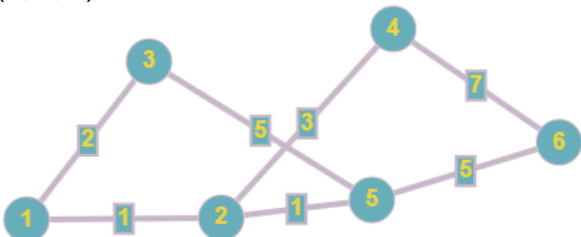
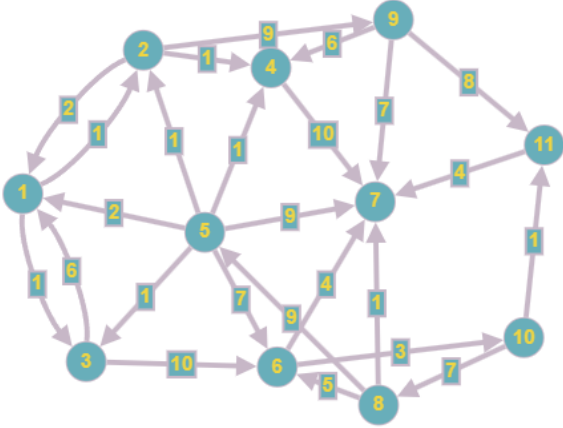
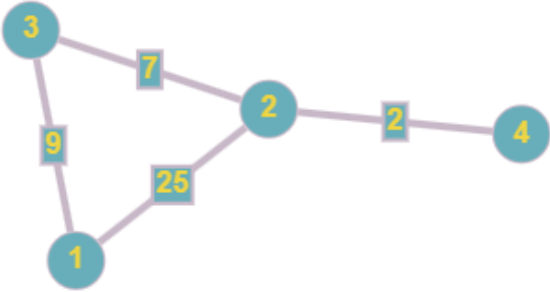


## Test Case Graph

Name	Class	Scenery
setUpStageSimpleGraph()	GraphAdjacentListTest GraphAdjacentMatrixTest	<p>A no directed and weighted graph implemented with adjacency matrix with the next vertexes: '1', '2', '3', '4', '5', '6'</p> <p>And the next edges, (From, to, weight):</p> <p>Edges:</p> <p>(1, 2, 1) (1, 3, 2) (2, 4, 3) (2, 5, 1) (3, 5, 5) (4, 6, 7) (5, 6, 5)</p> 
setUpStageDirected()	GraphAdjacentMatrixTest GraphAdjacentListTest	<p>A directed and weighted graph implemented with adjacency matrix with the next vertexes: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</p> <p>And the next edges, (From, to, weight):</p> <p>Edges:</p> <p>(1, 2, 1) (2, 1, 2) (1, 3, 1) (3, 1, 6) (2, 4, 1) (9, 7, 7) (9, 11, 8) (6, 10, 3) (11, 7, 4) (8, 6, 5) (10, 11, 1) (5, 1, 2) (5, 6, 7) (5, 3, 1) (9, 4, 6) (5, 7, 9) (3, 6, 10) (4, 7, 10) (6, 7, 4) (10, 8, 7) (8, 5, 9)</p>

		<p>(5, 2, 1) (5, 4, 1) (2, 9, 9) (8, 7, 1)</p> 
setUpGraphWithoutConected()	GraphAdjacentListTest GraphAdjacentMatrixTest	<p>A no directed and weighted graph implemented with adjacency matrix with the next vertices: 1, 2, 3, 4, 5, 6</p> <p>And the next edges, (From, to, weight)-&gt;∅</p>
setUpGraphSimpleWithKeyIntAndValueString()	GraphAdjacentListTest GraphAdjacentMatrixTest	<p>A no directed and weighted graph implemented with adjacency matrix with the next vertices: 1, 2,3,4</p> <p>And the next edges, (From, to, weight) Edges: (1, 3, 9) (2, 4, 2) (2, 3, 7) (1, 2, 25)</p> 
setUpGraphWithConectionWithSameWeight()	GraphAdjacentListTest GraphAdjacentMatrixTest	<p>A no directed and weighted graph implemented with adjacency matrix with the next vertices: 1, 2,3,4,5</p>

And the next edges, (From, to, weight)

Edges:

(1, 3, 9)

(1, 2, 1)

(1, 3, 1)

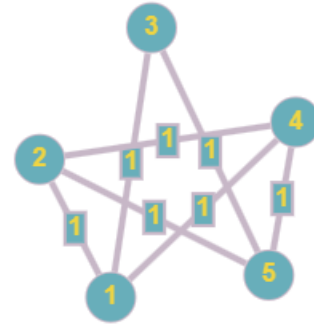
(2, 4, 1)

(2, 5, 1)

(3, 5, 1)

(4, 1, 1)

(5, 4, 1)



**Test objective:**

Que los métodos de añadir vértices de las clases GraphAdjacentListTest y GraphAdjacentMatrixTest funcione correctamente

Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	addVertex()	setUpStageSimpleGraph()	vertex{key=7, value=7}	Que se haya agregado correctamente la vértice al grafo simple, por lo cual debe retornar un true
GraphAdjacentListTest GraphAdjacentMatrixTest	addVertex()	setUpStageDirected()	vertex{key=12, value=12}	Que se haya agregado correctamente la vértice al grafo directo, por lo cual debe retornar un true
GraphAdjacentListTest GraphAdjacentMatrixTest	addVertex()	setUpStageSimpleGraph()	vertex{key=1, value=1}	Debe retornar false, ya que existe ese nodo en el grafo.

**Test objective:**

Que los métodos de añadir edges de las clases GraphAdjacentListTest y GraphAdjacentMatrixTest

funcione correctamente				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	addEdge()	setUpStageSimpleGraph()	graph.addEdge(7,10,10)  Vertex1=7 Vertex2=10 Weight=10	Debe retornar la excepcion exceptionNoVertexExist, ya que las vertices no existen en este escenario del grafo
GraphAdjacentListTest GraphAdjacentMatrixTest	addEdge()	setUpStageSimpleGraph()	graph.addEdge(1,1,10)  Vertex1=1 Vertex2=1 Weight=10	Debe retornar la excepción exceptionOnGraphTypeNotAllowed, ya que se agrega un edge en el grafo simple formando un bucle, algo que no permite este tipo de grafo.
GraphAdjacentListTest GraphAdjacentMatrixTest	addEdge()	setUpStageSimpleGraph()	Vertex1=1 Vertex2=2 Weight=10	Debe retornar un true, significando que se añadió el edge correctamente

<b>Test objective:</b> Que los métodos de eliminar vertex de las clases GraphAdjacentListTest y GraphAdjacentMatrixTest funcione correctamente				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	removeVertex()	setUpStageSimpleGraph()	Todas las llaves de los nodos	Se eliminan todos los 6 nodos del grafo simple, por lo cual debe retornar true. Y el tamaño del grafo debe ser 0.
GraphAdjacentListTest GraphAdjacentMatrixTest	removeVertex()	setUpStageDirected()	Todas las llaves de los nodos	Se eliminan todos los 11 nodos del grafo dirigido, por lo cual debe retornar true. Y el tamaño del grafo debe ser 0.permite este tipo de grafo.
GraphAdjacentListTest	removeVertex()	setUpStageSimpleGraph()	graph.removeV	Debe retornar un

st GraphAdjacentMatrix Test		pleGraph()	ertex(7);  vertex=17	false, ya que no existe el vértice 7 en el grafo simple
GraphAdjacentListTe st GraphAdjacentMatrix Test	removeVertex()	setUpStageDire cted()	graph.removeV ertex(12);  vertex=12	Debe retornar un false, ya que no existe el vértice 12 en el grafo directo

<b>Test objective:</b> Que los métodos de eliminar edges de las clases GraphAdjacentListTest y GraphAdjacentMatrixTest funcione correctamente				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTe st GraphAdjacentMatrix Test	removeEdge()	setUpStageSim pleGraph()	Todas las conexiones del grafo	Se eliminan todas las 7 conexiones del grafo simple, por lo cual los retornos deben ser true.
GraphAdjacentListTe st GraphAdjacentMatrix Test	removeEdge()	setUpStageDire cted()	Todas las conexiones del grafo	Se eliminan todas las conexiones del grafo dirigido, por lo cual debe retornar true.
GraphAdjacentListTe st GraphAdjacentMatrix Test	removeEdge()	setUpStageSim pleGraph()	graph.removeE dge(1,7)  vertex=1 vertex=7	Debe retornar la excepción exceptionNoVertexE xist, por tratar de eliminar una conexión y nodo que no existe en un grafo simple
GraphAdjacentListTe st GraphAdjacentMatrix Test	removeEdge()	setUpStageDire cted()	graph.removeE dge(1,12)	Debe retornar la excepción exceptionNoVertexE xist, por tratar de eliminar una conexión y nodo que no existe en un grafo dirigido

**Test objective:**

Que los métodos de BFS de las clases GraphAdjacentListTest y GraphAdjacentMatrixTest funcione correctamente, con grafo simple, dirigido, sin conexiones, mismas conexiones.

Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	BFS()	setUpStageDirected()	values of distances= 0,1,1,2,5,2,3,4,2,3,3.	Los nodos del grafo debe tener esta distancia  vertex=distance 1->0 2->1 3->1 4->2 5->5 6->2 7->3 8->4 9->2 10->3 11->3
GraphAdjacentListTest GraphAdjacentMatrixTest	BFS()	setUpStageSimpleGraph()	values of distances= 0,1,1,2,2,3	Los nodos del grafo debe tener esta distancia  vertex=distance 1->0 2->1 3->1 4->2 5->2 6->3
GraphAdjacentListTest GraphAdjacentMatrixTest	BFS()	setUpGraphWithoutConnected()		Todos los nodos deben retornar cada nodo del grafo debe retornar Color.WHITE y la distancia debe tender a infinito (un numero muy grande), excepto el primer nodo que es 0 y el color es negro.
GraphAdjacentListTest GraphAdjacentMatrixTest	BFS()	setUpGraphWithConnectionWithSameWeight()	values of distances= 0,1,1,1,2	Los nodos del grafo debe tener esta distancia

				vertex=distance 1->0 2->1 3->1 4->2 5->2
GraphAdjacentListTest GraphAdjacentMatrixTest	BFS()	setUpStageDirected()	vertexKey=12	Debe retornar la excepcion exceptionNoVertexExist, porque el vertexKey =12 no existe en el grafo.

<b>Test objective:</b> Que los métodos de Dijkstra de la clase GraphAdjacentListTest y GraphAdjacentMatrixTest funcione correctamente				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	Dijkstra()	setUpStageDirected()	keyVertexSource=1 graph.dijkstra(1)  result = new ArrayList<>(Arrays.asList(0,1,1,2,30,11,12,21,10,14,15));	El arraylist de distancia de conexiones del valor de entrada 'result' debe ser igual al que retorne el dijkstra, ya que será la distancia que cada nodo del grafo directo
GraphAdjacentListTest GraphAdjacentMatrixTest	Dijkstra()	setUpStageSimpleGraph()	keyVertexSource=1 graph.dijkstra(1)  result = new ArrayList<>(Arrays.asList(0,1,2,4,2,7));	El arraylist de distancia de conexiones del valor de entrada 'result' debe ser igual al que retorne el dijkstra, ya que será la distancia que cada nodo del grafo simple
GraphAdjacentListTest GraphAdjacentMatrixTest	Dijkstra()	setUpStageSimpleGraph()	keyVertexSource=7 graph.dijkstra(7)	debe retornar la excepcion exceptionNoVertexExist ya que la keyvertexSource=7 no existe en el grafo.

GraphAdjacentListTest GraphAdjacentMatrixTest	Dijkstra()	setUpGraphWithoutConected()	keyVertexSource=1 graph.dijkstra(1)  result = new ArrayList<>(Arrays.asList(0,infinity,infinity,infinity,infinity,infinity));  Valor de infinity (infinity=Integer.MAX_VALUE-100)	Los nodos del grafo debe retornar un valor que tienda a infinito todos en su distancia.
GraphAdjacentListTest GraphAdjacentMatrixTest	Dijkstra()	setUpGraphWithConecctionWithSameWeight	result = new ArrayList<>(Arrays.asList(0,1,1,1,2));	No debería lanzar una excepción

<b>Test objective:</b> Probar si el algoritmo de Kruskal funciona correctamente para las dos implementaciones de grafo.				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	Kruskal()	setUpGraphWithConecctionWithSameWeight	result=graph.kruskal();	Los vértices que tiene result deben ser los esperados, como todos los pesos son iguales entonces se debería retornar en el orden en el que se añadieron
GraphAdjacentListTest GraphAdjacentMatrixTest	Kruskal()	setUpGraphSimpleWithKeyIntAndValueString	result=graph2.kruskal()	Debe retornar en modo de arraylist la lista de aristas del grafo ordenados de forma ascendente dado el peso y ser los mismos que se insertaron en un principio al grafo
GraphAdjacentListTest GraphAdjacentMatrixTest	Kruskal()	setUpGraphWithoutConected	result=graph.kruskal()	El tamaño de result debe ser 0 porque el grafo no tiene aristas
GraphAdjacentListTest	Kruskal()	setUpStageDirected		Usar la función Kruskal debería tirar



GraphAdjacentMatrixTest				una excepción por ser un grafo dirigido
-------------------------	--	--	--	---

<b>Test objective:</b> Probar si el algoritmo de DFS funciona correctamente para las dos implementaciones de grafo.				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	DFS()	setUpGraphWithConecctionWithSameWeight		El recorrido del grafo debe hacerse por profundidad, aun cuando los pesos del grafo sean iguales
GraphAdjacentListTest GraphAdjacentMatrixTest	DFS()	setUpStageSimpleGraph		El recorrido del grafo debe hacerse por profundidad y los valores deben ser iguales en orden a los esperados.
GraphAdjacentListTest GraphAdjacentMatrixTest	DFS()	setUpStageDirected		Se verifica que el recorrido se dió en el orden a pesar de ser un grafo dirigido
GraphAdjacentListTest GraphAdjacentMatrixTest	DFS()	setUpGraphWithoutConeccted		Se verifica que el algoritmo de DFS funciona aun cuando el grafo no todos los vértices del grafo están conectados

<b>Test objective:</b> Probar si el algoritmo de DFS funciona correctamente para las dos implementaciones de grafo.				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	floydWarshall()	setUpGraphWithConecctionWithSameWeight	result = graph.floydWarshall()	Se debería retornar una matriz con los pesos mínimos para llegar de un nodo a cualquier otro.
GraphAdjacentListTest GraphAdjacentMatrixTest	floydWarshall()	setUpGraphWithoutConeccted	result = graph.floydWarshall()	Para un grafo en el que sus nodos no están conectados, el algoritmo de floyd

				warshall debe retornar una matriz llena de infinitos
GraphAdjacentListTest GraphAdjacentMatrixTest	floydWarshall()	setUpStageSimpleGraph	result = graph.floydWarshall()	Debe retornar la matriz esperada en un caso normal de un grafo simple
GraphAdjacentListTest GraphAdjacentMatrixTest	floydWarshall()	setUpGraphSimpleWithKeyIntAndValueString	result = graph2.floydWarshall()	Debe retornar la matriz esperada aun cuando las llaves son enteros y los valores Strings

<b>Test objective:</b> Probar si el algoritmo de Prim funciona correctamente para las dos implementaciones de grafo.				
Class	Method	Scenery	Inputs Value	Result
GraphAdjacentListTest GraphAdjacentMatrixTest	prim()	setUpGraphWithoutConected	result = prim()	La lista de arista debe tener tamaño cero porque el grafo no tenía aristas
GraphAdjacentListTest GraphAdjacentMatrixTest	prim()	setUpStageSimpleGraph	result = prim()	Se prueba el resultado del algoritmo de prim en un grafo simple
GraphAdjacentListTest GraphAdjacentMatrixTest	prim()	setUpStageDirected		Llamar al método de prim en un grafo dirigido debería lanzar una excepción : UnsupportedOperationException
GraphAdjacentListTest GraphAdjacentMatrixTest	prim()	setUpGraphWithConecctionWithSameWeight	result = prim()	En caso de que hayan varias aristas con el mismo peso, el algoritmo de prim analiza primero los que primero se añadieron al grafo