JavaScript is disabled on your browser.

[Skip navigation links](#1fob9te)

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/WeightedGraph.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/graph/MazeGraph.html)
* Next Class
* [Frames](http://docs.google.com/index.html?graph/WeightedGraph.html)
* [No Frames](http://docs.google.com/WeightedGraph.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* [Constr](#3znysh7) |
* [Method](#2et92p0)
* Detail:
* Field |
* [Constr](#3dy6vkm) |
* [Method](#4d34og8)

graph

## Class WeightedGraph<V>

* java.lang.Object
  + graph.WeightedGraph<V>
* Direct Known Subclasses: [MazeGraph](http://docs.google.com/graph/MazeGraph.html)  
    
  public class WeightedGraph<V>  
  extends java.lang.Object  
  This class represents a general "directed graph", which could be used for any purpose. The graph is viewed as a collection of vertices, which are sometimes connected by weighted, directed edges.  
  This graph will never store duplicate vertices.  
  The weights will always be non-negative integers.  
  The WeightedGraph will be capable of performing three algorithms: Depth-First-Search, Breadth-First-Search, and Djikatra's.  
  The Weighted Graph will maintain a collection of "GraphAlgorithmObservers", which will be notified during the performance of the graph algorithms to update the observers on how the algorithms are progressing.

### Constructor SummaryConstructors

|  |
| --- |
| * + Constructor and Description |
| * + [WeightedGraph](http://docs.google.com/graph/WeightedGraph.html#WeightedGraph--)() Initialize the data structures to "empty", including the collection of GraphAlgorithmObservers. |

### Method SummaryAll Methods Instance Methods Concrete Methods

|  |  |
| --- | --- |
| * + Modifier and Type | * + Method and Description |
| * + void | * + [addEdge](http://docs.google.com/graph/WeightedGraph.html#addEdge-V-V-java.lang.Integer-)([V](http://docs.google.com/graph/WeightedGraph.html) from, [V](http://docs.google.com/graph/WeightedGraph.html) to, java.lang.Integer weight) Add an edge from one vertex of the graph to another, with the weight specified. |
| * + void | * + [addObserver](http://docs.google.com/graph/WeightedGraph.html#addObserver-graph.GraphAlgorithmObserver-)([GraphAlgorithmObserver](http://docs.google.com/graph/GraphAlgorithmObserver.html)<[V](http://docs.google.com/graph/WeightedGraph.html)> observer) Add a GraphAlgorithmObserver to the collection maintained by this graph. |
| * + void | * + [addVertex](http://docs.google.com/graph/WeightedGraph.html#addVertex-V-)([V](http://docs.google.com/graph/WeightedGraph.html) vertex) Add a vertex to the graph. |
| * + boolean | * + [containsVertex](http://docs.google.com/graph/WeightedGraph.html#containsVertex-V-)([V](http://docs.google.com/graph/WeightedGraph.html) vertex) Searches for a given vertex. |
| * + void | * + [DoBFS](http://docs.google.com/graph/WeightedGraph.html#DoBFS-V-V-)([V](http://docs.google.com/graph/WeightedGraph.html) start, [V](http://docs.google.com/graph/WeightedGraph.html) end) This method will perform a Breadth-First-Search on the graph. |
| * + void | * + [DoDFS](http://docs.google.com/graph/WeightedGraph.html#DoDFS-V-V-)([V](http://docs.google.com/graph/WeightedGraph.html) start, [V](http://docs.google.com/graph/WeightedGraph.html) finish) This method will perform a Depth-First-Search on the graph. |
| * + void | * + [DoDijsktra](http://docs.google.com/graph/WeightedGraph.html#DoDijsktra-V-V-)([V](http://docs.google.com/graph/WeightedGraph.html) start, [V](http://docs.google.com/graph/WeightedGraph.html) end) Perform Dijkstra's algorithm, beginning at the "start" vertex. |
| * + java.lang.Integer | * + [getWeight](http://docs.google.com/graph/WeightedGraph.html#getWeight-V-V-)([V](http://docs.google.com/graph/WeightedGraph.html) from, [V](http://docs.google.com/graph/WeightedGraph.html) to) Returns weight of the edge connecting one vertex to another. |

### Methods inherited from class java.lang.Objectequals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### WeightedGraph public WeightedGraph() Initialize the data structures to "empty", including the collection of GraphAlgorithmObservers.

### Method Detail

#### addObserver public void addObserver([GraphAlgorithmObserver](http://docs.google.com/graph/GraphAlgorithmObserver.html)<[V](http://docs.google.com/graph/WeightedGraph.html)> observer) Add a GraphAlgorithmObserver to the collection maintained by this graph.Parameters: observer -

#### addVertex public void addVertex([V](http://docs.google.com/graph/WeightedGraph.html) vertex) Add a vertex to the graph. If the vertex is already in the graph, throw an IllegalArgumentException.Parameters: vertex - vertex to be added to the graph Throws: java.lang.IllegalArgumentException - if the vertex is already in the graph

#### containsVertex public boolean containsVertex([V](http://docs.google.com/graph/WeightedGraph.html) vertex) Searches for a given vertex.Parameters: vertex - the vertex we are looking for Returns: true if the vertex is in the graph, false otherwise.

#### addEdge public void addEdge([V](http://docs.google.com/graph/WeightedGraph.html) from, [V](http://docs.google.com/graph/WeightedGraph.html) to, java.lang.Integer weight) Add an edge from one vertex of the graph to another, with the weight specified. The two vertices must already be present in the graph. This method throws an IllegalArgumentExeption in three cases: 1. The "from" vertex is not already in the graph. 2. The "to" vertex is not already in the graph. 3. The weight is less than 0.Parameters: from - the vertex the edge leads from to - the vertex the edge leads to weight - the (non-negative) weight of this edge Throws: java.lang.IllegalArgumentException - when either vertex is not in the graph, or the weight is negative.

#### getWeight public java.lang.Integer getWeight([V](http://docs.google.com/graph/WeightedGraph.html) from, [V](http://docs.google.com/graph/WeightedGraph.html) to) Returns weight of the edge connecting one vertex to another. Returns null if the edge does not exist. Throws an IllegalArgumentException if either of the vertices specified are not in the graph.Parameters: from - vertex where edge begins to - vertex where edge terminates Returns: weight of the edge, or null if there is no edge connecting these vertices Throws: java.lang.IllegalArgumentException - if either of the vertices specified are not in the graph.

#### DoBFS public void DoBFS([V](http://docs.google.com/graph/WeightedGraph.html) start, [V](http://docs.google.com/graph/WeightedGraph.html) end) This method will perform a Breadth-First-Search on the graph. The search will begin at the "start" vertex and conclude once the "end" vertex has been reached. Before the search begins, this method will go through the collection of Observers, calling notifyBFSHasBegun on each one. Just after a particular vertex is visited, this method will go through the collection of observers calling notifyVisit on each one (passing in the vertex being visited as the argument.) After the "end" vertex has been visited, this method will go through the collection of observers calling notifySearchIsOver on each one, after which the method should terminate immediately, without processing further vertices.Parameters: start - vertex where search begins end - the algorithm terminates just after this vertex is visited

#### DoDFS public void DoDFS([V](http://docs.google.com/graph/WeightedGraph.html) start, [V](http://docs.google.com/graph/WeightedGraph.html) finish) This method will perform a Depth-First-Search on the graph. The search will begin at the "start" vertex and conclude once the "end" vertex has been reached. Before the search begins, this method will go through the collection of Observers, calling notifyDFSHasBegun on each one. Just after a particular vertex is visited, this method will go through the collection of observers calling notifyVisit on each one (passing in the vertex being visited as the argument.) After the "end" vertex has been visited, this method will go through the collection of observers calling notifySearchIsOver on each one, after which the method should terminate immediately, without visiting further vertices.Parameters: start - vertex where search begins finish - the algorithm terminates just after this vertex is visited

#### DoDijsktra public void DoDijsktra([V](http://docs.google.com/graph/WeightedGraph.html) start, [V](http://docs.google.com/graph/WeightedGraph.html) end) Perform Dijkstra's algorithm, beginning at the "start" vertex. The algorithm DOES NOT terminate when the "end" vertex is reached. It will continue until EVERY vertex in the graph has been added to the finished set. Before the algorithm begins, this method goes through the collection of Observers, calling notifyDijkstraHasBegun on each Observer. Each time a vertex is added to the "finished set", this method goes through the collection of Observers, calling notifyDijkstraVertexFinished on each one (passing the vertex that was just added to the finished set as the first argument, and the optimal "cost" of the path leading to that vertex as the second argument.) After all of the vertices have been added to the finished set, the algorithm will calculate the "least cost" path of vertices leading from the starting vertex to the ending vertex. Next, it will go through the collection of observers, calling notifyDijkstraIsOver on each one, passing in as the argument the "lowest cost" sequence of vertices that leads from start to end (I.e. the first vertex in the list will be the "start" vertex, and the last vertex in the list will be the "end" vertex.)Parameters: start - vertex where algorithm will start end - special vertex used as the end of the path reported to observers via the notifyDijkstraIsOver method.

[Skip navigation links](#qsh70q)

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/WeightedGraph.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/graph/MazeGraph.html)
* Next Class
* [Frames](http://docs.google.com/index.html?graph/WeightedGraph.html)
* [No Frames](http://docs.google.com/WeightedGraph.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* [Constr](#3znysh7) |
* [Method](#2et92p0)
* Detail:
* Field |
* [Constr](#3dy6vkm) |
* [Method](#4d34og8)

[Web Accessibility](https://www.umd.edu/web-accessibility)