# Projeto e Análise de Algoritmos

Flávio L. C. de Moura*

12 de abril de 2022

## 1 Caminhos Mínimos

Nesta seção veremos como resolver o problema do caminho mínimo em grafos com pesos[1]. Agora cada aresta do grafo está associada a um número real, de forma que o peso de um caminho $p$ em $G$ é a soma dos pesos das arestas que compõem o caminho $p$. Formalmente, temos a seguinte definição:

**Definição 1.1.** *Considere um grafo $G = (V, E)$ com função peso $w : E \to \mathbb{R}$. O peso $w(p)$ de um caminho $p = \langle v_0, v_1, \ldots, v_k \rangle$ é a soma dos pesos das arestas que formam $p$:*

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Os grafos sem peso podem ser vistos como um caso particular da definição acima. De fato, um grafo sem pesos $G = (V, E)$ pode ser visto como um grafo com pesos onde $w(e) = 1, \forall e \in E$, e apesar de termos utilizado o algoritmo BFS para computar os caminhos de comprimento mínimo (menor número de arestas) em grafos sem peso, ele não computa corretamente os caminhos mínimos em grafos com pesos (por que?). Assim, o contexto adequado para estudarmos o problema dos caminhos mínimos é com digrafos com pesos. Apesar disto, grafos com pesos também poderão ser considerados, bastando para isto trocarmos cada aresta não dirigidas $(u, v)$ com peso $w(u, v)$ pelas arestas dirigidas $(u, v)$ e $(v, u)$, ambas com peso $w(u, v)$. Definimos o peso do caminho mínimo de $u$ para $v$ no digrafo $G$, denotado por $\delta(u, v)$, como a seguir:

**Definição 1.2.** *Sejam $G = (V, E)$ um digrafo com função peso $w : E \to \mathbb{R}$, e $u, v \in V$. O* caminho mínimo *do vértice $u$ para o vértice $v$ é definido como sendo qualquer caminho $p$ com peso $w(p) = \delta(u, v)$.*

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\}, & \text{se existe um caminho de } u \text{ para } v; \\ \infty, & \text{caso contrário.} \end{cases}$$

Estudaremos os caminhos mínimos a partir de um dado vértice (uma fonte), isto é, dados um digrafo $G = (V, E)$ e um vértice $s \in V$, queremos encontrar o caminho mínimo a partir de $s$ para cada vértice $v \in V$ de $G$.

### 1.1 Subestrutura ótima do problema do caminho mínimo

Algoritmos para caminhos mínimos normalmente se baseiam na propriedade de que um caminho mínimo entre dois vértices são formados a partir de outros caminhos mínimos:

**Lema 1.3.** *Dado um digrafo $G = (V, E)$ com função peso $w : E \to \mathbb{R}$, seja $p = \langle v_0, v_1, \ldots, v_k \rangle$ um caminho mínimo do vértice $v_0$ para o vértice $v_k$, e para todo $i$ e $j$ tais que $0 \leq i \leq j \leq k$, seja $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ o subcaminho de $p$ do vértice $v_i$ para o vértice $v_j$. Então $p_{ij}$ é um caminho mínimo de $v_i$ para $v_j$.*

*Demonstração.* Prova por contradição. □

*flavio@flaviomoura.info

Se um digrafo possui um ciclo com peso negativo então o caminho mínimo entre dois vértice que contenham este ciclo não está bem definido. De fato, sempre podemos diminuir o peso do caminho dando um nova volta pelo ciclo de peso negativo. Assim, se existe um ciclo de peso negativo entre os vértice $s$ e $v$, definimos $\delta(s, v) = -\infty$.

Os caminhos mínimos a partir da fonte $s$ para qualquer vértice alcançável a partir de $s$ no digrafo $G$ serão representados pela árvore de caminho mínimo $G' = (V', E')$, onde $V' \subseteq V$ e $E' \subseteq E$ tal que:

1. $V'$ é o conjunto de vértices alcançáveis a partir de $s$ em $G$;
2. $G'$ é uma árvore com raiz $s$, e;
3. Para todo $v \in V'$, o único caminho simples de $s$ para $v$ em $G'$ é um caminho mínimo de $s$ para $v$ em $G$.

Utilizaremos o subgrafo predecessor $G_\pi = (V_\pi, E_\pi)$, onde

- $V_\pi = \{v \in V : v.\pi \neq NIL\} \cup \{s\}$ e;
- $E_\pi = \{(v.\pi, v) \in E : v \in V_\pi - \{s\}\}$, e mostraremos que $G_\pi$ coincide com a árvore de caminho mínimo de $G$.

É importante observar que caminhos mínimos não são necessariamente únicos, assim como as árvores de caminho mínimo. Os algoritmos que veremos a seguir utilizam a técnica conhecida como *relaxamento de arestas*. Para cada vértice $v \in V$, manteremos um atributo $v.d$, que é uma cota superior para o peso do caminho mínimo da fonte $s$ para $v$. O processo de relaxar uma aresta $(u, v)$ consiste em testar se é possível diminuir o peso do caminho para $v$ encontrado até o momento via o vértice $u$; se for possível, atualizamos $v.d$ e $v.\pi$. O código a seguir faz o relaxamento da aresta $(u, v)$ em tempo constante:

**Algorithm 1:** Relax$(u, v, w)$

```
1 if v.d > u.d + w(u, v) then
2 |   v.d = u.d + w(u, v);
3 |   v.π = u;
4 end
```

O lema a seguir é conhecido como *propriedade da convergência*:

**Lema 1.4.** *Sejam $G = (V, E)$ um digrafo com função peso $w$, $s \in V$, e $s \rightsquigarrow u \to v$ um caminho mínimo para os vértices $u$ e $v$. Suponha que $G$ tenha sido inicializado com Initialize-Single-Source:*

**Algorithm 2:** Initialize-Single-Source$(G, s)$

```
1 for each vertex v ∈ G.V do
2 |   v.d = ∞;
3 |   v.π = NIL;
4 end
5 s.d = 0;
```

*e então considere uma sequência de passos de relaxamento que incluem a chamada Relax$(u, v, w)$. Se $u.v = \delta(s, u)$ em qualquer momento antes desta chamada então $v.d = \delta(s, v)$ em qualquer momento após esta chamada.*

O lema a seguir é conhecido como *propriedade do relaxamento do caminho*:

**Lema 1.5.** *Se $p = \langle v_0, v_1, \ldots, v_k \rangle$ é um caminho mínimo de $s = v_0$ para $v_k$, e se relaxamos as arestas de $p$ na ordem $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, então $v_k.d = \delta(s, v_k)$.*

*Demonstração.* Indução no comprimento $k$ do caminho $p$. □

**Exercício 1.6.** *Prove que todo caminho mínimo é simples.*

## 1.2 O algoritmo de Dijkstra

Edsger W. Dijkstra nasceu em 1930 em Roterdã (Holanda). Filho de cientistas, seu pai era químico, e sua mãe, matemática. Estudou Física Teórica na Universidade de Leiden, e em 1952 começou a trabalhar no Centro de Matemática de Amsterdã, onde progressivamente foi se envolvendo com Computação. Em 1956 foi convidado para demostrar o poder do computador ARMAC, que ocupava uma sala inteira do Centro de Matemática, e começou a pensar no problema de determinar o menor caminho entre duas cidades em um mapa. Conta-se que na manhã de um domingo ensolarado enquanto tomava um café encontrou a solução do problema: a ideia básica é ir construindo um conjunto de cidades, digamos $X$, a partir da origem $s$ até atingirmos o destino $u$. Em qualquer momento da execução do algoritmo é possível saber a distância mínima de $s$ para qualquer cidade em $X$, que inicialmente contém apenas $s$. Em cada passo subsequente é possível encontrar uma cidade fora do conjunto $X$, digamos $v$, com a propriedade que a distância de $s$ para $v$ é menor do que a distância de $s$ para qualquer outra cidade fora do conjunto $X$. Como a distância entre duas cidades quaisquer é sempre não-negativa, temos que $v$ deve estar ligada diretamente com alguma cidade em $X$, digamos $w$. Assim, a distância mínima de $s$ para $v$ é dada pela distância mínima de $s$ para $w$ adicionada da distância de $w$ para $v$. Neste momento, adicionamos $v$ ao conjunto $X$, e repetimos o processo que vai parar quando $u$ for adicionado ao conjunto $X$.

Seja $G = (V, E)$ um digrafo com função peso $w$, e tal que o peso de cada aresta seja não-negativo. Ou seja, assumimos que $w(u, v) \geq 0$, para toda aresta $(u, v) \in E$.

**Algorithm 3:** Dijkstra$(G, w, s)$

```
1  Initialize-Single-Source(G, s);
2  S = ∅;
3  Q = G.V;
4  while Q ≠ ∅ do
5  |   u = Extract-Min(Q);
6  |   S = S ∪ {u};
7  |   for each vertex v ∈ G.Adj[u] do
8  |   |   Relax(u, v, w)
9  |   end
10 end
```

Observe que Dijkstra é um algoritmo guloso. Sabemos que a estratégia gulosa nem sempre resulta em uma solução ótima, mas o teorema a seguir mostra que Dijkstra$(G, w, s)$ computa corretamente os caminhos mínimos a partir do vértice $s$:

**Teorema 1.7.** *O algoritmo de Dijkstra, ao ser executado em um digrafo $G = (V, E)$ com função peso não-negativa $w$ e fonte $s$, termina com $u.d = \delta(s, u)$ para todo $u \in V$.*

## 1.3 O algoritmo de Bellman-Ford

Resolve o problema do caminho mínimo para uma fonte (a partir de um vértice dado) em digrafos quaisquer, i.e. inclusive em digrafos que tenham arestas com peso negativo, e produz a resposta correta desde que nenhum ciclo negativo seja alcançável a partir da fonte. Adicionalmente, na existência de

ciclos negativos, o algoritmo pode detectá-los e informar sua existência.

**Algorithm 4:** Bellman-Ford$(G, w, s)$

```
1  Initialize-Single-Source(G, s);
2  for i = 1 to |G.V| − 1 do
3  |   for each edge (u, v) ∈ G.E do
4  |   |   Relax(u, v, w);
5  |   end
6  end
7  for each edge (u, v) ∈ G.E do
8  |   if v.d > u.d + w(u, v) then
9  |   |   return false
10 |   end
11 end
12 return true
```

**Lema 1.8.** *Seja $G = (V, E)$ um digrafo com função peso $w$, e $s \in V$. Assuma que $G$ não possui ciclos negativos alcançáveis a partir de $s$. Então, após as $|V| - 1$ iterações do* ***for*** *das linhas 2-6 de Bellman-Ford, temos $v.d = \delta(s, v)$, para todos os vértices $v \in V$ alcançáveis a partir de $s$.*

**Teorema 1.9.** *Considere a execução de Bellman-Ford em um digrafo $G = (V, E)$ com função peso $w$, e $s \in V$. Se $G$ não possui ciclos negativos alcançáveis a partir de $s$ então o algoritmo retorna true, e temos $v.d = \delta(s, v)$ para todos os vértices $v \in V$, e o subgrafo predecessor é a árvore de caminhos mínimos com raiz $s$. Se $G$ possui ciclos negativos alcançáveis a partir de $s$ então o algoritmo retorna false.*

# Referências


[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.