Relatório Sobre o Trabalho de Implementação que Compõe a Terceira Nota da Disciplina Estrutura de Dados II

João Marcello Mendes Moreira

e-mail: joaomarcello.mm@gmail.com

Professor: João Dallyson Sousa de Almeida

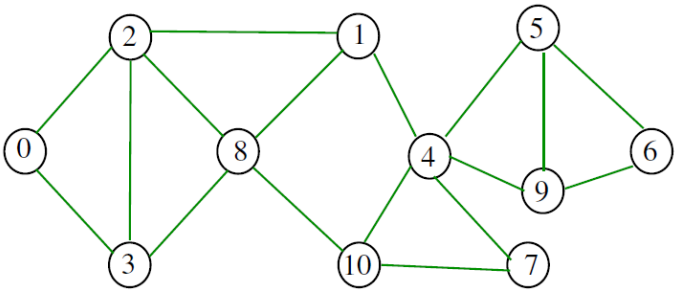
Estrutura de Dados II

Resumo. Um grafo é uma estrutura de dados formada por um conjunto de vértices e um conjunto de arestas. Seu uso mais conhecido seja, talvez, para descobrir o menor caminho entre dois pontos. No trabalho de implementação, pediu-se um programa capaz de ler um grafo a partir de arquivos de texto, realizar a busca em largura e a busca em profundidade dado um vértice inicial informado pelo usuário e, ainda, aplicar o algoritmo de Dijkstra entre outras funcionalidades. Neste relatório, faço uma breve descrição dos grafos, explico o funcionamento do código e ainda apresento o resultado dos testes feitos com o programa utilizando diferentes tipos de grafos.

1. Introdução

Um grafo G (V, E) é uma estrutura definida pelo par de conjuntos V e E onde V é conjunto não vazio que representa os vértices do grafo e E é conjuntos de arestas do grafo. Uma aresta é responsável por conectar dois vértices do grafo. Os grafos podem ser direcionados ou não direcionados e ponderados ou não. A figura 1.1 nos dá um exemplo de grafo não direcionado.

Figura 1.1 Exemplo de grafo não-direcionado



Fonte: Google imagens.

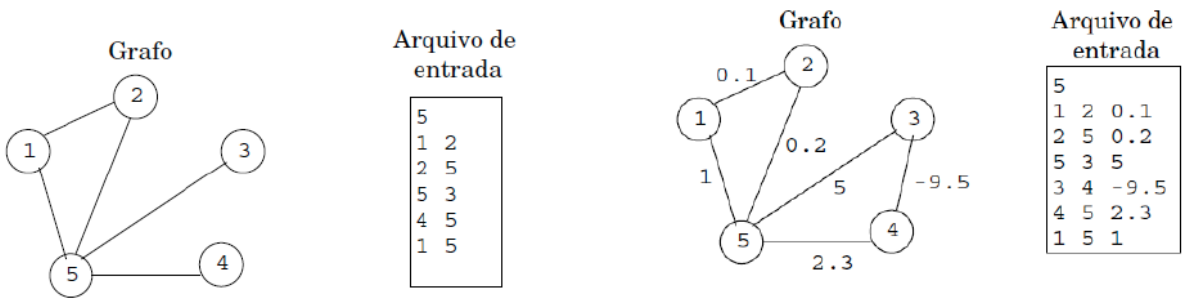
Várias são os problemas que podem ser resolvidos através de uma modelagem em grafos tais como:

- Máquinas de busca para localizar informação relevante na Web.
- Relação entre componentes em circuitos eletrônicos
- Descobrir qual é o roteiro mais curto para visitar as principais cidades de uma região turística.

1.2 O problema

Como atividade, pediu-se a implementação de um programa capaz de carregar grafos, orientados ou não orientados e ponderados ou não ponderados, a partir de documentos de texto. Os vértices do grafo deveriam ser alfanuméricos. A figura 1.2 nos dá alguns exemplos de grafos e suas respectivas representações na forma de arquivo de texto.

Figura 1. 2 Exemplos de grafos e o formato do arquivo



Fonte: Documento da atividade prática.

Depois de gerar o grafo, o programa deveria ser capaz de informar a quantidade de ciclos do grafo, realizar a busca em largura e a busca em profundidade, sendo que o usuário informaria o vértice inicial e o resultado deveria ser mostrado na tela do console ou em arquivo de texto. Ainda, o programa deveria ser capaz de calcular a distância entre todos os pares de vértices usando a busca em largura e retornar a matriz de distâncias. Por fim, também se pedia a implementação do algoritmo de Dijkstra e usá-lo para apresentar o caminho mais curto entre dois vértices informados pelo usuário.

2. Implementação

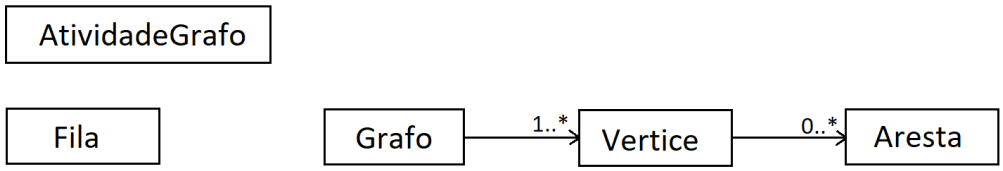
Para a implementação do programa, foi utilizada a linguagem Java e a IDE NetBeans 8.2. Foi utilizada a lista de adjacências para representação dos grafos.

Nesta seção apresento a funcionalidade das classes do programa assim como o funcionamento do mesmo.

2.1 Classes utilizadas no programa

O programa possui as seguintes classes: Grafo, Vertice, Aresta, Fila e AtividadeGrafo. A Fig. 2.1 mostra a organização e a composição das classes do programa. Em seguida, apresento uma breve descrição do funcionamento de cada uma das classes e seus principais métodos.

Fig. 2.1 Organização e composição das classes do programa



Fonte: próprio aluno.

2.1.1 Grafo

Essa classe representa um grafo e, portanto, necessita de dois conjuntos: um para os vértices do grafo e outro para as arestas do grafo. Para instanciar um objeto do tipo Grafo, é necessário passar como parâmetro uma *String* contendo o caminho para o arquivo de texto usado para carregar os dados do grafo, um booleano informando se o grafo é orientado e outro booleano informando se o grafo é ponderado.

Seus principais métodos são o **BFS** que realiza a busca em largura (descobre todos os vértices a uma distância *k* do vértice de origem informado pelo usuário), **DFS** que realiza a busca em profundidade (as arestas são exploradas a partir do vértice *v* mais recentemente descoberto que ainda tem arestas inexploradas saindo dele) e ainda o método **dijkstra** que implementa o algoritmo de Dijkstra (capaz de informar o caminho mais curto entre dois vértices do grafo).

2.1.2 Vertice

Classe que representa um vértice. Possui um atributo *color* que informa a cor do vértice e é usado na busca em largura e profundidade, assim como no algoritmo de Dijkstra. Possui ainda os atributos *value* (o valor do vértice), *pred* (antecessor do vértice), *d* (usado como medida de distância até a origem ou como tempo de descoberta), *f* (tempo de finalização) e um *arrayList* que representa um conjunto de arestas que o conectam o vértice a um outro vértice no grafo.

2.1.3 Aresta

Classe que representa uma aresta. Possui como como atributos *v* (vértice final da aresta), *weigth* (o peso da aresta) e *type* (o tipo da aresta, que é por padrão do tipo árvore e é alterado quando se realiza a busca em profundidade no grafo).

2.1.4 Fila

Classe que implementa uma estrutura de dados conhecida como fila. Essa estrutura permite a inserção e remoção de elementos que acontece da seguinte forma: o primeiro elemento a ser inserido é o primeiro a sair.

2.1.5 AtividadeGrafo

Classe que contém a função *main*.

2.2 Funcionamento do programa

No inicio da execução do programa, é solicitado ao usuário algumas informações que serão necessárias para a construção do grafo que são: o caminho do arquivo de texto usado para gerar o grafo (os arquivos estão na mesma pasta do programa com os nomes de “grafo1.txt”, “grafo2.txt”, etc e representam os grafos utilizados nos testes), dizer se o grafo é orientado e se é ponderado.

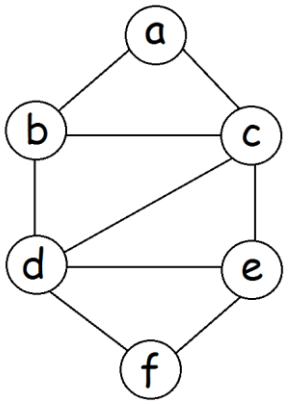
Depois de gerado o grafo, um menu aparece com as seguintes opções: Mostrar a lista de adjacências, informar a quantidade de ciclos (a quantidade de ciclos no grafo é igual a quantidade de arestas do tipo “retorno”), realizar busca em largura, realizar busca em profundidade, mostrar matriz de distâncias (distância entre todos os pares de vértices), mostrar caminho mais curto entre dois vértices (que são informados pelo usuário), realizar algoritmo de dijkstra e sair do programa.

3. Testes

Para os testes, utilizei 4 grafos e apliquei a busca em largura, a busca em profundidade e o algoritmo de Dijkstra (só nos grafos ponderados). No resultado da busca em largura, para cada vértice, são mostrados a distância até o vértice de origem e o predecessor. Já na busca em profundidade, são mostrados o tempo de descoberta e de finalização e o predecessor de cada vértice. Em ambos os tipos de busca, utilizei 3 vértices diferentes para assegurar o resultado. Também mostro a lista de adjacências e a matriz de distância de cada grafo assim como o resultado da aplicação do algoritmo de Dijkstra (também com 3 vértices diferentes) nos grafos ponderados e a quantidade de ciclos de cada grafo. Abaixo, seguem os resultados dos testes.

3.1 Grafo não orientado e não ponderado

Fig. 3.1.1 Grafo 1 - Grafo não orientado e não ponderado



Fonte: próprio aluno.

Fig. 3.1.2 Lista de adjacências do Grafo 1

```
a  --> b --> c
b  --> a --> c --> d
c  --> a --> b --> d --> e
d  --> b --> c --> e --> f
e  --> c --> d --> f
f  --> d --> e
```

Fonte: saída do programa implementado.

Fig. 3.1.3 Resultado da busca em largura no Grafo 1 com o vértice inicial a

Vertice	Distancia	Predecessor
a	0	---
b	1	a
c	1	a
d	2	b
e	2	c
f	3	d

Fonte: saída do programa implementado.

Fig. 3.1.4 Resultado da busca em largura no Grafo 1 com o vértice inicial b

Vertice	Distancia	Predecessor
a	1	b
b	0	---
c	1	b
d	1	b
e	2	c
f	2	d

Fonte: saída do programa implementado.

Fig. 3.1.5 Resultado da busca em largura no Grafo 1 com o vértice inicial c

Vertice	Distancia	Predecessor
a	1	c
b	1	c
c	0	---
d	1	c
e	1	c
f	2	d

Fonte: saída do programa implementado.

Fig. 3.1.6 Resultado da busca em profundidade no Grafo 1 com o vértice inicial a

Vertice	Tempo Desc.	Tempo Final.	Predecessor
a	1	12	---
b	2	11	a
c	3	10	b
d	4	9	c
e	5	8	d
f	6	7	e

Fonte: saída do programa implementado.

Fig. 3.1.7 Resultado da busca em profundidade no Grafo 1 com o vértice inicial b

Vertice	Tempo Desc.	Tempo Final.	Predecessor
a	2	11	b
b	1	12	---
c	3	10	a
d	4	9	c
e	5	8	d
f	6	7	e

Fonte: saída do programa implementado.

Fig. 3.1.8 Resultado da busca em profundidade no Grafo 1 com o vértice inicial c

Vertice	Tempo Desc.	Tempo Final.	Predecessor
a	2	11	c
b	3	10	a
c	1	12	---
d	4	9	b
e	5	8	d
f	6	7	e

Fonte: saída do programa implementado.

Fig. 3.1.9 Matriz de distância do Grafo 1

	a	b	c	d	e	f
a	0	1	1	2	2	3
b	1	0	1	1	2	2
c	1	1	0	1	1	2
d	2	1	1	0	1	1
e	2	2	1	1	0	1
f	3	2	2	1	1	0

Fonte: saída do programa implementado.

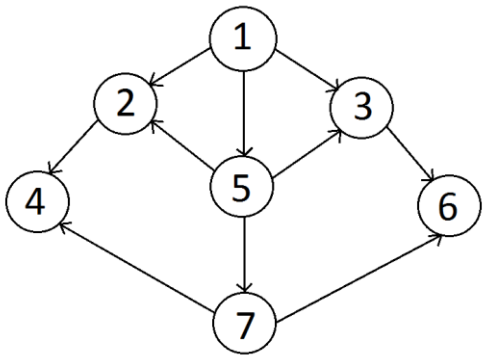
Fig. 3.1.10 Quantidade de ciclos no Grafo 1

```
A quantidade de ciclos do grafo eh 9.
```

Fonte: saída do programa implementado.

3.2 Grafo orientado e não ponderado

Fig. 3.2.1 Grafo 2 - Grafo não orientado e ponderado



Fonte: próprio aluno.

Fig. 3.2.2 Lista de adjacência do Grafo 2

```
1  --> 2 --> 3 --> 5
2  --> 4
3  --> 6
5  --> 2 --> 3 --> 7
6
7  --> 4 --> 6
4
```

Fonte: saída do programa implementado.

Fig. 3.2.3 Resultado da busca em largura no Grafo 2 com o vértice inicial 1

Vertice	Distancia	Predecessor
1	0	---
2	1	1
3	1	1
5	1	1
6	2	3
7	2	5
4	2	2

Fonte: saída do programa implementado.

Fig. 3.2.4 Resultado da busca em largura no Grafo 2 com o vértice inicial 2

Vertice	Distancia	Predecessor
1	-----	---
2	0	---
3	-----	---
5	-----	---
6	-----	---
7	-----	---
4	1	2

Fonte: saída do programa implementado.

Fig. 3.2.5 Resultado da busca em largura no Grafo 2 com o vértice inicial 5

Vertice	Distancia	Predecessor
1	-----	---
2	1	5
3	1	5
5	0	---
6	2	3
7	1	5
4	2	2

Fonte: saída do programa implementado.

Fig. 3.2.6 Resultado da busca em profundidade no Grafo 2 com o vértice inicial 1

Vertice	Tempo Desc.	Tempo Final.	Predecessor
1	1	14	---
2	2	5	1
3	6	9	1
5	10	13	1
6	7	8	3
7	11	12	5
4	3	4	2

Fonte: saída do programa implementado.

Fig. 3.2.7 Resultado da busca em profundidade no Grafo 2 com o vértice inicial 2

Vertice	Tempo Desc.	Tempo Final.	Predecessor
1	13	14	---
2	1	4	---
3	5	8	---
5	9	12	---
6	6	7	3
7	10	11	5
4	2	3	2

Fonte: saída do programa implementado.

Fig. 3.2.8 Resultado da busca em profundidade no Grafo 2 com o vértice inicial 5

Vertice	Tempo Desc.	Tempo Final.	Predecessor
1	13	14	---
2	2	5	5
3	6	9	5
5	1	12	---
6	7	8	3
7	10	11	5
4	3	4	2

Fonte: saída do programa implementado.

Fig. 3.2.9 Matriz de distância do Grafo 2

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

Fonte: saída do programa implementado.

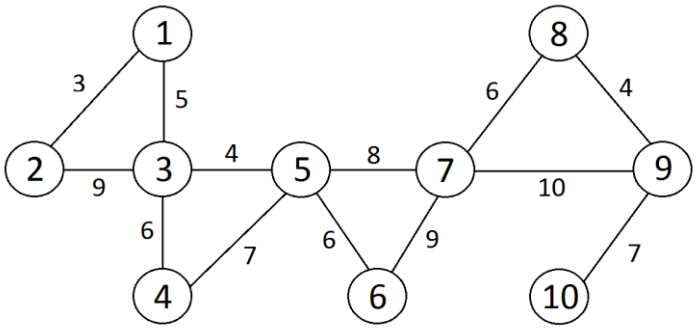
Fig. 3.2.10 Quantidade de ciclos do Grafo 2

```
Grafo aciclico.
```

Fonte: saída do programa implementado.

3.3 Grafo não orientado e ponderado

Fig. 3.3.1 Grafo 3 – Grafo não orientado e ponderado



Fonte: próprio aluno.

Fig. 3.3.2 Lista de adjacência do Grafo 3

```
1  --> 2/3.0 --> 3/5.0
2  --> 1/3.0 --> 3/9.0
3  --> 1/5.0 --> 2/9.0 --> 4/6.0 --> 5/4.0
4  --> 3/6.0 --> 5/7.0
5  --> 3/4.0 --> 4/7.0 --> 6/6.0 --> 7/8.0
6  --> 5/6.0 --> 7/9.0
7  --> 5/8.0 --> 6/9.0 --> 8/6.0 --> 9/10.0
8  --> 7/6.0 --> 9/4.0
9  --> 8/4.0 --> 7/10.0 --> 10/7.0
10  --> 9/7.0
```

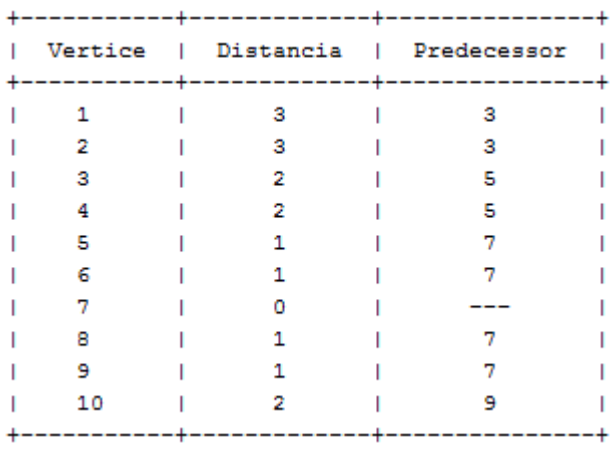
Fonte: saída do programa implementado.

Fig. 3.3.3 Resultado da busca em largura no Grafo 3 com o vértice inicial 3

Vertice	Distancia	Predecessor
1	1	3
2	1	3
3	0	---
4	1	3
5	1	3
6	2	5
7	2	5
8	3	7
9	3	7
10	4	9

Fonte: saída do programa implementado.

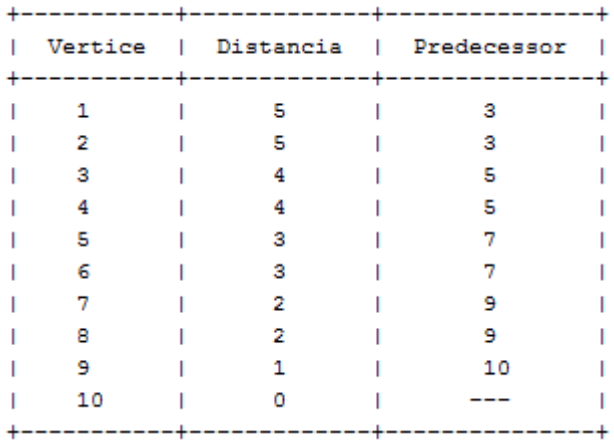
Fig. 3.3.4 Resultado da busca em largura no Grafo 3 com o vértice inicial 7



Vertice	Distancia	Predecessor
1	3	3
2	3	3
3	2	5
4	2	5
5	1	7
6	1	7
7	0	---
8	1	7
9	1	7
10	2	9

Fonte: saída do programa implementado.

Fig. 3.3.5 Resultado da busca em largura no Grafo 3 com o vértice inicial 10



Vertice	Distancia	Predecessor
1	5	3
2	5	3
3	4	5
4	4	5
5	3	7
6	3	7
7	2	9
8	2	9
9	1	10
10	0	---

Fonte: saída do programa implementado.

Fig. 3.3.6 Resultado da busca em profundidade no Grafo 3 com o vértice inicial 3

Vertice	Tempo Desc.	Tempo Final.	Predecessor
1	2	5	3
2	3	4	1
3	1	20	---
4	6	19	3
5	7	18	4
6	8	17	5
7	9	16	6
8	10	15	7
9	11	14	8
10	12	13	9

Fonte: saída do programa implementado.

Fig. 3.3.7 Resultado da busca em profundidade no Grafo 3 com o vértice inicial 7

Vertice	Tempo Desc.	Tempo Final.	Predecessor
1	4	7	3
2	5	6	1
3	3	10	5
4	8	9	3
5	2	13	7
6	11	12	5
7	1	20	---
8	14	19	7
9	15	18	8
10	16	17	9

Fonte: saída do programa implementado.

Fig. 3.3.8 Resultado da busca em profundidade no Grafo 3 com o vértice inicial 10

Vertice	Tempo Desc.	Tempo Final.	Predecessor
1	7	10	3
2	8	9	1
3	6	13	5
4	11	12	3
5	5	16	7
6	14	15	5
7	4	17	8
8	3	18	9
9	2	19	10
10	1	20	---

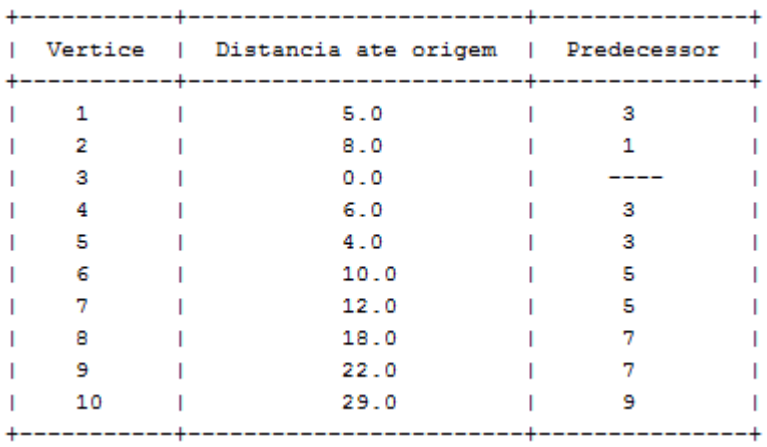
Fonte: saída do programa implementado.

Fig. 3.3.9 Matriz de distâncias do Grafo 3

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	2	2	3	3	4	4	5
2	1	0	1	2	2	3	3	4	4	5
3	1	1	0	1	1	2	2	3	3	4
4	2	2	1	0	1	2	2	3	3	4
5	2	2	1	1	0	1	1	2	2	3
6	3	3	2	2	1	0	1	2	2	3
7	3	3	2	2	1	1	0	1	1	2
8	4	4	3	3	2	2	1	0	1	2
9	4	4	3	3	2	2	1	1	0	1
10	5	5	4	4	3	3	2	2	1	0

Fonte: saída do programa implementado.

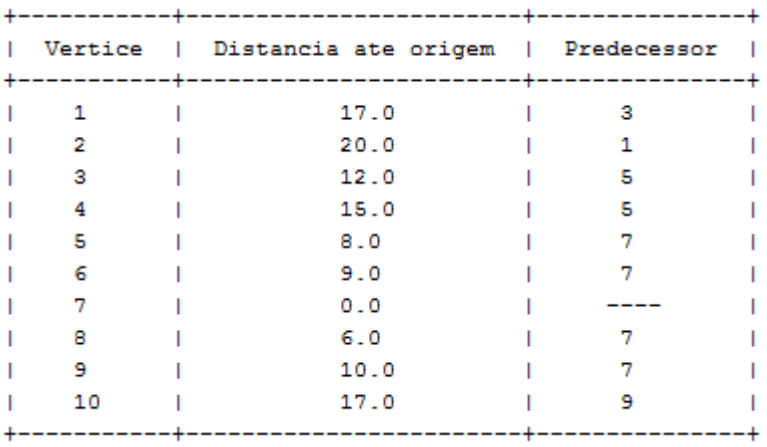
Fig. 3.3.10 Resultado de Dijkstra no Grafo 3 com o vértice inicial 3



Vertice	Distancia ate origem	Predecessor
1	5.0	3
2	8.0	1
3	0.0	----
4	6.0	3
5	4.0	3
6	10.0	5
7	12.0	5
8	18.0	7
9	22.0	7
10	29.0	9

Fonte: saída do programa implementado.

Fig. 3.3.11 Resultado de Dijkstra no Grafo 3 com o vértice inicial 7



Vertice	Distancia ate origem	Predecessor
1	17.0	3
2	20.0	1
3	12.0	5
4	15.0	5
5	8.0	7
6	9.0	7
7	0.0	----
8	6.0	7
9	10.0	7
10	17.0	9

Fonte: saída do programa implementado.

Fig. 3.3.12 Resultado de Dijkstra no Grafo 3 com o vértice inicial 10

Vertice	Distancia ate origem	Predecessor
1	34.0	3
2	37.0	1
3	29.0	5
4	32.0	5
5	25.0	7
6	26.0	7
7	17.0	9
8	11.0	9
9	7.0	10
10	0.0	----

Fonte: saída do programa implementado.

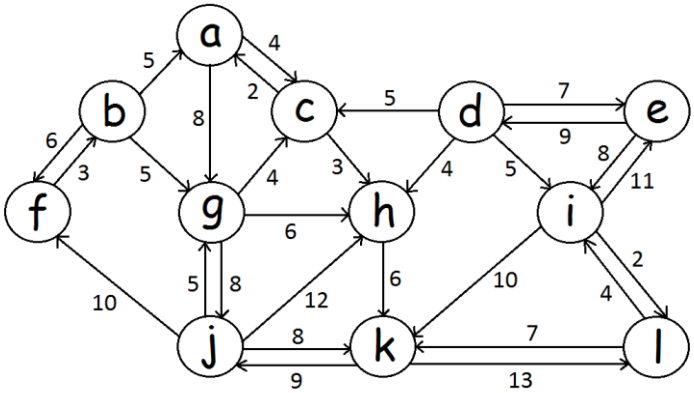
Fig. 3.3.13 Quantidade de ciclos do Grafo 3

```
A quantidade de ciclos do grafo eh 13.
```

Fonte: saída do programa implementado.

3.4 Grafo orientado e ponderado

Fig. 3.4.1 Grafo 4 - Grafo orientado e ponderado



Fonte: próprio aluno.

Fig. 3.4.2 Lista de adjacências do Grafo 4

```
a  --> c/4.0 --> g/8.0
c  --> a/2.0 --> h/3.0
g  --> c/4.0 --> h/6.0 --> j/8.0
b  --> a/5.0 --> f/6.0 --> g/5.0
f  --> b/3.0
h  --> k/6.0
d  --> c/5.0 --> e/7.0 --> h/4.0 --> i/5.0
e  --> d/9.0 --> i/8.0
i  --> e/11.0 --> k/10.0 --> l/2.0
j  --> f/10.0 --> g/5.0 --> h/12.0 --> k/8.0
k  --> j/9.0 --> l/13.0
l  --> i/4.0 --> k/7.0
```

Fonte: saída do programa implementado.

Fig. 3.4.3 Resultado da busca em largura no Grafo 4 com o vértice inicial a

Vertice	Distancia	Predecessor
a	0	---
c	1	a
g	1	a
b	4	f
f	3	j
h	2	c
d	7	e
e	6	i
i	5	l
j	2	g
k	3	h
l	4	k

Fonte: saída do programa implementado.

Fig. 3.4.4 Resultado da busca em largura no Grafo 4 com o vértice inicial g

Vertice	Distancia	Predecessor
a	2	c
c	1	g
g	0	---
b	3	f
f	2	j
h	1	g
d	6	e
e	5	i
i	4	l
j	1	g
k	2	h
l	3	k

Fonte: saída do programa implementado.

Fig. 3.4.5 Resultado da busca em largura no Grafo 4 com o vértice inicial i

Vertice	Distancia	Predecessor
a	4	c
c	3	d
g	3	j
b	4	f
f	3	j
h	3	d
d	2	e
e	1	i
i	0	---
j	2	k
k	1	i
l	1	i

Fonte: saída do programa implementado.

Fig. 3.4.6 Resultado da busca em profundidade no Grafo 4 com o vértice inicial a

Vertice	Tempo Desc.	Tempo Final.	Predecessor
a	1	24	---
c	2	23	a
g	8	9	b
b	7	10	f
f	6	11	j
h	3	22	c
d	16	17	e
e	15	18	i
i	14	19	l
j	5	12	k
k	4	21	h
l	13	20	k

Fonte: saída do programa implementado.

Fig. 3.4.7 Resultado da busca em profundidade no Grafo 4 com o vértice inicial g

Vertice	Tempo Desc.	Tempo Final.	Predecessor
a	3	4	c
c	2	23	g
g	1	24	---
b	9	10	f
f	8	11	j
h	5	22	c
d	16	17	e
e	15	18	i
i	14	19	l
j	7	12	k
k	6	21	h
l	13	20	k

Fonte: saída do programa implementado.

Fig. 3.4.8 Resultado da busca em profundidade no Grafo 4 com o vértice inicial i

Vertice	Tempo Desc.	Tempo Final.	Predecessor
a	5	20	c
c	4	21	d
g	6	19	a
b	11	12	f
f	10	13	j
h	7	18	g
d	3	22	e
e	2	23	i
i	1	24	---
j	9	14	k
k	8	17	h
l	15	16	k

Fonte: saída do programa implementado.

Fig. 3.4.9 Matriz de distâncias do Grafo 4

	a	c	g	b	f	h	d	e	i	j	k	l
a	0	1	1	4	3	2	7	6	5	2	3	4
c	1	0	2	5	4	1	6	5	4	3	2	3
g	2	1	0	3	2	1	6	5	4	1	2	3
b	1	2	1	0	1	2	7	6	5	2	3	4
f	2	3	2	1	0	3	8	7	6	3	4	5
h	5	4	3	4	3	0	5	4	3	2	1	2
d	2	1	3	5	4	1	0	1	1	3	2	2
e	3	2	4	5	4	2	1	0	1	3	2	2
i	4	3	3	4	3	3	2	1	0	2	1	1
j	3	2	1	2	1	1	5	4	3	0	1	2
k	4	3	2	3	2	2	4	3	2	1	0	1
l	5	4	3	4	3	3	3	2	1	2	1	0

Fonte: saída do programa implementado.

Fig.3.4.10 Resultado da aplicação de Dijkstra no Grafo 4 com o vértice inicial a

Vertice	Distancia ate origem	Predecessor
a	0.0	----
c	4.0	a
g	8.0	a
b	29.0	f
f	26.0	j
h	7.0	c
d	50.0	e
e	41.0	i
i	30.0	l
j	16.0	g
k	13.0	h
l	26.0	k

Fonte: saída do programa implementado.

Fig.3.4.11 Resultado da aplicação de Dijkstra no Grafo 4 com o vértice inicial g

Vertice	Distancia ate origem	Predecessor
a	6.0	c
c	4.0	g
g	0.0	----
b	21.0	f
f	18.0	j
h	6.0	g
d	49.0	e
e	40.0	i
i	29.0	l
j	8.0	g
k	12.0	h
l	25.0	k

Fonte: saída do programa implementado.

Fig.3.4.12 Resultado da aplicação de Dijkstra no Grafo 4 com o vértice inicial i

Vertice	Distancia ate origem	Predecessor
a	27.0	c
c	25.0	d
g	23.0	j
b	31.0	f
f	28.0	j
h	24.0	d
d	20.0	e
e	11.0	i
i	0.0	----
j	18.0	k
k	9.0	l
l	2.0	i

Fonte: saída do programa implementado.

Fig. 3.4.13 Quantidade de ciclos do Grafo 4

```
A quantidade de ciclos do grafo eh 16.
```

Fonte: saída do programa implementado.

4. Conclusão

Nesse trabalho, pude implementar uma aplicação que exigia o uso de grafos, que nada mais é do que um conjunto de vértices conectados por arestas. Pude ainda, implementar alguns algoritmos conhecidos na literatura referente aos grafos como a busca em largura, a busca em profundidade e Dijkstra.

Apesar de, a princípio, a estrutura do grafo possa ter parecido complicada, assim como alguns algoritmos exigidos (como o Dijkstra), o desenvolvimento do trabalho fluiu de maneira rápida, já que temos a nossa disposição os pseudocódigos desses algoritmos na literatura. Contudo, uma dificuldade encontrada foi compreender, de fato, alguns desses algoritmos.

Referências

CORMEN, T. H. et al**. Algoritmos Teoria e Prática**. Tradução da segunda edição [americana] Vandenberg D. de Souza. - Rio de Janeiro: Elsevier, 2002.

CARVALHO, M. A. G. **Teoria dos Grafos – Uma introdução**. Universidade Estadual de Campinas – UNICAMP, 2005, São Paulo.

WIKIPÉDIA. **Teoria dos grafos – Wikipédia, a enciclopédia livre**. Disponível em: </https://pt.wikipedia.org/wiki/Teoria_dos_grafos />. Acessado em: 3 de dezembro de 2018.

Notas de aula do professor **João Dallyson Sousa de Almeida.**