## NAME

GraphMatrix

## SYNOPSIS

use Graph::GraphMatrix;

use Graph::GraphMatrix qw(:all);

## DESCRIPTION

GraphMatrix class provides the following methods:

new, GenerateAdjacencyMatrix, GenerateAdmittanceMatrix, GenerateDegreeMatrix, GenerateDistanceMatrix, GenerateIncidenceMatrix, GenerateKirchhoffMatrix, GenerateLaplacianMatrix, GenerateNormalizedLaplacianMatrix, GenerateSiedelAdjacencyMatrix, GetColumnIDs, GetMatrix, GetMatrixType, GetRowIDs, StringifyGraphMatrix

## METHODS

### new

```
$NewGraphMatrix = new Graph::GraphMatrix($Graph);
```

Using specified *Graph*, new method creates a new GraphMatrix and returns newly created GraphMatrix.

### GenerateAdjacencyMatrix

```
$AdjacencyGraphMatrix = $GraphMatrix->GenerateAdjacencyMatrix();
```

Generates a new *AdjacencyGraphMatrix* for specified Graph and returns *AdjacencyGraphMatrix*.

For a simple graph G with n vertices, the adjacency matrix for G is a n x n square matrix and its elements Mij are:

```
. 0    if i == j
. 1    if i != j and vertex Vi is adjacent to vertex Vj
. 0    if i != j and vertex Vi is not adjacent to vertex Vj
```

### GenerateAdmittanceMatrix

```
$AdmittanceGraphMatrix = $GraphMatrix->GenerateAdmittanceMatrix();
```

Generates a new *AdmittanceGraphMatrix* for specified Graph and returns *AdmittanceGraphMatrix*.

AdmittanceMatrix is another name for LaplacianMatrix.

### GenerateDegreeMatrix

```
$DegreeGraphMatrix = $GraphMatrix->GenerateDegreeMatrix();
```

Generates a new *DegreeGraphMatrix* for specified Graph and returns *DegreeGraphMatrix*.

For a simple graph G with n vertices, the degree matrix for G is a n x n square matrix and its elements Mij are:

```
. deg(Vi)   if i == j and deg(Vi) is the degree of vertex Vi
. 0         otherwise
```

### GenerateDistanceMatrix

```
$DistanceGraphMatrix = $GraphMatrix->GenerateDistanceMatrix();
```

Generates a new *DistanceGraphMatrix* for specified Graph using Floyd-Marshall algorithm [Ref 67] and returns *DistanceGraphMatrix*.

For a simple graph G with n vertices, the distance matrix for G is a n x n square matrix and its elements Mij are:

```
. 0    if i == j
. d    if i != j and d is the shortest distance between vertex Vi and vertex Vj
```

In the final matrix, value of constant BigNumber defined in Constants.pm module corresponds to vertices with no edges.

### GenerateIncidenceMatrix

```
    $IncidenceGraphMatrix = $GraphMatrix->GenerateIncidenceMatrix();
```

Generates a new *IncidenceGraphMatrix* for specified Graph and returns *IncidenceGraphMatrix*.

For a simple graph G with n vertices and e edges, the incidence matrix for G is a n x e matrix its elements Mij are:

```
    . 1    if vertex Vi and the edge Ej are incident; in other words, Vi and Ej are
related
    . 0    otherwise
```

GenerateKirchhoffMatrix

```
    $KirchhoffGraphMatrix = $GraphMatrix->GenerateKirchhoffMatrix();
```

Generates a new *KirchhoffGraphMatrix* for specified Graph and returns *KirchhoffGraphMatrix*.

KirchhoffMatrix is another name for LaplacianMatrix.

GenerateLaplacianMatrix

```
    $LaplacianGraphMatrix = $GraphMatrix->GenerateLaplacianMatrix();
```

Generates a new *LaplacianGraphMatrix* for specified Graph and returns *LaplacianGraphMatrix*.

For a simple graph G with n vertices, the Laplacian matrix for G is a n x n square matrix and its elements Mij are:

```
    . deg(Vi)   if i == j and deg(Vi) is the degree of vertex Vi
    . -1        if i != j and vertex Vi is adjacent to vertex Vj
    . 0         otherwise
```

The Laplacian matrix is the difference between the degree matrix and adjacency matrix.

GenerateNormalizedLaplacianMatrix

```
    $NormalizedLaplacianGraphMatrix =
$GraphMatrix->GenerateNormalizedLaplacianMatrix();
```

Generates a new *NormalizedLaplacianGraphMatrix* for specified Graph and returns *NormalizedLaplacianGraphMatrix*.

For a simple graph G with n vertices, the normalized Laplacian matrix L for G is a n x n square matrix and its elements Lij are:

```
    .  1                          if i == j and deg(Vi) != 0
    .  -1/SQRT(deg(Vi) * deg(Vj))  if i != j and vertex Vi is adjacent to vertex Vj
    .  0                          otherwise
```

GenerateSiedelAdjacencyMatrix

```
    $SiedelAdjacencyGraphMatrix = $GraphMatrix->GenerateSiedelAdjacencyMatrix();
```

Generates a new *SiedelAdjacencyGraphMatrix* for specified Graph and returns *SiedelAdjacencyGraphMatrix*.

For a simple graph G with n vertices, the Siedal adjacency matrix for G is a n x n square matrix and its elements Mij are:

```
    . 0    if i == j
    . -1   if i != j and vertex Vi is adjacent to vertex Vj
    . 1    if i != j and vertex Vi is not adjacent to vertex Vj
```

GetColumnIDs

```
    @ColumnIDs = $GraphMatrix->GetColumnIDs();
```

Returns an array containing any specified column IDs for *GraphMatrix*.

GetMatrix

```
    $Matrix = $GraphMatrix->GetMatrix();
```

Returns *Matrix* object corresponding to *GraphMatrix* object.

GetMatrixType

```
    $MatrixType = $GraphMatrix->GetMatrixType();
```

Returns MatrixType of *GraphMatrix*.

GetRowIDs

```
@RowIDs = $GraphMatrix->GetRowIDs();
```

Returns an array containing any specified rowIDs IDs for *GraphMatrix*.

StringifyGraphMatrix

```
$String = $GraphMatrix->StringifyGraphMatrix();
```

Returns a string containing information about *GraphMatrix* object.

## AUTHOR

Manish Sud <msud@san.rr.com>

## SEE ALSO

Constants.pm, Graph.pm, Matrix.pm

## COPYRIGHT