Package ‘giRaph’

February 19, 2015

Title  The giRaph package for graph representation in R
Namespace giRaph
Description  Supply classes and methods to represent and manipulate graphs
Version 0.1.2
Date 2012/02/13
Author Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca
Maintainer Claus Dethlefsen <aas.claus.dethlefsen@gmail.com>
Depends R (>= 2.4.0), graphics, methods
Suggests mathgraph, dynamicGraph
License GPL (>= 2)
URL http://r.kvfc.dk/giRaph
Repository CRAN
Date/Publication 2013-12-11 09:00:29
NeedsCompilation no

R topics documented:

adjacencyList .................................................. 2
adjacencyList-class ........................................... 3
adjacencyMatrix .................................................. 5
adjacencyMatrix-class ........................................ 6
anyGraph-class .................................................. 7
areTheSame ...................................................... 9
card ............................................................ 10
directedEdge-class .............................................. 10
display .......................................................... 12
dynamic.Graph .................................................. 13
dynamic.digraph ............................................... 14
dynamicGraph .................................................. 15
dynamicGraph-class .......................................... 16
adjacencyList

Adjacency list representation of a graph

Description
Retrieve or set the adjacency list representation of a graph.

Usage
adjacencyList(object, …)
adjacencyList(x, force = TRUE) <- value

Arguments
object a graph object from which the representation should be retrieved
… additional parameters to be used when retrieving the representation
x a graph object in which the representation should be set
force a logical value telling whether the representation should be set even if this
amounts to changing the graph
value an object of class "adjacencyList" containing the representation to be set

Details
The functions adjacencyList and adjacencyList<- are generic.

Value
The function adjacencyList returns an object of class "adjacencyList" containing the adjacency
list representation to be retrieved. The function adjacencyList<- returns a graph object in which
the adjacency list representation has been set.
Description

A class for adjacency list representation of multi-graphs

Objects from the Class

Objects can be created by calls of the form `new("adjacencyList", id, pa, ne).

Slots

.Data: Object of class "list": each element represents a vertex and is in turn a list of (at most) three elements, namely pa, ne and ch storing, respectively, the numeric identifiers of parents, neighbours and children

Extends

Class "list", from data part. Class "vector", by class "list".

Methods

initialize signature(.Object = "adjacencyList"): constructs an adjacency list representation of a multi-graph from a vertex set id, a list pa of parent numeric identifiers, a list ch of children numeric identifiers, and a list ne of neighbour numeric identifiers

show signature(object = "adjacencyList"): displays an adjacency list representation

names signature(x = "adjacencyList"): gets the character vertex identifiers of an adjacency list

names<- signature(x = "adjacencyList"): sets the character vertex identifiers of an adjacency list

card signature(object = "adjacencyList"): returns the number of vertices and the total number of edge occurrences in an adjacency list

isEmpty signature(object = "adjacencyList"): an adjacency list is empty if it has no entries

isPresent signature(el = "undirectedEdge", ou = "adjacencyList"): tells whether an undirected edge occurs in the multi-graph represented by an adjacency list

isPresent signature(el = "directedEdge", ou = "adjacencyList"): tells whether a directed edge occurs in the multi-graph represented by an adjacency list
areTheSame signature(x = "adjacencyList", y = "adjacencyList"): x and y are the same adjacency list if they represent the same multi-graph

signature(x = "adjacencyList"): extracts the adjacency list of an induced subgraph

signature(x = "adjacencyList"): extracts the character identifier of a vertex

coerce signature(from = "incidenceList", to = "adjacencyList"): converts an incidence list to an adjacency list by dropping all but ordinary directed and undirected edges

coerce signature(from = "incidenceMatrix", to = "adjacencyList"): converts an incidence matrix to an adjacency matrix by dropping hyper-edges

coerce signature(from = "adjacencyMatrix", to = "adjacencyList"): converts an adjacency matrix to an adjacency list

+ signature(e1 = "adjacencyList", e2 = "vertexSet"): adds a vertex set to an adjacency list by making the new vertices isolated

- signature(e1 = "adjacencyList", e2 = "vertexSet"): removes a vertex set from an adjacency list by dropping all edges involving the vertex set

+ signature(e1 = "adjacencyList", e2 = "undirectedEdge"): adds an ordinary undirected edge to an adjacency list

+ signature(e1 = "adjacencyList", e2 = "directedEdge"): adds an ordinary directed edge to an adjacency list

- signature(e1 = "adjacencyList", e2 = "undirectedEdge"): removes an undirected edge from an adjacency list

- signature(e1 = "adjacencyList", e2 = "directedEdge"): removes a directed edge from an adjacency list

* signature(e1 = "adjacencyMatrix", e2 = "vertexSet"): restricts an adjacency matrix to a vertex set by dropping all edges involving vertices outside the vertex set

Warning

The pa, ch and ne constructor input lists are silently discarded if their length differs from the actual number of vertices (determined by id). All input numeric identifiers greater than this number are also silently discarded by the constructor. The id input to constructor is mandatory, since the constructor needs to identify vertices. In addition, at least one between pa and ch should be present, if directed edges have to be specified, and ne should be present, if undirected edges have to be specified. If both pa and ch are present, the union of the two corresponding multi-sets of edges is specified. If the neighbourhood relationship specified by ne is not symmetric, it is made symmetric by the constructor.

Note

The names<- replacement method works only if the names to be assigned can be used to construct a vertexSet object having the right cardinality, otherwise the names are left unchanged and a warning message is given.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca
adjacencyMatrix

See Also

adjacencyList and multiGraph-class

adjacencyMatrix Adjacency matrix representation of a graph

Description

Retrieve or set the adjacency matrix representation of a graph.

Usage

adjacencyMatrix(object, ...)  
adjacencyMatrix(x, force = TRUE) <- value

Arguments

object a graph object from which the representation should be retrieved  
... additional parameters to be used when retrieving the representation  
x a graph object in which the representation should be set  
force a logical value telling whether the representation should be set even if this amounts to changing the graph  
value an object of class "adjacencyMatrix" containing the representation to be set

Details

The functions adjacencyMatrix and adjacencyMatrix<- are generic.

Value

The function adjacencyMatrix returns an object of class "adjacencyMatrix" containing the adjacency matrix representation to be retrieved. The function adjacencyMatrix<- returns a graph object in which the adjacency matrix representation has been set.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

adjacencyMatrix-class and simpleGraph-class
adjacencyMatrix-class  Class "adjacencyMatrix"

Description
A class for adjacency matrix representation of simple-graphs

Objects from the Class
Objects can be created by calls of the form new("adjacencyMatrix",X).

Slots
.Data: Object of class "matrix"; standard 0-1 coding for ordinary directed and undirected edges

Extends
Class "matrix", from data part. Class "structure", by class "matrix". Class "array", by class "matrix". Class "vector", by class "matrix", with explicit coerce.

Methods
initialize signature(.Object = "adjacencyMatrix"): constructs an adjacency matrix representation of a simple-graph from a 0-1 matrix
show signature(object = "adjacencyMatrix"): displays an adjacency matrix representation
names signature(x = "adjacencyMatrix"): gets the character vertex identifiers of an adjacency matrix
names<- signature(x = "adjacencyMatrix"): sets the character vertex identifiers of an adjacency matrix
card signature(object = "adjacencyMatrix"): returns the number of vertices and the total number of edges (directed and undirected) in an incidence matrix
isEmpty signature(object = "adjacencyMatrix"): an adjacency matrix is empty if it has no entries
isPresent signature(el = "undirectedEdge", ou = "adjacencyMatrix"): tells whether an undirected edge occurs in the graph represented by an adjacency matrix
isPresent signature(el = "directedEdge", ou = "adjacencyMatrix"): tells whether a directed edge occurs in the graph represented by an adjacency matrix
areTheSame signature(x = "adjacencyMatrix", y = "adjacencyMatrix"): x and y are the same adjacency matrix if they represent the same simple-graph
[] signature(x = "adjacencyMatrix"): extracts the adjacency matrix of an induced subgraph
[[ signature(x = "adjacencyMatrix"): extracts the character identifier of a vertex
coerce signature(from = "incidencelist", to = "adjacencyMatrix"): converts an incidence list to an adjacency matrix by keeping ordinary directed and undirected edges and dropping loops and parallel edges


coerce signature(from = "incidenceMatrix", to = "adjacencyMatrix"): converts an incidence matrix to an adjacency matrix by dropping hyper-edges, loops and parallel edges

coerce signature(from = "adjacencyList", to = "adjacencyMatrix"): converts an adjacency list to an adjacency matrix by dropping loops and parallel edges

+ signature(e1 = "adjacencyMatrix", e2 = "vertexSet"): adds a vertex set to an adjacency matrix by making the new vertices isolated

- signature(e1 = "adjacencyMatrix", e2 = "vertexSet"): removes a vertex set from an adjacency matrix by dropping all edges involving the vertex set

+ signature(e1 = "adjacencyMatrix", e2 = "undirectedEdge"): adds an ordinary undirected edge (not a loop) to an adjacency matrix

+ signature(e1 = "adjacencyMatrix", e2 = "directedEdge"): adds an ordinary directed edge to an adjacency matrix

- signature(e1 = "adjacencyMatrix", e2 = "undirectedEdge"): removes an undirected edge from an adjacency matrix

- signature(e1 = "adjacencyMatrix", e2 = "directedEdge"): removes a directed edge from an adjacency matrix

* signature(e1 = "adjacencyMatrix", e2 = "vertexSet"): restricts an adjacency matrix to a vertex set by dropping all edges involving vertices outside the vertex set

Warning

All input non-zero diagonal entries are silently changed to zero by the constructor.

Note

The names<- replacement method works only if the names to be assigned can be used to construct a vertexSet object having the right cardinality, otherwise the names are left unchanged and a warning message is given.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

adjacencyMatrix and simpleGraph-class

anyGraph-class  Class "anyGraph"

Description

A class for graphs of any kind.
Objects from the Class

Objects can be created by calls of the form new("anyGraph", ...). An anyGraph object consists of a single slot (incidenceList) for the only possible representation.

Slots

incidenceList: Object of class "incidenceList"

Methods

initialize signature(.Object="anyGraph"): constructs any graph from incidence list representation

show signature(object = "anyGraph"): displays the available representations of any graph

display signature(x = "anyGraph"): static graphical representation via package 'mathgraph'

dynamic.Graph signature(object = "anyGraph"): dynamic graphical representation via package 'dynamicGraph'

incidenceList<- signature(x = "anyGraph"): sets the incidence list representation

incidenceList signature(object = "anyGraph"): gets the incidence list representation

names signature(x = "anyGraph"): gets the character vertex identifiers of any graph

names<- signature(x = "anyGraph"): sets the character vertex identifiers of any graph

card signature(object = "anyGraph"): returns the number of vertices and the total number of edge occurrences in any graph

isEmpty signature(object = "anyGraph"): a graph object is empty if all its possible representations are empty

isPresent signature(el = "edge", ou = "anyGraph"): an edge occurs in a graph object if it occurs in its non-empty slots

areTheSame signature(x = "anyGraph", y = "anyGraph"): x and y are the same if their non-empty slots represent the same graph

[ signature(x = "anyGraph"): extracts an induced subgraph

[[ signature(x = "anyGraph"): extracts the character identifier of a vertex

coerce signature(from = "generalGraph", to = "anyGraph"): no edges are lost in the conversion to the most general class of graphs

coerce signature(from = "multiGraph", to = "anyGraph"): no edges are lost in the conversion to the most general class of graphs

coerce signature(from = "simpleGraph", to = "anyGraph"): no edges are lost in the conversion to the most general class of graphs

coerce signature(from = "anyGraph", to = "dg.graph"): conversion to class 'dg.graph' of package 'dynamicGraph'

+ signature(e1 = "anyGraph", e2 = "vertexSet"): adds a vertex set to any graph by making the new vertices isolated

- signature(e1 = "anyGraph", e2 = "vertexSet"): removes a vertex set from any graph by dropping all edges involving the vertex set
+ signature(e1 = "anyGraph", e2 = "edge"): adds an edge to any graph
- signature(e1 = "anyGraph", e2 = "edge"): removes an edge from any graph
* signature(e1 = "anyGraph", e2 = "vertexSet"): restricts any graph to a vertex set by dropping all edges involving vertices outside the vertex set

**Note**

Graphical representation via package 'dynamicGraph' is based on coercion to class `dgNgraph`. Coercion to class `dgNgraph` is implemented via coercion to class `generalGraph`. Graphical representation via package 'mathgraph' is obtained by means of coercion to class `simpleGraph`.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

**See Also**

`incidenceList-class` and `incidenceList`

---

**Description**

Check whether two objects are the same mathematical entity.

**Usage**

`areTheSame(x, y)`

**Arguments**

- `x` an R object representing a mathematical entity
- `y` another R object possibly representing the same mathematical entity

**Details**

The function `areTheSame` is generic.

**Value**

Returns a logical value telling whether the two objects are the same mathematical entity or not.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca
card \hspace{1cm} \textit{Get the cardinality of an object}

**Description**

Gets the cardinality of an object.

**Usage**

\texttt{card(object, ...)}

**Arguments**

- \texttt{object} an R object whose cardinality is to be retrieved
- \texttt{...} additional parameters to be used when retrieving cardinality

**Details**

The function \texttt{card} is generic.

**Value**

Returns a numeric value corresponding to the cardinality of \texttt{object}.

**Note**

For vectors cardinality is the same as length.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

---

**directedEdge-class** \hspace{1cm} \textit{Class "directedEdge"}

**Description**

Class for directed edges

**Objects from the Class**

Objects can be created by calls of the form \texttt{new("directedEdge", ...)} which admit short-hands of the form \texttt{d(...)} and \texttt{r(...)}.
Slots

.Data: Object of class "list" storing strictly positive numbers that refer to a given "vertexSet" object

Extends

Class "edge", directly. Class "list", from data part. Class "vector", by class "list".

Methods

initialize signature(.Object = "directedEdge"): constructs a directed edge from a list of strictly positive integers

show signature(object = "directedEdge"): displays a directed edge as an ordered sequence of undirected edges joined by arrows (using numeric codes)

showRel signature(object = "directedEdge", code="vertexSet"): displays a directed edge as an ordered sequence of undirected edges joined by arrows (using character names

areTheSame signature(x = "directedEdge", y = "directedEdge"): x and y are the same directed edge if they are the same ordered sequence of undirected edges

[ signature(x = "directedEdge"): extracts a directed edge

[[' signature(x = "directedEdge"): extracts an undirected edge

card signature(object= "directedEdge"): counts all vertices in a directed edge

coerce signature(from = "vector", to = "directedEdge"): constructs a directed edge from vector input

coerce signature(from = "undirectedEdge", to = "directedEdge"): directs an undirected edge

maxId signature(x="directedEdge"): gets the maximum numeric identifier of a directed edge

recode signature(object="directedEdge", src = "vertexSet", dst = "vertexSet"): recodes a directed edge by making its numbers refer to another "vertexSet" object

Warning

The constructor will try to handle any vector input by silently transforming it into a list of strictly positive integers.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

directedEdge-class, undirectedEdge-class, edgelist-class, d and r.
Make a display of the graph using the graphics window

Description

Uses the package mathgraph to create a simple display of a simple graph

Usage

display(x, ...)

Arguments

x

an object of class simpleGraph-class.

...

further arguments passed to plot.mathgraph.

Value

A display in the graphics window.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

simpleGraph-class, plot.mathgraph

Examples

G <- new("incidenceList",
  V = letters[1:12],
  E = list(
    d(6,5), c(2,4), c(1,3)),
    u(2,4,5),
    d(2,4), d(4,2),
    d(1,7), d(3,7), d(4,7),
    d(5,8), d(5,8), d(5,8),
    u(6,9), d(6,9),
    u(9,9),
    d(9,8), d(9,12),
    u(7,8), u(8,12), u(12,11), u(11,7),
    u(11,8),
    d(11,10)
  )
)

sg <- new("simpleGraph", adjacencyMatrix=as(G, "adjacencyMatrix"))
display(sg)
gg <- new("generalGraph", incidenceList=G)
display(gg)

**dynamic.Graph**

*DynamicGraph display of simple graph*

**Description**

Uses the package **dynamicGraph** to create an advanced, interactive display of a simple graph.

**Usage**

```r
dynamicGraph(object, ...)
```

**Arguments**

- `object` an object of class `simpleGraph-class`.
- `...` further arguments passed to `DynamicGraph`.

**Value**

A `dynamicGraph` window is open in which the graph can be inspected interactively.

**Note**

This interface is still very experimental.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

**See Also**

`simpleGraph-class`, `plot.mathgraph`

**Examples**

```r
G <- new("incidenceList",
  V = letters[1:12],
  E = list(
    d(6,5),c(2,4),c(1,3)),
    u(2,4,5),
    d(2,4),d(4,2),
    d(1,7),d(3,7),d(4,7),
    d(5,8),d(5,8),d(5,8),
    u(6,9),d(6,9),
    u(9,9),
    d(9,8),d(9,12),
```
edge-class

virtual class

Description

Virtual Class for all edges

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

areTheSame signature(x = "edge", y = "edge"): always returns FALSE, implementing the idea that two edges of different kind are never the same

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

undirectedEdge-class, directedEdge-class and edgeList-class.
**Description**

Class for multi-sets of edges

**Objects from the Class**

Objects can be created by calls of the form `new("edgelist", ...)`.

**Slots**

`.Data`: Object of class "list" whose elements are of class "edge"

**Extends**

Class "list", from data part. Class "vector", by class "list".

**Methods**

- **initialize** signature(.Object = "edgelist"): constructs a multi-set of edges from a list of edges
- **show** signature(object = "edgelist"): displays a multi-set of edges in graph brackets (numeric codes)
- **showRel** signature(object = "edgelist", code="vertexSet"): displays a multi-set of edges in graph brackets (character names)
- **areTheSame** signature(x = "edgelist", y = "edgelist"): x and y are the same multi-set of edges if they contain the same edges with the same multiplicity
- **isPresent** signature(x = "edge", ou = "edgelist"): tells whether an edge belongs to a multi-set of edges
- `[]` signature(x = "edgelist"): extracts a multi-set of edges
- `+` signature(e1 = "edgeList", e2 = "edge"): adds an (occurrence of an) edge to a multi-set of edges
- `+` signature(e1 = "edge", e2 = "edgeList"): adds an (occurrence of an) edge to a multi-set of edges
- `-` signature(e1 = "edgeList", e2 = "edge"): drops (the first occurrence of) an edge from a multi-set of edges
- **maxId** signature(x="edgelist"): gets the maximum numeric identifier of a multi-set of edges
- **recode** signature(object = "edgelist", src="vertexSet", dst="vertexSet"): recodes a multi-sets of edges by making the numbers of its edges refer to another "vertexSet" object
- **coerce** signature(from = "list", to = "edgelist"): constructs a multi-set of edges from list input
Warning

All non-edge elements of the input list are silently discarded by the constructor.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

edge-class, undirectedEdge-class and directedEdge-class.

generalGraph-class Class "generalGraph"

Description

A class for general graphs.

Objects from the Class

Objects can be created by calls of the form new("generalGraph", ...). A generalGraph object consists of two slots, one for each possible representation: incidenceMatrix and incidenceList.

Slots

incidenceMatrix: Object of class "incidenceMatrix"

incidenceList: Object of class "incidenceList"

Extends

Class "anyGraph", directly, with explicit coerce.

Methods

initialize signature(.Object="generalGraph"): constructs a general graph from one of the two possible representations

show signature(object = "generalGraph"): displays the available representations of a general graph

display signature(x = "generalGraph"): static graphical representation via package 'mathgraph'

dynamicGraph signature(object = "generalGraph"): dynamic graphical representation via package 'dynamicGraph'

incidenceList<- signature(x = "generalGraph"): sets the incidence list representation

incidenceMatrix<- signature(x = "generalGraph"): sets the incidence matrix representation

incidenceList signature(object = "generalGraph"): gets the incidence list representation
**incidenceMatrix** signature(object = "generalGraph"): gets the incidence matrix representation

**names** signature(x = "generalGraph"): gets the character vertex identifiers of a general graph

**names<-** signature(x = "generalGraph"): sets the character vertex identifiers of a general graph

**card** signature(object = "generalGraph"): returns the number of vertices and the total number of edge occurrences in a general graph

**isEmpty** signature(object = "generalGraph"): a graph object is empty if all its possible representations are empty

**isPresent** signature(e1 = "edge", ou = "generalGraph"): an edge occurs in a graph object if it occurs in its non-empty slots

**areTheSame** signature(x = "generalGraph", y = "generalGraph"): x and y are the same if their non-empty slots represent the same graph

**[** signature(x = "generalGraph"): extracts an induced subgraph

**[[** signature(x = "generalGraph"): extracts the character identifier of a vertex

**coerce** signature(from = "anyGraph", to = "generalGraph"): all but directed and undirected edges are lost in the conversion

**coerce** signature(from = "multiGraph", to = "generalGraph"): no edges are lost in the conversion as every multi-graph is a general graph

**coerce** signature(from = "simpleGraph", to = "generalGraph"): no edges are lost in the conversion as every simple-graph is a general graph

**coerce** signature(from = "generalGraph", to = "dg.graph"): conversion to class 'dg.graph' of package 'dynamicGraph'

+ signature(e1 = "generalGraph", e2 = "vertexSet"): adds a vertex set to a general graph by making the new vertices isolated

- signature(e1 = "generalGraph", e2 = "vertexSet"): removes a vertex set from a general graph by dropping all edges involving the vertex set

+ signature(e1 = "generalGraph", e2 = "edge"): adds an edge to a general graph

- signature(e1 = "generalGraph", e2 = "edge"): removes an edge from a general graph

* signature(e1 = "generalGraph", e2 = "vertexSet"): restricts a general graph to a vertex set by dropping all edges involving vertices outside the vertex set

**Note**

Graphical representation via package 'dynamicGraph' is based on coercion to class dg.graph. Coercion to class dg.graph is obtained by expanding hyper edges to sets of ordinary edges, and using dashed lines for these. Graphical representation via package 'mathgraph' is obtained by means of coercion to class simpleGraph.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

**See Also**

*incidenceMatrix-class* and *incidenceMatrix*
The package `giRaph` provides classes and methods for graph representation and manipulation in R.

Details

- `giRaph` provides a general framework for dealing with mathematical graphs. The setting is very general and includes four types of representations and four types of graphs with conversions between them.
- Other packages for mathematical graphs in R include: Bioconductor `graph`, `mathgraph`, `dynamicGraph`, `gamm`.

The package is intended as a contribution to the gR-project described by Lauritzen (2002).

Authors

Jens Henrik Badsberg, Biometry Research Unit, Danish Institute of Agricultural Sciences, DK-8830 Tjele, Denmark

Claus Dethlefsen, Center for Cardiovascular Research, Aalborg Hospital, Aarhus University Hospital, DK-9000 Aalborg, Denmark

Luca La Rocca, Department of Social, Cognitive and Quantitative Sciences, University of Modena and Reggio Emilia, IT-42100 Reggio Emilia, Italy.

References


Description

Retrieve or set the incidence list representation of a graph.

Usage

```r
incidenceList(object, ...) 
incidenceList(x, force = TRUE) <- value
```
Arguments

object a graph object from which the representation should be retrieved
... additional parameters to be used when retrieving the representation
x a graph object in which the representation should be set
force a logical value telling whether the representation should be set even if this amounts to changing the graph
value an object of class "incidenceList" containing the representation to be set

Details

The functions incidenceList and incidenceList<- are generic.

Value

The function incidenceList returns an object of class "incidenceList" containing the incidence list representation to be retrieved. The function incidenceList<- returns a graph object in which the incidence list representation has been set.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

incidenceList-class and anyGraph-class

incidenceList-class  Class "incidenceList"

Description

A class for incidence list representation of any graph

Objects from the Class

Objects can be created by calls of the form new("incidenceList", V, E).

Slots

V: Object of class "vertexSet"
E: Object of class "edgeList"
Methods

**initialize** signature(.Object = "incidenceList"): constructs an incidence list representation of a graph from a vertex set and a mult-set of edges

**show** signature(object = "incidenceList"): displays an incidence list representation

**names** signature(x = "incidenceList"): gets the character vertex identifiers of an incidence list

**names<-** signature(x = "incidenceList"): sets the character vertex identifiers of an incidence list

**card** signature(object = "incidenceList"): returns the number of vertices and the total number of edge occurrences in an incidence list

**isEmpty** signature(object = "incidenceList"): an incidence list is empty if such is its vertex set

**isPresent** signature(el = "edge", ou = "incidenceList"): tells whether an edge occurs in the graph represented by an incidence list

**areTheSame** signature(x = "incidenceList", y = "incidenceList"): x and y are the same incidence list if they represent the same graph

**[** signature(x = "incidenceList"): extracts the incidence list of an induced subgraph

**[[** signature(x = "incidenceList"): extracts the character identifier of a vertex

**coerce** signature(from = "incidenceMatrix", to = "incidenceList"): converts an incidence matrix to an incidence list

**coerce** signature(from = "adjacencyList", to = "incidenceList"): converts an adjacency list to an incidence list

**coerce** signature(from = "adjacencyMatrix", to = "incidenceList"): converts an adjacency matrix to an incidence list

**+** signature(e1 = "incidenceList", e2 = "vertexSet"): adds a vertex set to an incidence list by making the new vertices isolated

**-** signature(e1 = "incidenceList", e2 = "vertexSet"): removes a vertex set from an incidence list by dropping all edges involving the vertex set

**+** signature(e1 = "incidenceList", e2 = "edge"): adds an edge to an incidence list

**-** signature(e1 = "incidenceList", e2 = "edge"): removes an edge from an incidence list

***** signature(e1 = "incidenceList", e2 = "vertexSet"): restricts an incidence list to a vertex set by dropping all edges involving vertices outside the vertex set

Warning

All input edges whose maximum numeric identifier is greater than the actual number of vertices are silently discarded by the constructor.

Note

The names< replacement method works only if the names to be assigned can be used to construct a vertexSet object having the right cardinality, otherwise the names are left unchanged and a warning message is given.
incidenceMatrix

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

incidenceList and anyGraph-class

incidenceMatrix

Incidence matrix representation of a graph

Description

Retrieve or set the incidence matrix representation of a graph

Usage

incidenceMatrix(object, ...)
incidenceMatrix(x, force = TRUE) <- value

Arguments

object a graph object from which the representation should be retrieved
... additional parameters to be used when retrieving the representation
x a graph object in which the representation should be set
force a logical value telling whether the representation should be set even if this amounts to changing the graph
value an object of class "incidenceMatrix" containing the representation to be set

Details

The functions incidenceMatrix and incidenceMatrix<- are generic.

Value

The function incidenceMatrix returns an object of class "incidenceMatrix" containing the incidence matrix representation to be retrieved. The function incidenceMatrix<- returns a graph object in which the incidence matrix representation has been set.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

incidenceMatrix-class and generalGraph-class
incidenceMatrix-class  Class "incidenceMatrix"

Description

A class for incidence matrix representation of general graphs

Objects from the Class

Objects can be created by calls of the form new("incidenceMatrix", I).

Slots

.Data: Object of class "matrix"; a column for each vertex and a row for each edge

Extends

Class "matrix", from data part. Class "structure", by class "matrix". Class "array", by class "matrix". Class "vector", by class "matrix", with explicit coerce.

Methods

initialize signature(.Object = "incidenceMatrix"): constructs an incidence matrix representation of a general graph from a matrix of positive integers

show signature(object = "incidenceMatrix"): displays an incidence matrix representation

names signature(x = "incidenceMatrix"): gets the character vertex identifiers of an incidence matrix

names<- signature(x = "incidenceMatrix"): sets the character vertex identifiers of an incidence matrix

card signature(object = "incidenceMatrix"): returns the number of vertices and the total number of edge occurrences in an incidence matrix

isEmpty signature(object = "incidenceMatrix"): an incidence matrix is empty if it has no columns

areTheSame signature(x = "incidenceMatrix", y = "incidenceMatrix"): x and y are the same incidence matrix if they represent the same general graph

isPresent signature(el = "undirectedEdge", ou = "incidenceMatrix"): tells whether an undirected edge occurs in the graph represented by an incidence matrix

isPresent signature(el = "directedEdge", ou = "incidenceMatrix"): tells whether a directed edge occurs in the graph represented by an incidence matrix

[ signature(x = "incidenceList"): extracts the incidence matrix of an induced subgraph

[[ signature(x = "incidenceList"): extracts the character identifier of a vertex

coerce signature(from = "incidenceList", to = "incidenceMatrix"): converts an incidence list to an incidence matrix by dropping all but undirected and directed edges
**isEmpty**

isEmpty signature(from = "adjacencyList", to = "incidenceMatrix"): converts an adjacency list to an incidence matrix

**coerce** signature(from = "adjacencyMatrix", to = "incidenceMatrix"): converts an adjacency matrix to an incidence matrix

+ signature(e1 = "incidenceMatrix", e2 = "vertexSet"): adds a vertex set to an incidence matrix by making the new vertices isolated

- signature(e1 = "incidenceMatrix", e2 = "vertexSet"): removes a vertex set from an incidence matrix by dropping all edges involving the vertex set

+ signature(e1 = "incidenceMatrix", e2 = "undirectedEdge"): adds an undirected edge to an incidence matrix

+ signature(e1 = "incidenceMatrix", e2 = "directedEdge"): adds a directed edge to an incidence matrix

- signature(e1 = "incidenceMatrix", e2 = "undirectedEdge"): removes an undirected edge from an incidence matrix

- signature(e1 = "incidenceMatrix", e2 = "directedEdge"): removes a directed edge from an incidence matrix

* signature(e1 = "incidenceMatrix", e2 = "vertexSet"): restricts an incidence matrix to a vertex set by dropping all edges involving vertices outside the vertex set

**Warning**

All zero input rows are discarded by the constructor.

**Note**

The names replacement method works only if the names to be assigned can be used to construct a vertexSet object having the right cardinality, otherwise the names are left unchanged and a warning message is given.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

**See Also**

**incidenceMatrix** and **generalGraph-class**

---

**isNotEmpty**

Is the object empty?

**Description**

Check whether an object is empty.
Usage

```plaintext
isEmpty(object, ...)
```

Arguments

- **object**: an object to be checked
- **...**: additional parameters to be used when checking the object

Details

The function `isEmpty` is generic.

Value

Returns a logical value telling whether the object is empty or not.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

---

**isPresent**

Is the first object present in the second one?

Description

Check whether the first object is present in the second one.

Usage

```plaintext
isPresent(el, ou)
```

Arguments

- **el**: element to look for
- **ou**: place where to look for it

Details

The function `isPresent` is generic.

Value

Returns a logical value telling whether the first object is present in the second one or not.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca
maxId

Get the maximum numeric identifier of an object

Description

Gets the maximum numeric identifier of an object.

Usage

maxId(x)

Arguments

x an R object with numeric identifiers

Details

The function maxId is generic.

Value

Returns a numeric value corresponding to the maximum numeric identifier of x.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

multiGraph-class

Class "multiGraph"

Description

A class for multi-graphs.

Objects from the Class

Objects can be created by calls of the form new("multiGraph", ...). A multiGraph object consists of three slots, one for each possible representation: adjacencyList, incidenceMatrix and incidenceList.

Slots

adjacencyList: Object of class "adjacencyList"
incidenceMatrix: Object of class "incidenceMatrix"
incidenceList: Object of class "incidenceList"
multiGraph-class

**Extends**

Class "generalGraph", directly, with explicit coerce. Class "anyGraph", directly, with explicit coerce.

**Methods**

- initialize signature(.Object="multiGraph"): constructs a multi-graph from one of the three possible representations
- show signature(object = "multiGraph"): displays the available representations of a multi-graph
- display signature(x = "multiGraph"): static graphical representation via package 'mathgraph'
- dynamicGraph signature(object = "multiGraph"): dynamic graphical representation via package 'dynamicGraph'
- incidenceList<- signature(x = "multiGraph"): sets the incidence list representation
- incidenceMatrix<- signature(x = "multiGraph"): sets the incidence matrix representation
- adjacencyList<- signature(x = "multiGraph"): sets the adjacency list representation
- incidenceList signature(object = "multiGraph"): gets the incidence list representation
- incidenceMatrix signature(object = "multiGraph"): gets the incidence matrix representation
- adjacencyList signature(object = "multiGraph"): gets the adjacency list representation
- names signature(x = "multiGraph"): gets the character vertex identifiers of a multi-graph
- names<- signature(x = "multiGraph"): sets the character vertex identifiers of a multi-graph
- card signature(object = "multiGraph"): returns the number of vertices and the total number of edge occurrences in a multi-graph
- isEmpty signature(object = "multiGraph"): a graph object is empty if all its possible representations are empty
- isPresent signature(el = "edge", ou = "multiGraph"): an edge occurs in a graph object if it occurs in its non-empty slots
- areTheSame signature(x = "multiGraph", y = "multiGraph"): x and y are the same if their non-empty slots represent the same graph
- [ signature(x = "multiGraph"): extracts an induced subgraph
- [[ signature(x = "multiGraph"): extracts the character identifier of a vertex
- coerce signature(from = "anyGraph", to = "multiGraph"): all but ordinary directed and undirected edges are lost in the conversion
- coerce signature(from = "generalGraph", to = "multiGraph"): hyper-edges are lost in the conversion
- coerce signature(from = "simpleGraph", to = "multiGraph"): no edges are lost in the conversion as every simple-graph is a multi-graph
- coerce signature(from = "multiGraph", to = "dg.simple.graph"): conversion to class 'dg.simple.graph' of package 'dynamicGraph'
- coerce signature(from = "multiGraph", to = "dg.graph"): conversion to class 'dg.graph' of package 'dynamicGraph'
recode

+ signature(e1 = "multiGraph", e2 = "vertexSet"): adds a vertex set to a multi-graph by making the new vertices isolated
- signature(e1 = "multiGraph", e2 = "vertexSet"): removes a vertex set from a multi-graph by dropping all edges involving the vertex set
+ signature(e1 = "multiGraph", e2 = "edge"): adds an edge to a multi-graph
- signature(e1 = "multiGraph", e2 = "edge"): removes an edge from a multi-graph
* signature(e1 = "multiGraph", e2 = "vertexSet"): restricts a multi-graph to a vertex set by dropping all edges involving vertices outside the vertex set

Note

Graphical representation via package 'dynamicGraph' is based on coercion to class dgNgraph, implemented via coercion to class dgNsimpleNgraph. Coercion to class dgNsimpleNgraph is implemented via coercion to class simplegraph, thus dropping loops and parallel edges. Graphical representation via package 'mathgraph' is obtained by means of coercion to class simpleGraph.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

adjacencyList-class and adjacencyList

---

recode

Function to recode an object from a given source code to a given target code

Description

Recodes an object from a given source code to a given target code

Usage

recode(object, src, dst)

Arguments

object an object to be recoded
src an object containing the source code
dst an object containing the target code

Details

The function recode is generic.
**Value**

The recoded object is returned.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

---

```
showRel
```

*Function to show an object relative to a given code*

**Description**

Shows an object relative to a given code.

**Usage**

```
showRel(object, code)
```

**Arguments**

- `object` : an R object to be shown
- `code` : object containing the code to be used

**Details**

The function `showRel` is generic.

**Value**

Returns an invisible NULL.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca
simpleGraph-class

Description

A class for simple-graphs

Objects from the Class

Objects can be created by calls of the form `new("simpleGraph", ...)`. A `simpleGraph` object consists of four slots, one for each possible representation: `adjacencyMatrix`, `adjacencyList`, `incidenceMatrix` and `incidenceList`.

Slots

- `adjacencyMatrix`: Object of class "adjacencyMatrix"
- `adjacencyList`: Object of class "adjacencyList"
- `incidenceMatrix`: Object of class "incidenceMatrix"
- `incidenceList`: Object of class "incidenceList"

Extends

Class "multiGraph", directly, with explicit coerce. Class "generalGraph", directly, with explicit coerce. Class "anyGraph", directly, with explicit coerce.

Methods

- `initialize` signature(`Object="simpleGraph"`): constructs a simple-graph from one of the four possible representations
- `show` signature(`object = "simpleGraph"`): displays the available representations of a simple graph
- `display` signature(`x = "simpleGraph"`): static graphical representation via package 'mathgraph'
- `dynamic.Graph` signature(`object = "simpleGraph"`): dynamic graphical representation via package 'dynamicGraph'
- `incidenceList<-` signature(`x = "simpleGraph"`): sets the incidence list representation
- `incidenceMatrix<-` signature(`x = "simpleGraph"`): sets the incidence matrix representation
- `adjacencyList<-` signature(`x = "simpleGraph"`): sets the adjacency list representation
- `adjacencyMatrix<-` signature(`x = "simpleGraph"`): sets the adjacency matrix representation
- `incidenceList` signature(`object = "simpleGraph"`): gets the incidence list representation
- `incidenceMatrix` signature(`object = "simpleGraph"`): gets the incidence matrix representation
- `adjacencyList` signature(`object = "simpleGraph"`): gets the adjacency list representation
adjacencyMatrix signature(object = "simpleGraph"): gets the adjacency matrix representation

names signature(x = "simpleGraph"): gets the character vertex identifiers of a simple graph

names<- signature(x = "simpleGraph"): sets the character vertex identifiers of a simple graph
card signature(object = "simpleGraph"): returns the number of vertices and the total number of edges (directed and undirected) in a simple graph

isEmpty signature(object = "simpleGraph"): a graph object is empty if all its possible representations are empty

isPresent signature(el = "edge", ou = "simpleGraph"): an edge occurs in a graph object if it occurs in its non-empty slots

areTheSame signature(x = "simpleGraph", y = "simpleGraph"): x and y are the same if their non-empty slots represent the same graph

[ signature(x = "simpleGraph"): extracts an induced subgraph

[[ signature(x = "simpleGraph"): extracts the character identifier of a vertex

cOerce signature(from = "anyGraph", to = "simpleGraph"): only ordinary directed and undirected edges, but no loops nor parallel edges, are kept in the conversion

cOerce signature(from = "generalGraph", to = "simpleGraph"): hyper-edges, loops and parallel edges are lost in the conversion

coerce signature(from = "multiGraph", to = "simpleGraph"): loops and parallel edges are lost in the conversion

coerce signature(from = "mathgraph", to = "simpleGraph"): conversion from class 'mathgraph' of package 'mathgraph'

coerce signature(from = "simpleGraph", to = "mathgraph"): conversion to class 'mathgraph' of package 'mathgraph'

coerce signature(from = "simpleGraph", to = "dg.simple.graph"): conversion to class 'dg.simple.graph' of package 'dynamicGraph'

coerce signature(from = "simpleGraph", to = "dg.graph"): conversion to class 'dg.graph' of package 'dynamicGraph'

+ signature(e1 = "simpleGraph", e2 = "vertexSet"): adds a vertex set to a simple graph by making the new vertices isolated

- signature(e1 = "simpleGraph", e2 = "vertexSet"): removes a vertex set from a simple graph by dropping all edges involving the vertex set

+ signature(e1 = "simpleGraph", e2 = "undirectedEdge"): adds an ordinary undirected edge (not a loop) to a simple graph

+ signature(e1 = "simpleGraph", e2 = "directedEdge"): adds an ordinary directed edge to a simple graph

- signature(e1 = "simpleGraph", e2 = "edge"): removes an edge from a simple graph

* signature(e1 = "simpleGraph", e2 = "vertexSet"): restricts a simple graph to a vertex set by dropping all edges involving vertices outside the vertex set

**Note**

Graphical representation via package 'dynamicGraph' is based on coercion to class dg.graph, implemented via coercion to class dg.simple.graph.
Class "undirectedEdge"

Description
Class for undirected edges

Objects from the Class
Objects can be created by calls of the form \texttt{new("undirectedEdge", \ldots)} which admit shorthands of the form \texttt{u(\ldots)}.

Slots
\begin{itemize}
\item [.Data]: Object of class "vector" storing strictly positive numbers that refer to a given "vertexSet" object
\end{itemize}

Extends
Class "edge", directly. Class "integer", from data part. Class "vector", by class "integer". Class "numeric", by class "integer".

Methods
\begin{itemize}
\item \texttt{initialize} signature(.Object = "undirectedEdge"): constructs an undirected edge from a vector of strictly positive integers
\item \texttt{show} signature(object = "undirectedEdge"): displays an undirected edge as numbers joined by lines
\item \texttt{showRel} signature(object = "undirectedEdge", code="vertexSet"): displays an undirected edge as names joined by lines
\item \texttt{areTheSame} signature(x = "undirectedEdge", y = "undirectedEdge"): \(x\) and \(y\) are the same undirected edge if they are the same set of numbers
\item \texttt{[} signature(x = "undirectedEdge"): extracts an undirected edge
\item \texttt{coerce} signature(from = "vector", to = "undirectedEdge"): constructs an undirected edge from vector input
\item \texttt{coerce} signature(from = "directedEdge", to = "undirectedEdge"): makes a directed edge undirected
\end{itemize}
**vertexSet-class**

**maxId** signature(x="undirectedEdge"): gets the maximum numeric identifier of an undirected edge

**recode** signature(object = "undirectedEdge", src="vertexSet", dst="vertexSet"): re-codes an undirected edge by making its numbers refer to another "vertexSet" object

**Warning**

The constructor will try to handle any vector input by silently transforming it into a list of strictly positive integers.

**Author(s)**

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

**See Also**

`edge-class, directedEdge-class, edgeList-class` and `u`.

---

**vertexSet-class**  
Class "vertexSet"

**Description**

Class for vertex sets

**Objects from the Class**

Objects can be created by calls of the form `new("vertexSet", ...)` which admit short-hands of the form `v(...)`.

**Slots**

.Data: Object of class "vector" storing unique character identifiers that are syntactically valid names

**Extends**

Class "character", from data part. Class "vector", by class "character".

**Methods**

initialize signature(.Object = "vertexSet"): constructs a vertex set from a vector of unique syntactically valid names

show signature(object = "vertexSet"): displays a vertex set as comma separated characters in graph brackets

areTheSame signature(x = "vertexSet", y = "vertexSet"): x and y are the same vertex set if they are the same set of character identifiers


wrappers

[ signature(x = "vertexSet"): extracts a vertex set
coerce signature(from = "vector", to = "vertexSet"): constructs a vertex set from vector input
names signature(x = "vertexSet"): gets the character vertex identifiers
+ signature(e1 = "vertexSet", e2 = "vertexSet"): performs the union of two vertex sets
* signature(e1 = "vertexSet", e2 = "vertexSet"): performs the intersection of two vertex sets
- signature(e1 = "vertexSet", e2 = "vertexSet"): performs the asymmetric difference of two vertex sets

Warning

The constructor will try to handle any vector input by silently transforming it into a vector of unique syntactically valid names obtained via make.names from the unique input elements.

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

The short-hand v.

---

wrappers

Short-hands for vertex set and edge construction

Description

Provide short-hands for vertex set and edge construction.

Usage

v(...)   u(...)   d(...)   r(...)

Arguments

...    unique vertex identifiers (should be characters for v and strictly positive integers for u, d and r)

Details

Function d builds a directed edge from tail to head, function r builds a directed edge from head to tail.
Value

Function v returns an object of class "vertexSet". Function u returns an object of class "undirectedEdge". Both functions d and r return an object of class "directedEdge".

Author(s)

Jens Henrik Badsberg, Claus Dethlefsen, Luca La Rocca

See Also

vertexSet-class, undirectedEdge-class and directedEdge-class.
Index

*Topic classes
  adjacencyList-class, 3
  adjacencyMatrix-class, 6
  anyGraph-class, 7
  directedEdge-class, 10
  edge-class, 14
  edgelist-class, 15
  generalGraph-class, 16
  incidenceList-class, 19
  incidenceMatrix-class, 22
  multigraph-class, 25
  simpleGraph-class, 29
  undirectedEdge-class, 31
  vertexSet-class, 32

*Topic graphs
  display, 12
  dynamicGraph, 13
  giraph, 18

*Topic methods
  adjacencyList, 2
  adjacencyMatrix, 5
  areTheSame, 9
  card, 10
  display, 12
  dynamicGraph, 13
  incidenceList, 18
  incidenceMatrix, 21
  isEmpty, 23
  isPresent, 24
  maxId, 25
  recode, 27
  showRel, 28
  wrappers, 33

*Topic models
  giraph, 18
  *,adjacencyList,vertexSet-method
    (adjacencyList-class), 3
  *,adjacencyMatrix,vertexSet-method
    (adjacencyMatrix-class), 6
  *,anyGraph,vertexSet-method
    (anyGraph-class), 7
  *,generalGraph,vertexSet-method
    (generalGraph-class), 16
  *,incidenceList,vertexSet-method
    (incidenceList-class), 19
  *,incidenceMatrix,vertexSet-method
    (incidenceMatrix-class), 22
  *,multiGraph,vertexSet-method
    (multiGraph-class), 25
  *,simpleGraph,vertexSet-method
    (simpleGraph-class), 29
  *,vertexSet,vertexSet-method
    (vertexSet-class), 32
  +,adjacencyList,directedEdge-method
    (adjacencyList-class), 3
  +,adjacencyList,undirectedEdge-method
    (adjacencyList-class), 3
  +,adjacencyList,vertexSet-method
    (adjacencyList-class), 3
  +,adjacencyMatrix,directedEdge-method
    (adjacencyMatrix-class), 6
  +,adjacencyMatrix,undirectedEdge-method
    (adjacencyMatrix-class), 6
  +,adjacencyMatrix,vertexSet-method
    (adjacencyMatrix-class), 6
  +,anyGraph,edge-method
    (anyGraph-class), 7
  +,anyGraph,vertexSet-method
    (anyGraph-class), 7
  +,edge,edgelist-method
    (edgelist-class), 15
  +,edgelist,edge-method
    (edgelist-class), 15
  +,generalGraph,edge-method
    (generalGraph-class), 16
  +,generalGraph,vertexSet-method
    (generalGraph-class), 16
  +,incidenceList,edge-method
(incidenceMatrix-class), 22
[[, multiGraph-method
  (multiGraph-class), 25
[[, simpleGraph-method
  (simpleGraph-class), 29

adjacencyList, 2, 5, 27
adjacencyList, multiGraph-method
  (multiGraph-class), 25
adjacencyList, simpleGraph-method
  (simpleGraph-class), 29
adjacencyList-class, 3
adjacencyList<- (adjacencyList), 2
adjacencyList<-, multiGraph-method
  (multiGraph-class), 25
adjacencyList<-, simpleGraph-method
  (simpleGraph-class), 29
adjacencyMatrix, 5, 7, 37
adjacencyMatrix, simpleGraph-method
  (simpleGraph-class), 29
adjacencyMatrix-class, 6
adjacencyMatrix<- (adjacencyMatrix), 5
adjacencyMatrix<-, simpleGraph-method
  (simpleGraph-class), 29
anyGraph-class, 7
areTheSame, 9
areTheSame, adjacencyList, adjacencyList-method
coerce, adjacencyList-method
  (adjacencyList-class), 3
areTheSame, adjacencyMatrix, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
areTheSame, anyGraph, anyGraph-method
  (anyGraph-class), 7
areTheSame, directedEdge, directedEdge-method
  (directedEdge-class), 10
areTheSame, edge, edge-method
  (edge-class), 14
areTheSame, edgelist, edgelist-method
  (edgelist-class), 15
areTheSame, generalGraph, generalGraph-method
  (generalGraph-class), 16
areTheSame, incidenceList, incidenceList-method
coerce, anyGraph, generalGraph-method
  (generalGraph-class), 16
areTheSame, incidenceMatrix, incidenceMatrix-method
  (incidenceMatrix-class), 22
areTheSame, multiGraph, multiGraph-method
  (multiGraph-class), 25
areTheSame, simpleGraph, simpleGraph-method
  (simpleGraph-class), 29

areTheSame, undirectedEdge, undirectedEdge-method
  (undirectedEdge-class), 31
areTheSame, vertexSet, vertexSet-method
  (vertexSet-class), 32
card, 10
card, adjacencyList-method
  (adjacencyList-class), 3
card, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
card, anyGraph-method (anyGraph-class), 7
card, directedEdge-method
  (directedEdge-class), 10
card, generalGraph-method
  (generalGraph-class), 16
card, incidenceList-method
  (incidenceList-class), 19
card, incidenceMatrix-method
  (incidenceMatrix-class), 22
card, multiGraph-method
  (multiGraph-class), 25
card, simpleGraph-method
  (simpleGraph-class), 29
card, vector-method (card), 10
coerce, adjacencyList, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
coerce, adjacencyList, incidenceList-method
  (incidenceList-class), 19
coerce, adjacencyMatrix, incidenceMatrix-method
  (incidenceMatrix-class), 22
coerce, anyGraph, dg.graph-method
  (anyGraph-class), 7
coerce, anyGraph, generalGraph-method
  (generalGraph-class), 16
coerce, directedEdge, undirectedEdge-method
  (undirectedEdge-class), 31
coerce, generalGraph, anyGraph-method
  (anyGraph-class), 7
coerce, generalGraph, dg.graph-method (generalGraph-class), 16
coerce, generalGraph, multiGraph-method (multiGraph-class), 25
coerce, generalGraph, simpleGraph-method (simpleGraph-class), 29
coerce, incidenceList, adjacencyList-method (adjacencyList-class), 3
coeerce, incidenceList, adjacencyMatrix-method (adjacencyMatrix-class), 6
coeerce, incidenceList, incidenceMatrix-method (incidenceMatrix-class), 6
coeerce, incidenceMatrix, adjacencyList-method (adjacencyList-class), 3
coeerce, incidenceMatrix, adjacencyMatrix-method (adjacencyMatrix-class), 6
coeerce, incidenceMatrix, incidenceList-method (incidenceList-class), 19
coerce, list, edgeList-method (edgeList-class), 15
coeerce, mathgraph, simpleGraph-method (simpleGraph-class), 29
coeerce, multiGraph, anyGraph-method (anyGraph-class), 7
coeerce, multiGraph, dg.graph-method (multiGraph-class), 25
coeerce, multiGraph, dg.simple.graph-method (multiGraph-class), 25
coeerce, multiGraph, generalGraph-method (generalGraph-class), 16
coeerce, multiGraph, simpleGraph-method (simpleGraph-class), 29
coeerce, simpleGraph, anyGraph-method (anyGraph-class), 7
coeerce, simpleGraph, dg.graph-method (simpleGraph-class), 29
coeerce, simpleGraph, dg.simple.graph-method (simpleGraph-class), 29
coeerce, simpleGraph, generalGraph-method (generalGraph-class), 16
coeerce, simpleGraph, mathgraph-method (simpleGraph-class), 29
coeerce, simpleGraph, multiGraph-method (multiGraph-class), 25
coeerce, undirectedEdge, directedEdge-method (directedEdge-class), 10
coeerce, vector, directedEdge-method (directedEdge-class), 10
coeerce, vector, undirectedEdge-method (undirectedEdge-class), 31
coeerce, vector, vertexSet-method (vertexSet-class), 32
d, 11
d (wrappers), 33
directedEdge-class, 10
display, 12
display, anyGraph-method (anyGraph-class), 7
display, generalGraph-method (generalGraph-class), 16
display, multiGraph-method (multiGraph-class), 25
display, simpleGraph-method (simpleGraph-class), 29
dynamicGraph, 13
dynamicGraph, anyGraph-method (anyGraph-class), 7
dynamicGraph, generalGraph-method (generalGraph-class), 16
dynamicGraph, multiGraph-method (multiGraph-class), 25
dynamicGraph, simpleGraph-method (simpleGraph-class), 29
DynamicGraph, 13
eledge-class, 14
edgelist-class, 15
generalGraph-class, 16
giraph, 18
incidenceList, 9, 18, 21
incidenceList, anyGraph-method (anyGraph-class), 7
incidenceList, generalGraph-method (generalGraph-class), 16
incidenceList, multiGraph-method (multiGraph-class), 25
incidenceList, simpleGraph-method (simpleGraph-class), 29
incidenceList-class, 19
incidenceList<-, incidenceList, 18
incidenceList<-, anyGraph-method (anyGraph-class), 7
incidenceList<-, generalGraph-method (generalGraph-class), 16
incidenceList<-, multiGraph-method
  (multiGraph-class), 25
incidenceList<-, simpleGraph-method
  (simpleGraph-class), 29
incidenceMatrix, 17, 21, 23
incidenceMatrix, generalGraph-method
  (generalGraph-class), 16
incidenceMatrix, multiGraph-method
  (multiGraph-class), 25
incidenceMatrix, simpleGraph-method
  (simpleGraph-class), 29

incidenceMatrix-class, 22
incidenceMatrix<-(incidenceMatrix), 21
incidenceMatrix<-, generalGraph-method
  (generalGraph-class), 16
incidenceMatrix<-, multiGraph-method
  (multiGraph-class), 25
incidenceMatrix<-, simpleGraph-method
  (simpleGraph-class), 29
initialize, adjacencyList-method
  (adjacencyList-class), 3
initialize, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
initialize, anyGraph-method
  (anyGraph-class), 7
initialize, directedEdge-method
  (directedEdge-class), 10
initialize, edgeList-method
  (edgeList-class), 15
initialize, generalGraph-method
  (generalGraph-class), 16
initialize, incidenceList-method
  (incidenceList-class), 19
initialize, incidenceMatrix-method
  (incidenceMatrix-class), 22
initialize, multiGraph-method
  (multiGraph-class), 25
initialize, simpleGraph-method
  (simpleGraph-class), 29
initialize, undirectedEdge-method
  (undirectedEdge-class), 31
initialize, vertexSet-method
  (vertexSet-class), 32
isEmpty, 23
isEmpty, adjacencyList-method
  (adjacencyList-class), 3
isEmpty, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
isEmpty, anyGraph-method
  (anyGraph-class), 7
isEmpty, generalGraph-method
  (generalGraph-class), 16
isEmpty, incidenceList-method
  (incidenceList-class), 19
isEmpty, incidenceMatrix-method
  (incidenceMatrix-class), 22
isEmpty, multiGraph-method
  (multiGraph-class), 25
isEmpty, NULL-method(isEmpty), 23
isEmpty, simpleGraph-method
  (simpleGraph-class), 29
isEmpty, vector-method(isEmpty), 23
isPresent, 24
isPresent, directedEdge, adjacencyList-method
  (adjacencyList-class), 3
isPresent, directedEdge, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
isPresent, directedEdge, incidenceMatrix-method
  (incidenceMatrix-class), 22
isPresent, edge, anyGraph-method
  (anyGraph-class), 7
isPresent, edge, edgeList-method
  (edgeList-class), 15
isPresent, edge, generalGraph-method
  (generalGraph-class), 16
isPresent, edge, incidenceList-method
  (incidenceList-class), 19
isPresent, edge, multiGraph-method
  (multiGraph-class), 25
isPresent, edge, simpleGraph-method
  (simpleGraph-class), 29
isPresent, undirectedEdge, adjacencyList-method
  (adjacencyList-class), 3
isPresent, undirectedEdge, adjacencyMatrix-method
  (adjacencyMatrix-class), 6
isPresent, undirectedEdge, incidenceMatrix-method
  (incidenceMatrix-class), 22
make.names, 33
maxId, 25
maxId, directedEdge-method
  (directedEdge-class), 10
maxId, edgeList-method(edgeList-class), 15
maxId, undirectedEdge-method
  (undirectedEdge-class), 31
multiGraph-class, 25
names, adjacencyList-method
(adjacencyList-class), 3
names, adjacencyMatrix-method
(adjacencyMatrix-class), 6
names, anyGraph-method (anyGraph-class), 7
names, generalGraph-method
(generalGraph-class), 16
names, incidenceList-method
(incidenceList-class), 19
names, incidenceMatrix-method
(incidenceMatrix-class), 22
names, multiGraph-method
(multiGraph-class), 25
names, simpleGraph-method
(simpleGraph-class), 29
names, vertexSet-method
(vertexSet-class), 32
names<-, adjacencyList-method
(adjacencyList-class), 3
names<-, adjacencyMatrix-method
(adjacencyMatrix-class), 6
names<-, anyGraph-method
(anyGraph-class), 7
names<-, generalGraph-method
(generalGraph-class), 16
names<-, incidenceList-method
(incidenceList-class), 19
names<-, incidenceMatrix-method
(incidenceMatrix-class), 22
names<-, multiGraph-method
(multiGraph-class), 25
names<-, simpleGraph-method
(simpleGraph-class), 29

plot.mathgraph, 12, 13

r, 11
r (wrappers), 33
recode, 27
recode, directedEdge, vertexSet, vertexSet-method
(directedEdge-class), 10
recode, edgeList, vertexSet, vertexSet-method
(edgeList-class), 15
recode, undirectedEdge, vertexSet, vertexSet-method
(undirectedEdge-class), 31

show, adjacencyList-method
(adjacencyList-class), 3
show, adjacencyMatrix-method
(adjacencyMatrix-class), 6
show, anyGraph-method (anyGraph-class), 7
show, directedEdge-method
(directedEdge-class), 10
show, edgeList-method (edgeList-class), 15
show, generalGraph-method
(generalGraph-class), 16
show, incidenceList-method
(incidenceList-class), 19
show, incidenceMatrix-method
(incidenceMatrix-class), 22
show, multiGraph-method
(multiGraph-class), 25
show, simpleGraph-method
(simpleGraph-class), 29
show, undirectedEdge-method
(undirectedEdge-class), 31
show, vertexSet-method
(vertexSet-class), 32

showRel, 28
showRel, directedEdge, vertexSet-method
(directedEdge-class), 10
showRel, edgeList, vertexSet-method
(edgeList-class), 15
showRel, undirectedEdge, vertexSet-method
(undirectedEdge-class), 31
simpleGraph-class, 29

u, 32
u (wrappers), 33
undirectedEdge-class, 31

v, 33
v (wrappers), 33
vertexSet-class, 32

wrappers, 33