Algoritmos e Teoria dos Grafos

Tópico 29: Distâncias entre Todos os Pares de Vértices

Renato Carmo
André Guedes
Murilo Silva

Departamento de Informática da UFPR

2025/1

(i, j, k)–caminhos e k–distâncias

(i, j, k)–caminhos e k–distâncias

- Grafo direcionado com pesos, (G, w), e uma ordenação de $V(G)$, $(v_1, \ldots, v_n)$.

(i, j, k)–caminhos e k–distâncias

- Grafo direcionado com pesos, (G, w), e uma ordenação de $V(G)$, $(v_1, \ldots, v_n)$.

- (i, j, k)–**caminho**: caminho de v_i a v_j em G cujos vértices internos estão em $\{v_1, \ldots, v_k\}$.

(i, j, k)–caminhos e k–distâncias

- Grafo direcionado com pesos, (G, w), e uma ordenação de $V(G)$, $(v_1, \ldots, v_n)$.
- (i, j, k)–**caminho**: caminho de v_i a v_j em G cujos vértices internos estão em $\{v_1, \ldots, v_k\}$.
- k–**distância** de v_i a v_j: peso de um (i, j, k)–caminho de peso mínimo $(d_{G,w}(i, j, k))$.

(i, j, k)–caminhos e k–distâncias

(i, j, k)–caminhos e k–distâncias

Observe que

1. se P é um (i, j, k)–caminho então P também é um $(i, j, k+1)$–caminho.

(i, j, k)–caminhos e k–distâncias

Observe que

1. se P é um (i, j, k)–caminho então P também é um $(i, j, k+1)$–caminho.

2. a 0–distância de v_i a v_j é

$$d_{G,w}(i, j, 0) \begin{cases} 0, & \text{se } i = j, \\ w(v_i, v_j), & \text{se } i \neq j \text{ e } (v_i, v_j) \in A(G), \\ \infty, & \text{se se } i \neq j \text{ e } (v_i, v_j) \notin A(G). \end{cases}$$

(i, j, k)–caminhos e k–distâncias

Observe que

1. se P é um (i, j, k)–caminho então P também é um $(i, j, k+1)$–caminho.

2. a 0–distância de v_i a v_j é

$$d_{G,w}(i, j, 0) \begin{cases} 0, & \text{se } i = j, \\ w(v_i, v_j), & \text{se } i \neq j \text{ e } (v_i, v_j) \in A(G), \\ \infty, & \text{se se } i \neq j \text{ e } (v_i, v_j) \notin A(G). \end{cases}$$

3. a n–distância é a distância em G, isto é,
$d_{G,w}(i, j, n) = d_{G,w}(v_i, v_j)$, para todo $i, j \in [1..n]$.

Teorema 114

Seja (G, w) um grafo direcionado com pesos nas arestas, sendo $V(G) = \{v_1, \ldots, v_n\}$. Para todo $i, j \in [1..n]$ e todo $k \in [0..n]$,

$$d_{G,w}(i,j,k) = \begin{cases} 0, & \text{se } i = j, \\ w(v_i, v_j), & \text{se } i \neq j \text{ e } k = 0 \text{ e } (v_i, v_j) \in A(G), \\ \infty, & \text{se } i \neq j \text{ e } k = 0 \text{ e } (v_i, v_j) \notin A(G), \\ \min\{d_{G,w}(i,j,k-1), & \\ d_{G,w}(i,k,k-1) + d_{G,w}(k,j,k-1)\}, & \text{caso contrário.} \end{cases}$$

Teorema 114

Seja (G, w) um grafo direcionado com pesos nas arestas, sendo $V(G) = \{v_1, \ldots, v_n\}$. Para todo $i, j \in [1..n]$ e todo $k \in [0..n]$,

$$d_{G,w}(i,j,k) = \begin{cases} 0, & \text{se } i = j, \\ w(v_i, v_j), & \text{se } i \neq j \text{ e } k = 0 \text{ e } (v_i, v_j) \in A(G), \\ \infty, & \text{se } i \neq j \text{ e } k = 0 \text{ e } (v_i, v_j) \notin A(G), \\ \min\{d_{G,w}(i,j,k-1), & \\ d_{G,w}(i,k,k-1) + d_{G,w}(k,j,k-1)\}, & \text{caso contrário.} \end{cases}$$

Demonstração.

Exercício 135. □

Algoritmo Recursivo

Distancias$_R(G, w)$

Para $i \leftarrow 1$ **até** $|V(G)|$
    Para $j \leftarrow 1$ **até** $|V(G)|$
        $d[i, j] \leftarrow$ Distancia$_R(G, w, i, j, |V(G)|)$
Devolva d

Distancia$_R(G, w, i, j, k)$

Se $i = j$
    Devolva 0
Se $k = 0$
    Se $(v_i, v_j) \in A(G)$
        Devolva $w(v_i, v_j)$
    Devolva ∞
$D \leftarrow$ Distancia$_R(G, w, i, j, k - 1)$
$D' \leftarrow$ Distancia$_R(G, w, i, k, k - 1)$ + Distancia$_R(G, w, k, j, k - 1)$
Se $D' < D$
    Devolva D'
Devolva D

Algoritmo Recursivo

Distancias$_R(G, w)$

Para $i \leftarrow 1$ **até** $|V(G)|$
    Para $j \leftarrow 1$ **até** $|V(G)|$
        $d[i, j] \leftarrow$ Distancia$_R(G, w, i, j, |V(G)|)$
Devolva d

Distancia$_R(G, w, i, j, k)$

Se $i = j$
    Devolva 0
Se $k = 0$
    Se $(v_i, v_j) \in A(G)$
        Devolva $w(v_i, v_j)$
    Devolva ∞
$D \leftarrow$ Distancia$_R(G, w, i, j, k - 1)$
$D' \leftarrow$ Distancia$_R(G, w, i, k, k - 1)$ + Distancia$_R(G, w, k, j, k - 1)$
Se $D' < D$
    Devolva D'
Devolva D

- Repetições de subproblemas.

Algoritmo Recursivo

Distancias$_R(G, w)$

Para $i \leftarrow 1$ **até** $|V(G)|$
    Para $j \leftarrow 1$ **até** $|V(G)|$
        $d[i, j] \leftarrow$ Distancia$_R(G, w, i, j, |V(G)|)$
Devolva d

Distancia$_R(G, w, i, j, k)$

Se $i = j$
    Devolva 0
Se $k = 0$
    Se $(v_i, v_j) \in A(G)$
        Devolva $w(v_i, v_j)$
    Devolva ∞
$D \leftarrow$ Distancia$_R(G, w, i, j, k-1)$
$D' \leftarrow$ Distancia$_R(G, w, i, k, k-1)$ + Distancia$_R(G, w, k, j, k-1)$
Se $D' < D$
    Devolva D'
Devolva D

- Repetições de subproblemas.
- Complexidade $\Theta(n^2 3^n)$.

Algoritmo de Floyd–Warshall - versão com 2 dimensões

Distancias(G, w)

```
d ← matriz |V(G)| × |V(G)|
Para k ← 0 até |V(G)|
    Distancias(G, w, d, k)
Devolva d
```

Distancias(G, w, d, k)

```
Para i ← 1 até |V(G)|
    Para j ← 1 até |V(G)|
        d[i, j] ← Distancia(G, w, d, i, j, k)
Devolva d
```

Distancia(G, w, d, i, j, k)

```
Se k = 0
    Se (v_i, v_j) ∈ A(G)
        Devolva w(v_i, v_j)
    Devolva ∞
D ← d[i, j]
D' ← d[i, k] + d[k, j]
Se D' < D
    Devolva D'
Devolva D
```

Algoritmo de Floyd–Warshall - versão com 2 dimensões

Distancias(G, w)

$d \leftarrow$ matriz $|V(G)| \times |V(G)|$
Para $k \leftarrow 0$ **até** $|V(G)|$
    Distancias(G, w, d, k)
Devolva d

Distancias(G, w, d, k)

Para $i \leftarrow 1$ **até** $|V(G)|$
    Para $j \leftarrow 1$ **até** $|V(G)|$
        $d[i, j] \leftarrow$ Distancia(G, w, d, i, j, k)
Devolva d

Distancia(G, w, d, i, j, k)

Se $k = 0$
    Se $(v_i, v_j) \in A(G)$
        Devolva $w(v_i, v_j)$
    Devolva ∞
$D \leftarrow d[i, j]$
$D' \leftarrow d[i, k] + d[k, j]$
Se $D' < D$
    Devolva D'
Devolva D

- Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de forma que

Algoritmo de Floyd–Warshall - versão com 2 dimensões

Distancias(G, w)

```
d ← matriz |V(G)| × |V(G)|
Para k ← 0 até |V(G)|
      Distancias(G, w, d, k)
Devolva d
```

Distancias(G, w, d, k)

```
Para i ← 1 até |V(G)|
      Para j ← 1 até |V(G)|
            d[i, j] ← Distancia(G, w, d, i, j, k)
Devolva d
```

Distancia(G, w, d, i, j, k)

```
Se k = 0
      Se (v_i, v_j) ∈ A(G)
            Devolva w(v_i, v_j)
      Devolva ∞
D ← d[i, j]
D' ← d[i, k] + d[k, j]
Se D' < D
      Devolva D'
Devolva D
```

- Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de forma que

$$d[i,j] = d_{G,w}(v_i, v_j, k),$$

para todo $1 \leq i, j \leq |V(G)|$,

Algoritmo de Floyd–Warshall - versão com 2 dimensões

Distancias(G, w)

```
d ← matriz |V(G)| × |V(G)|
Para k ← 0 até |V(G)|
    Distancias(G, w, d, k)
Devolva d
```

Distancias(G, w, d, k)

```
Para i ← 1 até |V(G)|
    Para j ← 1 até |V(G)|
        d[i, j] ← Distancia(G, w, d, i, j, k)
Devolva d
```

Distancia(G, w, d, i, j, k)

```
Se k = 0
    Se (v_i, v_j) ∈ A(G)
        Devolva w(v_i, v_j)
    Devolva ∞
D ← d[i, j]
D' ← d[i, k] + d[k, j]
Se D' < D
    Devolva D'
Devolva D
```

- Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de forma que

$$d[i,j] = d_{G,w}(v_i, v_j, k),$$

para todo $1 \leq i, j \leq |V(G)|$, assumindo que, ao início,

$$d[i,j] = d_{G,w}(v_i, v_j, k-1)$$

Algoritmo de Floyd–Warshall - versão com 2 dimensões

Distancias(G, w)

```
d ← matriz |V(G)| × |V(G)|
Para k ← 0 até |V(G)|
    Distancias(G, w, d, k)
Devolva d
```

Distancias(G, w, d, k)

```
Para i ← 1 até |V(G)|
    Para j ← 1 até |V(G)|
        d[i, j] ← Distancia(G, w, d, i, j, k)
Devolva d
```

Distancia(G, w, d, i, j, k)

```
Se k = 0
    Se (v_i, v_j) ∈ A(G)
        Devolva w(v_i, v_j)
    Devolva ∞
D ← d[i, j]
D' ← d[i, k] + d[k, j]
Se D' < D
    Devolva D'
Devolva D
```

- Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de forma que

$$d[i,j] = d_{G,w}(v_i, v_j, k),$$

para todo $1 \leq i, j \leq |V(G)|$, assumindo que, ao início,

$$d[i,j] = d_{G,w}(v_i, v_j, k-1)$$

- Distancia(G, w, d, i, j, k) devolve $d_{G,w}(v_i, v_j, k)$

Algoritmo de Floyd–Warshall - versão com 2 dimensões

Distancias(G, w)

```
d ← matriz |V(G)| × |V(G)|
Para k ← 0 até |V(G)|
      Distancias(G, w, d, k)
Devolva d
```

Distancias(G, w, d, k)

```
Para i ← 1 até |V(G)|
      Para j ← 1 até |V(G)|
            d[i, j] ← Distancia(G, w, d, i, j, k)
Devolva d
```

Distancia(G, w, d, i, j, k)

```
Se k = 0
      Se (v_i, v_j) ∈ A(G)
            Devolva w(v_i, v_j)
      Devolva ∞
D ← d[i, j]
D' ← d[i, k] + d[k, j]
Se D' < D
      Devolva D'
Devolva D
```

- Distancias(G, w, d, k) devolve as k–distâncias entre os vértices de forma que

$$d[i,j] = d_{G,w}(v_i, v_j, k),$$

para todo $1 \leq i, j \leq |V(G)|$, assumindo que, ao início,

$$d[i,j] = d_{G,w}(v_i, v_j, k-1)$$

- Distancia(G, w, d, i, j, k) devolve $d_{G,w}(v_i, v_j, k)$ assumindo que

$$\begin{aligned} d[i,j] &= d_{G,w}(v_i, v_j, k-1), \text{ e} \\ d[i,k] &= d_{G,w}(v_i, v_k, k-1), \text{ e} \\ d[k,j] &= d_{G,w}(v_k, v_j, k-1). \end{aligned}$$

Teorema 116

É possível computar as distâncias entre todos os pares de vértices de um grafo direcionado com pesos de n vértices em tempo $O(n^3)$.

Teorema 116

É possível computar as distâncias entre todos os pares de vértices de um grafo direcionado com pesos de n vértices em tempo $O(n^3)$.

Demonstração.

1. É possível implementar o algoritmo de maneira que cada execução de Distancia(G, w, d, i, j, k) toma tempo $O(1)$.

Teorema 116

É possível computar as distâncias entre todos os pares de vértices de um grafo direcionado com pesos de n vértices em tempo $O(n^3)$.

Demonstração.

1. É possível implementar o algoritmo de maneira que cada execução de Distancia(G, w, d, i, j, k) toma tempo $O(1)$.
2. Assim, cada execução de Distancia(G, w, d, k) toma tempo $O(n^2)$ e

Teorema 116

É possível computar as distâncias entre todos os pares de vértices de um grafo direcionado com pesos de n vértices em tempo $O(n^3)$.

Demonstração.

1. É possível implementar o algoritmo de maneira que cada execução de Distancia(G, w, d, i, j, k) toma tempo $O(1)$.
2. Assim, cada execução de Distancia(G, w, d, k) toma tempo $O(n^2)$ e
3. a execução de Distancias(G, w) toma tempo $O(n^3)$.

□