# PROBLEMA DO CAMINHO MÍNIMO

JOÃO COLOMBINI E RODRIGO DOS SANTOS

## 1. Grafos e caminhos

Nessa primeira seção, serão introduzidos os conceitos básicos que serão usados nas seções seguintes. Nós começaremos a seção com a definição e alguns exemplos de grafos, continuaremos com a definição e exemplos de caminhos em grafos, e terminaremos com a definição e exemplos de grafos ponderados.

Para começar, vamos construir um conjunto que será usado na definição de grafos. Dado um conjunto $V$, lembre que o produto cartesiano de $V$ com ele mesmo é o conjunto definido por:

$$V \times V = \{(v, w) \mid v, w \in V\}.$$

Nesse conjunto $V \times V$, considere a relação de equivalência dada por:

$$(v, w) \sim (w, v) \quad \text{para todos } v, w \in V.$$

O conjunto de classes de equivalência $V \times V/\sim$ será denotado por $V \square V$ e os elementos de $V \square V$ serão denotados por $[v, w]$. Observe que, se $v \neq w$ em $V$, então $(v, w) \neq (w, v)$ em $V \times V$ (ou seja, a ordem das entradas dos pares de $V \times V$ é importante). No entanto, $[v, w] = [w, v]$ em $V \square V$ (ou seja, a ordem das entradas dos pares de $V \square V$ não importa).

**Definição 1.1** (de grafo)**.** Um *grafo* $G$ é um par $(V, E)$, onde:

(i) $V$ é um conjunto não-vazio,
(ii) $E$ é um conjunto disjunto de $V$ (ou seja, $V \cap E = \emptyset$).

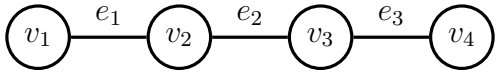
Além disso, no caso em que $E \neq \emptyset$, o par $(V, E)$ é munido de uma função $\phi : E \to V \square V$.

Os elementos do conjunto $V$ são chamados de *vértices de* $G$, os elementos de $E$ são chamados de *arestas de* $G$, e a função $\phi$ é chamada de *função de incidência*.

**Exemplo 1.2.** Um exemplo de um grafo é um par $(V, E)$, onde:

- o conjunto de vértices é $V = \{v_1, v_2, v_3, v_4\}$,
- o conjunto de arestas é $E = \{e_1, e_2, e_3\}$,
- a função de incidência é dada por $\phi(e_i) = [v_i, v_{i+1}]$.

Existem várias formas de representar esse grafo. Uma delas é a seguinte:




**Exemplo 1.3.** Existem vários exemplos de pares $(V, E)$ que não são grafos. Um deles seria o caso em que $V = E = \{v\}$, e outro seria o caso em que $V = \emptyset$ e $E = \{e\}$. Representações diagramáticas desses grafos são as seguintes:

*Date*: 29 de janeiro de 2024.


**Exemplo 1.4.** O menor grafo possível é aquele cujo par $(V, E)$ é dado por $V = \{v\}$ e $E = \emptyset$. Uma representação diagramática desse grafo é:


**Exemplo 1.5.** Depois do grafo dado no Exemplo 1.4, os menores grafos possíveis são:

(i) O grafo cujo par $(V, E)$ é dado por: $V = \{v_1, v_2\}$ e $E = \emptyset$.
(ii) O grafo cujo par $(V, E)$ é dado por: $V = \{v\}$, $E = \{e\}$.

Representações diagramáticas desses grafos são:


**Definição 1.6** (de caminho)**.** Dado um grafo $G = (V, E)$ com função de incidência $\phi$, um *caminho em* $G$ é uma sequência finita $v_0e_1v_1 \cdots e_nv_n$, onde:

(i) $v_0, \ldots, v_n \in V$ são vértices distintos,
(ii) $e_1, \ldots, e_n \in E$ são arestas distintas,
(iii) $\phi(e_i) = [v_i, v_{i-1}]$ para todo $i \in \{1, \ldots, n\}$.

Dado um caminho $v_0e_1v_1 \cdots e_nv_n$, o número inteiro $n \geq 0$ é chamado de *comprimento do caminho*.

**Exemplo 1.7.** Considere o grafo $G = (V, E)$ dado por:

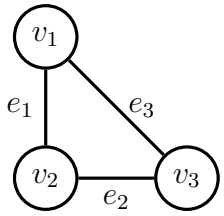
- $V = \{v\}$,
- $E = \{e\}$,
- função de incidência $\phi(e) = [v, v]$.

Uma representação diagramática desse grafo é:


Observe que o único caminho nesse grafo é de comprimento 0: $v$. Para mostrar que não existe nenhum caminho de comprimento $> 0$ nesse grafo, observe que o conjunto $V$ só contem um elemento, $v$. Assim, não existe nenhuma aresta $a \in E$ tal que $\phi(a) = [v, v_1]$ e $v \neq v_1$.

**Exemplo 1.8.** Considere o grafo $G = (V, E)$ dado por:

- $V = \{v_1, v_2, v_3\}$,
- $E = \{e_1, e_2, e_3\}$,




- $\phi(e_1) = [v_1, v_2]$, $\phi(e_2) = [v_2, v_3]$ e $\phi(e_3) = [v_1, v_3]$.

Uma representação diagramática desse grafo está contida na figura acima.

Observe que os caminhos de comprimento 0 neste grafo são: $v_1$, $v_2$ e $v_3$. Depois, os caminhos de comprimento 1 são: $v_1e_1v_2$, $v_2e_1v_1$, $v_2e_2v_3$, $v_3e_2v_2$, $v_3e_3v_1$, e $v_1e_3v_3$. Por fim, os caminhos de comprimento 2 são: $v_1e_1v_2e_2v_3$, $v_3e_2v_2e_1v_1$, $v_2e_2v_3e_3v_1$, $v_1e_3v_3e_2v_2$, $v_3e_3v_1e_1v_2$, e $v_2e_1v_1e_3v_3$.

Observe que não existem caminhos de comprimentos maiores. De fato, como os vértices de um caminho têm que ser todos distintos, se um grafo $G$ tem $n$ vértice, pelo *Principio da Casa dos Pombos*, todo caminho em $G$ tem comprimento $\leq n$. Nesse caso específico, todo caminho tem comprimento menor que 3.

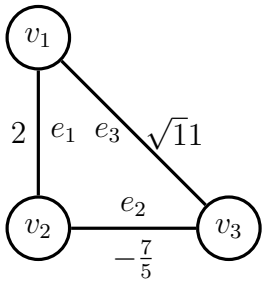
Observe ainda que $v_1e_1$ e $v_1e_1v_2e_2$ não são caminhos pois não terminam em um vértice. Além disso, $v_1e_1v_2e_1v_1$ não é um caminho pois a aresta $e_1$ e o vértice $v_1$ se repetem. Por fim, observe que $v_1e_1v_3$ não é um caminho pois $\phi(e_1) = [v_1, v_2]$ e $v_2 \neq v_3$.

**Definição 1.9** (de grafo ponderado)**.** Um grafo $G = (V, E)$ é dito *ponderado* quando $G$ é munido de uma função $w : E \to \mathbb{R}$, chamada de *peso*.

**Exemplo 1.10.** Considere o grafo do Exemplo 1.8 e a função $w : E \to \mathbb{R}$ dada por:

$$w(e_1) = 4, \quad w(e_2) = -\frac{7}{5} \quad \text{e} \quad w(e_3) = \sqrt{11}.$$

Esse grafo ponderado pode ser representado como:




Observe que, os pesos das arestas podem ser quaisquer números reais.

## 2. Problema do caminho mínimo

O *problema do caminho mínimo* é o seguinte: dado um grafo ponderado $G = (V, E, w)$ e dois vértices $v_a, v_b \in V$, encontrar um caminho $v_ae_1 \dots e_nv_b$ em $G$ tal que o valor da soma $w(e_1)+\cdots+w(e_n)$ é o menor possível. Vamos começar essa seção dando um exemplo concreto desse problema e, em seguida, explicar alguns algoritmos que podem ser usados para resolver esse problema em geral.

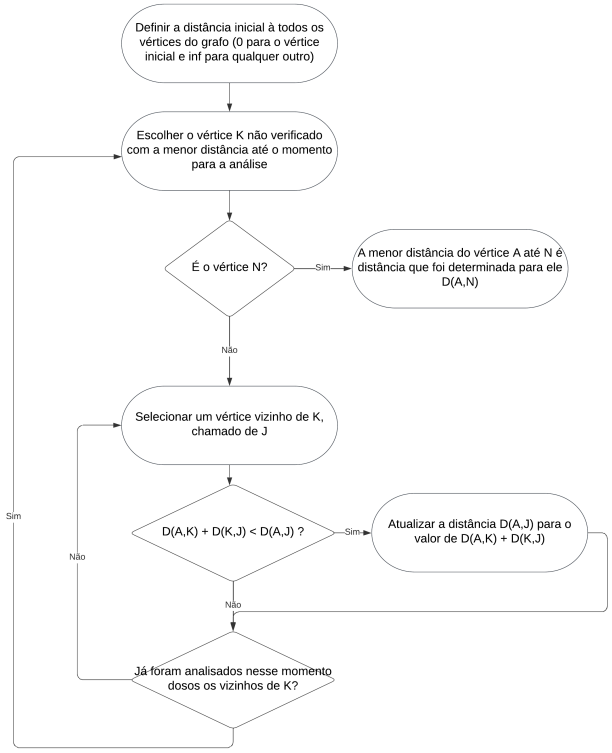



FIGURA 1. Fluxograma do Algoritmo de Dijkstra

**Exemplo 2.1.** Considere o grafo ponderado mostrado no Exemplo 1.10. Vamos procurar o caminho mínimo entre os vértices $v_1$ e $v_2$.

Lembre do Exemplo 1.8 que há apenas 2 caminhos que começam em $v_1$ e terminam em $v_2$: $v_1e_1v_2$ e $v_1e_3v_3e_2v_2$. No primeiro caminho, a soma dos valores de $w$ é $w(e_1) = 2$, e no segundo caminho, $w(e_3) + w(e_2) = \sqrt{11} - \frac{7}{5} \approx 1,92$. Como $1,92 < 2$, o caminho mínimo de $v_1$ a $v_2$ é $v_1e_3v_3e_2v_2$. Observe que, nesse caso, apesar do caminho $v_1e_1v_2$ ter menor comprimento, a soma dos pesos das arestas do caminho $v_1e_3v_3e_2v_2$ é menor. Por isso, $v_1e_3v_3e_2v_2$ é o caminho mínimo de $v_1$ a $v_2$.

No exemplo acima, nós listamos todos os caminhos que começam no vértice $v_1$ e terminam no vértice $v_2$, e comparamos as somas dos pesos das suas arestas. Esse método pode ser generalizado em um algoritmo, chamado de *Busca Exaustiva*, que sempre retorna o caminho mínimo. No entanto, quando a quantidade de caminhos no grafo cresce, essa busca exaustiva demorará muito tempo para retornar o caminho mínimo.

O *Algoritmo de Dijkstra* [3] é um algoritmo que também retorna o caminho mínimo entre dois vértices de um grafo, mas é mais rápido que a busca exaustiva. Para que o algoritmo de Dijkstra funcione, é necessário que o grafo ponderado satisfaça duas condições:

(i) existe pelo menos um caminho entre quaisquer dois vértices do grafo;
(ii) o peso de toda aresta do grafo é não-negativa, ou seja, $w(e) \geq 0$ para todo $e \in E$.

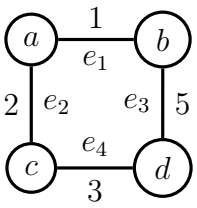
Um pseudocódigo do Algoritmo de Dijkstra está contido no Algoritmo 1, um fluxograma mostrando o seu funcionamento está contido na Figura 1, e uma implementação dele em *Python* está contida em [2].

**Algoritmo 1:** Pseudocódigo do Algoritmo de Dijkstra

**Dados:** Grafo $G = (V, E)$ e vértice $v_0 \in V$
`Inicialize variáveis para guardar as distâncias de cada vértice em relação ao vértice inicial` $v_0$`:`
$\text{dist}(v_0, v_0) \leftarrow 0$ e **para** $v \in V \setminus \{v_0\}$ **faça**
  $\text{dist}(v_0, v) \leftarrow \infty$;
**fim**
`Inicialize uma variável que guarda o vértice que está sendo analisado:`
vértice atual $\leftarrow v_0$;
`Inicialize um conjunto que guarda os vértices que ainda não foram analisados:`
vértices não-visitados $\leftarrow V \setminus \{v_0\}$;
`Inicialize uma lista ordenada chamada fila de prioridade:`
**para** *todos os vértices* $w$ *não-visitados e adjacentes a* $v_0$ **faça**
  Atualize $\text{dist}(v_0, w)$ com o peso da aresta que liga $v_0$ a $w$;
  Inclua o par $(w, \text{dist}(v_0, w))$ na fila de prioridade;
**fim**
**enquanto** *fila de prioridade é não-vazia* **faça**
  `Atualize o vértice atual:`
  Tome o vértice $w$ da lista de prioridade cuja dist(vértice atual, $w$) é mínima;
  vértice atual $\leftarrow w$;
  `Atualize o conjunto de vértices não-visitados:`
  Retire $w$ do conjunto de vértices não-visitados;
  `Atualize a lista de prioridade:`
  Esvazie a lista de prioridade;
  **para** *todos os vértices* $v$ *não-visitados e adjacentes a* $w$ **faça**
    **se** $d(v_0, w)$+*(peso da aresta que liga* $w$ *a* $v$*)* $< d(v_0, v)$ **então**
      $d(v_0, v) \leftarrow d(v_0, w)$+(peso da aresta que liga $w$ a $v$);
    **fim**
    Adicione $v$ à lista de prioridade;
  **fim**
**fim**

Vamos fazer alguns exemplos concretos do funcionamento desse algoritmo.

**Exemplo 2.2.** Considere o grafo ponderado representado por



Tomando o vértice inicial como sendo $a$, vamos usar o Algoritmo de Dijkstra para calcular os caminhos mínimos entre $a$ e os outros vértices desse grafo.

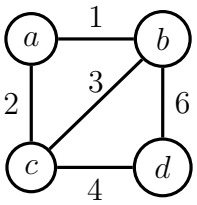
Denote por $D(a, v)$ a peso total entre o vértice $a$ e um vértice $v \in V = \{a, b, c, d\}$. Inicialmente, vamos definir $D(a, a) = 0$ e $D(a, b) = D(a, c) = D(a, d) = \infty$. Depois, observe que os vértices adjacentes ao vértice $a$ são $b$ e $c$. Assim, redefinimos $D(a, b) = w(e_1) = 1$ e $D(a, c) = w(e_2) = 2$. Nesse estágio, todas as adjacências ao vértice $a$ já foram todos analisadas, e assim ele não será mais considerado na busca.

Como $D(a, b) = 1 < 2 = D(a, c)$, então o próximo vértice a ser analisado será o vértice $b$. Observe que os vértices adjacentes ao vértice $b$ são $a$ e $d$. Como $D(a, b) = 1$ e $D(b, d) = w(e_3) = 5$, nós vamos atualizar $D(a, d) = 6$. Nesse estágio, todas as adjacências ao vértice $b$ também já foram analisadas, e assim ele também não será mais considerado na busca.

Para continuar, vamos analisar as adjacências do vértice $c$. Observe que os vértices adjacentes a $c$ são $a$ e $d$. Como $D(a, c) = 2$, $D(c, d) = w(e_4) = 3$ e $2 + 3 < D(a, d)$, então vamos atualizar $D(a, d) = 5$. Observe que, nesse estágio todos os vértices e suas adjacências foram analisadas. Assim, o algoritmo retorna:

$$D(a, a) = 0, \quad D(a, b) = 1, \quad D(a, c) = 2 \quad \text{e} \quad D(a, d) = 5.$$

**Exemplo 2.3.** Considere o grafo ponderado representado por




Tomando $a$ como o vértice inicial, vamos usar o Algoritmo de Dijkstra para calcular os caminhos mínimos entre $a$ e os outros vértices de $G$. Primeiro, definimos $D(a, a) = 0$ e $D(a, b) = D(a, c) = D(a, d) = \infty$. Depois, observe que os vértices adjacentes ao vértice $a$ são $b$ e $c$. Assim, redefinimos $D(a, b) = 1$ e $D(a, c) = 2$. Como $D(a, b) < D(a, c)$, o próximo vértice que nós analisaremos será $b$. Observe que, além de $a$, os vértices adjacentes ao vértice $b$ são $c$ e $d$. Como $D(b, c) = 3$, mantemos $D(a, c) = 2$, mas redefinimos $D(a, d) = D(a, b) + D(b, d) = 7$. Agora, como $D(b, c) < D(b, d)$, então o próximo vértice que nós analisaremos será $c$. Observe que, além de $a$ e $b$, o vértice $d$ também é adjacente ao vértice $c$. Além disso, como $D(a, c) + D(c, d) = 6 < 7 = D(a, d)$, então redefinimos $D(a, d) = 6$. Isso termina o Algoritmo de Dijkstra, com o seguinte resultado:

$$D(a, a) = 0, \quad D(a, b) = 1, \quad D(a, c) = 2 \quad \text{e} \quad D(a, d) = 6.$$

Os exemplos acima já motivam algumas perguntas naturais. Vamos analisá-las a seguir.

**Pergunta 1.** O Algoritmo de Dijkstra sempre retorna o caminho mínimo entre o vértice inicial e os outros vértices do grafo?

Para mostrar que isso sempre acontece, vamos usar indução na quantidade de vértices cujas adjacências foram todas analisadas.

A base da indução é o caso que só tem um ponto cujas adjacências foram todas analisadas, o vértice inicial. Nesse caso, é óbvio que o caminho mínimo tem comprimento 0 e só contem o vértice inicial.

A hipótese de indução é que o conjunto de vértices cujas adjacências foram todas analisadas contém $n$ elementos. Vamos denotá-los por $v_0, v_1, \ldots, v_{n-1}$, na ordem em que eles foram inseridos. (Em particular, $v_0$ é o vértice inicial.) Para aumentar esse conjunto para $n+1$ elementos, considere o próximo vértice segundo o Algoritmo de Dijkstra. Esse vértice será denotado por $v_n$ e é o vértice adjacente a $v_{n-1}$ cuja aresta tem o menor peso. Portanto como sempre é conectado ao menor peso entre todos os vértices $v_0, v_1, \ldots, v_{n-1}$, temos que ao percorrer todo o grafo será apresentado o menor caminho vértice a vértice.

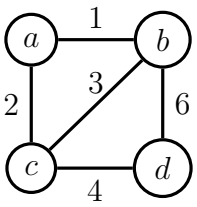
**Pergunta 2.** O Algoritmo de Dijkstra é mais eficiente do que a Busca Exaustiva?

Observe que o Algoritmo de Dijkstra é de fato mais eficiente do que a Busca Exaustiva, porque, como mostrado acima, o Algoritmo de Dijkstra separa o conjunto de vértices do grafo em dois subconjuntos: um subconjunto contendo os vértices cujas adjacências foram todas analisadas, e seu complementar. Essa separação é usada na sequência do algoritmo, quando os caminhos que passam pelos vértices do primeiro conjunto não são mais considerados. Isso faz com que a complexidade do Algoritmo de Dijkstra em um grafo $(V, E)$ seja de $O(|E| + |V| \log |V|)$ (quando utilizamos uma lista de prioridades), enquanto a complexidade da Busca Exaustiva é $O(|V|^2)$.

**Pergunta 3.** Existe alguma diferença entre o Algoritmo de Dijkstra e uma heurística gulosa?

Existem diversas diferenças, mas a principal delas é que, enquanto o Algoritmo de Dijkstra sempre retorna o caminho mínimo, uma heurística gulosa pode falhar. Vamos ver um exemplo concreto.

Lembre do Exemplo 2.3 que o menor caminho entre os vértices $a$ e $d$ tem peso total igual a 6 no grafo:




No entanto, usando uma heurística gulosa que prioriza as arestas com menor peso, nós obtemos que esse peso total seria igual a 8. De fato, começando no vértice $a$, a aresta com menor peso que incide em $a$ tem peso 1 e incide em $b$. Assim, no próximo passo, nós consideramos todas as arestas que incidem em $b$ e não incidem em $a$. Dessas, a aresta de menor peso tem peso 3 e também incide em $c$. Por fim, nós consideramos as arestas que incidem em $c$ e não incidem em $a$ e em $b$. A única dessas arestas tem peso 4 e incide também em $d$. Isso termina a heurística e retorna o peso total de $1 + 3 + 4 = 8$.

Observe que essa heurística não retorna o caminho mínimo, porque ela não continua analisando a outras adjacências dos vértices visitados, como faz o Algoritmo de Dijkstra.

Em particular, essa heurística gulosa não considera a possibilidade do menor caminho entre $a$ e $d$ ir diretamente de $b$ para $d$.

Para concluir essa seção, vamos citar outros algoritmos importantes para a resolução do Problema do Caminho Mínimo. O Algoritmo de Bellman-Ford [1, 4] é parecido com o Algoritmo de Dijkstra, mas permite que os pesos das arestas sejam negativos e não é tão rápido. O Algoritmo A* (*A-estrela*) [5] é baseado no Algoritmo de Dijkstra, mas utiliza uma heurística para obter os caminhos mínimos de forma mais rápida.

## Referências

[1] Bellman, R.. On a routing problem. Quart. Appl. Math. 16 (1958), 87–90.
[2] Colombini, J. V. e dos Santos, R.. Joao, Rodrigo - Problema do Caminho Minimo. Repositório do Github (2024).
[3] Dijkstra, E. W. A note on two problems in connexion with graphs. Numer. Math. 1 (1959), 269–271.
[4] Ford, L. R.. Network Flow Theory. Santa Monica, CA: RAND Corporation, 1956.
[5] Hart, P. E; Nilsson, N. J.; Raphael B.. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107 (1968).