initSetup | vertices: ["A","B","C"]  edges: [["A","B",10], ["A","C",5]]  edgeToRemove: ["A","B",10] | El vértice "A" tiene 1 vecino: "C". El grafo ya no contiene la arista ["A","B",10]. |
| Graph | getNeighbors | initSetup | vertices: ["A","B","C"]  edges: [["A","B",10], ["A","C",5]]  vertex: "A" | Los vecinos del vértice "A" son "B" y "C". |
| Graph | dfs | initSetup | vertices: ["A","B","C","D"]  edges: [["A","B",10], ["B","C",5], ["C","D",3]] | La salida de la ejecución del método dfs("A") es "A B C D". |
| Graph | bfs | initSetup | vertices: ["A","B","C","D"]  edges: [["A","B",10], ["A","C",5], ["B","D",7]] | La salida de la ejecución del método bfs("A") es "A B C D". |
| Graph | dijkstra | initSetup | vertices: ["A","B","C","D"]  edges: [["A","B",10], ["A","C",5], ["B","C",2], ["C","D",4]] | La distancia más corta desde "A" hasta "B" es 10, hasta "C" es 5, hasta "D" es 9. |
| Graph | emptyGraph | initSetup | No se agregaron vértices al grafo. | La lista de vértices es vacía.  La lista de vecinos del vértice "A" es vacía. bfs y dfs no producen salida. |
| Graph | negativeWeights | initSetup | vertices: ["A","B"]  edges: [["A","B",-5]] | La distancia más corta desde "A" hasta "B" es -5. |

**Maze tests**

**Case 1:**

**- input:**

4

4

1 2 4 2

3 -1 2 1

-1 -1 3 8

2 -1 1 0

3

1 0

0 3

3 0

3 2

**- output:**

'2'

'(0,3) -> (1,3) -> (1,2) -> (2,2) -> (3,2)'

'7'

**Case 2:**

4

5

1 2 4 2 1

3 -1 2 1 3

-1 5 3 -1 2

2 1 1 0 -1

3

0 0

1 1

3 3

3 2

**- output:**

'2'

'(3,3) -> (3,2)'

'1'

**Case 3:**

**- input:**

3

3

1 2 3

-1 -1 -1

3 2 1

1

0 0

2 2

**- output:**

'-1'

**Case 4:**

**- input:**

3

3

1 2 3

-1 -1 2

3 -1 1

2

0 0

2 0

2 2

**- output:**

'1'

'(0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2)'

'8'

**Case 5:**

**- input:**

4

4

1 2 4 2

3 -1 2 1

-1 -1 3 8

2 -1 1 0

3

0 0

0 2

3 0

3 2

**- output:**

'2'

'(0,2) -> (1,2) -> (2,2) -> (3,2)'

'6'

**Case 6:**

**- input:**

3

3

1 -1 3

2 -1 -1

3 2 -1

1

0 0

2 2

**- output:**

'-1'

**Case 7:**

**- input:**

3

3

-1 -1 -1

-1 -1 -1

-1 -1 -1

2

1 0

1 2

2 2

**- output:**

'-1'