Test Cases

Graph Adjacency List

Scenario Configuration

Name	Class	Scenario
setUp1	GraphALTest	Creates a new GraphAl empty
setUp2	GraphALTest	Adds 5 vertices and 8 edges to the graph: - Vertices: Vertex 1 with key 1 and value 1 Vertex 2 with data 1 and value 1 Vertex 3 with data 1 and value 1 Vertex 4 with data 1 and value 1 Vertex 5 with data 1 and value 1 - Edges: Edge from vertex 1 to vertex 2 with weight 1 Edge from vertex 1 to vertex 5 with weight 1 Edge from vertex 1 to vertex 4 with weight 1 Edge from vertex 2 to vertex 3 with weight 1 Edge from vertex 2 to vertex 5 with weight 1 Edge from vertex 4 to vertex 5 with weight 1 Edge from vertex 4 to vertex 3 with weight 1 Edge from vertex 3 to vertex 5 with weight 1

setUp3	GraphALTest	Adds 2 vertices and 2 edges to the graph: - Vertices: Vertex 1 with key 1 and value 100 Vertex 2 with data 2 and value 100 - Edges: Edge from vertex 1 to vertex 2 with weight 10 Edge from vertex 2 to vertex 1 with weight 20
setUp4	GraphALTest	Adds 5 vertices and 8 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 8 - Edge from vertex 1 to vertex 5 with weight 2 - Edge from vertex 1 to vertex 4 with weight 9 - Edge from vertex 2 to vertex 3 with weight 5 - Edge from vertex 2 to vertex 5 with weight 5 - Edge from vertex 4 to vertex 5 with weight 6 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 8

setUp5	GraphALTest	Adds 5 vertices and 8 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 8 - Edge from vertex 1 to vertex 5 with weight 2 - Edge from vertex 1 to vertex 4 with weight 9 - Edge from vertex 2 to vertex 3 with weight 5 - Edge from vertex 2 to vertex 5 with weight 5 - Edge from vertex 4 to vertex 5 with weight 6 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 8
setUp6	GraphALTest	Adds 5 vertices and 8 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 1

		- Edge from vertex 1 to vertex 5 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1 - Edge from vertex 2 to vertex 3 with weight 1 - Edge from vertex 2 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 1
setUp7	GraphALTest	Adds 3 vertices and 3 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2 - Vertex 3 with key 3 and value 3 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 3 with weight 3 - Edge from vertex 2 to vertex 3 with weight 1
setUp8	GraphALTest	Adds 5 vertices and 6 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1

		- Edges: - Edge from vertex 1 to vertex 2 with weight 3 - Edge from vertex 1 to vertex 5 with weight 1 - Edge from vertex 1 to vertex 4 with weight 5 - Edge from vertex 4 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 3 with weight 2 - Edge from vertex 3 to vertex 5 with weight 1
setUp9	GraphALTest	Adds 5 vertices and 6 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 3 - Edge from vertex 1 to vertex 5 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1 - Edge from vertex 4 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 1
setUp10	GraphALTest	Adds 5 vertices and 6 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2

		- Vertex 3 with key 3 and value 3 - Vertex 4 with key 4 and value 4 - Vertex 5 with key 5 and value 5 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 4 with weight 10 - Edge from vertex 2 to vertex 4 with weight 5 - Edge from vertex 4 to vertex 3 with weight 2 - Edge from vertex 4 to vertex 5 with weight 11 - Edge from vertex 3 to vertex 5 with weight 1

Tests:

Test Goal: Test the correct insertion of vertices with the insertVertex method

Class	Method	Scenario	Input values	Result
GraphAL	insertVertex	setUp1	key: 1, 2, 3, 4 value: 1, 2, 3, 4	Verifies that the number of vertices in the graph is 4
GraphAL	insertVertex	setUp1	key: 1, 1, 1, 1 value: 1, 1, 1, 1	Verifies that the number of vertices in the graph is 1
GraphAL	insertVertex	setUp1	key: 1, 1, 3, 3 value: 1, 2, 3, 4	Verifies that the number of vertices in the graph is 2

Test Goal: Test the correct insertion of edges with the insertEdge method				
Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAL	insertEdge	setUp1	- Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2 - Vertex 3 with key 3 and value 3 - Vertex 4 with key 4 and value 4 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 3 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1	Verifies that the number of edges is 3
GraphAL	insertEdge	setUp1	- Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2 - Vertex 3 with key 3 and value 3	Verifies that the number of edges is 2

			- Vertex 4 with key 4 and value 4 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1	
GraphAL	insertEdge	setUp2	vertex3 vertex5	Verifies that the vertices are connected

Test Goal: Test the correct behavior of the searchVertex method

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAL	searchVertex	setUp1	- Vertex 1 with key 1 and value 100	Verifies that the result of searching for the vertex5 is null
GraphAL	searchVertex	setUp1	- Vertex 1 with key 1 and value 100	Verifies that the result of searching for the vertex1 is not null
GraphAL	searchVertex	setUp1	Insert a vertex with key 1 and value 100, then it is removed	Verifies that the result of searching for the vertex1 is null

Test Goal: Test the correct behavior of the removeVertex method				
Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAL	removeVertex	setUp1	- Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Edge from vertex 1 to vertex 2 with weight 1 Then the vertex1 is removed	Verifies that the vertex1 is not connected with the vertex2
GraphAL	removeVertex	setUp2	Remove all the vertices in the graph	Verifies that the number of vertices in the graph is 0
GraphAL	removeVertex	setUp1	Remove the vertex1	Verifies that the number of vertices in the graph is 0

Test Goal: Test the correct behavior of the dijkstra algorithm				
Class	**Method**	**Scenario**	**Input values**	**Result**

GraphAL	dijkstra	setUp10	vertex1 vertex5	Verifies that the path of the vertices is 4
GraphAL	dijkstra	setUp10	vertex1 vertex5	Verifies that the path of the vertices is 4
GraphAL	dijkstra	setUp10	vertex1 vertex5	Verifies that the path of the vertices is 5

Test Goal: Test the correct behavior of the floydWarshall algorithm				
Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAL	floydWarshall	setUp5	int[][] expectedDistances	Verifies that the distance matrix matches the expected matrix
GraphAL	floydWarshall	setUp5	int[][] expectedDistances	Verifies that the distance matrix matches the expected matrix
GraphAL	floydWarshall	setUp5	int[][] expectedDistances	Verifies that the distance matrix matches the expected matrix

Test Goal: Test the correct behavior of the prim algorithm

Class	Method	Scenario	Input values	Result
GraphAL	prim	setUp4	graph Then create a minimum spanning tree of the graph	Verifies that: The minimum spanning tree (MST) has 5 vertices and 4 edges.
GraphAL	prim	setUp4	graph vertex1 vertex5 Then create a minimum spanning tree of the graph	Verifies that the vertex1 and the vertex5 are connected in the minimum spanning tree
GraphAL	prim	setUp4	graph vertex2 vertex3 Then create a minimum spanning tree of the graph	Verifies that the vertex2 and the vertex3 are connected in the minimum spanning tree

Test Goal: Test the correct behavior of the kruskal algorithm

Class	Method	Scenario	Input values	Result
GraphAL	kruskal	setUp4	graph vertex2 vertex5 Then create a minimum spanning tree of the graph	Verifies that the vertex2 and the vertex5 are connected in the minimum spanning tree
GraphAL	kruskal	setUp4	graph Then create a minimum spanning tree of the graph	Verifies that the vertex1 is in the minimum spanning tree
GraphAL	kruskal	setUp4	graph Then create a minimum spanning tree of the graph	Verifies that the minimum spanning tree has 5 vertices

Test Goal: Test the correct behavior of the BFS algorithm

Class	Method	Scenario	Input values	Result

GraphAL	BFS	setUp5	vertex1 vertex2 vertex3 vertex4 vertex5 Then performs the BFS algorithm in the graph	Verifies that all the vertices have been visited
GraphAL	BFS	setUp5	vertex3 vertex4 Then performs the BFS algorithm in the graph	Verifies that the vertex3 is the predecessor of the vertex4
GraphAL	BFS	setUp5	vertex5 Then performs the BFS algorithm in the graph starting from the vertex5.	Verifies that the distance of the vertex5 is 0

Test Goal: Test the correct behavior of the DFS algorithm

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAL	DFS	setUp7	graph Then performs the prim and	Verifies that the total weight of the minimum spanning tree is 2

			DFS algorithms in the graph	
GraphAL	DFS	setUp8	graph Then performs the prim and DFS algorithms in the graph	Verifies that the total weight of the minimum spanning tree is 7
GraphAL	DFS	setUp7	graph Then performs the prim and DFS algorithms in the graph	Verifies that the total weight of the minimum spanning tree is 6

Graph Adjacency Matrix

Scenario Configuration

Name	Class	Scenario
setUp1	GraphAMTest	Creates a new GraphAM empty
setUp2	GraphAMTest	Adds 5 vertices and 8 edges to the graph: - Vertices: Vertex 1 with key 1 and value 1 Vertex 2 with data 1 and value 1 Vertex 3 with data 1 and value 1 Vertex 4 with data 1 and value 1 Vertex 5 with data 1 and value 1 - Edges: Edge from vertex 1 to vertex 2 with weight 1 Edge from vertex 1 to vertex 5 with weight 1 Edge from vertex 1 to vertex 4 with weight 1 Edge from vertex 2 to vertex 3 with weight 1 Edge from vertex 2 to vertex 5 with weight 1 Edge from vertex 4 to vertex 5 with weight 1 Edge from vertex 4 to vertex 3 with weight 1

		Edge from vertex 3 to vertex 5 with weight 1
setUp3	GraphAMTest	Adds 2 vertices and 2 edges to the graph: - Vertices: Vertex 1 with key 1 and value 100 Vertex 2 with data 2 and value 100 - Edges: Edge from vertex 1 to vertex 2 with weight 10 Edge from vertex 2 to vertex 1 with weight 20
setUp4	GraphAMTest	Adds 5 vertices and 8 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 8 - Edge from vertex 1 to vertex 5 with weight 2 - Edge from vertex 1 to vertex 4 with weight 9 - Edge from vertex 2 to vertex 3 with weight 5 - Edge from vertex 2 to vertex 5 with weight 5 - Edge from vertex 4 to vertex 5 with weight 6 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 8

setUp5	GraphAMTest	Adds 5 vertices and 8 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 8 - Edge from vertex 1 to vertex 5 with weight 2 - Edge from vertex 1 to vertex 4 with weight 9 - Edge from vertex 2 to vertex 3 with weight 5 - Edge from vertex 2 to vertex 5 with weight 5 - Edge from vertex 4 to vertex 5 with weight 6 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 8
setUp6	GraphAMTest	Adds 5 vertices and 8 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 1

		- Edge from vertex 1 to vertex 5 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1 - Edge from vertex 2 to vertex 3 with weight 1 - Edge from vertex 2 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 1
setUp7	GraphAMTest	Adds 3 vertices and 3 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2 - Vertex 3 with key 3 and value 3 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 3 with weight 3 - Edge from vertex 2 to vertex 3 with weight 1
setUp8	GraphAMTest	Adds 5 vertices and 6 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1

		- Edges: - Edge from vertex 1 to vertex 2 with weight 3 - Edge from vertex 1 to vertex 5 with weight 1 - Edge from vertex 1 to vertex 4 with weight 5 - Edge from vertex 4 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 3 with weight 2 - Edge from vertex 3 to vertex 5 with weight 1
setUp9	GraphAMTest	Adds 5 vertices and 6 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Vertex 3 with key 3 and value 1 - Vertex 4 with key 4 and value 1 - Vertex 5 with key 5 and value 1 - Edges: - Edge from vertex 1 to vertex 2 with weight 3 - Edge from vertex 1 to vertex 5 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1 - Edge from vertex 4 to vertex 5 with weight 1 - Edge from vertex 4 to vertex 3 with weight 1 - Edge from vertex 3 to vertex 5 with weight 1
setUp10	GraphAMTest	Adds 5 vertices and 6 edges to the graph: - Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2

		- Vertex 3 with key 3 and value 3 - Vertex 4 with key 4 and value 4 - Vertex 5 with key 5 and value 5 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 4 with weight 10 - Edge from vertex 2 to vertex 4 with weight 5 - Edge from vertex 4 to vertex 3 with weight 2 - Edge from vertex 4 to vertex 5 with weight 11 - Edge from vertex 3 to vertex 5 with weight 1

Tests:

Test Goal: Test the correct insertion of vertices with the insertVertex method

Class	Method	Scenario	Input values	Result
GraphAM	insertVertex	setUp1	key: 1, 2, 3, 4 value: 1, 2, 3, 4	Verifies that the number of vertices in the graph is 4
GraphAM	insertVertex	setUp1	key: 1, 1, 1, 1 value: 1, 1, 1, 1	Verifies that the number of vertices in the graph is 1
GraphAM	insertVertex	setUp1	key: 1, 1, 3, 3 value: 1, 2, 3, 4	Verifies that the number of vertices in the graph is 2

Test Goal: Test the correct insertion of edges with the insertEdge method

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	insertEdge	setUp1	- Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2 - Vertex 3 with key 3 and value 3 - Vertex 4 with key 4 and value 4 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 3 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1	Verifies that the number of edges is 3
GraphAM	insertEdge	setUp1	- Vertices: - Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 2 - Vertex 3 with key 3 and value 3	Verifies that the number of edges is 2

			- Vertex 4 with key 4 and value 4 - Edges: - Edge from vertex 1 to vertex 2 with weight 1 - Edge from vertex 1 to vertex 4 with weight 1	
GraphAM	insertEdge	setUp2	vertex3 vertex5	Verifies that the vertices are connected

Test Goal: Test the correct behavior of the searchVertex method

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	searchVertex	setUp1	- Vertex 1 with key 1 and value 100	Verifies that the result of searching for the vertex5 is null
GraphAM	searchVertex	setUp1	- Vertex 1 with key 1 and value 100	Verifies that the result of searching for the vertex1 is not null
GraphAM	searchVertex	setUp1	Insert a vertex with key 1 and value 100, then it is removed	Verifies that the result of searching for the vertex1 is null

Test Goal: Test the correct behavior of the removeVertex method

Class	Method	Scenario	Input values	Result
GraphAM	removeVertex	setUp1	- Vertex 1 with key 1 and value 1 - Vertex 2 with key 2 and value 1 - Edge from vertex 1 to vertex 2 with weight 1 Then the vertex1 is removed	Verifies that the vertex1 is not connected with the vertex2
GraphAM	removeVertex	setUp2	Remove all the vertices in the graph	Verifies that the number of vertices in the graph is 0
GraphAM	removeVertex	setUp1	Remove the vertex1	Verifies that the number of vertices in the graph is 0

Test Goal: Test the correct behavior of the dijkstra algorithm

Class	Method	Scenario	Input values	Result
GraphAM	dijkstra	setUp10	vertex1 vertex5	Verifies that the path of the vertices is 4
GraphAM	dijkstra	setUp10	vertex1 vertex5	Verifies that the path of the vertices is 4

GraphAM	dijkstra	setUp10	vertex1 vertex5	Verifies that the path of the vertices is 5

Test Goal: Test the correct behavior of the floydWarshall algorithm				
Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	floydWarshal l	setUp5	int[][] expectedDistances	Verifies that the distance matrix matches the expected matrix
GraphAM	floydWarshal l	setUp5	int[][] expectedDistances	Verifies that the distance matrix matches the expected matrix
GraphAM	floydWarshal l	setUp5	int[][] expectedDistances	Verifies that the distance matrix matches the expected matrix

Test Goal: Test the correct behavior of the prim algorithm				
Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	prim	setUp4	graph	Verifies that: The minimum spanning tree (MST) has 5 vertices and 4 edges.

			Then create a minimum spanning tree of the graph	
GraphAM	prim	setUp4	graph vertex1 vertex5 Then create a minimum spanning tree of the graph	Verifies that the vertex1 and the vertex5 are connected in the minimum spanning tree
GraphAM	prim	setUp4	graph vertex2 vertex3 Then create a minimum spanning tree of the graph	Verifies that the vertex2 and the vertex3 are connected in the minimum spanning tree

Test Goal: Test the correct behavior of the kruskal algorithm

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	kruskal	setUp4	graph vertex2 vertex5 Then create a minimum spanning tree of the graph	Verifies that the vertex2 and the vertex5 are connected in the minimum spanning tree

GraphAM	kruskal	setUp4	graph Then create a minimum spanning tree of the graph	Verifies that the vertex1 is in the minimum spanning tree
GraphAM	kruskal	setUp4	graph Then create a minimum spanning tree of the graph	Verifies that the minimum spanning tree has 5 vertices

Test Goal: Test the correct behavior of the BFS algorithm

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	BFS	setUp5	vertex1 vertex2 vertex3 vertex4 vertex5 Then performs the BFS algorithm in the graph	Verifies that all the vertices have been visited
GraphAM	BFS	setUp5	vertex3 vertex4 Then performs the BFS algorithm in the graph	Verifies that the vertex3 is the predecessor of the vertex4

GraphAM	BFS	setUp5	vertex5 Then performs the BFS algorithm in the graph starting from the vertex5.	Verifies that the distance of the vertex5 is 0

Test Goal: Test the correct behavior of the DFS algorithm

Class	**Method**	**Scenario**	**Input values**	**Result**
GraphAM	DFS	setUp7	graph Then performs the prim and DFS algorithms in the graph	Verifies that the total weight of the minimum spanning tree is 2
GraphAM	DFS	setUp8	graph Then performs the prim and DFS algorithms in the graph	Verifies that the total weight of the minimum spanning tree is 7
GraphAM	DFS	setUp7	graph Then performs the prim and DFS algorithms in the graph	Verifies that the total weight of the minimum spanning tree is 6