Sumário

- Caminhos mais curtos entre todos os pares
  - Algoritmo Recursivo Simples
  - Algoritmo de Floyd-Warshall
  - Algoritmo de Johnson
    - Repesagem dos arcos

Aula 8

Definição [Problemas de Caminhos Mais Curtos]

- Caminhos mais curtos de origem única

  Dado um vértice s determinar para todo o vértice $v \in V$ o caminho p tal que: $s \overset{p}{\leadsto} v$ e $\delta(s, v) = w(p)$.

- Caminhos mais curtos de origem única e fonte única

  Dados dois vértices s e u, determinar o caminho p tal que: $s \overset{p}{\leadsto} u$ e $\delta(s, u) = w(p)$.

- Caminhos mais curtos entre todos os pares

  Para todos os vértices $u, v \in V$, determinar o caminho p tal que: $u \overset{p}{\leadsto} v$ e $\delta(u, v) = w(p)$.

Hoje

Objectivo

- Dado um grafo pesado $G = (V, E, w)$ calcular:
  - Matriz $\Pi_{(ij)}$: predecessor do nó j no caminho mais curto entre i e j
  - Matriz $D_{(ij)}$: peso do caminho mais curto entre i e j

Algoritmo Recursivo

- $l_{(ij)}^{(m)} \Rightarrow$ peso do caminho mais curto entre i e j de tamanho m

$$l_{(ij)}^{(0)} = \begin{cases} 0 & \text{se } i = j \\ \infty & \text{c.c.} \end{cases} \qquad l_{(ij)}^{(m)} = \min \left\{ l_{(ik)}^{(m)} + w(k,j) \mid 1 \leq k \leq n \right\}$$

Algorithmo Recursivo

- $l_{(ij)}^{(m)} \Rightarrow$ peso do caminho mais curto entre i e j de tamanho m → nº de arcos

$$l_{(ij)}^{(0)} = \begin{cases} 0 & \text{se } i = j \\ \infty & \text{c.c.} \end{cases}$$

$$l_{(ij)}^{(m)} = \min \left\{ l_{(ik)}^{(m)} + W(k,j) \mid 1 \leq k \leq n \right\}$$

```
UpdateAllPairsShortest(G, D)
    let D' be a new matrix with the dimensions of D
    for i = 1 to n
        for j = 1 to n
            for k = 1 to n
                if D'[i,j] > D[i,k] + W[i,j]
                    D'[i,j] := D[i,k] + W[i,j]
    Return D'
```

Complexidade:

$O(V^3)$

Algorithmo Recursivo

```
UpdateAllPairsShortestPaths(G, D)
  let D' be a new matrix with the dimensions of D
    for i = 1 to n
      for j = 1 to n
        for k = 1 to n
          if D'[i,j] > D[i,k] + W[i,j]
             D'[i,j] := D[i,k] + W[i,j]
    return D'
```

Complexidade:

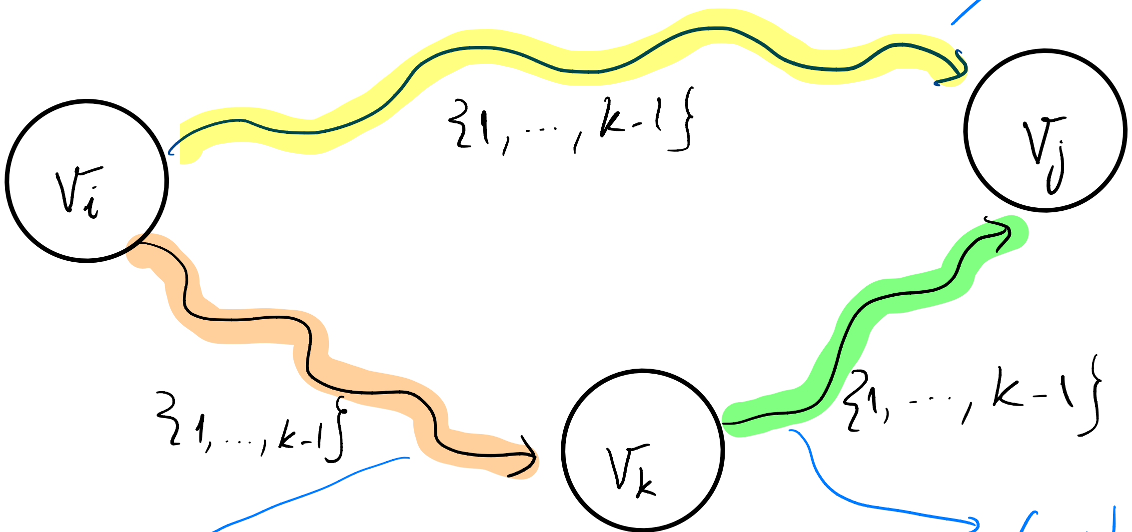
$O(V^3)$

```
ComputeAllPairsShortestPaths(G)
  let D^(1) := G.W
    for i := 2 to |G.V| - 1
      D^(i) := UpdateAllPairsShortestPaths(G, D^(i))
    return D^(n-1)
```

Complexidade

$O(V^4)$

Algoritmo de Floyd-Warshall

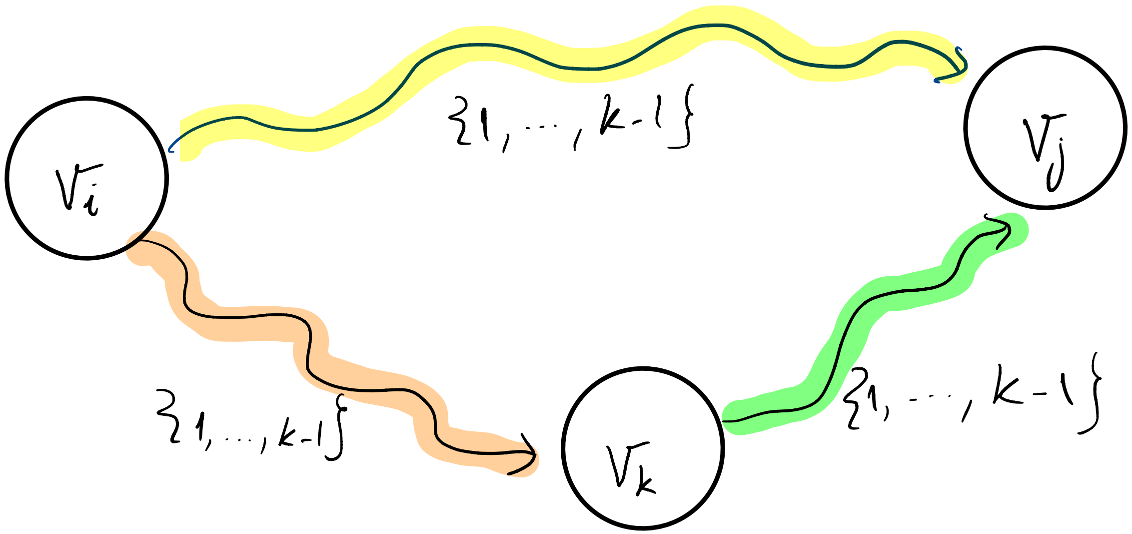


Caminho mais curto entre v_i e v_j que passa unicamente por vértices em $\{v_1, \dots, v_{k-1}\}$

Caminho mais curto entre v_i e v_k que passa unicamente por vértices em $\{v_1, \dots, v_{k-1}\}$

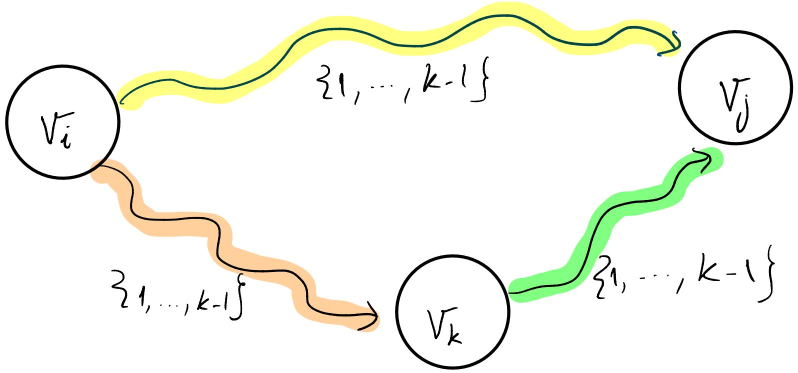
Caminho mais curto entre v_k e v_j q passa unicamente por vértices em $\{v_1, \dots, v_{k-1}\}$

Algoritmo de Floyd-Warshall



$$d_{ij}^{(k)} = \begin{cases} \min\left(d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{se } k > 0 \\ w_{ij} & \text{se } k = 0 \end{cases}$$

Algoritmo de Floyd-Warshall




```
Floyd Warshall (G)
  let n = |G.V|
  let D^(0) = G.W
  for k=1 to n
    let D^(k) be a new matrix with the same dimensions as D
    for i=1 to n
      for j=1 to n
        D^(k)[i,j] := min (D^(k-1)[i,j], D^(k-1)[i,k] + D^(k-1)[k,j])
```

$$d_{ij}^{(k)} = \begin{cases} \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{se } k > 0 \\ w_{ij} & \text{se } k = 0 \end{cases}$$

Complexidade: $\underline{\underline{O(|V|^3)}}$

Algoritmo de Floyd-Warshall

```
FloydWarshall(G)
  let n = |G.V|
  let D^(0) = G.W
  let Π = InitPi(W)
  for k=1 to n
    let D^(k) be a new matrix with the same dimensions as D
    for i=1 to n
      for j=1 to n
        d := min(D^(k-1)[i,j], D^(k-1)[i,k] + D^(k-1)[k,j])
        if d < D^(k-1)[i,j]
          D^(k)[i,j] := d
          Π[i,j] := Π[k,j]
```

Cálculo dos Predecessores

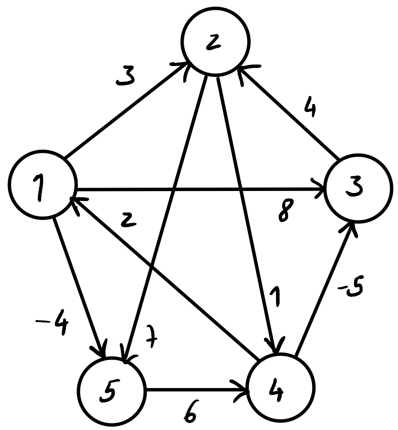
$$\Pi_{ij}^{(k)} = \begin{cases} \Pi_{ij}^{(k-1)} & \text{se } d_{ij}^{(k)} = d_{ij}^{(k-1)} \\ \Pi_{kj}^{k-1} & \text{se } d_{ij}^{(k)} = d_{kj}^{(k-1)} \end{cases}$$

$$\Pi_{ij}^{(0)} = \begin{cases} i & \text{se } (i,j) \in E \\ Nil & \text{c.c.} \end{cases}$$

Floyd-Warshall: Exemplo

$$d_{ij}^{(k)} = \min\left(d_{ik}^{(k-1)} + d_{kj}^{(k-1)},\ d_{ij}^{(k)} \right)$$

$$D^{(0)} = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$

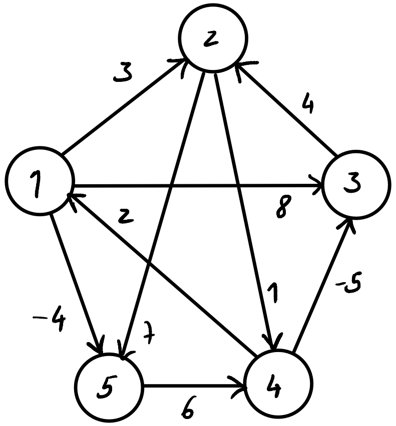


$$D^{(1)} = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$

$$D^{(2)} = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$

Floyd-Warshall: Exemplo

$$d_{ij}^{(k)} = \min\left(d_{ik}^{(k-1)} + d_{kj}^{(k-1)},\ d_{ij}^{(k)} \right)$$



$$D^{(2)} = \begin{array}{cc} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$

$$D^{(3)} = \begin{array}{cc} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$

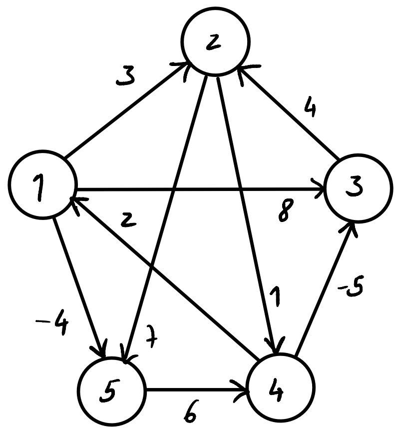
$$D^{(4)} = \begin{array}{cc} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix} \end{array}$$

$$D^{(5)} = \begin{array}{cc} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix} \end{array}$$

Floyd - Warshall: Exemplo

$$d_{ij}^{(k)} = \min\left(d_{ik}^{(k-1)} + d_{kj}^{(k-1)},\ d_{ij}^{(k)} \right)$$

$$D^{(0)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$



Fecho Transitivo

Definição [Fecho Transitivo]
O fecho transitivo de um grafo $G=(V,E)$ é definido como o grafo $G^*=(V,E^*)$ onde $E^* = \{(u,v) \mid u \leadsto v\}$

Calcular o fecho transitivo

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)} \right) \quad \text{se } k>0$$

$$t_{ij}^{(0)} = \begin{cases} true & \text{se } i=j \vee (i,j) \in E \\ false & \text{c.c.} \end{cases}$$

Fecho Transitivo

Definição [Fecho Transitivo]
O fecho transitivo de um grafo $G = (V, E)$ é definido como o grafo $G^* = (V, E^*)$
onde $E^* = \{ (u, v) \mid u \rightsquigarrow v \}$

Calcular o fecho transitivo

```
Transitive Closure (G)
  let T^(0) = initMatrix (G)
  let n = |G.V|
  for k=1 to n
    let T^(k) be a new matrix with the dimensions of T
    for i=1 to n
      for j=1 to n
        T^(k)[i,j] = T^(k-1)[i,j] ∨ (T^(k-1)[i,k] ∧ T^(k-1)[k,j])
  return T^(n)
```

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)} \right) \quad \text{se } k > 0$$

$$t_{ij}^{(0)} = \begin{cases} \text{true} & \text{se } i = j \vee (i,j) \in E \\ \text{false} & \text{c.c.} \end{cases}$$

Complexidade: $O(V^3)$

Algoritmo de Johnson

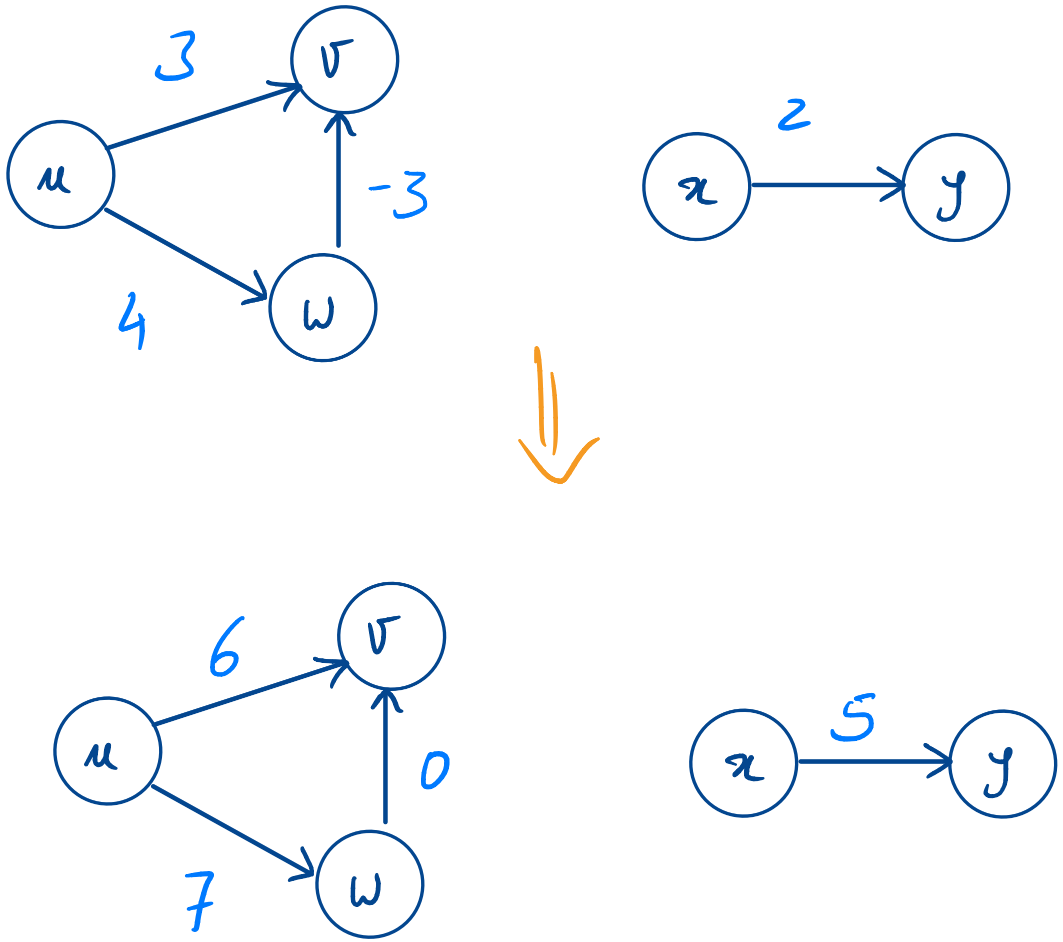
Ideia

- Dado um grafo G com pesos negativos, calcular um grafo G' cujos caminhos mais curtos coincidem com os de G mas **sem** arcos com pesos negativos

Repesagem dos Arcos

- Aplicar o algoritmo de Dijkstra a todos os vértices de G'

Repesagem dos Arcos - 1ª Ideia

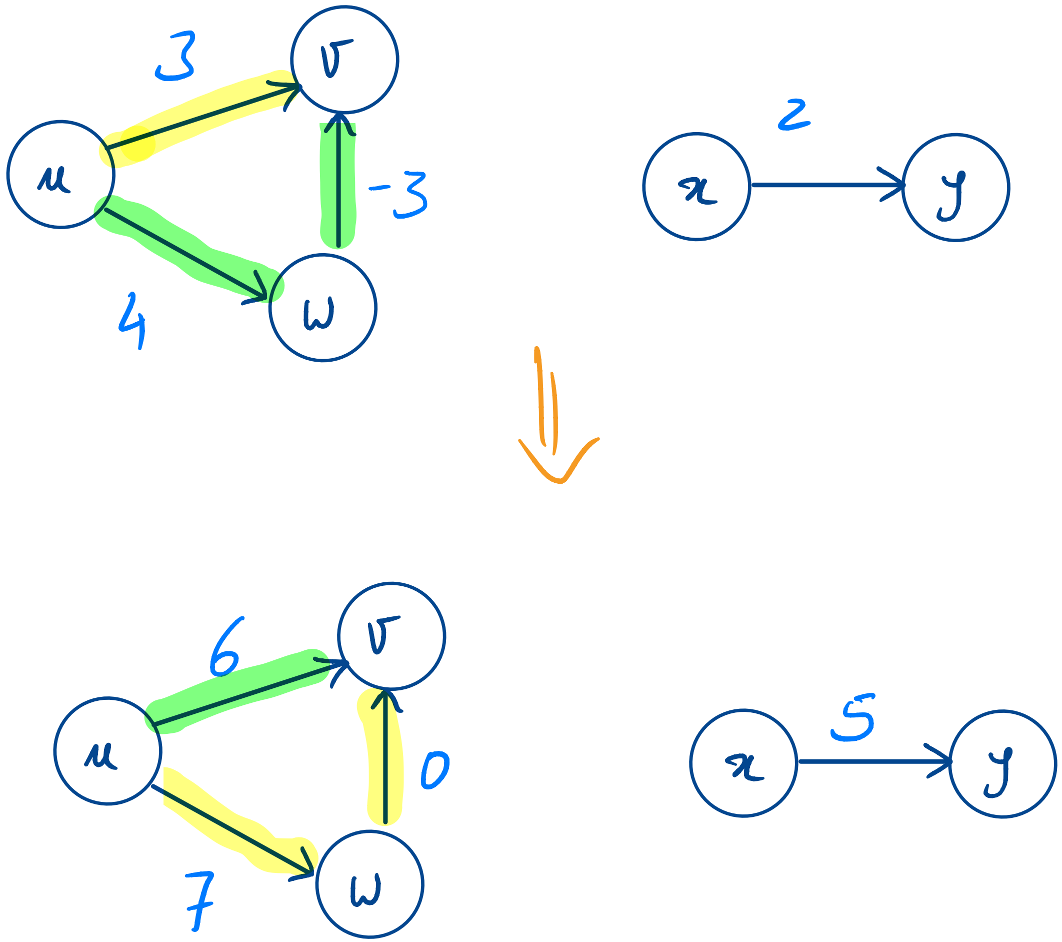


Ideia Naïf:

- Somar o módulo do maior dos pesos negativos a todos os arcos

Funcionou?

Repesagem dos Arcos - 1ª Ideia



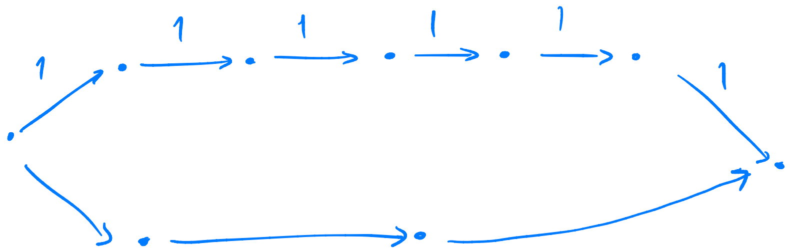
Ideia Naïf:

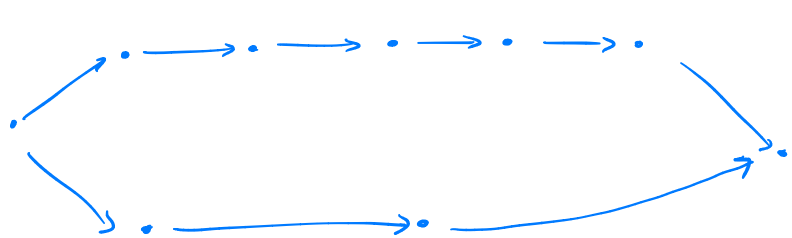
- Somar o módulo do maior dos pesos negativos a todos os arcos

Funciona?

Não! Este método penaliza caminhos maiores!

Repesagem dos Arcos - 1ª Ideia





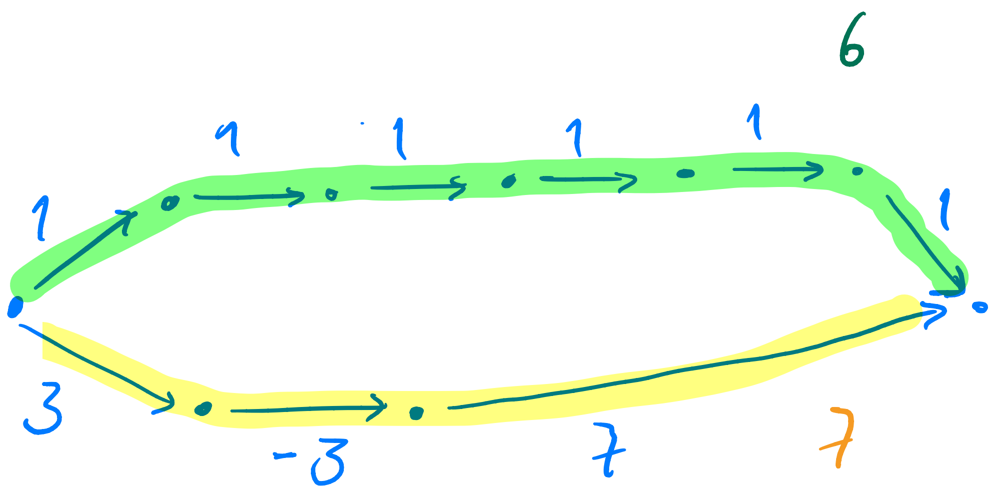
Ideia Naïf:

- Somar o módulo do maior dos pesos negativos a todos os arcos

Funcionou?

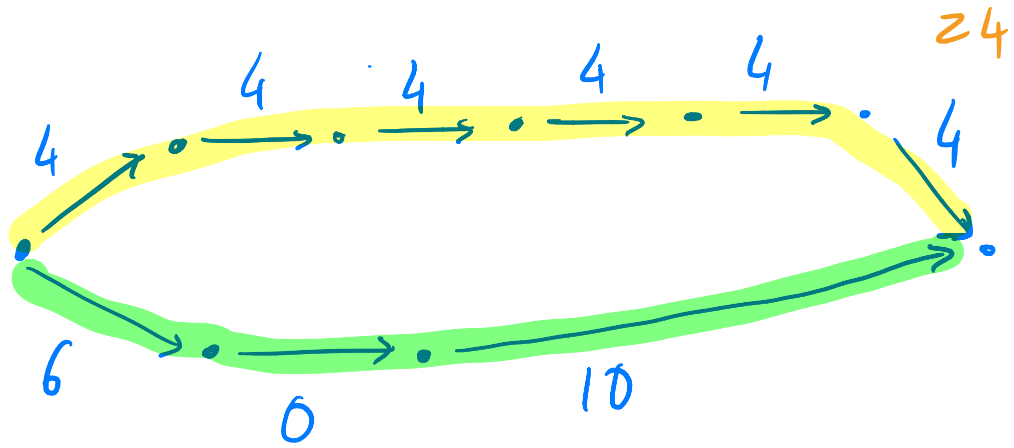
Não! Este método penaliza caminhos maiores!

Repesagem dos Arcos - 1ª Ideia



Ideia Naïf:

- Somar o módulo do maior dos pesos negativos a todos os arcos



Funcionou?

Não! Este método penaliza caminhos maiores!

Repesagem de Johnson

- Encontrar uma função de alturas $h : V \to \mathbb{R}$:

$$G = (V, E, w)$$

$$\Downarrow$$

$$G = (V, E, \hat{w}) \qquad \text{onde: } \hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

- Quem é h?

Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

$$\cdot\ \bar{E} = E \cup \{(s,v) \mid v \in V\} \qquad \underline{\underline{h(v)}} = \delta(s,v)$$

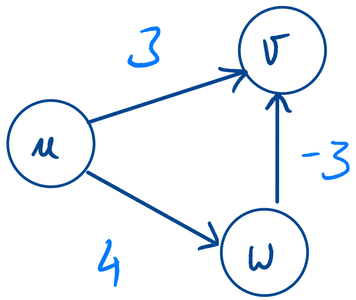
$$\cdot\ \bar{w}(u,v) = \begin{cases} w(u,v) & \text{se } (u,v) \in E \\ 0 & \text{se } u = s \wedge v \neq s \end{cases}$$

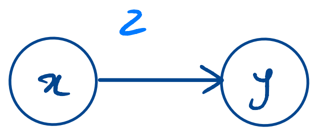
Repesagem de Johnson

- Encontrar uma função de alturas $h : V \to \mathbb{R}$:

  $G = (V, E, w) \Rightarrow G = (V, E, \hat{w})$

  onde: $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$





- Quem é h?

  Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

  · $\bar{E} = E \cup \{(s,v) \mid v \in V\}$ $\qquad \underline{\underline{h(v)}} = \delta(s,v)$

Repesagem de Johnson

- Encontrar uma função de alturas $h : V \to \mathbb{R}$:

  $G = (V, E, w) \Rightarrow G = (V, E, \hat{w})$

  onde: $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$

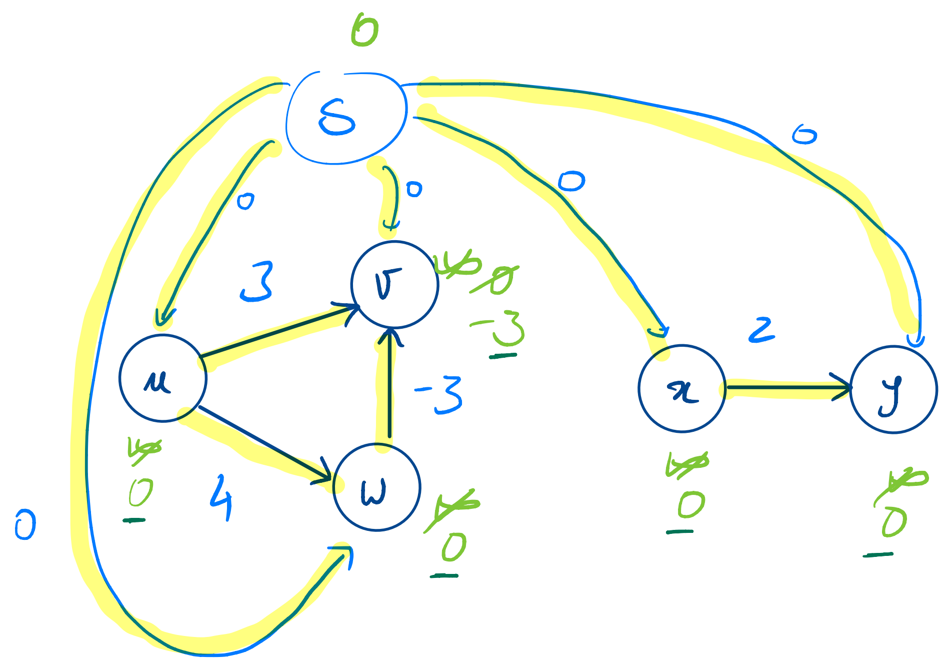
- Quem é h?

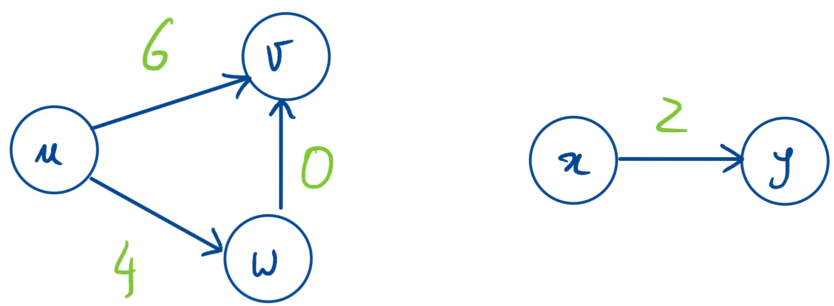
  Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

  - $\bar{E} = E \cup \{(s, v) \mid v \in V\}$

- Alg. de Johnson:

  $h(v) = \delta_{\bar{G}}(s, v)$





Repesagem de Johnson - Connecção

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

1.

Repesagem de Johnson - Correcção

① Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

② Se p é um caminho mais curto em G
então p é um caminho mais curto em $\hat{G}$

③ Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

①
$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$
$$= w(u,v) + \delta(s,u) - \delta(s,v) \quad \Big| \text{ Desigualdade Triangular}$$
$$\geq 0$$

Repesagem de Johnson – Correcção

(1) Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

(2) Se p é um caminho mais curto em G
então p é um caminho mais curto em $\hat{G}$

(3) Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

(2)

Repesagem de Johnson – Correcção

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

2. Seja $p = \langle v_1, \dots, v_n \rangle$ um caminho mais curto em G.

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

$$= \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + \sum_{i=1}^{n-1} \left(h(v_i) - h(v_{i+1}) \right)$$

$$= w(p) + h(v_1) - h(v_n)$$

Suponhamos, por contradição, que p é o caminho mais curto em G, mas existe p' mais curto que p em $\hat{G}$:

$$\hat{w}(p') < \hat{w}(p)$$

Sabemos que:

$$\hat{w}(p) = w(p) + h(v_0) - h(v_n)$$

$$\hat{w}(p') = w(p') + h(v_0) - h(v_n)$$

De onde concluímos que:

$$w(p') < w(p) \quad \lightning$$

Repesagem de Johnson – Correcção

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G
   então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

Repesagem de Johnson – Correcção

① Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

② Se p é um caminho mais curto em G
então p é um caminho mais curto em $\hat{G}$

③ Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

Seja $p = \langle v_0, \ldots, v_n \rangle$ com $v_n = v_0$ um ciclo em G

$$\hat{w}(p) = w(p) + h(v_0) - h(v_n)$$
$$= w(p)$$

Algoritmo de Johnson

[Complexidade]

- Calcular os caminhos curtos entre todos os pares em $G = (V, E, w)$

① Calcular o grafo estendido $\bar{G}_s = (\bar{V}, \bar{E}, \bar{w})$ com $s \notin V$

② Usar o algoritmo de Bellman-Ford p/ determinar a função de altura h

Se o algoritmo de Bellman-Ford retornar falso, o algoritmo de Johnson tb retorna falso.

③ Calcular o grafo repesado $\hat{G} = (V, \hat{E}, \hat{w})$

④ Para cada vértice $u \in V$, usar o algoritmo de Dijkstra

D_{uv} e Π_{uv} para todo $v \in V$

⑤ Retornar D e Π.

Algoritmo de Johnson

- Calcular os caminhos curtos entre todos os pares em $G = (V, E, w)$

① Calcular o grafo estendido $\bar{G}_s = (\bar{V}, \bar{E}, \bar{w})$ com $s \notin V$ $O(V+E)$

[Complexidade]

$O(V E \lg V)$

② Usar o algoritmo de Bellman-Ford p/ determinar a função de altura h
Se o algoritmo de Bellman-Ford retornar falso, o algoritmo de Johnson tb retorna falso. $\| O(E \cdot V)$

③ Calcular o grafo repesado $\hat{G} = (V, \hat{E}, \hat{w})$ $\| O(V+E)$

④ Para cada vértice $u \in V$, usar o algoritmo de Dijkstra
D_{uv} e Π_{uv} para todo $v \in V$ $\| O(E \lg V)$ $\| O(V \cdot E \cdot \lg V)$

⑤ Retornar D e Π.