My Project

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 draw Namespace Reference

Functions

- draw_network (G, node_mapping, origem, destino, caminho1, caminho2, algoritmo)

  Desenha o grafo da rede, destacando os caminhos e nós importantes.
- draw_empty_network (G, node_mapping)

  Desenha uma rede vazia, sem destacar caminhos ou nós.
- draw_suurballe (G, origem_split, destino_split, caminho1_split, caminho2_split, filename)

  Desenha um grafo dividido ou transformado e destaca os caminhos encontrados pelo algoritmo Suurballe.

3.1.1 Function Documentation

3.1.1.1 draw_empty_network()

```
draw.draw_empty_network (
              G,
              node_mapping)
```

Desenha uma rede vazia, sem destacar caminhos ou nós.

Esta função gera uma visualização gráfica do grafo sem destacar nenhum caminho ou nó específico.

Parameters

G	O grafo direcionado a ser desenhado.
node_mapping	Mapa de nós, associando índices aos nomes dos nós no grafo.

Note

Esta função é útil para exibir apenas a estrutura do grafo sem informações adicionais, como os caminhos ou a origem/destino.

3.1.1.2 draw_network()

```
draw.draw_network (
              G,
              node_mapping,
              origem,
              destino,
              caminho1,
              caminho2,
              algoritmo)
```

Desenha o grafo da rede, destacando os caminhos e nós importantes.

Esta função gera uma visualização gráfica do grafo, destacando o caminho mais curto entre o nó de origem e o nó de destino. Além disso, permite destacar dois caminhos diferentes (caso existam) e colorir os nós de origem e destino de forma distinta. O algoritmo usado para encontrar os caminhos pode ser especificado.

Parameters

G	O grafo direcionado a ser desenhado.
node_mapping	Mapa de nós, associando índices aos nomes dos nós no grafo.
origem	Índice do nó de origem, que será destacado na visualização.
destino	Índice do nó de destino, que será destacado na visualização.
caminho1	Lista de nós que representam o primeiro caminho a ser destacado.
caminho2	Lista de nós que representam o segundo caminho a ser destacado (pode ser None).
algoritmo	Inteiro indicando o algoritmo utilizado: 1. "Two Step Approach" 2. "Suurballe"

Note

A função gera uma imagem do grafo e a salva no diretório de saída como "rede_final.png".

3.1.1.3 draw_suurballe()

```
draw.draw_suurballe (
              G,
              origem_split,
              destino_split,
              caminho1_split,
              caminho2_split,
              filename)
```

Desenha um grafo dividido ou transformado e destaca os caminhos encontrados pelo algoritmo Suurballe.

A função visualiza o grafo após a aplicação do algoritmo Suurballe, destacando os caminhos mais curtos (se encontrados) e as arestas internas (no formato '_in'/'_out'). Cada etapa do algoritmo é ilustrada, mostrando o estado atual do grafo, o primeiro caminho mais curto em verde e o segundo caminho (se houver) em azul.

Parameters

G	Grafo direcionado (potencialmente dividido), representando a rede.
origem_split	Nó de origem no formato dividido (com sufixo '_out').
destino_split	Nó de destino no formato dividido (com sufixo '_in').
caminho1_split	Lista de nós representando o primeiro caminho encontrado (caminho mais curto).
caminho2_split	Lista de nós representando o segundo caminho encontrado (pode ser None).
filename	Nome do arquivo usado para salvar a imagem gerada, representando o estado atual do grafo.

Note

A função cria uma visualização do grafo, destacando os caminhos e as arestas em diferentes cores para facilitar a compreensão do algoritmo.

As arestas internas (de um nó dividido) são desenhadas com uma linha tracejada e de cor prata.

3.2 functions Namespace Reference

Functions

- retrieve_data (data)

  Processa os dados de entrada, para criar um grafo direcionado.
- find_best_paths (G, origem, destino)

  Encontra os dois melhores caminhos disjuntos entre os dois nós com base no menor custo.
- suurballe (G, origem_orig, destino_orig)

  Implementa o algoritmo de Suurballe para encontrar dois caminhos disjuntos em um grafo.
- split_nodes (G, source_orig, target_orig)

  Divide os nós de um grafo em nós de entrada e saída.
- merge_split_path (split_path)

  Converte um caminho do grafo dividido (com nós '_in'/'_out') de volta para os nomes originais dos nós.

3.2.1 Function Documentation

3.2.1.1 find_best_paths()

```
functions.find_best_paths (
            G,
            origem,
            destino)
```

Encontra os dois melhores caminhos disjuntos entre os dois nós com base no menor custo.

Esta função primeiro calcula o caminho mais curto entre os nós de origem e destino. Em seguida, remove todos os nós e arestas intermediários envolvidos no primeiro caminho, garantindo que o segundo caminho seja completamente disjunto. Se não existir um segundo caminho, notifica o usuário.

Parameters

G	O grafo direcionado.
origem	O nó de origem.
destino	O nó de destino.

Returns

Tuple (path1, cost1, path2, cost2), onde:

- path1: O primeiro caminho mais curto.
- cost1: O custo total do primeiro caminho.
- path2: O segundo caminho mais curto (caso contrário, None).
- cost2: O custo total do segundo caminho (caso contrário, None).

3.2.1.2 merge_split_path()

```
functions.merge_split_path (
              split_path)
```

Converte um caminho do grafo dividido (com nós '_in'/'_out') de volta para os nomes originais dos nós.

Esta função recebe um caminho que foi gerado a partir de um grafo dividido (onde os nós foram modificados para incluir sufixos '_in' e '_out') e converte esse caminho de volta para os nomes dos nós originais. Ela remove os sufixos '_in' e '_out' e retorna o caminho com os nós originais.

Parameters

split_path	Caminho com os nós divididos (com sufixos '_in' ou '_out').

Returns

Um caminho com os nomes dos nós originais, sem os sufixos '_in' ou '_out'.

Note

Caso o caminho esteja vazio, a função retorna uma lista vazia.

3.2.1.3 retrieve_data()

```
functions.retrieve_data (
              data)
```

Processa os dados de entrada, para criar um grafo direcionado.

Esta função extrai informações de nós e links dos dados de entrada fornecidos pelo ficheiro de entrada. Cria um grafo direcionado, onde cada nó é adicionado com suas coordenadas e cada link é estabelecido.

Parameters

data	A string de entrada contém os dados a serem analisados. Deve conter seções para nós e links.

Returns

Um grafo direcionado, onde os nós são conectados por links conforme o ficheiro de entrada.

3.2.1.4 split_nodes()

```
functions.split_nodes (
              G,
              source_orig,
              target_orig)
```

Divide os nós de um grafo em nós de entrada e saída.

Esta função divide cada nó de um grafo em dois nós: um nó de entrada (no formato `A_in`) e um nó de saída (no formato `A_out`). Ela cria um novo grafo com esses nós divididos e ajusta as arestas entre eles. A origem e o destino também são ajustados para os nós de saída e entrada, respectivamente.

Parameters

G	Grafo direcionado original a ser dividido.
source_orig	Nó de origem no grafo original.
target_orig	Nó de destino no grafo original.

Returns

Um grafo direcionado `H` com nós divididos, o novo nó de origem `s` (com sufixo `_out`), e o novo nó de destino `t` (com sufixo `_in`).

3.2.1.5 suurballe()

```
functions.suurballe (
              G,
              origem_orig,
              destino_orig)
```

Implementa o algoritmo de Suurballe para encontrar dois caminhos disjuntos em um grafo.

Esta função utiliza o algoritmo de Suurballe para encontrar dois caminhos mais curtos disjuntos entre dois nós de um grafo direcionado. O primeiro caminho é encontrado usando o algoritmo de Dijkstra, e o segundo caminho é calculado após a remoção dos nós pertencentes ao primeiro caminho.

Parameters

G	Grafo direcionado em que os caminhos serão calculados.
origem_orig	Nó de origem no grafo.
destino_orig	Nó de destino no grafo.

Returns

Uma lista contendo dois caminhos: o primeiro caminho mais curto e o segundo caminho mais curto disjunto.

Note

Caso o segundo caminho não exista, o valor retornado será `None` para o segundo caminho.

3.3 menus Namespace Reference

Functions

- ask_network ()

  Solicita ao user o nome do ficheiro da rede.
- ask_origin_destiny (node_mapping)

  Solicita ao user os nós de origem e destino.
- clear_screen ()

  Limpa a tela do terminal.
- ask_which_algorithm ()

  Exibe um menu para o usuário escolher um algoritmo a ser utilizado.

3.3.1 Function Documentation

3.3.1.1 ask_network()

```
menus.ask_network ()
```

Solicita ao user o nome do ficheiro da rede.

Esta função solicita ao user o nome do ficheiro da rede que deseja adicionar.

Returns

caminho_ficheiro Caminho do ficheiro da rede.

3.3.1.2 ask_origin_destiny()

```
menus.ask_origin_destiny (
              node_mapping)
```

Solicita ao user os nós de origem e destino.

Esta função solicita ao user os números correspondentes aos nós de origem e de destino.

Parameters

node_mapping	Mapa de nós.

Returns

Tuple (origem, destino) onde:

Parameters

origem	O número do nó de origem selecionado.
destino	O número do nó de destino selecionado.

3.3.1.3 ask_which_algorithm()

```
menus.ask_which_algorithm ()
```

Exibe um menu para o usuário escolher um algoritmo a ser utilizado.

Esta função exibe um menu interativo que permite ao usuário escolher entre três opções de algoritmos: "Two Step Approach", "Suurballe" ou uma opção ainda não implementada para usar ambos os métodos. A função valida a entrada do usuário para garantir que a escolha seja uma opção válida.

Returns

A opção escolhida pelo usuário como um número inteiro: 1 para "Two Step Approach", 2 para "Suurballe", 3 para "Usar ambos os métodos" (não implementado).

Note

A opção 3 ainda não está implementada.

3.3.1.4 clear_screen()

```
menus.clear_screen ()
```

Limpa a tela do terminal.

3.4 task Namespace Reference

Variables

- list networks = ["networks/abilene.txt", "networks/atlanta.txt", "networks/nobel-eu.txt", "networks/nobel-germany.txt"]
- display_name = ficheiro.replace('networks/', '')
- int escolha = int(input("Digite a opção pretendida: ")) - 1
- novo_ficheiro = ask_network()
- network_data = file.read()
- G
- node_mapping
- origem
- destino
- algoritmo = ask_which_algorithm()
- caminho1
- custo1
- caminho2
- custo2

3.4.1 Variable Documentation

3.4.1.1 algoritmo

```
task.algoritmo = ask_which_algorithm()
```

3.4.1.2 caminho1

```
task.caminho1
```

3.4.1.3 caminho2

```
task.caminho2
```

3.4.1.4 custo1

```
task.custo1
```

3.4.1.5 custo2

```
task.custo2
```

3.4.1.6 destino

```
task.destino
```

3.4.1.7 display_name

```
task.display_name = ficheiro.replace('networks/', '')
```

3.4.1.8 escolha

```
int task.escolha = int(input("Digite a opção pretendida:  ")) - 1
```

3.4.1.9 G

```
task.G
```

3.4.1.10 network_data

```
task.network_data = file.read()
```

3.4.1.11 networks

```
list task.networks = ["networks/abilene.txt", "networks/atlanta.txt", "networks/nobel-eu.txt",
"networks/nobel-germany.txt"]
```

3.4.1.12 node_mapping

```
task.node_mapping
```

3.4.1.13 novo_ficheiro

```
task.novo_ficheiro = ask_network()
```

3.4.1.14 origem

```
task.origem
```

Chapter 4

File Documentation

4.1 draw.py File Reference

Namespaces

- namespace draw

Functions

- draw.draw_network (G, node_mapping, origem, destino, caminho1, caminho2, algoritmo)

  Desenha o grafo da rede, destacando os caminhos e nós importantes.
- draw.draw_empty_network (G, node_mapping)

  Desenha uma rede vazia, sem destacar caminhos ou nós.
- draw.draw_suurballe (G, origem_split, destino_split, caminho1_split, caminho2_split, filename)

  Desenha um grafo dividido ou transformado e destaca os caminhos encontrados pelo algoritmo Suurballe.

4.2 functions.py File Reference

Namespaces

- namespace functions

Functions

- functions.retrieve_data (data)

  Processa os dados de entrada, para criar um grafo direcionado.
- functions.find_best_paths (G, origem, destino)

  Encontra os dois melhores caminhos disjuntos entre os dois nós com base no menor custo.
- functions.suurballe (G, origem_orig, destino_orig)

  Implementa o algoritmo de Suurballe para encontrar dois caminhos disjuntos em um grafo.
- functions.split_nodes (G, source_orig, target_orig)

  Divide os nós de um grafo em nós de entrada e saída.
- functions.merge_split_path (split_path)

  Converte um caminho do grafo dividido (com nós '_in'/'_out') de volta para os nomes originais dos nós.

4.3 menus.py File Reference

Namespaces

- namespace menus

Functions

- menus.ask_network ()

  Solicita ao user o nome do ficheiro da rede.
- menus.ask_origin_destiny (node_mapping)

  Solicita ao user os nós de origem e destino.
- menus.clear_screen ()

  Limpa a tela do terminal.
- menus.ask_which_algorithm ()

  Exibe um menu para o usuário escolher um algoritmo a ser utilizado.

4.4 task.py File Reference

Namespaces

- namespace task

Variables

- list task.networks = ["networks/abilene.txt", "networks/atlanta.txt", "networks/nobel-eu.txt", "networks/nobel-germany.txt"]
- task.display_name = ficheiro.replace('networks/', '')
- int task.escolha = int(input("Digite a opção pretendida: ")) - 1
- task.novo_ficheiro = ask_network()
- task.network_data = file.read()
- task.G
- task.node_mapping
- task.origem
- task.destino
- task.algoritmo = ask_which_algorithm()
- task.caminho1
- task.custo1
- task.caminho2
- task.custo2

Index