

Empirical Study project on network flow

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	Class Documentation	3
2.1	BipartiteGraph Class Reference	3
2.2	graphGenerationCode.Random.BuildGraph Class Reference	3
2.2.1	Member Function Documentation	3
2.2.1.1	main(String[] args)	3
2.3	graphCode.Edge Class Reference	4
2.3.1	Detailed Description	4
2.3.2	Constructor & Destructor Documentation	4
2.3.2.1	Edge(Vertex v, Vertex w, Object data, Object name)	4
2.3.3	Member Function Documentation	4
2.3.3.1	getData()	5
2.3.3.2	getFirstEndpoint()	5
2.3.3.3	getName()	5
2.3.3.4	getSecondEndpoint()	5
2.3.3.5	setData(Object data)	5
2.4	graphCode.GraphInput Class Reference	6
2.4.1	Detailed Description	6
2.4.2	Member Function Documentation	6
2.4.2.1	LoadSimpleGraph(SimpleGraph newgraph)	6
2.4.2.2	LoadSimpleGraph(SimpleGraph newgraph, String pathandfilename)	6
2.4.2.3	main(String args[])	7

2.5	graphCode.KeyboardReader Class Reference	7
2.5.1	Detailed Description	7
2.5.2	Member Function Documentation	8
2.5.2.1	main(String[] args)	8
2.5.2.2	readDouble()	8
2.5.2.3	readInt()	8
2.5.2.4	readString()	8
2.5.3	Member Data Documentation	8
2.5.3.1	EOI_DOUBLE	8
2.5.3.2	EOI_INT	8
2.5.3.3	EOI_STRING	9
2.5.3.4	ERROR_DOUBLE	9
2.5.3.5	ERROR_INT	9
2.5.3.6	ERROR_MESSAGES	9
2.5.3.7	ERROR_STRING	9
2.6	networkflowstudy.logging Class Reference	9
2.6.1	Detailed Description	9
2.6.2	Member Function Documentation	9
2.6.2.1	printEdge(Edge e)	9
2.6.2.2	printFlow(LinkedHashMap< Edge, Integer > flow)	10
2.6.2.3	printGraph(SimpleGraph G)	10
2.6.2.4	printPath(List< Vertex > path)	10
2.7	networkflowstudy.MaxFlow Class Reference	11
2.7.1	Detailed Description	11
2.7.2	Member Function Documentation	11
2.7.2.1	calculateFlow(Vertex sourceG, Vertex sinkG)	11
2.8	graphGenerationCode.Mesh.MeshGenerator Class Reference	12
2.8.1	Detailed Description	12
2.8.2	Constructor & Destructor Documentation	12
2.8.2.1	MeshGenerator(String[] args)	12

2.8.3	Member Function Documentation	13
2.8.3.1	generate()	13
2.8.3.2	main(String[] args)	13
2.9	networkflowstudy.PreflowPush Class Reference	13
2.9.1	Detailed Description	14
2.10	RandomGraph Class Reference	14
2.10.1	Member Function Documentation	14
2.10.1.1	graphBuilder(int v, int e, int min, int max)	14
2.10.1.2	main(String[] args)	14
2.11	networkflowstudy.SaveOutput Class Reference	15
2.11.1	Detailed Description	15
2.12	networkflowstudy.ScalingMaxFlow Class Reference	15
2.12.1	Detailed Description	16
2.12.2	Constructor & Destructor Documentation	16
2.12.2.1	ScalingMaxFlow(SimpleGraph G)	16
2.12.3	Member Function Documentation	16
2.12.3.1	calculateFlow(Vertex sourceG, Vertex sinkG)	16
2.12.3.2	getDelta(Vertex source)	16
2.13	graphCode.SimpleGraph Class Reference	17
2.13.1	Detailed Description	18
2.13.2	Member Function Documentation	18
2.13.2.1	aVertex()	18
2.13.2.2	edges()	18
2.13.2.3	getEdgeMap()	18
2.13.2.4	getVertex(String vertexName)	18
2.13.2.5	getVertexMap()	19
2.13.2.6	incidentEdges(Vertex v)	19
2.13.2.7	insertEdge(Vertex v, Vertex w, Object data, Object name)	19
2.13.2.8	insertVertex(Object data, Object name)	19
2.13.2.9	main(String[] args)	20

2.13.2.10 numEdges()	20
2.13.2.11 numVertices()	20
2.13.2.12 opposite(Vertex v, Edge e)	20
2.13.2.13 tail(Vertex v, Edge e)	21
2.13.2.14 vertices()	21
2.14 networkflowstudy.tc5543 Class Reference	21
2.14.1 Detailed Description	21
2.14.2 Member Function Documentation	21
2.14.2.1 main(String[] args)	21
2.15 networkflowstudy.utils Class Reference	22
2.15.1 Detailed Description	22
2.15.2 Member Function Documentation	22
2.15.2.1 augment(SimpleGraph G, SimpleGraph Gf, LinkedHashMap< Edge, Integer > flow, List< Vertex > path)	22
2.15.2.2 createResidualGraph(SimpleGraph G, LinkedHashMap< Edge, Integer > flow)	22
2.15.2.3 get_bottleneck(SimpleGraph Gf, List< Vertex > st_path)	23
2.15.2.4 getSTPath(SimpleGraph Gf, Vertex Sink, Vertex Source)	23
2.15.2.5 getSTPath(SimpleGraph Gf, Vertex sink, Vertex source, int delta)	23
2.15.2.6 initFlow(SimpleGraph G)	24
2.16 graphCode.Vertex Class Reference	24
2.16.1 Detailed Description	25
2.16.2 Constructor & Destructor Documentation	25
2.16.2.1 Vertex(Object data, Object name)	25
2.16.3 Member Function Documentation	25
2.16.3.1 getData()	25
2.16.3.2 getName()	25
2.16.3.3 setData(Object data)	25

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BipartiteGraph	3
graphGenerationCode.Random.BuildGraph	3
graphCode.Edge	4
graphCode.GraphInput	6
graphCode.KeyboardReader	7
networkflowstudy.logging	9
networkflowstudy.MaxFlow	11
graphGenerationCode.Mesh.MeshGenerator	12
networkflowstudy.PreflowPush	13
RandomGraph	14
networkflowstudy.SaveOutput	15
networkflowstudy.ScalingMaxFlow	15
graphCode.SimpleGraph	17
networkflowstudy.tc5543	21
networkflowstudy.utils	22
graphCode.Vertex	24

Chapter 2

Class Documentation

2.1 BipartiteGraph Class Reference

Static Public Member Functions

- static void **main** (String[] args) throws Exception
- static String **GetString** () throws IOException
- static int **GetInt** () throws IOException
- static double **GetReal** () throws IOException

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphGenerationCode/Bipartite/BipartiteGraph.java

2.2 graphGenerationCode.Random.BuildGraph Class Reference

Static Public Member Functions

- static void **main** (String[] args)

2.2.1 Member Function Documentation

2.2.1.1 static void graphGenerationCode.Random.BuildGraph.main (String[] args) [inline],[static]

Computes and saves a random graph based on the number of vertices, density of graph, lower bound on capacities, upper bound on capacities and the output file path.

Parameters

args	
------	--

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphGenerationCode/Random/BuildGraph.java

2.3 graphCode.Edge Class Reference

Public Member Functions

- [Edge](#) ([Vertex](#) *v*, [Vertex](#) *w*, Object *data*, Object *name*)
- [Vertex](#) [getFirstEndpoint](#) ()
- [Vertex](#) [getSecondEndpoint](#) ()
- Object [getData](#) ()
- void [setData](#) (Object *data*)
- Object [getName](#) ()

2.3.1 Detailed Description

Class that represents an edge in a graph. An object (usually some sort of data) can be associated with the edge.

A label (also represented by an object (e.g., a string) can also be associated with an edge. This could be useful, for example, if you need to mark an edge as being visited in some graph traversal.

Author

edhong

Version

0.0

2.3.2 Constructor & Destructor Documentation

2.3.2.1 graphCode.Edge.Edge ([Vertex](#) *v*, [Vertex](#) *w*, Object *data*, Object *name*) [inline]

Constructor that allows data and a name to be associated with the edge.

Parameters

<i>v</i>	the first endpoint of this edge
<i>w</i>	the second endpoint of this edge
<i>data</i>	data to be associated with this edge
<i>name</i>	a name to be associated with this edge

2.3.3 Member Function Documentation

2.3.3.1 Object graphCode.Edge.getData () [inline]

Return the data associated with this edge.

Returns

the data of this edge

2.3.3.2 Vertex graphCode.Edge.getFirstEndpoint () [inline]

Return the first endpoint of this edge.

Returns

the first endpoint of this edge

2.3.3.3 Object graphCode.Edge.getName () [inline]

Return the name associated with this edge.

Returns

the name of this edge

2.3.3.4 Vertex graphCode.Edge.getSecondEndpoint () [inline]

Return the second endpoint of this edge.

Returns

the second endpoint of this edge

2.3.3.5 void graphCode.Edge.setData (Object data) [inline]

Set the data associated with this edge.

Parameters

<i>data</i>	the data of this edge
-------------	-----------------------

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphCode/Edge.java

2.4 graphCode.GraphInput Class Reference

Static Public Member Functions

- static Hashtable [LoadSimpleGraph](#) ([SimpleGraph](#) newgraph)
- static Hashtable [LoadSimpleGraph](#) ([SimpleGraph](#) newgraph, String pathandfilename)
- static void [main](#) (String args[])

2.4.1 Detailed Description

A class that can read a graph (in a specific format) from a file.

Author

edhong

Version

0.0

2.4.2 Member Function Documentation

2.4.2.1 static Hashtable graphCode.GraphInput.LoadSimpleGraph ([SimpleGraph](#) newgraph) `[inline], [static]`

Load graph data from a text file via user interaction. This method asks the user for a directory and path name. It returns a hashtable of (String, [Vertex](#)) pairs. newgraph needs to already be initialized.

Parameters

<i>newgraph</i>	a simple graph
-----------------	----------------

Returns

a hash table of (String, [Vertex](#)) pairs

2.4.2.2 static Hashtable graphCode.GraphInput.LoadSimpleGraph ([SimpleGraph](#) newgraph, String pathandfilename) `[inline], [static]`

Load graph data from a text file. The format of the file is: Each line of the file contains 3 tokens, where the first two are strings representing vertex labels and the third is an edge weight (a double). Each line represents one edge.

This method returns a hashtable of (String, [Vertex](#)) pairs.

Parameters

<i>newgraph</i>	a graph to add edges to. newgraph should already be initialized
<i>pathandfilename</i>	the name of the file, including full path.

Returns

a hash table of (String, [Vertex](#)) pairs

2.4.2.3 static void graphCode.GraphInput.main (String args[]) [inline],[static]

Code to test the methods of this class.

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphCode/GraphInput.java

2.5 graphCode.KeyboardReader Class Reference

Static Public Member Functions

- static int [readInt](#) ()
- static double [readDouble](#) ()
- static String [readString](#) ()
- static void [main](#) (String[] args)

Static Public Attributes

- static final int [EOI_INT](#) = Integer.MAX_VALUE
- static final double [EOI_DOUBLE](#) = Double.MAX_VALUE
- static final String [EOI_STRING](#) = "END_OF_INFO_1234"
- static final int [ERROR_INT](#) = Integer.MIN_VALUE
- static final double [ERROR_DOUBLE](#) = Double.MIN_VALUE
- static final String [ERROR_STRING](#) = "I/O_ERROR_1234"
- static boolean [ERROR_MESSAGES](#) = true

2.5.1 Detailed Description

A class to read strings and numbers from the keyboard.

This class is intended for beginning Java programmers. It constructs the necessary buffered reader, converts strings to numbers, and handles exceptions.

The methods in this class are **static** methods. This means that they can be invoked using the class name, without needing to create a [KeyboardReader](#) object.

The following examples illustrate how to use this class to read from the keyboard:

```
int x = KeyboardReader.readInt();           // read an int
double y = KeyboardReader.readDouble();      // read a double
String s = KeyboardReader.readString();      // read a String
```

This class also defines return values to indicate end-of-information (EOI) and data conversion errors.

The default behavior of the [readInt\(\)](#) and [readDouble\(\)](#) methods is to write a comment to the screen in response to inappropriate data. This behavior may be controlled by the [ERROR_MESSAGES](#) field.

Author

Bill Conlen

2.5.2 Member Function Documentation

2.5.2.1 `static void graphCode.KeyboardReader.main (String[] args) [inline],[static]`

Tests the [KeyboardReader](#) methods.

This method would not normally be invoked by users of the [KeyboardReader](#) class.

2.5.2.2 `static double graphCode.KeyboardReader.readDouble () [inline],[static]`

Reads a line of input and converts it into a *double*.

Returns

the number that the user typed, or `EOI_DOUBLE` to indicate end-of-information, or `ERROR_DOUBLE` to indicate a conversion or I/O error.

2.5.2.3 `static int graphCode.KeyboardReader.readInt () [inline],[static]`

Reads a line of input and converts it into an *int*.

Returns

the integer that the user typed, or `EOI_INT` to indicate end-of-information, or `ERROR_INT` to indicate a conversion or I/O error.

2.5.2.4 `static String graphCode.KeyboardReader.readString () [inline],[static]`

Reads a line of character input.

Returns

the line of input that the user typed, or `EOI_STRING` to indicate end-of-information, or `ERROR_STRING` to indicate a conversion or I/O error.

2.5.3 Member Data Documentation

2.5.3.1 `final double graphCode.KeyboardReader.EOI_DOUBLE = Double.MAX_VALUE [static]`

Returned by the [readDouble\(\)](#) method to indicate EOI.

2.5.3.2 `final int graphCode.KeyboardReader.EOI_INT = Integer.MAX_VALUE [static]`

Returned by the [readInt\(\)](#) method to indicate EOI.

2.5.3.3 `final String graphCode.KeyboardReader.EOI_STRING = "END_OF_INFO_1234" [static]`

Returned by the [readString\(\)](#) method to indicate EOI.

2.5.3.4 `final double graphCode.KeyboardReader.ERROR_DOUBLE = Double.MIN_VALUE [static]`

Returned by the [readDouble\(\)](#) method to indicate an error.

2.5.3.5 `final int graphCode.KeyboardReader.ERROR_INT = Integer.MIN_VALUE [static]`

Returned by the [readInt\(\)](#) method to indicate an error.

2.5.3.6 `boolean graphCode.KeyboardReader.ERROR_MESSAGES = true [static]`

Controls the output of error messages to the console in response to inappropriate input.

2.5.3.7 `final String graphCode.KeyboardReader.ERROR_STRING = "I/O_ERROR_1234" [static]`

Returned by the [readString\(\)](#) method to indicate an error.

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphCode/KeyboardReader.java

2.6 networkflowstudy.logging Class Reference

Static Public Member Functions

- static void [printGraph](#) ([SimpleGraph](#) G)
- static void [printEdge](#) ([Edge](#) e)
- static void [printFlow](#) ([LinkedHashMap](#)< [Edge](#), Integer > flow)
- static void [printPath](#) ([List](#)< [Vertex](#) > path)

2.6.1 Detailed Description

Author

anisha

2.6.2 Member Function Documentation

2.6.2.1 `static void networkflowstudy.logging.printEdge (Edge e) [inline], [static]`

Given an edge e, print it's values

Parameters

<i>e</i>	An edge
----------	---------

2.6.2.2 `static void networkflowstudy.logging.printFlow (LinkedHashMap< Edge, Integer > flow)` `[inline]`,
`[static]`

Given an flow, print the value across all the edges

Parameters

<i>flow</i>	Hashmap of flow value for each edge in a graph
-------------	--

2.6.2.3 `static void networkflowstudy.logging.printGraph (SimpleGraph G)` `[inline]`,`[static]`

Given a graph G, print all the nodes and their incident edges

Parameters

<i>G</i>	Graph
----------	-------

2.6.2.4 `static void networkflowstudy.logging.printPath (List< Vertex > path)` `[inline]`,`[static]`

Print the vertices in a given path

Parameters

<i>path</i>	List of vertices inclusive of s and t
-------------	---------------------------------------

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/logging.java

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/MaxFlow.java

2.8 graphGenerationCode.Mesh.MeshGenerator Class Reference

Public Member Functions

- void [generate](#) ()
- [MeshGenerator](#) (String[] args)

Static Public Member Functions

- static void [main](#) (String[] args)

2.8.1 Detailed Description

This class generates text files representing mesh graph flow networks. The mesh graphs have edges from s to each node in the first column, from each internal node to the node on its right, both ways between every internal node and the nodes above and below it, and from the last column nodes to the sink.

The file format is the standard TCSS 343/543 format of 'first vertex' 'second vertex' 'capacity'. The program takes command line arguments. They are: [# of rows] [# of columns][capacity or maximum capacity][filename][-cc flag]. All arguments are optional. If you enter no arguments, you get a 3 x 4 mesh with capacity of 1 on all edges, printed to System.out. The arguments are:

#rows/columns - self-explanatory...defaults to 3x4 if no arguments are given

capacity - defaults to 1(fixed) if <3 arguments. Otherwise random on the range 1 to capacity, unless '-cc' set.

filename - the name of file to write to. Defaults to System.out if <4 parameters (or if -cc is last parameter)

-cc flag...With at least the first three parameters specified, ending the line with '-cc' will cause edge capacities to have a constant value of c.

Author

TCSS 543 group 2: Apaporn Boonyaratta, Richard Hill, Quang Lu, & David Thaler

Version

November 21, 2008

2.8.2 Constructor & Destructor Documentation

2.8.2.1 graphGenerationCode.Mesh.MeshGenerator.MeshGenerator (String[] args) [inline]

Constructor for mesh generator parses the command line arguments and sets the defaults. See the class comment for arguments/defaults.

GetMaxFlow*Returns*

the flow of the graph in a linked hash map of Edge to Integer.

- LinkedHashMap< [Edge](#), Integer > **GetMaxFlow** ()

2.9.1 Detailed Description**Author**

jason

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/PreflowPush.java

2.10 RandomGraph Class Reference**Static Public Member Functions**

- static void [main](#) (String[] args)
- static StringBuffer [graphBuilder](#) (int v, int e, int min, int max)

2.10.1 Member Function Documentation**2.10.1.1 static StringBuffer RandomGraph.graphBuilder (int v, int e, int min, int max)** `[inline],[static]`

This method creates a 3 token representation of a graph.

Parameters

<i>v</i>	The number of vertices in the graph
<i>e</i>	The number of edges leaving each vertice
<i>min</i>	The lowerbound on the capacity value of each edge
<i>max</i>	The upperbound on the capacity value of each edge

Returns

A string buffer, each line contains 3 tokens corresponding to a directed edge: the tail, the head, and the capacity.

2.10.1.2 static void RandomGraph.main (String[] args) `[inline],[static]`

Entrance point for the program. java [RandomGraph](#) v, e, m, f

Parameters

<i>v</i>	- the number of vertices
<i>e</i>	- the number of edges leaving each node
<i>min</i>	- the lower bound on the edge capacities
<i>max</i>	- the upper bound on the edge capacities
<i>f</i>	- path and file name for saving the graph

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphGenerationCode/FixedDegree/RandomGraph.java

2.11 networkflowstudy.SaveOutput Class Reference

Static Public Member Functions

- static void **writeToCSV** (String graphType, String algorithmName, int numberOfVertices, long runningTime, int maxFlow) throws IOException
- static void **writeToCSV** (String graphType, String algorithmName, int numberOfVertices, long runningTime, int maxFlow, int capacity) throws IOException

2.11.1 Detailed Description

Author

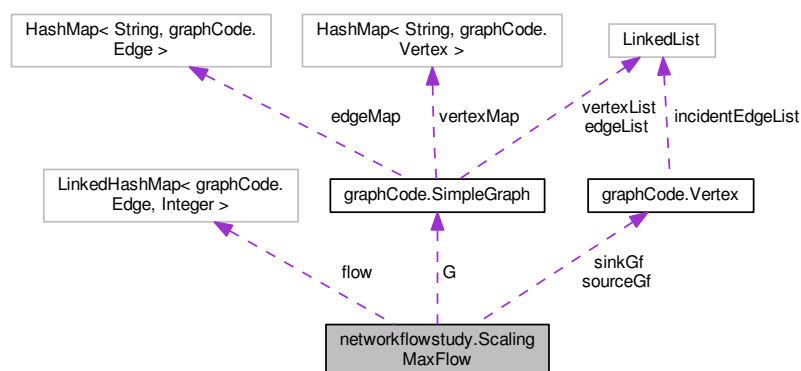
anisha

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/SaveOutput.java

2.12 networkflowstudy.ScalingMaxFlow Class Reference

Collaboration diagram for networkflowstudy.ScalingMaxFlow:



Public Member Functions

- [ScalingMaxFlow](#) ([SimpleGraph](#) G)
- [LinkedHashMap](#)<[Edge](#), [Integer](#)> [calculateFlow](#) ([Vertex](#) sourceG, [Vertex](#) sinkG)
- [int](#) [getDelta](#) ([Vertex](#) source)

2.12.1 Detailed Description

Calculates and returns flow using Scaling max Flow algorithm

Author

anisha

2.12.2 Constructor & Destructor Documentation

2.12.2.1 `networkflowstudy.ScalingMaxFlow.ScalingMaxFlow (SimpleGraph G)` `[inline]`

Constructor to initialize the flow along all edges and create the corresponding graph

Parameters

<i>G</i>	
----------	--

2.12.3 Member Function Documentation

2.12.3.1 `LinkedHashMap<Edge, Integer> networkflowstudy.ScalingMaxFlow.calculateFlow (Vertex sourceG, Vertex sinkG)` `[inline]`

Calculate flow of a network using Scaling max- flow algorithm

Parameters

<i>source</i>	vertex sourceG
<i>sink</i>	vertex sinkG

Returns

flow

2.12.3.2 `int networkflowstudy.ScalingMaxFlow.getDelta (Vertex source)` `[inline]`

calculate delta for a residual graph G

Parameters

<i>source</i>	vertex in residual graph
---------------	--------------------------

Returns

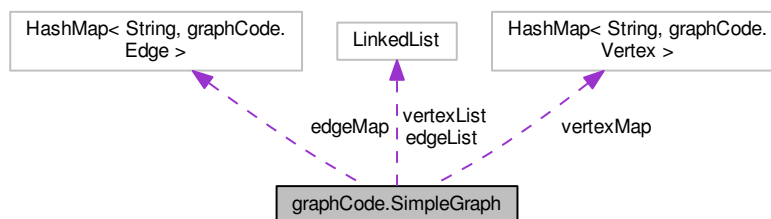
delta

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/ScalingMaxFlow.java

2.13 graphCode.SimpleGraph Class Reference

Collaboration diagram for graphCode.SimpleGraph:



Public Member Functions

- Iterator `vertices` ()
- Iterator `edges` ()
- HashMap `getVertexMap` ()
- HashMap `getEdgeMap` ()
- Iterator `incidentEdges` (Vertex v)
- Vertex `aVertex` ()
- Vertex `insertVertex` (Object data, Object name)
- Edge `insertEdge` (Vertex v, Vertex w, Object data, Object name)
- Vertex `opposite` (Vertex v, Edge e)
- Vertex `tail` (Vertex v, Edge e)
- int `numVertices` ()
- int `numEdges` ()
- Vertex `getVertex` (String vertexName)

Static Public Member Functions

- static void `main` (String[] args)

2.13.1 Detailed Description

A class that represents a graph.

Author

edhong

Version

0.0

2.13.2 Member Function Documentation

2.13.2.1 Vertex `graphCode.SimpleGraph.aVertex ()` [inline]

Return an arbitrary vertex of this graph

Returns

some vertex of this graph

2.13.2.2 Iterator `graphCode.SimpleGraph.edges ()` [inline]

Return the edge list of this graph.

Returns

edge list of this graph

2.13.2.3 HashMap `graphCode.SimpleGraph.getEdgeMap ()` [inline]

get graph edge hashMap

Returns

edgeMap

2.13.2.4 Vertex `graphCode.SimpleGraph.getVertex (String vertexName)` [inline]

Returns the vertex in a given [SimpleGraph](#) object

Parameters

<i>vertexName</i>	
-------------------	--

Returns

source vertex

2.13.2.5 `HashMap graphCode.SimpleGraph.getVertexMap ()` `[inline]`

get graph vertex hashMap

Returns

vertexMap

2.13.2.6 `Iterator graphCode.SimpleGraph.incidentEdges (Vertex v)` `[inline]`

Given a vertex, return an iterator to the edge list of that vertex

Parameters

<i>v</i>	a vertex
----------	----------

Returns

an iterator to the edge list of that vertex

2.13.2.7 `Edge graphCode.SimpleGraph.insertEdge (Vertex v, Vertex w, Object data, Object name)` `[inline]`

Add an edge to this graph.

Parameters

<i>v</i>	the first endpoint of the edge
<i>w</i>	the second endpoint of the edge
<i>data</i>	data to be associated with the new edge
<i>name</i>	name to be associated with the new edge

Returns

the new edge

2.13.2.8 `Vertex graphCode.SimpleGraph.insertVertex (Object data, Object name)` `[inline]`

Add a vertex to this graph.

Parameters

<i>data</i>	an object to be associated with the new vertex
<i>name</i>	a name to be associated with the new vertex

Returns

the new vertex

2.13.2.9 `static void graphCode.SimpleGraph.main (String[] args)` `[inline],[static]`

Code to test the correctness of the [SimpleGraph](#) methods.

2.13.2.10 `int graphCode.SimpleGraph.numEdges ()` `[inline]`

Return the number of edges in this graph.

Returns

the number of edges

2.13.2.11 `int graphCode.SimpleGraph.numVertices ()` `[inline]`

Return the number of vertices in this graph.

Returns

the number of vertices

2.13.2.12 `Vertex graphCode.SimpleGraph.opposite (Vertex v, Edge e)` `[inline]`

Given a vertex and an edge, if the vertex is one of the endpoints of the edge, return the other endpoint of the edge. Otherwise, return null.

Parameters

<i>v</i>	a vertex
<i>e</i>	an edge

Returns

the other endpoint of the edge (or null, if v is not an endpoint of e)

2.13.2.13 Vertex graphCode.SimpleGraph.tail (Vertex *v*, Edge *e*) [inline]

Given a vertex and an edge, if the vertex is the head of the edge, return the other endpoint (tail) of the edge. Otherwise, return null.

Parameters

<i>v</i>	a vertex
<i>e</i>	an edge

Returns

the other endpoint of the edge (or null, if *v* is not an endpoint of *e*)

2.13.2.14 Iterator graphCode.SimpleGraph.vertices () [inline]

Return the vertex list of this graph.

Returns

vertex list of this graph

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphCode/SimpleGraph.java

2.14 networkflowstudy.tc543 Class Reference**Static Public Member Functions**

- static void [main](#) (String[] args) throws IOException

2.14.1 Detailed Description**Author**

anisha

2.14.2 Member Function Documentation**2.14.2.1** static void networkflowstudy.tc543.main (String[] args) throws IOException [inline],[static]**Parameters**

<i>args</i>	the command line arguments
-------------	----------------------------

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/tcss543.java

2.15 networkflowstudy.utils Class Reference

Static Public Member Functions

- static LinkedHashMap< [Edge](#), Integer > [initFlow](#) ([SimpleGraph](#) G)
- static [SimpleGraph](#) [createResidualGraph](#) ([SimpleGraph](#) G, LinkedHashMap< [Edge](#), Integer > flow)
- static List< [Vertex](#) > [getSTPath](#) ([SimpleGraph](#) Gf, [Vertex](#) Sink, [Vertex](#) Source)
- static List< [Vertex](#) > [getSTPath](#) ([SimpleGraph](#) Gf, [Vertex](#) sink, [Vertex](#) source, int delta)
- static int [get_bottleneck](#) ([SimpleGraph](#) Gf, List< [Vertex](#) > st_path)
- static void [augment](#) ([SimpleGraph](#) G, [SimpleGraph](#) Gf, LinkedHashMap< [Edge](#), Integer > flow, List< [Vertex](#) > path)

2.15.1 Detailed Description

Author

anisha

2.15.2 Member Function Documentation

2.15.2.1 static void networkflowstudy.utils.augment ([SimpleGraph](#) G, [SimpleGraph](#) Gf, LinkedHashMap< [Edge](#), Integer > flow, List< [Vertex](#) > path) [inline],[static]

Calculates the increase in flow using [get_bottleneck\(\)](#) and updates the flow LinkedHashMap

Parameters

<i>G</i>	
<i>Gf</i>	
<i>flow</i>	
<i>path</i>	

2.15.2.2 static [SimpleGraph](#) networkflowstudy.utils.createResidualGraph ([SimpleGraph](#) G, LinkedHashMap< [Edge](#), Integer > flow) [inline],[static]

Create a residual graph based on a network flow $G=(V,E)$

Parameters

<i>G</i>	a simple graph G
<i>flow</i>	flow across the edges

Returns

Residual Graph Gf

2.15.2.3 `static int networkflowstudy.utils.get_bottleneck (SimpleGraph Gf, List< Vertex > st_path)` `[inline]`,
`[static]`

Returns bottleneck for the given s-t path of graph Gf

Parameters

<i>Gf</i>	
<i>st_path</i>	

Returns

b_neck

2.15.2.4 `static List<Vertex> networkflowstudy.utils.getSTPath (SimpleGraph Gf, Vertex Sink, Vertex Source)`
`[inline]`,`[static]`

Returns s-t path of graph Gf, if there is no simple path , returns 0

Parameters

<i>Gf</i>	
<i>Source</i>	
<i>Sink</i>	

Returns

null if no s-t path else valid s-t path

2.15.2.5 `static List<Vertex> networkflowstudy.utils.getSTPath (SimpleGraph Gf, Vertex sink, Vertex source, int delta)` `[inline]`,`[static]`

Return s-t path for a graph Gf based on the limiting capacity delta

Parameters

<i>Gf</i>	
<i>sink</i>	
<i>source</i>	
<i>delta</i>	

Returns

2.15.2.6 `static LinkedHashMap<Edge, Integer> networkflowstudy.utils.initFlow (SimpleGraph G)` `[inline]`,
`[static]`

Initialize the flow LinkedHashMap for the graph G

Parameters

G	simple graph G
---	----------------

Returns

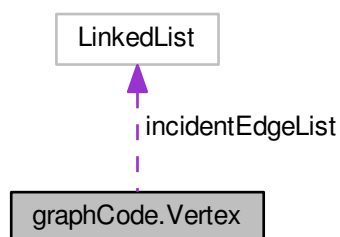
LinkedHashMap containing <Edge, flow value> pairs

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/networkflowstudy/utils.java

2.16 graphCode.Vertex Class Reference

Collaboration diagram for graphCode.Vertex:



Public Member Functions

- [Vertex](#) (Object data, Object name)
- Object [getName](#) ()
- Object [getData](#) ()
- void [setData](#) (Object data)

2.16.1 Detailed Description

Class that represents a vertex in a graph. A name (usually a string, but it can be an arbitrary object) can be associated with the vertex.

Data (also represented by an object (e.g., a string)) can also be associated with a vertex. This could be useful, for example, if you need to mark a vertex as being visited in some graph traversal.

Author

edhong

Version

0.0

2.16.2 Constructor & Destructor Documentation

2.16.2.1 graphCode.Vertex.Vertex (Object *data*, Object *name*) [inline]

Constructor that allows data and a name to be associated with the vertex.

Parameters

<i>data</i>	an object to be associated with this vertex
<i>name</i>	a name to be associated with this vertex

2.16.3 Member Function Documentation

2.16.3.1 Object graphCode.Vertex.getData () [inline]

Return the data associated with this vertex.

Returns

the data of this vertex

2.16.3.2 Object graphCode.Vertex.getName () [inline]

Return the name associated with this vertex.

Returns

the name of this vertex

2.16.3.3 void graphCode.Vertex.setData (Object *data*) [inline]

Set the data associated with this vertex.

Parameters

<i>data</i>	the data of this vertex
-------------	-------------------------

The documentation for this class was generated from the following file:

- /home/anisha/AlgoProject/NetworkFlowStudy/src/graphCode/Vertex.java

Index

- aVertex
 - graphCode::SimpleGraph, 18
- augment
 - networkflowstudy::utils, 22
- BipartiteGraph, 3
- calculateFlow
 - networkflowstudy::MaxFlow, 11
 - networkflowstudy::ScalingMaxFlow, 16
- createResidualGraph
 - networkflowstudy::utils, 22
- EOI_DOUBLE
 - graphCode::KeyboardReader, 8
- EOI_INT
 - graphCode::KeyboardReader, 8
- EOI_STRING
 - graphCode::KeyboardReader, 8
- ERROR_DOUBLE
 - graphCode::KeyboardReader, 9
- ERROR_INT
 - graphCode::KeyboardReader, 9
- ERROR_MESSAGES
 - graphCode::KeyboardReader, 9
- ERROR_STRING
 - graphCode::KeyboardReader, 9
- Edge
 - graphCode::Edge, 4
- edges
 - graphCode::SimpleGraph, 18
- generate
 - graphGenerationCode::Mesh::MeshGenerator, 13
- get_bottleneck
 - networkflowstudy::utils, 23
- getData
 - graphCode::Edge, 4
 - graphCode::Vertex, 25
- getDelta
 - networkflowstudy::ScalingMaxFlow, 16
- getEdgeMap
 - graphCode::SimpleGraph, 18
- getFirstEndpoint
 - graphCode::Edge, 5
- getName
 - graphCode::Edge, 5
 - graphCode::Vertex, 25
- getSTPath
 - networkflowstudy::utils, 23
- getSecondEndpoint
 - graphCode::Edge, 5
- getVertex
 - graphCode::SimpleGraph, 18
- getVertexMap
 - graphCode::SimpleGraph, 19
- graphBuilder
 - RandomGraph, 14
- graphCode.Edge, 4
- graphCode.GraphInput, 6
- graphCode.KeyboardReader, 7
- graphCode.SimpleGraph, 17
- graphCode.Vertex, 24
- graphCode::Edge
 - Edge, 4
 - getData, 4
 - getFirstEndpoint, 5
 - getName, 5
 - getSecondEndpoint, 5
 - setData, 5
- graphCode::GraphInput
 - LoadSimpleGraph, 6
 - main, 7
- graphCode::KeyboardReader
 - EOI_DOUBLE, 8
 - EOI_INT, 8
 - EOI_STRING, 8
 - ERROR_DOUBLE, 9
 - ERROR_INT, 9
 - ERROR_MESSAGES, 9
 - ERROR_STRING, 9
 - main, 8
 - readDouble, 8
 - readInt, 8
 - readString, 8
- graphCode::SimpleGraph
 - aVertex, 18
 - edges, 18
 - getEdgeMap, 18
 - getVertex, 18
 - getVertexMap, 19
 - incidentEdges, 19
 - insertEdge, 19
 - insertVertex, 19
 - main, 20
 - numEdges, 20
 - numVertices, 20
 - opposite, 20
 - tail, 20

- vertices, 21
- graphCode::Vertex
 - getData, 25
 - getName, 25
 - setData, 25
 - Vertex, 25
- graphGenerationCode.Mesh.MeshGenerator, 12
- graphGenerationCode.Random.BuildGraph, 3
- graphGenerationCode::Mesh::MeshGenerator
 - generate, 13
 - main, 13
 - MeshGenerator, 12
- graphGenerationCode::Random::BuildGraph
 - main, 3
- incidentEdges
 - graphCode::SimpleGraph, 19
- initFlow
 - networkflowstudy::utils, 24
- insertEdge
 - graphCode::SimpleGraph, 19
- insertVertex
 - graphCode::SimpleGraph, 19
- LoadSimpleGraph
 - graphCode::GraphInput, 6
- main
 - graphCode::GraphInput, 7
 - graphCode::KeyboardReader, 8
 - graphCode::SimpleGraph, 20
 - graphGenerationCode::Mesh::MeshGenerator, 13
 - graphGenerationCode::Random::BuildGraph, 3
 - networkflowstudy::tcss543, 21
 - RandomGraph, 14
- MeshGenerator
 - graphGenerationCode::Mesh::MeshGenerator, 12
- networkflowstudy.logging, 9
- networkflowstudy.MaxFlow, 11
- networkflowstudy.PreflowPush, 13
- networkflowstudy.SaveOutput, 15
- networkflowstudy.ScalingMaxFlow, 15
- networkflowstudy.tcss543, 21
- networkflowstudy.utils, 22
- networkflowstudy::MaxFlow
 - calculateFlow, 11
- networkflowstudy::ScalingMaxFlow
 - calculateFlow, 16
 - getDelta, 16
 - ScalingMaxFlow, 16
- networkflowstudy::logging
 - printEdge, 9
 - printFlow, 10
 - printGraph, 10
 - printPath, 10
- networkflowstudy::tcss543
 - main, 21
- networkflowstudy::utils
 - augment, 22
 - createResidualGraph, 22
 - get_bottleneck, 23
 - getSTPath, 23
 - initFlow, 24
- numEdges
 - graphCode::SimpleGraph, 20
- numVertices
 - graphCode::SimpleGraph, 20
- opposite
 - graphCode::SimpleGraph, 20
- printEdge
 - networkflowstudy::logging, 9
- printFlow
 - networkflowstudy::logging, 10
- printGraph
 - networkflowstudy::logging, 10
- printPath
 - networkflowstudy::logging, 10
- RandomGraph, 14
 - graphBuilder, 14
 - main, 14
- readDouble
 - graphCode::KeyboardReader, 8
- readInt
 - graphCode::KeyboardReader, 8
- readString
 - graphCode::KeyboardReader, 8
- ScalingMaxFlow
 - networkflowstudy::ScalingMaxFlow, 16
- setData
 - graphCode::Edge, 5
 - graphCode::Vertex, 25
- tail
 - graphCode::SimpleGraph, 20
- Vertex
 - graphCode::Vertex, 25
- vertices
 - graphCode::SimpleGraph, 21