

Graph

Generated by Doxygen 1.6.1

Fri May 14 22:05:52 2010

Contents

1	Todo List	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	grafo Struct Reference	7
4.1.1	Member Function Documentation	8
4.1.1.1	aresta	8
4.1.1.2	bellman_ford	9
4.1.1.3	bfs	9
4.1.1.4	bfs_fluxo	9
4.1.1.5	dfs	10
4.1.1.6	dijkstra	10
4.1.1.7	edmonds_karp	10
4.1.1.8	inic	11
4.1.1.9	inverso	11
4.1.1.10	print	11
4.1.2	Member Data Documentation	12
4.1.2.1	adj	12
4.1.2.2	busca	12
4.1.2.3	cap	12
4.1.2.4	dest	12
4.1.2.5	dist	12
4.1.2.6	fluxo	12
4.1.2.7	M	12

4.1.2.8	nadj	12
4.1.2.9	nar	12
4.1.2.10	nvt	12
4.1.2.11	peso	12
4.1.2.12	prev	13
4.1.2.13	prev_aresta	13
4.1.2.14	prev_busca	13
4.1.2.15	vis	13
5	File Documentation	15
5.1	/home/troll/prog/algorithmc/algorithms/graph/src/graph.cpp File Reference	15
5.1.1	Typedef Documentation	16
5.1.1.1	Weight	16
5.1.2	Variable Documentation	16
5.1.2.1	AR	16
5.1.2.2	INFINITY	16
5.1.2.3	VT	16

Chapter 1

Todo List

Member `grafo::dijkstra(int src, int dst)` Fazer o método gravar as arestas usadas no vetor `prev_aresta`.

Member `grafo::print()` Fazer o método imprimir `c` e `f`, e colocar uma opção de imprimir as arestas inseridas para fluxos ou não.

Chapter 2

Class Index

2.1 Class List

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

grafo	7
---------------------------------	---

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

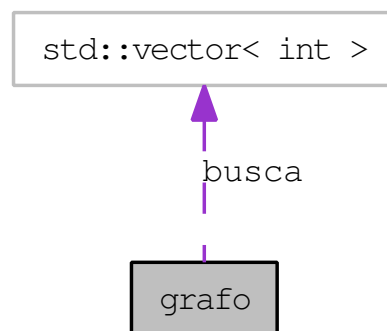
/home/troll/prog/algoricmc/algorithms/graph/src/[graph.cpp](#) 15

Chapter 4

Class Documentation

4.1 grafo Struct Reference

Collaboration diagram for grafo:



Public Member Functions

- void `inic` (int n)
Inicializador do `grafo`.
- void `aresta` (int src, int dst, `Weight` p, int c)
Insere uma aresta no `grafo`.
- void `print` ()
Imprime o `grafo`.
- void `dfs` (int src)
Busca em profundidade.
- void `bfs` (int n)
Busca em largura.
- void `dijkstra` (int src, int dst)

Algoritmo de Dijkstra.

- int [bellman_ford](#) (int src)

Bellman-Ford.

- int [edmonds_karp](#) (int src, int dst)

Algoritmo de fluxo máximo de Edmonds-Karp.

- int [bfs_fluxo](#) (int src, int dst)

Busca em largura, modificada para o ser usada por Edmonds-Karp.

- int [inverso](#) (int a)

Retorna a aresta inversa.

Public Attributes

- int [dest](#) [AR]
- int [fluxo](#) [AR]
- int [cap](#) [AR]
- int [M](#) [VT]
- int [adj](#) [VT][VT]
- int [nadj](#) [VT]
- int [nvt](#)
- int [nar](#)
- int [vis](#) [VT]
- int [dist](#) [VT]
- int [prev](#) [VT]
- [Weight](#) [peso](#) [AR]
- vector< int > [busca](#)
- int [prev_busca](#) [VT]
- int [prev_aresta](#) [VT]

4.1.1 Member Function Documentation

4.1.1.1 void grafo::aresta (int src, int dst, Weight p = 0, int c = 0)

Insere uma aresta no [grafo](#).

Parameters:

- ← *src* Vértice origem.
- ← *dst* Vértice destino.
- ← *p* Peso da aresta. Default 0.
- ← *c* Capacidade da aresta. Default 0.

Note:

Para usar os algoritmos de fluxo, é necessário descomentar uma parte do código desse método. A parte está indicada em seu código. Isso faz com que o [grafo](#) insira a aresta desejada e, além dela, uma aresta inversa, usada internamente pelos algoritmos de fluxo, para que o fluxo "possa ser mandado de volta".

4.1.1.2 int grafo::bellman_ford (int *src*)

Bellman-Ford. Acha a menor distância até cada nó. Funciona em grafos com arestas cujos pesos são negativos.

Parameters:

- ← *src* Nó origem.
- *prev* Predecessor de cada nó.
- *dist* Distância até cada nó.

Returns:

1 se existe pelo menos um ciclo negativo no [grafo](#), 0 caso contrário.

Note:

Limp automaticamente os vetores *prev* e *busca* antes de executar o algoritmo.

4.1.1.3 void grafo::bfs (int *n*)

Busca em largura. Veja os comentários da busca em profundidade. Os mesmos se aplicam aqui.

Note:

Não limpa *vis*, *prev_busca*, *prev_aresta* e o vector *busca* internamente.

4.1.1.4 int grafo::bfs_fluxo (int *src*, int *dst*)

Busca em largura, modificada para o ser usada por Edmonds-Karp.

Parameters:

- ← *src* Nó origem.
- ← *dst* Nó destino.
- *vis* Nós que podem ser alcançados nesse [grafo](#) residual. Determina o Min-Cut ao final do algoritmo de Edmonds-Karp.

Returns:

Capacidade do caminho encontrado.

Note:

Limpa o vetor *vis*. Atualiza os vetores *prev_busca* e *prev_aresta* para que sejam usados por Edmonds-Karp. Esses vetores não precisam ser limpadados.

Here is the caller graph for this function:



4.1.1.5 void grafo::dfs (int *src*)

Busca em profundidade.

Parameters:

- ← *src* Nó inicial.
- *busca* Ordem em que os nós são visitados pela primeira vez.
- *prev_busca* Predecessor de cada nó.
- *prev_aresta* Aresta usada pelo predecessor de cada nó para chegar nele.
- *vis* Nós que foram visitados por essa busca.

Note:

Não limpa vis, prev_busca, prev_aresta e o vector busca internamente.

4.1.1.6 void grafo::dijkstra (int *src*, int *dst* = -1)

Algoritmo de Dijkstra. O algoritmo para quando encontra a menor distância até o nó destino, ou acha a menor distância até cada nó, caso *dst* == -1.

Parameters:

- ← *src* Nó origem.
- ← *dst* Nó destino. Default -1.
- *prev* Predecessor de cada nó. -1 indica que não existe predecessor.
- *vis* Nós visitados.
- *dist* Distância total até cada nó, partindo-se de *src*.

Note:

Não limpa o vetor vis, e limpa prev e dist antes de executar o algoritmo.

Todo

Fazer o método gravar as arestas usadas no vetor prev_aresta.

4.1.1.7 int grafo::edmonds_karp (int *src*, int *dst*)

Algoritmo de fluxo máximo de Edmonds-Karp. Acha o fluxo máximo do [grafo](#). Para isso, realiza várias buscas em largura para encontrar caminhos que adicionam fluxo.

Parameters:

- ← *src* Nó origem.
- ← *dst* Nó final.
- *fluxo* Fluxo passando em cada aresta.
- *vis* Determina o Min-Cut do [grafo](#).

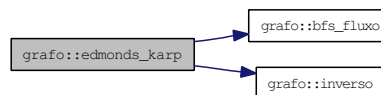
Returns:

Valor do fluxo máximo no [grafo](#).

Note:

O método limpa o vetor fluxo antes de executar o algoritmo. Utiliza os vetores `prev_busca` e `prev_aresta`, mas como parte da implementação, por isso não são marcados como parâmetros de saída aqui. `vis` não é alterado por esse método diretamente, e sim pela busca para fluxos.

Here is the call graph for this function:

**4.1.1.8 void grafo::inic (int *n* = 0)**

Inicializador do [grafo](#). Após esse método ser executado, o [grafo](#) tem *n* nós e nenhuma aresta.

Parameters:

← *n* Numero de vertices que o [grafo](#) tem. Deve ser menor ou igual a VT. Default 0.

4.1.1.9 int grafo::inverso (int *a*) [inline]

Retorna a aresta inversa. Método usado por algoritmos de fluxo. Retorna a aresta inversa de uma dada aresta.

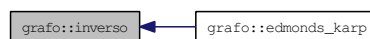
Parameters:

← *a* Índice de uma aresta.

Returns:

Índice da aresta inversa.

Here is the caller graph for this function:

**4.1.1.10 void grafo::print ()**

Imprime o [grafo](#). Imprime na forma {dst, p, c, f}, onde p é o peso da aresta, c, sua capacidade e f, o fluxo nela.

Todo

Fazer o método imprimir c e f, e colocar uma opção de imprimir as arestas inseridas para fluxos ou não.

4.1.2 Member Data Documentation

4.1.2.1 `int grafo::adj[VT][VT]`

Tipo de matriz de adjacência. Armazena o índice de cada aresta.

4.1.2.2 `vector<int> grafo::busca`

Armazena o resultado de uma busca.

4.1.2.3 `int grafo::cap[AR]`

Capacidade de cada aresta.

4.1.2.4 `int grafo::dest[AR]`

Vértices de destino de uma dada aresta.

4.1.2.5 `int grafo::dist[VT]`

Distância até cada vértice. Usado por Dijkstra e Bellman-Ford.

4.1.2.6 `int grafo::fluxo[AR]`

Fluxo em cada aresta.

4.1.2.7 `int grafo::M[VT]`

Usado para achar a capacidade dos caminhos na busca em largura de fluxos.

4.1.2.8 `int grafo::nadj[VT]`

Número de arestas saindo de cada vértice.

4.1.2.9 `int grafo::nar`

Número de arestas no [grafo](#).

4.1.2.10 `int grafo::nvt`

Número de vertices no [grafo](#). Esse numero dever ser menor ou igual a VT.

4.1.2.11 `Weight grafo::peso[AR]`

Peso de cada aresta.

4.1.2.12 int grafo::prev[VT]

Predecessor de cada vértice. Usado por Dijkstra e Bellman-Ford.

4.1.2.13 int grafo::prev_aresta[VT]

Aresta usada para chegar aos vértices em uma busca.

4.1.2.14 int grafo::prev_busca[VT]

Predecessores de cada vértice em uma busca.

4.1.2.15 int grafo::vis[VT]

Nós visitados. 1 indica que o nó foi visitado e 0 indica que não foi nos métodos que o utilizam.

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

- /home/troll/prog/algorithmc/algorithms/graph/src/[graph.cpp](#)

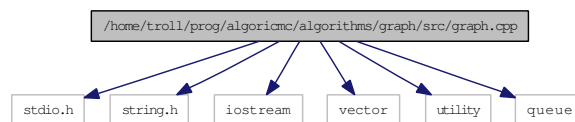
Chapter 5

File Documentation

5.1 /home/troll/prog/algoricmc/algorithms/graph/src/graph.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <vector>
#include <utility>
#include <queue>
```

Include dependency graph for graph.cpp:



Classes

- struct [grafo](#)

Typedefs

- typedef int [Weight](#)

Variables

- const int [VT](#) = 10
- const int [AR](#) = [VT](#) * [VT](#)
- const int [INFINITY](#) = 99999

5.1.1 Typedef Documentation

5.1.1.1 typedef int Weight

5.1.2 Variable Documentation

5.1.2.1 const int AR = VT * VT

Numero máximo de arestas que o [grafo](#) pode comportar.

5.1.2.2 const int INFINITY = 99999

Valor de infinito, usado por Dijkstra e outros algoritmos.

5.1.2.3 const int VT = 10

Numero máximo de vértices que o [grafo](#) pode comportar. Pode estourar a memória estática.

Index

/home/troll/prog/algorithmc/algorithmc/graph/src/graph.cppdfs, [9](#)
[15](#)

adj
 [grafo, 12](#)

AR
 [graph.cpp, 16](#)

aresta
 [grafo, 8](#)

bellman_ford
 [grafo, 8](#)

bfs
 [grafo, 9](#)

bfs_fluxo
 [grafo, 9](#)

busca
 [grafo, 12](#)

cap
 [grafo, 12](#)

dest
 [grafo, 12](#)

dfs
 [grafo, 9](#)

dijkstra
 [grafo, 10](#)

dist
 [grafo, 12](#)

edmonds_karp
 [grafo, 10](#)

fluxo
 [grafo, 12](#)

grafo, [7](#)
 [adj, 12](#)
 [aresta, 8](#)
 [bellman_ford, 8](#)
 [bfs, 9](#)
 [bfs_fluxo, 9](#)
 [busca, 12](#)
 [cap, 12](#)
 [dest, 12](#)
 [dijkstra, 10](#)
 [dist, 12](#)
 [edmonds_karp, 10](#)
 [fluxo, 12](#)
 [inic, 11](#)
 [inverso, 11](#)
 [M, 12](#)
 [nadj, 12](#)
 [nar, 12](#)
 [nvt, 12](#)
 [peso, 12](#)
 [prev, 12](#)
 [prev_aresta, 13](#)
 [prev_busca, 13](#)
 [print, 11](#)
 [vis, 13](#)
 graph.cpp
 AR, [16](#)
 INFINITY, [16](#)
 VT, [16](#)
 Weight, [16](#)
 INFINITY
 [graph.cpp, 16](#)
 inic
 [grafo, 11](#)
 inverso
 [grafo, 11](#)
 M
 [grafo, 12](#)
 nadj
 [grafo, 12](#)
 nar
 [grafo, 12](#)
 nvt
 [grafo, 12](#)
 peso
 [grafo, 12](#)
 prev
 [grafo, 12](#)
 prev_aresta
 [grafo, 13](#)

prev_busca
 grafo, [13](#)
print
 grafo, [11](#)

vis
 grafo, [13](#)
VT
 graph.cpp, [16](#)

Weight
 graph.cpp, [16](#)