# Universidade Federal de Alfenas

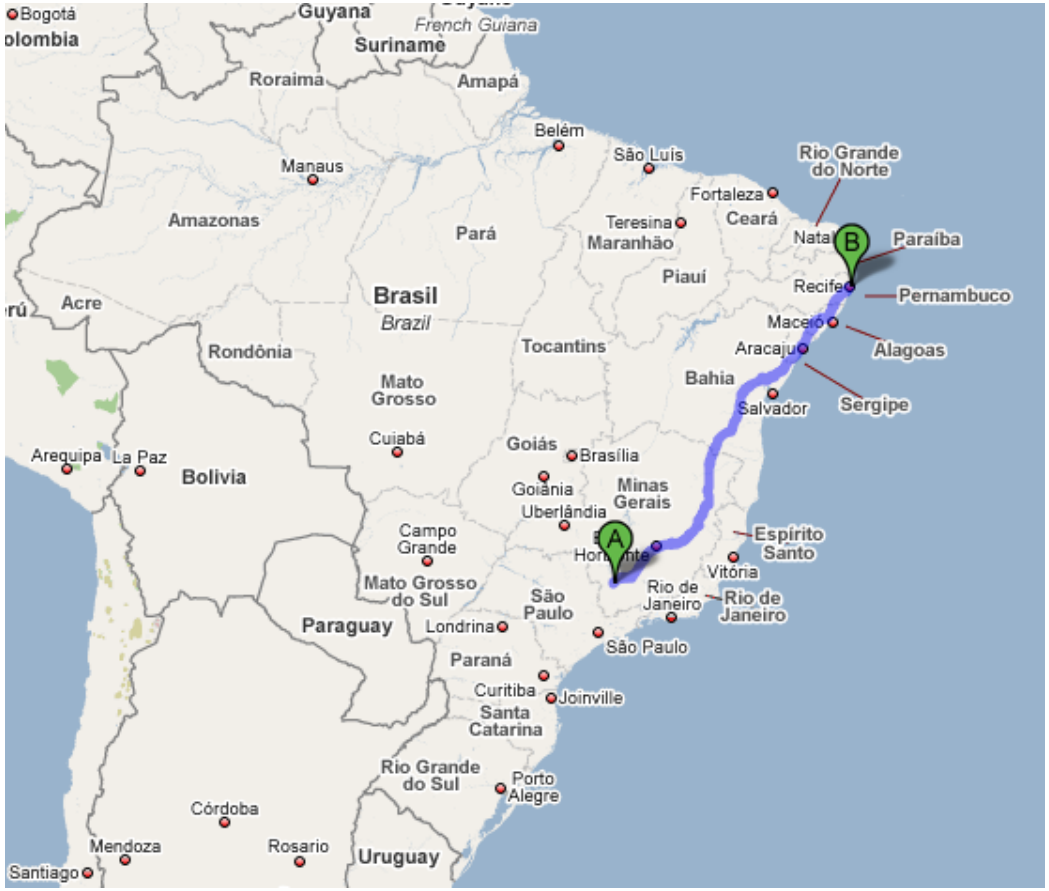
## Algoritmos em Grafos

Aula 12 – Caminho Mínimo: Algoritmo de Bellman-Ford

Prof. Humberto César Brandão de Oliveira
humberto@bcc.unifal-mg.edu.br

# Caminho Mínimo

- Suponha que você deseja encontrar um caminho mais curto de Alfenas/MG para Recife/PE;

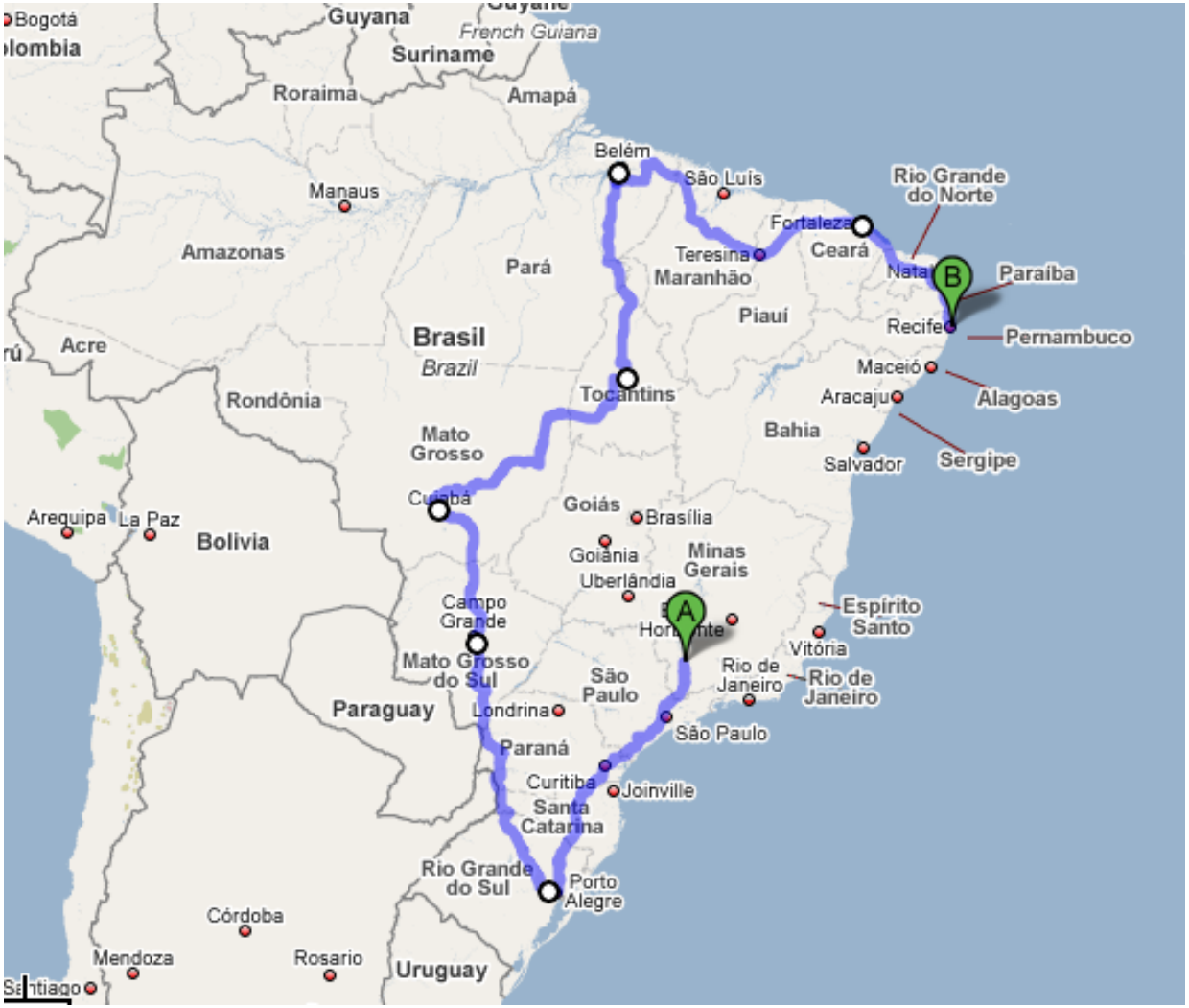


# Caminhos mais curtos de origem única

- A principio, podemos *imaginar* um algoritmo que *enumera* todas as possíveis rotas;
  - Se você conhece TODAS as possibilidades a tarefa é apenas computar a distancia de cada uma delas e escolher aquela que oferece o menor trajeto.
    - O problema é computável;
    - Pode ser resolvido por uma máquina de Turing ou máquina equivalente.
  - Obviamente, tomando o cuidado de não considerar trajetos com ciclos; Seria equivalente a processar um autômato com ciclo(s). A linguagem correspondente possui infinitas palavras (rotas) – apesar das palavras possuírem tamanho finito.

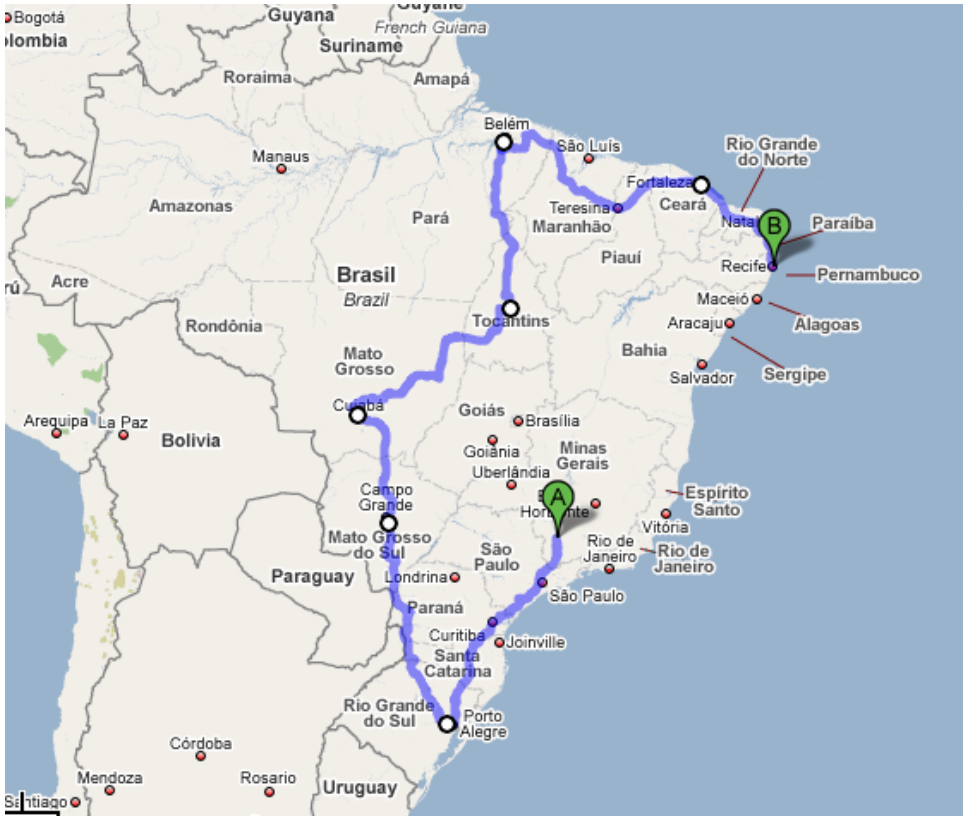
# Caminhos mais curtos de origem única

- O problema da abordagem da enumeração, é que haverá milhões/bilhões de possibilidades;
- Sendo que a maioria não vale a pena considerar...
- Veja  exemplo na figura...



# Caminhos mais curtos de origem única

- Podemos realizar obviamente uma enumeração implícita.
- Se a minha rota incompleta atual já possui distância maior que a minha melhor rota completa até então conhecida, então, não precisa continuar com o caminho incompleto atual.
  - Política de corte em uma busca em profundidade.



# Caminhos mais curtos de origem única

- Para o problema dos caminhos mais curtos, temos um grafo orientado ponderado

$$G = (V, A)$$

- Com função de peso

$$w : A \rightarrow \Re$$

- Mapeando as arestas em valores reais.

# Caminhos mais curtos de origem única

- O peso do caminho $p$

$$p = \langle v_0, v_1, v_2, \ldots, v_k \rangle$$

- é o somatório dos pesos de suas arestas

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

# Caminhos mais curtos de origem única

- Definimos peso do caminho mais curto de *u* até *v* como:

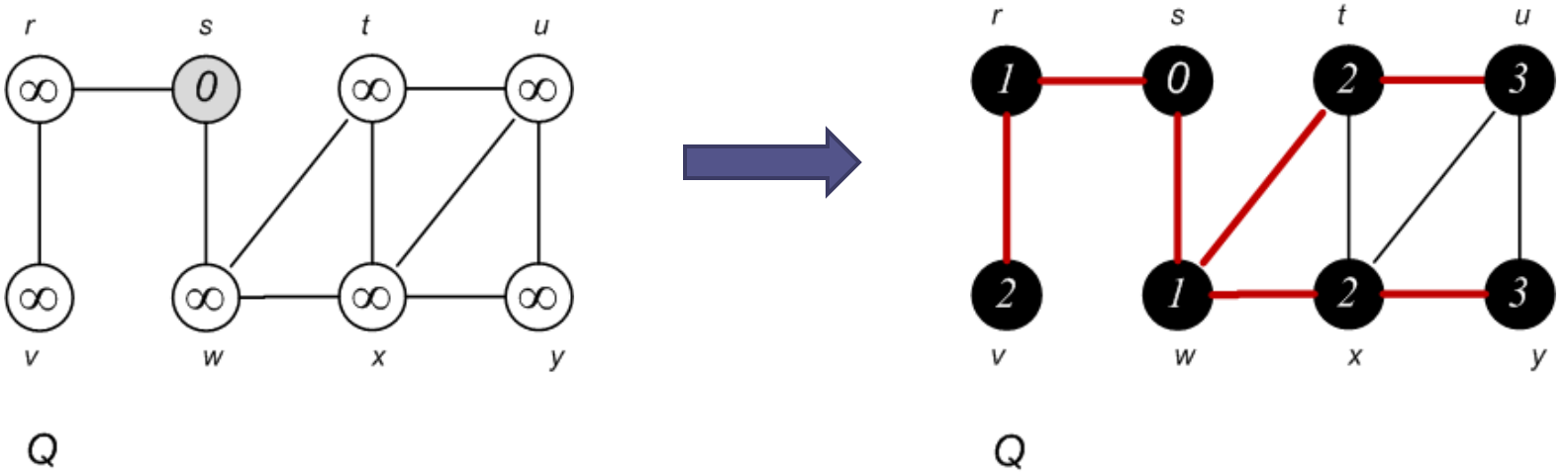
$$\delta(u,v) = \begin{cases} \min\{ w(p) : u \overset{p}{\rightsquigarrow} v \}, \text{ se existe caminho de u até v} \\ \infty, \text{ caso contrário} \end{cases}$$

- Assim, um caminho mais curto de u para v é definido como qualquer caminho p onde

$$w(p) = \delta(u,v)$$

# Caminhos mais curtos de origem única

- Relembrando...
  - O algoritmo de busca em largura (BFS) é um algoritmo que calcula caminhos mais curtos em grafos não ponderados...



# Caminhos mais curtos de origem única

- Variantes do Problema
  - Caminhos mais curtos de origem única;
  - Caminhos mais curtos de destino único;
  - Caminho mais curto de um único par;
  - Caminhos mais curtos de todos para todos;

# Sub-estrutura ótima

# Caminhos mais curtos de origem única

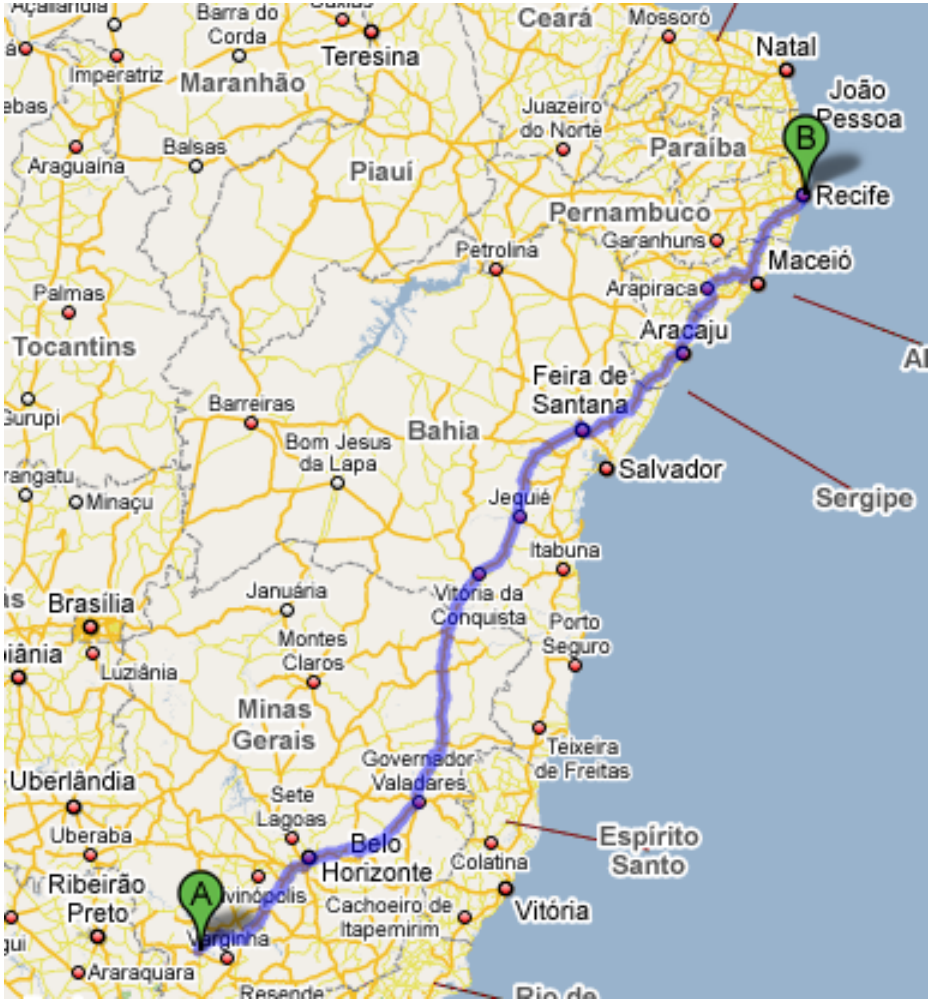
- Sub-estrutura ótima de um caminho mais curto
  - Serve de base para todos os algoritmos exatos que calculam as variantes do caminho mais curto;
  - Seja $p=<v_1,v_2,\ldots,v_k>$ um caminho mais curto de $v_1$ para $v_k$;
    - Para quaisquer $i$ e $j$ tais que $1<=i<=j<=k$, e seja $p_{ij}=<v_i, v_{i+1},\ldots,v_j>$ o subcaminho $p'$ desde o vértice $v_i$ até $v_j$.
    - Então $p_{ij}$ é um caminho mais curto de $v_i$ para $v_j$.

# Caminhos mais curtos de origem única

- Subestrutura ótima de um caminho mais curto

Sub-caminho de Aracaju para Maceió é ótimo;

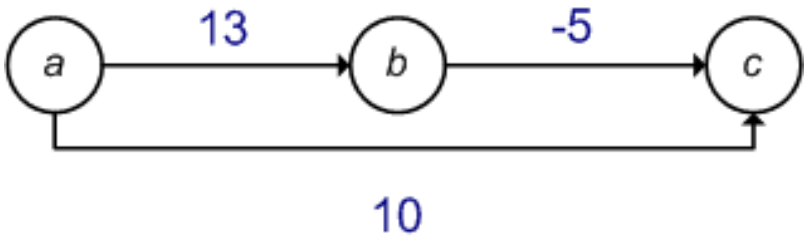
Sub-caminho de Maceió para Recife também é ótimo.



# Arestas de peso negativo em um Algoritmo de Caminho Mínimo

# Caminhos mais curtos de origem única

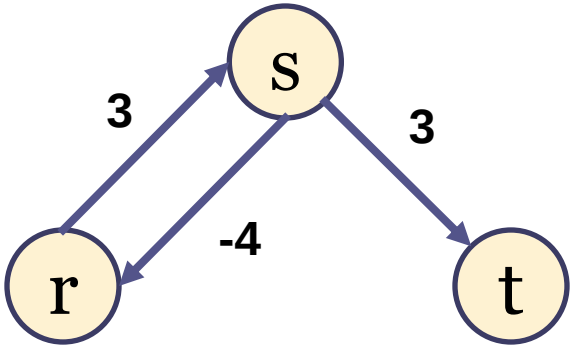
- Arestas de peso negativo
  - Grafos podem ter arestas de peso negativo;
  - Qual é o custo do menor caminho de *a* até *c*?




  - Você visualiza alguma dificuldade envolvendo arestas de pesos negativos?

# Caminhos mais curtos de origem única

- Arestas de peso negativo
  - Se existe um ciclo de peso negativo acessível a partir da origem *s*, os pesos dos caminhos mais curtos perdem a referencia;
  - Sempre será possível encontrar um caminho mais curto de peso menor que o já encontrado;



# Caminhos mais curtos de origem única

- Arestas de peso negativo
  - Se existe um ciclo de peso negativo em algum caminho desde $s$ até $v$, definimos

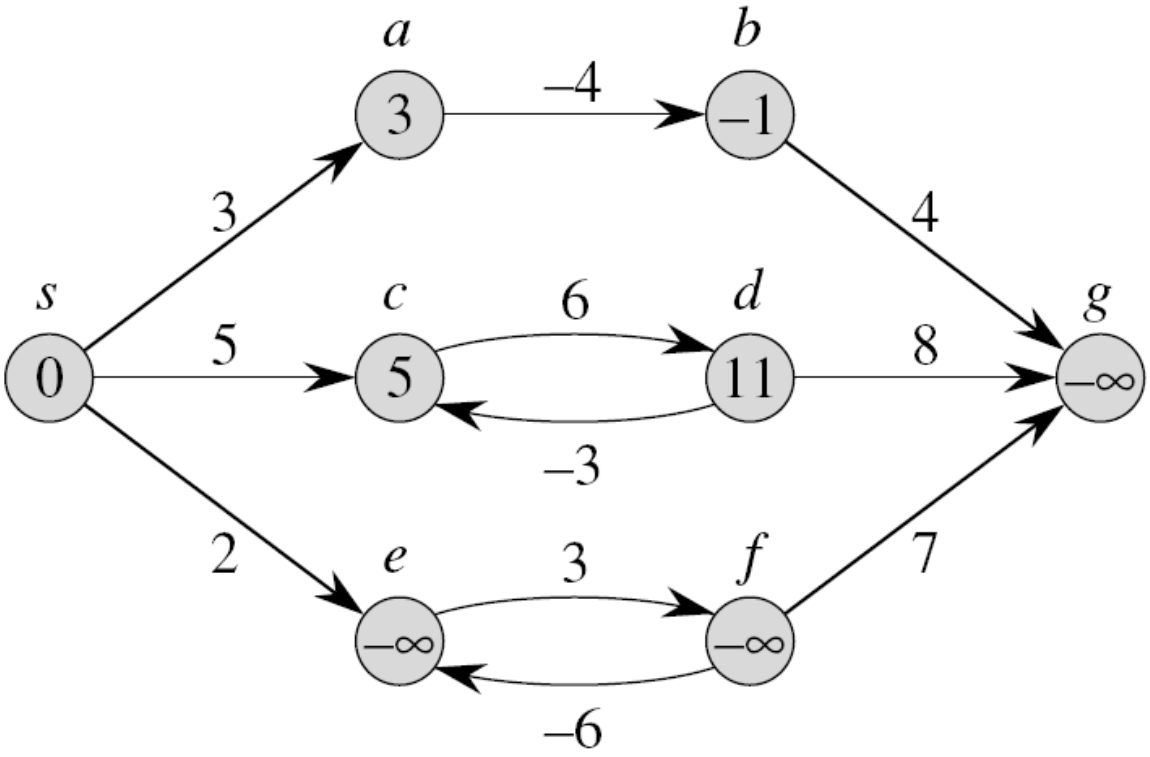
$$\delta(s,v) = -\infty$$

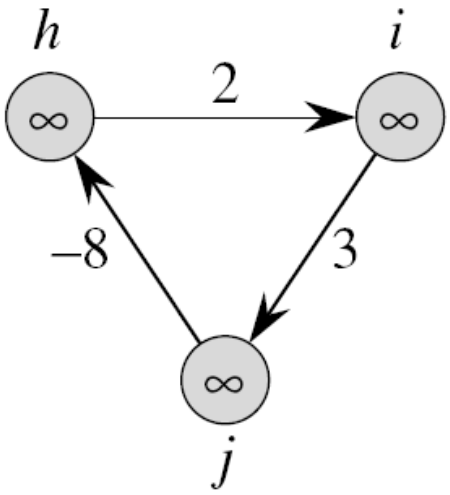
  - Relembrando...

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

# Caminhos mais curtos de origem única

- Arestas de peso negativo



# Caminhos mais curtos de origem única

- Arestas de peso negativo
  - O algoritmo de Dijkstra assume que todos os pesos de arestas no grafo de entrada são não negativos;
    - Ideal para aplicação em mapas rodoviários;
    - Algoritmo mais aplicado na prática em sistemas comerciais;
  - O algoritmo de Bellman-Ford permite a existência de arestas de peso negativo, e produz a resposta correta;
    - *Não entra em loop infinito.*

# Caminhos mais curtos de origem única

- Ciclos
  - Como vimos, caminhos mais curtos não podem conter ciclos de peso negativo;
  - Mas eles podem conter ciclo de peso positivo?

# Caminhos mais curtos de origem única

- Ciclos
  - Como vimos, caminhos mais curtos não podem conter ciclos de peso negativo;
  - Mas eles podem conter ciclo de peso positivo?
  - E ciclo de peso zero?

# Caminhos mais curtos de origem única

- Representação de caminhos mais curtos
  - Assim como na busca em largura (BFS) iremos utilizar os vetores $d$ e $\pi$ para recuperar caminhos após a aplicação dos algoritmos;
  - Relembrando:

$$\pi[u] = \begin{cases} \text{pai do vértice u, se u é alcançável;} \\ \text{NULL, em caso contrário.} \end{cases}$$

$$d[u] = \begin{cases} \delta(s,u), \text{ se u é alcançável;} \\ \infty, \text{ em caso contrário.} \end{cases}$$

# Caminhos mais curtos de origem única

- **Relembrando:**
  - Imprimir caminho de s até v através da estrutura π

$$
\begin{aligned}
&PRINT\_PATH(G,s,v)\\
&\quad if\ \ v=s\\
&\qquad cout << s\\
&\quad else\ if\ \ \pi[v]=NULL\\
&\qquad cout << "não\ existe\ trajeto"\\
&\quad else\\
&\qquad PRINT\_PATH(G,s,\pi[v])\\
&\qquad cout << v\\
&\quad end\ \ if\\
&end\ \ function
\end{aligned}
$$

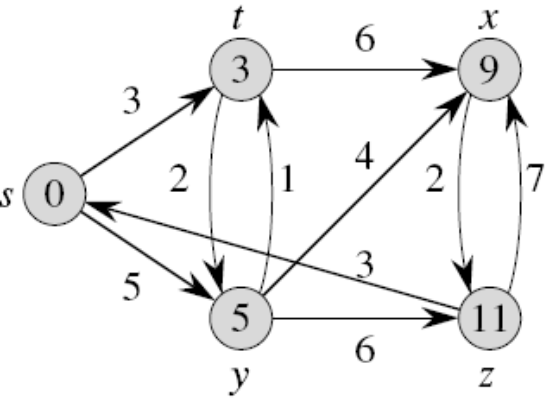
# Caminhos mais curtos de origem única

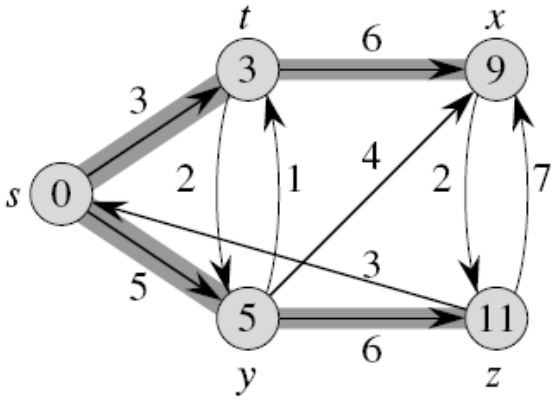
- Árvore de caminhos mais curtos

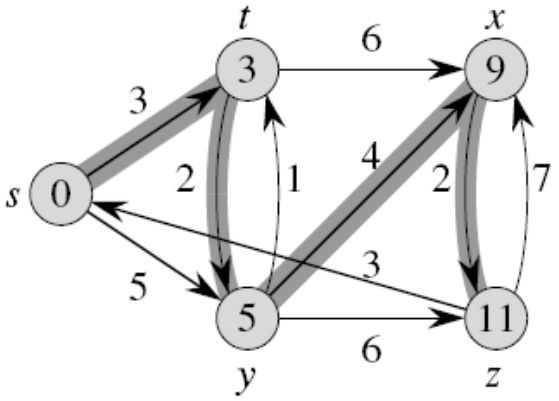
$$G' = (V', A')$$

$$V' \subseteq V \text{ e } A' \subseteq A$$

- V' é o conjunto de vértices acessíveis a partir de *s* no grafo;
- G' forma uma árvore enraizada com raiz *s*;
- Para todo vértice do grafo, o único caminho simples desde *s* até *v* em G' é um caminho mais curto desde *s* até *v* em G.



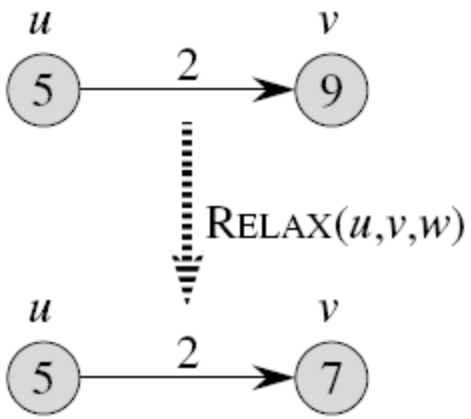
# Caminhos mais curtos de origem única

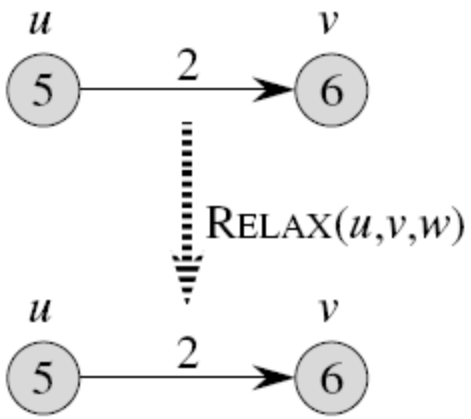
- **Inicialização dos vetores**
  - Os algoritmos que veremos para esta classe de problemas usam um método para inicializar os vetores auxiliares:

$$\text{INICIALIZA}(G = (V, A), s)$$
$$para\ cada\ v \in V$$
$$d[v] = \infty$$
$$\pi[v] = NULL$$
$$fim\ \ para$$
$$d[s] = 0$$
$$fim$$

# Caminhos mais curtos de origem única

- Técnica do relaxamento
  - "O processo de relaxar uma aresta *(u,v)* consiste em testar se podemos melhorar o caminho mais curto para *v* encontrado até agora pela passagem de *u* e, neste caso, atualizar *d[v]* e $\pi[v]$."



# Caminhos mais curtos de origem única

- Técnica do relaxamento

$$\text{RELAXA}(u, v, w)$$

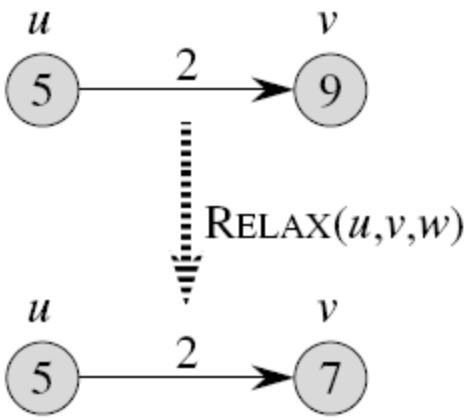
$$se\ d[v] > (d[u] + w(u,v))\ então$$

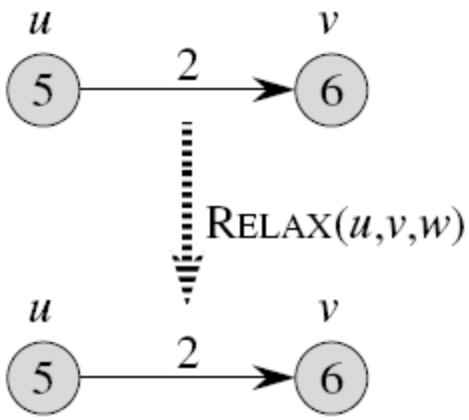
$$d[v] \leftarrow d[u] + w(u,v)$$

$$\pi[v] = u$$

$$fim\ \ se$$

$$fim$$



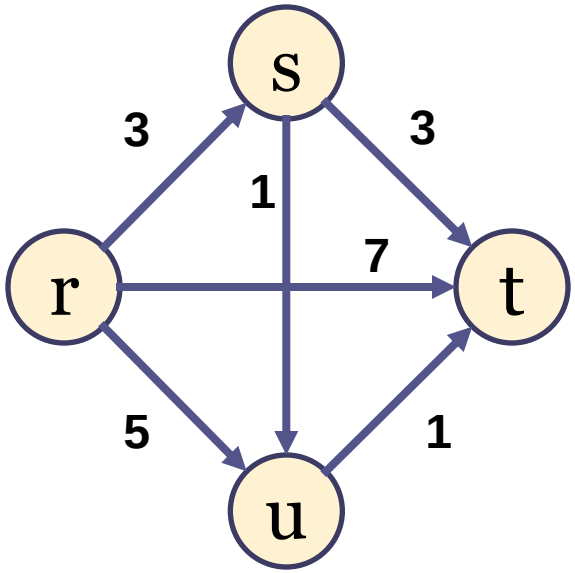
# Algoritmo de Bellman-Ford

# Algoritmo de Bellman-Ford

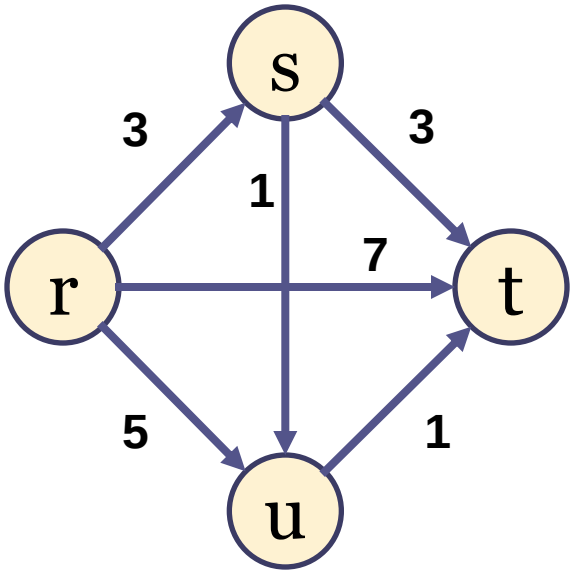
- Resolve o problema de caminhos mais curtos de uma única origem;
- São permitidas arestas com peso negativo (resolve o caso mais geral);
- Retorna verdadeiro se não existe ciclo negativo, e falo em caso contrário;

# Algoritmo de Bellman-Ford

- Vamos encontrar os caminhos mais curtos de *r* para todos os outros vértices do seguinte grafo:



# Algoritmo de Bellman-Ford




$$\text{INICIALIZA}(G = (V, A), s)$$

*para cada* $v \in V$

$d[v] = \infty$

$\pi[v] = NULL$

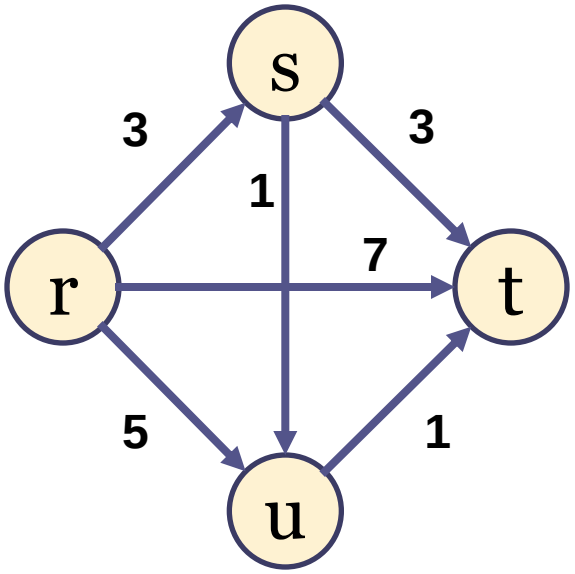
*fim para*

$d[s] = 0$

*fim*

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ |
| π | NULL | NULL | NULL | NULL |

# Algoritmo de Bellman-Ford




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ |
| π | NULL | NULL | NULL | NULL |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

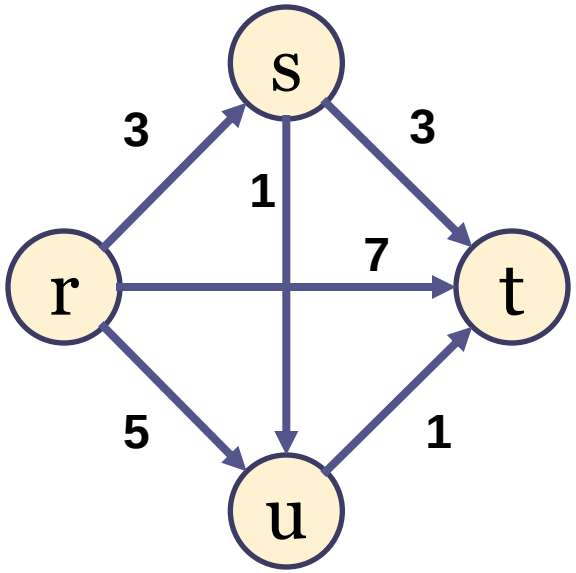
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ |
| π | NULL | NULL | NULL | NULL |

$$BELLMAN-FORD(G=(V,A),w,s)$$
$$INICIALIZA(G,s)$$
$$para\ i \leftarrow 1\ até\ |V|-1$$
$$para\ cada\ aresta\ (u,v) \in A$$
$$RELAXA(u,v,w)$$
$$fim\ para$$
$$fim\ para$$
$$para\ cada\ aresta\ (u,v) \in A$$
$$se\ d[v] > d[u] + w(u,v)$$
$$retorna\ falso$$
$$fim\ se$$
$$fim\ para$$
$$retorna\ verdadeiro$$
$$fim$$

# Algoritmo de Bellman-Ford

(s,t)

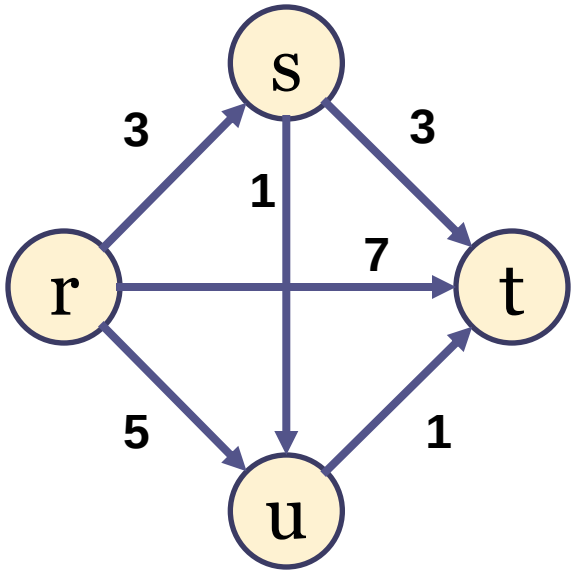
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ |
| π | NULL | NULL | NULL | NULL |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

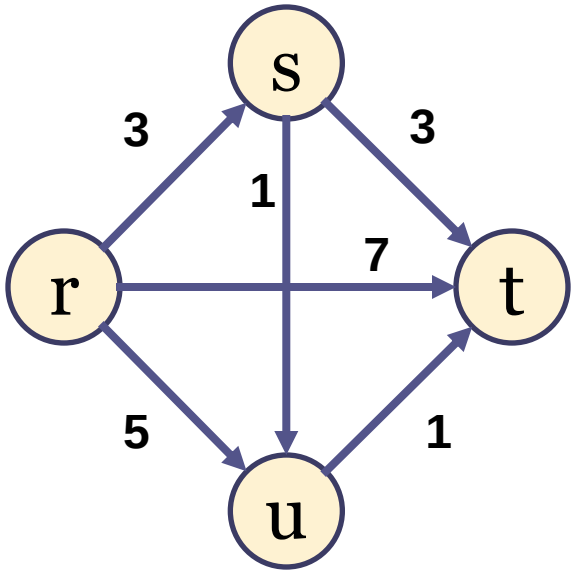
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ |
| π | NULL | NULL | NULL | NULL |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

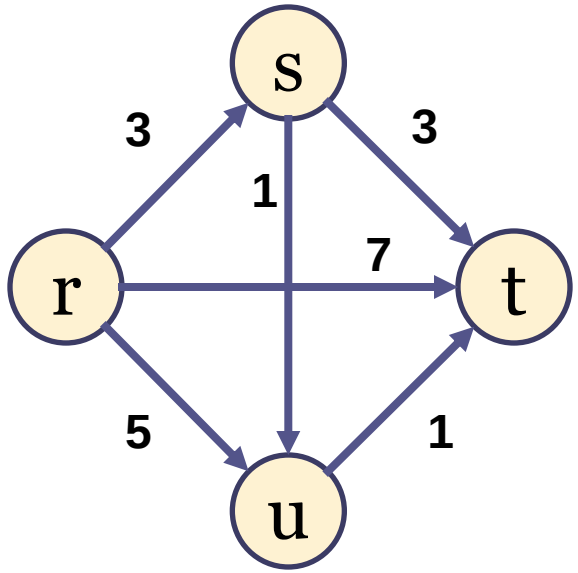
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ |
| π | NULL | NULL | NULL | NULL |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

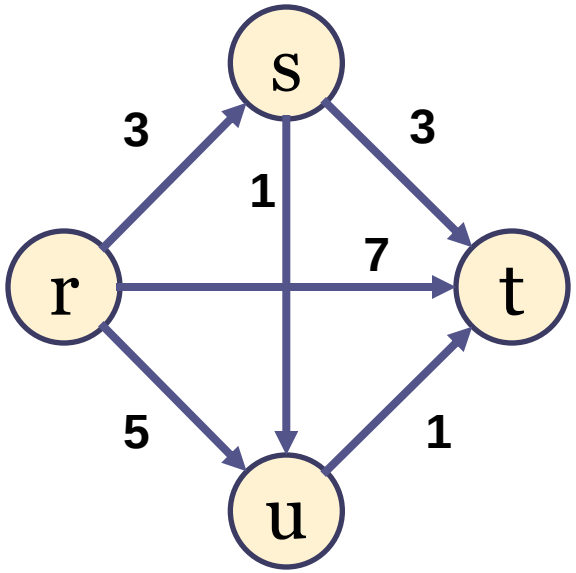
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | ∞ | 7 | ∞ |
| π | NULL | NULL | r | NULL |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

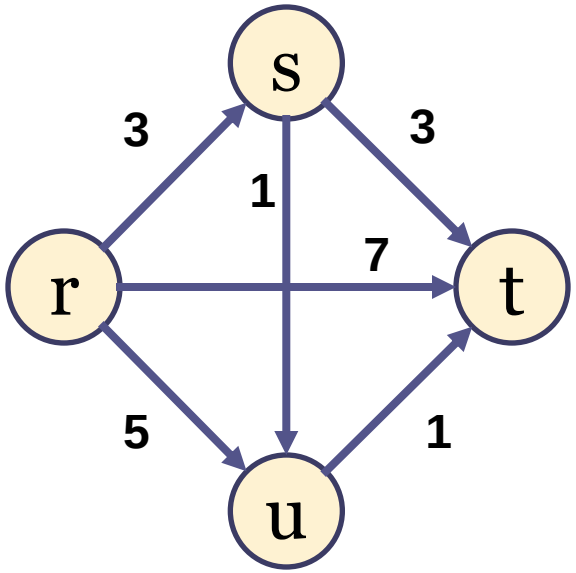
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 7 | ∞ |
| π | NULL | r | r | NULL |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

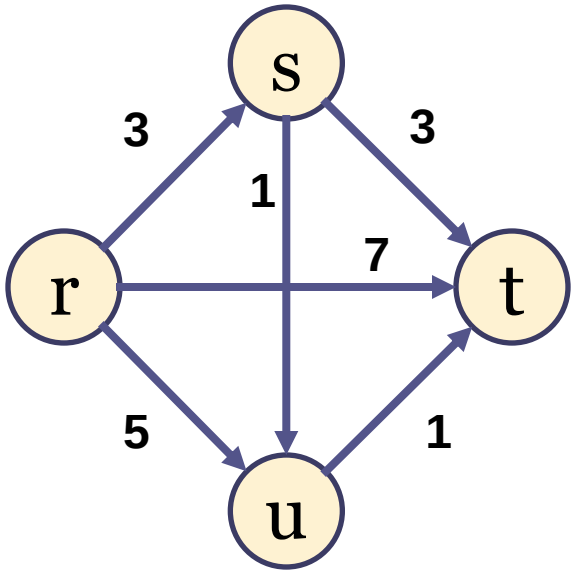
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 1 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 7 | 5 |
| π | NULL | r | r | r |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

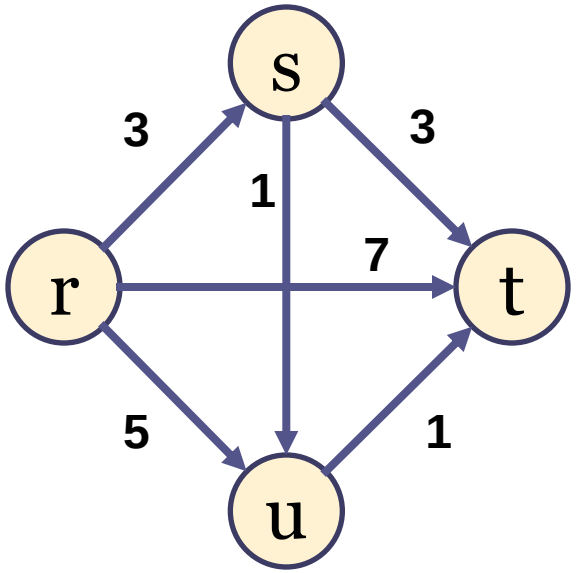
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 7 | 5 |
| π | NULL | r | r | r |

```
BELLMAN – FORD(G = (V, A), w, s)
    INICIALIZA(G, s)
    para i ← 1 até |V| – 1
        para cada aresta (u, v) ∈ A
            RELAXA(u, v, w)
        fim para
    fim para
    para cada aresta (u, v) ∈ A
        se d[v] > d[u] + w(u, v)
            retorna falso
        fim se
    fim para
    retorna verdadeiro
fim
```

# Algoritmo de Bellman-Ford

(s,t)

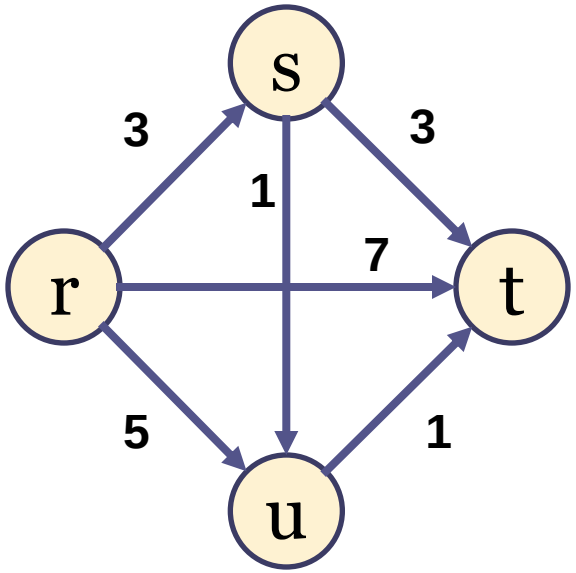
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 6 | 5 |
| π | NULL | r | s | r |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

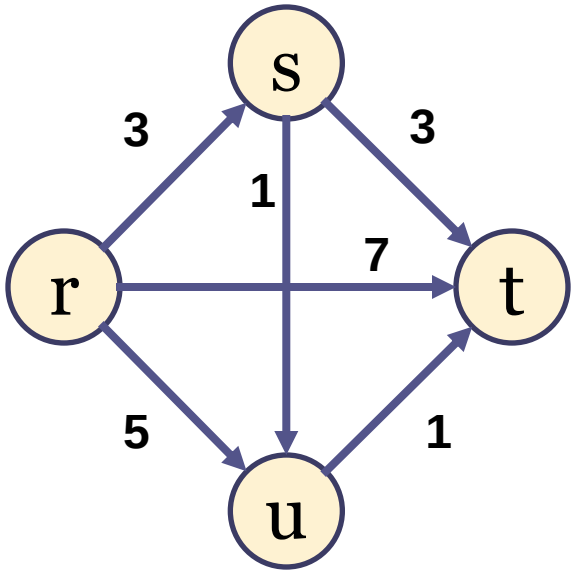
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 6 | 4 |
| π | NULL | r | s | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

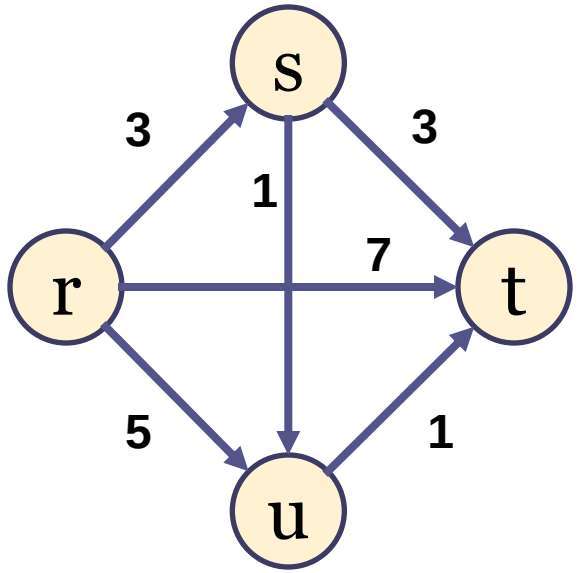
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

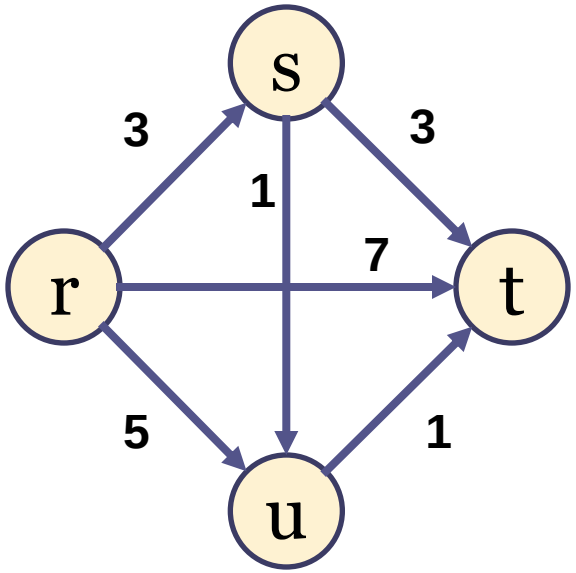
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

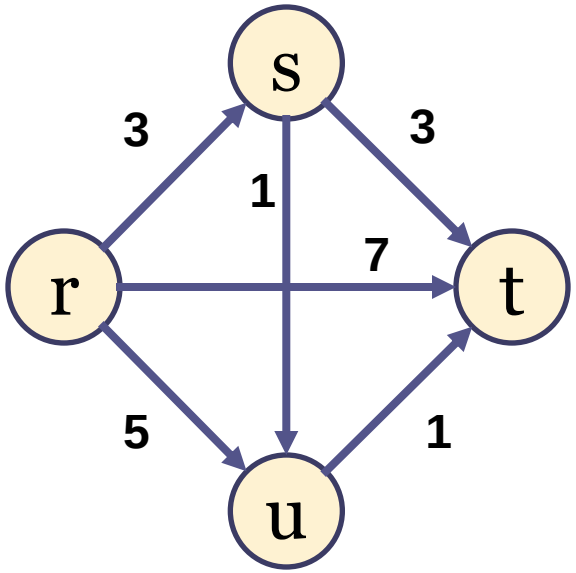
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

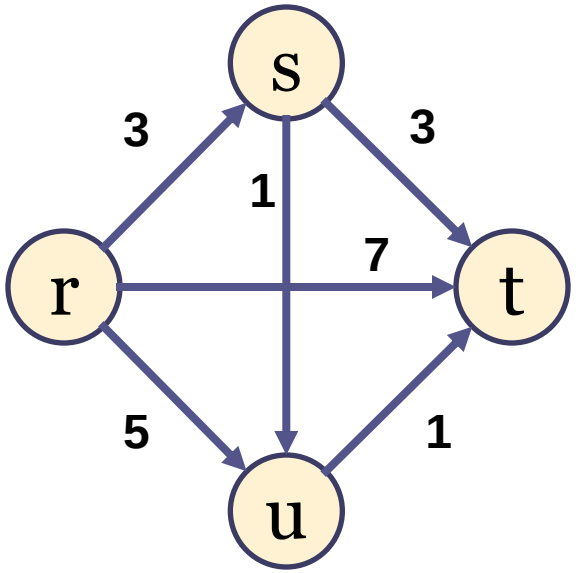
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 2 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

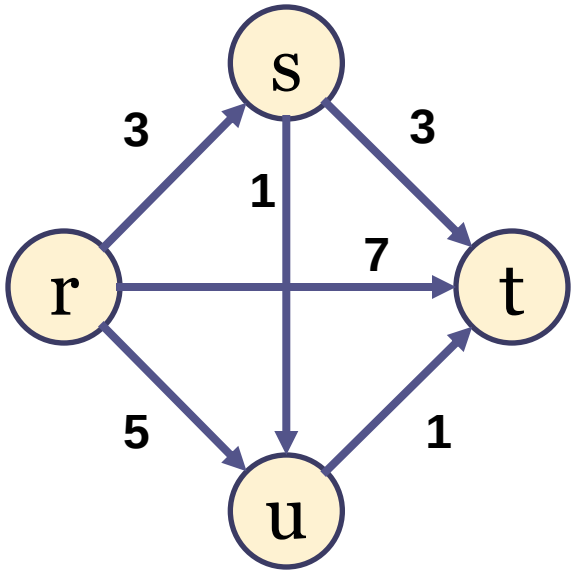
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | o | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

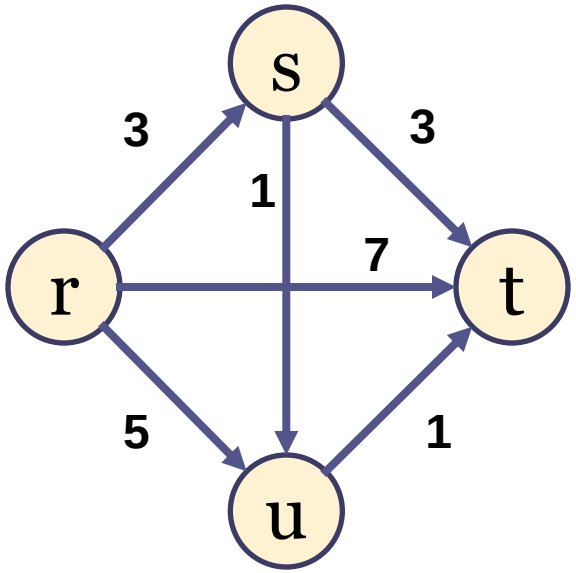
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | ∞ | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

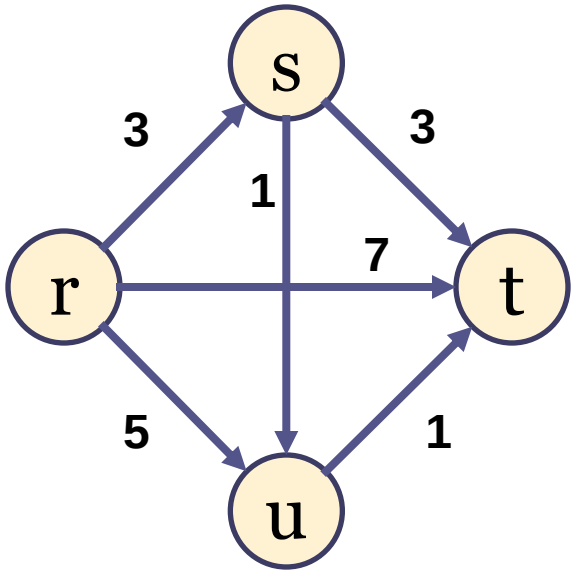
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

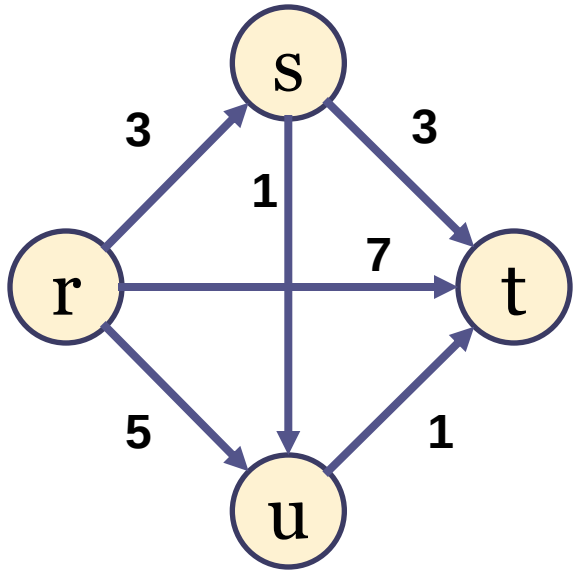
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

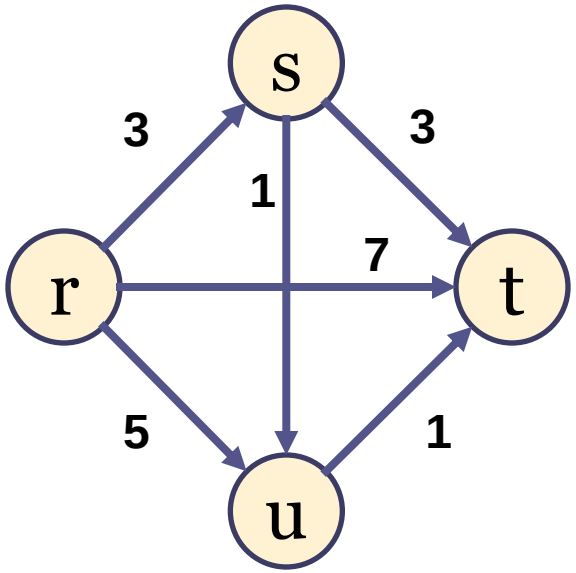
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | ∞ | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

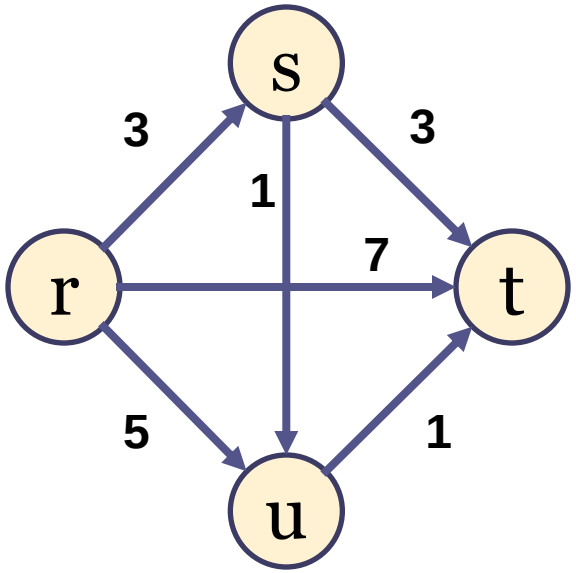
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | ∞ | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

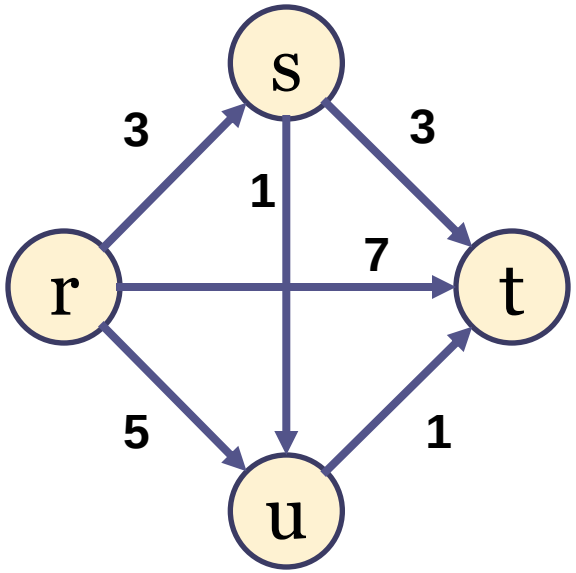
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | ∞ | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

(s,t)

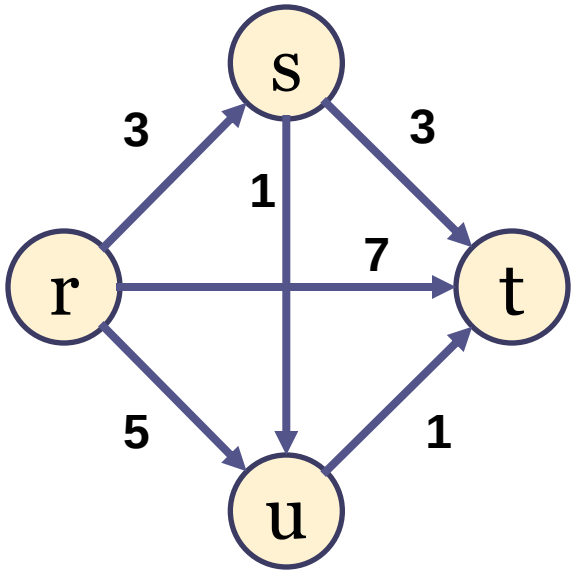
(s,u)

(u,t)

(r,t)

(r,s)

(r,u)




| variável | valor |
|---|---|
| $i$ | 3 |

| vértice | r | s | t | u |
|---|---|---|---|---|
| d | o | 3 | 5 | 4 |
| π | NULL | r | u | s |

$BELLMAN-FORD(G=(V,A),w,s)$

$INICIALIZA(G,s)$

$para\ i \leftarrow 1\ até\ |V|-1$

$para\ cada\ aresta\ (u,v) \in A$

$RELAXA(u,v,w)$

$fim\ para$

$fim\ para$

$para\ cada\ aresta\ (u,v) \in A$

$se\ d[v] > d[u] + w(u,v)$

$retorna\ falso$

$fim\ se$

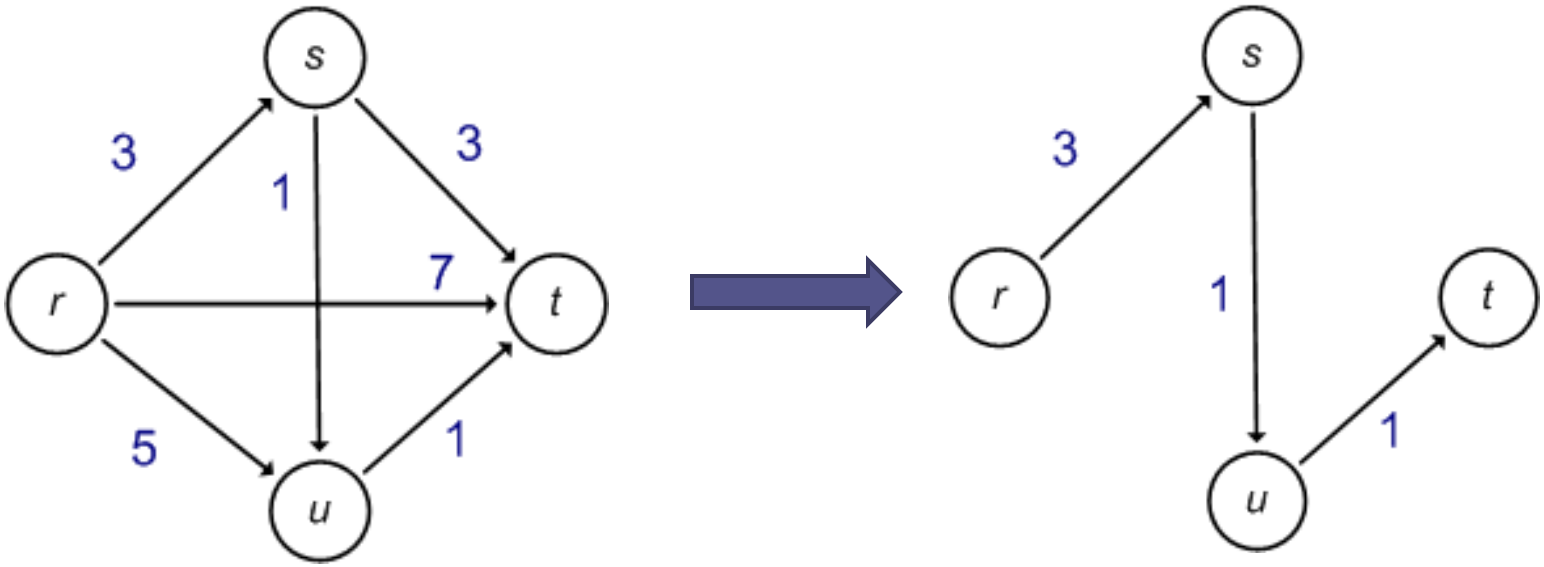
$fim\ para$

$retorna\ verdadeiro$

$fim$

# Algoritmo de Bellman-Ford

- Solução do exemplo...




| vértice | r | s | t | u |
|---|---|---|---|---|
| d | 0 | 3 | 5 | 4 |
| π | NULL | r | u | s |

# Bellman-Ford

- Aplicação:
  - Uma variação **distribuída** deste algoritmo é implementada na área de redes de computadores;
  - O algoritmo de roteamento chamado **vetor de distâncias**;
  - Seu objetivo é fornecer informação para cada *host* que deseja enviar pacotes (criar tabelas de encaminhamento);
  - Por qual saída ele deve enviar um pacote de modo que o mesmo chegue rapidamente ao destinatário;

# Bellman-Ford

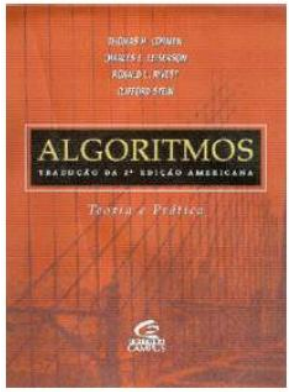
- Aplicação:
  - Na área de redes de computadores devemos tomar cuidado com os "caminhos mais curtos", ou os "caminhos de maior vazão";
  - Podemos congestionar a toda a rede se, na execução do Algoritmo do Vetor de Distâncias, um *host* se tornar o "gargalo" da rede.
  - Geralmente este problema é "resolvido" no protocolo TCP;

# Exercício

- Proponha um grafo de 5 vértices, com um ciclo de peso negativo, e apresente passo a passo a execução do algoritmo Bellman-Ford.

# Bibliografia


- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002). Algoritmos – Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro. Editora Campus.




- ZIVIANI, N. (2007). Projeto e Algoritmos com implementações em Java e C++. São Paulo. Editora Thomson;