Algoritmo de Bellman-Ford

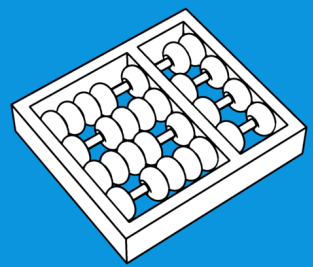
MC558 - Projeto e Análise de Algoritmos II

09/24

13

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

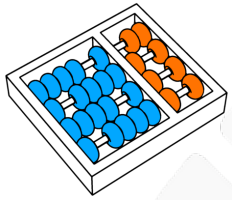
UNICAMP



“A menor distância entre pontos é uma linha reta.”

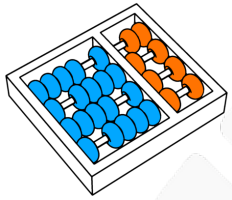
Arquimedes

Aulas anteriores



Representando caminhos mínimos

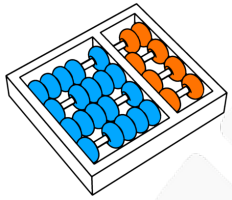
- Para cada **v** $\in$ **V[G]**, associamos um **PREDECESSOR** π[**v**].
- O vetor π induz uma **ÁRVORE DE CAMINHOS MÍNIMOS** com raiz em **s**.
- Um caminho de **s** a **v** na árvore é um caminho mínimo de **s** a **v** no grafo G.



Inicialização

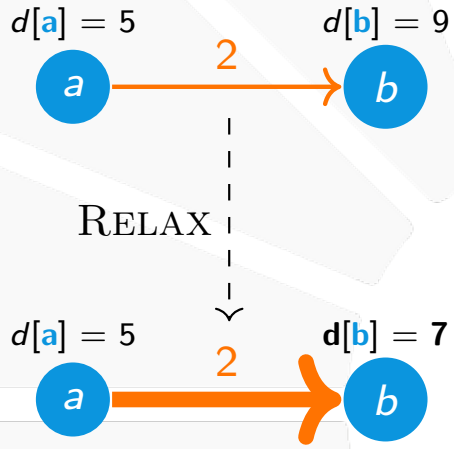
Algoritmo: INITIALIZE-SINGLE-SOURCE(G, s)

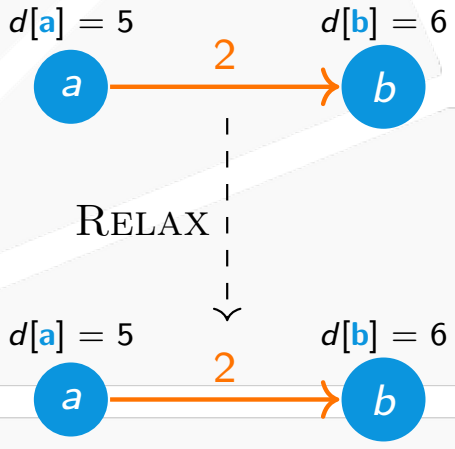
```
1 para cada v ∈ V[G]
2     d[v] ← ∞
3     π[v] ← NIL
4 d[s] ← 0
```



Relaxação

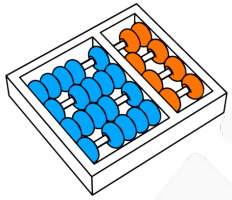
Tenta melhorar a estimativa $d[\mathbf{v}]$ examinando **(u,v)**.





Algoritmo: $\textsc{Relax}(u, v, w)$

```
1 se d[v] > d[u] + w(u, v)
2     d[v] ← d[u] + w(u, v)
3     π[v] ← u
```



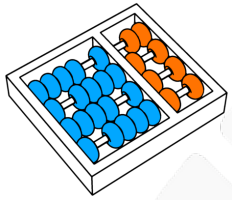
Algoritmos baseados em relaxação

Características:

1. Inicializam d e π com uma sub-rotina INITIALIZE-SINGLE-SOURCE.
2. Alteram d e π apenas com uma sub-rotina RELAX.

Esses algoritmos mantêm algumas invariantes:

- Existe um caminho de **s** a **v** com peso d[**v**].
- Esse caminho pode ser recuperado por meio π.
- Assim, d[**v**] é sempre **MAIOR OU IGUAL** a dist(**s**,**v**).
- Queremos que no final valha d[**v**] = dist(**s**,**v**).

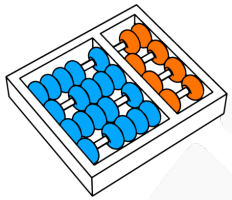


Propriedade de algoritmos baseados em relaxação

- **Limite superior:** Vale $d[v] \geq \text{dist}(s,v)$ e, tão logo $d[v]$ alcança $\text{dist}(s,v)$, nunca mais muda.
- **Inexistência de caminho:** Se não existe caminho de s a v, então $d[v] = \infty$.
- **Subgrafo de predecessores:** Se $d[v] < \infty$, então o subgrafo dos predecessores induzido por π é um caminho de peso $d[v]$.
- **Convergência:** Se p é um caminho mínimo de s até v terminando com a aresta (u,v) e $d[u] = \text{dist}(s,u)$, então ao relaxar (u,v), $d[v] = \text{dist}(s,v)$, que nunca mais muda.
- **Relaxamento de caminho:** Se $p = (v_0, v_1, \ldots, v_k)$ é um caminho mínimo de $s = v_0$ a v_k e relaxamos as arestas de p na ordem $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, então $d[v_k] = \text{dist}(s, v_k)$.

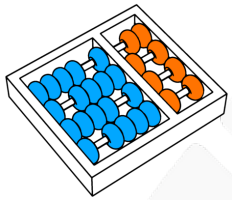
  A propriedade vale mesmo se tivermos realizado quaisquer outras relaxações durante a execução.

Algoritmo de Bellman-Ford



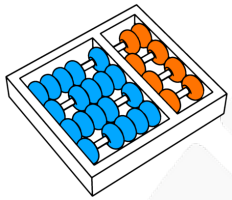
Arestas vs ciclos de peso negativo

- O algoritmo de Dijkstra resolve o Problema dos Caminhos Mínimos quando (G, w) **NÃO TEM ARESTAS DE PESO NEGATIVO**.
- Quando (G, w) possui arestas negativas, o algoritmo de Dijkstra não funciona.
- Uma das dificuldades com arestas negativas é a possível existência de **CICLOS DE PESO NEGATIVO** ou simplesmente ciclos negativos.



Ciclos negativos — uma dificuldade

- O Problema dos Caminhos Mínimos para instâncias com ciclos negativos é NP-**DIFÍCIL**.
  - Acreditamos que **NÃO** existem algoritmos eficientes para resolver problemas NP-difíceis.
  - Assim, vamos nos restringir ao Problema de Caminhos Mínimos **SEM CICLOS NEGATIVOS**.
- Um algoritmo que resolve o problema restrito é o algoritmo de Bellman-Ford, que também é baseado em relaxação.

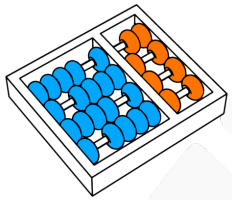


Definição do problema

Problema

Entrada: *Um grafo direcionado* $G = (V, E)$, *uma função de peso* w *nas arestas e um vértice origem* s.

Saída: FALSE, *se existe um ciclo negativo atingível a partir de* s. *Caso contrário, além de* TRUE, *também devolve um vetor* d *com* $d[v] = \text{dist}(s,v)$ *para* $v \in V$ *e um vetor* π *definindo uma* ***ÁRVORE DE CAMINHOS MÍNIMOS***.



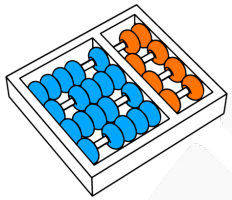
Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **QUALQUER** caminho mínimo $(v_0, v_1, \ldots, v_k)$, queremos relaxar $(v_0, v_1),\ (v_1, v_2),\ \ldots,\ (v_{k-1}, v_k)$ em ordem.

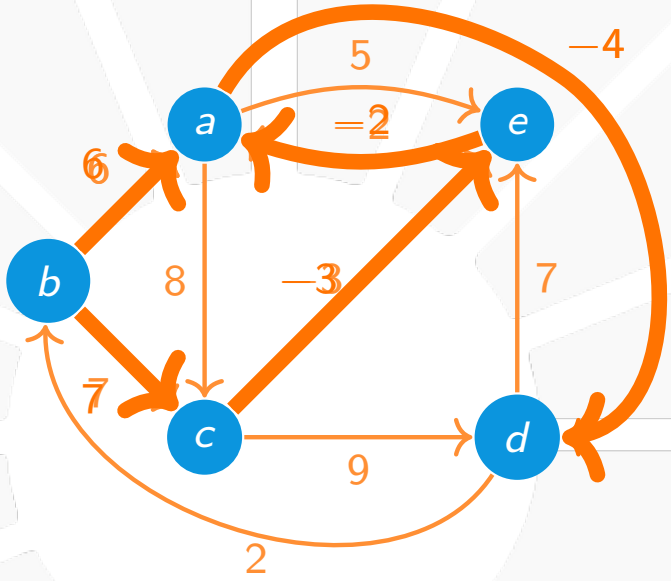
1. Executamos RELAX para todas as arestas:
   $\Rightarrow (v_0, v_1)$ relaxada.
2. Executamos novamente RELAX para todas as arestas:
   $\Rightarrow (v_0, v_1),\ (v_1, v_2)$ relaxadas em ordem.
3. Executamos novamente RELAX para todas as arestas:
   $\Rightarrow (v_0, v_1),\ (v_1, v_2),\ (v_2, v_3)$ relaxadas em ordem.
4. Repetimos esse passo até $|V| - 1$ vezes. Por quê?
   $\Rightarrow (v_0, v_1),\ (v_1, v_2),\ \ldots,\ (v_{k-1}, v_k)$ relaxadas em ordem.

Podemos verificar se o grafo contém **CICLOS NEGATIVOS** executando mais uma vez:

- Se algum valor $d[v]$ diminuir, então há ciclo negativo.

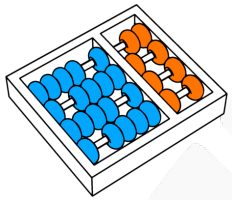


Exemplo



	a	b	c	d	e
d	~~∞~~ 8	0	~~∞~~ 7	~~∞~~ ~~2~~ 2	~~∞~~ 4

Ordem: **(a,c),(a,d),(a,e),(c,d),(c,e),(d,b),(d,e),(e,a),(b,a),(b,c)**.

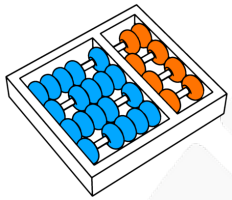


O algoritmo de Bellman-Ford

Algoritmo: BELLMAN-FORD(G, w, s)

```
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 para  i ← 1 até |V[G]| − 1
3 |   para cada (u, v) ∈ E[G]
4 |   |   RELAX(u, v, w)
5 para cada (u, v) ∈ E[G]
6 |   se d[v] > d[u] + w(u, v)
7 |   |   devolva FALSE
8 devolva TRUE, d, π
```

Complexidade de tempo: $O(VE)$

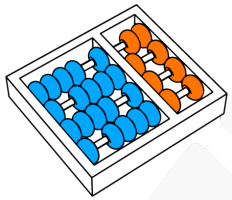


Correção do algoritmo Bellman-Ford

Teorema

BELLMAN-FORD *devolve:*

- FALSE, *se existe um ciclo negativo atingível a partir de* ***s***.
- TRUE, *caso contrário; neste caso devolve também:*
  - *Um vetor* d *com* $d[\boldsymbol{v}] = \text{dist}(\boldsymbol{s},\boldsymbol{v})$ *para* $\boldsymbol{v} \in \boldsymbol{V}$.
  - *Um vetor* π *definindo uma* ***ÁRVORE DE CAMINHOS MÍNIMOS***.

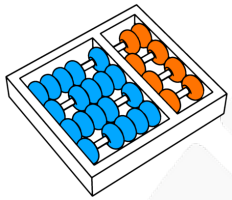


Correção do algoritmo Bellman-Ford

Primeiro, suponha que não há ciclos negativos atingíveis por **s**.

Considere $\mathbf{v} \in \mathbf{V[G]}$ e os valores de d e π após o primeiro laço:

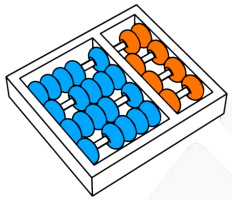
- Se **v** não é atingível, $d[\mathbf{v}] = \infty$ (por **Inexistência de caminho**).
- Senão, existe caminho mínimo $(\mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_k})$ de $\mathbf{s} = \mathbf{v_0}$ a $\mathbf{v} = \mathbf{v_k}$.
- Como $k \leq |V| - 1$, então $(\mathbf{v_0}, \mathbf{v_1}),\ (\mathbf{v_1}, \mathbf{v_2}),\ \ldots,\ (\mathbf{v_{k-1}}, \mathbf{v_k})$ foram relaxadas **NESTA ORDEM**.
- Por **Relaxamento de caminho**, $d[\mathbf{v}] = \text{dist}(\mathbf{s},\mathbf{v})$.
- Também, por **Sugrafo de predecessores**: π induz um caminho mínimo de **s** a **v**.



Correção do algoritmo Bellman-Ford

Também temos que mostrar que nesse caso BELLMAN-FORD devolve TRUE.

- Considere d imediatamente após o primeiro laço.
- Nesse instante, $d[\mathbf{v}] = \text{dist}(\mathbf{s},\mathbf{v})$ para todo vértice **v**.
- Por **Convergência**, sabemos que d nunca mais muda.
- Portanto, o teste da linha 6 falha sempre.
- Concluindo que o algoritmo devolve TRUE.



Correção do algoritmo Bellman-Ford

Suponha agora que (G, w) contenha **CICLO NEGATIVO** alcançável por $\mathbf{s}$.

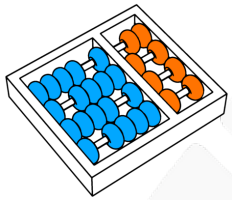
Queremos mostrar que o algoritmo devolve FALSE:

- Seja $C = (\mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_k} = \mathbf{v_0})$ um ciclo tal que

$$w(C) = \sum_{i=1}^{k} w(\mathbf{v_{i-1}}, \mathbf{v_i}) < 0.$$

- Suponha, por contradição que o algoritmo devolve TRUE.
- Como relaxamos cada aresta $(\mathbf{v_{i-1}}, \mathbf{v_i})$:

$$d[\mathbf{v_i}] \leq d[\mathbf{v_{i-1}}] + w(\mathbf{v_{i-1}}, \mathbf{v_i}).$$



Correção do algoritmo Bellman-Ford

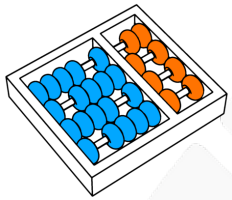
- Somando as desigualdades anteriores para cada aresta do ciclo, temos:

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} (d[v_{i-1}] + w(v_{i-1}, v_i))$$

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

- Como $v_0 = v_k$, temos que $\sum_{i=1}^{k} d[v_i] = \sum_{i=1}^{k} d[v_{i-1}]$.
- Logo, $0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C)$.
- Mas isso é uma contradição, pois C é ciclo negativo.
- Concluindo que, nesse caso, o algoritmo devolve FALSE.

Sistemas de restrições de diferenças



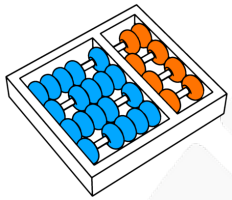
Exemplo

Uma aplicação de caminhos mínimos é encontrar $x_1, x_2, \ldots, x_n$ que satisfaçam:

$$\begin{array}{rcl} x_1 - x_2 & \leq & 0 \\ x_1 - x_5 & \leq & -1 \\ x_2 - x_5 & \leq & 1 \\ x_3 - x_1 & \leq & 5 \\ x_4 - x_1 & \leq & 4 \\ x_4 - x_3 & \leq & -1 \\ x_5 - x_3 & \leq & -3 \\ x_5 - x_4 & \leq & -3 \end{array}$$

Onde:

- x_i representa a hora do evento i
- $x_j - x_i \leq b_k$ significa que deve haver um intervalo de pelo menos b_k horas entre eventos i e j



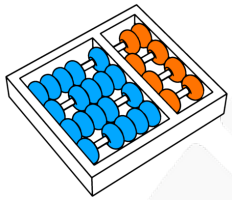
Exemplo

Podemos reescrever as restrições de forma matricial:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

Algumas soluções:

- $x = (-5, -3, 0, -1, -4)$,
- $x' = (0, 2, 5, 4, 1)$,...



Soluções de sistemas de restrições de diferenças

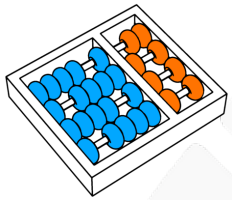
Lema

Seja $x = (x_1, \ldots, x_n)$ uma solução de um sistema de restrições de diferença $Ax \leq b$ e d uma constante. Então

$$x + d = (x_1 + d, \ldots, x_n + d)$$

também é uma solução de $Ax \leq b$.

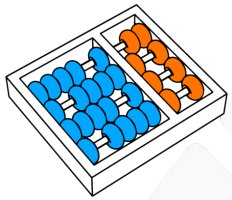
Mostraremos a seguir como encontrar uma solução de um sistema $Ax \leq b$ de restrições de diferença resolvendo um problema de caminhos mínimos.



Grafo de restrições

Construímos o chamado **GRAFO DE RESTRIÇÕES** fazendo o seguinte:

1. Primeiro criamos um grafo em que:
   - Cada vértice $\mathbf{v_i}$ corresponde a uma variável x_i.
   - Cada aresta $(\mathbf{v_i}, \mathbf{v_j})$ tem peso b_k e corresponde a uma restrição $x_j - x_i \leq b_k$.
   - Logo, a matriz A do sistema de restrições corresponde à transposta matriz de incidência desse grafo obtido
2. Finalmente, adicionamos um vértice especial $\mathbf{v_0}$ e uma aresta de $\mathbf{v_0}$ a cada outro vértice $\mathbf{v_i}$ com peso 0.



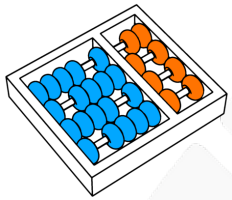
Grafo de restrições

Formalmente, dado o sistema $Ax \leq b$ de restrições de diferença, construímos o grafo $G = (\mathbf{V}, \mathbf{E})$ tal que:

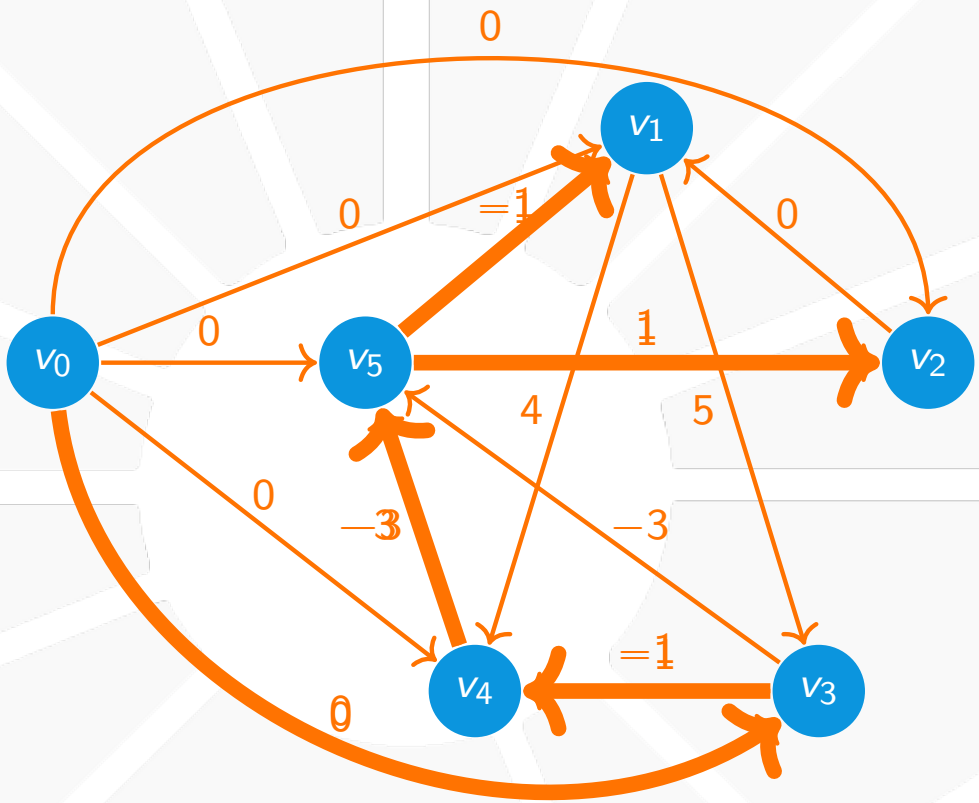
- $\mathbf{V} = \{\mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_n}\}$.
- $\mathbf{E} = \{(\mathbf{v_0}, \mathbf{v_1}), \ldots, (\mathbf{v_0}, \mathbf{v_n})\} \cup \{(\mathbf{v_i}, \mathbf{v_j}) : \mathbf{x_j} - \mathbf{x_i} \leq \mathbf{b_k}$ é restrição$\}$

E associamos pesos:

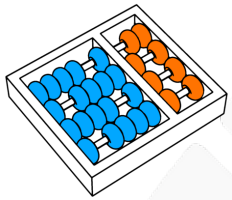
- $w(\mathbf{v_i}, \mathbf{v_j}) = \begin{cases} b_k & \text{se } x_j - x_i \leq b_k \text{ é restrição} \\ 0 & \text{se } i = 0 \end{cases}$



Grafo de restrições



Uma solução viável é $\mathbf{x} = (-\mathbf{5}, -\mathbf{3}, \mathbf{0}, -\mathbf{1}, -\mathbf{4})$.



Grafo de restrições

Teorema

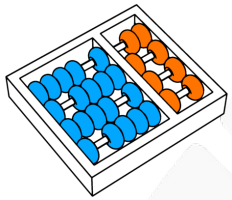
Seja $Ax \leq b$ um sistema de restrições de diferença e $G = (V, E)$ o grafo de restrições associado. Então:

- *Se G não contém ciclos negativos,*

$$x = (\text{dist}(v_0, v_1), \text{dist}(v_0, v_2), \ldots, \text{dist}(v_0, v_n))$$

  é uma solução viável do sistema.

- *Se G contém ciclos negativos, o sistema não possui solução. viável.*



Resolvendo um sistema de restrições de diferença

Para **RESOLVER O SISTEMA**, basta resolver caminhos mínimos:

- Executamo BELLMAN-FORD a partir de $\mathbf{v_0}$ no grafo de restrições G.
- Como todo vértice é atingível de $\mathbf{v_0}$, se houver ciclo negativo, o algoritmo devolve FALSE.
- Se não existir ciclo negativo, então obtemos a solução $x = (d[\mathbf{v_1}], d[\mathbf{v_2}], \ldots, d[\mathbf{v_n}])$.

O **TEMPO DE EXECUÇÃO** é calculado como segue:

- A matriz A tem dimensões $m \times n$.
- Então, G possui $n + 1$ vértices e $n + m$ arestas.
- Logo, o tempo de execução de BELLMAN-FORD em G é $O((n + 1)(n + m)) = O(n^2 + nm)$.

Algoritmo de Bellman-Ford

MC558 - Projeto e Análise de Algoritmos II

09/24

13

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP