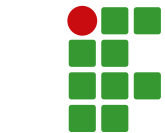
Menor Caminho

Teoria dos Grafos

Prof. Daniel Saad Nogueira Nunes



INSTITUTO FEDERAL
Brasília

Campus
Taguatinga

Sumário

Introdução

SSSP

APSP

Introdução

Sejam $G = (V, E)$ um grafo direcionado e $w : E \to \mathbb{R}$ uma função que atribua pesos nas arestas. A grosso modo O problema do menor caminho consiste em encontrar o caminho de menor custo entre dois vértices u e v.



Figura: `https://download.osgeo.org/pgrouting/foss4g2010/workshop/docs/html/chapters/shortest_path.html`

Aplicações

- Roteamento de pacotes em redes de computadores.
- Planejamento de rotas em sistemas de navegação.
- Planejamento de rotas em jogos.
- Similaridade de genomas (distância de reversão).
- ...

Formalização

Definição (Custo de um caminho)

Seja $G = (V, E)$ um grafo direcionado e $w : E \to \mathbb{R}$ uma função que atribua pesos às arestas. Seja $P = \langle v_1, v_2, \cdots, v_k \rangle$ um caminho em G. O custo de P é definido como:

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

Formalização

Definição (Função δ)

O custo $\delta(u, v)$ do menor caminho entre dois vértices u e v é definido como:

$$\delta(u,v) = \begin{cases} \infty & \text{se não existe caminho de } u \text{ para } v \\ \min\{w(P) \mid P \text{ é um caminho de } u \text{ para } v\} & \text{, caso contrário.} \end{cases}$$

Variações do problema do menor caminho

Existem duas variações do problema do menor caminho:

- **SSSP (Single Source Shortest Path)**: Encontrar o menor caminho de um vértice s para todos os outros vértices do grafo.
- **APSP (All Pairs Shortest Path)**: Encontrar o menor caminho entre quaisquer pares de vértices do grafo.

Algoritmos para o problema do menor caminho

Examinaremos diferentes algoritmos para resolver o problema do menor caminho, cada um com suas próprias características, restrições, e complexidades.

Sumário

Introdução

SSSP

APSP

SSSP

Definição (Menor caminho com origem única (SSSP))

Dado um grafo direcionado $G = (V, E)$ e uma função de peso $w : E \to \mathbb{R}$, o problema do menor caminho com origem única consiste em encontrar o menor caminho de um vértice $s \in V$ para todos os outros vértices do grafo.

Em outras, palavras, o objetivo é calcular $\delta(s, v)$ para todo $v \in V$.

SSSP

- Antes de comentar os algoritmos de menor caminho com origem única, vamos discutir alguns conceitos importantes.

Sumário

SSSP
Conceitos importantes
Dijkstra
Bellman-Ford
SSSP em DAGs

Distância atual

Definição (Distância atual $d[u]$)

A distância atual $d[u]$ de um vértice $u \in V$ é uma estimativa do custo do menor caminho de s para u. Em outras palavras, $d[u] \geq \delta(s, u)$ para todo $u \in V$.

Conforme os algoritmos de SSSP são executados, os valores de $d[u]$ são atualizados para refletir estimativas mais precisas do custo do menor caminho de s para u, até que $d[u] = \delta(s, u)$ para todos os vértices $u \in V$.

Predecessores

Definição (Predecessor $p[u]$)

O predecessor $p[u]$ de um vértice $u \in V$ é o vértice que precede u no menor caminho de s para u. Em outras palavras, se existe um caminho de s para u passando imediatamente por vértice v antes de chegar a u, então $p[u] = v$.

Os predecessores são usados para reconstruir o menor caminho de s para qualquer vértice $u \in V$.

Relaxamento

Definição (Relaxamento)

O relaxamento de uma aresta $(u, v) \in E$ *é o processo de atualizar a distância atual* $d[v]$ *do vértice* v *com base na distância atual* $d[u]$ *do vértice* u *e no peso da aresta* $w((u, v))$.

O relaxamento é realizado da seguinte forma:

1. *Se* $d[v] > d[u] + w((u, v))$, *então atualizamos* $d[v]$ *para* $d[u] + w(u, v)$ *e o predecessor de* v *para* u.
2. *Caso contrário, não fazemos nenhuma alteração.*

O processo retorna **true** *se a distância atual* $d[v]$ *foi atualizada, e* **false** *caso contrário.*

Relaxamento

Definição (Relaxamento)

Algoritmo 1: $\text{RELAX}(u, v)$

```
1 if d[v] > d[u] + w((u, v)) then
2     d[v] ← d[u] + w((u, v))
3     p[v] ← u
4     return true
5 return false
```

Em outras palavras, o relaxamento de uma aresta (u, v) verifica se o caminho de s para v passando por u é mais curto do que o caminho atualmente conhecido de s para v. Se for, atualizamos a distância atual $d[v]$ para refletir essa descoberta.

Inicialização

Definição (Inicialização)

Antes de iniciar os algoritmos de SSSP, é necessário inicializar as distâncias atuais $d[u]$ e os predecessores $p[u]$para todos os vértices $u \in V$. A inicialização é feita da seguinte forma:

1. *Para o vértice de origem s, definimos $d[s] = 0$ e $p[s] = \perp$.*
2. *Para todos os outros vértices $u \in V$, definimos $d[u] = \infty$ e $p[u] = \perp$.*

Algoritmo 2: INITIALIZE-SSSP(G, s)

```
1 foreach u ∈ V do
2 |   d[u] ← ∞
3 |   p[u] ← ⊥
4 d[s] ← 0
```

Propriedades importantes

Propriedades de menores caminhos

- **Desigualdade triangular**:a função δ satisfaz a desigualdade triangular: $\delta(s, v) \leq \delta(s, u) + w((u, v))$ para todo $(u, v) \in E$.
- **Cota superior de** $d[u]$: $d[u] \geq \delta(s, u)$ para todo $u \in V$.
- **Inexistência de caminho**: se não existe caminho de s para u, então $d[u] = \delta(s, u) = \infty$.
- **Convergência:** Seja P um caminho de s para v que tem u como predecessor. Se $d[u] = \delta(s, u)$ em qualquer momento antes do relaxamento da aresta (u, v), então, em qualquer momento após o relaxamento de (u, v), teremos $d[v] = \delta(s, v)$.

Propriedades importantes

Propriedades de menores caminhos

- **Relaxamento de caminhos**: Se $P = \langle v_0, v_1, \cdots, v_k \rangle$ é um menor caminho de s para v_k e as arestas (v_{i-1}, v_i) são relaxadas na ordem em que aparecem em P, para $1 \leq i \leq k$, então, após o relaxamento de todas essas arestas, teremos $d[v_k] = \delta(s, v_k)$, mesmo que outros relaxamentos sejam feitos entre eles.
- **Predecessores de menor caminho**: uma vez que $d[u] = \delta(s, u)$ para todo $u \in V$, os vértices $\langle u, p[u], p[p[u]], \ldots, s \rangle$ geram um subgrafo de árvore de menor caminho enraizada em s.

Convenção

Aritmética com ∞ e $-\infty$

Como as distâncias atuais $d[u]$ podem ser inicialmente definidas como ∞, é importante estabelecer algumas convenções para a aritmética envolvendo ∞ e $-\infty$:

- $\infty + x = \infty = \infty + x = \infty$ para qualquer número real x.
- $-\infty + x = -\infty = -\infty + x = -\infty$ para qualquer número real x.
- $\infty + \infty = \infty$
- $-\infty + -\infty = -\infty$
- $\infty + -\infty$ é indefinido.

Na prática, ao implementar algoritmos de menor caminho, é comum usar um valor muito grande para representar ∞ e um valor muito pequeno para representar $-\infty$, garantindo que as operações aritméticas sejam tratadas corretamente.

Sumário

SSSP
- Conceitos importantes
- Dijkstra
- Bellman-Ford
- SSSP em DAGs

Dijkstra

- Proposto por Edsger W. Dijkstra em 1956.
- A cada passo, o algoritmo seleciona o vértice com a menor distância atual $d[u]$ e relaxa todas as arestas que saem desse vértice.
- O algoritmo de Dijkstra funciona apenas para grafos com pesos de arestas não-negativos, ou seja, $w(e) \geq 0$ para todo $e \in E$.
- Caso haja presença de arestas com pesos negativos, o algoritmo de Dijkstra pode produzir resultados incorretos.

Dijkstra

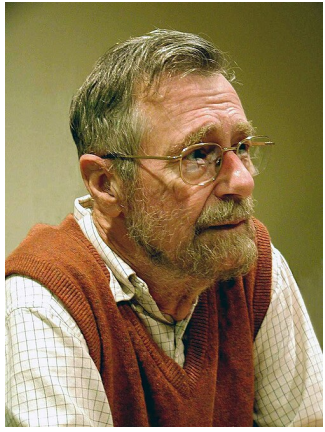


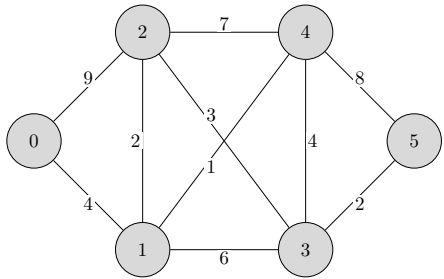
Figura: Fonte: `https://en.wikipedia.org/wiki/Edsger_W._Dijkstra`

Dijkstra

Algoritmo 3: DIJKSTRA(G, s)

```
1 INITIALIZE-SSSP(G, s)
2 Q ← V
3 while Q ≠ ∅ do
4     u ← EXTRACT-MIN(Q)
5     Q ← Q \ {u}
6     foreach (u, v) ∈ E do
7         if RELAX(u, v) then
8             DECREASE-KEY(Q, v)
```

Dijkstra: exemplo

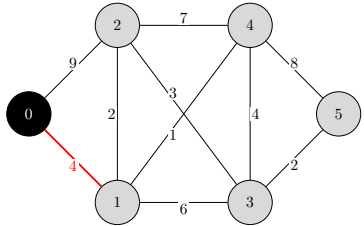


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & \infty & \infty & \infty & \infty & \infty \end{matrix}$

$Q = \{0, 1, 2, 3, 4, 5\}$

inicialização

Dijkstra: exemplo

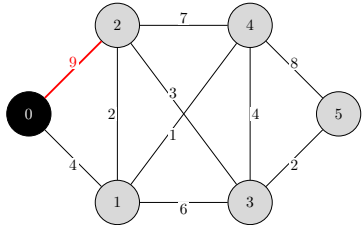


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & \infty & \infty & \infty & \infty \end{matrix}$

$Q = \{1, 2, 3, 4, 5\}$

processando vértice 0, relaxamento da aresta $(0, 1)$

Dijkstra: exemplo

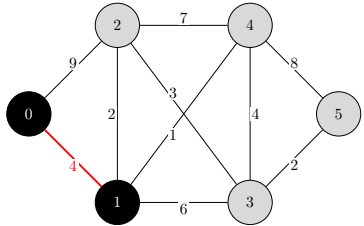


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 9 & \infty & \infty & \infty \end{matrix}$

$Q = \{1, 2, 3, 4, 5\}$

processando vértice 0, relaxamento da aresta $(0, 2)$

Dijkstra: exemplo

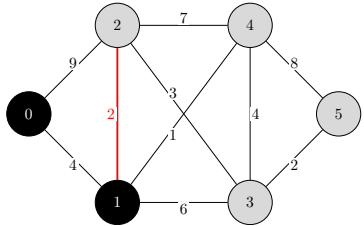


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 9 & \infty & \infty & \infty \end{matrix}$

$Q = \{2, 3, 4, 5\}$

processando vértice 1, relaxamento da aresta $(1, 0)$

Dijkstra: exemplo

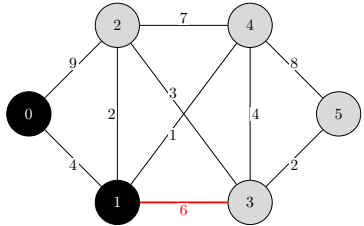


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & \infty & \infty & \infty \end{matrix}$

$Q = \{2, 3, 4, 5\}$

processando vértice 1, relaxamento da aresta $(1, 2)$

Dijkstra: exemplo

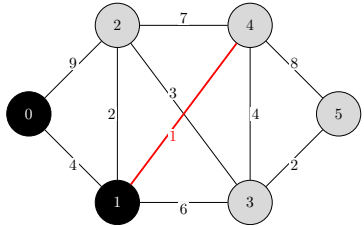


$d = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 10 & \infty & \infty \end{array}$

$Q = \{2, 3, 4, 5\}$

processando vértice 1, relaxamento da aresta $(1, 3)$

Dijkstra: exemplo

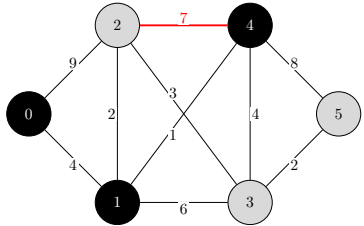


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 10 & 5 & \infty \end{matrix}$

$Q = \{2, 3, 4, 5\}$

processando vértice 1, relaxamento da aresta $(1, 4)$

Dijkstra: exemplo

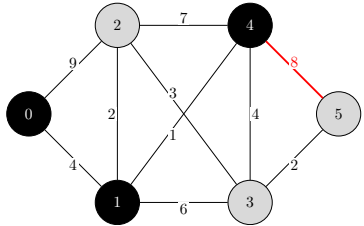


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 10 & 5 & \infty \end{matrix}$

$Q = \{2, 3, 5\}$

processando vértice 4, relaxamento da aresta $(4, 2)$

Dijkstra: exemplo

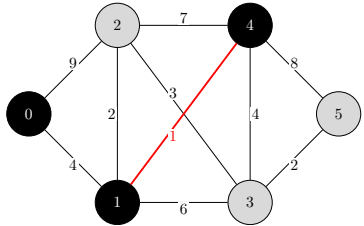


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 10 & 5 & 13 \end{matrix}$

$Q = \{2, 3, 5\}$

processando vértice 4, relaxamento da aresta $(4, 5)$

Dijkstra: exemplo

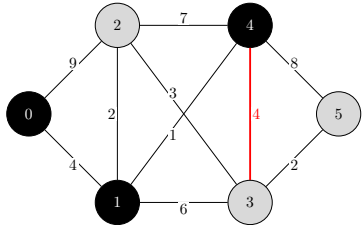


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 10 & 5 & 13 \end{matrix}$

$Q = \{2, 3, 5\}$

processando vértice 4, relaxamento da aresta $(4, 1)$

Dijkstra: exemplo

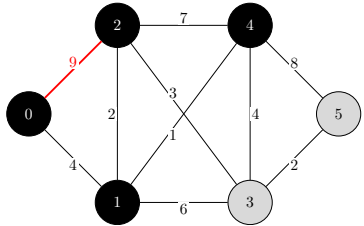


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 13 \end{matrix}$

$Q = \{2, 3, 5\}$

processando vértice 4, relaxamento da aresta $(4, 3)$

Dijkstra: exemplo

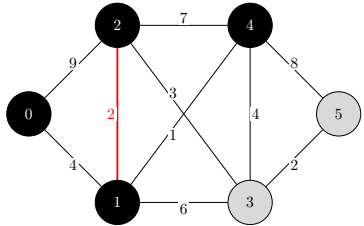


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 13 \end{matrix}$

$Q = \{3, 5\}$

processando vértice 2, relaxamento da aresta $(2, 0)$

Dijkstra: exemplo

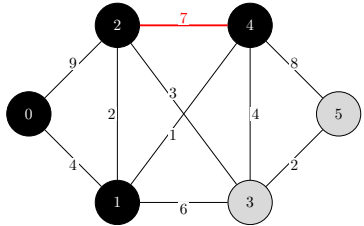


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 13 \end{matrix}$

$Q = \{3, 5\}$

processando vértice 2, relaxamento da aresta $(2, 1)$

Dijkstra: exemplo

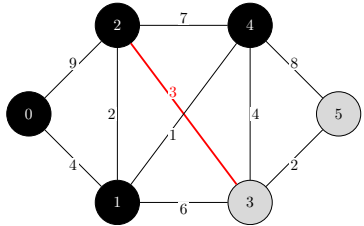


$d = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 13 \end{array}$

$Q = \{3, 5\}$

processando vértice 2, relaxamento da aresta $(2, 4)$

Dijkstra: exemplo

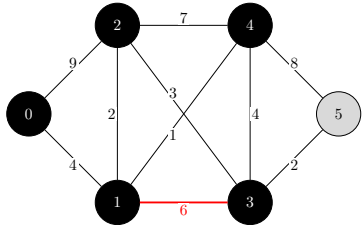


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 13 \end{matrix}$

$Q = \{3, 5\}$

processando vértice 2, relaxamento da aresta $(2, 3)$

Dijkstra: exemplo

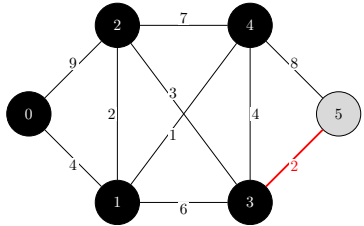


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 13 \end{matrix}$

$Q = \{5\}$

processando vértice 3, relaxamento da aresta $(3, 1)$

Dijkstra: exemplo

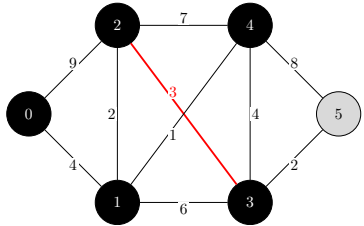


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 11 \end{matrix}$

$Q = \{5\}$

processando vértice 3, relaxamento da aresta $(3, 5)$

Dijkstra: exemplo

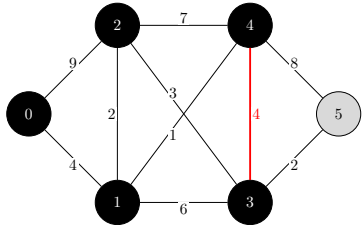


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 11 \end{matrix}$

$Q = \{5\}$

processando vértice 3, relaxamento da aresta $(3, 2)$

Dijkstra: exemplo

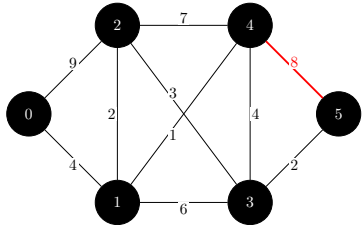


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 11 \end{matrix}$

$Q = \{5\}$

processando vértice 3, relaxamento da aresta $(3, 4)$

Dijkstra: exemplo

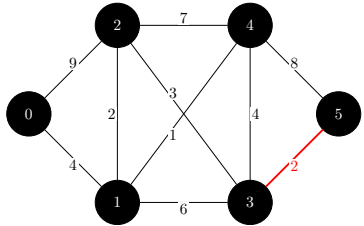


$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 11 \end{matrix}$

$Q = \{\}$

processando vértice 5, relaxamento da aresta $(5, 4)$

Dijkstra: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 4 & 6 & 9 & 5 & 11 \end{matrix}$

$Q = \{\}$

processando vértice 5, relaxamento da aresta $(5, 3)$

Dijkstra

- EXTRACT-MIN(Q) retorna o vértice com a menor distância atual $d[u]$ do conjunto Q.
- DECREASE-KEY(Q, v) atualiza $d[v]$ na estrutura de dados Q após o relaxamento da aresta (u, v).
- No pior caso: n chamadas a EXTRACT-MIN(Q) e m chamadas a DECREASE-KEY(Q, v).
- A estrutura utilizara para responder essas duas operações afeta a complexidade final do algoritmo de Dijkstra.

Dijkstra

Análise de complexidade

Fazemos n chamadas a EXTRACT-MIN(Q) e m chamadas a DECREASE-KEY(Q, v). A complexidade dependerá de qual estrutura utilizaremos para representar Q.

Vetor:

Caso Q seja implementado como um vetor, achar o vértice u com a menor distância atual $d[u]$ leva $\Theta(n)$ tempo, por outro lado, e atualizar $d[v]$ leva $\Theta(1)$ tempo. Assim, a complexidade total do algoritmo de Dijkstra é $\Theta(n^2 + m) = \Theta(n^2)$.

Dijkstra

Análise de complexidade

Fazemos n chamadas a EXTRACT-MIN(Q) e m chamadas a DECREASE-KEY(Q, v). A complexidade dependerá de qual estrutura utilizaremos para representar Q.

Heap binária:

Caso Q seja implementado como uma fila de prioridade baseada em **heap binária**, achar o vértice u com a menor distância atual $d[u]$ leva $\Theta(\log n)$ tempo, e atualizar $d[v]$ leva $\Theta(\lg n)$ tempo. Assim, a complexidade total do algoritmo de Dijkstra é $\Theta((n + m) \lg n)$. No caso típico em que $m \geq n - 1$, a complexidade do algoritmo de Dijkstra é $\Theta(m \lg n)$.

Dijkstra

Análise de complexidade

Fazemos n chamadas a EXTRACT-MIN(Q) e m chamadas a DECREASE-KEY(Q, v). A complexidade dependerá de qual estrutura utilizaremos para representar Q.

Heap de Fibonacci:

Caso Q seja implementado como uma fila de prioridade baseada em **heap de Fibonacci**, achar o vértice u com a menor distância atual $d[u]$ leva $\Theta(\lg n)$ tempo, e atualizar $d[v]$ leva $\Theta(1)$ tempo. Assim, a complexidade total do algoritmo de Dijkstra é $\Theta(n \lg n + m)$.

Dijkstra

Análise de complexidade

Fazemos n chamadas a EXTRACT-MIN(Q) e m chamadas a DECREASE-KEY(Q, v). A complexidade dependerá de qual estrutura utilizaremos para representar Q.

Conclusão:

A escolha da estrutura de dados para representar Q afeta significativamente a complexidade do algoritmo de Dijkstra. Para grafos com $m = \Theta(n^2)$, a implementação com vetor é suficiente, enquanto para grafos com $m = O(n^2/\lg n)$, a implementação com heap binária ou heap de Fibonacci é mais eficiente.

Sumário

SSSP
- Conceitos importantes
- Dijkstra
- Bellman-Ford
- SSSP em DAGs

Bellman-Ford

Ideia do algoritmo

- O algoritmo de Bellman-Ford é um algoritmo de menor caminho com origem única que pode lidar com grafos que possuem arestas com pesos negativos.
- Contudo, ele não funciona se houver ciclos cuja soma dos pesos seja negativa no grafo.
- Ele se baseia em uma ideia análoga à indução matemática: se o menor caminho de s para v tiver no máximo k arestas, então após k iterações do algoritmo, teremos $d[v] = \delta(s, v)$.
- Caso base: $k = 0$, o menor caminho de s para s é o próprio vértice s, e $d[s] = \delta(s, s) = 0$.

Bellman-Ford

Ideia do algoritmo

- Em $n-1$ iterações, o algoritmo consegue encontrar o menor caminho de s para todos os vértices do grafo, desde que não haja ciclos de peso negativo.
- Funciona, pois o menor caminho entre dois vértices não pode ter mais do que $n-1$ arestas.

itemize

Detecção de ciclos de peso negativo

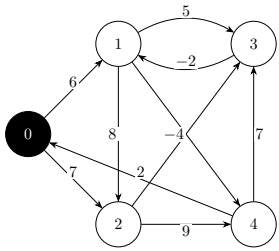
- Na ocorrência de um ciclo cuja soma dos pesos seja negativa, o algoritmo de Bellman-Ford consegue detectar a presença desse ciclo e retorna **false**.
- A detecção é realizada após as $n - 1$ iterações do algoritmo.
- A premissa é que se houver um ciclo de peso negativo, então haverá pelo menos uma aresta que ainda pode ser relaxada, mesmo após as $n - 1$ iterações.
- As arestas envolvidas no ciclo de peso negativo podem ser relaxadas indefinidamente, o que indica a presença do ciclo.

Bellman-Ford

Algoritmo 4: BELLMAN-FORD(G, s)

```
1 INITIALIZE-SSSP(G, s)
2 for i ← 1 to n − 1 do
3     foreach (u, v) ∈ E do
4         RELAX(u, v)
5 foreach (u, v) ∈ E do
6     if RELAX(u, v) then
7         return false
8 return true
```

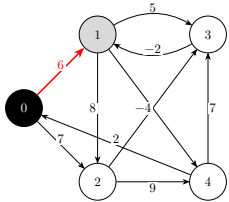
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & \infty & \infty & \infty & \infty \end{matrix}$

inicialização

Bellman-Ford: exemplo

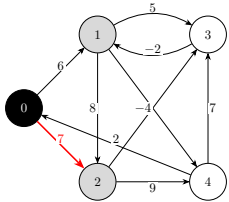


$d =$

0	1	2	3	4
0	6	∞	∞	∞

interação 1, relaxamento da aresta 0: $(0, 1)$

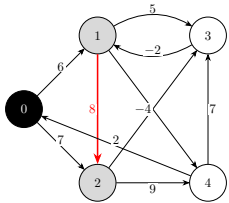
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 6 & 7 & \infty & \infty \end{matrix}$

interação 1, relaxamento da aresta 1: $(0, 2)$

Bellman-Ford: exemplo

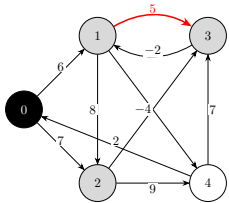


$d =$

0	1	2	3	4
0	6	7	∞	∞

interação 1, relaxamento da aresta 2: $(1, 2)$

Bellman-Ford: exemplo

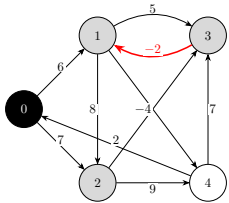


$d =$

0	1	2	3	4
0	6	7	11	∞

interação 1, relaxamento da aresta 3: $(1, 3)$

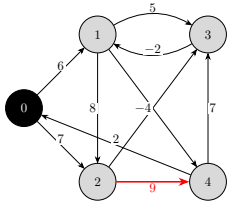
Bellman-Ford: exemplo



$d =$ 0 1 2 3 4
0 6 7 11 ∞

interação 1, relaxamento da aresta 4: $(3, 1)$

Bellman-Ford: exemplo

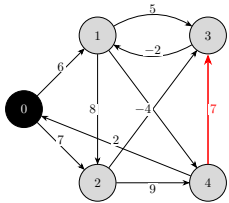


$d =$

0	1	2	3	4
0	6	7	11	16

interação 1, relaxamento da aresta 5: $(2, 4)$

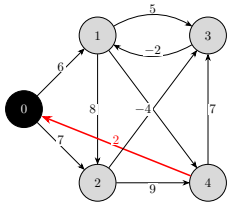
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 6 & 7 & 11 & 16 \end{matrix}$

interação 1, relaxamento da aresta 6: $(4, 3)$

Bellman-Ford: exemplo

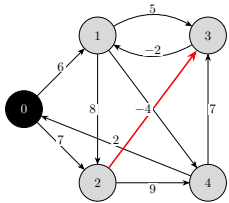


$d =$

0	1	2	3	4
0	6	7	11	16

interação 1, relaxamento da aresta 7: $(4, 0)$

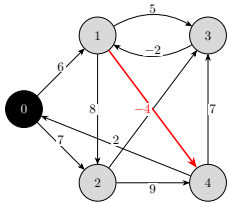
Bellman-Ford: exemplo



$d =$ 0 1 2 3 4
0 6 7 4 16

interação 1, relaxamento da aresta 8: $(2, 3)$

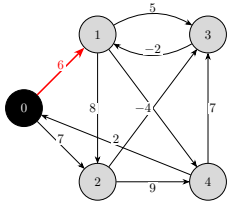
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 6 & 7 & 4 & 2 \end{matrix}$

interação 1, relaxamento da aresta 9: $(1, 4)$

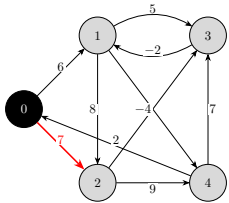
Bellman-Ford: exemplo



$d =$ 0 1 2 3 4
0 6 7 4 2

interação 2, relaxamento da aresta 0: $(0, 1)$

Bellman-Ford: exemplo

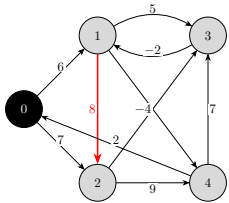


$d =$

0	1	2	3	4
0	6	7	4	2

interação 2, relaxamento da aresta 1: $(0, 2)$

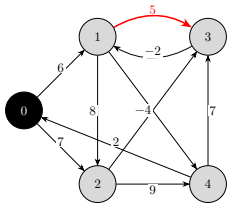
Bellman-Ford: exemplo



$d =$ 0 1 2 3 4
0 6 7 4 2

interação 2, relaxamento da aresta 2: $(1, 2)$

Bellman-Ford: exemplo

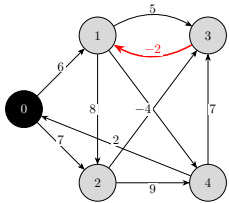


$d =$

0	1	2	3	4
0	6	7	4	2

interação 2, relaxamento da aresta 3: $(1, 3)$

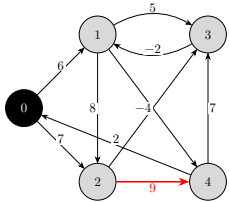
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & 2 \end{matrix}$

interação 2, relaxamento da aresta 4: $(3, 1)$

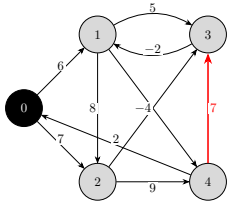
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & 2 \end{matrix}$

interação 2, relaxamento da aresta 5: $(2, 4)$

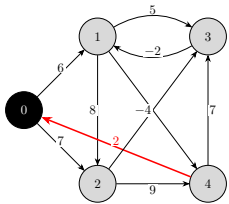
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & 2 \end{matrix}$

interação 2, relaxamento da aresta 6: $(4, 3)$

Bellman-Ford: exemplo

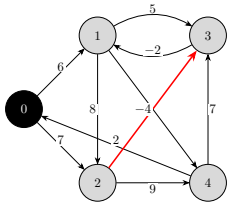


$d =$

0	1	2	3	4
0	2	7	4	2

interação 2, relaxamento da aresta 7: $(4, 0)$

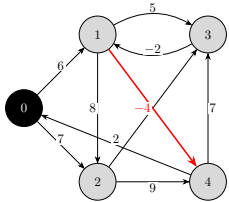
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & 2 \end{matrix}$

interação 2, relaxamento da aresta 8: $(2, 3)$

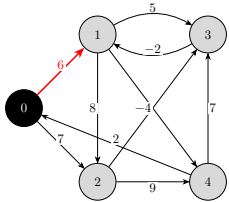
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 2, relaxamento da aresta 9: $(1, 4)$

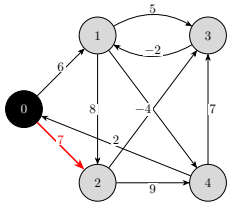
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 0: $(0, 1)$

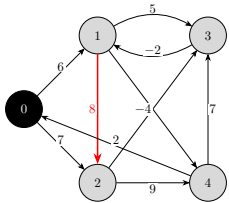
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 1: $(0, 2)$

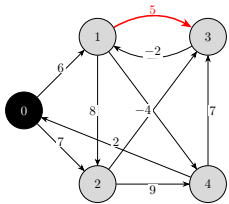
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 2: $(1, 2)$

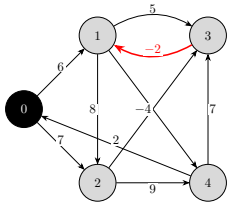
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 3: $(1, 3)$

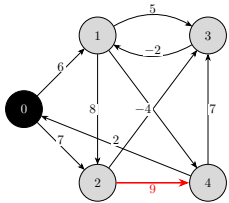
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 4: $(3, 1)$

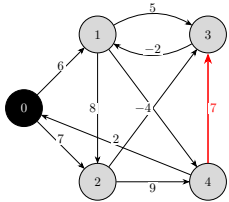
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 5: $(2, 4)$

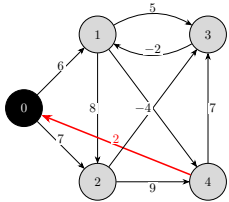
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 6: $(4, 3)$

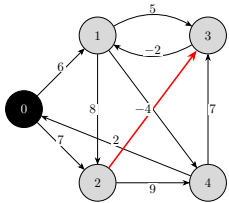
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 7: $(4, 0)$

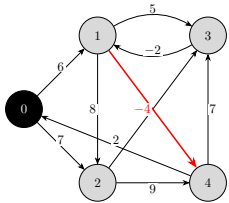
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 8: $(2, 3)$

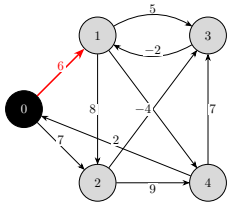
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 3, relaxamento da aresta 9: $(1, 4)$

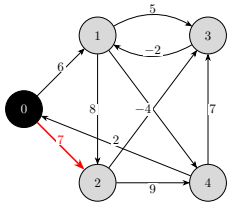
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 0: $(0, 1)$

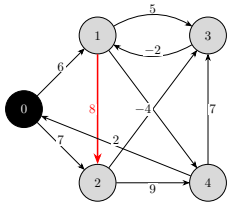
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 1: $(0, 2)$

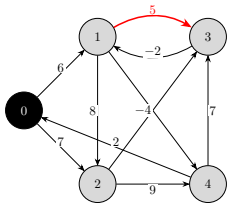
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 2: $(1, 2)$

Bellman-Ford: exemplo

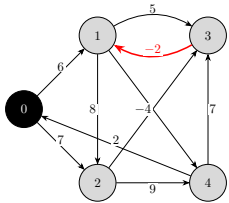


$d =$

0	1	2	3	4
0	2	7	4	−2

interação 4, relaxamento da aresta 3: $(1, 3)$

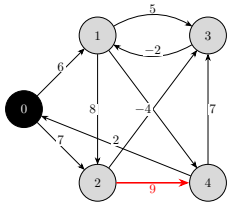
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 4: $(3, 1)$

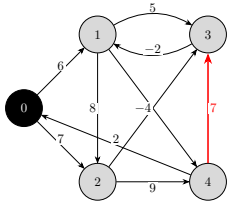
Bellman-Ford: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 5: $(2, 4)$

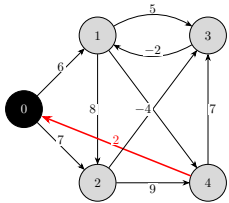
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 6: $(4, 3)$

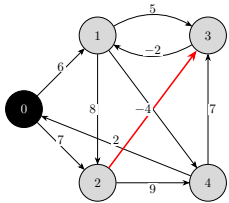
Bellman-Ford: exemplo



$d =$ 0 1 2 3 4
0 2 7 4 −2

interação 4, relaxamento da aresta 7: $(4, 0)$

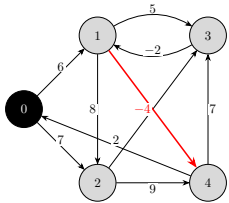
Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 8: $(2, 3)$

Bellman-Ford: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 7 & 4 & -2 \end{matrix}$

interação 4, relaxamento da aresta 9: $(1, 4)$

Bellman-Ford

Análise de complexidade

O algoritmo de Bellman-Ford possui uma complexidade de tempo de $\Theta(n \cdot m)$, no pior caso. Isso ocorre porque o algoritmo realiza $n - 1$ iterações sobre todas as arestas do grafo, e cada iteração envolve a execução do relaxamento para cada aresta.

É possível parar o algoritmo de Bellman-Ford prematuramente caso nenhuma aresta seja relaxada em alguma das $n - 1$ iterações, mas isso não muda o comportamento de pior caso do algoritmo, que continua tendo custo $\Theta(n \cdot m)$.

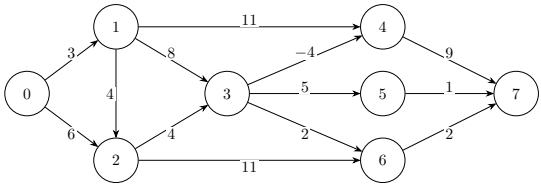
Sumário

SSSP
- Conceitos importantes
- Dijkstra
- Bellman-Ford
- SSSP em DAGs

SSSP em DAGs

- No caso especial em que o grafo G é um DAG, podemos resolver o problema do menor caminho com origem única em tempo $\Theta(n + m)$ usando ordenação topológica ou uma busca em profundida, ou largura modificada.
- Um caminho mais curto para um nó v passa necessariamente por um de seus predecessores, e como o grafo é acíclico, podemos processar os vértices em ordem topológica.

Menor caminho em DAGs: exemplo

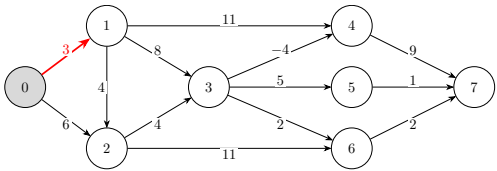


$d =$

0	1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞	∞

inicialização

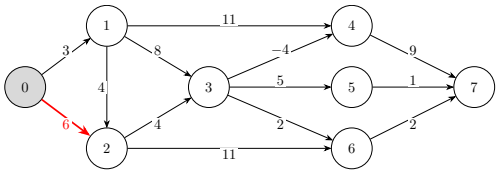
Menor caminho em DAGs: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & \infty & \infty & \infty & \infty & \infty & \infty \end{matrix}$

processando vértice 0, relaxamento da aresta $(0, 1)$

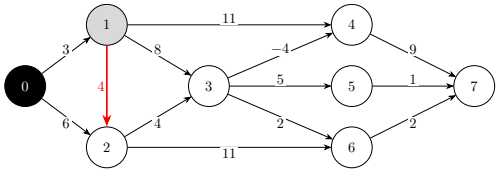
Menor caminho em DAGs: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & \infty & \infty & \infty & \infty & \infty \end{matrix}$

processando vértice 0, relaxamento da aresta $(0, 2)$

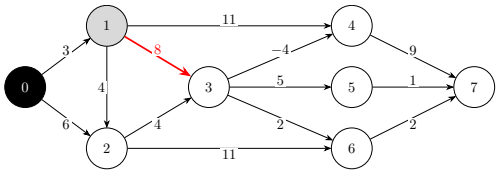
Menor caminho em DAGs: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & \infty & \infty & \infty & \infty & \infty \end{matrix}$

processando vértice 1, relaxamento da aresta $(1, 2)$

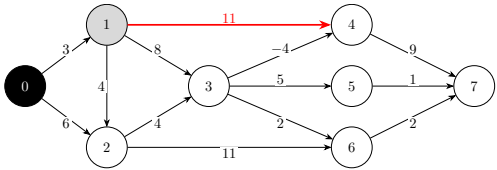
Menor caminho em DAGs: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 11 & \infty & \infty & \infty & \infty \end{matrix}$

processando vértice 1, relaxamento da aresta $(1, 3)$

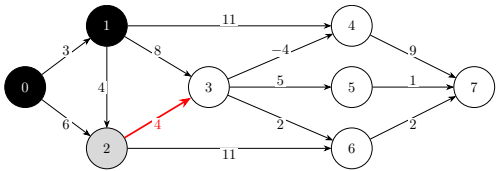
Menor caminho em DAGs: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 11 & 14 & \infty & \infty & \infty \end{matrix}$

processando vértice 1, relaxamento da aresta $(1, 4)$

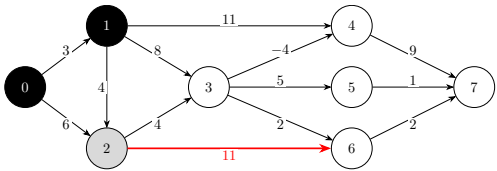
Menor caminho em DAGs: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 14 & \infty & \infty & \infty \end{matrix}$

processando vértice 2, relaxamento da aresta $(2, 3)$

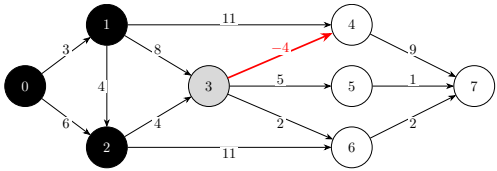
Menor caminho em DAGs: exemplo



$d = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 14 & \infty & 17 & \infty \end{array}$

processando vértice 2, relaxamento da aresta $(2, 6)$

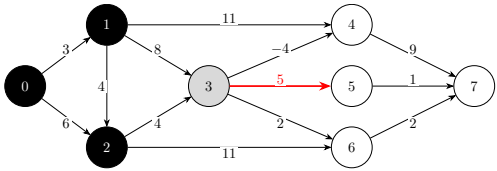
Menor caminho em DAGs: exemplo



$d = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & \infty & 17 & \infty \end{array}$

processando vértice 3, relaxamento da aresta $(3, 4)$

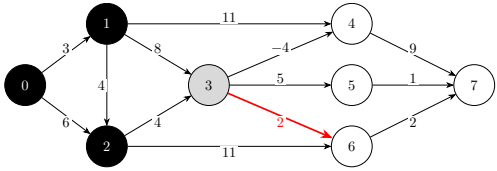
Menor caminho em DAGs: exemplo



$d = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & 15 & 17 & \infty \end{array}$

processando vértice 3, relaxamento da aresta $(3, 5)$

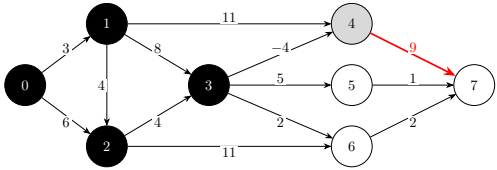
Menor caminho em DAGs: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & 15 & 12 & \infty \end{matrix}$

processando vértice 3, relaxamento da aresta $(3, 6)$

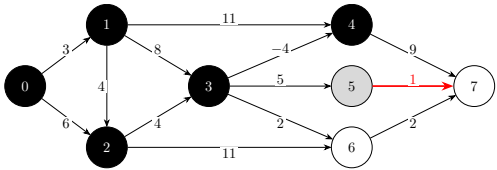
Menor caminho em DAGs: exemplo



$d = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & 15 & 12 & 15 \end{array}$

processando vértice 4, relaxamento da aresta $(4, 7)$

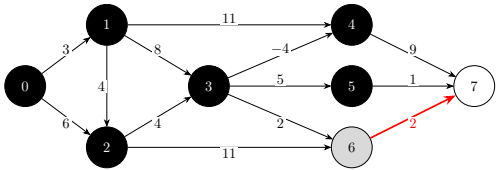
Menor caminho em DAGs: exemplo



$d = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & 15 & 12 & 15 \end{array}$

processando vértice 5, relaxamento da aresta $(5, 7)$

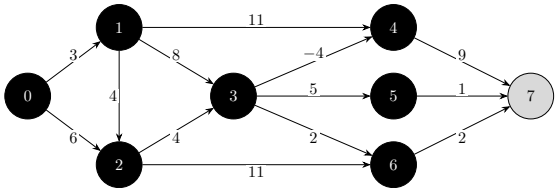
Menor caminho em DAGs: exemplo



$d =$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0 | 3 | 6 | 10 | 6 | 15 | 12 | 14 |

processando vértice 6, relaxamento da aresta $(6, 7)$

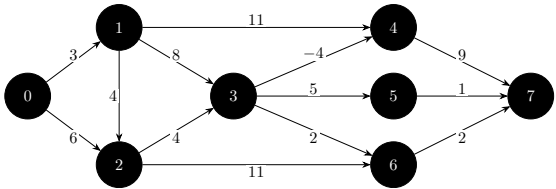
Menor caminho em DAGs: exemplo



$d =$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & 15 & 12 & 14 \end{matrix}$

processando vértice 7

Menor caminho em DAGs: exemplo



$d = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 10 & 6 & 15 & 12 & 14 \end{matrix}$

todos os vértices processados

Sumário

Introdução

SSSP

APSP

APSP

- No problema de menor caminho com todas as origens (APSP – All Pairs Shortest Path), queremos encontrar o menor caminho entre todos os pares de vértices do grafo.
- Podemos utilizar o algoritmo de Floyd-Warshall, que é um algoritmo baseado em programação dinâmica, para resolver o problema de APSP.

Sumário

APSP
Floyd-Warshall

Floyd-Warshall

Modelagem da solução

Seja $D(u, v, k)$ a distância atual do menor caminho de u para v considerando apenas os vértices $\{x_1, x_2, \cdots, x_k\}$ como possíveis vértices intermediários no caminho. A ideia é construir a solução de forma incremental, considerando cada vértice como um possível intermediário no caminho entre todos os pares de vértices.

Floyd-Warshall

Modelagem da solução

A equação de recorrência para $D(u, v, k)$ é dada por:

$$D(k, u, v) = \begin{cases} 0, & \text{se } k = 0 \text{ e } u = v \\ w((u, v)), & \text{se } k = 0 \text{ e } u \neq v \\ \min\{D(k-1, u, v), D(k-1, u, k) + D(k-1, k, v)\}, & \text{se } k > 0 \end{cases}$$

Onde $D(u, v, 0)$ é a distância direta entre u e v, ou seja, o peso da aresta (u, v) se ela existir, ou ∞ caso contrário.

O algoritmo de Floyd-Warshall itera sobre todos os vértices como possíveis intermediários, atualizando as distâncias atuais $d[u][v]$ de acordo com a relação de recorrência acima.

Intuição

- A ideia por trás do algoritmo de Floyd-Warshall é que, para cada par de vértices (u, v), podemos considerar se o caminho mais curto entre eles passa ou não por um determinado vértice intermediário k.
- Se o caminho mais curto de u para v passar por k, então a distância pode ser atualizada para a soma das distâncias de u para k e de k para v.
- Caso contrário, mantemos a distância atual como está. Ao iterar sobre todos os vértices como possíveis intermediários, garantimos que todas as combinações de caminhos sejam consideradas, resultando na determinação das distâncias mínimas entre todos os pares de vértices.

Floyd-Warshall

Otimização de memória

$$D(k,u,v) = \begin{cases} 0, & \text{se } k = 0 \text{ e } u = v \\ w((u,v)), & \text{se } k = 0 \text{ e } u \neq v \\ \min\left\{D(k-1,u,v), D(k-1,u,k) + D(k-1,k,v)\right\}, & \text{se } k > 0 \end{cases}$$

Como $D(k,u,v)$ depende apenas de $D(k-1,u,v)$, podemos otimizar o uso de memória utilizando apenas uma matriz bidimensional $d[u][v]$ para armazenar as distâncias atuais entre todos os pares de vértices. A cada iteração sobre os vértices intermediários, atualizamos essa matriz com as novas distâncias calculadas.

Floyd-Warshall

Representação

Pela equação de recorrência apresentada, é mais natural representar o grafo como uma matriz de adjacências, onde cada célula (u, v) contém o peso da aresta (u, v) se ela existir, ou ∞ caso contrário.

Floyd-Warshall

Algoritmo bottom-up

Algoritmo 5: initialize-apsp(G)

```
1 for u ← 0 to n − 1 do
2     for v ← 0 to n − 1 do
3         if u = v then d[u][v] ← 0
4         else if (u, v) ∈ E then d[u][v] ← w((u, v))
5         else d[u][v] ← ∞
6         p[u][v] ← ⊥
```

Floyd-Warshall

Algoritmo bottom-up

Algoritmo 6: FLOYD-WARSHALL(G)

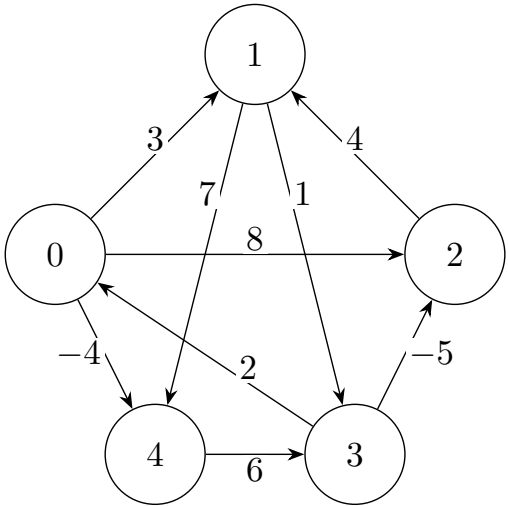
```
1 INITIALIZE-APSP(G)
2 for k ← 0 to n − 1 do
3     for u ← 0 to n − 1 do
4         for v ← 0 to n − 1 do
5             if d[u][k] + d[k][v] < d[u][v] then
6                 d[u][v] ← min(d[u][v], d[u][k] + d[k][v])
7                 p[u][v] ← k
```

Floyd-Warshall

Análise de complexidade

O algoritmo de Floyd-Warshall possui uma complexidade de tempo de $\Theta(n^3)$, pois ele realiza três loops aninhados, cada um iterando sobre todos os vértices do grafo. A complexidade de espaço é $\Theta(n^2)$, devido à necessidade de armazenar a matriz de distâncias entre todos os pares de vértices.

Floyd-Warshall: exemplo

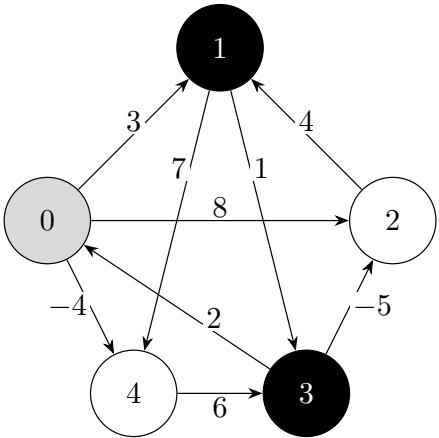


Matriz inicial

$$\begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

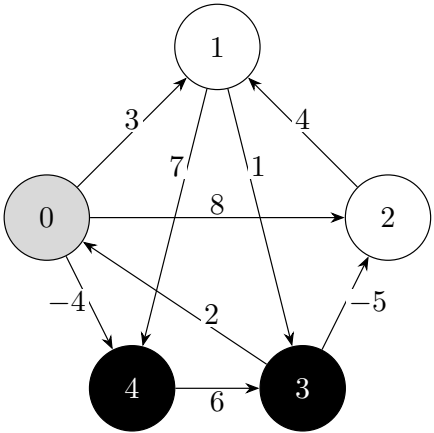
$k = 0$: $d[3][1]$ atualizado para 5 via vértice 0



$$\begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \mathbf{5} & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

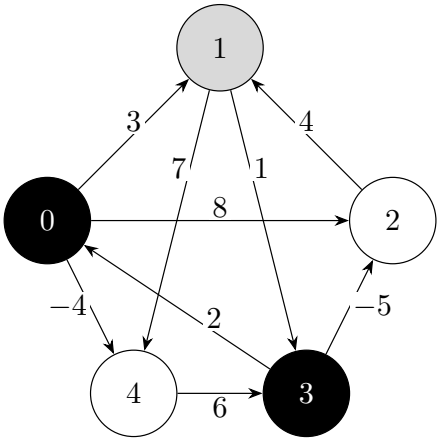
$k = 0$: $d[3][4]$ atualizado para -2 via vértice 0



$$\begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & \mathbf{-2} \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

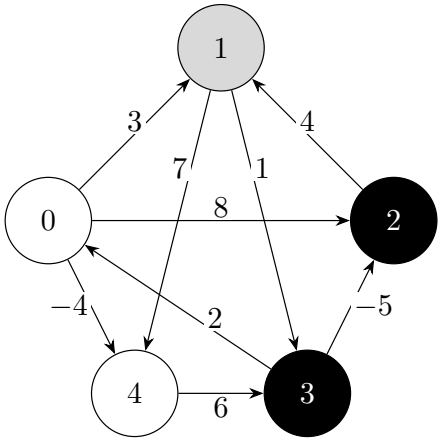
$k = 1$: $d[0][3]$ atualizado para 4 via vértice 1



$$\begin{bmatrix} 0 & 3 & 8 & \mathbf{4} & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

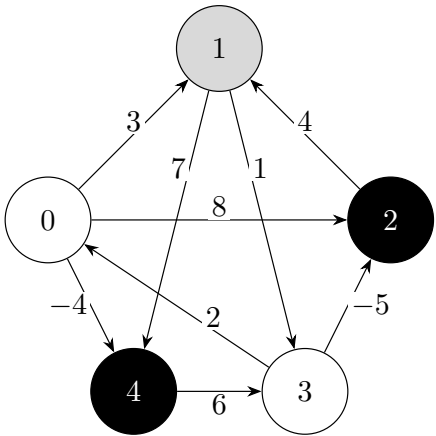
$k = 1$: $d[2][3]$ atualizado para 5 via vértice 1



$$\begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \mathbf{5} & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

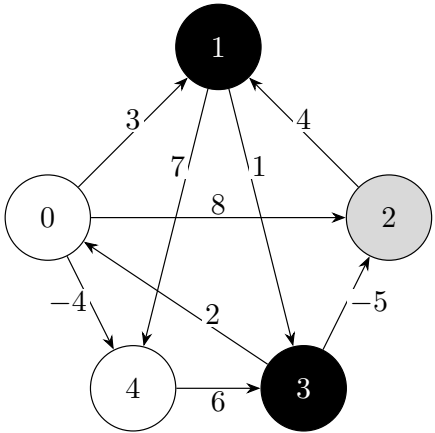
$k = 1$: $d[2][4]$ atualizado para 11 via vértice 1



$$\begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & \mathbf{11} \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

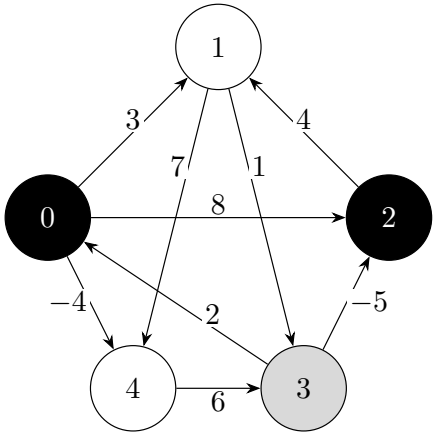
$k = 2$: $d[3][1]$ atualizado para -1 via vértice 2



$$\begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & \mathbf{-1} & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

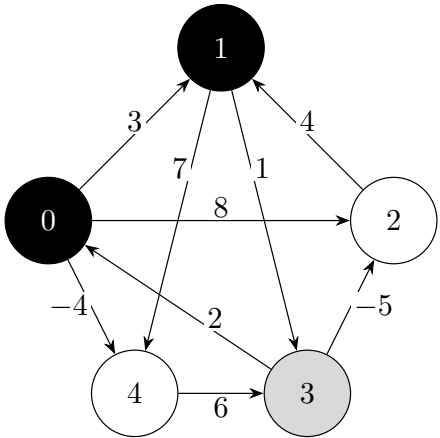
$k = 3$: $d[0][2]$ atualizado para -1 via vértice 3



$$\begin{bmatrix} 0 & 3 & \mathbf{-1} & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

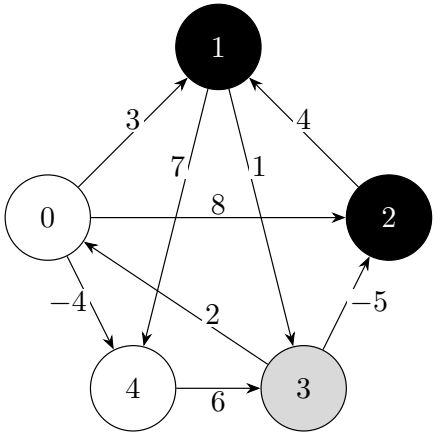
$k = 3$: $d[1][0]$ atualizado para 3 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ \mathbf{3} & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

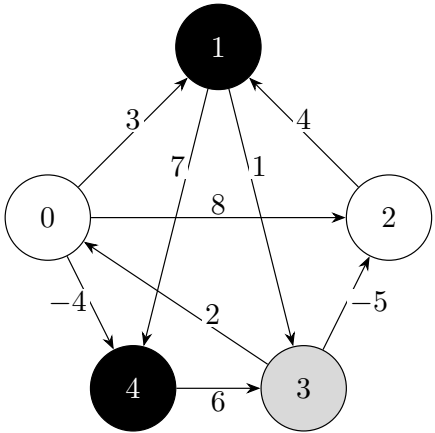
$k = 3$: $d[1][2]$ atualizado para -4 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & \mathbf{-4} & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

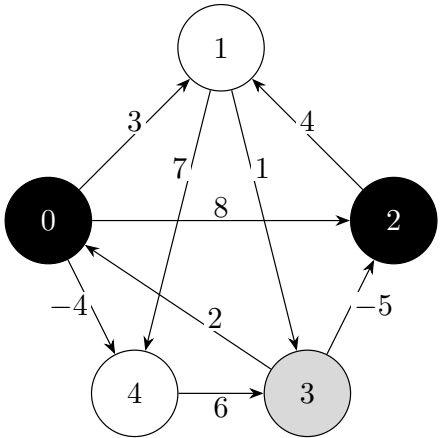
$k = 3$: $d[1][4]$ atualizado para -1 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & \mathbf{-1} \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

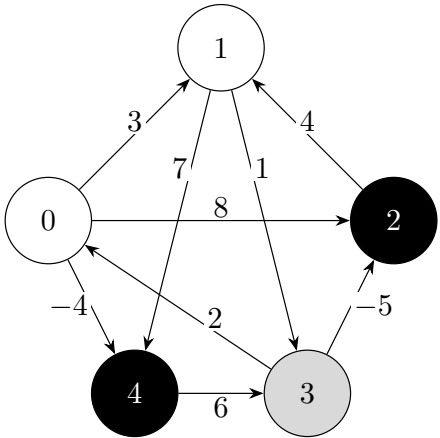
$k = 3$: $d[2][0]$ atualizado para 7 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ \mathbf{7} & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

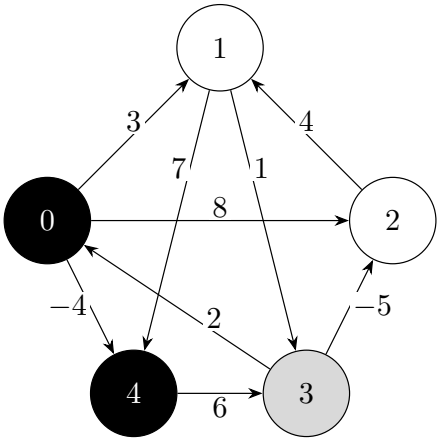
$k = 3$: $d[2][4]$ atualizado para 3 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & \mathbf{3} \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

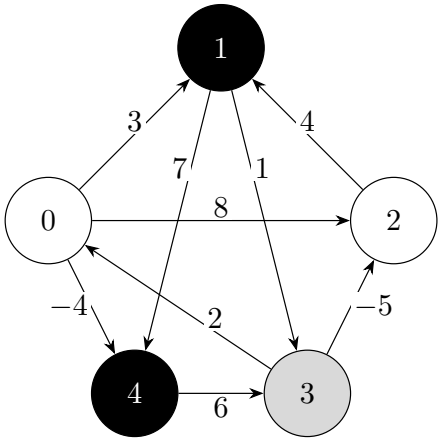
$k = 3$: $d[4][0]$ atualizado para 8 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ \mathbf{8} & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

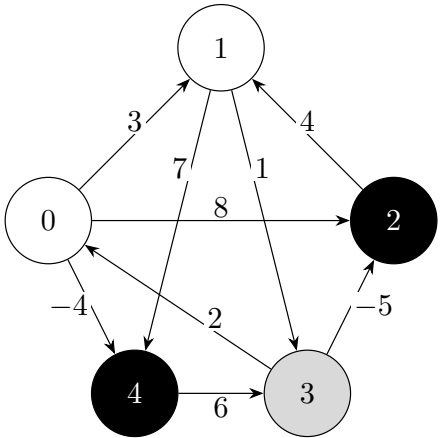
$k = 3$: $d[4][1]$ atualizado para 5 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \mathbf{5} & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

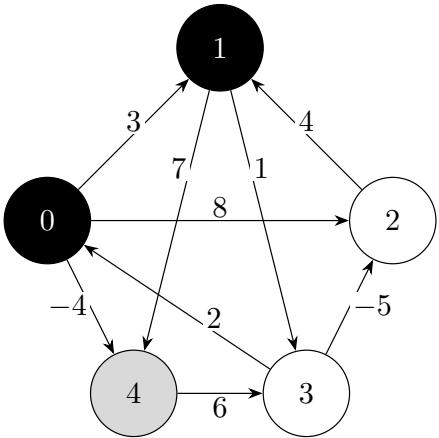
$k = 3$: $d[4][2]$ atualizado para 1 via vértice 3



$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & \mathbf{1} & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

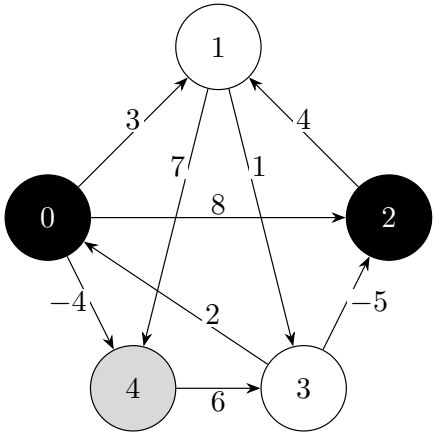
$k = 4$: $d[0][1]$ atualizado para 1 via vértice 4



$$\begin{bmatrix} 0 & \mathbf{1} & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

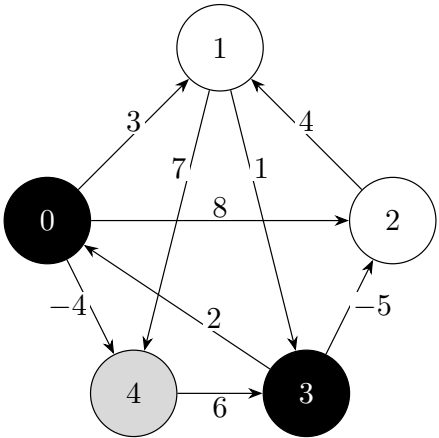
$k = 4$: $d[0][2]$ atualizado para -3 via vértice 4



$$\begin{bmatrix} 0 & 1 & \mathbf{-3} & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

Floyd-Warshall: exemplo

$k = 4$: $d[0][3]$ atualizado para 2 via vértice 4



$$\begin{bmatrix} 0 & 1 & -3 & \mathbf{2} & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$