"Algorithmique - Complexité - M1 - Montpellier

Generated by Doxygen 1.7.4

Thu Dec 1 2011 17:49:16

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **5**

3.1 Graph Struct Reference 5

3.1.1 Detailed Description 5

3.1.2 Field Documentation 5

3.1.2.1 _lg 5

3.1.2.2 _mg 5

3.2 Item_s Struct Reference 6

3.2.1 Field Documentation 6

3.2.1.1 _next 6

3.2.1.2 _num 6

3.2.1.3 _prev 6

3.2.1.4 _val 6

3.3 List Struct Reference 6

3.3.1 Field Documentation 7

3.3.1.1 _begin 7

3.3.1.2 _end 7

3.3.1.3 _nb 7

3.4 ListGraph Struct Reference 7

3.4.1 Detailed Description 7

3.4.2 Field Documentation 7

3.4.2.1 _inNeighb . . . 7
3.4.2.2 _nbEdge . . . 7
3.4.2.3 _nbVert . . . 7
3.4.2.4 _outNeighb . . . 8
3.5 MatGraph Struct Reference . . . 8
3.5.1 Detailed Description . . . 8
3.5.2 Field Documentation . . . 8
3.5.2.1 _mat . . . 8
3.5.2.2 _nbEdge . . . 8
3.5.2.3 _nbVert . . . 8

4 File Documentation **9**
4.1 src/graph.c File Reference . . . 9
4.1.1 Function Documentation . . . 10
4.1.1.1 allocGraph . . . 10
4.1.1.2 allocListGraph . . . 10
4.1.1.3 allocMatGraph . . . 10
4.1.1.4 copyGraph . . . 11
4.1.1.5 edgeVal . . . 11
4.1.1.6 freeGraph . . . 11
4.1.1.7 inNeighb . . . 11
4.1.1.8 nbEdge . . . 12
4.1.1.9 nbVert . . . 12
4.1.1.10 outNeighb . . . 12
4.1.1.11 printGraph . . . 12
4.1.1.12 printGraphList . . . 12
4.1.1.13 randFill . . . 12
4.1.1.14 setEdge . . . 12
4.2 src/graph.h File Reference . . . 12
4.2.1 Define Documentation . . . 13
4.2.1.1 lg . . . 13
4.2.1.2 mg . . . 13
4.2.2 Function Documentation . . . 13
4.2.2.1 allocGraph . . . 14

4.2.2.2 allocListGraph . . . 14
4.2.2.3 allocMatGraph . . . 14
4.2.2.4 copyGraph . . . 14
4.2.2.5 edgeVal . . . 15
4.2.2.6 freeGraph . . . 15
4.2.2.7 inNeighb . . . 15
4.2.2.8 nbEdge . . . 15
4.2.2.9 nbVert . . . 15
4.2.2.10 outNeighb . . . 16
4.2.2.11 printGraph . . . 16
4.2.2.12 printGraphList . . . 16
4.2.2.13 randFill . . . 16
4.2.2.14 setEdge . . . 16
4.3 src/graphe_algos.c File Reference . . . 16
4.3.1 Function Documentation . . . 16
4.3.1.1 grapheEcart . . . 16
4.3.1.2 plusCourtChemin . . . 17
4.3.1.3 plusCourtCheminRec . . . 17
4.4 src/graphe_algos.h File Reference . . . 17
4.4.1 Function Documentation . . . 17
4.4.1.1 grapheEcart . . . 17
4.4.1.2 plusCourtChemin . . . 17
4.5 src/graphe_test.c File Reference . . . 17
4.5.1 Function Documentation . . . 17
4.5.1.1 remplirTest1 . . . 17
4.6 src/list.c File Reference . . . 18
4.6.1 Function Documentation . . . 18
4.6.1.1 allocList . . . 18
4.6.1.2 copyList . . . 18
4.6.1.3 freeList . . . 18
4.6.1.4 insertNum . . . 18
4.6.1.5 popHead . . . 18
4.6.1.6 popNum . . . 18
4.6.1.7 popTail . . . 18

4.6.1.8 printList . . . 18
4.6.1.9 printListReverse . . . 18
4.6.1.10 pushHead . . . 18
4.6.1.11 pushTail . . . 18
4.6.1.12 valOfNum . . . 18
4.7 src/list.h File Reference . . . 19
4.7.1 Define Documentation . . . 20
4.7.1.1 begin . . . 20
4.7.1.2 end . . . 20
4.7.1.3 nb . . . 20
4.7.1.4 next . . . 20
4.7.1.5 num . . . 20
4.7.1.6 prev . . . 20
4.7.1.7 val . . . 20
4.7.2 Typedef Documentation . . . 20
4.7.2.1 Item . . . 20
4.7.3 Function Documentation . . . 20
4.7.3.1 allocList . . . 20
4.7.3.2 copyList . . . 20
4.7.3.3 freeList . . . 20
4.7.3.4 insertNum . . . 20
4.7.3.5 popHead . . . 20
4.7.3.6 popNum . . . 20
4.7.3.7 popTail . . . 20
4.7.3.8 printList . . . 20
4.7.3.9 printListReverse . . . 20
4.7.3.10 pushHead . . . 20
4.7.3.11 pushTail . . . 20
4.7.3.12 valOfNum . . . 20
4.8 src/list_graph.c File Reference . . . 21
4.8.1 Define Documentation . . . 21
4.8.1.1 in . . . 21
4.8.1.2 nbEdge . . . 21
4.8.1.3 nbVert . . . 21

4.8.1.4 out . . . 21
4.8.2 Function Documentation . . . 21
4.8.2.1 l_allocGraph . . . 22
4.8.2.2 l_copyGraph . . . 22
4.8.2.3 l_edgeVal . . . 22
4.8.2.4 l_freeGraph . . . 22
4.8.2.5 l_inNeighb . . . 23
4.8.2.6 l_outNeighb . . . 23
4.8.2.7 l_printListGraph . . . 23
4.8.2.8 l_setEdge . . . 23
4.9 src/list_graph.h File Reference . . . 24
4.9.1 Function Documentation . . . 24
4.9.1.1 l_allocGraph . . . 24
4.9.1.2 l_copyGraph . . . 25
4.9.1.3 l_edgeVal . . . 25
4.9.1.4 l_freeGraph . . . 25
4.9.1.5 l_inNeighb . . . 25
4.9.1.6 l_outNeighb . . . 26
4.9.1.7 l_printListGraph . . . 26
4.9.1.8 l_setEdge . . . 26
4.10 src/main.c File Reference . . . 26
4.10.1 Function Documentation . . . 26
4.10.1.1 main . . . 27
4.11 src/mat_graph.c File Reference . . . 27
4.11.1 Define Documentation . . . 27
4.11.1.1 mat . . . 27
4.11.1.2 nbEdge . . . 27
4.11.1.3 nbVert . . . 27
4.11.2 Function Documentation . . . 27
4.11.2.1 m_allocGraph . . . 27
4.11.2.2 m_copyGraph . . . 28
4.11.2.3 m_edgeVal . . . 28
4.11.2.4 m_freeGraph . . . 28
4.11.2.5 m_inNeighb . . . 28

4.11.2.6 m_outNeighb 29
4.11.2.7 m_setEdge 29
4.12 src/mat_graph.h File Reference 29
4.12.1 Function Documentation 30
4.12.1.1 m_allocGraph 30
4.12.1.2 m_copyGraph 30
4.12.1.3 m_edgeVal 30
4.12.1.4 m_freeGraph 31
4.12.1.5 m_inNeighb 31
4.12.1.6 m_outNeighb 31
4.12.1.7 m_setEdge 31

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Graph (General graph structure) 5
Item_s 6
List 6
ListGraph (Implementation of a graph data strucuture represented by incidence list) 7
MatGraph (Implementation of a graph with help of an incidence matrix) 8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/graph.c . . . 9
src/graph.h . . . 12
src/graphe_algos.c . . . 16
src/graphe_algos.h . . . 17
src/graphe_test.c . . . 17
src/list.c . . . 18
src/list.h . . . 19
src/list_graph.c . . . 21
src/list_graph.h . . . 24
src/main.c . . . 26
src/mat_graph.c . . . 27
src/mat_graph.h . . . 29

Chapter 3

Data Structure Documentation

3.1 Graph Struct Reference

General graph structure.

`#include <graph.h>`

Data Fields

- MatGraph * _mg
- ListGraph * _lg

3.1.1 Detailed Description

General graph structure.

The goal of this declaration is to manipulate a single data structure no matter which representation of the graph is used. To do that, we use a data structure with only two pointers on a different kind of graph. Depending on the non null pointer, the graph which will be contained by this structure will be a matrix-represented or a list-represented graph.

3.1.2 Field Documentation

3.1.2.1 ListGraph* _lg

pointer on a list-represented graph

3.1.2.2 MatGraph* _mg

pointer on a matrix-represented graph

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

- src/graph.h

3.2 Item_s Struct Reference

`#include <list.h>`

Data Fields

- int _num
- int _val
- struct Item_s * _next
- struct Item_s * _prev

3.2.1 Field Documentation

3.2.1.1 struct Item_s* _next

3.2.1.2 int _num

3.2.1.3 struct Item_s* _prev

3.2.1.4 int _val

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

- src/list.h

3.3 List Struct Reference

`#include <list.h>`

Data Fields

- int _nb
- Item * _begin
- Item * _end

3.3.1 Field Documentation

3.3.1.1 Item* _begin

3.3.1.2 Item* _end

3.3.1.3 int _nb

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

- src/list.h

3.4 ListGraph Struct Reference

Implementation of a graph data strucuture represented by incidence list.

`#include <list_graph.h>`

Data Fields

- List ** _inNeighb
- List ** _outNeighb
- int _nbVert
- int _nbEdge

3.4.1 Detailed Description

Implementation of a graph data strucuture represented by incidence list.

3.4.2 Field Documentation

3.4.2.1 List** _inNeighb

Represents incoming neighbourhood

3.4.2.2 int _nbEdge

Stores the number of edges in the graph

3.4.2.3 int _nbVert

Stores the number of nodes in the graph

3.4.2.4 List** _outNeighb

Represents outgoing neighbourhood

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

- src/list_graph.h

3.5 MatGraph Struct Reference

Implementation of a graph with help of an incidence matrix.

`#include <mat_graph.h>`

Data Fields

- int ** _mat
- int _nbVert
- int _nbEdge

3.5.1 Detailed Description

Implementation of a graph with help of an incidence matrix.

3.5.2 Field Documentation

3.5.2.1 int** _mat

Matrix containing value of edges of the graph

3.5.2.2 int _nbEdge

Integer containing the number of edges in the graph

3.5.2.3 int _nbVert

Integer containing the number of nodes in the graph

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

- src/mat_graph.h

Chapter 4

File Documentation

4.1 src/graph.c File Reference

```
#include "graph.h"
```

Functions

- Graph * allocMatGraph (int n)

  Implements the memory allocation for a graph structure containing a matrix represented graph.
- Graph * allocListGraph (int n)

  Implements the memory allocation for a graph structure containing a list represented graph.
- Graph * allocGraph (Graph *g)

  Allows user to copy an empty graph, its usefull to create a new graph which have the same representation as the given one.
- void freeGraph (Graph *g)

  Implements memory release for a graph structure.
- Graph * copyGraph (Graph *g)

  Allows user to simply copy a graph.
- int edgeVal (Graph *g, int i, int j)

  Allows users to easily get the value of the edge between i and j.
- void setEdge (Graph *g, int i, int j, int val)

  Allows user to modify the value of the edge (i,j)
- List * inNeighb (Graph *g, int k)

  allows user to get the incoming neighbourhood of node k
- List * outNeighb (Graph *g, int k)

  allows user to get the outcoming neighbourhood of node k
- void printGraph (FILE *f, Graph *g)
- void printGraphList (FILE *f, Graph *g)

- int nbVert (Graph ∗g)
- int nbEdge (Graph ∗g)
- void randFill (Graph ∗g, int m, int cMax, int cMin, Graph ∗g2)

4.1.1 Function Documentation

4.1.1.1 Graph∗ allocGraph (Graph ∗ *g*)

Allows user to copy an empty graph, its usefull to create a new graph which have the same representation as the given one.

Parameters

g	pointer on the graph which representation will be copy

Returns

pointer on an empty graph

4.1.1.2 Graph∗ allocListGraph (int *n*)

Implements the memory allocation for a graph structure containing a list represented graph.

Parameters

n	number of nodes in the created list

Returns

pointer on the created graph

4.1.1.3 Graph∗ allocMatGraph (int *n*)

Implements the memory allocation for a graph structure containing a matrix represented graph.

Parameters

n	number of nodes in the created matrix

Returns

pointer on the created graph

4.1.1.4 Graph* copyGraph (Graph * g)

Allows user to simply copy a graph.

Parameters

g	pointer on the graph to copy

Returns

pointer on the created graph

4.1.1.5 int edgeVal (Graph * g, int i, int j)

Allows users to easily get the value of the edge between i and j.

* Interface of Graph structure *

Parameters

g	pointer on the graph which contains the edge
i	first extremity of the edge
j	second extremity

Returns

value of the edge (i,j)

4.1.1.6 void freeGraph (Graph * g)

Implements memory release for a graph structure.

Parameters

g	pointer on the graph to destroy

4.1.1.7 List* inNeighb (Graph * g, int k)

allows user to get the incoming neighbourhood of node k

Parameters

g	pointer on the graph containing k
k	node for which the neighbourhood is desired

Returns

pointer on the requested neighbourhood

4.1.1.8 int nbEdge (Graph * g)

4.1.1.9 int nbVert (Graph * g)

4.1.1.10 List* outNeighb (Graph * g, int k)

allows user to get the outcoming neighbourhood of node k

Parameters

g	pointer on the graph containing k
k	node for which the neighbourhood is desired

Returns

pointer on the requested neighbourhood

4.1.1.11 void printGraph (FILE * f, Graph * g)

4.1.1.12 void printGraphList (FILE * f, Graph * g)

4.1.1.13 void randFill (Graph * g, int m, int cMax, int cMin, Graph * g2)

4.1.1.14 void setEdge (Graph * g, int i, int j, int val)

Allows user to modify the value of the edge (i,j)

Parameters

g	pointer on the graph containing (i,j)
i	first extremity of (i,j)
j	second extremity of (i,j)
val	new capacity of (i,j)

4.2 src/graph.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "mat_graph.h"
#include "list_graph.h"
```

Data Structures

- struct Graph

General graph structure.

Defines

- #define mg(g) g->_mg
- #define lg(g) g->_lg

Functions

- Graph * allocMatGraph (int n)

  Implements the memory allocation for a graph structure containing a matrix represented graph.
- Graph * allocListGraph (int n)

  Implements the memory allocation for a graph structure containing a list represented graph.
- Graph * allocGraph (Graph *g)

  Allows user to copy an empty graph, its usefull to create a new graph which have the same representation as the given one.
- void freeGraph (Graph *g)

  Implements memory release for a graph structure.
- Graph * copyGraph (Graph *g)

  Allows user to simply copy a graph.
- int edgeVal (Graph *g, int i, int j)

  Allows users to easily get the value of the edge between i and j.
- void setEdge (Graph *g, int i, int j, int val)

  Allows user to modify the value of the edge (i,j)
- List * inNeighb (Graph *g, int k)

  allows user to get the incoming neighbourhood of node k
- List * outNeighb (Graph *g, int k)

  allows user to get the outcoming neighbourhood of node k
- void printGraph (FILE *f, Graph *g)
- void printGraphList (FILE *f, Graph *g)
- int nbVert (Graph *g)
- int nbEdge (Graph *g)
- void randFill (Graph *g, int m, int cMax, int cMin, Graph *g2)

4.2.1 Define Documentation

4.2.1.1 #define lg(*g*) g->_lg

4.2.1.2 #define mg(*g*) g->_mg

4.2.2 Function Documentation

4.2.2.1 Graph* allocGraph (Graph * g)

Allows user to copy an empty graph, its usefull to create a new graph which have the same representation as the given one.

Parameters

g	pointer on the graph which representation will be copy

Returns

pointer on an empty graph

4.2.2.2 Graph* allocListGraph (int n)

Implements the memory allocation for a graph structure containing a list represented graph.

Parameters

n	number of nodes in the created list

Returns

pointer on the created graph

4.2.2.3 Graph* allocMatGraph (int n)

Implements the memory allocation for a graph structure containing a matrix represented graph.

Parameters

n	number of nodes in the created matrix

Returns

pointer on the created graph

4.2.2.4 Graph* copyGraph (Graph * g)

Allows user to simply copy a graph.

Parameters

g	pointer on the graph to copy

Returns

pointer on the created graph

4.2.2.5 int edgeVal (Graph * g, int i, int j)

Allows users to easily get the value of the edge between i and j.

* Interface of Graph structure *

Parameters

g	pointer on the graph which contains the edge
i	first extremity of the edge
j	second extremity

Returns

value of the edge (i,j)

4.2.2.6 void freeGraph (Graph * g)

Implements memory release for a graph structure.

Parameters

g	pointer on the graph to destroy

4.2.2.7 List* inNeighb (Graph * g, int k)

allows user to get the incoming neighbourhood of node k

Parameters

g	pointer on the graph containing k
k	node for which the neighbourhood is desired

Returns

pointer on the requested neighbourhood

4.2.2.8 int nbEdge (Graph * g)

4.2.2.9 int nbVert (Graph * g)

4.2.2.10 List* outNeighb (Graph * g, int k)

allows user to get the outcoming neighbourhood of node k

Parameters

g	pointer on the graph containing k
k	node for which the neighbourhood is desired

Returns

pointer on the requested neighbourhood

4.2.2.11 void printGraph (FILE * f, Graph * g)

4.2.2.12 void printGraphList (FILE * f, Graph * g)

4.2.2.13 void randFill (Graph * g, int m, int cMax, int cMin, Graph * g2)

4.2.2.14 void setEdge (Graph * g, int i, int j, int val)

Allows user to modify the value of the edge (i,j)

Parameters

g	pointer on the graph containing (i,j)
i	first extremity of (i,j)
j	second extremity of (i,j)
val	new capacity of (i,j)

4.3 src/graphe_algos.c File Reference

```
#include "graphe_algos.h"
```

Functions

- Graphe * grapheEcart (Graphe *gCapa, Graphe *gFlot)
- Chemin * plusCourtCheminRec (Graphe *g, int s, int p, int *d, Chemin *c, int *v)
- Chemin * plusCourtChemin (Graphe *g, int s, int p)

4.3.1 Function Documentation

4.3.1.1 Graphe* grapheEcart (Graphe * gCapa, Graphe * gFlot)

gCapa : graphe des capacités gFlot : graphe du flot actuel Return : graphe d'écart

4.3.1.2 Chemin* plusCourtChemin (Graphe * *g*, int *s*, int *p*)

g : graphe orienté s : premier sommet p : dernier sommet Return : Plus court chemin en nombre d'arc de s à p Ou NULL si pas de chemin.

4.3.1.3 Chemin* plusCourtCheminRec (Graphe * *g*, int *s*, int *p*, int * *d*, Chemin * *c*, int * *v*)

4.4 src/graphe_algos.h File Reference

```
#include "graphe.h"
```

Functions

- Graphe * grapheEcart (Graphe *gCapa, Graphe *gFlot)
- Chemin * plusCourtChemin (Graphe *g, int s, int p)

4.4.1 Function Documentation

4.4.1.1 Graphe* grapheEcart (Graphe * *gCapa*, Graphe * *gFlot*)

gCapa : graphe des capacités gFlot : graphe du flot actuel Return : graphe d'écart

4.4.1.2 Chemin* plusCourtChemin (Graphe * *g*, int *s*, int *p*)

g : graphe orienté s : premier sommet p : dernier sommet Return : Plus court chemin en nombre d'arc de s à p Ou NULL si pas de chemin.

4.5 src/graphe_test.c File Reference

```
#include "graphe.h"
```

Functions

- void remplirTest1 (Graphe *g)

4.5.1 Function Documentation

4.5.1.1 void remplirTest1 (Graphe * *g*)

4.6 src/list.c File Reference

```
#include "list.h"
```

Functions

- List ∗ allocList ()
- void freeList (List ∗l)
- List ∗ copyList (List ∗l)
- void printList (FILE ∗f, List ∗l)
- void printListReverse (FILE ∗f, List ∗l)
- void pushHead (List ∗l, int num, int val)
- void pushTail (List ∗l, int num, int val)
- void insertNum (List ∗l, int num, int val)
- void popHead (List ∗l)
- void popTail (List ∗l)
- void popNum (List ∗l, int num)
- int valOfNum (List ∗l, int num)

4.6.1 Function Documentation

4.6.1.1 List∗ allocList ( )

4.6.1.2 List∗ copyList (List ∗ *l*)

4.6.1.3 void freeList (List ∗ *l*)

4.6.1.4 void insertNum (List ∗ *l*, int *num*, int *val*)

4.6.1.5 void popHead (List ∗ *l*)

4.6.1.6 void popNum (List ∗ *l*, int *num*)

4.6.1.7 void popTail (List ∗ *l*)

4.6.1.8 void printList (FILE ∗ *f*, List ∗ *l*)

4.6.1.9 void printListReverse (FILE ∗ *f*, List ∗ *l*)

4.6.1.10 void pushHead (List ∗ *l*, int *num*, int *val*)

4.6.1.11 void pushTail (List ∗ *l*, int *num*, int *val*)

4.6.1.12 int valOfNum (List ∗ *l*, int *num*)

4.7 src/list.h File Reference

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
```

Data Structures

- struct Item_s
- struct List

Defines

- #define num(e) e->_num
- #define val(e) e->_val
- #define next(e) e->_next
- #define prev(e) e->_prev
- #define begin(l) l->_begin
- #define end(l) l->_end
- #define nb(l) l->_nb

Typedefs

- typedef struct Item_s Item

Functions

- List * allocList ()
- void freeList (List *l)
- List * copyList (List *l)
- void printList (FILE *f, List *l)
- void printListReverse (FILE *f, List *l)
- void pushHead (List *l, int num, int val)
- void pushTail (List *l, int num, int val)
- void insertNum (List *l, int num, int val)
- void popHead (List *l)
- void popTail (List *l)
- void popNum (List *l, int num)
- int valOfNum (List *l, int num)

4.7.1 Define Documentation

4.7.1.1 #define begin(*l*) l->_begin

4.7.1.2 #define end(*l*) l->_end

4.7.1.3 #define nb(*l*) l->_nb

4.7.1.4 #define next(*e*) e->_next

4.7.1.5 #define num(*e*) e->_num

4.7.1.6 #define prev(*e*) e->_prev

4.7.1.7 #define val(*e*) e->_val

4.7.2 Typedef Documentation

4.7.2.1 typedef struct Item_s Item

4.7.3 Function Documentation

4.7.3.1 List* allocList ( )

4.7.3.2 List* copyList (List * *l*)

4.7.3.3 void freeList (List * *l*)

4.7.3.4 void insertNum (List * *l*, int *num*, int *val*)

4.7.3.5 void popHead (List * *l*)

4.7.3.6 void popNum (List * *l*, int *num*)

4.7.3.7 void popTail (List * *l*)

4.7.3.8 void printList (FILE * *f*, List * *l*)

4.7.3.9 void printListReverse (FILE * *f*, List * *l*)

4.7.3.10 void pushHead (List * *l*, int *num*, int *val*)

4.7.3.11 void pushTail (List * *l*, int *num*, int *val*)

4.7.3.12 int valOfNum (List * *l*, int *num*)

4.8 src/list_graph.c File Reference

```
#include "list_graph.h"
#include <stdio.h>
```

Defines

- #define in(lm) lm->_inNeighb
- #define out(lm) lm->_outNeighb
- #define nbVert(lm) lm->_nbVert
- #define nbEdge(lm) lm->_nbEdge

Functions

- int l_edgeVal (ListGraph *lg, int i, int j)

  Allows users to know value of the edge between i and j.
- void l_setEdge (ListGraph *lg, int i, int j, int val)

  Allows user to change the value of the edge between i and j.
- List * l_inNeighb (ListGraph *lg, int k)

  Allows user to get incoming neighbourhood of the selected node.
- List * l_outNeighb (ListGraph *lg, int k)

  Allows user to get outgoing neighbourhood of the selected node.
- ListGraph * l_copyGraph (ListGraph *lg)

  Allows the user to copy a list-represented graph.
- ListGraph * l_allocGraph (int n)

  Implements memory allocation for a list represented graph with n nodes.
- void l_freeGraph (ListGraph *lg)

  Implements the memory release of a list represented graph structure.
- void l_printListGraph (FILE *f, ListGraph *lg)

  Allows users to print on given stream.

4.8.1 Define Documentation

4.8.1.1 #define in(*lm*) lm->_inNeighb

4.8.1.2 #define nbEdge(*lm*) lm->_nbEdge

4.8.1.3 #define nbVert(*lm*) lm->_nbVert

4.8.1.4 #define out(*lm*) lm->_outNeighb

4.8.2 Function Documentation

4.8.2.1 ListGraph* l_allocGraph (int *n*)

Implements memory allocation for a list represented graph with n nodes.

Parameters

n	number of nodes in the graph

Returns

a pointer on allocated memory space

4.8.2.2 ListGraph* l_copyGraph (ListGraph * *lg*)

Allows the user to copy a list-represented graph.

Parameters

g	pointer on the graph which will be copied

Returns

pointer on the created graph

4.8.2.3 int l_edgeVal (ListGraph * *lg*, int *i*, int *j*)

Allows users to know value of the edge between i and j.

Parameters

g	pointer on the graph which is supposed to contain the edge
i	a node
j	another node

Returns

value of the edge, 0 if it doesn't exist

4.8.2.4 void l_freeGraph (ListGraph * *lg*)

Implements the memory release of a list represented graph structure.

Parameters

g	pointer on the graph to destroy

4.8.2.5 List* l_inNeighb (ListGraph * *lg*, int *k*)

Allows user to get incoming neighbourhood of the selected node.

Parameters

g	pointer on the graph which is supposed to contain the edge
k	node for which the incoming neighbourhood is desired

Returns

pointer on the desired Neghibourhood if exists, NULL else

4.8.2.6 List* l_outNeighb (ListGraph * *lg*, int *k*)

Allows user to get outgoing neighbourhood of the selected node.

Parameters

g	pointer on the graph which is supposed to contain the edge
k	node for which the outgoing neighbourhood is desired

Returns

pointer on the desired Neghibourhood if exists, NULL else

4.8.2.7 void l_printListGraph (FILE * *f*, ListGraph * *lg*)

Allows users to print on given stream.

Parameters

g	the graph to print

4.8.2.8 void l_setEdge (ListGraph * *lg*, int *i*, int *j*, int *val*)

Allows user to change the value of the edge between i and j.

Parameters

g	pointer on the graph which is supposed to contain the edge
i	a node
j	another node
val	new value of the edge

4.9 src/list_graph.h File Reference

```
#include "list.h"
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
```

Data Structures

- struct ListGraph

  Implementation of a graph data strucuture represented by incidence list.

Functions

- int l_edgeVal (ListGraph *lg, int i, int j)

  Allows users to know value of the edge between i and j.
- void l_setEdge (ListGraph *lg, int i, int j, int val)

  Allows user to change the value of the edge between i and j.
- List * l_inNeighb (ListGraph *lg, int k)

  Allows user to get incoming neighbourhood of the selected node.
- List * l_outNeighb (ListGraph *lg, int k)

  Allows user to get outgoing neighbourhood of the selected node.
- ListGraph * l_copyGraph (ListGraph *lg)

  Allows the user to copy a list-represented graph.
- ListGraph * l_allocGraph (int n)

  Implements memory allocation for a list represented graph with n nodes.
- void l_freeGraph (ListGraph *lg)

  Implements the memory release of a list represented graph structure.
- void l_printListGraph (FILE *f, ListGraph *lg)

  Allows users to print on given stream.

4.9.1 Function Documentation

4.9.1.1 ListGraph* l_allocGraph (int *n*)

Implements memory allocation for a list represented graph with n nodes.

Parameters

n	number of nodes in the graph

Returns

a pointer on allocated memory space

4.9.1.2 ListGraph* l_copyGraph (ListGraph * *lg*)

Allows the user to copy a list-represented graph.

Parameters

g	pointer on the graph which will be copied

Returns

pointer on the created graph

4.9.1.3 int l_edgeVal (ListGraph * *lg*, int *i*, int *j*)

Allows users to know value of the edge between i and j.

Parameters

g	pointer on the graph which is supposed to contain the edge
i	a node
j	another node

Returns

value of the edge, 0 if it doesn't exist

4.9.1.4 void l_freeGraph (ListGraph * *lg*)

Implements the memory release of a list represented graph structure.

Parameters

g	pointer on the graph to destroy

4.9.1.5 List* l_inNeighb (ListGraph * *lg*, int *k*)

Allows user to get incoming neighbourhood of the selected node.

Parameters

g	pointer on the graph which is supposed to contain the edge
k	node for which the incoming neighbourhood is desired

Returns

pointer on the desired Neghibourhood if exists, NULL else

4.9.1.6 List* l_outNeighb (ListGraph * *lg*, int *k*)

Allows user to get outgoing neighbourhood of the selected node.

Parameters

g	pointer on the graph which is supposed to contain the edge
k	node for which the outgoing neighbourhood is desired

Returns

pointer on the desired Neghibourhood if exists, NULL else

4.9.1.7 void l_printListGraph (FILE * *f*, ListGraph * *lg*)

Allows users to print on given stream.

Parameters

g	the graph to print

4.9.1.8 void l_setEdge (ListGraph * *lg*, int *i*, int *j*, int *val*)

Allows user to change the value of the edge between i and j.

Parameters

g	pointer on the graph which is supposed to contain the edge
i	a node
j	another node
val	new value of the edge

4.10 src/main.c File Reference

```
#include "graph.h"
#include <stdio.h>
```

Functions

- int main (int argc, char *argv[])

4.10.1 Function Documentation

4.10.1.1 int main (int *argc*, char * *argv[]*)

4.11 src/mat_graph.c File Reference

```
#include "mat_graph.h"
```

Defines

- #define mat(mg) mg->_mat
- #define nbVert(mg) mg->_nbVert
- #define nbEdge(mg) mg->_nbEdge

Functions

- int m_edgeVal (MatGraph *mg, int i, int j)

  Allows user to get value of the edge between i and j.
- void m_setEdge (MatGraph *mg, int i, int j, int val)

  Allows user to change the value of the edge (i,j)
- List * m_inNeighb (MatGraph *mg, int k)

  Allows user to get a copy of incoming neighbourhooh of a node.
- List * m_outNeighb (MatGraph *mg, int k)

  Allows user to get a copy of outgoing neighbourhooh of a node.
- MatGraph * m_copyGraph (MatGraph *mg)

  Allows user to create a new copy of desired matrix represented graph.
- MatGraph * m_allocGraph (int n)

  Implements memory allocation for a matrix-represented graph.
- void m_freeGraph (MatGraph *mg)

  Implements the memory release of a matrix-represented graph.

4.11.1 Define Documentation

4.11.1.1 #define mat(*mg*) mg->_mat

4.11.1.2 #define nbEdge(*mg*) mg->_nbEdge

4.11.1.3 #define nbVert(*mg*) mg->_nbVert

4.11.2 Function Documentation

4.11.2.1 MatGraph* m_allocGraph (int *n*)

Implements memory allocation for a matrix-represented graph.

Parameters

n	stands for the number of nodes which will be included in the graph

Returns

pointer on the new matrix-represented graph

4.11.2.2 MatGraph* m_copyGraph (MatGraph * *mg*)

Allows user to create a new copy of desired matrix represented graph.

Parameters

g	pointer on the matrix represented graph to copy

Returns

pointer on the new copy of graph

4.11.2.3 int m_edgeVal (MatGraph * *mg*, int *i*, int *j*)

Allows user to get value of the edge between i and j.

Parameters

g	pointer on a matrix-represented graph
i	a node
j	the other node

Returns

value of the (i,j) server

4.11.2.4 void m_freeGraph (MatGraph * *mg*)

Implements the memory release of a matrix-represented graph.

Parameters

g	pointer on the graph to destroy

4.11.2.5 List* m_inNeighb (MatGraph * *mg*, int *k*)

Allows user to get a copy of incoming neighbourhooh of a node.

Parameters

g	pointer on the graph which is supposed to contain the desired node
k	node for which the neighbourhood is asked

Returns

desired neighbourhood

4.11.2.6 List* m_outNeighb (MatGraph * *mg*, int *k*)

Allows user to get a copy of outgoing neighbourhooh of a node.

Parameters

g	pointer on the graph which is supposed to contain the desired node
k	node for which the neighbourhood is asked

Returns

desired neighbourhood

4.11.2.7 void m_setEdge (MatGraph * *mg*, int *i*, int *j*, int *val*)

Allows user to change the value of the edge (i,j)

Parameters

g	pointer on a matrix-represented graph
i	one of the extremity of the edge
j	the other extremity of the edge

4.12 src/mat_graph.h File Reference

```
#include "list.h"
#include <stdlib.h>
```

Data Structures

- struct MatGraph

  Implementation of a graph with help of an incidence matrix.

Functions

- int m_edgeVal (MatGraph *mg, int i, int j)

Allows user to get value of the edge between i and j.

- void m_setEdge (MatGraph *mg, int i, int j, int val)

  Allows user to change the value of the edge (i,j)
- List * m_inNeighb (MatGraph *mg, int k)

  Allows user to get a copy of incoming neighbourhooh of a node.
- List * m_outNeighb (MatGraph *mg, int k)

  Allows user to get a copy of outgoing neighbourhooh of a node.
- MatGraph * m_copyGraph (MatGraph *mg)

  Allows user to create a new copy of desired matrix represented graph.
- MatGraph * m_allocGraph (int n)

  Implements memory allocation for a matrix-represented graph.
- void m_freeGraph (MatGraph *mg)

  Implements the memory release of a matrix-represented graph.

4.12.1 Function Documentation

4.12.1.1 MatGraph* m_allocGraph (int *n*)

Implements memory allocation for a matrix-represented graph.

Parameters

n	stands for the number of nodes which will be included in the graph

Returns

pointer on the new matrix-represented graph

4.12.1.2 MatGraph* m_copyGraph (MatGraph * *mg*)

Allows user to create a new copy of desired matrix represented graph.

Parameters

g	pointer on the matrix represented graph to copy

Returns

pointer on the new copy of graph

4.12.1.3 int m_edgeVal (MatGraph * *mg*, int *i*, int *j*)

Allows user to get value of the edge between i and j.

Parameters

g	pointer on a matrix-represented graph
i	a node
j	the other node

Returns

value of the (i,j) server

4.12.1.4 void m_freeGraph (MatGraph * *mg*)

Implements the memory release of a matrix-represented graph.

Parameters

g	pointer on the graph to destroy

4.12.1.5 List* m_inNeighb (MatGraph * *mg*, int *k*)

Allows user to get a copy of incoming neighbourhooh of a node.

Parameters

g	pointer on the graph which is supposed to contain the desired node
k	node for which the neighbourhood is asked

Returns

desired neighbourhood

4.12.1.6 List* m_outNeighb (MatGraph * *mg*, int *k*)

Allows user to get a copy of outgoing neighbourhooh of a node.

Parameters

g	pointer on the graph which is supposed to contain the desired node
k	node for which the neighbourhood is asked

Returns

desired neighbourhood

4.12.1.7 void m_setEdge (MatGraph * *mg*, int *i*, int *j*, int *val*)

Allows user to change the value of the edge (i,j)

Parameters

g	pointer on a matrix-represented graph
i	one of the extremity of the edge
j	the other extremity of the edge

Index

_begin
    List, 7
_end
    List, 7
_inNeighb
    ListGraph, 7
_lg
    Graph, 5
_mat
    MatGraph, 8
_mg
    Graph, 5
_nb
    List, 7
_nbEdge
    ListGraph, 7
    MatGraph, 8
_nbVert
    ListGraph, 7
    MatGraph, 8
_next
    Item_s, 6
_num
    Item_s, 6
_outNeighb
    ListGraph, 7
_prev
    Item_s, 6
_val
    Item_s, 6

allocGraph
    graph.c, 10
    graph.h, 13
allocList
    list.c, 18
    list.h, 20
allocListGraph
    graph.c, 10
    graph.h, 14
allocMatGraph
    graph.c, 10
    graph.h, 14

begin
    list.h, 20

copyGraph
    graph.c, 10
    graph.h, 14
copyList
    list.c, 18
    list.h, 20

edgeVal
    graph.c, 11
    graph.h, 15
end
    list.h, 20

freeGraph
    graph.c, 11
    graph.h, 15
freeList
    list.c, 18
    list.h, 20

Graph, 5
    _lg, 5
    _mg, 5
graph.c
    allocGraph, 10
    allocListGraph, 10
    allocMatGraph, 10
    copyGraph, 10
    edgeVal, 11
    freeGraph, 11
    inNeighb, 11
    nbEdge, 11
    nbVert, 12
    outNeighb, 12
    printGraph, 12
    printGraphList, 12

randFill, 12
setEdge, 12
graph.h
allocGraph, 13
allocListGraph, 14
allocMatGraph, 14
copyGraph, 14
edgeVal, 15
freeGraph, 15
inNeighb, 15
lg, 13
mg, 13
nbEdge, 15
nbVert, 15
outNeighb, 15
printGraph, 16
printGraphList, 16
randFill, 16
setEdge, 16
graphe_algos.c
grapheEcart, 16
plusCourtChemin, 16
plusCourtCheminRec, 17
graphe_algos.h
grapheEcart, 17
plusCourtChemin, 17
graphe_test.c
remplirTest1, 17
grapheEcart
graphe_algos.c, 16
graphe_algos.h, 17

in
list_graph.c, 21
inNeighb
graph.c, 11
graph.h, 15
insertNum
list.c, 18
list.h, 20
Item
list.h, 20
Item_s, 6
_next, 6
_num, 6
_prev, 6
_val, 6

l_allocGraph
list_graph.c, 21
list_graph.h, 24
l_copyGraph
list_graph.c, 22
list_graph.h, 25
l_edgeVal
list_graph.c, 22
list_graph.h, 25
l_freeGraph
list_graph.c, 22
list_graph.h, 25
l_inNeighb
list_graph.c, 22
list_graph.h, 25
l_outNeighb
list_graph.c, 23
list_graph.h, 25
l_printListGraph
list_graph.c, 23
list_graph.h, 26
l_setEdge
list_graph.c, 23
list_graph.h, 26
lg
graph.h, 13
List, 6
_begin, 7
_end, 7
_nb, 7
list.c
allocList, 18
copyList, 18
freeList, 18
insertNum, 18
popHead, 18
popNum, 18
popTail, 18
printList, 18
printListReverse, 18
pushHead, 18
pushTail, 18
valOfNum, 18
list.h
allocList, 20
begin, 20
copyList, 20
end, 20
freeList, 20
insertNum, 20
Item, 20
nb, 20

next, 20
num, 20
popHead, 20
popNum, 20
popTail, 20
prev, 20
printList, 20
printListReverse, 20
pushHead, 20
pushTail, 20
val, 20
valOfNum, 20
list_graph.c
in, 21
l_allocGraph, 21
l_copyGraph, 22
l_edgeVal, 22
l_freeGraph, 22
l_inNeighb, 22
l_outNeighb, 23
l_printListGraph, 23
l_setEdge, 23
nbEdge, 21
nbVert, 21
out, 21
list_graph.h
l_allocGraph, 24
l_copyGraph, 25
l_edgeVal, 25
l_freeGraph, 25
l_inNeighb, 25
l_outNeighb, 25
l_printListGraph, 26
l_setEdge, 26
ListGraph, 7
_inNeighb, 7
_nbEdge, 7
_nbVert, 7
_outNeighb, 7

m_allocGraph
mat_graph.c, 27
mat_graph.h, 30
m_copyGraph
mat_graph.c, 28
mat_graph.h, 30
m_edgeVal
mat_graph.c, 28
mat_graph.h, 30
m_freeGraph
mat_graph.c, 28
mat_graph.h, 31
m_inNeighb
mat_graph.c, 28
mat_graph.h, 31
m_outNeighb
mat_graph.c, 29
mat_graph.h, 31
m_setEdge
mat_graph.c, 29
mat_graph.h, 31
main
main.c, 26
main.c
main, 26
mat
mat_graph.c, 27
mat_graph.c
m_allocGraph, 27
m_copyGraph, 28
m_edgeVal, 28
m_freeGraph, 28
m_inNeighb, 28
m_outNeighb, 29
m_setEdge, 29
mat, 27
nbEdge, 27
nbVert, 27
mat_graph.h
m_allocGraph, 30
m_copyGraph, 30
m_edgeVal, 30
m_freeGraph, 31
m_inNeighb, 31
m_outNeighb, 31
m_setEdge, 31
MatGraph, 8
_mat, 8
_nbEdge, 8
_nbVert, 8
mg
graph.h, 13

nb
list.h, 20
nbEdge
graph.c, 11
graph.h, 15
list_graph.c, 21
mat_graph.c, 27

nbVert
graph.c, 12
graph.h, 15
list_graph.c, 21
mat_graph.c, 27
next
list.h, 20
num
list.h, 20

out
list_graph.c, 21
outNeighb
graph.c, 12
graph.h, 15

plusCourtChemin
graphe_algos.c, 16
graphe_algos.h, 17
plusCourtCheminRec
graphe_algos.c, 17
popHead
list.c, 18
list.h, 20
popNum
list.c, 18
list.h, 20
popTail
list.c, 18
list.h, 20
prev
list.h, 20
printGraph
graph.c, 12
graph.h, 16
printGraphList
graph.c, 12
graph.h, 16
printList
list.c, 18
list.h, 20
printListReverse
list.c, 18
list.h, 20
pushHead
list.c, 18
list.h, 20
pushTail
list.c, 18
list.h, 20

randFill
graph.c, 12
graph.h, 16
remplirTest1
graphe_test.c, 17

setEdge
graph.c, 12
graph.h, 16
src/graph.c, 9
src/graph.h, 12
src/graphe_algos.c, 16
src/graphe_algos.h, 17
src/graphe_test.c, 17
src/list.c, 18
src/list.h, 19
src/list_graph.c, 21
src/list_graph.h, 24
src/main.c, 26
src/mat_graph.c, 27
src/mat_graph.h, 29

val
list.h, 20
valOfNum
list.c, 18
list.h, 20