Class Graph

java.lang.Object
    Graph

```
public class Graph
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`Graph(java.lang.String gfn)` Instantiate a graph

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`Edge`	`getEdge(java.lang.String src, java.lang.String dst)` Returns the edge from src to dst.
`Edge`	`getEdge(Vertex vsrc, Vertex vdst)` Returns the edge from vsrc to vdst.
`java.util.ArrayList<Edge>`	`getEdgesUndirected()` (to be used for Kruskal's algorithm) internally convert directed graph to undirected graph and return the edges sorted by cost and label in increasing order
`double[][]`	`getMat()` Returns the adjacency matrix of this graph
`Vertex`	`getVertex(int index)` Returns the index-th vertex (0-based).

Vertex	**getVertex**(java.lang.String label) Returns the vertex associated with label.
java.util.ArrayList<Vertex>	**getVerts**() returns the vertex list of this graph
static void	**main**(java.lang.String[] args)
int	**nEdges**() returns the number of edges in this graph
int	**nVerts**() returns the number of vertices in this graph
void	**resetVerts**()
void	**resetVerts**(boolean mark, double newCost, Vertex newPred) Resets predecessor, cost, and marked of all vertices in this graph to the given values.
java.lang.String	**toString**() returns the information of this graph in String format
void	**toUndirected**() (to be used before applying the MST algorithms) for each directed edge (v to u), add edge (u to v) if it doesn't exist if it does, update edge (u to v)'s weight to the weight of edge (v to u) update adjacency matrix accordingly

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Graph

public Graph(java.lang.String gfn)

Instantiate a graph

Parameters:

gfn - text file containing the information of the graph

Method Detail

resetVerts

```
public void resetVerts(boolean mark,
                       double newCost,
                       Vertex newPred)
```

Resets predecessor, cost, and marked of all vertices in this graph to the given values.

Parameters:

`mark - the mark value`

`newCost - the new cost`

`newPred - the new predecessor`

resetVerts

```
public void resetVerts()
```

getVerts

```
public java.util.ArrayList<Vertex> getVerts()
```

returns the vertex list of this graph

Returns:

`the vertex list of this graph`

nVerts

```
public int nVerts()
```

returns the number of vertices in this graph

Returns:

`number of vertices in the graph`

nEdges

```
public int nEdges()
```

returns the number of edges in this graph

Returns:

the number of edges in this graph

getVertex

```
public Vertex getVertex(java.lang.String label)
```

Returns the vertex associated with label. Returns null if no vertex in this graph matches the label.

Parameters:

label - the label of the target edge

Returns:

returns the target vertex, null if not found

getVertex

```
public Vertex getVertex(int index)
```

Returns the index-th vertex (0-based). Returns null if index is invalid.

Parameters:

index - the ID of the vertex

Returns:

the target vertex, null if not found

getEdge

```
public Edge getEdge(Vertex vsrc,
                    Vertex vdst)
```

Returns the edge from vsrc to vdst. Returns null if no edge exists from vsrc to vdst.

Parameters:

vsrc - the source vertex

vdst - the destination vertex

Returns:

the target edge, null if not found

getEdge

```
public Edge getEdge(java.lang.String src,
                    java.lang.String dst)
```

Returns the edge from src to dst. Returns null if no edge exists from src to dst.

Parameters:

`src - the source vertex`

`dst - the source vertex`

Returns:

`the target edge, if not found returns null`

getMat

```
public double[][] getMat()
```

Returns the adjacency matrix of this graph

Returns:

`the adjacency matrix of this graph`

toString

```
public java.lang.String toString()
```

returns the information of this graph in String format

Overrides:

`toString in class java.lang.Object`

getEdgesUndirected

```
public java.util.ArrayList<Edge> getEdgesUndirected()
```

(to be used for Kruskal's algorithm) internally convert directed graph to undirected graph and return the edges sorted by cost and label in increasing order

Returns:

`list of edges sorted by cost first then by edge label in increasing order`

toUndirected

```
public void toUndirected()
```

(to be used before applying the MST algorithms) for each directed edge (v to u), add

edge (u to v) if it doesn't exist if it does, update edge (u to v)'s weight to the weight of edge (v to u) update adjacency matrix accordingly

main

```
public static void main(java.lang.String[] args)
```