Documentação Atividade 1

Gabriel Carneiro

Outubro 2020

1 Introdução

O trabalho consiste em uma programa em C que recebe um grafo num arquivo de texto e faz diversas operações com ele.

2 Como utilizar

O programa foi feito para linux e não foi testado em nenhuma outra plataforma, para usar o programa basta executar os seguintes comandos em um terminal:
$ cd caminho/para/a/pasta
$ make
$ ./Grafo

2.1 Entrada

A entrada do programa consiste num arquivo de texto (grafo.txt) que deve ser preenchido de forma que o vértice de origem deve vir primeiro, seguido pelo vertice de destino e então o peso da aresta, todos separados por um espaço, apenas uma aresta por linha do arquivo
Vorig Vdest peso
Vorig Vdest peso
Vorig e Vdest é qualquer numero natural positivo {0,...,n}
peso é qualquer numero inteiro positvo

2.1.1 Exemplo de entrada

```
Entrada grafo.txt
0 1 1
1 2 1
2 3 1
2 4 1
3 0 1
4 5 1
5 6 1
6 4 1
6 7 1
```

2.1.2 Exemplo de saída

```
Terminal:

  A B C D E F G H
A 0 1 0 0 0 0 0 0
B 0 0 1 0 0 0 0 0
C 0 0 0 1 1 0 0 0
D 1 0 0 0 0 0 0 0
E 0 0 0 0 0 1 0 0
F 0 0 0 0 0 0 1 0
G 0 0 0 0 1 0 0 1
H 0 0 0 0 0 0 0 0

0->1->null
1->2->null
2->3->4->null
3->0->null
4->5->null
5->6->null
6->4->7->null
7->null

Numero de vertices: 8
Numero de arestas: 9
```

```
Densidade: 0.16
Grau de saida do vertice 1: 1
Grau de entrada do vertice 1: 1
Sucessores do vertice 1: C
Antecessores do vertice 1: A
Componentes reduzidas por f* conexidade:
0 3 2 1
4 6 5
7

subgrafos.txt
0 3 2 1
4 6 5
7
```

3 Funções

- grafo criaGrafo(); —> Lê do arquivo grafo.txt e chama as funções de criação de uma matriz de adjacência e uma lista de adjacência
- grafo criaListaAdj(grafo g, int val[ ][3]); —> Cria uma lista de adjacência a partir das informações do arquivo
- grafo criaMatrixAdj(grafo g, int val[][3]); —> Cria uma matriz de adjacência a partir das informações do arquivo
- void printaSumario(grafo g); —> Printa o número de vértices, arestas e a densidade do grafo
- void grau(grafo g, int indice); —> Printa o grau de saida e de entrada de um vértice
- void sucessores(grafo g, int indice); —> Printa os sucessores de um vertice
- void antecessores(grafo g, int indice); —> Printa os antecessores de um vertice
- void kosaraju(grafo g); —> Usa o algoritmo de kosaraju para encontrar os subgrafos reduzidos por f conexidade

- void printaGrafo(grafo g); —> Printa o grafo em forma de matriz de adjacência e lista de adjacência
- void dfs(grafo g, int i, FILE *fp); —> Executa uma busca em profundidade