Projeto - EDA Fase 2

Gerado por Doxygen 1.13.2

Capítulo 1

Índice das estruturas de dados

1.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Capítulo 2

Índice dos ficheiros

2.1 Lista de ficheiros

Lista de todos os ficheiros documentados com uma breve descrição:

Capítulo 3

Documentação da estruturas de dados

3.1 Referência à estrutura Antena

Estrutura de uma antena (vértice)

`#include <estruturas.h>`

Campos de Dados

- int **x**
- int **y**
- char **freq**
- int **visitado**
- int **id**
- struct Aresta * **adj**
- struct Antena * **next**

3.1.1 Descrição detalhada

Estrutura de uma antena (vértice)

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

- include/estruturas.h

3.2 Referência à estrutura Aresta

Estrutura de uma aresta.

`#include <estruturas.h>`

Campos de Dados

- struct Antena * **destino**
- struct Aresta * **next**

3.2.1 Descrição detalhada

Estrutura de uma aresta.

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

- include/estruturas.h

3.3 Referência à estrutura bfslista

Estrutura de uma lista de resultados da pesquisa em largura (bfs)

```
#include <estruturas.h>
```

Campos de Dados

- Antena * **antena**
- struct bfslista * **prox**

3.3.1 Descrição detalhada

Estrutura de uma lista de resultados da pesquisa em largura (bfs)

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

- include/estruturas.h

3.4 Referência à estrutura Caminho

Estrutura de um caminho (lista ligada)

```
#include <estruturas.h>
```

Campos de Dados

- int **tamanho**
- struct NoCaminho * **head**

3.4.1 Descrição detalhada

Estrutura de um caminho (lista ligada)

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

- include/estruturas.h

3.5 Referência à estrutura dfslista

Estrutura de uma lista de resultados da pesquisa em profundidade (dfs)

```
#include <estruturas.h>
```

Campos de Dados

- Antena ∗ **antena**
- struct dfslista ∗ **prox**

3.5.1 Descrição detalhada

Estrutura de uma lista de resultados da pesquisa em profundidade (dfs)

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

- include/estruturas.h

3.6 Referência à estrutura Fila

Estrutura de uma Fila (lista dinámica para bfs)

```
#include <estruturas.h>
```

Campos de Dados

- bfslista ∗ **inicio**
- bfslista ∗ **fim**

3.6.1 Descrição detalhada

Estrutura de uma Fila (lista dinámica para bfs)

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

- include/estruturas.h

3.7 Referência à estrutura Grafo

Estrutura do Gráfo.

`#include <estruturas.h>`

Campos de Dados

- int **V**
- struct Antena ∗ **vertices**

3.7.1 Descrição detalhada

Estrutura do Gráfo.

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

- include/estruturas.h

3.8 Referência à estrutura interlista

Estrutura de uma lista de resultados de interseção.

`#include <estruturas.h>`

Campos de Dados

- int **x1**
- int **y1**
- int **x2**
- int **y2**
- int **x3**
- int **y3**
- int **x4**
- int **y4**
- char **freq**
- struct interlista ∗ **prox**

3.8.1 Descrição detalhada

Estrutura de uma lista de resultados de interseção.

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

- include/estruturas.h

3.9 Referência à estrutura NoCaminho

Estrutura de um Nó de um caminho.

`#include <estruturas.h>`

Campos de Dados

- int **id_antena**
- struct NoCaminho * **prox**

3.9.1 Descrição detalhada

Estrutura de um Nó de um caminho.

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

- include/estruturas.h

Capítulo 4

Documentação do ficheiro

4.1 estruturas.h

```
00001
00009
00010 #ifndef ESTRUTURAS_H
00011 #define ESTRUTURAS_H
00012
00016
00017 typedef struct Antena
00018 {
00019     int x, y;            // coordenadas da antena
00020     char freq;           // frequência
00021     int visitado;        // inteiro para verificar se foi visitado
00022     int id;              // id
00023     struct Aresta *adj;  // lista de adjacência (arestas)
00024     struct Antena *next; // pointer prox antena
00025 } Antena;
00026
00030
00031 typedef struct Aresta
00032 {
00033     struct Antena *destino; // proxima antena
00034     struct Aresta *next;    // pointer prox aresta
00035 } Aresta;
00036
00040
00041 typedef struct Grafo
00042 {
00043     int V;                   //    número de vértices
00044     struct Antena *vertices; //    pointer para a lista ligada de antenas (head)
00045 } Grafo;
00046
00050
00051 typedef struct dfslista
00052 {
00053     Antena *antena;        // antena
00054     struct dfslista *prox; //   pointer prox antena
00055 } dfslista;
00056
00060
00061 typedef struct bfslista
00062 {
00063     Antena *antena;        //    antena
00064     struct bfslista *prox; //   pointer prox antena
00065 } bfslista;
00066
00070
00071 typedef struct Fila
00072 {
00073     bfslista *inicio; // inicio da fila
00074     bfslista *fim;    // fim da fila
00075 } Fila;
00076
00080
00081 typedef struct Caminho
00082 {
00083     int tamanho;            // número de saltos em um caminho
00084     struct NoCaminho *head; // inicio da lista
00085 } Caminho;
```

```
00086
00090
00091 typedef struct NoCaminho
00092 {
00093     int id_antena;          // id da antena no nó do caminho
00094     struct NoCaminho *prox; // pointer para o prox
00095 } NoCaminho;
00096
00100
00101 typedef struct interlista
00102 {
00103     int x1, y1, x2, y2;      //   coordenadas do primeiro segmento
00104     int x3, y3, x4, y4;      //   coordenadas do segundo segmento
00105     char freq;               // frequencia das antenas
00106     struct interlista *prox; // pointer para o prox
00107 } interlista;
00108
00109 #endif
```

4.2 Referência ao ficheiro include/func.h

```
#include "estruturas.h"
```

Funções

- Grafo * criarGrafo ()

  Função para criar e inicializar um grafo.
- Antena * criarVertice (Grafo *grafo, int x, int y, char freq)

  Função para criar uma antena/vértice no grafo.
- int criarAresta (Antena *origem, Antena *destino)

  Função para criar uma aresta para ligar duas antenas.
- Grafo * carregarAntenas (const char *ficheiro)

  Função principal de carregamento de antenas.
- int inserirArestas (Grafo *grafo)

  Função para criar uma aresta de 2 antenas com a mesma função.
- Antena * procurarAntenaId (Grafo *grafo, int id)

  Função para procurar uma antena no grafo pelo id.
- dfslista * dfs (Antena *atual, dfslista *lista)

  Função para fazer pesquisa em profundidade (DFS)
- dfslista * execDfs (Grafo *grafo, int id_inicio)

  Função para executar o dfs através de um id de antena.
- int inicializarFila (Fila *f)

  Função para inicializar a estrutura de uma fila vazia.
- int enfileirar (Fila *f, Antena *a)

  Função que insere uma antena no final da fila.
- Antena * desenfileirar (Fila *f)

  Função para remover a antena do inicio da lista.
- int filaVazia (Fila *f)

  verifica se a fila está vazia
- bfslista * execBfs (Grafo *grafo, int id_inicio)

  Função para fazer pesquisa em largura (bfs) com id de uma antena.
- int adicionarAoCamino (Caminho *c, int id_antena)

  Função para adicionar uma nova antena (id) a um caminho.
- int removerDoCaminho (Caminho *c)

  Função para remover uma antena (id) de um caminho.

- int dfsBacktracking (Antena *atual, Antena *destino, Caminho *caminho)
  Função dfs para procurar caminhos.
- int encontrarCaminhoDFS (Grafo *grafo, int id_origem, int id_destino, Caminho *caminho)
  Função principal para procurar caminhos entre 2 antenas no grafo.
- int segmentosSeCruzam (int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4)
  Função para verificar se 2 segmentos se cruzam.
- interlista * detetarIntersecoes (Grafo *grafo)
  Função para detetar interseções em um grafo.
- int escreverBloco (char *tag, FILE *f)
  Função para escrever um bloco em um ficheiro bin.
- int escreverAntenas (Grafo *grafo, FILE *f)
  Função para escrever antenas e os seus dados em um ficheiro bin.
- int escreverDFS (dfslista *dfs, FILE *f)
  Função para escrever pontos encontrados pelo bfs em um ficheiro bin.
- int escreverBFS (bfslista *bfs, FILE *f)
  Função para escrever pontos encontrados pelo bfs em um ficheiro bin.
- int escreverCaminho (Caminho *caminho, FILE *f)
  Função para escrever caminhos em um ficheiro bin.
- int escreverIntersecoes (interlista *lista, FILE *f)
  Função para escrever interseçoes em um ficheiro bin.
- int lerFicheiroEscreverOutput (const char *nome_ficheiro)
  Função para ler e escrever na consola o conteudo do ficheiro binário.
- int freeMem (dfslista *dfs, bfslista *bfs, interlista *ints)
  Função para libertar memória.

4.2.1 Descrição detalhada

Autor

Bruno (a31496@alunos.ipca.pt)

Versão

1.2

Data

2025-05-18

Copyright

Copyright (c) 2025

4.2.2 Documentação das funções

4.2.2.1 adicionarAoCamino()

```
int adicionarAoCamino (
            Caminho * c,
            int id_antena)
```

Função para adicionar uma nova antena (id) a um caminho.

Parâmetros

c	caminho
id_antena	id da antena

Retorna

1 = sucesso

4.2.2.2 carregarAntenas()

```
Grafo * carregarAntenas (
            const char * ficheiro)
```

Função principal de carregamento de antenas.

Parâmetros

ficheiro	nome do ficheiro txt com antenas

Retorna

grafo com antenas inseridas

4.2.2.3 criarAresta()

```
int criarAresta (
            Antena * origem,
            Antena * destino)
```

Função para criar uma aresta para ligar duas antenas.

Parâmetros

origem	antena de origem
destino	antena de destino

Retorna

1 = sucesso

4.2.2.4 criarGrafo()

```
Grafo * criarGrafo ()
```

Função para criar e inicializar um grafo.

Retorna

grafo criado

4.2.2.5 criarVertice()

```
Antena * criarVertice (
            Grafo * grafo,
            int x,
            int y,
            char freq)
```

Função para criar uma antena/vértice no grafo.

Parâmetros

grafo	grafo para inserir as antena
x	coordenada x da antena
y	coordenada y da antena
freq	frequência da antena

Retorna

pointer para a antena criada

4.2.2.6 desenfileirar()

```
Antena * desenfileirar (
            Fila * f)
```

Função para remover a antena do inicio da lista.

Parâmetros

f	fila

Retorna

antena do inicio da lista

4.2.2.7 detetarIntersecoes()

```
interlista * detetarIntersecoes (
            Grafo * grafo)
```

Função para detetar interseções em um grafo.

Parâmetros

grafo	grafo com as antenas

Retorna

Lista com os pontos que se intersetam

4.2.2.8 dfs()

```
dfslista * dfs (
            Antena * atual,
            dfslista * lista)
```

Função para fazer pesquisa em profundidade (DFS)

Parâmetros

atual	antena atual
lista	lista de antenas já visitadas

Retorna

lista com todas as antenas alcançadas pelo dfs

4.2.2.9 dfsBacktracking()

```
int dfsBacktracking (
            Antena * atual,
            Antena * destino,
            Caminho * caminho)
```

Função dfs para procurar caminhos.

Parâmetros

atual	antena atual
destino	antena destino
caminho	lista do caminho

Retorna

1 = caminho encontrado, 0 caso contrario

4.2.2.10 encontrarCaminhoDFS()

```
int encontrarCaminhoDFS (
            Grafo * grafo,
            int id_origem,
            int id_destino,
            Caminho * caminho)
```

Função principal para procurar caminhos entre 2 antenas no grafo.

Parâmetros

grafo	grafo com antenas
id_origem	id da antena de origem
id_destino	id da antena de origem
caminho	lista do caminho

Retorna

1 = caminho encontrado, 0 caso contrario

4.2.2.11 enfileirar()

```
int enfileirar (
            Fila * f,
            Antena * a)
```

Função que insere uma antena no final da fila.

Parâmetros

f	fila
a	antena que será inserida

Retorna

1 = sucesso

4.2.2.12 escreverAntenas()

```
int escreverAntenas (
            Grafo * grafo,
            FILE * f)
```

Função para escrever antenas e os seus dados em um ficheiro bin.

Parâmetros

grafo	grafo com antenas
f	ficheiro binário aberto

Retorna

1 = sucesso

4.2.2.13 escreverBFS()

```
int escreverBFS (
            bfslista * bfs,
            FILE * f)
```

Função para escrever pontos encontrados pelo bfs em um ficheiro bin.

Parâmetros

bfs	lista de pontos encontrados pelo bfs
f	ficheiro binário aberto

Retorna

1 = sucesso

4.2.2.14 escreverBloco()

```
int escreverBloco (
            char * tag,
            FILE * f)
```

Função para escrever um bloco em um ficheiro bin.

Parâmetros

tag	tag de 3 caracteres para identificar os dados exmpl "ANT" - antenas
f	ficheiro binário aberto

Retorna

1 = sucesso

4.2.2.15 escreverCaminho()

```
int escreverCaminho (
            Caminho * caminho,
            FILE * f)
```

Função para escrever caminhos em um ficheiro bin.

Parâmetros

caminho	lista de caminho
f	ficheiro binário aberto

Retorna

1 = sucesso

4.2.2.16 escreverDFS()

```
int escreverDFS (
            dfslista * dfs,
            FILE * f)
```

Função para escrever pontos encontrados pelo bfs em um ficheiro bin.

Parâmetros

dfs	lista de pontos encontrados pelo dfs
f	ficheiro binário aberto

Retorna

1 = sucesso

4.2.2.17 escreverIntersecoes()

```
int escreverIntersecoes (
            interlista * lista,
            FILE * f)
```

Função para escrever interseçoes em um ficheiro bin.

Parâmetros

lista	lista de inter
f	ficheiro binário aberto

Retorna

1 = sucesso

4.2.2.18 execBfs()

```
bfslista * execBfs (
            Grafo * grafo,
            int id_inicio)
```

Função para fazer pesquisa em largura (bfs) com id de uma antena.

Parâmetros

grafo	grafo com as antenas
id_inicio	id da antena inicial

Retorna

lista com as antenas visitadas pelo bfs

4.2.2.19 execDfs()

```
dfslista * execDfs (
            Grafo * grafo,
            int id_inicio)
```

Função para executar o dfs através de um id de antena.

Parâmetros

grafo	grafo com antenas
id_inicio	id da antena que começa a pesquisa

Retorna

lista com todos os pontos alcançados pelo dfs

4.2.2.20 filaVazia()

```
int filaVazia (
            Fila * f)
```

verifica se a fila está vazia

Parâmetros

f	fila

Retorna

1 se está vazia, caso contrario 0

4.2.2.21 freeMem()

```
int freeMem (
            dfslista * dfs,
            bfslista * bfs,
            interlista * ints)
```

Função para libertar memória.

Parâmetros

dfs	lista de dfs
bfs	lista de dfs
ints	lista de interseções

Retorna

1 = sucesso

4.2.2.22 inicializarFila()

```
int inicializarFila (
            Fila * f)
```

Função para inicializar a estrutura de uma fila vazia.

Parâmetros

f	fila que será inicializada

Retorna

1 = sucesso

4.2.2.23 inserirArestas()

```
int inserirArestas (
            Grafo * grafo)
```

Função para criar uma aresta de 2 antenas com a mesma função.

Parâmetros

grafo	grafo com antenas inseridas

Retorna

1 = sucesso

4.2.2.24 lerFicheiroEscreverOutput()

```
int lerFicheiroEscreverOutput (
            const char * nome_ficheiro)
```

Função para ler e escrever na consola o conteudo do ficheiro binário.

Parâmetros

nome_ficheiro	nome do ficheiro binário

Retorna

1 = sucesso

4.2.2.25 procurarAntenaId()

```
Antena * procurarAntenaId (
            Grafo * grafo,
            int id)
```

Função para procurar uma antena no grafo pelo id.

Parâmetros

grafo	grafo com as antenas
id	id da antena

Retorna

antena pedida

4.2.2.26 removerDoCaminho()

```
int removerDoCaminho (
            Caminho * c)
```

Função para remover uma antena (id) de um caminho.

Parâmetros

c	caminho

Retorna

1 = sucesso

4.2.2.27 segmentosSeCruzam()

```
int segmentosSeCruzam (
            int x1,
            int y1,
            int x2,
            int y2,
            int x3,
            int y3,
            int x4,
            int y4)
```

Função para verificar se 2 segmentos se cruzam.

Parâmetros

x1	coordenada x do primeiro ponto
y1	coordenada y do primeiro ponto
x2	coordenada x do segundo ponto
y2	coordenada y do segundo ponto
x3	coordenada x do terceiro ponto
y3	coordenada y do terceiro ponto
x4	coordenada x do quarto ponto
y4	coordenada y do quarto ponto

Retorna

1 se os segmentos (x1,y1)-(x2,y2) e (x3,y3)-(x4,y4) se cruzam, 0 caso contrário

4.3 func.h

Ir para a documentação deste ficheiro.

```
00001
00009
00010 #ifndef FUNC_H
00011 #define FUNC_H
00012
00013 #include "estruturas.h"
00014
00015 Grafo *criarGrafo();
00016 Antena *criarVertice(Grafo *grafo, int x, int y, char freq);
00017 int criarAresta(Antena *origem, Antena *destino);
00018
00019 Grafo *carregarAntenas(const char *ficheiro);
00020 int inserirArestas(Grafo *grafo);
00021
```

```
00022 Antena *procurarAntenaId(Grafo *grafo, int id);
00023
00024 dfslista *dfs(Antena *atual, dfslista *lista);
00025 dfslista *execDfs(Grafo *grafo, int id_inicio);
00026
00027 int inicializarFila(Fila *f);
00028 int enfileirar(Fila *f, Antena *a);
00029 Antena *desenfileirar(Fila *f);
00030 int filaVazia(Fila *f);
00031 bfslista *execBfs(Grafo *grafo, int id_inicio);
00032
00033 int adicionarAoCamino(Caminho *c, int id_antena);
00034 int removerDoCaminho(Caminho *c);
00035 int dfsBacktracking(Antena *atual, Antena *destino, Caminho *caminho);
00036 int encontrarCaminhoDFS(Grafo *grafo, int id_origem, int id_destino, Caminho *caminho);
00037
00038 int segmentosSeCruzam(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4);
00039 interlista *detetarIntersecoes(Grafo *grafo);
00040
00041 int escreverBloco(char *tag, FILE *f);
00042 int escreverAntenas(Grafo *grafo, FILE *f);
00043 int escreverDFS(dfslista *dfs, FILE *f);
00044 int escreverBFS(bfslista *bfs, FILE *f);
00045 int escreverCaminho(Caminho *caminho, FILE *f);
00046 int escreverIntersecoes(interlista *lista, FILE *f);
00047 int lerFicheiroEscreverOutput(const char *nome_ficheiro);
00048
00049 int freeMem(dfslista *dfs, bfslista *bfs, interlista *ints);
00050
00051 #endif
```

4.4 Referência ao ficheiro src/func.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "../include/estruturas.h"
#include "../include/func.h"
```

Funções

- Grafo ∗ criarGrafo ()

  Função para criar e inicializar um grafo.
- Antena ∗ criarVertice (Grafo ∗grafo, int x, int y, char freq)

  Função para criar uma antena/vértice no grafo.
- int criarAresta (Antena ∗origem, Antena ∗destino)

  Função para criar uma aresta para ligar duas antenas.
- Grafo ∗ carregarAntenas (const char ∗ficheiro)

  Função principal de carregamento de antenas.
- int inserirArestas (Grafo ∗grafo)

  Função para criar uma aresta de 2 antenas com a mesma função.
- Antena ∗ procurarAntenaId (Grafo ∗grafo, int id)

  Função para procurar uma antena no grafo pelo id.
- dfslista ∗ dfs (Antena ∗atual, dfslista ∗lista)

  Função para fazer pesquisa em profundidade (DFS)
- dfslista ∗ execDfs (Grafo ∗grafo, int id_inicio)

  Função para executar o dfs através de um id de antena.
- int inicializarFila (Fila ∗f)

  Função para inicializar a estrutura de uma fila vazia.

- int enfileirar (Fila *f, Antena *a)

  Função que insere uma antena no final da fila.
- Antena * desenfileirar (Fila *f)

  Função para remover a antena do inicio da lista.
- int filaVazia (Fila *f)

  verifica se a fila está vazia
- bfslista * execBfs (Grafo *grafo, int id_inicio)

  Função para fazer pesquisa em largura (bfs) com id de uma antena.
- int adicionarAoCamino (Caminho *c, int id_antena)

  Função para adicionar uma nova antena (id) a um caminho.
- int removerDoCaminho (Caminho *c)

  Função para remover uma antena (id) de um caminho.
- int dfsBacktracking (Antena *atual, Antena *destino, Caminho *caminho)

  Função dfs para procurar caminhos.
- int encontrarCaminhoDFS (Grafo *grafo, int id_origem, int id_destino, Caminho *caminho)

  Função principal para procurar caminhos entre 2 antenas no grafo.
- int segmentosSeCruzam (int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4)

  Função para verificar se 2 segmentos se cruzam.
- interlista * detetarIntersecoes (Grafo *grafo)

  Função para detetar interseções em um grafo.
- int escreverBloco (char *tag, FILE *f)

  Função para escrever um bloco em um ficheiro bin.
- int escreverAntenas (Grafo *grafo, FILE *f)

  Função para escrever antenas e os seus dados em um ficheiro bin.
- int escreverDFS (dfslista *dfs, FILE *f)

  Função para escrever pontos encontrados pelo bfs em um ficheiro bin.
- int escreverBFS (bfslista *bfs, FILE *f)

  Função para escrever pontos encontrados pelo bfs em um ficheiro bin.
- int escreverCaminho (Caminho *caminho, FILE *f)

  Função para escrever caminhos em um ficheiro bin.
- int escreverIntersecoes (interlista *lista, FILE *f)

  Função para escrever interseçoes em um ficheiro bin.
- int lerFicheiroEscreverOutput (const char *nome_ficheiro)

  Função para ler e escrever na consola o conteudo do ficheiro binário.
- int freeMem (dfslista *dfs, bfslista *bfs, interlista *ints)

  Função para libertar memória.

4.4.1 Descrição detalhada

Autor

Bruno (a31496@alunos.ipca.pt)

Versão

1.2

Data

2025-05-18

Copyright

4.4.2 Documentação das funções

4.4.2.1 adicionarAoCamino()

```
int adicionarAoCamino (
            Caminho * c,
            int id_antena)
```

Função para adicionar uma nova antena (id) a um caminho.

Parâmetros

c	caminho
id_antena	id da antena

Retorna

1 = sucesso

4.4.2.2 carregarAntenas()

```
Grafo * carregarAntenas (
            const char * ficheiro)
```

Função principal de carregamento de antenas.

Parâmetros

ficheiro	nome do ficheiro txt com antenas

Retorna

grafo com antenas inseridas

4.4.2.3 criarAresta()

```
int criarAresta (
            Antena * origem,
            Antena * destino)
```

Função para criar uma aresta para ligar duas antenas.

Parâmetros

origem	antena de origem
destino	antena de destino

Retorna

1 = sucesso

4.4.2.4 criarGrafo()

```
Grafo * criarGrafo ()
```

Função para criar e inicializar um grafo.

Retorna

grafo criado

4.4.2.5 criarVertice()

```
Antena * criarVertice (
            Grafo * grafo,
            int x,
            int y,
            char freq)
```

Função para criar uma antena/vértice no grafo.

Parâmetros

grafo	grafo para inserir as antena
x	coordenada x da antena
y	coordenada y da antena
freq	frequência da antena

Retorna

pointer para a antena criada

4.4.2.6 desenfileirar()

```
Antena * desenfileirar (
            Fila * f)
```

Função para remover a antena do inicio da lista.

Parâmetros

f	fila

Retorna

antena do inicio da lista

4.4.2.7 detetarIntersecoes()

```
interlista * detetarIntersecoes (
            Grafo * grafo)
```

Função para detetar interseções em um grafo.

Parâmetros

grafo	grafo com as antenas

Retorna

Lista com os pontos que se intersetam

4.4.2.8 dfs()

```
dfslista * dfs (
            Antena * atual,
            dfslista * lista)
```

Função para fazer pesquisa em profundidade (DFS)

Parâmetros

atual	antena atual
lista	lista de antenas já visitadas

Retorna

lista com todas as antenas alcançadas pelo dfs

4.4.2.9 dfsBacktracking()

```
int dfsBacktracking (
            Antena * atual,
            Antena * destino,
            Caminho * caminho)
```

Função dfs para procurar caminhos.

Parâmetros

atual	antena atual
destino	antena destino
caminho	lista do caminho

Retorna

1 = caminho encontrado, 0 caso contrario

4.4.2.10 encontrarCaminhoDFS()

```
int encontrarCaminhoDFS (
            Grafo * grafo,
            int id_origem,
            int id_destino,
            Caminho * caminho)
```

Função principal para procurar caminhos entre 2 antenas no grafo.

Parâmetros

grafo	grafo com antenas
id_origem	id da antena de origem
id_destino	id da antena de origem
caminho	lista do caminho

Retorna

1 = caminho encontrado, 0 caso contrario

4.4.2.11 enfileirar()

```
int enfileirar (
            Fila * f,
            Antena * a)
```

Função que insere uma antena no final da fila.

Parâmetros

f	fila
a	antena que será inserida

Retorna

1 = sucesso

4.4.2.12 escreverAntenas()

```
int escreverAntenas (
            Grafo * grafo,
            FILE * f)
```

Função para escrever antenas e os seus dados em um ficheiro bin.

Parâmetros

grafo	grafo com antenas
f	ficheiro binário aberto

Retorna

1 = sucesso

4.4.2.13 escreverBFS()

```
int escreverBFS (
            bfslista * bfs,
            FILE * f)
```

Função para escrever pontos encontrados pelo bfs em um ficheiro bin.

Parâmetros

bfs	lista de pontos encontrados pelo bfs
f	ficheiro binário aberto

Retorna

1 = sucesso

4.4.2.14 escreverBloco()

```
int escreverBloco (
            char * tag,
            FILE * f)
```

Função para escrever um bloco em um ficheiro bin.

Parâmetros

tag	tag de 3 caracteres para identificar os dados exmpl "ANT" - antenas
f	ficheiro binário aberto

Retorna

1 = sucesso

4.4.2.15 escreverCaminho()

```
int escreverCaminho (
            Caminho * caminho,
            FILE * f)
```

Função para escrever caminhos em um ficheiro bin.

Parâmetros

caminho	lista de caminho
f	ficheiro binário aberto

Retorna

1 = sucesso

4.4.2.16 escreverDFS()

```
int escreverDFS (
            dfslista * dfs,
            FILE * f)
```

Função para escrever pontos encontrados pelo bfs em um ficheiro bin.

Parâmetros

dfs	lista de pontos encontrados pelo dfs
f	ficheiro binário aberto

Retorna

1 = sucesso

4.4.2.17 escreverIntersecoes()

```
int escreverIntersecoes (
            interlista * lista,
            FILE * f)
```

Função para escrever interseçoes em um ficheiro bin.

Parâmetros

lista	lista de inter
f	ficheiro binário aberto

Retorna

1 = sucesso

4.4.2.18 execBfs()

```
bfslista * execBfs (
            Grafo * grafo,
            int id_inicio)
```

Função para fazer pesquisa em largura (bfs) com id de uma antena.

Parâmetros

grafo	grafo com as antenas
id_inicio	id da antena inicial

Retorna

lista com as antenas visitadas pelo bfs

4.4.2.19 execDfs()

```
dfslista * execDfs (
            Grafo * grafo,
            int id_inicio)
```

Função para executar o dfs através de um id de antena.

Parâmetros

grafo	grafo com antenas
id_inicio	id da antena que começa a pesquisa

Retorna

lista com todos os pontos alcançados pelo dfs

4.4.2.20 filaVazia()

```
int filaVazia (
            Fila * f)
```

verifica se a fila está vazia

Parâmetros

f	fila

Retorna

1 se está vazia, caso contrario 0

4.4.2.21 freeMem()

```
int freeMem (
            dfslista * dfs,
            bfslista * bfs,
            interlista * ints)
```

Função para libertar memória.

Parâmetros

dfs	lista de dfs
bfs	lista de dfs
ints	lista de interseções

Retorna

1 = sucesso

4.4.2.22 inicializarFila()

```
int inicializarFila (
            Fila * f)
```

Função para inicializar a estrutura de uma fila vazia.

Parâmetros

f	fila que será inicializada

Retorna

1 = sucesso

4.4.2.23 inserirArestas()

```
int inserirArestas (
            Grafo * grafo)
```

Função para criar uma aresta de 2 antenas com a mesma função.

Parâmetros

grafo	grafo com antenas inseridas

Retorna

1 = sucesso

4.4.2.24 lerFicheiroEscreverOutput()

```
int lerFicheiroEscreverOutput (
            const char * nome_ficheiro)
```

Função para ler e escrever na consola o conteudo do ficheiro binário.

Parâmetros

nome_ficheiro	nome do ficheiro binário

Retorna

1 = sucesso

4.4.2.25 procurarAntenaId()

```
Antena * procurarAntenaId (
            Grafo * grafo,
            int id)
```

Função para procurar uma antena no grafo pelo id.

Parâmetros

grafo	grafo com as antenas
id	id da antena

Retorna

antena pedida

4.4.2.26 removerDoCaminho()

```
int removerDoCaminho (
            Caminho * c)
```

Função para remover uma antena (id) de um caminho.

Parâmetros

c	caminho

Retorna

1 = sucesso

4.4.2.27 segmentosSeCruzam()

```
int segmentosSeCruzam (
            int x1,
            int y1,
            int x2,
            int y2,
            int x3,
            int y3,
            int x4,
            int y4)
```

Função para verificar se 2 segmentos se cruzam.

Parâmetros

x1	coordenada x do primeiro ponto
y1	coordenada y do primeiro ponto
x2	coordenada x do segundo ponto
y2	coordenada y do segundo ponto
x3	coordenada x do terceiro ponto
y3	coordenada y do terceiro ponto
x4	coordenada x do quarto ponto
y4	coordenada y do quarto ponto

Retorna

1 se os segmentos (x1,y1)-(x2,y2) e (x3,y3)-(x4,y4) se cruzam, 0 caso contrário

Índice