

Teoria dos Grafos



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Caminhos em Grafos

Responsável pelo Conteúdo:

Prof. Dr. Cleber Silva Ferreira da Luz

Revisão Textual:

Prof. Me. Luciano Vieira Francisco

UNIDADE

Caminhos em Grafos



- Introdução;
- Caminhos em Grafos.



OBJETIVO DE APRENDIZADO

- Conhecer todos os conceitos de caminhos em grafos, bem como os principais algoritmos para obter caminhos em grafos.

Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Introdução

Nesta Unidade estudaremos o conceito de **caminho**, que é importante em grafos. Dado um grafo G , podemos compreender caminho como uma sequência de vértices conectados por uma sequência de arestas. Esta conexão permite sair do vértice v e chegar a um vértice w .

O conceito de caminho é importante para resolver diversos problemas tidos como modelos por meio dos grafos – um dos quais é o problema para encontrar o menor caminho entre um vértice v e um vértice w . Este problema aparece em diversas aplicações, tais como **encontrar uma rota com o menor caminho percorrido em uma cidade**, **encontrar um menor caminho para conectar um computador ao outro**, entre outros. Serão abordados todos os aspectos que envolvem caminhos em grafos.

Caminhos em Grafos

Um caminho pode ser definido como uma sequência de vértices tal que para cada vértice da sequência há uma aresta para o próximo vértice da sequência (CORMEN *et al.*, 2002). Formalmente, um caminho pode ser denotado como:

$$C_{(VE)} = \{<V_1, V_2 \dots V_k> \mid <V_i, V_{i+1}>, \in E, 1 \leq i < K\}$$

Em diversas aplicações envolvendo grafos, muitas vezes é necessário obter menor caminho entre os vértices v e w . As próximas seções abordarão o problema para encontrar o menor caminho, dado um par de vértices em um grafo G .

Caminho Mais Curto em Grafos Não Ponderados

Segundo Goldbarg e Goldbarg (2012), o **caminho mais curto** entre dois vértices v e w de um grafo G não ponderado é aquele que acumula a **menor quantidade de arestas** entre v e w . Analisaremos um exemplo: A Figura 1a apresenta um grafo, de modo que obteremos um caminho entre os vértices v e w ; a Figura 1b apresenta o menor caminho possível entre os vértices v e w .

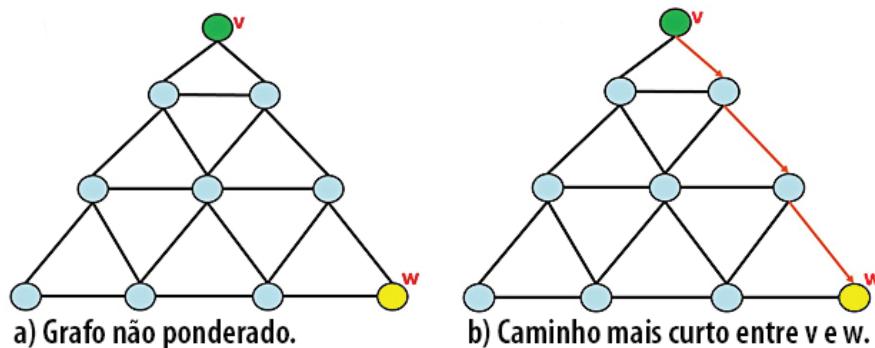


Figura 1 – Caminho mais curto em grafos não ponderados

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Caminho Mais Curto em Grafos Ponderados

Em grafos ponderados, o caminho mais curto entre dois vértices v e w de um grafo G é o caminho cuja **soma dos pesos das arestas** possui o menor valor possível entre todos os caminhos entre v e w (GOLDBARG; GOLDBARG, 2012). A Figura 2b apresenta o menor caminho possível entre os vértices v e w do grafo apresentado na Figura 2a:

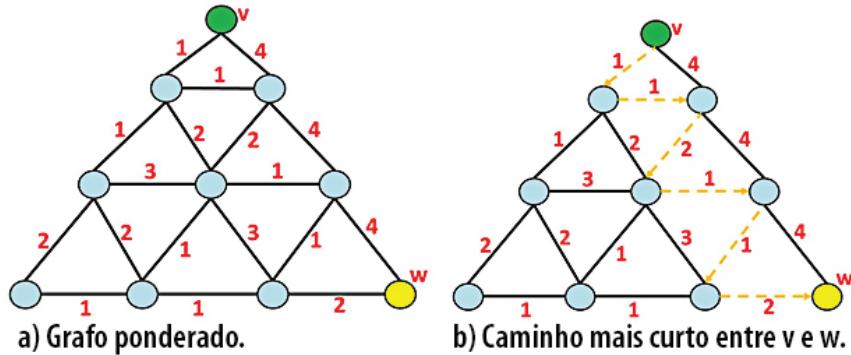


Figura 2 – Caminho mais curto em grafos ponderados

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Caminho Mais Longo em Grafos

Dado um **grafo G ponderado**, o **caminho mais longo** que existe em G é o que **acumula o maior valor possível** entre todos os caminhos cabíveis entre v e w . Em um **grafo não ponderado**, o **caminho mais longo** é aquele que **percorre o maior número de arestas** entre v e w (GOLDBARG; GOLDBARG, 2012). Considerando o grafo não ponderado apresentado na Figura 1a, a Figura 3a ilustra o maior caminho deste grafo; já a Figura 3b apresenta o maior caminho para o grafo ponderado:

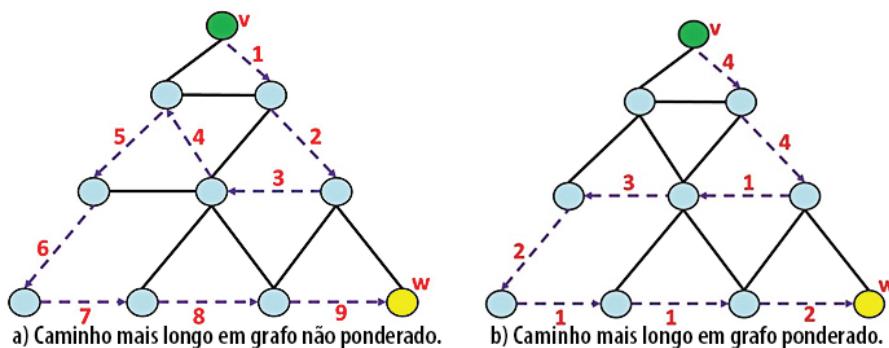


Figura 3 – Caminho mais longo em grafos

Fonte: Adaptada de Goldbarg e Goldbarg (2012)

Caminho Mais Curto com Custo nos Vértices

Dado dois vértices v e w em um grafo G ponderado em vértices, o caminho mais curto entre v e w é aquele cuja soma dos pesos das arestas e dos vértices possui o menor valor possível entre os demais caminhos entre v e w (GOLDBARG; GOLD-

BARG, 2012). Analisaremos um exemplo: a Figura 4a apresenta um grafo **G** ponderado; já a Figura 4b apresenta o caminho mais curto entre **v** e **w**. No processamento para achar o caminho mais curto, sempre que um vértice é incluído no caminho, o custo de seu vértice é adicionado ao custo do caminho.

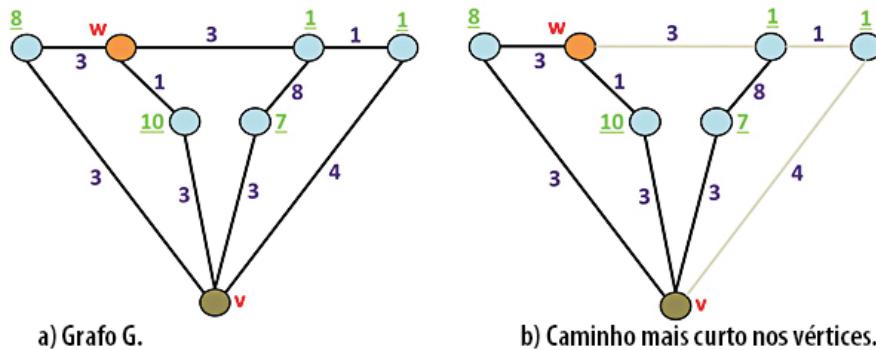


Figura 4 – Caminho mais curto com custos nos vértices

Fonte: Adaptada de Goldbarg e Goldbarg (2012)

Algoritmo de Dijkstra

Existem diversos algoritmos para achar o caminho mais curto entre dois vértices em um grafo. Em 1959, o cientista da computação Edsger Dijkstra propôs um algoritmo eficiente para obter o caminho mais curto em um grafo. A técnica utilizada por este algoritmo consiste em selecionar o vértice de menor potencial (CORMEN *et al.*, 2002; GOLDBARG; GOLDBARG, 2012). Ademais, este algoritmo possui complexidade de $O(n^2)$.

Ler $G = (N, M)$ e $D = [d_{ij}]$, onde d_{ij} é o custo da aresta (i, j)

$d_t[1] \leftarrow 0$

$rot[1] \leftarrow \infty$

Para $i \leftarrow 2$ ate n Faça

$d_t[i] \leftarrow \infty$

$rot[i] \leftarrow 0$

$A \leftarrow \{N\}$

$F \leftarrow \emptyset$

Enquanto $F \neq N$ Faça

$r \leftarrow j \in A$, tal que $d_t[j]$ é mínimo entre os elementos de A

$F \leftarrow F \cup \{r\}$

$A \leftarrow A \setminus \{r\}$

```

V ← V \ F

Para i ∈ V Faça
  p ← min{dt[i], (dt[r]+ dri)}
  Se p < dt[i]
    dt[i] ← p
    rot[i] ← r
  Fim_Se
Fim_Para
Fim_enquanto
  
```

O algoritmo de Dijkstra utiliza duas listas de vértices: a lista dos nós fechados (**F**) e a lista dos nós abertos (**A**). Um vértice está na lista **F** se o caminho mínimo da origem até aquele vértice já é conhecido, se não for, o vértice se encontra na lista **A**. O algoritmo possui dois vetores, um vetor denominado **dt** e outro denominado **rot**. Este algoritmo, o i -ésimo elemento do vetor **dt** guarda a distância calculada pelo algoritmo entre o vértice origem e o vértice **i**. Por sua vez, o i -ésimo elemento guarda o vértice anterior ao vértice **i** no caminho entre a origem e o vértice **i**, segundo a distância calculada (GOLDBARG; GOLDBARG, 2012).

No início é atribuído o valor zero para $dt[1]$, que corresponde à distância do vértice 1 a esse mesmo, e atribuído o infinito para $rot[1]$. Para os demais vértices do grafo é atribuído infinito em **dt** e zero para o **rot** dos demais vértices. A lista de nós abertos (**A**) recebe **N** e a lista fechada permanece vazia. O algoritmo itera até que todos os vértices tenham sido incluídos na lista **F**. A cada iteração, o algoritmo escolhe e remove um vértice de **A**.

O vértice **r** escolhido é aquele que tiver o menor valor de $dt[r]$. Depois de escolhido o vértice **r**, é incluído na lista **F** e todos os seus vizinhos que não estão em **F** são verificados. Estes vértices formam o conjunto **V**, descrito no algoritmo anterior. Para cada vértice **i** de **V** é verificado se a distância da origem até o qual é maior que a distância da origem até **r** mais o valor da aresta d_{ri} . Se sim, o caminho da origem até o vértice **i** passado pelo vértice **r** é menor do que o caminho anteriormente encontrado entre a origem e **i**. Neste caso, o valor $dt[i]$ é atualizado com $dt[r] + d_{ri}$ e $rot[i]$ é atualizado com **r**, indicando que o vértice anterior a **i** no caminho entre a origem e **i** é o vértice **r** (GOLDBARG; GOLDBARG, 2012).

Em uma primeira impressão o algoritmo de Dijkstra pode parecer complexo; todavia, é simples e eficiente. Analisaremos um exemplo prático: considere o grafo apresentado na Figura 5a; deseja-se obter o caminho mais curto entre os vértices **a** e **j**. A Figura 5b apresenta o grafo rotulado que caracteriza a iniciação da execução do algoritmo:

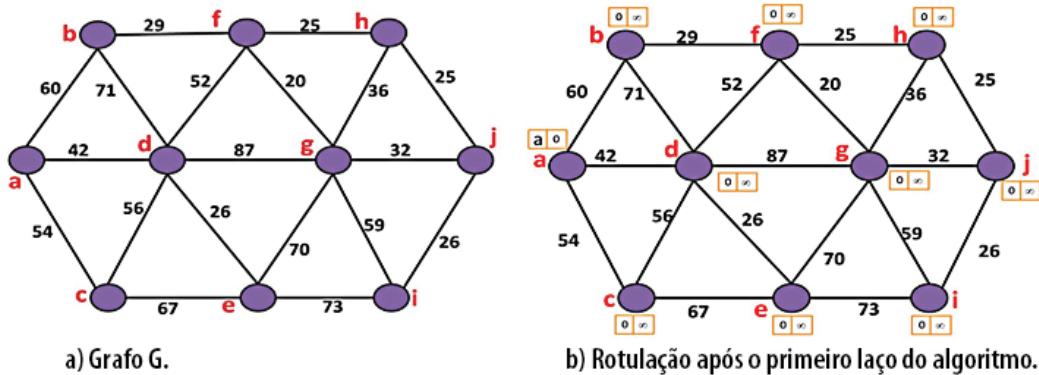


Figura 5 – Rotulação do algoritmo de Dijkstra

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Na Figura 5b é possível observar que para cada vértice w existe um rótulo com dois valores, sendo $\text{rot}[w]$ o primeiro valor, e o segundo correspondendo ao valor do $\text{dt}[w]$. Na inicialização, os vértices do grafo, com exceção de **a**, recebem a primeira posição do rótulo $\text{rot}[w] = 0$, e valor ∞ para a segunda posição, representando que $\text{dt}[w]$ ainda não foi calculado ou não existe. Dessa forma, o vértice **a** possui $\text{rot}[0] = 0$ e $\text{dt}[0] = a$.

O próximo passo é analisar os vértices adjacentes de **a** (Figura 6). Neste passo, os valores de dt e rot dos vértices **b**, **c** e **d** são revistos. É importante observar que estes vértices (**b**, **c**, **d**) são alcançados pelo vértice de origem (**a**). Os valores dos rótulos destes vértices são atualizados de modo a expressar a distância acumulada recém-calculada. Dessa forma, o caminho que chega até o vértice **b** recebe a marca de sua origem, $\text{rot}[b] = a$, e também o valor do $\text{dt}[b] = 60$, correspondendo à aresta (**a**, **b**). Os vértices **c** e **d** também são atualizados. Agora, tem-se: $F = \{a\}$ e $A = \{b, c, d, e, f, g, h, i, j\}$.

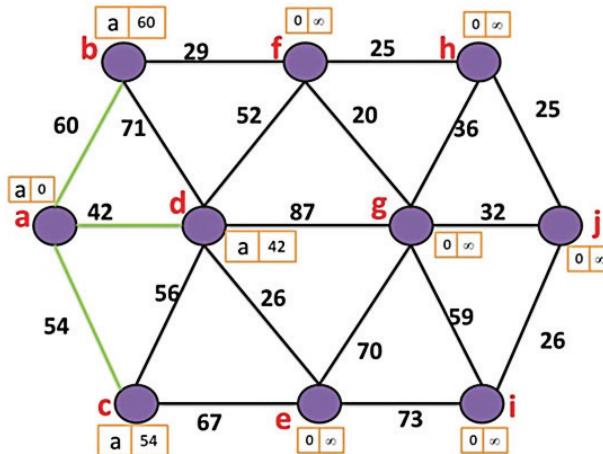


Figura 6 – Verificação do vértice a

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Na próxima iteração, um dos vértices de A é escolhido. A opção pelos vértices segue a forma gulosa do algoritmo de Dijkstra. Assim, o vértice a ser verificado em cada iteração é aquele que acumula o menor valor relativo à distância entre esse e o vértice de origem. Em nosso exemplo, o próximo vértice a ser verificado é o **d**.

Os vértices adjacentes a **d** são **b**, **c**, **e** e **f** – e estão em **A**. Os rótulos dos vértices **b** e **c** não são alterados porque a inclusão do vértice **d** no caminho da origem a cada um dos quais não produz um caminho menor do que o anteriormente existente. Agora, os vértices **e** e **f** têm os seus rótulos alterados porque existem caminhos com os custos de 68 e 92, respectivamente – a Figura 7 ilustra a verificação do vértice **d**:

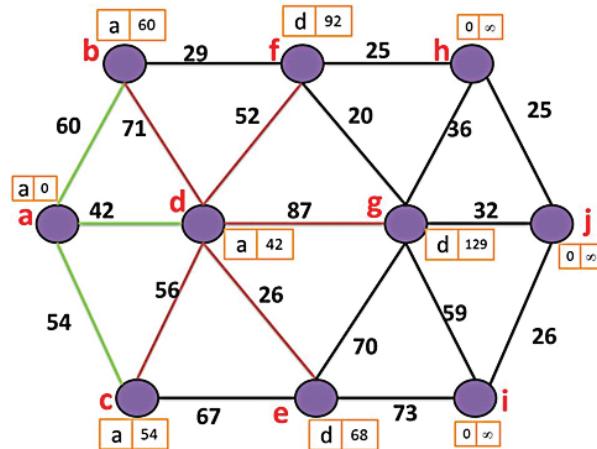


Figura 7 – Verificação do vértice **d**

Fonte: adaptada de Goldbarg e Goldbarg (2012)

No processamento do nosso grafo de exemplo, o próximo vértice a ser verificado é o **c**. Este vértice acumula 54 em seu rótulo. Neste momento, o único vértice adjacente a **c** que está em **A** é o **e**. Pela verificação do vértice **c**, a rotulação de **e** não é alterada – a Figura 8 ilustra a verificação do vértice **c**:

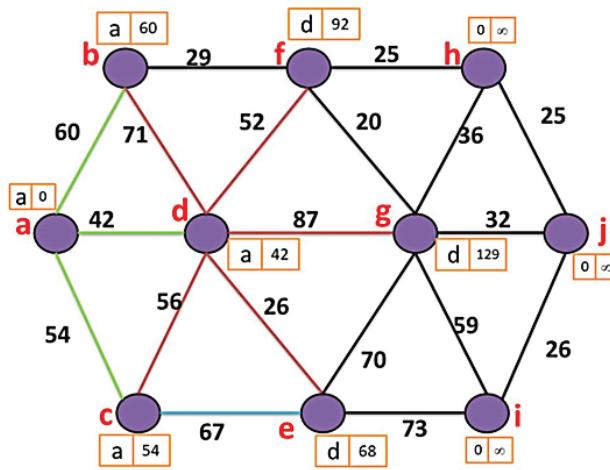
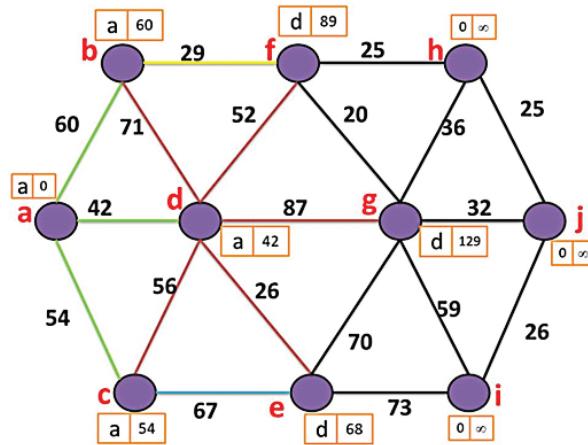


Figura 8 – Verificação do vértice **c**

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Na próxima iteração do algoritmo o vértice **b** é selecionado. O único vértice adjacente a **b** que não está em **F** é o vértice **f**. O rótulo de **f** é atualizado porque existe um caminho de **a** até **f** passando por **b** com valor menor – a verificação do vértice **b** é representada na Figura 9:

Figura 9 – Verificação do vértice **b**

Fonte: adaptada de Goldbarg e Goldbarg (2012)

A verificação continua nos demais vértices. Agora pelo vértice **e**:

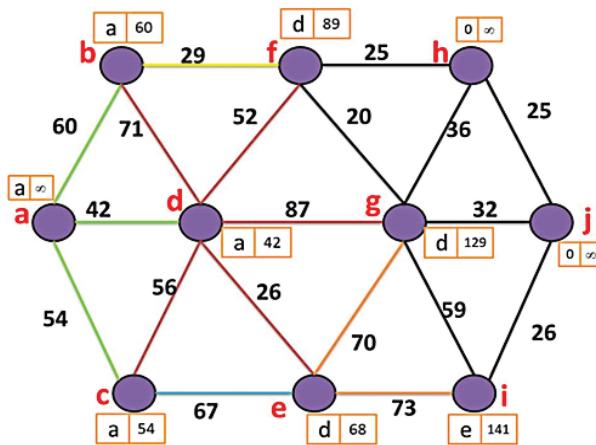
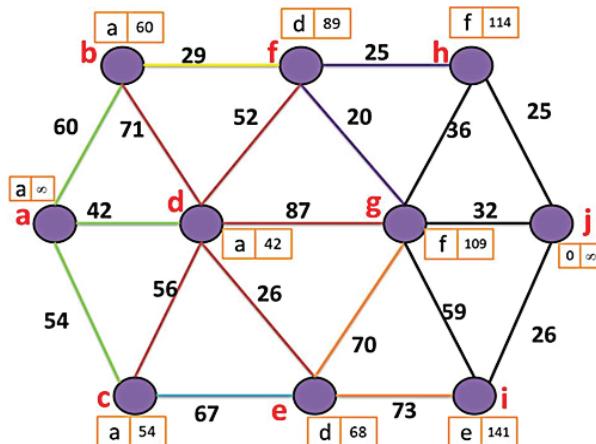


Figura 10 – Verificação do vértice “e”

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Verificação do vértice **f**:

Figura 11 – Verificação do vértice **f**

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Verificação do vértice g:

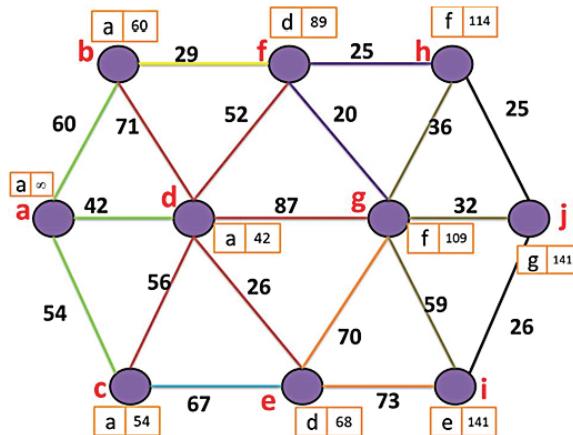


Figura 12 – Verificação do vértice g

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Verificação do vértice h:

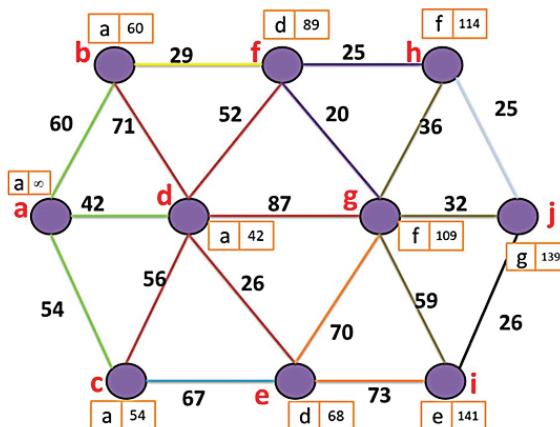


Figura 13 – Verificação do vértice h

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Verificação do vértice j:

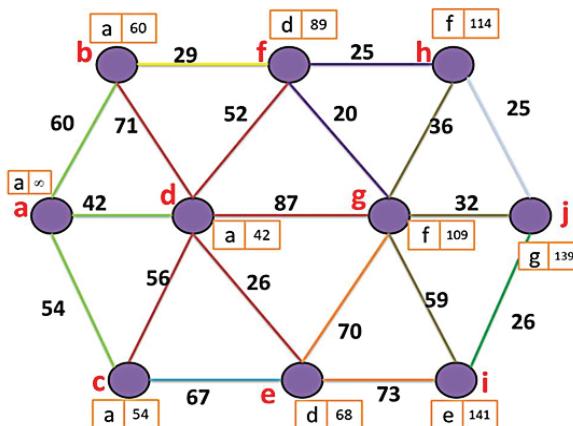


Figura 14 – Verificação do vértice j

Fonte: adaptada de Goldbarg e Goldbarg (2012)

O caminho mais curto de **a** até **j** é apresentado na Figura 15:

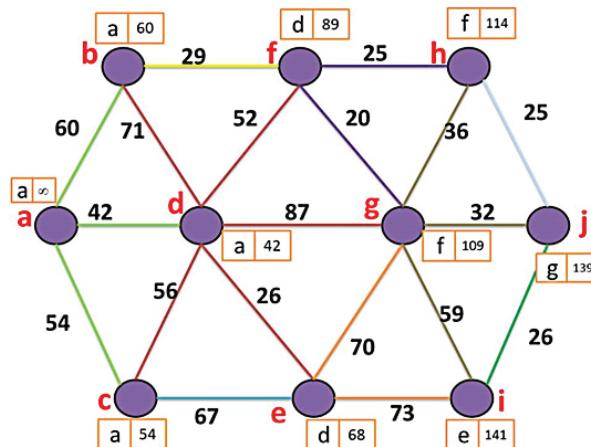


Figura 15 – Caminho mais curto de **a** até **j**

Fonte: adaptada de Goldbarg e Goldbarg (2012)

Algoritmo de Ford-Moore-Bellman

Assim como o algoritmo de Dijkstra, o algoritmo de Ford-Moore-Bellman obtém o menor caminho entre dois vértices **v** e **w** em um grafo **G**. Contudo, diferentemente do de Dijkstra, este algoritmo abre mão da possibilidade de fechar um vértice a cada iteração e se obriga a verificar todos os vértices até que melhorias não sejam mais possíveis. Utilizando essa estratégia, o algoritmo consegue calcular o caminho mais curto em grafos que contêm arestas negativas. A ideia principal desse algoritmo é que um caminho **s** para **j** com **k** + 1 arestas pode ser obtido de um caminho de **s** para **j** com **k** arestas. Em tal algoritmo, o critério de parada está baseado no fato de que, se em alguma iteração todos os rótulos dos vértices não foram alterados, não há melhorias nas próximas iterações (GOLDBARG; GOLDBARG, 2012).

Ler $G = (N, M)$ e $D = [d_{ij}]$

$dt[1] \leftarrow 0$

$rot[1] \leftarrow 0$

Para $i \leftarrow 2$ até n Faça

$dt[i] \leftarrow d_{1i}$

Se $\exists (1, i) \in M$

$rot[i] = 1$

senão

$rot[i] \leftarrow 0$

$dt[i] \leftarrow \infty$

Fim_Para

```

Para k ← 1 até n-1 Faça
    altera ← ‘Falso’
    Para i ← 2 até n Faça
        v ← ‘-- (1)
        Para j ∈ V Faça
            Se dt[i] > dt[j] + dji
                dt[i] ← dt[j] + dji
                rot[i] ← j
                altera ← ‘Verdadeiro’
            Fim_se
        Fim_Para
    Fim_Para
Fim_Para
  
```

Passeio em Grafos

Dado um grafo **G**, um passeio consiste de uma sequência finita alternada de vértices e arestas, que começa e termina por vértices tal que cada aresta é incidente ao vértice que a precede e ao que a sucede.

Grafos Hamiltonianos

De uma forma bem simples, podemos definir um grafo como hamiltoniano se neste existir um caminho que contém todos os vértices de **G**. Podemos compreender, também, que um ciclo hamiltoniano é aquele que contém todos os vértices do grafo exatamente uma vez, com exceção dos vértices inicial e final – a Figura 16 ilustra um grafo hamiltoniano:

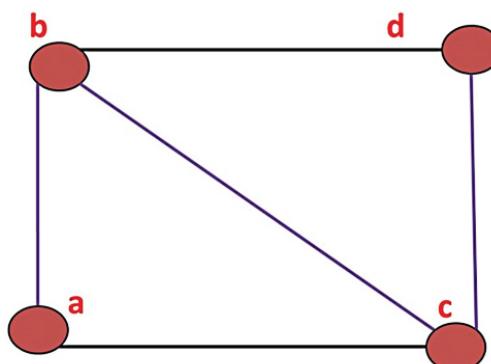


Figura 16 – Grafo hamiltoniano
 Fonte: elaborada pelo professor conteudista

Na literatura existe um problema bem conhecido e que é modelado por grafos hamiltonianos, trata-se do **problema do caixeleiro viajante**, no qual há um vendedor que necessita ir a diversas cidades; todavia, deseja achar um caminho que passará por todas essas localidades, mas que nunca passe duas vezes por uma mesma cidade.

Grafos Eulerianos

Um grafo é denominado euleriano se prover um ciclo que passe por todas as arestas de \mathbf{G} sem repetição. Ou seja, um ciclo euleriano é aquele que passa por todas as arestas do grafo uma única vez. A Figura 17 ilustra um grafo euleriano, pois trata-se de um ciclo que passa por todas as arestas apenas uma vez ($u_1, u_2, u_3, u_4, u_5, u_3, u_1, u_6, u_2, u_7, u_3, u_6, u_7, u_1$):

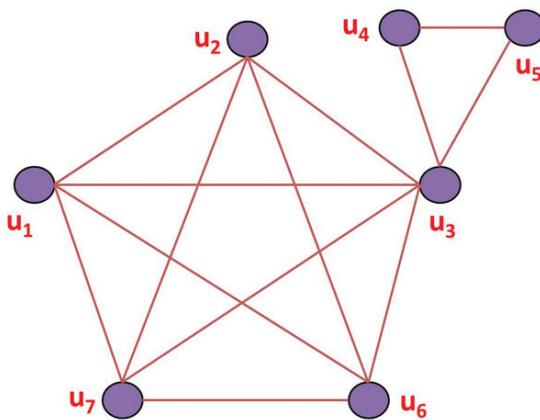


Figura 17 – Grafo euleriano

Fonte: elaborada pelo professor conteudista



Em Síntese

Caminhos em grafo correspondem a um dos principais conceitos utilizados para modelar diversos problemas. Um caminho pode ser definido como uma sequência de vértices tal que para cada vértice da sequência há uma aresta para o próximo vértice da mesma sequência.

Diversos problemas que são modelados com grafos necessitam achar o caminho mais curto entre um par de vértices; de modo que o caminho mais curto entre dois vértices v e w de um grafo \mathbf{G} não ponderado é aquele que acumula a menor quantidade de arestas entre v e w .

Existem diversos algoritmos para obter o caminho mais curto entre dois vértices em um grafo, por exemplo, os algoritmos de Dijkstra e de Ford-Moore-Bellman – ambos são eficientes e elegantes, provendo sempre o menor caminho entre um par de vértices.

Por fim, os grafos de tipo hamiltoniano e euleriano são aplicados na resolução de diversos problemas, de modo que um grafo hamiltoniano é aquele que possui um caminho que contém todos os vértices de G ; já um grafo euleriano contém um caminho que passa por todas as arestas, sem repetição.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Livros

Introdução à teoria dos grafos

CLÁUDIO, L. L. **Introdução à teoria dos grafos**. [S.l.]: Impar, 2016.

Fundamentos da teoria dos grafos para computação

NICOLETTI, A. M.; HRUSCHKA JÚNIOR, E. R. **Fundamentos da teoria dos grafos para computação**. São Carlos, SP: Edufscar, 2006.

Grafos e redes: teoria e algoritmos básicos

SIMÕES, J. M. S. **Grafos e redes: teoria e algoritmos básicos**. [S.l.]: Interciência, 2013.

Leitura

Matemática discreta: combinatória, teoria dos grafos e algoritmos

CARDOSO, M.; SZYMANSKI, J.; ROSTAMI, M. **Matemática discreta: combinatória, teoria dos grafos e algoritmos**. [S.l.]: Escolar, 2009.

<https://bit.ly/2K0N8LQ>

Referências

- BOAVENTURA NETTO, P. O. **Grafos**: teoria, modelos, algoritmos. 2. ed. São Paulo: Edgard Blucher, 2001.
- _____. JURKIEWICZ, S. **Grafos**: introdução e prática. 2. ed. [S.l.]: Blucher, 2017.
- CORMEN, T. *et al.* **Algoritmos – teoria e prática**. 2. ed. [S.l.]: Campus, 2002.
- FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. **Uma introdução sucinta à teoria dos grafos**. 2018. Disponível em: <<http://www.ime.usp.br/~pf/teoriadosgrafos>>. Acesso em: 24 nov. 2018.
- FURTADO, A. L. **Teoria dos grafos algoritmos**. Rio de Janeiro: Livros Técnicos e Científicos, 1973.
- GOLDBARG, M.; GOLDBARG, E. **Grafos – conceitos, algoritmos e aplicações**. [S.l.]: Campus, 2012.
- NICOLETTI, A. M.; HRUSCHKA JÚNIOR, E. R. **Fundamentos da teoria dos grafos para computação**. São Carlos, SP: Edufscar, 2006.
- SIMÕES, J. M. S. **Grafos e redes**: teoria e algoritmos básicos. [S.l.]: Interciência, 2013.
- SZWARCFITER, J. L. **Teoria computacional de grafos**. [S.l.]: Elsevier, 2018.



Cruzeiro do Sul
Educacional