

Trabalho Prático - Fase 2

Gerado por Doxygen 1.13.2

1 Índice das estruturas de dados	1
1.1 Estruturas de dados	1
2 Índice dos ficheiros	3
2.1 Lista de ficheiros	3
3 Documentação da estruturas de dados	5
3.1 Referência à estrutura Grafo	5
3.1.1 Descrição detalhada	5
3.1.2 Documentação dos campos e atributos	5
3.1.2.1 tamanho	5
3.1.2.2 vertices	5
3.2 Referência à estrutura Ligacao	6
3.2.1 Descrição detalhada	6
3.2.2 Documentação dos campos e atributos	6
3.2.2.1 destino	6
3.2.2.2 seguinte	6
3.3 Referência à estrutura Vertice	6
3.3.1 Descrição detalhada	7
3.3.2 Documentação dos campos e atributos	7
3.3.2.1 freq	7
3.3.2.2 ligacoes	7
3.3.2.3 x	7
3.3.2.4 y	7
3.4 Referência à estrutura VerticeSimples	7
3.4.1 Descrição detalhada	8
3.4.2 Documentação dos campos e atributos	8
3.4.2.1 freq	8
3.4.2.2 x	8
3.4.2.3 y	8
4 Documentação do ficheiro	9
4.1 Referência ao ficheiro Grafos/grafos.c	9
4.1.1 Documentação das funções	10
4.1.1.1 adicionarAntena()	10
4.1.1.2 bft()	10
4.1.1.3 carregarGrafoBinario()	10
4.1.1.4 criarLigacao()	11
4.1.1.5 dft()	11
4.1.1.6 encontrarAntena()	11
4.1.1.7 guardarGrafoBinario()	12
4.1.1.8 iniciarGrafo()	12
4.1.1.9 ligarAntenasIguais()	12

4.1.1.10 limparLigacoes()	13
4.1.1.11 mostrarAntenas()	13
4.1.1.12 mostrarIntersecoes()	13
4.1.1.13 mostrarTodosCaminhos()	14
4.1.1.14 visitarDFT()	14
4.2 Referência ao ficheiro Grafos/grafos.h	15
4.2.1 Documentação das macros	16
4.2.1.1 MAX_VERTICES	16
4.2.2 Documentação dos tipos	16
4.2.2.1 Grafo	16
4.2.2.2 Ligacao	16
4.2.2.3 Vertice	16
4.2.2.4 VerticeSimples	16
4.2.3 Documentação das funções	16
4.2.3.1 adicionarAntena()	16
4.2.3.2 bft()	17
4.2.3.3 carregarGrafoBinario()	18
4.2.3.4 dft()	18
4.2.3.5 encontrarAntena()	19
4.2.3.6 guardarGrafoBinario()	19
4.2.3.7 iniciarGrafo()	20
4.2.3.8 ligarAntenasIguais()	20
4.2.3.9 limparLigacoes()	21
4.2.3.10 mostrarAntenas()	21
4.2.3.11 mostrarIntersecoes()	22
4.2.3.12 mostrarTodosCaminhos()	23
4.3 grafos.h	23
4.4 Referência ao ficheiro main.c	24
4.4.1 Documentação das funções	24
4.4.1.1 main()	24
Índice	25

Capítulo 1

Índice das estruturas de dados

1.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Grafo	Representa um grafo com lista fixa de vértices	5
Ligacao	Representa uma ligação (aresta) entre dois vértices do grafo	6
Vertice	Representa um vértice (antena) no grafo	6
VerticeSimples	Estrutura simples usada para guardar e carregar vértices em ficheiros binários	7

Capítulo 2

Índice dos ficheiros

2.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

main.c	24
Grafos/ grafos.c	9
Grafos/ grafos.h	15

Capítulo 3

Documentação da estruturas de dados

3.1 Referência à estrutura Grafo

Representa um grafo com lista fixa de vértices.

```
#include <grafos.h>
```

Campos de Dados

- [Vertice vertices](#) [MAX_VERTICES]
Lista de antenas (vértices)
- int [tamanho](#)
Número atual de vértices.

3.1.1 Descrição detalhada

Representa um grafo com lista fixa de vértices.

3.1.2 Documentação dos campos e atributos

3.1.2.1 tamanho

```
int tamanho
```

Número atual de vértices.

3.1.2.2 vertices

```
Vertice vertices[MAX_VERTICES]
```

Lista de antenas (vértices)

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- Grafos/[grafos.h](#)

3.2 Referência à estrutura Ligacao

Representa uma ligação (aresta) entre dois vértices do grafo.

```
#include <grafos.h>
```

Campos de Dados

- int [destino](#)
Índice do vértice de destino.
- struct [Ligacao](#) * [seguinte](#)
Próxima ligação.

3.2.1 Descrição detalhada

Representa uma ligação (aresta) entre dois vértices do grafo.

3.2.2 Documentação dos campos e atributos

3.2.2.1 destino

```
int destino
```

Índice do vértice de destino.

3.2.2.2 seguinte

```
struct Ligacao* seguinte
```

Próxima ligação.

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- Grafos/[grafos.h](#)

3.3 Referência à estrutura Vertice

Representa um vértice (antena) no grafo.

```
#include <grafos.h>
```

Campos de Dados

- char `freq`
Frequência da antena.
- int `x`
- int `y`
Coordenadas da antena.
- `Ligacao *` `ligacoes`
Lista de ligações para outras antenas.

3.3.1 Descrição detalhada

Representa um vértice (antena) no grafo.

3.3.2 Documentação dos campos e atributos

3.3.2.1 `freq`

```
char freq
```

Frequência da antena.

3.3.2.2 `ligacoes`

```
Ligacao* ligacoes
```

Lista de ligações para outras antenas.

3.3.2.3 `x`

```
int x
```

3.3.2.4 `y`

```
int y
```

Coordenadas da antena.

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- Grafos/[grafos.h](#)

3.4 Referência à estrutura VerticeSimples

Estrutura simples usada para guardar e carregar vértices em ficheiros binários.

```
#include <grafos.h>
```

Campos de Dados

- char [freq](#)
Frequência.
- int [x](#)
Coordenada X.
- int [y](#)
Coordenada Y.

3.4.1 Descrição detalhada

Estrutura simples usada para guardar e carregar vértices em ficheiros binários.

3.4.2 Documentação dos campos e atributos

3.4.2.1 freq

```
char freq
```

Frequência.

3.4.2.2 x

```
int x
```

Coordenada X.

3.4.2.3 y

```
int y
```

Coordenada Y.

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- Grafos/[grafos.h](#)

Capítulo 4

Documentação do ficheiro

4.1 Referência ao ficheiro Grafos/grafos.c

```
#include <stdio.h>
#include <stdlib.h>
#include "grafos.h"
```

Funções

- int [iniciarGrafo](#) ([Grafo](#) *g)
Inicializa o grafo, definindo tamanho 0 e ligações a NULL.
- int [adicionarAntena](#) ([Grafo](#) *g, char freq, int x, int y)
Adiciona uma antena ao grafo.
- int [criarLigacao](#) ([Grafo](#) *g, int origem, int destino)
Cria uma ligação entre duas antenas.
- int [ligarAntenasIguais](#) ([Grafo](#) *g)
Liga todas as antenas que têm a mesma frequência.
- int [limparLigacoes](#) ([Grafo](#) *g)
Liberta todas as ligações (arestas) existentes no grafo.
- int [mostrarAntenas](#) ([Grafo](#) *g)
Mostra todas as antenas do grafo.
- int [encontrarAntena](#) ([Grafo](#) *g, int x, int y)
Procura o índice de uma antena pelas coordenadas.
- int [visitarDFT](#) ([Grafo](#) *g, int atual, int *visitado)
Função auxiliar de DFT (profundidade).
- int [dft](#) ([Grafo](#) *g, int inicio)
Executa DFT (Depth-First Traversal) a partir de uma antena.
- int [bft](#) ([Grafo](#) *g, int inicio)
Executa BFT (Breadth-First Traversal) a partir de uma antena.
- int [mostrarTodosCaminhos](#) ([Grafo](#) *g, int origem, int destino, int *caminho, int comprimento, int *visitado)
Mostra todos os caminhos possíveis entre duas antenas.
- int [mostrarIntersecoes](#) ([Grafo](#) *g, char freqA, char freqB)
Verifica interseções geométricas entre ligações de duas frequências distintas (A-A vs B-B).
- int [guardarGrafoBinario](#) ([Grafo](#) *g, const char *nome)
Guarda o grafo num ficheiro binário.
- int [carregarGrafoBinario](#) ([Grafo](#) *g, const char *nome)
Carrega um grafo de um ficheiro binário.

4.1.1 Documentação das funções

4.1.1.1 adicionarAntena()

```
int adicionarAntena (  
    Grafo * g,  
    char freq,  
    int x,  
    int y)
```

Adiciona uma antena ao grafo.

Adiciona uma nova antena ao grafo.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>freq</i>	Frequência da antena
<i>x</i>	Coordenada X
<i>y</i>	Coordenada Y

Retorna

Índice da antena adicionada ou -1 se falhar

4.1.1.2 bft()

```
int bft (  
    Grafo * g,  
    int inicio)
```

Executa BFT (Breadth-First Traversal) a partir de uma antena.

Realiza percurso em largura (BFT) a partir de uma antena.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>inicio</i>	Índice da antena de partida

Retorna

1 após execução

4.1.1.3 carregarGrafoBinario()

```
int carregarGrafoBinario (  
    Grafo * g,  
    const char * nome)
```

Carrega um grafo de um ficheiro binário.

Carrega um grafo a partir de um ficheiro binário.

Parâmetros

<i>g</i>	Grafo
<i>nome</i>	Nome do ficheiro

Retorna

1 se for bem sucedido

4.1.1.4 criarLigacao()

```
int criarLigacao (  
    Grafo * g,  
    int origem,  
    int destino)
```

Cria uma ligação entre duas antenas.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>origem</i>	Índice do vértice de origem
<i>destino</i>	Índice do vértice de destino

Retorna

1 se for bem sucedido, 0 caso contrário

4.1.1.5 dft()

```
int dft (  
    Grafo * g,  
    int inicio)
```

Executa DFT (Depth-First Traversal) a partir de uma antena.

Realiza percurso em profundidade (DFT) a partir de uma antena.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>inicio</i>	Índice da antena de partida

Retorna

1 após execução

4.1.1.6 encontrarAntena()

```
int encontrarAntena (  
    Grafo * g,  
    int x,  
    int y)
```

Procura o índice de uma antena pelas coordenadas.

Encontra o índice de uma antena com coordenadas específicas.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>x</i>	Coordenada X
<i>y</i>	Coordenada Y

Retorna

Índice da antena ou -1 se não existir

4.1.1.7 guardarGrafoBinario()

```
int guardarGrafoBinario (  
    Grafo * g,  
    const char * nome)
```

Guarda o grafo num ficheiro binário.

Guarda o estado atual do grafo num ficheiro binário.

Parâmetros

<i>g</i>	Grafo
<i>nome</i>	Nome do ficheiro

Retorna

1 se for bem sucedido

4.1.1.8 iniciarGrafo()

```
int iniciarGrafo (  
    Grafo * g)
```

Inicializa o grafo, definindo tamanho 0 e ligações a NULL.

Inicializa um grafo vazio.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 se for bem sucedido

4.1.1.9 ligarAntenasIguais()

```
int ligarAntenasIguais (  
    Grafo * g)
```

Liga todas as antenas que têm a mesma frequência.

Liga entre si todas as antenas que têm a mesma frequência.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 se for bem sucedido

4.1.1.10 limparLigacoes()

```
int limparLigacoes (  
    Grafo * g)
```

Liberta todas as ligações (arestas) existentes no grafo.

Liberta todas as ligações atuais do grafo, evitando duplicações.

Esta função percorre todos os vértices e liberta a memória das suas listas de ligações, deixando cada lista vazia. É útil para evitar duplicações de ligações quando se chamam funções como [ligarAntenasIguais\(\)](#) mais de uma vez.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 após a limpeza

4.1.1.11 mostrarAntenas()

```
int mostrarAntenas (  
    Grafo * g)
```

Mostra todas as antenas do grafo.

Mostra na consola todas as antenas registadas no grafo.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 após mostrar

4.1.1.12 mostrarIntersecoes()

```
int mostrarIntersecoes (  
    Grafo * g,  
    char freqA,  
    char freqB)
```

Verifica interseções geométricas entre ligações de duas frequências distintas (A-A vs B-B).

Mostra as interseções entre ligações de frequências distintas (A-A vs B-B).

Considera como interseção o cruzamento entre arestas da frequência A com arestas da frequência B, se os segmentos (x1,y1)-(x2,y2) se cruzam no espaço.

Parâmetros

<i>g</i>	Grafo
<i>freqA</i>	Primeira frequência (ex: 'A')
<i>freqB</i>	Segunda frequência (ex: 'B')

Retorna

1 se correr com sucesso

4.1.1.13 mostrarTodosCaminhos()

```
int mostrarTodosCaminhos (  
    Grafo * g,  
    int origem,  
    int destino,  
    int * caminho,  
    int comprimento,  
    int * visitado)
```

Mostra todos os caminhos possíveis entre duas antenas.

Parâmetros

<i>g</i>	Grafo
<i>origem</i>	Índice da antena de origem
<i>destino</i>	Índice da antena de destino
<i>caminho</i>	Vetor auxiliar de caminho
<i>comprimento</i>	Comprimento atual do caminho
<i>visitado</i>	Vetor de visitados

Retorna

1

4.1.1.14 visitarDFT()

```
int visitarDFT (  
    Grafo * g,  
    int atual,  
    int * visitado)
```

Função auxiliar de DFT (profundidade).

Parâmetros

<i>g</i>	Grafo
<i>atual</i>	Índice atual
<i>visitado</i>	Vetor de visitados

Retorna

1

4.2 Referência ao ficheiro Grafos/grafos.h

Estruturas de Dados

- struct [Ligacao](#)
Representa uma ligação (aresta) entre dois vértices do grafo.
- struct [Vertice](#)
Representa um vértice (antena) no grafo.
- struct [Grafo](#)
Representa um grafo com lista fixa de vértices.
- struct [VerticeSimples](#)
Estrutura simples usada para guardar e carregar vértices em ficheiros binários.

Macros

- #define [MAX_VERTICES](#) 100
Número máximo de antenas (vértices) no grafo.

Definições de tipos

- typedef struct Ligacao [Ligacao](#)
Representa uma ligação (aresta) entre dois vértices do grafo.
- typedef struct Vertice [Vertice](#)
Representa um vértice (antena) no grafo.
- typedef struct Grafo [Grafo](#)
Representa um grafo com lista fixa de vértices.
- typedef struct VerticeSimples [VerticeSimples](#)
Estrutura simples usada para guardar e carregar vértices em ficheiros binários.

Funções

- int [iniciarGrafo](#) ([Grafo](#) *g)
Inicializa um grafo vazio.
- int [adicionarAntena](#) ([Grafo](#) *g, char freq, int x, int y)
Adiciona uma nova antena ao grafo.
- int [ligarAntenasIguais](#) ([Grafo](#) *g)
Liga entre si todas as antenas que têm a mesma frequência.
- int [limparLigacoes](#) ([Grafo](#) *g)
Liberta todas as ligações atuais do grafo, evitando duplicações.
- int [mostrarAntenas](#) ([Grafo](#) *g)
Mostra na consola todas as antenas registadas no grafo.
- int [encontrarAntena](#) ([Grafo](#) *g, int x, int y)
Encontra o índice de uma antena com coordenadas específicas.
- int [dft](#) ([Grafo](#) *g, int inicio)
Realiza percurso em profundidade (DFT) a partir de uma antena.
- int [bft](#) ([Grafo](#) *g, int inicio)
Realiza percurso em largura (BFT) a partir de uma antena.
- int [mostrarTodosCaminhos](#) ([Grafo](#) *g, int origem, int destino, int *caminho, int comprimento, int *visitado)
Mostra todos os caminhos possíveis entre duas antenas.
- int [mostrarIntersecoes](#) ([Grafo](#) *g, char freqA, char freqB)
Mostra as interseções entre ligações de frequências distintas (A-A vs B-B).
- int [guardarGrafoBinario](#) ([Grafo](#) *g, const char *nome)
Guarda o estado atual do grafo num ficheiro binário.
- int [carregarGrafoBinario](#) ([Grafo](#) *g, const char *nome)
Carrega um grafo a partir de um ficheiro binário.

4.2.1 Documentação das macros

4.2.1.1 MAX_VERTICES

```
#define MAX_VERTICES 100
```

Número máximo de antenas (vértices) no grafo.

4.2.2 Documentação dos tipos

4.2.2.1 Grafo

```
typedef struct Grafo Grafo
```

Representa um grafo com lista fixa de vértices.

4.2.2.2 Ligacao

```
typedef struct Ligacao Ligacao
```

Representa uma ligação (aresta) entre dois vértices do grafo.

4.2.2.3 Vertice

```
typedef struct Vertice Vertice
```

Representa um vértice (antena) no grafo.

4.2.2.4 VerticeSimples

```
typedef struct VerticeSimples VerticeSimples
```

Estrutura simples usada para guardar e carregar vértices em ficheiros binários.

4.2.3 Documentação das funções

4.2.3.1 adicionarAntena()

```
int adicionarAntena (  
    Grafo * g,  
    char freq,  
    int x,  
    int y)
```

Adiciona uma nova antena ao grafo.

Parâmetros

<i>g</i>	Grafo
<i>freq</i>	Frequência da antena
<i>x</i>	Coordenada X
<i>y</i>	Coordenada Y

Retorna

Índice da antena adicionada, ou -1 em caso de erro

Adiciona uma nova antena ao grafo.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>freq</i>	Frequência da antena
<i>x</i>	Coordenada X
<i>y</i>	Coordenada Y

Retorna

Índice da antena adicionada ou -1 se falhar

4.2.3.2 bft()

```
int bft (  
    Grafo * g,  
    int inicio)
```

Realiza percurso em largura (BFT) a partir de uma antena.

Parâmetros

<i>g</i>	Grafo
<i>inicio</i>	Índice da antena de partida

Retorna

1 após execução

Realiza percurso em largura (BFT) a partir de uma antena.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>inicio</i>	Índice da antena de partida

Retorna

1 após execução

4.2.3.3 carregarGrafoBinario()

```
int carregarGrafoBinario (
    Grafo * g,
    const char * nome)
```

Carrega um grafo a partir de um ficheiro binário.

Parâmetros

<i>g</i>	Grafo
<i>nome</i>	Nome do ficheiro

Retorna

1 se bem sucedido, 0 caso contrário

Carrega um grafo a partir de um ficheiro binário.

Parâmetros

<i>g</i>	Grafo
<i>nome</i>	Nome do ficheiro

Retorna

1 se for bem sucedido

4.2.3.4 dft()

```
int dft (
    Grafo * g,
    int inicio)
```

Realiza percurso em profundidade (DFT) a partir de uma antena.

Parâmetros

<i>g</i>	Grafo
<i>inicio</i>	Índice da antena de partida

Retorna

1 após execução

Realiza percurso em profundidade (DFT) a partir de uma antena.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>inicio</i>	Índice da antena de partida

Retorna

1 após execução

4.2.3.5 encontrarAntena()

```
int encontrarAntena (  
    Grafo * g,  
    int x,  
    int y)
```

Encontra o índice de uma antena com coordenadas específicas.

Parâmetros

<i>g</i>	Grafo
<i>x</i>	Coordenada X
<i>y</i>	Coordenada Y

Retorna

Índice da antena ou -1 se não for encontrada

Encontra o índice de uma antena com coordenadas específicas.

Parâmetros

<i>g</i>	Apontador para o grafo
<i>x</i>	Coordenada X
<i>y</i>	Coordenada Y

Retorna

Índice da antena ou -1 se não existir

4.2.3.6 guardarGrafoBinario()

```
int guardarGrafoBinario (  
    Grafo * g,  
    const char * nome)
```

Guarda o estado atual do grafo num ficheiro binário.

Parâmetros

<i>g</i>	Grafo
<i>nome</i>	Nome do ficheiro

Retorna

1 se bem sucedido, 0 caso contrário

Guarda o estado atual do grafo num ficheiro binário.

Parâmetros

<i>g</i>	Grafo
<i>nome</i>	Nome do ficheiro

Retorna

1 se for bem sucedido

4.2.3.7 iniciarGrafo()

```
int iniciarGrafo (  
    Grafo * g)
```

Inicializa um grafo vazio.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 se for bem sucedido

Inicializa um grafo vazio.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 se for bem sucedido

4.2.3.8 ligarAntenasIguais()

```
int ligarAntenasIguais (  
    Grafo * g)
```

Liga entre si todas as antenas que têm a mesma frequência.

Parâmetros

<i>g</i>	Grafo
----------	-------

Retorna

1 após criação das ligações

Liga entre si todas as antenas que têm a mesma frequência.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 se for bem sucedido

4.2.3.9 limparLigacoes()

```
int limparLigacoes (  
    Grafo * g)
```

Liberta todas as ligações atuais do grafo, evitando duplicações.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 após limpar

Liberta todas as ligações atuais do grafo, evitando duplicações.

Esta função percorre todos os vértices e liberta a memória das suas listas de ligações, deixando cada lista vazia. É útil para evitar duplicações de ligações quando se chamam funções como [ligarAntenasIguais\(\)](#) mais de uma vez.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 após a limpeza

4.2.3.10 mostrarAntenas()

```
int mostrarAntenas (  
    Grafo * g)
```

Mostra na consola todas as antenas registadas no grafo.

Parâmetros

<i>g</i>	Grafo
----------	-------

Retorna

1 após mostrar

Mostra na consola todas as antenas registadas no grafo.

Parâmetros

<i>g</i>	Apontador para o grafo
----------	------------------------

Retorna

1 após mostrar

4.2.3.11 mostrarIntersecoes()

```
int mostrarIntersecoes (
    Grafo * g,
    char freqA,
    char freqB)
```

Mostra as interseções entre ligações de frequências distintas (A-A vs B-B).

Parâmetros

<i>g</i>	Grafo
<i>freqA</i>	Primeira frequência
<i>freqB</i>	Segunda frequência

Retorna

1 após mostrar

Mostra as interseções entre ligações de frequências distintas (A-A vs B-B).

Considera como interseção o cruzamento entre arestas da frequência A com arestas da frequência B, se os segmentos (x1,y1)-(x2,y2) se cruzam no espaço.

Parâmetros

<i>g</i>	Grafo
<i>freqA</i>	Primeira frequência (ex: 'A')
<i>freqB</i>	Segunda frequência (ex: 'B')

Retorna

1 se correr com sucesso

4.2.3.12 mostrarTodosCaminhos()

```
int mostrarTodosCaminhos (
    Grafo * g,
    int origem,
    int destino,
    int * caminho,
    int comprimento,
    int * visitado)
```

Mostra todos os caminhos possíveis entre duas antenas.

Parâmetros

<i>g</i>	Grafo
<i>origem</i>	Índice da antena de origem
<i>destino</i>	Índice da antena de destino
<i>caminho</i>	Vetor auxiliar para armazenar o caminho atual
<i>comprimento</i>	Comprimento atual do caminho
<i>visitado</i>	Vetor de visitados

Retorna

1 após mostrar

Parâmetros

<i>g</i>	Grafo
<i>origem</i>	Índice da antena de origem
<i>destino</i>	Índice da antena de destino
<i>caminho</i>	Vetor auxiliar de caminho
<i>comprimento</i>	Comprimento atual do caminho
<i>visitado</i>	Vetor de visitados

Retorna

1

4.3 grafos.h

[Ir para a documentação deste ficheiro.](#)

```
00001 #ifndef GRAFOS_H
00002 #define GRAFOS_H
00003
00005 #define MAX_VERTICES 100
00006
00010 typedef struct Ligacao {
00011     int destino;
00012     struct Ligacao* seguinte;
00013 } Ligacao;
00014
00018 typedef struct Vertice {
00019     char freq;
00020     int x, y;
```

```
00021     Ligacao* ligacoes;
00022 } Vertice;
00023
00027 typedef struct Grafo {
00028     Vertice vertices[MAX_VERTICES];
00029     int tamanho;
00030 } Grafo;
00031
00035 typedef struct VerticeSimples {
00036     char freq;
00037     int x;
00038     int y;
00039 } VerticeSimples;
00040
00046 int iniciarGrafo(Grafo* g);
00047
00056 int adicionarAntena(Grafo* g, char freq, int x, int y);
00057
00063 int ligarAntenasIguais(Grafo* g);
00064
00070 int limparLigacoes(Grafo* g);
00071
00077 int mostrarAntenas(Grafo* g);
00078
00086 int encontrarAntena(Grafo* g, int x, int y);
00087
00094 int dft(Grafo* g, int inicio);
00095
00102 int bft(Grafo* g, int inicio);
00103
00114 int mostrarTodosCaminhos(Grafo* g, int origem, int destino, int* caminho, int comprimento, int*
    visitado);
00115
00123 int mostrarIntersecoes(Grafo* g, char freqA, char freqB);
00124
00131 int guardarGrafoBinario(Grafo* g, const char* nome);
00132
00139 int carregarGrafoBinario(Grafo* g, const char* nome);
00140
00141 #endif
```

4.4 Referência ao ficheiro main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "Grafos/grafos.h"
```

Funções

- int `main()`

4.4.1 Documentação das funções

4.4.1.1 `main()`

```
int main ()
```

Índice

- adicionarAntena
 - grafos.c, 10
 - grafos.h, 16
- bft
 - grafos.c, 10
 - grafos.h, 17
- carregarGrafoBinario
 - grafos.c, 10
 - grafos.h, 17
- criarLigacao
 - grafos.c, 11
- destino
 - Ligacao, 6
- dft
 - grafos.c, 11
 - grafos.h, 18
- encontrarAntena
 - grafos.c, 11
 - grafos.h, 19
- freq
 - Vertice, 7
 - VerticeSimples, 8
- Grafo, 5
 - grafos.h, 16
 - tamanho, 5
 - vertices, 5
- grafos.c
 - adicionarAntena, 10
 - bft, 10
 - carregarGrafoBinario, 10
 - criarLigacao, 11
 - dft, 11
 - encontrarAntena, 11
 - guardarGrafoBinario, 12
 - iniciarGrafo, 12
 - ligarAntenasIguais, 12
 - limparLigacoes, 13
 - mostrarAntenas, 13
 - mostrarIntersecoes, 13
 - mostrarTodosCaminhos, 14
 - visitarDFT, 14
- grafos.h
 - adicionarAntena, 16
 - bft, 17
- carregarGrafoBinario, 17
 - dft, 18
 - encontrarAntena, 19
 - Grafo, 16
 - guardarGrafoBinario, 19
 - iniciarGrafo, 20
 - Ligacao, 16
 - ligarAntenasIguais, 20
 - limparLigacoes, 21
 - MAX_VERTICES, 16
 - mostrarAntenas, 21
 - mostrarIntersecoes, 22
 - mostrarTodosCaminhos, 22
 - Vertice, 16
 - VerticeSimples, 16
- Grafos/grafos.c, 9
- Grafos/grafos.h, 15, 23
- guardarGrafoBinario
 - grafos.c, 12
 - grafos.h, 19
- iniciarGrafo
 - grafos.c, 12
 - grafos.h, 20
- Ligacao, 6
 - destino, 6
 - grafos.h, 16
 - seguinte, 6
- ligacoes
 - Vertice, 7
- ligarAntenasIguais
 - grafos.c, 12
 - grafos.h, 20
- limparLigacoes
 - grafos.c, 13
 - grafos.h, 21
- main
 - main.c, 24
- main.c, 24
 - main, 24
- MAX_VERTICES
 - grafos.h, 16
- mostrarAntenas
 - grafos.c, 13
 - grafos.h, 21
- mostrarIntersecoes
 - grafos.c, 13
 - grafos.h, 22

mostrarTodosCaminhos

grafos.c, [14](#)

grafos.h, [22](#)

seguinte

Ligacao, [6](#)

tamanho

Grafo, [5](#)

Vertice, [6](#)

freq, [7](#)

grafos.h, [16](#)

ligacoes, [7](#)

x, [7](#)

y, [7](#)

vertices

Grafo, [5](#)

VerticeSimples, [7](#)

freq, [8](#)

grafos.h, [16](#)

x, [8](#)

y, [8](#)

visitarDFT

grafos.c, [14](#)

x

Vertice, [7](#)

VerticeSimples, [8](#)

y

Vertice, [7](#)

VerticeSimples, [8](#)