COLÉGIO PEDRO II

Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura

Mestrado Profissional em Matemática em Rede Nacional

Thiago Valladares da Silva

CAMINHOS MÍNIMOS EM GRAFOS:
Uma Proposta para Além da Sala de Aula

Rio de Janeiro
2018

Thiago Valladares da Silva

CAMINHOS MÍNIMOS EM GRAFOS:
Uma Proposta para Além da Sala de Aula

Dissertação de Mestrado apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional, vinculado à Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura do Colégio Pedro II, como requisito parcial para obtenção do título de Mestre em Matemática.

Orientador(a): Prof(a). Dra. Patrícia Erthal de Moraes

Rio de Janeiro
2018

COLÉGIO PEDRO II

PRÓ-REITORIA DE PÓS-GRADUAÇÃO, PESQUISA, EXTENSÃO E CULTURA

BIBLIOTECA PROFESSORA SILVIA BECHER

CATALOGAÇÃO NA FONTE

S586 Silva, Thiago Valladares da
Caminhos mínimos em grafos: uma proposta para além da sala de aula / Thiago Valladares da Silva. – Rio de Janeiro, 2018.
77 f.

Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Colégio Pedro II. Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura.
Orientador: Patricia Erthal.

1. Matemática – Estudo e ensino. 2. Grafo. 3. Algorítmo de Dijkstra. 4. Algorítmo de Floyd-Warshall. I. Erthal, Patricia. II. Título.

CDD 510

Ficha catalográfica elaborada pelo Bibliotecário Andre Dantas – CRB7 5026

Thiago Valladares da Silva

CAMINHOS MÍNIMOS EM GRAFOS
Uma Proposta para Além da Sala de Aula

Dissertação de Mestrado apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional, vinculado à Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura do Colégio Pedro II, como requisito parcial para obtenção do título de Mestre em Matemática.

Aprovado em: _____/_____/_____.

Banca Examinadora:

_____
Prof. Dra. Patrícia Erthal de Moraes – Colégio Pedro II

_____
Prof. Dra. Christine Sertã Costa

_____
Prof. Dra. Emília Carolina Santana Teixeira Alves

Rio de Janeiro
2018

AGRADECIMENTOS

Agradeço aos meus pais por terem feito de mim o homem que hoje sou. Se cheguei até aqui, devo isso a eles.

Agradeço aos meus colegas pelas diversas vezes que estudamos, mesmo que à distância por intermédio de um aplicativo de mensagens instantâneas. Em especial, agradeço aos amigos Álvaro Domingos, Vinícius Trevezzini e Denise Tokuyama pelos diversos momentos de descontração que tivemos. A caminhada foi mais fácil por conta deles.

Agradeço a todos os funcionários da PROPGEPEC e a todo corpo docente do PROFMAT/CPII, em especial ao Prof. Dr. Daniel Martins que também me incentivou a prestar o exame de acesso e a minha orientadora e amiga, a Prof. Dra. Patrícia Erthal, que me conduziu na árdua tarefa de escrever este trabalho. Meu muito obrigado a vocês.

Agradeço à Capes pelo fomento fornecido durante o curso, sem o qual seria impossível conclui-lo.

Agradeço a minha amada esposa que me incentivou a prestar a prova de acesso para o mestrado apesar de eu não demonstrar interesse. Agradeço a ela por ter estudado comigo diversas vezes, por compreender a ausência em algumas atividades devido ao tempo destinado aos estudos e por me dar suporte ao longo dessa jornada. Sem ela ao meu lado eu nunca teria chegado aqui.

“Se enxerguei mais longe, foi porque me apoiei sobre ombros de gigantes. ”

- Isaac Newton

RESUMO

SILVA, Thiago Valladares da. **Caminhos Mínimos em Grafos: Uma Proposta para Além da Sala de Aula**. 2018. 77 f. Dissertação (Mestrado) – Colégio Pedro II, Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura, Programa de Mestrado Profissional em Matemática em Rede Nacional, Rio de Janeiro, 2018.

O presente trabalho apresenta conceitos e resultados básicos acerca da Teoria dos Grafos e tem por objetivo fornecer a fundamentação teórica necessária para que o tema possa ser explorado e discutido com alunos do Ensino Médio. A partir da apresentação de um problema aparentemente simples e de fácil entendimento, O Problema de Caminho Mínimo em um Grafo, estuda-se dois algoritmos: o Algoritmo de Dijkstra e o Algoritmo de Floyd-Warshall. Com relação a esse último, é discutido um Problema de Alocação. Ao final do trabalho é apresentada uma proposta de oficina para alunos do Ensino Médio que tem por objetivo desenvolver os algoritmos citados ao longo do Trabalho, aplicando-os na resolução de um problema previamente sugerido. A intenção é que os alunos, a partir de uma situação concreta, construam um modelo matemático aplicando a Teoria de Grafos, utilizem um algoritmo, compreendam passo a passo o seu funcionamento, para assim obterem uma solução desejada.

Palavras-chave: Grafo. Problema de Caminho Mínimo. Algoritmo de Dijkstra. Algoritmo de Floyd-Warshall.

ABSTRACT

SILVA, Thiago Valladares da. **Caminhos Mínimos em Grafos: Uma Proposta para Além da Sala de Aula**. 77 f. Dissertação (Mestrado) – Colégio Pedro II, Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura. Programa de Mestrado Profissional em Matemática em Rede Nacional, Rio de Janeiro, 2018.

This study presents concepts and basic results on the Graph Theory. Its main objective is to provide the necessary theoretical basis so that the theme can be explored and discussed with High School students. Starting from the presentation of an apparently simple and easily understandable problem, the Shortest Path in a Graph problem, we study two algorithms: the Dijkstra Algorithm and the Floyd-Warshall Algorithm. Concerning the latter, it is discussed the Allocation Problem. By the end of the study, it is presented a proposition of a workshop for High School students, which aims to develop the algorithms cited throughout the study, applying them to the solution of a previoulsy suggested problem. The intention is that the students, starting from a concrete situation, construct a mathematical model applying the Graph Theory, use an algorithm, understand each step of its operation, in order to obtain the desired solution.

Keywords: Graph. Short Path Problem. Dijkstra Algorithm. Floyd-Warshall Algorithm.

SUMÁRIO

1 INTRODUÇÃO

Uma busca constante de professores de matemática é a contextualização dos conteúdos de maneira a dar resposta a perguntas como “Onde é que usarei isso professor?” ou “Pra que serve isso?”. Por vezes cria-se uma situação muito artificial para justificar o ensino de determinados conteúdos. De fato, há assuntos que tem pouquíssima ou nenhuma aplicação fora da matemática em si. Contudo, há aqueles que podem ser adaptados com relativa facilidade, de maneira significativa para o aluno. Neste trabalho apresentaremos um tópico que julgamos se encaixar no segundo grupo: a Teoria de Grafos.

A Teoria dos Grafos é um assunto que remonta ao século XVIII e possui diversas aplicações em situações reais. Seu conceito foi introduzido pelo matemático suíço Leonhard Euler em 1736 para resolver um problema que hoje é conhecido como “As sete pontes de Köningsberg”. Este problema constituiu um dos primeiros resultados de uma nova área da matemática, a topologia. De maneira informal, um grafo pode ser interpretado como um conjunto de pontos no plano, denominado vértices, e um conjunto de segmentos, chamados de arestas, que conectam dois destes pontos. É possível modelar diversas situações concretas utilizando grafos, tais como: fluxo de dados em sistemas de computação, malha de tráfego urbano, aplicações em circuitos elétricos, modelos de química orgânica, coloração de mapas entre outras. A simplicidade desse conceito, a quase não existência de pré-requisitos necessários para o seu entendimento e a sua vasta aplicabilidade tornam o assunto bastante atrativo para ser apresentado a alunos do ensino médio.

Atribuindo valores às arestas podemos estabelecer a noção de custo de travessia entre os vértices de um grafo. Dependendo do problema modelado esse custo pode representar um valor monetário, tempo, fluxo de objetos, distância, entre outros. Muitos problemas de otimização, conhecidos como Problema de Caminhos Mínimos, têm por objetivo minimizar esse custo. Existem diversos algoritmos utilizados para esse fim. Neste trabalho abordaremos dois deles, o Algoritmo de Dijkstra e o Algoritmo de Floyd-Warshall.

A fim de introduzir o conceito de grafos para alunos do ensino médio, propomos uma oficina ao final deste trabalho. Nela apresentamos um problema de ordem prática que pode ser modelado utilizando-se um grafo. A partir deste problema é possível apresentar os conceitos iniciais de grafos, os algoritmos supracitados e levá-los a resolver esse problema utilizando-os. Essa oficina foi elaborada em uma sequência de tópicos que julgamos necessários para tal

abordagem. O nosso objetivo é que, ao final da oficina, o aluno tenha desenvolvido um pouco do pensamento algorítmico além de ter se apropriado de um novo conceito matemático.

Este trabalho está estruturado de modo que no capítulo 2 são estudados alguns conceitos introdutórios em Teoria dos Grafos bem como alguns de seus aspectos históricos. No capítulo 3 apresentamos O Problema de Caminhos Mínimos em um Grafo e estudamos os algoritmos de Dijkstra e Floyd-Warshall, descrevendo em detalhes o funcionamento de cada um deles. Afim de aplicarmos a teoria apresentada nos capítulos anteriores, propomos, no capítulo 4, a oficina descrita acima. No capítulo 5 são feitas algumas considerações a respeito do trabalho como um todo, bem como as dificuldades apresentadas em sua elaboração.

2 TEORIA DOS GRAFOS

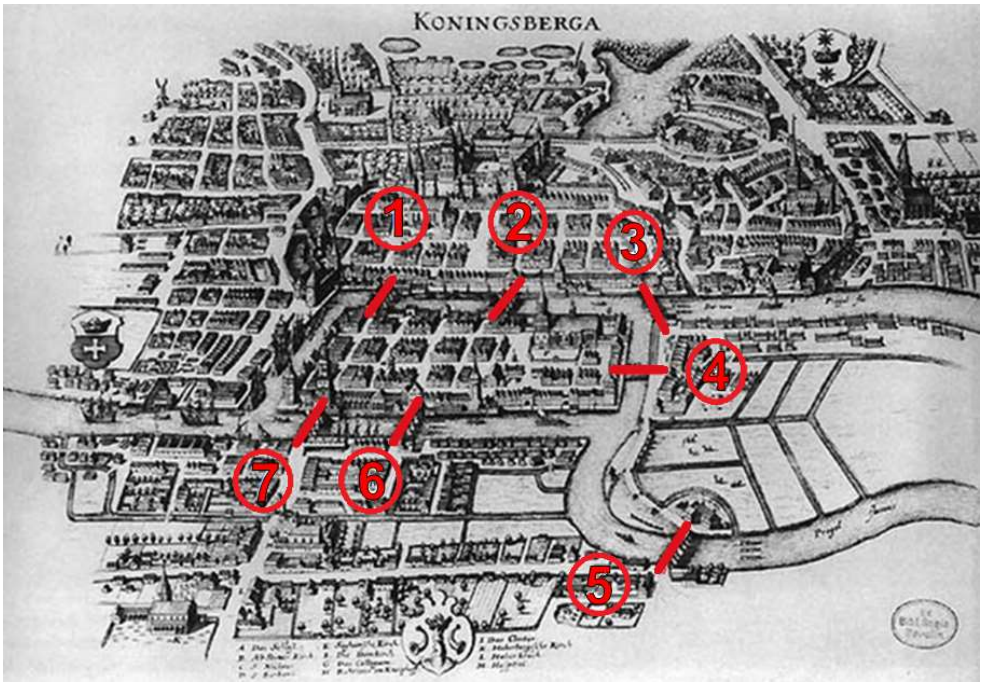
Esse trabalho tem a intenção de ser o mais autocontido possível. Por conta disso foram elencados alguns conceitos e resultados básicos sobre a Teoria de Grafos que consideramos necessários para a compreensão do tema. Além disso, apresentaremos um apanhado histórico de sua origem.

2.1 Resumo Histórico

Nessa seção será feita uma abordagem histórica acerca da Teoria de Grafos que foi elaborada com base em Boaventura e Jurkiewicz (2009).

O primeiro registro de um problema envolvendo grafos remonta ao séc. XVIII. No ano de 1736 o matemático suíço Leonhard Euler se encontrava na cidade de Königsberg (hoje Kaliningrd, na Rússia), onde um problema aparentemente simples, mas ainda sem solução, estava sendo discutido pelos intelectuais do local. No rio Preguel, que cortava a cidade, havia duas ilhas que eram interligadas por uma ponte. Essas ilhas ainda se interligavam às margens do rio por mais seis pontes (ver Figura 1). O problema consistia em encontrar um caminho com início em uma das margens que percorresse as sete pontes, sem repetição, e terminasse no ponto de partida.

Figura 1 – Cidade de Königsberg

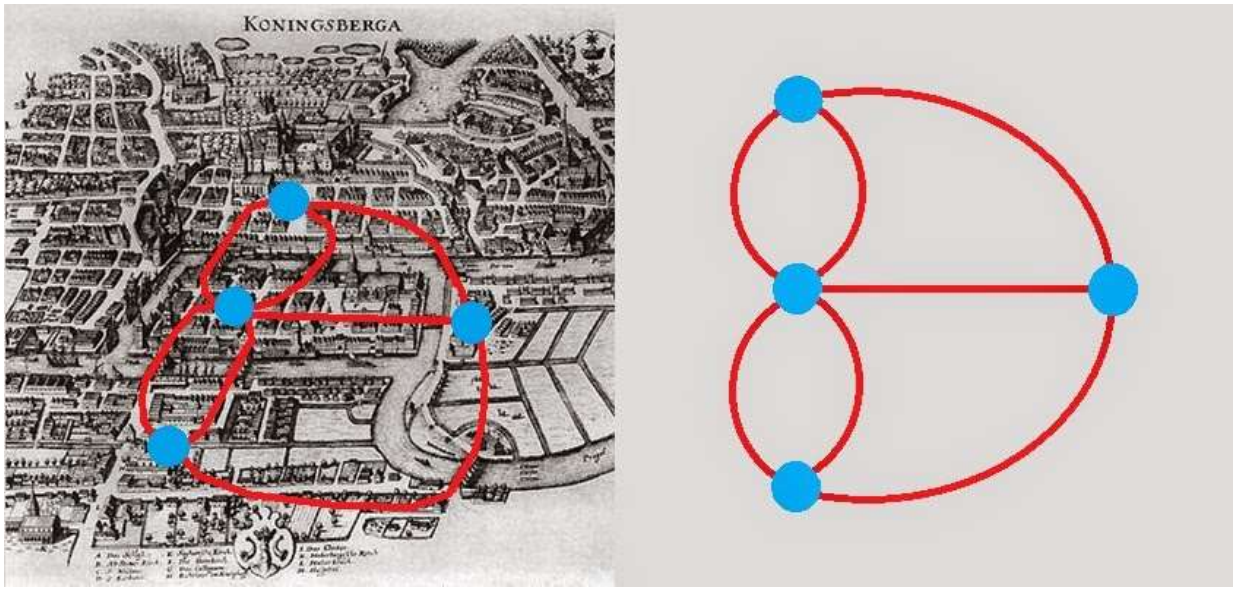


Fonte: http://matematicacomge.blogspot.com.br/2015/01/as-sete-pontes-de-konigsberg.html

Utilizando uma representação simplificada do problema, onde as massas de terra eram representadas por pontos e as pontes por linhas, Euler demonstrou que tal percurso era

impossível. Ele notou que o número de passagens de uma margem para uma ilha, ou entre as ilhas, era ímpar (ver Figura 2).

Figura 2 – Representação das pontes de Königsberg



Fonte: http://matematicacomge.blogspot.com.br/2015/01/as-sete-pontes-de-konigsberg.html

Se o número de pontes com extremidade em uma massa de terra é ímpar, em algum momento se chegará a ela, mas não será possível sair, pois as demais pontes já foram utilizadas. Euler demonstrou que o percurso só seria possível se o número de ligações entre as massas de terra fosse um número par.

O esquema utilizado por Euler é uma representação gráfica do que hoje se chama de *grafo não orientado*, ou simplesmente grafo. É possível compreender um grafo como uma estrutura composta por elementos de dois conjuntos: um ao qual chamamos de conjunto de vértices, e outro, chamado conjunto de arestas, onde descrevemos as relações entre os vértices. Graficamente, representamos os vértices por pontos e as arestas por segmentos de retas que possuem os vértices como extremidade.

Euler resolveu o problema das pontes de Königsberg mas não se preocupou em estudar a fundo ou achar alguma aplicação para esse novo tipo de estrutura. Foi apenas que 1847 que Gustav Robert **Kirchhoff** publicou um resultado envolvendo circuitos elétricos que utilizava um modelo de grafos. Dez anos mais tarde, Arthur **Cayley** utilizou um modelo de grafo para determinar o número de isômeros diferentes de um hidrocarboneto com cadeia aberta.

Francis **Guthrie** também se deparou com grafos ao analisar um suposto fato sobre coloração de mapas. Era de conhecimento de todos os cartógrafos da época que quatro cores eram suficientes para colorir qualquer mapa. Guthrie enunciou esse "fato" como um teorema e tentou prova-lo utilizando grafos. Apesar de seus esforços, Guthrie não logrou êxito, e o teorema, que ficou conhecido como "problema das quatro cores", ficou em aberto por mais de um século.

Somente em 1976, com o uso de computadores que Kenneth Appel e Wolfgang Haken, demonstraram o teorema.

A Teoria de Grafos se desenvolveu com mais intensidade a partir de 1950 com o advento da computação, como explica Boaventura e Jurkiewicz (2009, p. 5):

> A partir da década de 1950 a pesquisa operacional (...) começou a utilizar intensamente os modelos de grafo em busca de melhores soluções para problemas de projeto, organização e distribuição. Essas aplicações, viabilizadas pela invenção do computador, promoveram uma grande divulgação desse tipo de modelagem (...).

A modelagem de problemas por grafos é empregada hoje em diversos campos como gestão de recursos, planejamento de transportes, projetos de processadores eletrônicos, estudo de estrutura de DNA, otimização de recursos humanos, entre outros.

2.2 Definições e Conceitos acerca da Teoria dos Grafos

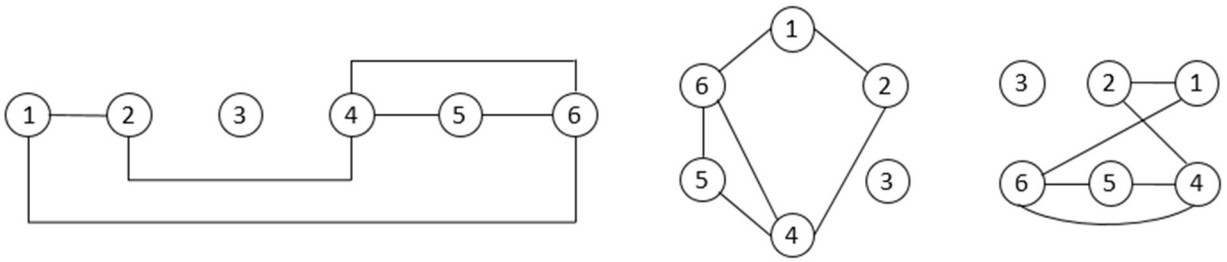
Nesta seção elencaremos alguns conceitos e definições acerca da Teoria de Grafos. Utilizamos como base para esse texto: BOAVENTURA (2009), SCHEINERMAN (2011) e DIESTEL (2000).

Definição 2.1: Um *grafo não orientado*, ou simplesmente um *grafo*, é um par $G = (V, E)$, onde V é um conjunto finito e E é um conjunto de subconjuntos de dois elementos de V. Os elementos de V são chamados de *vértices* do grafo e os elementos de E são as *arestas* do grafo que possuem como extremidades os vértices pertencentes a esse subconjunto de dois elementos.

Para um grafo de n vértices adotaremos $V = \{1,2, \dots, n\}$, $n \in \mathbb{N}$.

Exemplo 2.1: Seja $G = (V, E)$, onde $V = \{1,2,3,4,5,6\}$ e $E = \{\{1,2\},\{2,4\},\{4,5\},\{1,6\},\{4,6\},\{5,6\}\}$. De acordo com a definição 2.1, G é um grafo pois V é finito e E é um conjunto de subconjuntos de V. Nesse exemplo, o grafo possui seis vértices e seis arestas. Como se pode imaginar, é possível representar esse grafo por meio de uma figura. Para tanto assinalamos seis pontos no plano e os rotulamos de 1 a 6. Para cada aresta $e = \{i, j\} \in E$ assinalamos uma curva com extremos em i e j. A figura a seguir representa três diagramas para o mesmo grafo do exemplo 2.1.

Figura 3 – Grafos



Fonte: O autor, 2018

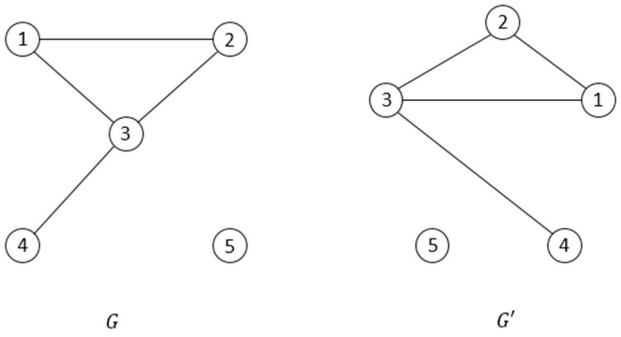
No terceiro diagrama, o fato das linhas se cruzarem não constituem problema, pois a informação transmitida pelo desenho (vértices e ligações entre eles) não se altera. Contudo, os cruzamentos podem tornar a leitura do grafo um pouco mais difícil.

Definição 2.2: Seja $G = (V, E)$ um grafo e $i, j \in V$. Dizemos que i é *adjacente* a j se, e somente se, existe $\{i, j\} \in E$. A notação $i \sim j$ significa que i é adjacente a j.

Definição 2.3 (Isomorfismo): Dois grafos $G = (V, E)$ e $G' = (V', E')$ são isomorfos se existe uma bijeção $\varphi: V \to V'$ que preserva a adjacência dos vértices, isto é, se a aresta $(i, j) \in E$ então, a aresta $(\varphi(i), \varphi(j)) \in E'$. Escrevemos $G \cong G'$ para indicar que os grafos G e G' são isomorfos.

Exemplo 1: Considere os grafos da Figura 4.

Figura 4 – Grafos Isomorfos



Fonte: O autor, 2018

G e G' são isomorfos e a função que garante esse isomorfismo é

$$\varphi: V \to V'$$
$$k \to k$$

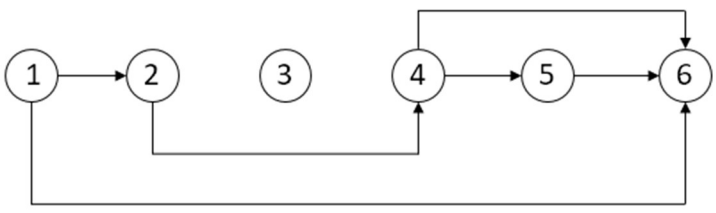
onde $k \in \{1,2,3,4,5\}$.

Definição 2.4 (Digrafo): Um *grafo orientado*, ou *digrafo*, é um par $G = (V, E)$, onde V é um conjunto finito e E é o conjunto de pares ordenados (i, j), com $i, j \in V$.

Apesar da semelhança dessa definição com a Definição 2.1, a situação é completamente diferente. Num grafo não orientado, se existe uma aresta unindo dois vértices i e j dizemos indistintamente que $i \sim j$ e que $j \sim i$. Em um grafo orientado, o par (i, j) indica apenas que $i \sim j$. Em outras palavras: a relação de adjacência não se dá nos dois sentidos. Para representar o sentido da relação graficamente orientamos as arestas por meio de setas. Veja o exemplo a seguir.

Exemplo 2.2: Sejam $V = \{1,2,3,4,5,6\}$ e $E = \{(1,2),(2,4),(4,5),(1,6),(4,6),(5,6)\}$ e o par $G = (V, E)$. Uma representação desse grafo pode ser vista na Figura 5.

Figura 5 – Digrafo



Fonte: O autor, 2018

Definição 2.5 (Vizinhança e Grau de um Vértice): Dado um grafo $G = (V, E)$ e $i \in V$ chama-se *vizinhança* de i, e denotamos por $N(i)$, o conjunto de todos os vértices que são adjacentes a i. Isto é,

$$N(i) = \{j \in V | i \sim j\}$$

Para o grafo do Exemplo 2.1, temos

$$N(1) = \{2,6\} \quad N(2) = \{1,4\} \quad N(3) = \emptyset$$
$$N(4) = \{2,5,6\} \quad N(5) = \{4,6\} \quad N(6) = \{1,4,5\}$$

Para o Exemplo 2.2, temos

$$N(1) = \{2,6\} \quad N(2) = \{4\} \quad N(3) = \emptyset$$
$$N(4) = \{5,6\} \quad N(5) = \{6\} \quad N(6) = \emptyset$$

O *grau* de um vértice v, denotado por $deg(i)$, é o número de vértices adjacentes a v, ou seja,

$$deg(i) = |N(i)|$$

Para o grafo do Exemplo 2.1, temos

$$deg(1) = 2 \quad deg(2) = 2 \quad deg(3) = 0 \quad deg(4) = 3 \quad deg(5) = 2 \quad deg(6) = 3.$$

O resultado a seguir relaciona os graus dos vértices ao número de arestas do grafo.

Teorema 2.1: *A soma dos graus dos vértices de um grafo é igual ao dobro do número de arestas desse grafo. Ou seja:*

$$\sum_{i=1}^{n} deg(i) = 2.|E|$$

Demonstração. Como cada aresta de um grafo incide em dois vértices distintos, ao somarmos os graus dos vértices estaremos contando a mesma aresta em dois momentos, de onde segue a verificação do teorema.

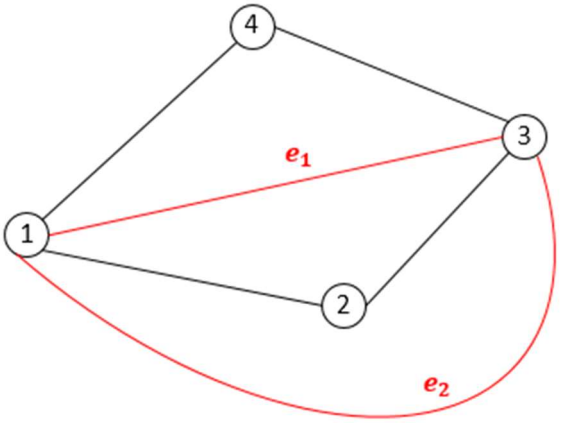
Corolário 2.1: *Todo grafo G possui um número par de vértices de grau ímpar.*

Demonstração: Seja G um grafo. Suponha por absurdo que o número de vértices de grau ímpar seja ímpar. Dessa forma, a soma dos graus desses vértices será um número ímpar I. Por outro lado, a soma dos graus dos vértices de grau par sempre será par. Seja P esse valor. Logo a soma

dos graus de todos os vértices será $I + P$, que é um número ímpar. Absurdo, pois o Teorema 2.1 garante que a soma dos graus é o dobro do número de arestas, portanto, um número par.

Definição 2.6 (Arestas Paralelas e Multigrafo): Duas arestas distintas de um grafo G são paralelas se possuírem os mesmos extremos. Um grafo que possui pelo menos um par de arestas paralelas é chamado de *multigrafo*, conforme a Figura 6.

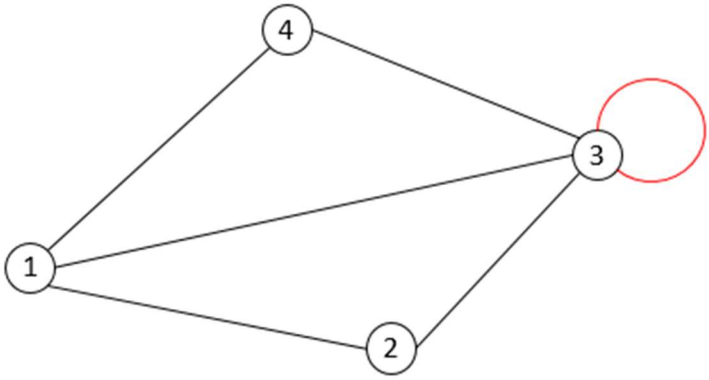
Figura 6 – Grafo com arestas paralelas e_1 e e_2



Fonte: O autor, 2018

Definição 2.7 (Laço): Chama-se laço a aresta que liga um vértice a ele mesmo, conforme Figura 7.

Figura 7 – Grafo com um laço



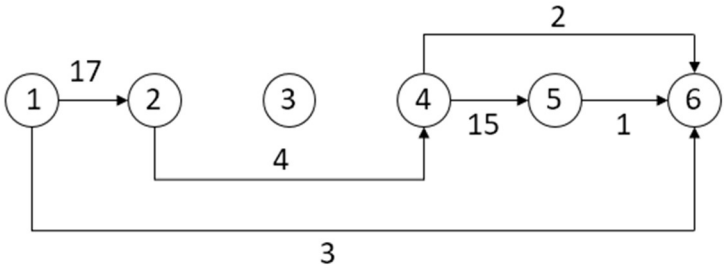
Fonte: O autor, 2018

Grafos que não possuem arestas paralelas nem laços são chamados de *grafos simples*.

Definição 2.8 (Grafo Ponderado ou Valorado): Um grafo é dito *ponderado*, ou simplesmente *valorado*, quando a cada uma de suas arestas é atribuído um valor chamado de *peso*. Geralmente esse valor está associado ao custo da ligação entre os vértices.

Exemplo 2.3: Podemos atribuir pesos ao grafo do exemplo 2.2, conforme a Figura 8.

Figura 8 – Grafos Valorados



Fonte: O autor, 2018

Definição 2.9 (Matriz de Adjacência de um Grafo): Dados um grafo $G = (V, E)$, e $i, j \in V$, chama-se matriz de adjacência de G, denotado por A_G, à matriz

$$A_G = a_{ij} = \begin{cases} 1, se\ i \sim j \\ 0, se\ i \nsim j \end{cases}$$

Essa definição nos diz que é possível organizar a informação das adjacências dos vértices de um grafo em uma matriz onde as entradas são apenas 0 ou 1. Se o vértice i é adjacente ao vértice j temos que $a_{ij} = 1$, caso contrário $a_{ij} = 0$. Para grafos não orientados, A_G será uma matriz simétrica, uma vez que $i \sim j$ implica que j$\sim i$.

Exemplo 2.4: Para os grafos dos exemplos 2.1 e 2.2 temos, respectivamente, as matrizes

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \boldsymbol{e} \quad \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Num grafo valorado há uma outra matriz bastante utilizada chamada *matriz dos pesos* $P = [p_{ij}]$, definida como

$$p_{ij} = \begin{cases} 0, \text{se não existe a aresta } (i, j) \\ k, \text{onde k é o peso da aresta } (i, j) \end{cases}$$

As matrizes de pesos dos grafos representados na Figura 8 são

$$\begin{bmatrix} 0 & 8 & 0 & 0 & 0 & 13 \\ 8 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 & 0 & 4 \\ 13 & 0 & 0 & 5 & 4 & 0 \end{bmatrix} \quad \boldsymbol{e} \quad \begin{bmatrix} 0 & 17 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 15 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

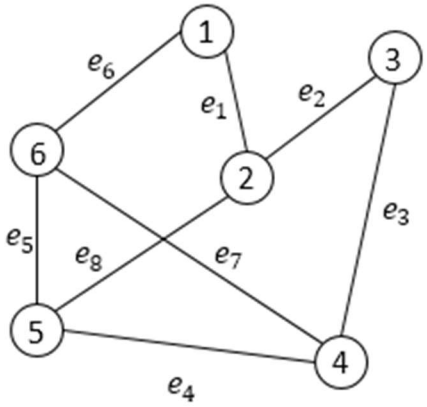
A matriz de adjacência armazena a informação de adjacência relacionando vértice a vértice. Também podemos armazenar essa informação relacionando arestas a vértices por meio de uma matriz, chamada *matriz de incidência*. Vejamos a definição.

Definição 2.10 (Matriz de Incidência): Chama-se *matriz de incidência* do grafo $G = (V, E)$, à matriz $B = [b_{ij}]_{m \times n}$, onde m é o número de vértices e n é o número de arestas G, e

$$b_{ij} = \begin{cases} 1, \text{se a aresta j possui o vértice i como extremidade} \\ 0, \text{se a aresta j não possui o vértice i como extremidade} \end{cases}.$$

A Figura 9 ilustra um exemplo.

Figura 9 – Grafos G e sua matriz de incidência

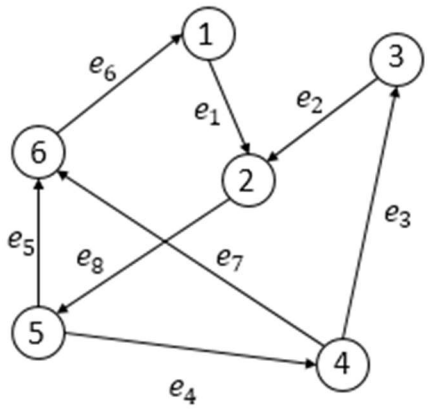


	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
1	1	0	0	0	0	1	0	0
2	1	1	0	0	0	0	0	1
3	0	1	1	0	0	0	0	0
4	0	0	1	1	0	0	1	0
5	0	0	0	1	1	0	0	1
6	0	0	0	0	1	1	1	0

Fonte: O autor, 2018

Cada coluna dessa matriz representa uma aresta e conterá apenas dois elementos 1, uma vez que uma aresta contém apenas dois extremos. As linhas em que esses elementos se encontram indicam quais vértices são extremos dessa aresta. Caso o grafo seja orientado, o vértice-origem do arco é identificado com -1 e o vértice-fim com $+1$, como mostra a Figura 10.

Figura 10 – Grafos orientado e sua matriz de incidência



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
1	-1	0	0	0	0	$+1$	0	0
2	$+1$	$+1$	0	0	0	0	0	-1
3	0	-1	$+1$	0	0	0	0	0
4	0	0	-1	$+1$	0	0	-1	0
5	0	0	0	-1	-1	0	0	$+1$
6	0	0	0	0	$+1$	-1	$+1$	0

Fonte: O autor, 2018

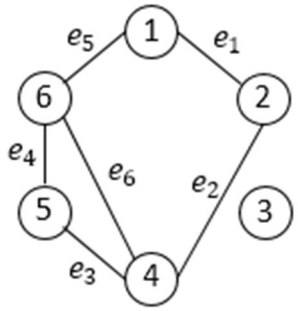
Definição 2.11 (Percurso ou Passeio): Chama-se *percurso* ou *passeio* num grafo $G = (V, E)$ a uma sequência formada por elementos de V e de E, alternadamente

$$(i_1, e_1, i_2, e_2, \dots, , i_{n-1}, e_{n-1}, i_n)$$

de modo que $e_k = \{i_k, i_{k+1}\}$, $k = 1, \dots, n-1$. Em outras palavras, um percurso é determinado por uma coleção de vértices sucessivamente adjacentes.

Considere o grafo da Figura 3. Nomeamos as arestas para descrever um percurso.

Figura 11 – Descrição de percursos



Fonte: O autor, 2018

Podemos destacar, por exemplo, o passeio $\theta_1 = (4, e_6, 6, e_5, 1, e_1, 2, e_2, 4, e_3, 5)$. Note que é possível repetir vértices. Quando o vértice inicial coincide com o vértice final temos um *passeio fechado*. O passeio $\theta_2 = (4, e_6, 6, e_4, 5, e_3, 4)$ é fechado. O *comprimento* de um passeio num grafo não ponderado é igual ao número de arestas que ele possui. Em um grafo ponderado, o comprimento de um passeio é igual a soma dos pesos das arestas que esse passeio possui.

Vejamos agora mais algumas definições obtidas quando alguma restrição é feita ao passeio.

Definição 2.12 (Trilha ou Cadeia): Chama-se *trilha* ou *cadeia* ao passeio em que não há repetição de arestas.

O percurso θ_1, por exemplo, é uma trilha. Note que não há repetição de arestas (apesar do vértice 4 se repetir).

Definição 2.13 (Caminho): Um *caminho* é uma trilha que em que não há repetição de vértices.

O percurso $\theta_3 = (4, e_2, 2, e_1, 1, e_5, 6)$, por exemplo, é um caminho.

Definição 2.14 (Ciclo): Um *ciclo* é um passeio fechado sem repetição de arestas e sem repetição de vértices a menos do inicial e do final. Por exemplo, o percurso fechado θ_2 também é um ciclo. Num grafo ponderado, um ciclo é dito negativo se a soma das arestas que o compõe resultar num número negativo.

Se existe um caminho ligando i a j, partindo de i, dizemos que j é *atingível* a partir de i.

Definição 2.15 (Fecho Transitivo): Num grafo não orientado $G = (V, E)$, o fecho transitivo de um vértice i em G, denotado por $R(i)$, é o conjunto de todos os vértices que são atingíveis a partir de i.

Caso o grafo seja orientado haverá dois tipos desses fechos:

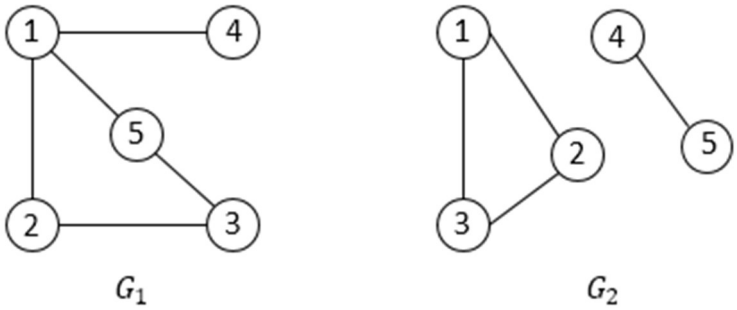
Fecho Transitivo Direto: Seja i um vértice de G. O conjunto de todos os vértices atingíveis a partir de i, denotado por $R^+(i)$, é chamado fecho transitivo direto. Os elementos de $R^+(i)$ são chamados de *descendentes* de i.

Fecho Transitivo Inverso: Seja i um vértice de G. O conjunto de todos os vértices j para os quais i é atingível a partir de j, denotado por $R^-(i)$, é chamado fecho transitivo inverso. Os elementos de $R^-(i)$ são chamados de *ascendentes* de i.

Definiremos a seguir um conceito muito importante em teria dos grafos chamado *conexidade*. Esse conceito classifica os grafos de acordo com as ligações entre seus vértices. Vamos a ele.

Definição 2.16 (Conexidade): Um grafo não orientado é dito *conexo* se existe um caminho que une dois vértices quaisquer desse grafo. Um grafo que não possui essa propriedade é chamado de *desconexo*. A Figura 12 ilustra um exemplo.

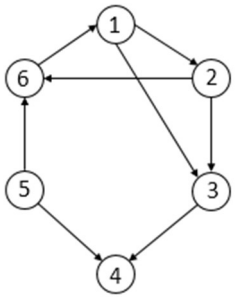
Figura 12 – Grafo conexo G_1 e grafo desconexo G_2



Fonte: O autor, 2018

Observe o grafo orientado apresentado na Figura 13.

Figura 13 – Grafo G



Fonte: O autor, 2018

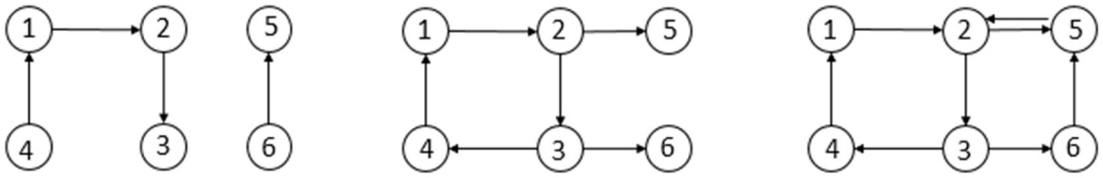
Se escolhermos dois vértices quaisquer neste exemplo, sempre existe um caminho que os une. Contudo, esse caminho não ocorre nos dois sentidos. Tome por exemplo os vértices 6 e 4. Existem um caminho que une 6 a 4, mas não existe nenhum que une 4 a 6 (pois 4 só recebe arcos). Precisamos então estender a definição para grafos orientados.

Para um grafo orientado, temos as seguintes classificações com relação a conexidade: não conexo, simplesmente conexo, semi-fortemente conexo e fortemente conexo. Vamos a elas.

Seja G um grafo orientado.

- G é *não conexo* se, desconsiderando a orientação, existe pelo menos um par de vértices que não é ligado por nenhum tipo de caminho.
- G é *simplesmente conexo* se, desconsiderando a orientação, existem caminhos ligando quaisquer par de vértices.
- G é dito *semi-fortemente conexo* se, para quaisquer vértices u e v existir um caminho de u até v ou de v até u. O grafo da figura 10 é desse tipo.
- G é *fortemente conexo* se, para todo par de vértices u e v, existe um caminho de u até v **e** existe um caminho de v até u. Na figura abaixo mostramos alguns exemplos.

Figura 14 – Grafo desconexo, simplesmente conexo e fortemente conexo



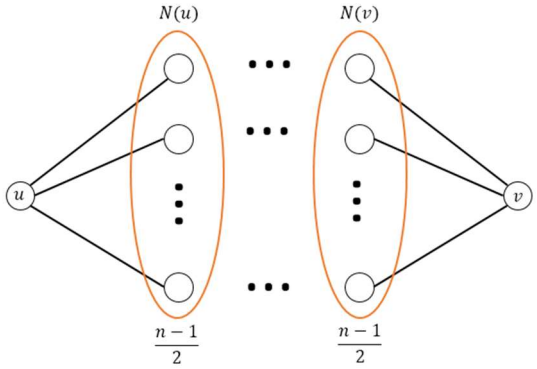
Fonte: O autor, 2018

Enunciaremos agora um teorema que garante a conexidade num grafo simples não orientado.

Teorema 2.2: Seja $G = (V, E)$ um grafo com n vértices. Se $deg(v) \geq \frac{n-1}{2}$ para todo $v \in V$, então G é conexo.

Demonstração: Suponha por absurdo que G seja desconexo. Então existem pelo menos dois vértices $u, v \in V$ que não estão conectados. De cada um destes vértices partem pelo menos $\frac{n-1}{2}$ arestas, os conectando a pelo menos $\frac{n-1}{2}$ vértices distintos como mostra a Figura 15.

Figura 15 – Vértices adjacentes a u e vértices adjacentes a v



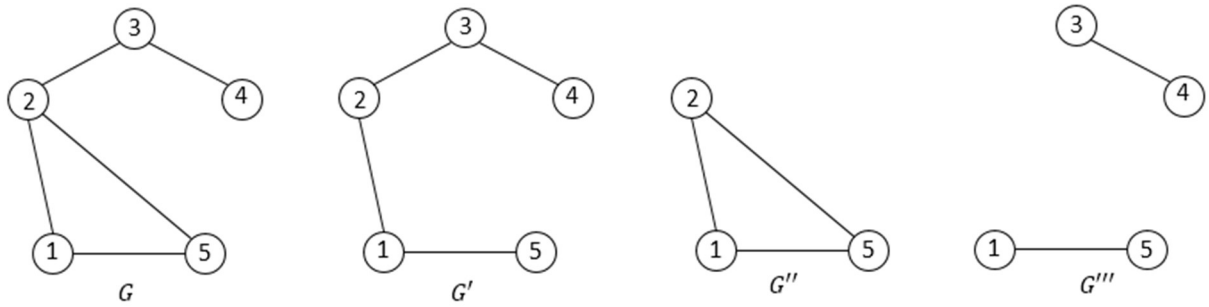
Fonte: O autor, 2018

Como u e v não estão conectados $N(u) \cap N(v) = \emptyset$. Observando a figura 12, temos que G possui, no mínimo, $2 + 2 \cdot \left(\frac{n-1}{2}\right) = n + 1$ vértices, o que é um absurdo. Logo, G é conexo.

Definição 2.17 (Subgrafo): Considere os grafos $G = (V, E)$ e $G' = (V', E')$. Se G' é tal que $V' \subseteq V$ e $E' \subseteq E$ dizemos que que G' é um *subgrafo* de G e $G' \subseteq G$.

Naturalmente, um grafo é subgrafo de si próprio. Informalmente dizemos que G contém G'. G é chamado de *supergrafo* de G'. A figura 5 mostra um grafo e alguns exemplos de subgrafo.

Figura 16 – Grafo G e alguns subgrafos.



Fonte: O autor, 2018

O subgrafo G' e G'' são conexos. O primeiro é obtido pela eliminação da aresta $\{2,5\}$ enquanto o segundo é obtido pela deleção dos vértices 3 e 4 e todas as arestas que neles incidem. Já G''' é desconexo e foi obtido pela supressão do vértice 2 e suas arestas. Um vértice pode ter todas as suas arestas deletadas e ser mantido. O subgrafo resultante terá um vértice isolado e será desconexo.

Mostraremos agora que o problema inicial do passeio pela cidade de Koningsberg não possui solução. Mas primeiro definiremos Trilha Euleriana e Grafo Euleriano.

Definição 2.18 (Trilha Euleriana e Grafo Euleriano): Uma trilha fechada é dita *Euleriana* se atravessa todas as arestas do grafo. Um grafo é dito Euleriano se possui uma Trilha Euleriana.

Teorema 2.3 (Caracterização de Grafos Eulerianos): Um grafo conexo é Euleriano se, e somente se, cada um de seus vértices possui grau par.

Demonstração:

(⇒) Se um grafo possui uma trilha euleriana, significa que, para percorrer todas as arestas, a trilha "entrou" e "saiu" pelo menos uma vez de cada vértice. Logo, todos os vértices têm grau par.

(⇐) Seja G um grafo conexo onde os graus de todos os vértices sejam pares e seja

$$T = (i_1, e_1, i_2, e_2, \dots, i_{n-1}, e_{n-1}, i_n)$$

a trilha de maior comprimento em G. Como T não pode ser estendido, ele contém todas as arestas incidentes em i_1 e i_n. Por hipótese, o número de arestas é par. Logo, devemos ter forçosamente $i_1 = i_n$ e T é uma trilha fechada. Suponha que T não seja uma trilha euleriana. Então G possui uma aresta e fora de T mas incidente em um vértice de T, uma vez que G é conexo. Sem perda de generalidade seja $e = \{u, i_2\}$. Então o passeio

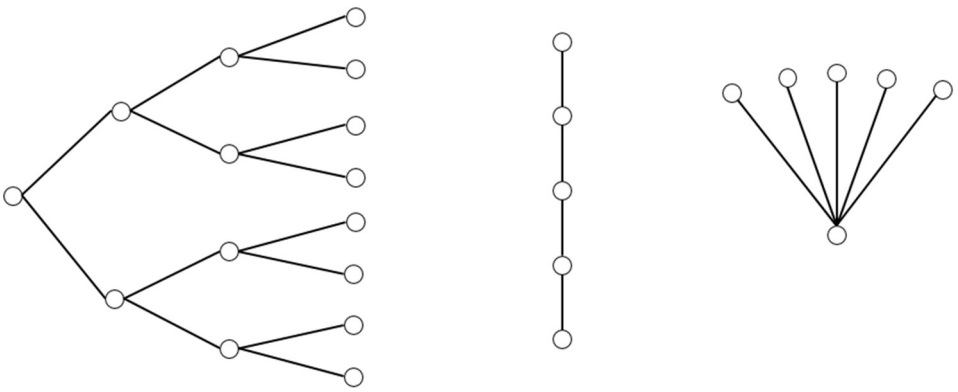
$$T' = (u, e, i_2, e_2, \dots, , i_{n-1}, e_{n-1}, i_n, e_1, i_2)$$

possui comprimento maior do que T. Absurdo.

Dessa forma fica evidente que o problema das pontes de Köningsberg, introduzido na seção 2.1, não possui solução, uma vez que três de seus vértices possuem graus ímpar.

Definição 2.19 (Árvores): Uma *árvore* é um grafo conexo e sem ciclos. Note que uma árvore é um grafo simples, uma vez que arestas paralelas e laços formam ciclos. A figura abaixo mostra alguns exemplos de árvores.

Figura 17 – Exemplos de árvores.



Fonte: O autor, 2018

A estrutura de uma árvore está presente em diversas situações do cotidiano. Ela pode ser usada para representar, por exemplo, o organograma de uma empresa ou um fluxograma de tomada de decisão. O teorema a seguir nos dará uma maneira de caracterizar uma árvore.

Teorema 2.4: Um grafo G com n vértices é uma árvore se:

i. existe exatamente um caminho simples entre cada par de vértices ou;
ii. é conexo e possui $n-1$ arestas ou;
iii. é conexo e a deleção de qualquer aresta torna o grafo desconexo.

Demonstração:

i) Se existe exatamente um caminho entre cada par de vértices então G é conexo. Suponha agora que G não seja uma árvore. Então, G possui pelo menos um ciclo. Mas em todo par de vértices de um ciclo existem pelo menos dois caminhos, absurdo. Logo, G é uma árvore.

ii) A prova se dá por indução. Para $n = 1$ a verificação é imediata. Suponha agora que G seja uma árvore com n vértices e $n-1$ arestas. Seja G' o grafo obtido a partir de G ao acrescentar mais um vértice v_{n+1} de grau 1 a ele. Devemos mostrar que G' é uma árvore. De fato, o acréscimo de v_{n+1} a G não gera um ciclo pois v_{n+1} possui grau 1 e G' é conexo pois v_{n+1} está conectado a G que é conexo. Logo, G' é uma árvore, o que encerra a prova.

iii) Suponha por absurdo que G não seja uma árvore. Como G é conexo e não é uma árvore há um ciclo em G. Mas se há um ciclo em G, a deleção de uma aresta neste ciclo não o torna desconexo. Absurdo. Logo, G é uma árvore.

No capítulo seguinte veremos como podemos determinar a menor distância entre dois vértices de um grafo ponderado por meio de dois algoritmos: O Algoritmo de Dijkstra e o Algoritmo de Floyd-Warshall

3 O PROBLEMA DE CAMINHO MÍNIMO EM UM GRAFO

Em Teoria dos Grafos, o Problema de Caminho Mínimo consiste em determinar a sequência de vértices que minimiza o custo de travessia entre dois vértices de um grafo valorado. Esse custo é determinado pela soma dos pesos das arestas que o compõe. Se não houver nenhum caminho que una os vértices em questão o problema não possui solução. Se o vértice de origem coincide com o vértice de destino e as arestas tem pesos não negativos[1], a solução é única e tem comprimento zero. Utilizaremos **distância** para designar o custo de um caminho.

Conforme Boaventura e Jurkiewicz (2009), podemos subdividir este problema em, pelo menos, três categorias de subproblemas de caminho mínimo:

- Origem Única: determinar a menor distância entre um vértice e os demais;
- Origem-Destino: determinar a menor distância entre dois vértices dados;
- Par a Par: determinar a menor distância entre cada par de vértices do grafo.

Existem diversos algoritmos especializados em determinar o caminho mínimo em um grafo. Como exemplo temos o Algoritmo de Bellmand-Ford, Algoritmo de Johnson, Algoritmo de Dijkstra e Algoritmo de Floyd-Warshall. Neste capítulo abordaremos o funcionamento e aplicações dos dois últimos, que tem enfoque nos subproblemas Origem-Destino e Par a Par respectivamente.

3.1 O Algoritmo de Dijkstra[2]

Edsger Wybe Dijkstra foi um matemático Holandês pioneiro no setor de programação computacional. Sua carreira como cientista de computação teve início em 1951. Após ter sido aprovado no exame de admissão para cursar física e matemática na *University of Leiden*, Dijkstra participou de um curso de programação em Cambrige, Inglaterra. Em 1952, o diretor do *Mathematisch Centrum* de Amsterdã, Aad van Wijngaarden, tomou conhecimento de sua participação no curso e ofereceu a ele uma vaga de meio período como programador.

Em 1955, após três anos de trabalho como programador, Dijkstra decidiu abandonar a física para se tornar um programador em tempo integral, como ele mesmo relata:

[1] Alguns problemas em grafos, como aqueles que envolvem fluxos, utilizam arestas com pesos negativos em sua modelagem.

[2] Esta seção foi composta de que informações foram compiladas de diversas fontes de pesquisa entre elas: jornais, obituário, entrevista etc.

> (…) Eu conclui que o desafio intelectual de programar era muito maior que o desafio intelectual de física teórica e, como resultado, eu escolhi programar. Programar era tão implacável. Se alguma coisa desse errado, quero dizer, um zero é um zero e um um é um um. Eu nunca usei o software de outra pessoa. Se alguma coisa desse errado, eu era o responsável. E foi isso que me desafiou. (DIJKSTRA, 2010, p. 42, tradução nossa)[3]

Após tomar essa decisão Dijkstra terminou seus estudos o mais rápido possível, se graduando em 1956, uma vez que não se sentia mais bem-vindo na universidade:

> (…) os físicos me consideraram um desertor e os matemáticos se mostraram desdenhosos com relação a computação. Na cultura matemática daquele tempo seu trabalho tinha que lidar com o infinito para ser cientificamente respeitável. (DIJKSTRA, 2010, p. 42, tradução nossa)

No mesmo ano, a construção do segundo computador automático do *Mathematisch Centrum*, o ARMAC, estava prestes a ser concluída. Para celebrar sua inauguração foi preparada uma apresentação que deveria ser simples o suficiente para que pessoas leigas em matemática pudessem entender. Dijkstra então projetou um programa que determina a menor distância entre duas cidades da Holanda, considerando uma malha ferroviária simplificada contendo apenas 64 cidades. Nesse programa ele utilizou um algoritmo que desenvolveu, segundo ele próprio, em vinte minutos durante um passeio com sua noiva, enquanto estavam sentados descansando após uma caminhada. A demonstração foi um grande sucesso e esse algoritmo, que hoje é amplamente utilizado, leva seu nome.

Dijkstra trabalhou como programador no *Mathematisch Centrum* até 1962, quando foi indicado para uma cadeira na *Eindhoven University of Technology* onde permaneceu até 1973. Na universidade, desenvolveu um sistema operacional para o sistema de computadores interno. A elegância e estrutura hierárquica do sistema foi reconhecida e tópico de interesse no 1º Simpósio de Princípios de Sistemas Operacionais.

Em 1973, Dijkstra deixou a universidade para se tornar um pesquisador associado na *Burroughs Corporation*, onde obteve total liberdade para efetuar sua pesquisa da maneira que desejasse.

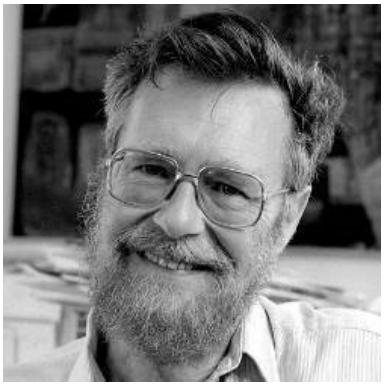
Em 1984 decidiu voltar a lecionar e aceitou a *Schlumberger Centennial Chair* da *The University of Texas at Austin* onde permaneceu até sua aposentadoria em 1999.

Suas contribuições cobrem diversas áreas da ciência da computação, incluindo sistemas operacionais, pesquisa em linguagens de programação, programação sequencial e concorrente, algoritmos para grafos entre outras. Ele ajudou a modelar a ciência da computação como ela é conhecida hoje. Muitas de suas publicações foram fonte de pesquisa para novas áreas de

[3] Embora Dijkstra tenha falecido em 2002, a publicação desta entrevista foi feita somente em 2010.

conhecimento. Diversos conceitos e problemas que hoje são padrões em ciência da computação foram primeiramente identificados e/ou levam nomes cunhados por ele.

Figura 18 – Edsger W. Dijkstra



Fonte: Dijkstra, 2010

3.1.1 Funcionamento do Algoritmo de Dijkstra

Seja $G = (V, F)$ um grafo ponderado com arestas de pesos não negativos. O Algoritmo de Dijkstra oferece uma solução para determinar o menor caminho entre dois vértices de G. Fixado o vértice de início i e vértice final f o algoritmo determina durante sua execução quais são os vértices de G que devem ser inseridos entre i e f para que se tenha o menor caminho.

O funcionamento do algoritmo pode ser dividido em três etapas. Na primeira etapa ele cria as variáveis que serão atualizadas ao longo da execução:

- $d(j)$: é a menor distância do vértice de origem i ao vértice j,
- $a(j)$: indica qual é o vértice que antecede o vértice j, na sequência de vértices que vai compor o caminho desejado. Quando não houver vértice antecedente, representaremos sua ausência com o algarismo zero.
- Conjunto dos vértices fechados, F: é o conjunto que contém os vértices para os quais a menor distância já foi determinada;
- Conjunto dos vértices abertos, A: é o conjunto que contém os vértices para os quais a menor distância ainda não foi determinada durante a execução do algoritmo. Note que $A = V - F$.

Seja $P = [p_{qk}]$ a matriz dos pesos do grafo atribuídos. Considere $M = [m_{qk}]$ obtida a partir da matriz P, onde

$$m_{qk} = \begin{cases} p_{qk}, \text{se } q \text{ é adjacente a } k \\ 0, \text{se } q = k \\ \infty, \text{se } q \text{ não é adjacente a } k \end{cases}$$

Utilizamos o símbolo ∞ (infinito) para diferenciar uma aresta de peso zero entre q e k da não existência de uma aresta entre esses vértices. Chamaremos M de *Matriz de Manejo*.

Inicialmente, $d(i) = a(i) = 0$, $F = \{i\}, A = V - \{i\}$ e, para todo $j \in A$, temos

$$d(j) = \begin{cases} m_{ij}, \text{se } m_{ij} \neq \infty \\ \infty, \text{se } m_{ij} = \infty \end{cases}, a(j) = \begin{cases} i, \text{se } m_{ij} \neq \infty \\ 0, \text{se } m_{ij} = \infty \end{cases}.$$

Na segunda etapa o algoritmo utiliza a matriz M e procura iterativamente o vértice $r \in A$ mais próximo a i. Esse vértice, chamado "vértice da vez", é então movido do conjunto A para o conjunto F. Em seguida, para cada vértice $s \in A \cap R^+(r)$ adjacente ao vértice da vez, o algoritmo verifica se $d(s)$ é maior que a $d(r) + m_{rs}$. Em caso afirmativo, o caminho do vértice inicial até o vértice s, passando por r é menor que o caminho previamente existente. Nesse caso, a entrada $d(s)$ do vértice s é atualizada com o menor valor e a entrada $a(s)$ é atualizada com r. O algoritmo executa essa rotina até que f o vértice da vez.[4]

Na terceira e última etapa o algoritmo criará (se possível) um vetor que contém a sequência de vértices que descreve o menor caminho de i até f. Quando o vértice f for o vértice da vez teremos uma das seguintes situações:

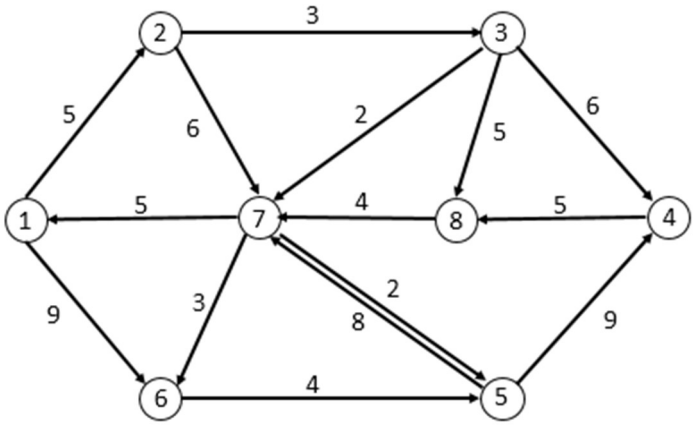
- O valor da entrada $a(f)$ é nulo. Se isso ocorrer significa que nenhuma das iterações anteriores encontrou um caminho de i até f. Logo o valor $d(f)$ não foi alterado e permanece infinito, de onde segue que o vértice f não é adjacente de nenhum outro vértice do grafo.
- O valor da entrada $a(f)$ é não nulo. Nesse caso, f é adjacente a algum vértice para o qual é possível traçar um caminho até ele a partir de i. O algoritmo então executa uma rotina para varrer os vértices anteriores partindo de f, criando um vetor C que determina caminho mínimo.

Fixado um vértice inicial i, o algoritmo busca iterativamente o primeiro vértice aberto mais próximo, o segundo mais próximo, o terceiro mais próximo e assim sucessivamente, até encontrar o vértice desejado f. Como a busca é feita pelo menor caminho e as arestas tem pesos não negativos, uma vez que o vértice esteja fechado, não há um caminho com menor distância até ele.

Apresentamos o pseudocódigo e o fluxograma que descrevem o funcionamento do algoritmo no Apêndice A e B respectivamente. Para ilustrar o funcionamento do Algoritmo de Dijskstra, como explicado acima, vamos determinar o menor caminho entre os vértices 1 e 5 no grafo G apresentado na Figura 19.

[4] É possível escrever o algoritmo de Dijkstra de maneira que ele obtenha a menor distância de um vértice a todos os outros, o que o caracterizaria como Origem Única. Para os fins deste trabalho, escolhemos uma versão que se encerra ao encontrar a menor distância até um vértice específico.

Figura 19 – Grafo G.



Fonte: O autor, 2018

Baseado no grafo G temos as seguistes matrizes de Peso e de Manejo:

Figura 20: Matriz de Peso e de Manejo do Grafo G.

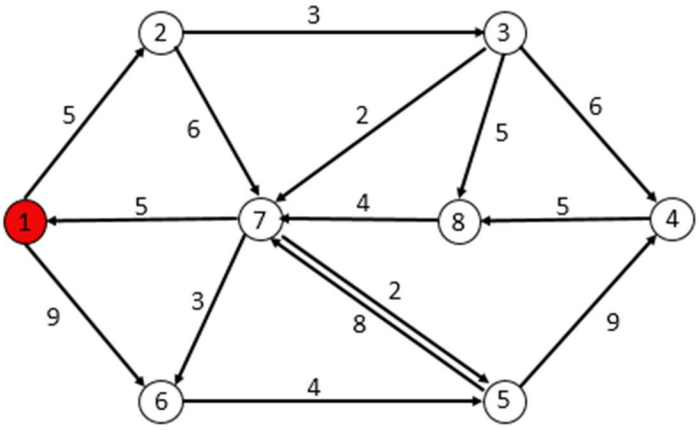
P	1	2	3	4	5	6	7	8
1	0	5	0	0	0	9	0	0
2	0	0	3	0	0	0	6	0
3	0	0	0	6	0	0	2	5
4	0	0	0	0	0	0	0	5
5	0	0	0	9	0	0	2	0
6	0	0	0	0	4	0	0	0
7	5	0	0	0	2	3	0	0
8	0	0	0	0	0	0	7	0

M	1	2	3	4	5	6	7	8
1	0	5	∞	∞	∞	9	∞	∞
2	∞	0	3	∞	∞	∞	6	∞
3	∞	∞	0	6	∞	∞	2	5
4	∞	∞	∞	0	∞	∞	∞	5
5	∞	∞	∞	9	0	∞	2	∞
6	∞	∞	∞	∞	4	0	∞	∞
7	5	∞	∞	∞	2	3	0	∞
8	∞	∞	∞	∞	∞	∞	7	0

Fonte: O autor, 2018

A sequência de figuras 21 a 26 mostram o desenvolvimento do algoritmo. Inicialmente é preciso criar as variáveis que serão atualizadas em cada iteração. Atribuímos zero a $d(1)$ e $a(1)$. Se $m_{1j} = 0$, definimos $d(j) = \infty$ e $a(j) = \{\}$. Caso contrário fazemos $d(j) = p_{1j}$ e $a(j) = \{1\}$. Por último, definimos o Conjunto dos Vértices Abertos e o Conjunto dos Vértices Fechados (ver Figura 13).

Figura 21 – Inicialização do Algoritmo de Dijkstra



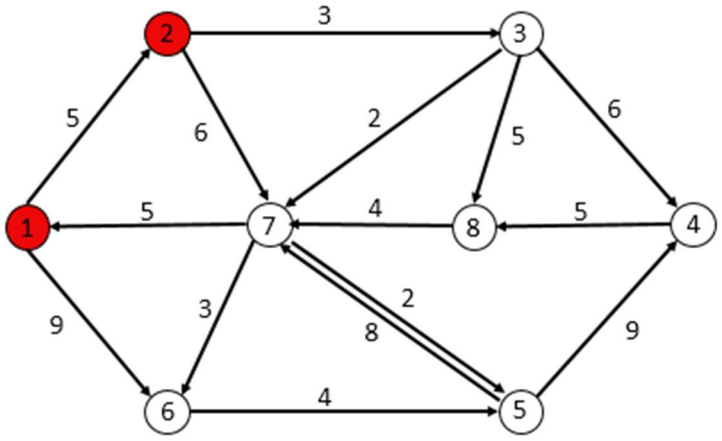
j	1	2	3	4	5	6	7	8
$d(j)$	0	5	∞	∞	∞	9	∞	∞
$a(j)$	-	1	-	-	-	1	-	-

$F = \{1\}$; $A = \{2,3,4,5,6,7,8\}$

Fonte: O autor, 2018

A partir do vértice de origem 1 devemos procurar o vértice mais próximo. Nesse caso o vértice mais próximo é 2, que então é fechado. Para cada vértice aberto adjacente a ele verificamos se sua distância até a origem é maior que a distância da origem até ele passando por 2. Como não há nenhum arco unindo 3 e 7 à origem, a estimativa passando por 2 é melhor e as variáveis desses vértices devem ser atualizadas (Figura 22).

Figura 22 – Algoritmo de Dijkstra após primeira iteração.



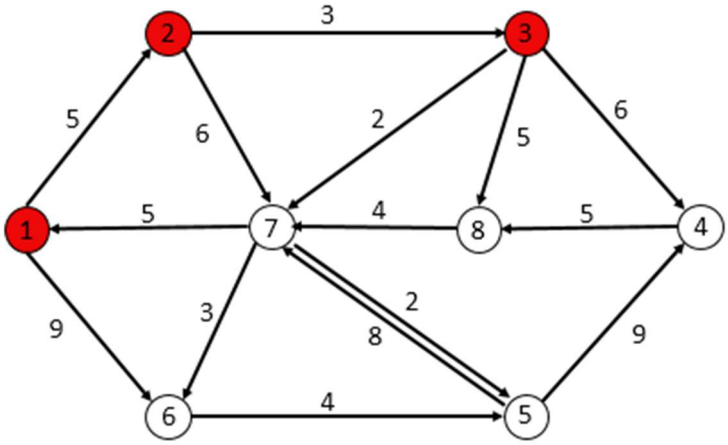
j	1	2	3	4	5	6	7	8
$d(j)$	0	5	8	∞	∞	9	11	∞
$a(j)$	-	1	2	-	-	1	2	-

$F = \{1,2\}$; $A = \{3,4,5,6,7,8\}$

Fonte: O autor, 2018

O próximo vértice a ser fechado é 3, pois possui a menor distância $d(j)$, $j \in A$. Como 4 e 8 não possuem estimativa de menor distância, o caminho passando por 3 é menor. Como $d(7) > d(3) + m_{37}$, melhoramos a estimativa para o menor caminho de 1 para 7 e fazemos as devidas alterações.

Figura 23 – Algoritmo de Dijkstra após terceira iteração.



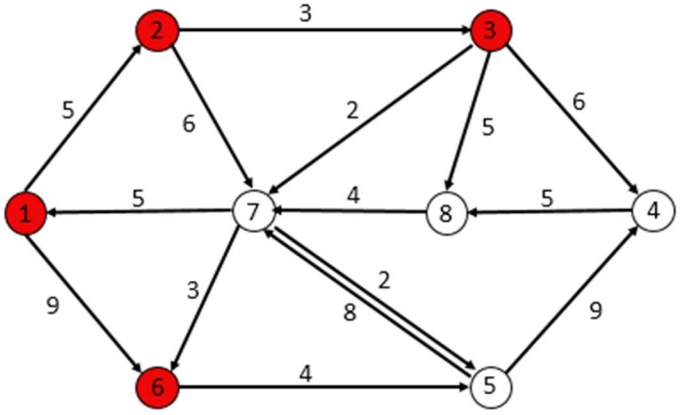
j	1	2	3	4	5	6	7	8
$d(j)$	0	5	8	14	∞	9	11	13
$a(j)$	-	1	2	3	-	1	3	3

$F = \{1,2,3\}$; $A = \{4,5,6,7,8\}$

Fonte: O autor, 2018

Prosseguimos procurando o vértice aberto mais próximo da origem. Nessa iteração fecharemos o vértice 6. O único vértice adjacente a 6 é 5. Como $d(5) > d(6) + m_{65}$, atualizamos os valores de $d(5)$ e $a(5)$.

Figura 24 – Algoritmo de Dijkstra após quarta iteração.



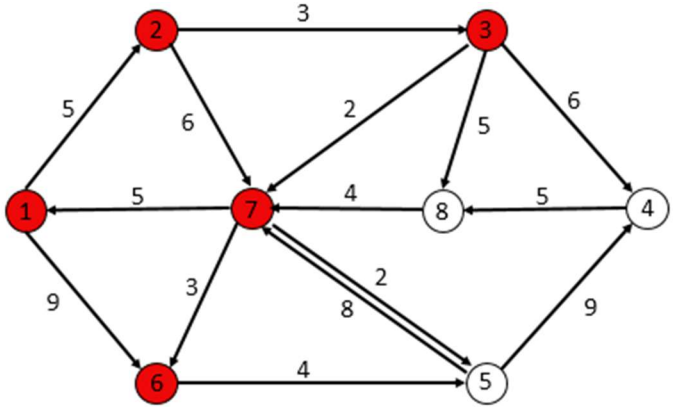
j	1	2	3	4	5	6	7	8
$d(j)$	0	5	8	14	13	9	11	13
$a(j)$	-	1	2	3	6	1	3	3

$F = \{1,2,3,6\}$; $A = \{4,5,7,8\}$

Fonte: O autor, 2018

De acordo com a variável $d(j)$, o próximo vértice a ser fechado é 7. O único vértice adjacente aberto que ele possui é 5. Como $d(5) > d(7) + m_{75}$, melhoramos a estimativa e fazemos as devidas alterações em $d(5)$ e $a(5)$.

Figura 25 – Algoritmo de Dijkstra após quarta iteração.



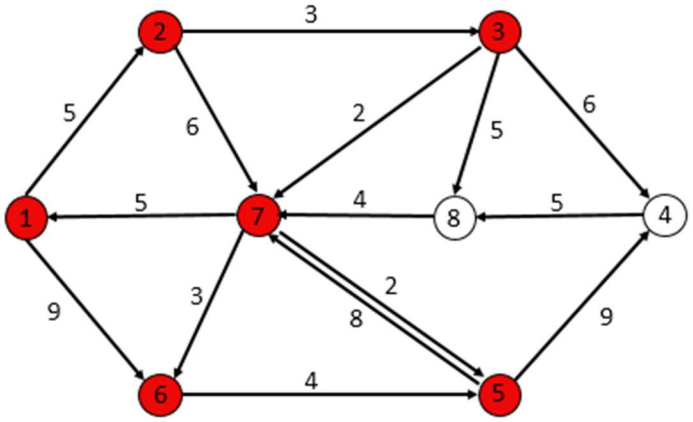
j	1	2	3	4	5	6	7	8
$d(j)$	0	5	8	14	12	9	11	13
$a(j)$	-	1	2	3	7	1	3	3

$F = \{1,2,3,6,7\}$; $A = \{4,5,8\}$

Fonte: O autor, 2018

Finalmente, fechamos o vértice 5. Como $a(5) \neq \{\}$, existe uma sequência de vértices adjacentes que liga 1 a 5 de custo total $d(5)$. Além disso essa sequência possui o menor custo possível.

Figura 26 – Algoritmo de Dijkstra finalizado.



j	1	2	3	4	5	6	7	8
$d(j)$	0	5	8	14	12	9	11	13
$a(j)$	-	1	2	3	7	1	3	3

$F = \{1,2,3,5,6,7\}$; $A = \{4,8\}$

Fonte: O autor, 2018

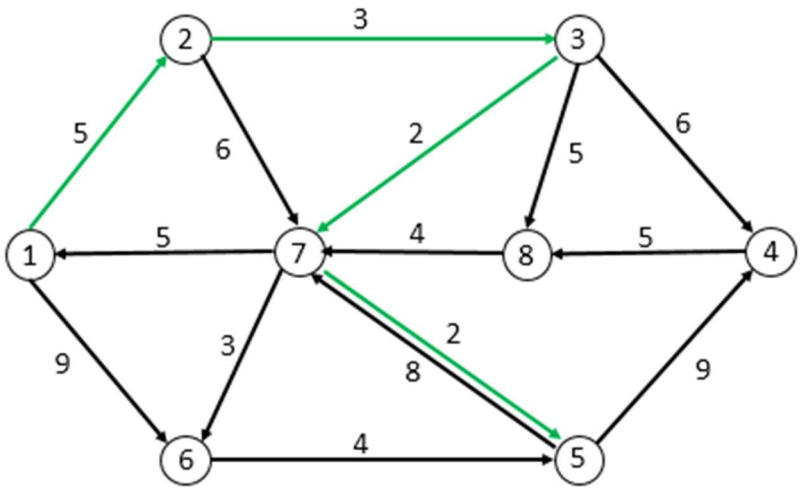
Nesse ponto o algoritmo varrerá os valores de $a(j)$ a partir de $a(5)$ e construirá o vetor C que contém os endereços dos vértices que compõem esse caminho:

$$a(5) = 7; \quad a(7) = 3; \quad a(3) = 2; \quad a(2) = 1.$$

$$C = (1,2,3,7,5)$$

Portanto, o menor caminho que une o vértice 1 ao 5 mede 12 unidades e ele segue a ordem $1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 5$.

Figura 27 – Caminho de menor distância do vértice 1 até o vértice 5.



Fonte: O autor, 2018

3.2 O Algoritmo de Floyd-Warshall (ou Roy-Warshall)[5]

Ao contrário do algoritmo de Dijkstra, o algoritmo de Floyd-Warshall, ou ainda Roy - Warshall, foi desenvolvido independentemente por três pessoas. O primeiro a se deparar com ele foi o professor francês Bernard Roy (1934-2017), em 1959 quando era então consultor da *Société de Mathématiques Appliquées (SMA)*. No campo de Grafos, sua principal contribuição se deu teoria de fluxos em redes e agendamento de projetos, com a invenção do método de *Atividade no Vértice*. Uma outra grande contribuição foi a invenção da família de métodos ELECTRE (*ELimination Et Choix Traduisant la Realité*) utilizados amplamente no campo de análise decisões envolvendo múltiplos critérios.

A segunda pessoa a elaborar o algoritmo foi Robert W. Floyd (1936-2001), no ano de 1962. Ele foi um cientista computacional americano e vencedor do prêmio Turing, maior honraria em ciência da computação, em 1978. Floyd foi uma criança prodígio tendo terminado sua

[5] As informações desta seção foram compiladas de sites de universidades, artigos online, biografias não oficiais e obituários.

primeira graduação, em artes liberais, aos dezessete anos. Thomas Haigh descreve como foi sua transição para a ciência da computação:

> Como toda criança em programas avançados ele logo perdeu o gosto pela escola. Enquanto isso, conseguiu um emprego na *Armour Research Foundation of Illinois Institute of Technology*, primeiro como um operador de computador autodidata e depois como um programador sênior e analista. Ele recebeu seu bacharelado em física da *University of Chicago* em 1958 e publicou um artigo sobre interferência de rádio no mesmo ano. Não demorou muito para que seu amor pela computação ficasse claro. De fato, ele começou a publicar artigos para revistas de computação em 1959. (HAIGH, 2004, p. 75, tradução nossa)

Suas maiores contribuições aconteceram no campo de análise sintática (*theory of parsing*), semântica de linguagens de programação, verificação automática de programas, síntese de programa automática e análise de algoritmos.

No mesmo ano de 1962, Stephen Warshall (1935-2006) publicou um artigo descrevendo o algoritmo enquanto trabalhava na *Massachusetts Computer Associates*. Ele recebeu seu bacharelado em matemática pela *Harvard University* em 1956, aos 21 anos. Ele não fez questão de continuar os estudos para obter o grau de mestre pois os cursos oferecidos pela universidade não o interessavam.

Figura 24 – Bernard Roy, Robert Floyd e Stephen Warshall

Fonte: <http://www.lamsade.dauphine.fr/~roy/>; <https://cs.stanford.edu/memoriam/professor-robert-w-floyd>; <https://en.wikipedia.org/wiki/Stephen_Warshall>

3.2.1 Funcionamento do Algoritmo de Floyd-Warshal

O algoritmo de Floyd-Warshal é um algoritmo que determina o caminho de menor custo entre todos os vértices de um grafo ponderado com arestas de peso positivo ou negativo. O algoritmo original não determina quais vértices devem ser percorridos. Contudo é possível

contornar esse problema fazendo algumas alterações simples no código. Nesse trabalho utilizaremos a versão modificada do algoritmo. Vale ainda ressaltar que estamos supondo um grafo simples sem ciclos negativos. Adiante explicaremos o porquê dessa escolha.

Considere um grafo $G = (V, E)$, com n vértices rotulados utilizando-se os números naturais $1,2, \dots, n$. Com base na matriz de manejo $M^0 = [m_{ij}^0]$, o algoritmo cria uma matriz $R^0 = [r_{ij}^0]$ tal que

$$r_{ij}^0 = \begin{cases} j, \text{se } m_{ij}^0 \neq \infty \\ 0, \text{se } m_{ij}^0 = \infty. \end{cases}$$

Os índices superiores das matrizes, bem como de seus elementos, se referem a iteração na qual a matriz foi obtida. Inicialmente esses índices são zero, significando valores iniciais. Conforme as iterações do algoritmo acontecem, esses índices são atualizados.

A matriz R recebe o nome de **matriz de roteamento** do grafo e ela servirá para construir o vetor C que contém em suas entradas os rótulos dos vetores que comprão o menor caminho. Para cada par de vértices (i, j) ela nos informa qual é o vértice seguinte r_{ij} na sequência de vértices que irá compor o menor caminho possível de i até j. Se $m_{ij}^0 = \infty$, ou seja, não existe arco (i, j), assinalamos $r_{ij}^0 = 0$, o que nos diz que não existe um vértice seguinte a i na composição do caminho. Se $m_{ij}^0 \neq \infty$ então há duas possibilidades:

- $m_{ij}^0 \neq 0$. Dessa forma existe o arco (i, j).
- $m_{ij}^0 = 0$. Então, não existe o arco (i, j), ou seja, m_{ij}^0 é um elemento da diagonal principal.

Em ambos os casos, o vértice seguinte na construção do caminho é o próprio vértice de destino j.

Por exemplo, seja G um grafo de 5 vértices. Se no início da execução do algoritmo temos $m_{25}^0 = 18$, significa que existe a o arco $(2,5)$ e ele possui peso 18. Além disso o próximo vértice no menor caminho $2 \rightarrow 5$ (por enquanto) é o próprio 5. Assim, $r_{25}^0 = 5$. Suponha ainda, nesse mesmo grafo, que $m_{34}^0 = \infty$. Então, não existe o arco $(3,4)$. Logo, devemos ter $r_{34}^0 = 0$.

A cada iteração, o algoritmo fixa o vértice de menor rótulo k dessa lista, chamado **vértice intermediário**. Então ele cria as matrizes M^k e R^k que serão atualizadas durante a execução. Cada elemento m_{ij}^k de M^k e r_{ij}^k de R^k são tais que

$$m_{ij}^k = m_{ij}^{k-1} \text{ e } r_{ij}^k = r_{ij}^{k-1}.$$

Em seguida, para cada par (i, j), com i e j distintos de k, o algoritmo verifica se a distância entre esses vértices, passando pelo intermediário k, é menor que distância sem passar por ele. Em outras palavras, o algoritmo verifica se

$$m_{ik}^{k-1} + m_{kj}^{k-1} < m_{ij}^{k-1}.$$

Em caso afirmativo, m_{ij}^{k} é substituída pelo valor da soma.

Por último, a matriz de roteamento é atualizada. Sempre que m_{ij}^{k} for substituída, r_{ij}^{k} é substituída por r_{ik}^{k-1}. Essa escolha está de acordo com a definição da matriz R, pois se ao tomarmos k como intermediário, a distância $i \to j$ é melhorada, então r_{ij}^{k} deve ser atualizada com o vértice seguinte a i, no caminho de i até k, ou seja r_{ik}^{k-1}.

Retomando o exemplo acima, considere que o algoritmo esteja na segunda iteração e que

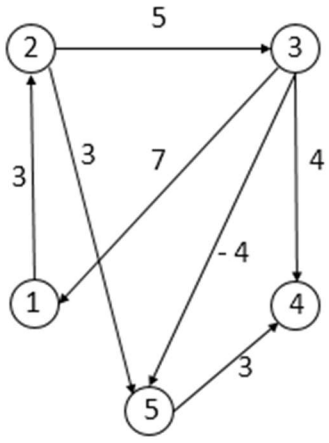
$$m_{32}^{1} + m_{25}^{1} < m_{35}^{1}.$$

Então, m_{35}^{2} é substituída por $m_{32}^{1} + m_{25}^{1}$ na matriz M^2 e r_{35}^{2} é substituída por r_{32}^{1}, uma vez que o caminho é mais curto tomando o vértice 2 como intermediário no caminho de 3 até 5.

O processo para quando k varrer todos os vértices do grafo e o algoritmo tem como saída as matrizes M^n e R^n. É possível ainda criar uma rotina que obtém o caminho entre dois vértices após a execução do algoritmo. O fluxograma e pseudocódigo do algoritmo se encontram nos apêndices C e D respectivamente.

Utilizaremos o grafo G da Figura 25 para um exemplo de execução do algoritmo. As matrizes M^0 e R^0 são mostradas ao lado do grafo.

Figura 25 – Grafo G e suas matrizes de Pesos e de Roteamento



M^0	1	2	3	4	5
1	0	3	∞	∞	∞
2	∞	0	5	∞	3
3	7	∞	0	4	−4
4	∞	∞	∞	0	∞
5	∞	∞	∞	3	0

R^0	1	2	3	4	5
1	0	2	0	0	0
2	0	0	3	0	5
3	1	0	0	4	5
4	0	0	0	0	0
5	0	0	0	4	0

Fonte: O autor, 2018

Para esse grafo o algoritmo fará cinco iterações tomando o vértice 1 como primeiro intermediário.

1ª Iteração ($k = 1$): Para cada par de vértices (i, j), com $i, j \neq 1$, o algoritmo verificará se a distância entre esses vértices, tendo 1 como intermediário, é menor que a distância entre eles, ou seja, se $m_{i1}^{0} + m_{1j}^{0} < m_{ij}^{0}$. Em caso afirmativo, $m_{ij}^{1} \leftarrow m_{i1}^{0} + m_{1j}^{0}$ e $r_{ij}^{1} \leftarrow r_{i1}^{0}$. Sempre que $m_{i1}^{0} = \infty$ ou $m_{1j}^{0} = \infty$ a estimativa do caminho não será melhorada pois a soma $m_{i1}^{0} + m_{1j}^{0}$ nunca poderá ser inferior a ∞. Assim temos:

$i = 2\ e\ j = 2,3,4,5$: Não há alterações na segunda linha pois $m_{21}^{0} = \infty$;

$i = 3\ e\ j = 2$: $m_{31}^{0} + m_{12}^{0} = 7 + 3 < \infty = m_{32}^{0}$, então $m_{32}^{1} \leftarrow 10$ e $r_{32}^{1} \leftarrow 1$;

$i = 3\ e\ j = 3,4,5$: Não há alterações na terceira linha pois $m_{13}^{0} = m_{14}^{0} = m_{15}^{0} = \infty$;

$i = 4\ e\ j = 2,3,4,5$: Não há alterações na quarta linha pois $m_{41}^{0} = \infty$;

$i = 5\ e\ j = 2,3,4,5$: Não há alterações na quinta linha pois $m_{51}^{0} = \infty$;

Após a 1ª iteração temos as matrizes de manejo e de roteamento mostradas na Figura 26.

Figura 26 – Matrizes de Pesos e de Roteamento após a 1ª iteração

M^1	1	2	3	4	5
1	0	3	∞	∞	∞
2	∞	0	5	∞	3
3	7	**10**	0	4	−4
4	∞	∞	∞	0	∞
5	∞	∞	∞	3	0

R^1	1	2	3	4	5
1	0	2	0	0	0
2	0	0	3	0	5
3	1	**1**	0	4	5
4	0	0	0	0	0
5	0	0	0	4	0

Fonte: O autor, 2018

2ª Iteração ($k = 2$):

$i = 1\ e\ j = 1$: Inicialmente supomos que trabalharíamos em grafos simples sem ciclos negativos. Fizemos essa suposição pois o algoritmo não consegue lidar com tais ciclos, pois de outra forma ele sempre conseguiria encontrar um caminho menor de um vértice a ele mesmo e entraria em loop. Portanto, m_{ii}^{k} nunca sofrerá ajuste. Assim, m_{11}^{2} não será atualizado. De agora em diante não analisaremos mais o caso m_{ii}^{k}.

$i = 1\ e\ j = 3$: $m_{12}^{1} + m_{23}^{1} = 3 + 5 < \infty = m_{13}^{1}$, então $m_{13}^{2} \leftarrow 8$ e $r_{13}^{2} \leftarrow 2$;

$i = 1\ e\ j = 4$: Não há alterações pois $m_{24}^{1} = \infty$;

$i = 1\ e\ j = 5$: $m_{12}^{1} + m_{25}^{1} = 3 + 3 < \infty = m_{15}^{1}$, então $m_{15}^{2} \leftarrow 6$ e $r_{13}^{2} \leftarrow 2$;

$i = 3\ e\ j = 1,4$: Não há alterações pois $m_{21}^{1} = m_{24}^{1} = \infty$;

$i = 3\ e\ j = 5$: $m_{32}^{1} + m_{25}^{1} = 10 + 3 > -4 = m_{35}^{1}$, portanto não há alteração na distância (3,5);

$i = 4\ e\ j = 1,3,5$: Não há alterações na quarta linha pois $m_{42}^{1} = \infty$;

$i = 5\ e\ j = 1,3,4$: Não há alterações na quinta linha pois $m_{52}^{1} = \infty$.

Figura 27 – Matrizes de Pesos e de Roteamento após a 2ª iteração

M^2	1	2	3	4	5
1	0	3	8	∞	6
2	∞	0	5	∞	3
3	7	10	0	4	−4
4	∞	∞	∞	0	∞
5	∞	∞	∞	3	0

R^2	1	2	3	4	5
1	0	2	2	0	2
2	0	0	3	0	5
3	1	1	0	4	5
4	0	0	0	0	0
5	0	0	0	4	0

Fonte: O autor, 2018

3ª Iteração ($k = 3$):

$i = 1$ *e* $j = 2$: $m_{13}^2 + m_{32}^2 > m_{13}^2$, sem alteração em m_{13}^3.

$i = 1$ *e* $j = 4$: $m_{13}^2 + m_{34}^2 = 8 + 4 < \infty = m_{14}^2$, então $m_{14}^3 \leftarrow 12$ e $r_{14}^3 \leftarrow r_{13}^2 = 2$;

$i = 1$ *e* $j = 5$: $m_{13}^2 + m_{35}^2 = 8 + (-4) < 6 = m_{15}^2$, então $m_{15}^3 \leftarrow 4$ e $r_{15}^3 \leftarrow r_{13}^2 = 2$;

$i = 2$ *e* $j = 1$: $m_{23}^2 + m_{31}^2 = 5 + 7 < \infty = m_{21}^2$, então $m_{21}^3 \leftarrow 12$ e $r_{21}^3 \leftarrow r_{23}^2 = 3$;

$i = 2$ *e* $j = 4$: $m_{23}^2 + m_{34}^2 = 5 + 4 < \infty = m_{24}^2$, então $m_{24}^3 \leftarrow 9$ e $r_{24}^3 \leftarrow r_{23}^2 = 3$;

$i = 2$ *e* $j = 5$: $m_{23}^2 + m_{35}^2 = 5 + (-4) < 3 = m_{25}^2$, então $m_{25}^3 \leftarrow 1$ e $r_{25}^3 \leftarrow r_{23}^2 = 3$;

$i = 4$ *e* $j = 1,2,5$: Não há alterações na quarta linha pois $m_{43}^2 = \infty$;

$i = 5$ *e* $j = 1,2,4$: Não há alterações na quarta linha pois $m_{53}^2 = \infty$;

Após a 3ª Iteração temos:

Figura 28 – Matrizes de Pesos e de Roteamento após a 3ª iteração

M^3	1	2	3	4	5
1	0	3	8	12	4
2	12	0	5	9	1
3	7	10	0	4	−4
4	∞	∞	∞	0	∞
5	∞	∞	∞	3	0

R^3	1	2	3	4	5
1	0	2	2	2	2
2	3	0	3	3	3
3	1	1	0	4	5
4	0	0	0	0	0
5	0	0	0	4	0

Fonte: O autor, 2018

4ª Iteração ($k = 4$):

Como $m_{4j}^3 = \infty$ para todo $j \neq 4$, não haverá melhora nas estimativas tomando o vértice 4 como intermediário. Isso fica evidente ao observarmos na representação gráfica que o vértice 4 não é adjacente a nenhum outro. Assim,

Figura 29 – Matrizes de Pesos e de Roteamento após a 4ª iteração

M^4	1	2	3	4	5
1	0	3	8	12	4
2	12	0	5	9	1
3	7	10	0	4	−4
4	∞	∞	∞	0	∞
5	∞	∞	∞	3	0

R^4	1	2	3	4	5
1	0	2	2	2	2
2	3	0	3	3	3
3	1	1	0	4	5
4	0	0	0	0	0
5	0	0	0	4	0

Fonte: O autor, 2018

5ª Iteração ($k = 5$):

$i = 1$ *e* $j = 1,2,3$: Não há alterações pois $m_{51}^4 = m_{52}^4 = m_{53}^4 = \infty$;

$i = 1$ *e* $j = 4$: $m_{15}^4 + m_{54}^4 = 4 + 3 < 12 = m_{14}^4$, então $m_{14}^5 \leftarrow 7$ e $r_{14}^5 \leftarrow r_{15}^4 = 2$;

$i = 2$ *e* $j = 1,2,3$: Não há alterações pois $m_{51}^4 = m_{52}^4 = m_{53}^4 = \infty$;

$i = 2$ *e* $j = 4$: $m_{25}^4 + m_{54}^4 = 1 + 3 < 9 = m_{24}^4$, então $m_{24}^5 \leftarrow 4$ e $r_{24}^5 \leftarrow r_{25}^4 = 3$;

$i = 3$ *e* $j = 1,2,3$: Não há alterações pois $m_{51}^4 = m_{52}^4 = m_{53}^4 = \infty$;

$i = 3$ *e* $j = 4$: $m_{35}^4 + m_{54}^4 = (-4) + 3 < 4 = m_{34}^4$, então $m_{34}^5 \leftarrow -1$ e $r_{34}^5 \leftarrow r_{35}^4 = 5$;

$i = 4$ *e* $j = 1,2,3$: Não há alterações pois $m_{51}^4 = m_{52}^4 = m_{53}^4 = \infty$;

Finalmente, após as cinco iterações temos as matrizes:

Figura 30 – Matrizes de Pesos e de Roteamento após a 5ª iteração

M^5	1	2	3	4	5
1	0	3	8	7	4
2	12	0	5	4	1
3	7	10	0	−1	−4
4	∞	∞	∞	0	∞
5	∞	∞	∞	3	0

R^5	1	2	3	4	5
1	0	2	2	2	2
2	3	0	3	3	3
3	1	1	0	5	5
4	0	0	0	0	0
5	0	0	0	4	0

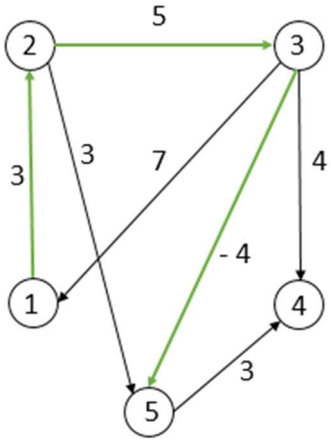
Fonte: O autor, 2018

Na matriz M^5 temos o valor do custo mínimo para ir do vértice i ao vértice j. Se algum $m_{ij}^5 = \infty$ então o vértice j não é atingível a partir de i. A matriz de roteamento fornecerá os vértices que irão compor o caminho $i \rightarrow j$.

Vamos determinar a sequência de vértices que irão compor o caminho $1 \rightarrow 5$. Observando a matriz de manejo temos $m_{15}^5 = 4$, então existe o trajeto $1 \rightarrow 5$ e seu custo mínimo é de 4 unidades. Consultando agora a matriz R^5 temos $r_{15}^5 = 2$. Isso quer dizer que para chegar a 5, primeiro devemos passar por 2. Buscamos agora qual é o vértice seguinte a 2 no caminho $2 \rightarrow 5$ 5: $r_{25}^5 = 3$. Ou seja, para atingir 5 a partir de 2, devemos passar por 3. Devemos agora buscar

qual é o próximo vértice no caminho $3 \rightarrow 5$. Consultando novamente a matriz encontramos $r_{35}^{5} = 5$, que é o nosso vértice final. Então a sequência de vértices procurada é $1 \rightarrow 2 \rightarrow 3 \rightarrow 5$.

Figura 31 – Menor caminho que une o vértice 1 ao vértice 5.



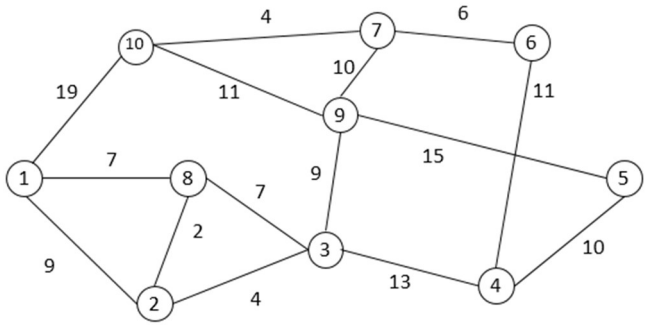
Fonte: O autor, 2018

3.2.2 Aplicações do Algoritmo de Floyd-Warshal

Os algoritmos de caminho mínimo podem nos ajudar na tomada de decisão em diversas situações. Considere a seguinte:

O dono de uma marca de chocolates que deseja expandir seus negócios comprou dez imóveis numa dada região. Em um desses imóveis ele estabelecerá a fábrica que irá produzir e distribuir os chocolates. Nos demais imóveis serão construídas as lojas que efetuarão as vendas para o público. No grafo abaixo os vértices representam os terrenos e os pesos das arestas a distância entre eles em quilômetros.

Figura 32 – Grafo que representa os imóveis comprados.



Fonte: O autor, 2018

Em qual dos imóveis a fábrica deverá ser instalada? Por que?

Para responder à pergunta acima devemos estabelecer algum tipo critério. A escolha do imóvel deve se dar de maneira que o custo de distribuição seja mínimo, dessa forma o lucro será

máximo. Esse é um dos critérios mais recorrentes e é comumente chamado de *critério de lucro* (BOAVENTURA; JURKIEWICZ, 2009). Em geral, os custos com distribuição são menores quando a distância entre os pontos de entrega também o são. Devemos então determinar o vértice cuja soma das menores distâncias até os outros vértices é mínima. Para isso, basta aplicar o algoritmo de Floyd-Warshall e calcular a soma dos elementos de cada linha da matriz de saída[6].

Figura 33 – Matrizes das menores distâncias e da distância acumulada.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	D. A.
1	0	9	13	26	36	29	23	7	22	19	184
2	9	0	4	17	27	28	23	2	13	24	147
3	13	4	0	13	23	24	19	6	9	20	**131**
4	26	17	13	0	10	11	17	19	22	21	156
5	36	27	23	10	0	21	25	29	15	26	212
6	29	28	24	11	21	0	6	30	16	10	175
7	23	23	19	17	25	6	0	25	10	4	152
8	7	2	6	19	29	30	25	0	15	26	159
9	22	13	9	22	15	16	10	15	0	11	133
10	19	24	20	21	26	10	4	26	11	0	161

Fonte: O autor, 2018

Ao lado da matriz de distâncias encontra-se a coluna distância acumulada (D.A.). Podemos observar que o vértice 3 possui a menor distância acumulada dos outros vértices. Isso quer dizer que um veículo que faça o transporte da fábrica para as lojas percorrerá uma distância menor, gerando um custo de transporte menor e, por consequência um maior lucro para a empresa. Em teoria dos grafos o vértice que tem essa propriedade é chamado de *mediana* do grafo. Esse tipo de problema é conhecido como problema de mediana ou *minissoma*.

Um outro critério bastante empregado é o *critério de emergência*. Ele é aplicável em situações onde o vértice onde se deseja alocar a instalação ficar o mais próximo possível dos outros vértices, como por exemplo a construção de um hospital, posto policial, quartel do corpo de bombeiros, etc. A ideia por trás dessa escolha é que o usuário (vértice) mais distante não tenha que esperar muito para ser atendido. Vamos apresentar um exemplo.

Suponha que o grafo da Figura 31 representa a distribuição de casas numa região rural fictícia e deseja-se instalar um posto policial para atender a essas famílias. O local escolhido deve ficar o mais próximo possível de todas as casas. Para que isso aconteça, a maior distância do posto policial a uma dessas casas deve ser a menor possível, ou seja, estamos procurando o vértice

[6] Para obtenção dos valores apresentados na Figura 32 utilizamos uma versão do algoritmo implementada numa planilha *excel*. Disponível em: <https://www.excelforum.com/excel-programming-vba-macros/>. Acesso em: 12 Jan. 2018.

cuja maior distância em relação aos outros é a menor possível. Consultando novamente a matriz da Figura 33 podemos construir a tabela a seguir.

Figura 34 – Maior distância do posto policial.

Vértice	Afastamento
1	36
2	28
3	24
4	26
5	36
6	29
7	25
8	29
9	**22**
10	26

Fonte: O autor, 2018

Nessa matriz a coluna Afastamento representa a maior das distâncias de um vértice aos outros. Podemos observar que o vértice 9 apresenta a menor dessas distâncias. Portanto ele é o melhor candidato para a instalação do nosso posto policial. Um vértice com tal propriedade é chamado de *centro* do grafo e seu afastamento é o *raio* do grafo.

4 UMA PROPOSTA PARA ALÉM DA SALA DE AULA

Este capítulo tem por objetivo apresentar uma proposta de oficina para alunos do ensino médio onde serão introduzidos alguns conceitos básicos da teoria de Grafos e uma aplicação dos algoritmos de Dijkstra e Floyd-Warshall. A partir de um problema contextualizado de caminhos mínimos o aluno será encorajado a encontrar soluções utilizando tais algoritmos.

Para isso, sugerimos uma sequência de tópicos que podem ser trabalhados com alunos do 2º ou 3º ano do ensino médio. É desejável, mas não necessariamente obrigatório, que esses alunos tenham conhecimento básico do conceito de Matrizes. A estrutura de desenvolvimento desses tópicos se relaciona com os conteúdos dos capítulos 2 e 3 desse trabalho pois acreditamos que os modelos apresentados possam ser construídos por esses alunos, seja para demonstrar aplicação, verificar resultados ou mesmo despertar interesse pelo tema.

4.1 Tópico 1: O Problema Motivador

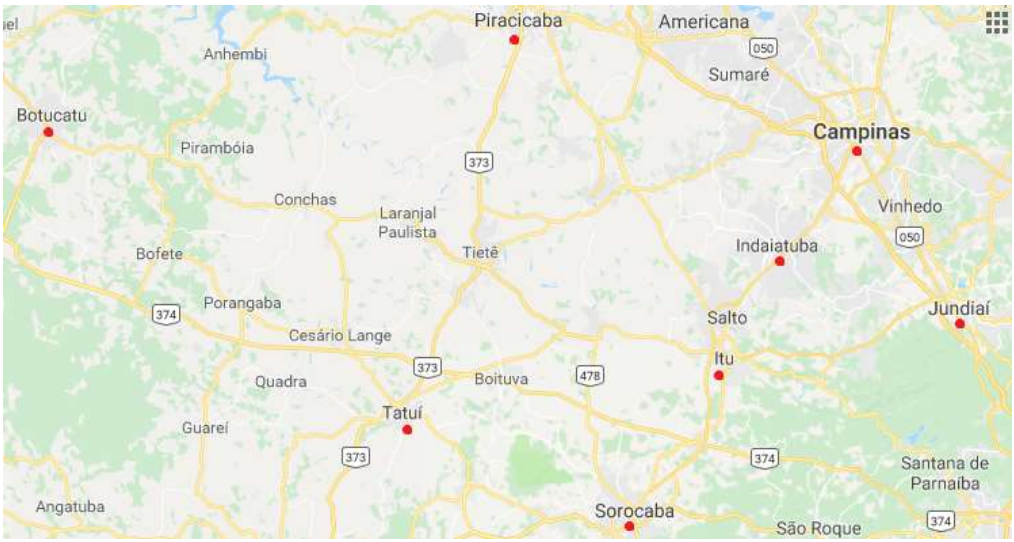
Objetivo: Através de um problema motivador apresentar alguns conceitos básicos da teoria de Grafos;

Desenvolvimento:

- Apresentação do Problema Motivador:

"O dono de uma loja de departamentos deseja expandir seus negócios numa região do interior de São Paulo. Ele pretende abrir sete lojas nas cidades de Piracicaba, Botucatu, Tatuí, Sorocaba, Itu, Jundiaí, Indaiatuba e um centro de distribuição em Campinas. Sempre que se deseja fazer uma entrega do Centro de distribuição para uma das lojas aproveita-se a viagem para recolher notas fiscais nas demais lojas situadas ao longo do caminho por onde o entregador passará. A escolha do caminho utilizado pelo entregador deve ser feita de modo que a distância percorrida por ele seja a menor possível. A Figura 35 apresenta o mapa da região onde serão instaladas as lojas.

Figura 35 – Mapa com as cidades onde as lojas serão abertas.



Fonte: <https://www.google.com.br/maps>

De posse dessas informações e das distâncias apresentadas na Figura 36 responda as perguntas abaixo.

a) Quantos quilômetros serão percorridos para fazer uma entrega para a loja situada em Tatuí? Por quais cidades esse entregador passará?

b) Suponha agora que a central de distribuição possa ser instalada em qualquer uma das cidades. Em qual delas você a instalaria? Por que?"

Figura 36 – Tabela com distâncias conhecidas.

Cidades	Distância (Km)
$Piracicaba \leftrightarrow Botucatu$	115
$Botucatu \rightarrow Itu$	148
$Itu \rightarrow Botucatu$	164
$Botucatu \leftrightarrow Tatuí$	107
$Tatuí \rightarrow Sorocaba$	61
$Sorocaba \rightarrow Tatuí$	52
$Itu \leftrightarrow Sorocaba$	38
$Itu \leftrightarrow Jundiaí$	48
$Itu \leftrightarrow Indaiatuba$	27
$Jundiaia \leftrightarrow Indaiatuba$	47
$Indaiatuba \leftrightarrow Campinas$	28
$Campinas \leftrightarrow Jundiaí$	39
$Campinas \leftrightarrow Piracicaba$	71

Fonte: O autor, 2018

- Introdução aos conceitos básicos de Teoria de Grafos

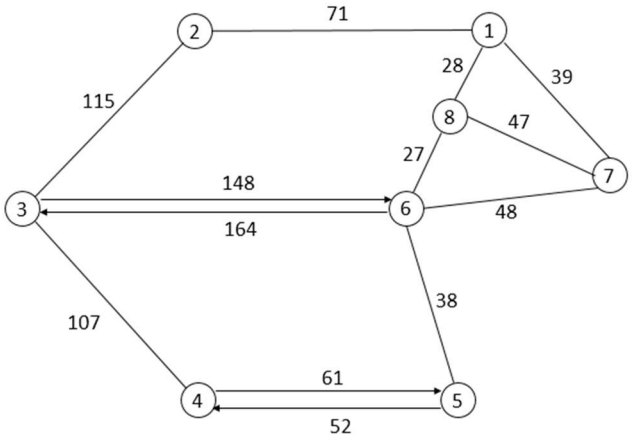
Com o objetivo de modelar o problema motivador, serão apresentados os conceitos fundamentais acerca da teoria de grafos abordados no Capítulo II. Espera-se que a introdução desse novo objeto de estudo, associado a um problema de ordem prática, propicie uma aprendizagem significativa para alunos. Além disso, tais conceitos serão importantes tanto para

a compreensão da modelagem como para o entendimento dos algoritmos que serão tratados nos tópicos posteriores.

- Modelagem do Problema

A partir do que foi visto anteriormente, o problema será modelado por um digrafo ponderado como o da figura a seguir.

Figura 37 – Situação problema modelada por um grafo.



Legenda	
Cidade	Rótulo
Campinas	1
Piracicaba	2
Botucatu	3
Tatuí	4
Sorocaba	5
Itu	6
Jundiaí	7
Indaiatuba	8

Fonte: O autor, 2018

- Relacionar o Problema Motivador com a determinação de um caminho mínimo entre dois vértices do grafo.

4.2 Tópico 2: Algoritmos

Objetivo: Conceituar o que é um algoritmo e descrever seus passos.

Desenvolvimento:

- Sugerir que os alunos proponham soluções para algumas situações tais como atravessar uma rua, fazer uma pipoca, calcular uma média final escolar, procurar palavras no dicionário, etc, descrevendo de maneira eficiente um passo a passo de suas ações encadeadas logicamente. Duas sugestões de situações problema e seus respectivos algoritmos podem ser encontradas no Apêndices F.
- Conceituar algoritmo segundo BACKES (2011): “Um *algoritmo* pode ser definido como uma sequência finita e ordenada de instruções para solucionar um determinado problema.”
- Fazer um apanhado histórico da origem dos algoritmos. Uma sugestão de material para consulta encontra-se no apêndice G.

4.3 Tópico 3: O Algoritmo de Dijkstra

Objetivo: Apresentar o passo a passo do Algoritmo de Dijkstra.

Desenvolvimento:

- Descrever a finalidade do algoritmo;
- Apresentar o passo a passo do Algoritmo de Dijkstra, fazendo uma adequação de linguagem do conteúdo apresentado no capítulo 3 de maneira a torna-lo mais acessível para a compreensão dos alunos, o que será feito a seguir.

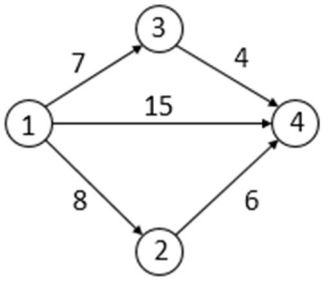
Sejam i e f dois vértices em um grafo $G = (V, E)$. Deseja-se determinar o custo do menor caminho que tem origem em i e término em f. Note que o caminho possui orientação, uma vez que em um grafo orientado caminho $i \to f$ é distinto do caminho $f \to i$.

Para o funcionamento do algoritmo, precisamos definir dois parâmetros relacionados com os vértices do grafo. Seja $j \in V$:

- $d(j)$: é o comprimento do menor caminho que tem origem em i e término em j;
- $a(j)$: indica qual é o vértice que antecede v na sequência de vértices que vão compor o caminho desejado.

Por exemplo, considere o grafo da Figura 38.

Figura 38 – Grafo.



Fonte: O autor, 2018

Suponha que desejamos buscar o menor caminho do vértice 1 até o vértice 4. Por inspeção verificamos que o caminho procurado é $(1,3,4)$ e ele mede 11 unidades de comprimento. Assim $d(4) = 11$, $a(4) = 3$, $a(3) = 1$ e $a(1) = \nexists$.

Durante a execução do algoritmo obteremos sucessivamente o menor caminho do vértice de origem a cada vértice do grafo, até que o vértice de nosso interesse seja atingido. Dessa forma, os vértices podem ser classificados de duas formas, para as quais definiremos os conjuntos abaixo:

- Conjunto F dos vértices fechados: formados pelos vértices j para os quais já se obteve o menor caminho de i a j.

- Conjunto A dos vértices abertos: é o complementar de F em relação a V, ou seja, $A = V - F$

  O passo a passo do algoritmo será apresentado a seguir.

Algoritmo de Dijkstra

Seja $M = [m_{ij}]$ a matriz de manejo associada ao grafo, i o vértice de origem, f o vértice de destino e j um vértice qualquer.

1º Passo: Fazemos $d(i) = 0$, e $a(i) = \nexists$

2º Passo: Fazemos $d(j) = m_{ij}$ se existe a aresta (i, j). Caso contrário e $d(j) = \infty$.

3º Passo: Fazemos $a(j) = \nexists$ se $d(j) = \infty$. Caso contrário $a(j) = i$;

4º Passo: Fazemos $F = \{i\}$ e $A = V - F$;

5º Passo: Seja $r \in A$ tal que $d(r)$ seja mínimo;

6º Passo: Fechamos o vértice r. O retiramos de A e o incluímos em F;

7º Passo: Se $r = f$ o algoritmo cessa. Caso contrário faça:

(1): Para todo vértice $s \in A$ tal que s sucessor de r calcule a soma $d(r) + m_{rs}$;

(2): Caso alguma soma da etapa (1) seja inferior a $d(s)$, substitua $d(s)$ por essa soma e faça $a(s) = r$.

(3): Retorne para o 5º passo.

4.4 Tópico 4: Aplicação do Algoritmo de Dijkstra

Objetivo: Responder ao item a) do problema motivador.

Desenvolvimento:

- Retomar o item a) do problema motivador: "Quantos quilômetros serão percorridos para fazer uma entrega para a loja situada em Tatuí? Por quais cidades esse entregador passará?"
- Aplicar o passo a passo do algoritmo

Podemos responder à primeira pergunta aplicando-se o Algoritmo de Dijkstra. Nesse caso, o custo será a distância entre Campinas e Tatuí e os vértices que constituirão esse menor caminho serão as cidades percorridas pelo entregador. A Matriz de Manejo relacionada ao grafo é

$$M = \begin{bmatrix} 0 & 71 & \infty & \infty & \infty & \infty & 39 & 28 \\ 71 & 0 & 115 & \infty & \infty & \infty & \infty & \infty \\ \infty & 115 & 0 & 107 & \infty & 148 & \infty & \infty \\ \infty & \infty & 107 & 0 & 61 & \infty & \infty & \infty \\ \infty & \infty & \infty & 52 & 0 & 38 & \infty & \infty \\ \infty & \infty & 164 & \infty & 38 & 0 & 48 & 27 \\ 39 & \infty & \infty & \infty & \infty & 48 & 0 & 47 \\ 28 & \infty & \infty & \infty & \infty & 27 & 47 & 0 \end{bmatrix}$$

Seguindo a rotulação proposta na Figura 37 temos $i = 1$ e $f = 4$. Executando os 4 primeiros passos do algoritmo:

1º Passo: $d(1) = 0$ e $a(1) = \nexists$

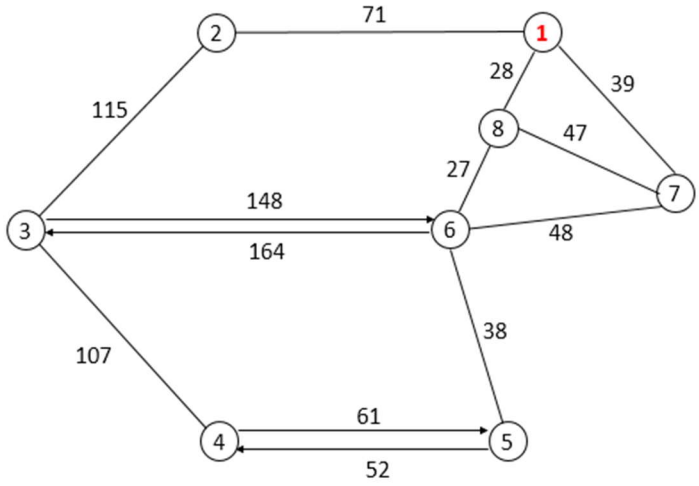
2º Passo: $d(2) = 71; d(7) = 39; d(8) = 28$ *e* $d(3) = d(4) = d(5) = d(6) = \infty$

3º Passo: $a(2) = a(7) = a(8) = 1$ *e* $a(3) = a(4) = a(5) = a(6) = \nexists$

4º Passo: $F = \{1\}$ *e* $A = \{2,3,4,5,6,7,8\}$

A Figura 39 indica o estado atual do algoritmo. Os vértices fechados serão assinalados em vermelho.

Figura 39 – Inicialização do Algoritmo de Dijkstra.



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	∞	∞	∞	∞	39	28
$a(j)$	–	1	–	–	–	–	1	1

$F = \{1\}; A = \{2,3,4,5,6,7,8\}$

Fonte: O autor, 2018

A partir do 5º passo buscaremos sucessivamente o vértice mais próximo da origem. Consultando a tabela da Figura 39 temos que o menor dos $d(8)$ é o mais próximo da origem. Executando o 5º e 6º passo temos:

5º Passo: Fazemos $r = 8$;

6º Passo: $F = \{1,8\}$ *e* $A = \{2,3,4,5,6,7\}$

7º Passo: Neste passo verificamos se o vértice r selecionado é o nosso vértice de destino. Como isso não ocorre o algoritmo segue. As etapas (1) e (2) desse passo verificam se o comprimento do caminho $i \to r \to s$, onde s é um vértice aberto e adjacente a r, é menor que o comprimento $d(s)$ que não possui r como intermediário. Em caso afirmativo atualizamos os valores $d(s)$ e $a(s)$. Nesse caso, avaliaremos os vértices 6 e 7 que são os únicos vértices adjacentes e abertos a 8.

$r = 8 \neq 4$,

(1) $s = 6 \Rightarrow d(8) + m_{86} = 55$

$s = 7 \Rightarrow d(8) + m_{87} = 75$

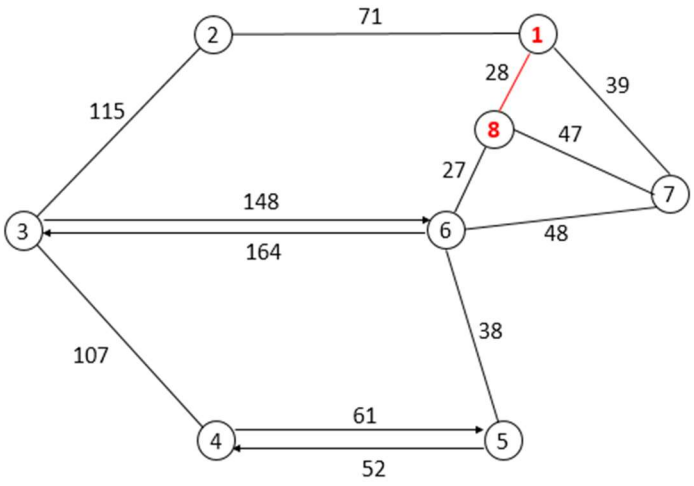
(2) Como $d(6) = \infty > 55$ atualizamos $d(6) = 55$ e $a(6) = 8$

Como $d(7) = 39 < 75$ não alteramos $d(7)$ e $a(7)$.

(3) Retorne para o 5º Passo.

O caminho $1 \to 8 \to 6$ menor comprimento que o caminho $1 \to 8$ que não possui 6 como intermediário. Logo, melhoramos a estimativa da menor distância de 1 até 6 e atualizamos o vértice anterior a 6 para 8, pois é passando por 8 que temos esse caminho. Finalmente o algoritmo retorna para o 5º Passo, onde um novo vértice mais próximo a origem é escolhido. A Figura 40 mostra estado atual do algoritmo. Assinalamos a aresta (1,8) em vermelho para indicar que ela comporá o menor caminho de algum vértice (o vértice 8 até o momento).

Figura 40 – 1ª iteração do Algoritmo de Dijkstra



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	∞	∞	∞	55	39	28
$a(j)$	–	1	–	–	–	8	1	1

$F = \{1,8\}; A = \{2,3,4,5,6,7\}$

Fonte: O autor, 2018

5º Passo: Consultando a tabela da Figura 40 observamos que $d(7)$ é o mínimo do conjunto $\{d(j), j \in A\}$. Portanto $r = 7$.

6º Passo: $F = \{1,7,8\}$ *e* $A = \{2,3,4,5,6\}$

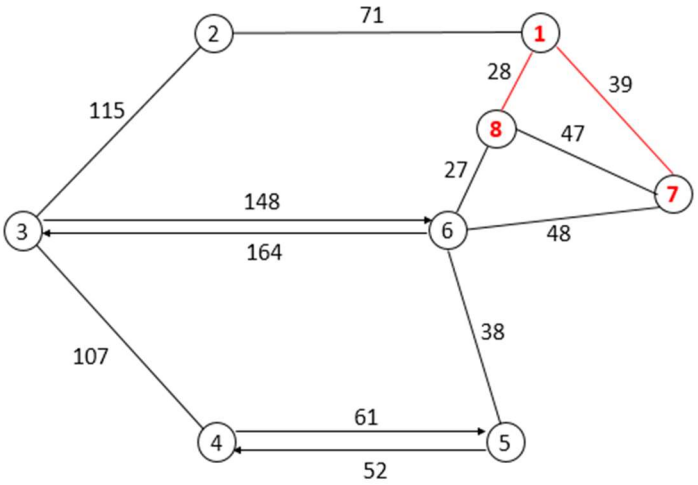
7º Passo: Como $r = 7 \neq 4$ o algoritmo prossegue. O único vértice aberto adjacente a 7 é 6. Então:

(1) $s = 6 \Rightarrow d(7) + m_{76} = 87$

(2) Como $d(6) = 55 < 87$ não alteramos $d(6)$ e $a(6)$.

(3) Retorne para o 5º Passo.

Figura 41 – 2ª iteração do Algoritmo de Dijkstra



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	∞	∞	∞	55	39	28
$a(j)$	–	1	–	–	–	8	1	1

$F = \{1,7,8\}; A = \{2,3,4,5,6\}$

Fonte: O autor, 2018

5º Passo: Consultando a tabela da Figura 41 observamos que $d(6)$ é o mínimo do conjunto $\{d(j), j \in A\}$. Portanto $r = 6$.

6º Passo: $F = \{1,6,7,8\}$ *e* $A = \{2,3,4,5\}$

7º Passo: Como $r = 6 \neq 4$ o algoritmo prossegue. Os vértices abertos adjacentes a 6 são 3 e 5. Então:

(1) $s = 3 \Rightarrow d(6) + m_{63} = 203$

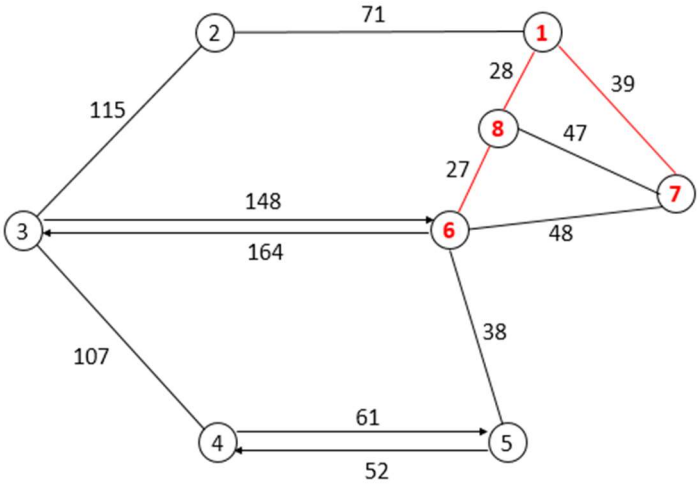
$s = 5 \Rightarrow d(6) + m_{65} = 93$

(2) Como $d(3) = \infty > 203$ atualizamos $d(3) = 203$ e $a(3) = 6$

Como $d(5) = \infty > 93$ atualizamos $d(5) = 93$ e $a(5) = 6$

(3) Retorne para o 5º Passo.

Figura 42 – 3ª iteração do Algoritmo de Dijkstra



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	203	∞	93	55	39	28
$a(j)$	–	1	6	–	6	8	1	1

$F = \{1,6,7,8\}; A = \{2,3,4,5\}$

Fonte: O autor, 2018

5º Passo: Consultando a tabela da Figura 42 observamos que $d(2)$ é o mínimo do conjunto $\{d(j), j \in A\}$. Portanto $r = 2$.

6º Passo: $F = \{1,2,6,7,8\}$ *e* $A = \{3,4,5\}$

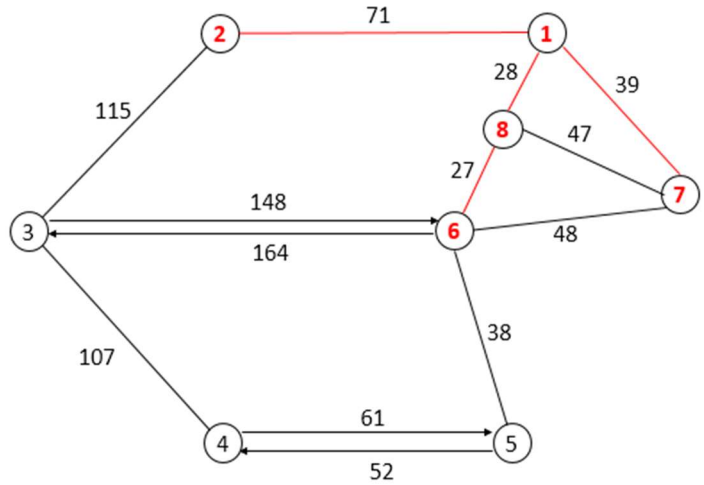
7º Passo: Como $r = 2 \neq 4$ o algoritmo prossegue. O único vértice aberto adjacente a 2 é o vértice 3. Então:

(1) $s = 3 \Rightarrow d(2) + m_{23} = 186$

(2) Como $d(3) = 203 > 203$ atualizamos $d(3) = 186$ e $a(3) = 2$

(3) Retorne para o 5º Passo.

Figura 43 – 4ª iteração do Algoritmo de Dijkstra



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	186	∞	93	55	39	28
$a(j)$	–	1	2	–	6	8	1	1

$F = \{1,2,6,7,8\}; A = \{3,4,5\}$

Fonte: O autor, 2018

5º Passo: Consultando a tabela da Figura 43 observamos que $d(5)$ é o mínimo do conjunto $\{d(j), j \in A\}$. Portanto $r = 5$.

6º Passo: $F = \{1,2,5,6,7,8\}$ *e* $A = \{3,4\}$

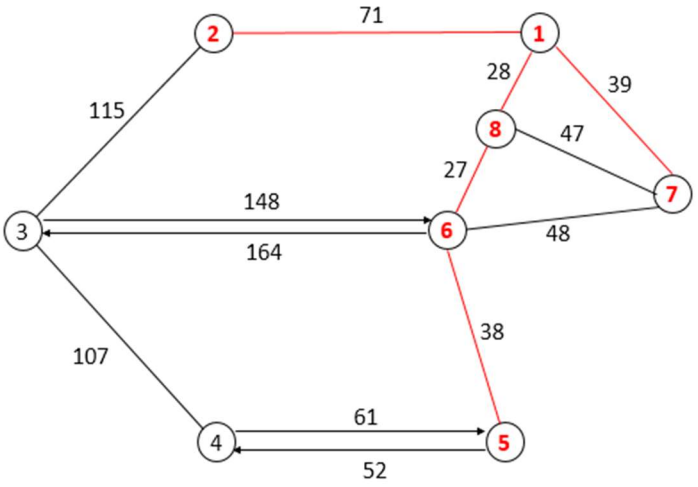
7º Passo: Como $r = 5 \neq 4$ o algoritmo prossegue. O único vértice aberto adjacente a 5 é o vértice 4. Então:

(1) $s = 4 \Rightarrow d(5) + m_{54} = 145$

(2) Como $d(4) = \infty > 145$ atualizamos $d(4) = 145$ e $a(4) = 5$

(3) Retorne para o 5º Passo.

Figura 44 – 5ª iteração do Algoritmo de Dijkstra



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	186	145	93	55	39	28
$a(j)$	–	1	2	5	6	8	1	1

$F = \{1,2,5,6,7,8\}; A = \{3,4\}$

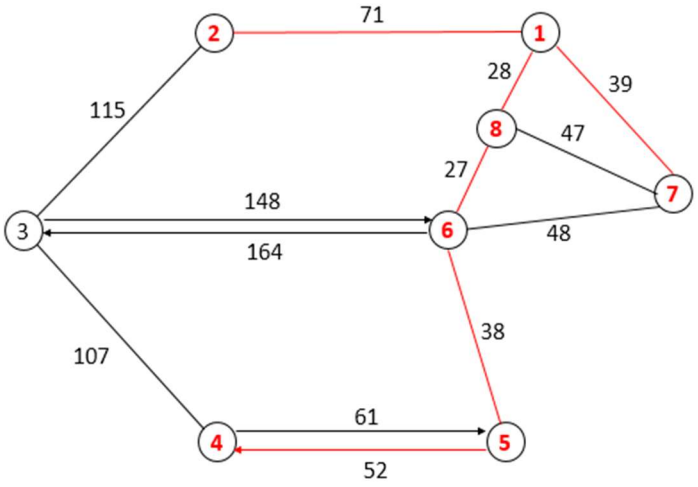
Fonte: O autor, 2018

5º Passo: Consultando a tabela da Figura 44 observamos que $d(4)$ é o mínimo do conjunto $\{d(j), j \in A\}$. Portanto $r = 4$.

6º Passo: $F = \{1,2,4,5,6,7,8\}$ *e* $A = \{3\}$

7º Passo: Como $r = 4$ o algoritmo o algoritmo cessa.

Figura 45 – Algoritmo de Dijkstra finalizado.



j	1	2	3	4	5	6	7	8
$d(j)$	0	71	186	145	93	55	39	28
$a(j)$	–	1	2	5	6	8	1	1

$F = \{1,2,4,5,6,7,8\}; A = \{3\}$

Fonte: O autor, 2018

Portanto, a distância percorrida de Campinas até Tatuí será de 145 km. Para respondermos à segunda pergunta do item a) comporemos o caminho utilizando a informação na lista $a(v)$.

Iniciamos por $a(4) = 5$, pois 4 é o nosso vértice destino. Verificamos então que $a(5) = 6$. O processo se repete até que $a(v) = 1$, que é o nosso vértice de partida. Assim:

$$a(4) = 5$$
$$a(5) = 6$$
$$a(6) = 8$$
$$a(8) = 1$$

Portanto o caminho percorrerá os vértices 1, 8, 6, 5 e 4, sendo 1 e 4 os pontos de partida e de chegada respectivamente. Consultando a tabela da figura x verificamos que o caminhão passará nas cidades de Indaiatuba, Itu e Sorocaba, nesta ordem, antes de chegar em Tatuí.

4.5 Tópico 5: O Algoritmo de Floyd-Warshall

Objetivo: Apresentar o passo a passo do Algoritmo de Floyd-Warshall.

Desenvolvimento:

- Apresentar historicamente Bernard Roy, Robert Floyd e Stephen Warshall.

Explicar a finalidade do algoritmo bem como o seu passo a passo. Como no algoritmo de Dijkstra, foi feita uma adequação da linguagem do conteúdo apresentado no capítulo 3 como descreveremos a seguir.

Seja G um grafo com n vértices. Baseado na Matriz de Manejo de G, devemos construir uma outra matriz $R_{n\times n} = [r_{ij}]$ denominada Matriz de Roteamento. Essa matriz fornecerá a "rota" que devemos estabelecer entre os vértices na construção do menor caminho entre dois vértices do grafo. Ela é definida como se segue:

$$r_{ij} = \begin{cases} j, se\ existe\ o\ arco\ (i,j) \\ 0, se\ não\ existe\ o\ arco\ (i,j) \end{cases}$$

O elemento r_{ij} indica qual é o vértice seguinte a i no menor caminho de i até j (caso exista). Inicialmente temos que o menor caminho entre dois vértices é igual ao peso do arco que os une. Dessa forma, $r_{ij} = j$ se existir o arco (i,j), ou seja, o vértice seguinte a i no caminho $i \to j$ é o próprio j. Caso não exista esse arco, não há inicialmente um caminho possível de i até j e indicamos a ausência desse caminho fazendo $r_{ij} = j$.

O algoritmo fará uma iteração para cada um de seus vértices e atualizará as matrizes de manejo e de roteamento. Em cada iteração k será verificado, com base na matriz de manejo, se o comprimento do caminho do vértice i até o vértice j, tendo k como intermediário, é menor do que o comprimento já existente entre esses vértices. Em símbolos:

$$m_{ik} + m_{kj} < m_{ij}.$$

Caso se verifique a última desigualdade substituímos o valor de m_{ij} por $m_{ik} + m_{kj}$ na matriz de manejo. Em seguida devemos atualizar a matriz de roteamento. Se para atingir j a partir de i devemos passar primeiro por k, então r_{ij} deve ser atualizado com o vértice seguinte a i no caminho $i \to k$, ou seja r_{ik}.

Esse processo se repete em cada iteração para todo $i, j \neq k$, já que ao tomarmos k como intermediário, não podemos tê-lo como vértice de origem ou de destino.

Cabe ressaltar que se $m_{ik} = \infty$ ou $m_{kj} = \infty$, a distância de i até j não será melhorada (reduzida) pois o membro esquerdo da desigualdade será sempre maior. Isso pode ser útil quando executamos o algoritmo manualmente, poupando checagens desnecessárias.

Para obter a sequências de vértices que determina o caminho mínimo de $i \to j$, devemos consultar o elemento r_{ij} da matriz de roteamento. Se $r_{ij} = a$, significa que o caminho passa por

a depois de i. Devemos agora localizar o elemento r_{aj}, que indicará o próximo vértice depois de a. Se $r_{aj} = b$ então b é o vértice seguinte a a no caminho mínimo. Esse processo continua até que $r_{xj} = j$, gerando o caminho

$$(i, a, b, \dots, j)$$

Abaixo temos o passo a passo do algoritmo.

Algoritmo de Floyd-Warshall

Seja $G = (V, E)$ um grafo com n vértices, onde $V = \{1,2, \dots, n\}$.

1º Passo: $k = \min(V)$. Remova k de V.

2º Passo: Para todo m_{ij}, com $i, j \neq k$, verifique se $m_{ik} + m_{kj} < m_{ij}$. Em caso afirmativo substitua m_{ij} por $m_{ik} + m_{kj}$ e r_{ij} por r_{ik}.

3º Passo: Se $k = n$ o algoritmo cessa. Caso contrário retorne ao 1º Passo.

4.6 Tópico 6: Aplicação do Algoritmo de Floyd-Warshall

Objetivo: Responder ao item b) do problema motivador.

Desenvolvimento:

- Retomar o item b) do problema motivador: "Suponha agora que a central de distribuição possa ser instalada em qualquer uma das cidades. Em qual delas você a instalaria? Por que?"
- Aplicar o passo a passo do algoritmo.

É mais vantajoso instalar o centro de distribuição na cidade que seja mais próxima de todas as outras, pois assim o custo com combustível será o menor possível. Em outras palavras, estamos procurando uma cidade cuja soma das distâncias até as outras seja a menor possível. Podemos aplicar o algoritmo de Floyd-Warshall para determinar qual é a menor distância entre todas as cidades e em seguida calcular a soma dessas distâncias para cada cidade. Resgatando o grafo do problema motivador temos as seguintes matrizes de manejo e de roteamento.

Figura 46 – Matrizes de Manejo e de Roteamento do Problema Motivador.

M	1	2	3	4	5	6	7	8
1	0	71	∞	∞	∞	∞	39	28
2	71	0	115	∞	∞	∞	∞	∞
3	∞	115	0	107	∞	148	∞	∞
4	∞	∞	107	0	61	∞	∞	∞
5	∞	∞	∞	52	0	38	∞	∞
6	∞	∞	164	∞	38	0	48	27
7	39	∞	∞	∞	∞	48	0	47
8	28	∞	∞	∞	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	0	0	0	0	7	8
2	1	0	3	0	0	0	0	0
3	0	2	0	4	0	6	0	0
4	0	0	3	0	5	0	0	0
5	0	0	0	4	0	6	0	0
6	0	0	3	0	5	0	7	8
7	1	0	0	0	0	6	0	8
8	1	0	0	0	0	6	7	0

Fonte: O autor, 2018

Para este grafo temos $V = \{1,2,3,4,5,6,7,8\}$. Vamos aplicar o algoritmo.

1ª Iteração

1º Passo: $\min(V) = 1$. Excluímos o vértice 1 de V: $V = \{2,3,4,5,6,7,8\}$

2º Passo:

Para $i = 2$ temos que

$$m_{21} + m_{17} < m_{27} \; e \; m_{21} + m_{18} < m_{28}.$$

Nos outros, $m_{23} = m_{24} = m_{25} = m_{26} = \infty$, logo as distâncias não serão melhoradas. Substituímos os valores de m_{27} e m_{28} com os valores das respectivas somas:

$$m_{27} = m_{21} + m_{17} = 120 \; e \; m_{28} = \; m_{21} + m_{18} = 99.$$

Em seguida substituímos o valor de r_{27} por $r_{21} = 1$, que é o vértice pelo qual se deve passar a partir de 2 para chegar a 1. O mesmo é feito com relação a m_{28}. A Figura 46 ilustra essas mudanças na 1ª iteração. A cada iteração sempre fixaremos a linha e coluna k na matriz de manejo e a coluna k na matriz de pesos de maneira a tornar o processo manual mais simples.

Figura 47 – Matrizes de Manejo e de Roteamento durante a 1ª iteração do algoritmo.

M	1	2	3	4	5	6	7	8
1	0	71	∞	∞	∞	∞	39	28
2	71	0	115	∞	∞	∞	120	99
3	∞	115	0	107	∞	148	∞	∞
4	∞	∞	107	0	61	∞	∞	∞
5	∞	∞	∞	52	0	38	∞	∞
6	∞	∞	164	∞	38	0	48	27
7	39	∞	∞	∞	∞	48	0	47
8	28	∞	∞	∞	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	0	0	0	0	7	8
2	1	0	3	0	0	0	1	1
3	0	2	0	4	0	6	0	0
4	0	0	3	0	5	0	0	0
5	0	0	0	4	0	6	0	0
6	0	0	3	0	5	0	7	8
7	1	0	0	0	0	6	0	8
8	1	0	0	0	0	6	7	0

Fonte: O autor, 2018

Para $i = 3,4,5,6$ temos $m_{31} = m_{41} = m_{51} = m_{61} = \infty$. Logo as distâncias 2,3,4,5 e 6, tomando o vértice 1 como intermediário, até os outros vértices não poderá ser melhorada (não há conexão desses vértices com o vértice 1). Assim, não alteraremos as linhas 2,3,4,5 e 6 nesta iteração.

Para $i = 7$ temos apenas

$$m_{71} + m_{12} < m_{72}.$$

Assim, $m_{72} = m_{71} + m_{12} = 120$ e $r_{72} = r_{71} = 1$

Para $i = 8$ temos apenas

$$m_{71} + m_{12} < m_{72}.$$

Assim, $m_{82} = m_{81} + m_{12} = 99$ e $r_{82} = r_{71} = 1$

3º Passo: Como $k = 1 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Ao final da 1ª iteração temos as seguintes matrizes de manejo e de roteamento conforme mostrado na Figura 47.

Figura 48 – Matrizes de Manejo e de Roteamento após a 1ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	∞	∞	∞	∞	39	28
2	71	0	115	∞	∞	∞	120	99
3	∞	115	0	107	∞	148	∞	∞
4	∞	∞	107	0	61	∞	∞	∞
5	∞	∞	∞	52	0	38	∞	∞
6	∞	∞	164	∞	38	0	48	27
7	39	120	∞	∞	∞	48	0	47
8	28	99	∞	∞	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	0	0	0	0	7	8
2	1	0	3	0	0	0	1	1
3	0	2	0	4	0	6	0	0
4	0	0	3	0	5	0	0	0
5	0	0	0	4	0	6	0	0
6	0	0	3	0	5	0	7	8
7	1	1	0	0	0	6	0	8
8	1	1	0	0	0	6	7	0

Fonte: O autor, 2018

2ª Iteração

1º Passo: $\min(V) = 2$. Excluímos o vértice 2 de V: $V = \{3,4,5,6,7,8\}$

2º Passo:

Figura 49 – Matrizes preparadas para o início das comparações da 2ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	∞	∞	∞	∞	39	28
2	71	0	115	∞	∞	∞	120	99
3	∞	115	0	107	∞	148	∞	∞
4	∞	∞	107	0	61	∞	∞	∞
5	∞	∞	∞	52	0	38	∞	∞
6	∞	∞	164	∞	38	0	48	27
7	39	120	∞	∞	∞	48	0	47
8	28	99	∞	∞	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	0	0	0	0	7	8
2	1	0	3	0	0	0	1	1
3	0	2	0	4	0	6	0	0
4	0	0	3	0	5	0	0	0
5	0	0	0	4	0	6	0	0
6	0	0	3	0	5	0	7	8
7	1	1	0	0	0	6	0	8
8	1	1	0	0	0	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ e

  $j = 3$: $m_{12} + m_{23} < m_{13}$. Assim $m_{13} = m_{12} + m_{23} = 186$ e $r_{13} = r_{12} = 2$;

  $j = 4,5,6,7,8$: $m_{12} + m_{2j} > m_{1j}$. Não há alterações a fazer.

- Para $i = 3$ e

  $j = 1$: $m_{32} + m_{21} < m_{31}$. Portanto $m_{31} = m_{32} + m_2 = 186$ e $r_{31} = r_{32} = 2$;

  $j = 4,5,6$: não há nada a se fazer pois $m_{3j} = \infty$;

  $j = 7,8$: $m_{32} + m_{27} < m_{37}$ e $m_{32} + m_{28} < m_{38}$. Então

  $m_{37} = m_{32} + m_{27} = 235$ e $r_{37} = r_{32} = 2$;

  $m_{38} = m_{32} + m_{28} = 214$ e $r_{38} = r_{32} = 2$.

- Para $i = 4,5,6$ não há nada a se fazer pois $m_{i2} = \infty$.

- Para $i = 7$ e

  $j = 1,4,5,6,8$: $m_{72} + m_{2j} > m_{7j}$ e não há nada a fazer;

  $j = 3$: $m_{72} + m_{23} < m_{73}$. Então $m_{73} = m_{72} + m_{23} = 235$ e $r_{73} = r_{72} = 1$.

- Para $i = 8$ e

  $j = 1,4,5,6,7$: $m_{82} + m_{2j} > m_{8j}$ e não há nada a fazer;

  $j = 3$: $m_{82} + m_{23} < m_{83}$. Então $m_{83} = m_{82} + m_{23} = 214$ e $r_{83} = r_{82} = 1$.

3º Passo: Como $k = 1 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Figura 50 – Matrizes de Manejo e de Roteamento após a 2ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	∞	∞	∞	39	28
2	71	0	115	∞	∞	∞	120	99
3	286	115	0	107	∞	148	235	214
4	∞	∞	107	0	61	∞	∞	∞
5	∞	∞	∞	52	0	38	∞	∞
6	∞	∞	164	∞	38	0	48	27
7	39	120	235	∞	∞	48	0	47
8	28	99	214	∞	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	0	0	0	7	8
2	1	0	3	0	0	0	1	1
3	2	2	0	4	0	6	2	2
4	0	0	3	0	5	0	0	0
5	0	0	0	4	0	6	0	0
6	0	0	3	0	5	0	7	8
7	1	1	1	0	0	6	0	8
8	1	1	1	0	0	6	7	0

Fonte: O autor, 2018

3ª Iteração

1º Passo: $\min(V) = 3$. Excluímos o vértice de V: $V = \{4,5,6,7,8\}$

2º Passo:

Figura 51 – Matrizes preparadas para o início das comparações da 3ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	∞	∞	∞	39	28
2	71	0	115	∞	∞	∞	120	99
3	286	115	0	107	∞	148	235	214
4	∞	∞	107	0	61	∞	∞	∞
5	∞	∞	∞	52	0	38	∞	∞
6	∞	∞	164	∞	38	0	48	27
7	39	120	235	∞	∞	48	0	47
8	28	99	214	∞	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	0	0	0	7	8
2	1	0	3	0	0	0	1	1
3	2	2	0	4	0	6	2	2
4	0	0	3	0	5	0	0	0
5	0	0	0	4	0	6	0	0
6	0	0	3	0	5	0	7	8
7	1	1	1	0	0	6	0	8
8	1	1	1	0	0	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ e

  $j = 2,5,7,8$: $m_{13} + m_{3j} > m_{1j}$ e não há nada a fazer;

  $j = 4,6$: $m_{13} + m_{34} < m_{14}$ e $m_{13} + m_{36} < m_{16}$. Então,

  $m_{14} = m_{13} + m_{34} = 393$ e $r_{14} = r_{13} = 2$;

  $m_{16} = m_{13} + m_{36} = 434$ e $r_{16} = r_{13} = 2$;

- Para $i = 2$ e

  $j = 1,5,7,8$: $m_{23} + m_{3j} > m_{1j}$ e não há nada a fazer;

  $j = 4,6$: $m_{23} + m_{34} < m_{24}$ e $m_{23} + m_{36} < m_{26}$. Então,

  $m_{24} = m_{23} + m_{34} = 263$ e $r_{24} = r_{23} = 3$;

  $m_{26} = m_{23} + m_{36} = 434$ e $r_{26} = r_{23} = 3$;

- Para $i = 4$ e

  $j = 1$: $m_{43} + m_{31} < m_{41}$. Portanto $m_{41} = m_{43} + m_{31} = 393$ e $r_{41} = r_{43} = 3$;

  $j = 2$: $m_{43} + m_{32} < m_{42}$. Portanto $m_{42} = m_{43} + m_{32} = 222$ e $r_{42} = r_{43} = 3$;

  $j = 5$: Não há nada a fazer pois $m_{35} = \infty$;

  $j = 6$: $m_{43} + m_{36} < m_{46}$. Portanto $m_{46} = m_{43} + m_{36} = 255$ e $r_{46} = r_{43} = 3$;

  $j = 7$: $m_{43} + m_{37} < m_{47}$. Portanto $m_{47} = m_{43} + m_{37} = 342$ e $r_{47} = r_{43} = 3$;

  $j = 8$: $m_{43} + m_{38} < m_{48}$. Portanto $m_{48} = m_{43} + m_{38} = 321$ e $r_{48} = r_{43} = 3$;

- Para $i = 5$ não há nada a fazer pois $m_{53} = \infty$;

- Para $i = 6$ e

  $j = 1$: $m_{63} + m_{31} < m_{61}$. Portanto $m_{61} = m_{63} + m_{31} = 450$ e $r_{61} = r_{63} = 1$;

  $j = 2$: $m_{63} + m_{32} < m_{62}$. Portanto $m_{62} = m_{63} + m_{32} = 279$ e $r_{62} = r_{63} = 1$;

  $j = 4$: $m_{63} + m_{34} < m_{64}$. Portanto $m_{64} = m_{63} + m_{34} = 271$ e $r_{64} = r_{63} = 1$;

  $j = 5$: Não há nada a fazer pois $m_{35} = \infty$;

  $j = 7{,}8$: $m_{63} + m_{3j} > m_{6j}$ e não há nada a fazer;

- Para $i = 7$ e

  $j = 1{,}2{,}5{,}6{,}8$: $m_{73} + m_{3j} > m_{7j}$ e não há nada a fazer;

  $j = 4$: $m_{73} + m_{34} < m_{74}$. Portanto $m_{74} = m_{73} + m_{34} = 342$ e $r_{74} = r_{73} = 1$;

- Para $i = 8$ e

  $j = 1{,}2{,}5{,}6{,}7$: $m_{83} + m_{3j} > m_{8j}$ e não há nada a fazer;

  $j = 4$: $m_{83} + m_{34} < m_{84}$. Portanto $m_{84} = m_{83} + m_{34} = 321$ e $r_{74} = r_{83} = 1$;

3º Passo: Como $k = 3 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Figura 52 – Matrizes de Manejo e de Roteamento após a 3ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	∞	434	39	28
2	71	0	115	222	∞	263	120	99
3	286	115	0	107	∞	148	235	214
4	393	222	107	0	61	255	342	321
5	∞	∞	∞	52	0	38	∞	∞
6	450	279	164	271	38	0	48	27
7	39	120	235	342	∞	48	0	47
8	28	99	214	321	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	0	2	7	8
2	1	0	3	3	0	3	1	1
3	2	2	0	4	0	6	2	2
4	3	3	3	0	5	3	3	3
5	0	0	0	4	0	6	0	0
6	3	3	3	3	5	0	7	8
7	1	1	1	1	0	6	0	8
8	1	1	1	1	0	6	7	0

Fonte: O autor, 2018

4ª Iteração

1º Passo: $\min(V) = 4$. Excluímos o vértice de V: $V = \{5,6,7,8\}$

2º Passo:

Figura 53 – Matrizes preparadas para o início das comparações da 4ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	∞	434	39	28
2	71	0	115	222	∞	263	120	99
3	286	115	0	107	∞	148	235	214
4	393	222	107	0	61	255	342	321
5	∞	∞	∞	52	0	38	∞	∞
6	450	279	164	271	38	0	48	27
7	39	120	235	342	∞	48	0	47
8	28	99	214	321	∞	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	0	2	7	8
2	1	0	3	3	0	3	1	1
3	2	2	0	4	0	6	2	2
4	3	3	3	0	5	3	3	3
5	0	0	0	4	0	6	0	0
6	3	3	3	3	5	0	7	8
7	1	1	1	1	0	6	0	8
8	1	1	1	1	0	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ e

  $j = 2,3,6,7,8$: $m_{14} + m_{4j} > m_{1j}$ e não há nada a fazer;

  $j = 5$: $m_{14} + m_{45} < m_{15}$. Portanto $m_{15} = m_{14} + m_{45} = 454$ e $r_{15} = r_{14} = 2$;

- Para $i = 2$ e

  $j = 1,3,6,7,8$: $m_{24} + m_{4j} > m_{2j}$ e não há nada a fazer;

  $j = 5$: $m_{24} + m_{45} < m_{15}$. Portanto $m_{25} = m_{24} + m_{45} = 283$ e $r_{25} = r_{24} = 3$;

- Para $i = 3$ e

  $j = 1,2,6,7,8$: $m_{34} + m_{4j} > m_{3j}$ e não há nada a fazer;

  $j = 5$: $m_{34} + m_{45} < m_{15}$. Portanto $m_{35} = m_{24} + m_{45} = 168$ e $r_{35} = r_{34} = 4$;

- Para $i = 5$ e

  $j = 1$: $m_{54} + m_{41} < m_{51}$. Portanto $m_{51} = m_{54} + m_{41} = 445$ e $r_{51} = r_{54} = 4$;

  $j = 2$: $m_{54} + m_{42} < m_{52}$. Portanto $m_{52} = m_{54} + m_{42} = 274$ e $r_{52} = r_{54} = 4$;

  $j = 3$: $m_{54} + m_{43} < m_{53}$. Portanto $m_{53} = m_{54} + m_{43} = 159$ e $r_{53} = r_{54} = 4$;

  $j = 6$: $m_{54} + m_{46} > m_{56}$ e não há nada a fazer;

  $j = 7$: $m_{54} + m_{47} < m_{57}$. Portanto $m_{57} = m_{54} + m_{47} = 394$ e $r_{57} = r_{54} = 4$;

  $j = 8$: $m_{54} + m_{48} < m_{58}$. Portanto $m_{58} = m_{54} + m_{48} = 373$ e $r_{51} = r_{54} = 4$;

- Para $i = 6$ não há nada a fazer pois $m_{64} + m_{4j} > m_{6j}$, para todo $j = 1,2,3,5,7,8$;

- Para $i = 7$ e

  $j = 1,2,3,6,8$: $m_{74} + m_{4j} > m_{7j}$ e não há nada a fazer;

  $j = 5$: $m_{74} + m_{45} < m_{75}$. Portanto $m_{75} = m_{74} + m_{45} = 403$ e $r_{75} = r_{74} = 1$;

- Para $i = 8$ e

$j = 1,2,3,6,7$: $m_{84} + m_{4j} > m_{8j}$ e não há nada a fazer;

$j = 5$: $m_{84} + m_{45} < m_{85}$. Portanto $m_{85} = m_{84} + m_{45} = 382$ e $r_{85} = r_{84} = 1$;

3º Passo: Como $k = 4 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Figura 54 – Matrizes de Manejo e de Roteamento após a 4ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	454	434	39	28
2	71	0	115	222	283	263	120	99
3	286	115	0	107	186	148	235	214
4	393	222	107	0	61	255	342	321
5	445	274	159	52	0	38	394	373
6	450	279	164	271	38	0	48	27
7	39	120	235	342	403	48	0	47
8	28	99	214	321	382	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	2	2	7	8
2	1	0	3	3	3	3	1	1
3	2	2	0	4	4	6	2	2
4	3	3	3	0	5	3	3	3
5	4	4	4	4	0	6	4	4
6	3	3	3	3	5	0	7	8
7	1	1	1	1	1	6	0	8
8	1	1	1	1	1	6	7	0

Fonte: O autor, 2018

5ª Iteração

1º Passo: $\min(V) = 5$. Excluímos o vértice de V: $V = \{6,7,8\}$

2º Passo:

Figura 55 – Matrizes preparadas para o início das comparações da 5ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	454	434	39	28
2	71	0	115	222	283	263	120	99
3	286	115	0	107	186	148	235	214
4	393	222	107	0	61	255	342	321
5	445	274	159	52	0	38	394	373
6	450	279	164	271	38	0	48	27
7	39	120	235	342	403	48	0	47
8	28	99	214	321	382	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	2	2	7	8
2	1	0	3	3	3	3	1	1
3	2	2	0	4	4	6	2	2
4	3	3	3	0	5	3	3	3
5	4	4	4	4	0	6	4	4
6	3	3	3	3	5	0	7	8
7	1	1	1	1	1	6	0	8
8	1	1	1	1	1	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ não há nada a fazer pois $m_{15} + m_{5j} > m_{1j}$, para todo $j = 2,3,4,6,7,8$;
- Para $i = 2$ não há nada a fazer pois $m_{25} + m_{5j} > m_{2j}$, para todo $j = 1,3,4,6,7,8$;
- Para $i = 3$ não há nada a fazer pois $m_{35} + m_{5j} > m_{3j}$, para todo $j = 1,2,4,6,7,8$;
- Para $i = 4$ e

  $j = 1,2,3,7,8$: $m_{45} + m_{5j} > m_{4j}$ e não há o que fazer;

  $j = 6$: $m_{45} + m_{56} < m_{46}$. Portanto $m_{46} = m_{45} + m_{56} = 99$ e $r_{46} = r_{45} = 5$;

- Para $i = 6$ e

  $j = 1,2,3,7,8$: $m_{65} + m_{5j} > m_{4j}$ e não há o que fazer;

  $j = 4$: $m_{65} + m_{54} < m_{64}$. Portanto $m_{64} = m_{65} + m_{54} = 90$ e $r_{64} = r_{65} = 5$;
- Para $i = 7$ não há nada a fazer pois $m_{75} + m_{5j} > m_{7j}$, para todo $j = 1,2,3,4,6,8$;
- Para $i = 8$ não há nada a fazer pois $m_{85} + m_{5j} > m_{8j}$, para todo $j = 1,2,3,4,7,8$;

3º Passo: Como $k = 5 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Figura 56 – Matrizes de Manejo e de Roteamento após a 5ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	454	434	39	28
2	71	0	115	222	283	263	120	99
3	286	115	0	107	186	148	235	214
4	393	222	107	0	61	99	342	321
5	445	274	159	52	0	38	394	373
6	450	279	164	90	38	0	48	27
7	39	120	235	342	403	48	0	47
8	28	99	214	321	382	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	2	2	7	8
2	1	0	3	3	3	3	1	1
3	2	2	0	4	4	6	2	2
4	3	3	3	0	5	5	3	3
5	4	4	4	4	0	6	4	4
6	3	3	3	5	5	0	7	8
7	1	1	1	1	1	6	0	8
8	1	1	1	1	1	6	7	0

Fonte: O autor, 2018

6ª Iteração

1º Passo: $\min(V) = 6$. Excluímos o vértice de V: $V = \{7,8\}$

2º Passo:

Figura 57 – Matrizes preparadas para o início das comparações da 6ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	454	434	39	28
2	71	0	115	222	283	263	120	99
3	286	115	0	107	186	148	235	214
4	393	222	107	0	61	99	342	321
5	445	274	159	52	0	38	394	373
6	450	279	164	90	38	0	48	27
7	39	120	235	342	403	48	0	47
8	28	99	214	321	382	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	2	2	7	8
2	1	0	3	3	3	3	1	1
3	2	2	0	4	4	6	2	2
4	3	3	3	0	5	5	3	3
5	4	4	4	4	0	6	4	4
6	3	3	3	5	5	0	7	8
7	1	1	1	1	1	6	0	8
8	1	1	1	1	1	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ não há nada a fazer pois $m_{16} + m_{6j} > m_{1j}$, para todo $j = 2,3,4,5,7,8$;
- Para $i = 2$ não há nada a fazer pois $m_{26} + m_{6j} > m_{2j}$, para todo $j = 1,3,4,5,7,8$;
- Para $i = 3$ e

$j = 1,2,4,5$: $m_{36} + m_{6j} > m_{3j}$ e não há nada a fazer;

$j = 7$: $m_{36} + m_{67} < m_{37}$. Portanto $m_{37} = m_{36} + m_{67} = 196$ e $r_{37} = r_{36} = 6$;

$j = 8$: $m_{36} + m_{68} < m_{38}$. Portanto $m_{38} = m_{36} + m_{68} = 175$ e $r_{37} = r_{36} = 6$;

- Para $i = 4$ e

  $j = 1,2,3,5$: $m_{46} + m_{6j} > m_{4j}$ e não há nada a fazer;

  $j = 7$: $m_{46} + m_{67} < m_{47}$. Portanto $m_{47} = m_{46} + m_{67} = 147$ e $r_{47} = r_{46} = 5$;

  $j = 8$: $m_{46} + m_{68} < m_{48}$. Portanto $m_{48} = m_{46} + m_{68} = 126$ e $r_{48} = r_{46} = 5$;

- Para $i = 5$ e

  $j = 1,2,3,4$: $m_{56} + m_{6j} > m_{5j}$ e não há nada a fazer;

  $j = 7$: $m_{56} + m_{67} < m_{57}$. Portanto $m_{57} = m_{56} + m_{67} = 86$ e $r_{57} = r_{56} = 6$;

  $j = 8$: $m_{46} + m_{68} < m_{48}$. Portanto $m_{58} = m_{56} + m_{68} = 65$ e $r_{58} = r_{56} = 6$;

- Para $i = 7$ e

  $j = 1,2,8$: $m_{76} + m_{6j} > m_{7j}$ e não há nada a fazer;

  $j = 3$: $m_{76} + m_{63} < m_{73}$. Portanto $m_{73} = m_{76} + m_{63} = 212$ e $r_{74} = r_{76} = 6$;

  $j = 4$: $m_{76} + m_{64} < m_{74}$. Portanto $m_{74} = m_{76} + m_{64} = 138$ e $r_{74} = r_{76} = 6$;

  $j = 5$: $m_{76} + m_{65} < m_{75}$. Portanto $m_{75} = m_{76} + m_{65} = 86$ e $r_{75} = r_{76} = 6$;

- Para $i = 8$ e

  $j = 1,2,7$: $m_{86} + m_{6j} > m_{7j}$ e não há nada a fazer;

  $j = 3$: $m_{86} + m_{63} < m_{83}$. Portanto $m_{83} = m_{86} + m_{63} = 191$ e $r_{83} = r_{86} = 6$;

  $j = 4$: $m_{86} + m_{64} < m_{84}$. Portanto $m_{84} = m_{86} + m_{64} = 117$ e $r_{84} = r_{86} = 6$;

  $j = 5$: $m_{86} + m_{65} < m_{85}$. Portanto $m_{85} = m_{86} + m_{65} = 65$e $r_{85} = r_{86} = 6$;

3º Passo: Como $k = 6 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Figura 58 – Matrizes de Manejo e de Roteamento após a 6ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	454	434	39	28
2	71	0	115	222	283	263	120	99
3	286	115	0	107	186	148	196	175
4	393	222	107	0	61	99	147	126
5	445	274	159	52	0	38	86	65
6	450	279	164	90	38	0	48	27
7	39	120	212	138	86	48	0	47
8	28	99	191	117	65	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	2	2	7	8
2	1	0	3	3	3	3	1	1
3	2	2	0	4	4	6	6	6
4	3	3	3	0	5	5	5	5
5	4	4	4	4	0	6	6	6
6	3	3	3	5	5	0	7	8
7	1	1	6	6	6	6	0	8
8	1	1	6	6	6	6	7	0

Fonte: O autor, 2018

7ª Iteração

1º Passo: $\min(V) = 7$. Excluímos o vértice de V: $V = \{8\}$

2º Passo:

Figura 59 – Matrizes preparadas para o início das comparações da 7ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	286	393	454	434	39	28
2	71	0	115	222	283	263	120	99
3	286	115	0	107	186	148	196	175
4	393	222	107	0	61	99	147	126
5	445	274	159	52	0	38	86	65
6	450	279	164	90	38	0	48	27
7	39	120	212	138	86	48	0	47
8	28	99	191	117	65	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	2	2	2	2	7	8
2	1	0	3	3	3	3	1	1
3	2	2	0	4	4	6	6	6
4	3	3	3	0	5	5	5	5
5	4	4	4	4	0	6	6	6
6	3	3	3	5	5	0	7	8
7	1	1	6	6	6	6	0	8
8	1	1	6	6	6	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ e

  $j = 2,8$: $m_{17} + m_{7j} > m_{1j}$ e não há nada a fazer;

  $j = 3$: $m_{17} + m_{73} < m_{13}$. Portanto $m_{13} = m_{17} + m_{73} = 251$ e $r_{13} = r_{17} = 7$;

  $j = 4$: $m_{17} + m_{74} < m_{14}$. Portanto $m_{14} = m_{17} + m_{74} = 177$ e $r_{14} = r_{17} = 7$;

  $j = 5$: $m_{17} + m_{75} < m_{15}$. Portanto $m_{15} = m_{17} + m_{75} = 125$ e $r_{15} = r_{17} = 7$;

  $j = 6$: $m_{17} + m_{76} < m_{16}$. Portanto $m_{16} = m_{17} + m_{76} = 87$ e $r_{16} = r_{17} = 7$;

- Para $i = 2$ e

  $j = 1,3,4,5,8$: $m_{27} + m_{7j} > m_{2j}$ e não há nada a fazer;

  $j = 5$: $m_{27} + m_{75} < m_{25}$. Portanto $m_{25} = m_{27} + m_{75} = 206$ e $r_{25} = r_{27} = 1$;

  $j = 6$: $m_{27} + m_{76} < m_{26}$. Portanto $m_{26} = m_{27} + m_{76} = 168$ e $r_{27} = r_{27} = 1$;

- Para $i = 3$ e

  $j = 1$: $m_{37} + m_{71} < m_{31}$. Portanto $m_{31} = m_{37} + m_{71} = 186$ e $r_{31} = r_{37} = 6$;

  $j = 2,4,5,6,8$: $m_{37} + m_{7j} > m_{3j}$ e não há nada a fazer;

- Para $i = 4$ e

  $j = 1$: $m_{47} + m_{71} < m_{41}$. Portanto $m_{41} = m_{47} + m_{71} = 186$ e $r_{41} = r_{47} = 6$;

  $j = 2,3,5,6,8$: $m_{47} + m_{7j} > m_{3j}$ e não há nada a fazer;

- Para $i = 5$ e

  $j = 1$: $m_{57} + m_{71} < m_{51}$. Portanto $m_{51} = m_{57} + m_{71} = 186$ e $r_{51} = r_{57} = 5$;

  $j = 2$: $m_{57} + m_{72} < m_{52}$. Portanto $m_{52} = m_{57} + m_{71} = 206$ e $r_{52} = r_{57} = 5$;

  $j = 3,4,6,8$: $m_{57} + m_{7j} > m_{5j}$ e não há nada a fazer;

- Para $i = 6$ e

  $j = 1$: $m_{67} + m_{71} < m_{61}$. Portanto $m_{61} = m_{67} + m_{71} = 87$ e $r_{61} = r_{67} = 7$;

  $j = 2$: $m_{67} + m_{72} < m_{52}$. Portanto $m_{62} = m_{67} + m_{72} = 168$ e $r_{62} = r_{67} = 7$;

  $j = 3,4,5,8$: $m_{57} + m_{7j} > m_{5j}$ e não há nada a fazer;

- Para $i = 8$ não há nada a fazer pois $m_{87} + m_{7j} > m_{8j}$, para todo $j = 1,2,3,4,5,6,8$;

3º Passo: Como $k = 7 \neq 8$ o algoritmo prossegue e retornamos ao 1º Passo.

Figura 60 – Matrizes de Manejo e de Roteamento após a 7ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	251	177	125	87	39	28
2	71	0	115	222	206	168	120	99
3	245	115	0	107	186	148	196	175
4	186	222	107	0	61	99	147	126
5	125	206	159	52	0	38	86	65
6	87	168	164	90	38	0	48	27
7	39	120	212	138	86	48	0	47
8	28	99	191	117	65	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	7	7	7	7	7	8
2	1	0	3	3	1	1	1	1
3	6	2	0	4	4	6	6	6
4	5	3	3	0	5	5	5	5
5	6	6	4	4	0	6	6	6
6	7	7	3	5	5	0	7	8
7	1	1	6	6	6	6	0	8
8	1	1	6	6	6	6	7	0

Fonte: O autor, 2018

8ª Iteração

1º Passo: $\min(V) = 8$. Excluímos o vértice de V: $V = \{\}$

2º Passo:

Figura 61 – Matrizes preparadas para o início das comparações da 8ª iteração.

M	1	2	3	4	5	6	7	8
1	0	71	251	177	125	87	39	28
2	71	0	115	222	206	168	120	99
3	245	115	0	107	186	148	196	175
4	186	222	107	0	61	99	147	126
5	125	206	159	52	0	38	86	65
6	87	168	164	90	38	0	48	27
7	39	120	212	138	86	48	0	47
8	28	99	191	117	65	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	7	7	7	7	7	8
2	1	0	3	3	1	1	1	1
3	6	2	0	4	4	6	6	6
4	5	3	3	0	5	5	5	5
5	6	6	4	4	0	6	6	6
6	7	7	3	5	5	0	7	8
7	1	1	6	6	6	6	0	8
8	1	1	6	6	6	6	7	0

Fonte: O autor, 2018

- Para $i = 1$ e

  $j = 2,7$: $m_{18} + m_{87} > m_{1j}$ e não há nada a fazer;

  $j = 3$: $m_{18} + m_{83} < m_{13}$. Portanto $m_{13} = m_{18} + m_{83} = 219$ e $r_{13} = r_{18} = 8$;

$j = 4$: $m_{18} + m_{84} < m_{14}$. Portanto $m_{14} = m_{18} + m_{84} = 205$ e $r_{14} = r_{18} = 8$;

$j = 5$: $m_{18} + m_{85} < m_{15}$. Portanto $m_{15} = m_{18} + m_{85} = 93$ e $r_{15} = r_{18} = 8$;

$j = 6$ $m_{18} + m_{85} < m_{16}$. Portanto $m_{16} = m_{18} + m_{86} = 55$ e $r_{16} = r_{18} = 8$;

- Para $i = 2$ e

  $j = 1,3,7$: $m_{28} + m_{8j} > m_{2j}$ e não há nada a fazer;

  $j = 4$: $m_{28} + m_{84} < m_{24}$. Portanto $m_{24} = m_{28} + m_{84} = 216$ e $r_{24} = r_{28} = 1$;

  $j = 5$: $m_{28} + m_{85} < m_{25}$. Portanto $m_{25} = m_{28} + m_{85} = 164$ e $r_{25} = r_{28} = 1$;

  $j = 6$: $m_{28} + m_{86} < m_{26}$. Portanto $m_{26} = m_{28} + m_{86} = 126$ e $r_{26} = r_{28} = 1$;

- Para $i = 3$ e

  $j = 2,4,5,6,7$: $m_{38} + m_{8j} > m_{3j}$ e não há nada a fazer;

  $j = 1$: $m_{38} + m_{81} < m_{31}$. Portanto $m_{31} = m_{38} + m_{81} = 203$ e $r_{34} = r_{38} = 6$;

- Para $i = 4$ e

  $j = 2,3,5,6,7$: $m_{48} + m_{8j} > m_{4j}$ e não há nada a fazer;

  $j = 1$: $m_{48} + m_{81} < m_{41}$. Portanto $m_{41} = m_{48} + m_{81} = 154$ e $r_{41} = r_{48} = 5$;

- Para $i = 5$ e

  $j = 3,4,6,7$: $m_{58} + m_{8j} > m_{5j}$ e não há nada a fazer;

  $j = 1$: $m_{58} + m_{81} < m_{51}$. Portanto $m_{51} = m_{58} + m_{81} = 93$ e $r_{51} = r_{58} = 6$;

  $j = 2$: $m_{58} + m_{82} < m_{52}$. Portanto $m_{52} = m_{58} + m_{82} = 164$ e $r_{52} = r_{58} = 6$;

- Para $i = 6$ e

  $j = 3,4,5,7$: $m_{68} + m_{8j} > m_{6j}$ e não há nada a fazer;

  $j = 1$: $m_{68} + m_{81} < m_{61}$. Portanto $m_{61} = m_{68} + m_{81} = 55$ e $r_{61} = r_{68} = 8$;

  $j = 2$: $m_{68} + m_{82} < m_{52}$. Portanto $m_{62} = m_{68} + m_{82} = 126$ e $r_{62} = r_{68} = 8$;

- Para $i = 7$ não há nada a fazer pois $m_{78} + m_{8j} > m_{7j}$ para $j = 1,2,3,4,5,6$;

3º Passo: Como $k = 8$ o algoritmo cessa. Na Figura 62 ilustra as matrizes Manejo e de Roteamento após o término do algoritmo.

Figura 62 – Matrizes de Manejo e de Roteamento ao fim do algoritmo.

M	1	2	3	4	5	6	7	8
1	0	71	219	145	93	55	39	28
2	71	0	115	216	164	126	120	99
3	203	115	0	107	186	148	196	175
4	154	222	107	0	61	99	147	126
5	93	164	159	52	0	38	86	65
6	55	126	164	90	38	0	48	27
7	39	120	212	138	86	48	0	47
8	28	99	191	117	65	27	47	0

R	1	2	3	4	5	6	7	8
1	0	2	8	8	8	8	7	8
2	1	0	3	1	1	1	1	1
3	6	2	0	4	4	6	6	6
4	5	3	3	0	5	5	5	5
5	6	6	4	4	0	6	6	6
6	8	8	3	5	5	0	7	8
7	1	1	6	6	6	6	0	8
8	1	1	6	6	6	6	7	0

Fonte: O autor, 2018

Finalmente podemos responder a pergunta do item b) do problema motivador. Vamos calcular, para cada cidade, a soma das distâncias até todas as outras.

Figura 63 – Matriz de manejo e soma dos menores caminhos.

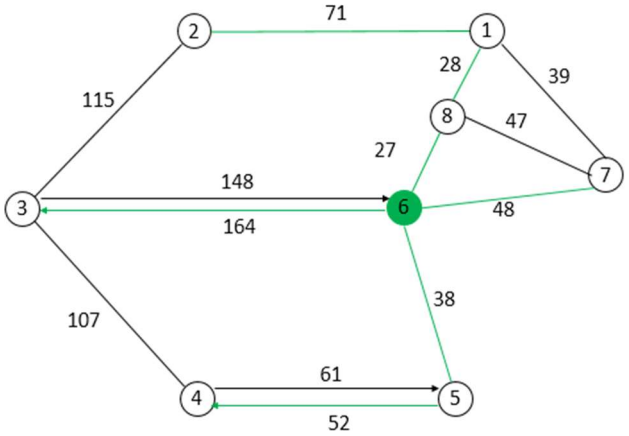
	1	2	3	4	5	6	7	8	Soma
1	0	71	219	145	93	55	39	28	650
2	71	0	115	216	164	126	120	99	911
3	203	115	0	107	186	148	196	175	1130
4	154	222	107	0	61	99	147	126	916
5	93	164	159	52	0	38	86	65	657
6	55	126	164	90	38	0	48	27	**548**
7	39	120	212	138	86	48	0	47	690
8	28	99	191	117	65	27	47	0	574

Cidade	Rótulo
Campinas	1
Piracicaba	2
Botucatu	3
Tatuí	4
Sorocaba	5
Itu	6
Jundiaí	7
Indaiatuba	8

Fonte: O autor, 2018

Dessa forma, a cidade que é menos distante de todas as outras é a indicada pelo rótulo 6, ou seja, Itu.

Figura 64 – Melhor escolha para alocar o centro de distribuição.



Legenda	
Cidade	Rótulo
Campinas	1
Piracicaba	2
Botucatu	3
Tatuí	4
Sorocaba	5
Itu	6
Jundiaí	7
Indaiatuba	8

Fonte: O autor, 2018

Embora o desenvolvimento do algoritmo, indicando todas as comparações, seja longo, sua execução num quadro ou numa apresentação em slides é relativamente mais curta.

5 CONSIDERAÇÕES FINAIS

Esse trabalho foi pensado de maneira a oferecer aos alunos um problema contextualizado em que eles possam aplicar um conceito matemático para resolvê-lo. Contudo, qualquer problema que envolva determinar o caminho mínimo em um grafo pode ser utilizado em seu lugar.

Os tópicos apresentados na última parte desse trabalho foram construídos de maneira a dar uma direção na condução de uma oficina sobre o assunto. Infelizmente, devido a problemas com o tempo, não foi possível aplica-los numa sala de aula, o que limitou de maneira significativa sua construção. Esperamos uma futura oportunidade para aplicação do mesmo e, de acordo com os resultados colhidos, remodela-los.

É possível trabalhar em paralelo com a disciplina de informática educativa e implementar ambos os algoritmos no ambiente *Visualg,* utilizando os respectivos fluxogramas e pseudo-códigos descritos nos apêndices. Acreditamos que dessa forma o aluno consolidará o pensamento algorítmico e uma nova janela de opções acadêmicas será aberta para ele.

REFERÊNCIAS

BACKES, André. **Algoritmos e Fluxogramas**. Notas de Aula.

BOAVENTURA, P. Oswaldo; JURKIEWICZ, S. **Grafos: Introdução e Prática**. 2 ed. São Paulo: Blucher, 2011.

BOYER, Carl B. **História da Matemática**. Tradução de Elza Gomide. 2ª ed. São Paulo: Edgard Blücher, 1996.

DIESTEL, R. **Graph Theory**. 2 ed. New York: Sping-Verlag, 2000.

DIJKSTRA, Edsger W.; MISA, Thomas J. An Interwiew with Edsger W. Dijkstra. **Comunications of the ACM**, v. 53, n. 8, p. 41-47, ago. 2010.

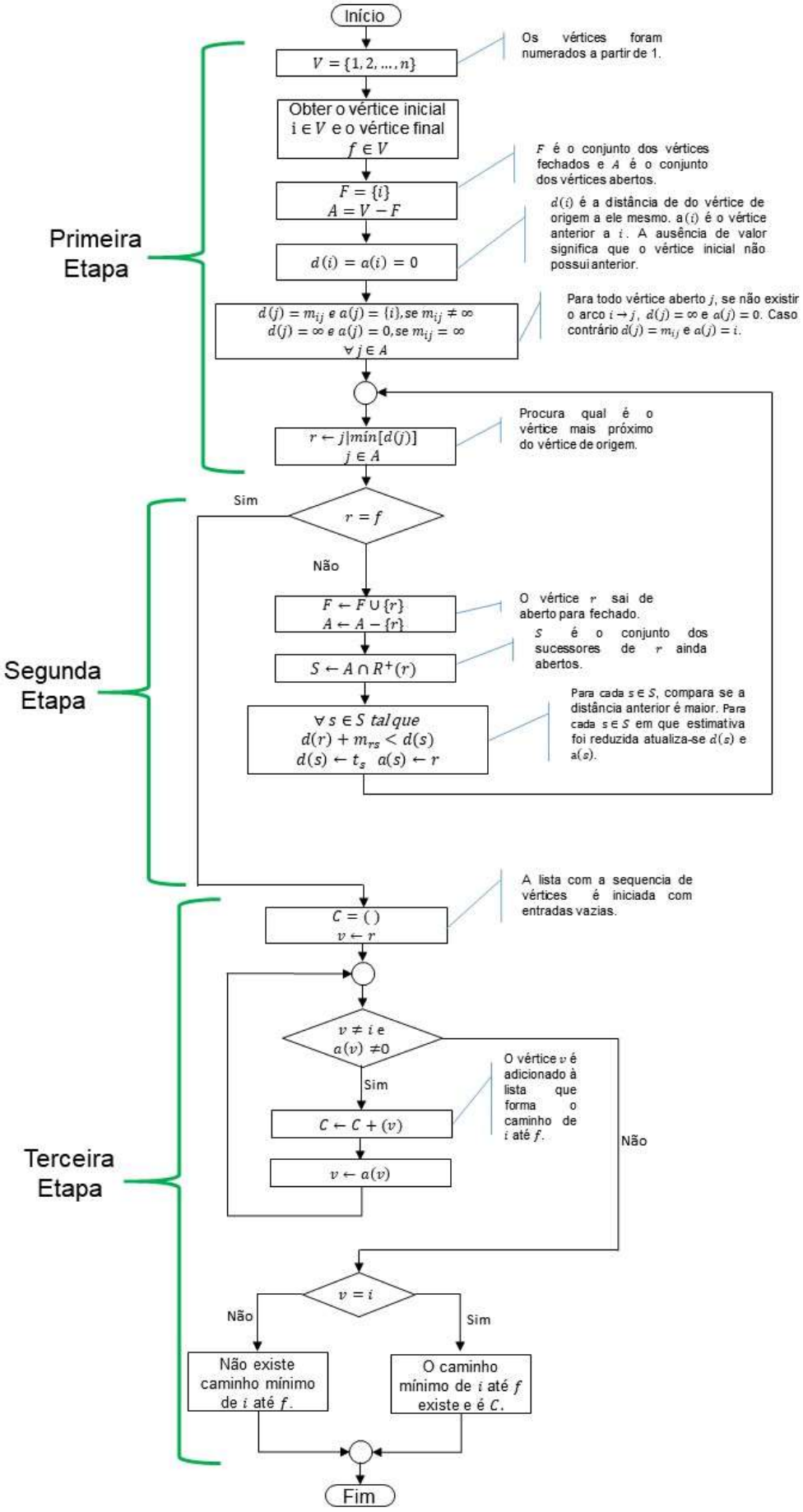
BARROS, Edson; PAMBOUKIAN, Sérgio; ZAMBONI, Lincoln, **Algoritmo de Dijkstra: Apoio diático na implementação, simulação e utilização computacional**, São Paulo, 2007.

FARIAS, Gilberto; MEDEIROS, Eduardo Santana. **Introdução à Computação**, [S.l.]: 2013. Versão 1.0. Disponível em < http://producao.virtual.ufpb.br/books/gilbertofarias/introducao-a-computacao-livro/livro/livro.pdf>. Acesso em: 12 de Janeiro de 2018.

HAIGH, Thomas. Biographies. **IEEE Annals of the History of Computing**, p.75-83, Abr./Jun. 2004.

SCHIENERMAN, Edward R. **Matemática Discreta: Uma introdução**. Tradução de All Tasks. 1 ed. São Paulo: Cengage Learning, 2011.

APÊNDICE A – FLUXOGRAMA DO ALGORITMO DE DIJKSTRA



APÊNDICE B – PSEUDOCÓDIGO DO ALGORITMO DE DIJKSTRA

$Dijkstra(V, M, i, f) = (d(f), C)$ O algoritmo é uma função que com quatro parâmetros de entrada (o conjunto de vértices, a matriz de manejo do grafo, o vértice inicial e o vértice final) e dois de saída (a distância mínima e o caminho percorrido).

início

$d(i) = 0, a(i) = 0\ F = \{i\}, A = V - F$

$$d(j) = \begin{cases} m_{ij}\ se\ m_{ij} \neq \infty \\ \infty\ se\ m_{ij} = \infty \end{cases}, a(j) = \begin{cases} i\ se\ m_{ij} \neq \infty \\ 0\ se\ m_{ij} = \infty \end{cases};\ \forall\ j \in A$$

$r \leftarrow j \mid mín[d(j)], j \in A$

enquanto $r \neq f$ **fazer**

$r \leftarrow v | mín[d(j)], j \in A$ Busca o vértice aberto mais próximo da origem.

$F \leftarrow F \cup \{r\}$ "Fecha" o vértice mais próximo da origem.

$A \leftarrow A - \{r\}$ Remove o vértice fechado de A.

$S \leftarrow A \cap R^{+}(r)$ Cria o conjunto S dos vértices adjacentes a r que ainda estão abertos.

para todo $s \in S$ **fazer**

$p_s = mín[d(s), d(r) + m_{rs}]\ \forall\ s \in S$ Compara as distâncias.

se $p_s < d(s)$ **então**

$d(s) \leftarrow p_s$ Atualiza o valor da menor distância.

$a(s) \leftarrow r$ Atualiza o parâmetro anterior.

fim-se

fim-para todo

fim-enquanto

$C = ( )$

$v \leftarrow r$

enquanto $v \neq i$ e $a(v) \neq 0$ **fazer**

$C \leftarrow C + (v)$ e $v \leftarrow a(v)$ Insere o vértice v no vetor caminho e atualiza v.

fim-enquanto

se $v = i$ **então**

O caminho mínimo de i até f existe, mede $d(f)$ e é C.

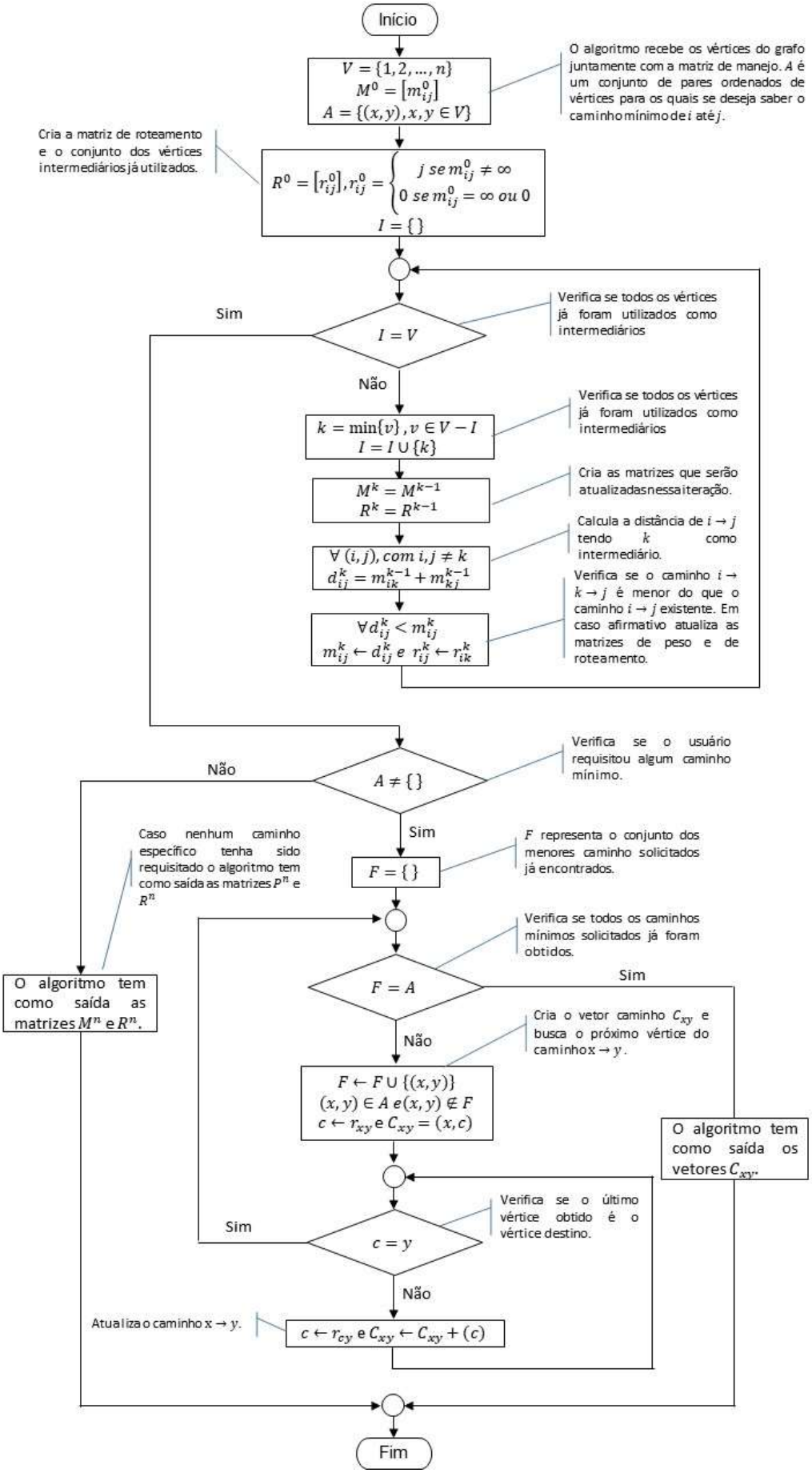
senão

Não existe caminho de i até f.

fim-se

fim

APÊNDICE C – FLUXOGRAMA DO ALGORITMO DE FLOYD-WARSHAL



APÊNDICE D – PSEUDOCÓDIGO DO ALGORITMO DE FLOYD-WARSHALL

$floydwarshall(M, R, A) = (M^*, R^*, C)$

O Algoritmo tem como entradas a matriz de manejo, a matriz de roteamento e um conjunto de pares ordenados para os quais se deseja saber o menor caminho.

início

$M^0 \leftarrow M, R^0 \leftarrow R$

para $k = 1, \ldots, n$ **fazer**

$M^k \leftarrow M^{k-1}$ e $R^k \leftarrow R^{k-1}$ Cria as matrizes de manejo e de roteamento que serão atualizadas.

para todo $i \neq k$ **fazer**

para todo $j \neq k$ **fazer**

Se $m_{ik}^k + m_{kj}^k < m_{ij}^k$ **então**

$m_{ij}^k \leftarrow m_{ik}^k + m_{kj}^k$

$r_{ij}^k \leftarrow r_{ik}^k$

Compara se a distância do vértice i ao vértice j tomando k como intermediário é menor que a distância existente entre esses vértices. Em caso afirmativo atualiza as matrizes de manejo e de roteamento.

fim-se

fim-para todo

fim-para todo

fim-para

$M^* \leftarrow M^n, R^* \leftarrow R^n$

se $A = \{\}$ **fazer**

$C = \{\}$

senão

para todo $(x, y) \in A$ **fazer**

$c \leftarrow r_{xy}^*$

$C_{xy} = (x, c)$

enquanto $c \neq n$ **fazer**

$c \leftarrow r_{xc}^*$

$C_{xy} \leftarrow (C_{xy}, c)$

Para todo par (x, y) em A, descreve cria o vetor C_{xy} que contém os rótulos dos vetores que compõe o caminho mínimo entre x e v.

fim-enquanto

fim-para-todo

$C = \{C_{xy}, \forall\ (x, y) \in A\}$

fim-se

fim

APÊNDICE F – SITUAÇÕES MOTIVADORAS PARA A INTRODUÇÃO DO CONCEITO DE ALGORITMOS

Situação A: “Como preparar uma pipoca?”

Num primeiro momento, com a ajuda dos alunos, descreve-se o passo a passo do preparo de uma pipoca. Os passos sugeridos pelos alunos devem ser escritos no quadro, em ordem e ser o mais claro possível. Abaixo temos um exemplo de uma lista de passos para essa tarefa.

Preparo de uma Pipoca Salgada

Início

1º Passo: Pegar milho de pipoca e uma pipoqueira no armário;

2º Passo: Adicionar o milho de pipoca e óleo à pipoqueira. Levar pipoqueira ao fogo;

3º Passo: Mexer a manivela da pipoqueira até que parem os estouros;

4º Passo: Servir a pipoca com sal a gosto.

Fim

Situação B: “Na escola de Pedro o ano letivo é dividido em três trimestres. Para compor a Média Anual (MA) calcula-se a média ponderada das três notas, atribuindo-se peso 3 às duas primeiras e peso 4 à última. Se a Média Anual for maior ou igual a 7, o aluno é aprovado na disciplina, caso contrário, o aluno deverá fazer a Avaliação Final (AF). Calcula-se então a Média Final (MF), obtida pela média ponderada entre a Média Anual e a Avaliação Final, com pesos 2 e 3 respectivamente. Se a média final é maior ou igual a 5, o aluno é aprovado, caso contrário ele é retido naquela disciplina. ”

Algoritmo Verificador de Aprovação/Reprovação

Início

1º Passo: Calcula-se a Média Anual (MA) das notas dos três trimestres;

$$MA = \frac{3 * Nota\ I + 3 * Nota\ II + 4 * Nota\ III}{10}$$

2º Passo: Se $MA \geq 7{,}0$ o algoritmo cessa e o aluno está aprovado. Caso contrário o aluno deverá fazer a Avaliação Final (AF). Vá para o 3º passo.

3º Passo: Calcula-se a média final (MF)

$$MF = \frac{3 * MA + 2 * MF}{5}$$

4º Passo: Se $MF \geq 5{,}0$ o aluno é aprovado. Caso contrário ele é reprovado.

Fim

APÊNDICE G – RESUMO HISTÓRICO DO SURGIMENTO DOS ALGORITMOS

A origem da palavra remete ao sobrenome do matemático persa Mohammed Ibn Musa al-Khwarizmi e seu significado encerra a descrição sistemática da maneira de se realizar alguma tarefa (BOYER, 1964). Dessa forma, qualquer conjunto finito de instruções que resolvem um problema é um algoritmo. A definição inclui não apenas algoritmos quem envolvam cálculos (como o Algoritmo de Euclides para obter o máximo divisor comum de um número) mas qualquer conjunto finito de regras que possa ser executado por uma pessoa contanto que ela possua o tempo necessário para fazê-lo. Essa definição abarca todos os programas de computadores existentes. Contudo, a implementação de algoritmos em máquinas ocorreu bem antes do surgimento de computadores como os conhecemos hoje.

O primeiro registro que se tem da implementação de um algoritmo remonta à revolução industrial. Em 1801, o mecânico e tecelão francês Joseph Marie Jacquard (1752-1834) construiu um tear mecânico controlado por grandes cartões perfurados.

Influenciado por Jacquard, o matemático inglês Charles Babagge propôs em 1837 a construção de uma máquina de propósito mais genérico chamada *Máquina Analítica*. Ele percebeu que poderia utilizar os cartões perfurados para armazenar ideias abstratas ou números, que poderiam ser acessados posteriormente, o que chamamos hoje num computador de *memória*. Infelizmente a Máquina Analítica nunca foi construída.

Durante o período em que Babagge tentava construir a Máquina Analítica, Ada Byron, filha de Lord Byron, mostrou bastante interesse pelo projeto e propôs diversas melhorias como relata Farias:

> A condessa de Lovelace, Ada Byron, se interessou pela máquina analítica de Babbage e se comunicava com ele através de cartas e encontros. Ela passou a escrever programas que a máquina poderia ser capaz de executar, caso fosse construída. Ela foi a primeira a reconhecer a necessidade de loops e sub-rotinas. Por esta contribuição, Ada ficou reconhecida na história como a primeira programadora. (FARIAS; MEDEIROS, 2013, p. 9)

Na primeira metade do século XX, Alan Turing formalizou o conceito de algoritmo como o conhecemos hoje através de uma invenção sua, a *Máquina de Turing*. Durante a Segunda Guerra Mundial, foi chefe do setor responsável por decifrar a máquina de criptografia alemã *Enigma*. Suas contribuições delinearam o campo da ciência da computação, sendo aclamado como o pai da ciência da computação.