ACH2024

Aula 11 – Grafos:

Caminhos de peso mínimo (shortest paths na literatura) – Algoritmo de Bellman-Ford

Profa. Ariane Machado Lima

Aulas passadas

O que já vimos em grafos

- Conceitos básicos (**direcionados e não direcionados**)
- Implementação por **matriz e lista de adjacências** (mantendo uma única interface)
- **Makefile**
- **Busca em profundidade** e aplicações: detecção de ciclos, de um caminho, ordenação topológica, identificação de componentes conectados (fraca ou fortemente)
- **Busca em largura** e aplicações: caminho mais curto, todos os vértices alcançáveis dentro de um certo raio, etc.
- **Árvore geradora mínima** (algoritmos de Prim e Kruskal): conecta todos os vértices do grafo (sem redundância) com soma mínima de pesos (se fosse árvore geradora máxima a soma seria máxima)

Caminhos de Peso Mínimo

Caminhos de peso mínimo

Um caminho **mais curto** é aquele com **menor número de arestas**

Muitas vezes não estamos interessados no número de arestas, e sim no custo do caminho (soma dos pesos das arestas do caminho), ou seja, no caminho de peso mínimo

A aplicação direta da busca em largura, como feita para caminhos mais curtos, não é mais suficiente

Infelizmente, esse problema é também chamado “caminho mais curto” (ex: Cormen e Ziviani)

Usarei o termo “caminho mais curto” como sinônimo de “caminho de peso mínimo” nesta aula por usar os slides do Ziviani

Assume-se um grafo **direcionado** e **ponderado**

Caminhos Mais Curtos: Aplicação

- Um motorista procura o caminho mais curto entre Diamantina e Ouro Preto. Possui mapa com as distâncias entre cada par de interseções adjacentes.
- Modelagem:
  - $G = (V, A)$: grafo direcionado ponderado, mapa rodoviário.
  - V: interseções.
  - A: segmentos de estrada entre interseções
  - $p(u, v)$: peso de cada aresta, distância entre interseções.
- Peso de um caminho: $p(c) = \sum_{i=1}^{k} p(v_{i-1}, v_i)$
- Peso do caminho de peso mínimo (do caminho “mais curto”):

$$\delta(u, v) = \begin{cases} min\left\{p(c) : u \overset{c}{\rightsquigarrow} v\right\} & \text{se existir caminho de } u \text{ a } v \\ \infty & \text{caso contrário} \end{cases}$$

- **Caminho mais curto** do vértice u ao vértice v: qualquer caminho c com peso $p(c) = \delta(u, v)$.

Ou seja, pode haver mais de um caminho mais curto

Caminhos Mais Curtos

- **Caminhos mais curtos a partir de uma origem**: dado um grafo ponderado $G = (V, A)$, desejamos obter o caminho mais curto a partir de um dado vértice origem $s \in V$ até cada $v \in V$.
- Muitos problemas podem ser resolvidos pelo algoritmo para o problema origem única:
  - **Caminhos mais curtos com destino único**: reduzido ao problema origem única invertendo a direção de cada aresta do grafo.
  - **Caminhos mais curtos entre um par de vértices**: o algoritmo para origem única é a melhor opção conhecida.
  - **Caminhos mais curtos entre todos os pares de vértices**: resolvido aplicando o algoritmo origem única $|V|$ vezes, uma vez para cada vértice origem.

Caminhos mais curtos – quando eles não existem

Considerando a modelagem matemática proposta, há dois casos em que não existe um caminho mais curto entre dois vértices

Caminhos mais curtos – quando eles não existem

Considerando a modelagem matemática proposta, há dois casos em que não existe um caminho mais curto entre dois vértices

Para um vértice origem s:

Se um vértice v não é alcançável por s:

$\delta(s, v) = infinito$

Se houver algum ciclo alcancável por s com peso total negativo:

- Para cada vértice v deste ciclo ou para vértice v para o qual existe um caminho de s a v passando pelo ciclo,
  - Não existe um caminho mais curto de s a v,
  - Ou seja, $\delta(s, v) = -infinito$

Caminhos mais curtos – quando eles não existem

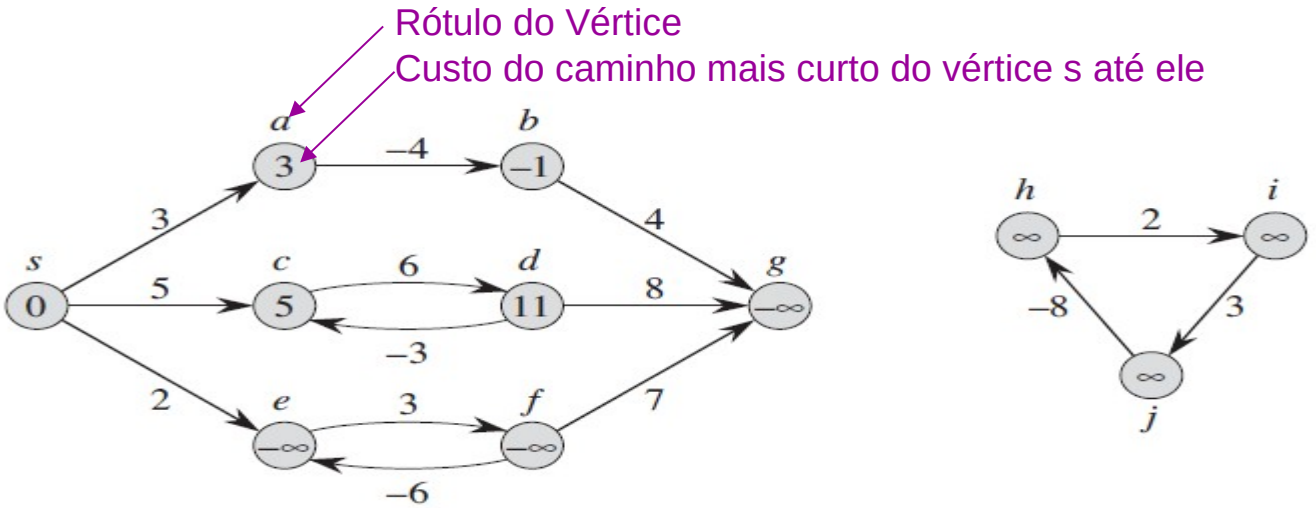


Figura: Livro do Cormem cap 24

Caminhos Mais Curtos

- A representação de caminhos mais curtos pode ser realizada pela variável *Antecessor*.
- Para cada vértice $v \in V$ o $Antecessor[v]$ é um outro vértice $u \in V$ ou *nil* (-1).
- O algoritmo atribui a *Antecessor* os rótulos de vértices de uma cadeia de antecessores com origem em v e que anda para trás ao longo de um caminho mais curto até o vértice origem s.
- Dado um vértice v no qual $Antecessor[v] \neq nil$, o procedimento *ImprimeCaminho* pode imprimir o caminho mais curto de s até v. (aula 8 sl 71)
- Os valores em $Antecessor[v]$, em um passo intermediário, não indicam necessariamente caminhos mais curtos.
- Entretanto, ao final do processamento, *Antecessor* contém uma árvore de caminhos mais curtos definidos em termos dos pesos de cada aresta de G, ao invés do número de arestas. (vetor "distancia" armazenará soma dos pesos)
- Caminhos mais curtos não são necessariamente únicos.

π[v] no livro do Cormen

Caminhos Mais Curtos

- A representação de caminhos mais curtos pode ser realizada pela variável *Antecessor*.
- Para cada vértice $v \in V$ o $Antecessor[v]$ é um outro vértice $u \in V$ ou *nil* (-1).
- O algoritmo atribui a *Antecessor* os rótulos de vértices de uma cadeia de antecessores com origem em v e que anda para trás ao longo de um caminho mais curto até o vértice origem s.
- Dado um vértice v no qual $Antecessor[v] \neq nil$, o procedimento *ImprimeCaminho* pode imprimir o caminho mais curto de s até v.
- Os valores em $Antecessor[v]$, em um passo intermediário, não indicam necessariamente caminhos mais curtos.
- Entretanto, ao final do processamento, *Antecessor* contém uma árvore de caminhos mais curtos definidos em termos dos pesos de cada aresta de G, ao invés do número de arestas. (vetor "distancia" armazenará soma dos pesos)
- Caminhos mais curtos não são necessariamente únicos.

d[v] no livro do Cormen

Árvore de caminhos mais curtos

- Uma árvore de caminhos mais curtos com raiz em $s \in V$ é um subgrafo direcionado $G' = (V', A')$, onde $V' \subseteq V$ e $A' \subseteq A$, tal que:
  1. V' é o conjunto de vértices alcançáveis a partir de $s \in G$,
  2. G' forma uma árvore de raiz s,
  3. para todos os vértices $v \in V'$, o caminho simples de s até v é um caminho mais curto de s até v em G.

Árvore de caminhos mais curtos

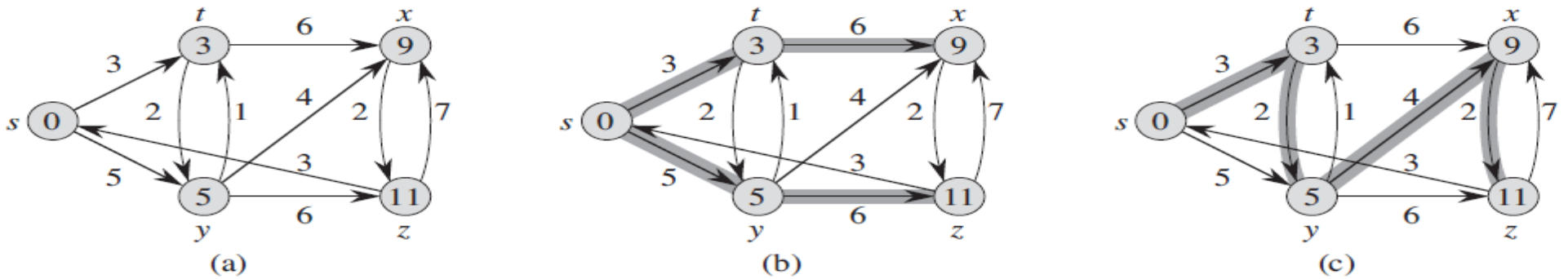


Figure 24.2 (a) A weighted, directed graph with shortest-path weights from source s. (b) The shaded edges form a shortest-paths tree rooted at the source s. (c) Another shortest-paths tree with the same root.

Algoritmos para construção de árvores de caminhos mais curtos (origem única)

- Caso quase geral (arestas podem possuir pesos negativos mas não ciclos negativos)
  - **Bellman-Ford**
- Caso em que todas as arestas possuem valores de peso não negativos (**mais eficiente**)
  - **Dijkstra**

Relaxamento

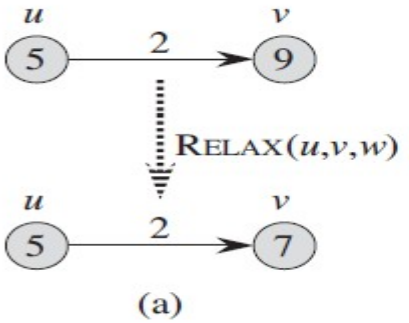
Técnica usada por algoritmos de caminhos mais curtos

Cada vértice v terá um valor d[v], que é uma estimativa de pior caso (limite superior) do custo mínimo do caminho de s (origem) a v

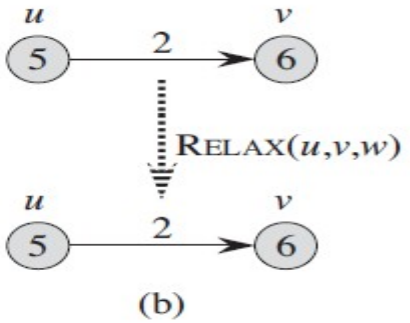
```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2        d[v]  = ∞
3        π[v]  = NIL
4  d[s] = 0
```

Relaxamento

Relaxar uma aresta (u,v): verificar se d[v] pode ser decrementada ao se considerar um caminho de s a v passando por u (ou seja, verificar se usar essa aresta melhora a estimativa atual):



(a)



(b)

w representa a informação dos pesos

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Algoritmo Bellman-Ford

Resolve o caso geral (arestas podem ter pesos negativos)

Retorna falso se o grafo tiver um ciclo negativo

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

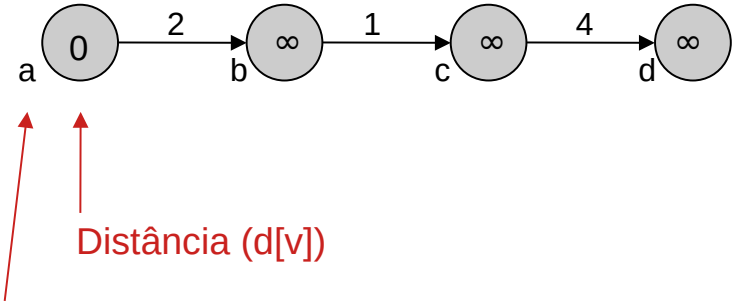
Como o caminho de peso mínimo não tem ciclo, tem comprimento no máximo |V|-1

Logo, em |V|-1 rodadas, todas as arestas deste caminho são corretamente relaxadas para seus valores reais

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     d[v]  = d[u] + w(u, v)
3     π[v]  = u
```



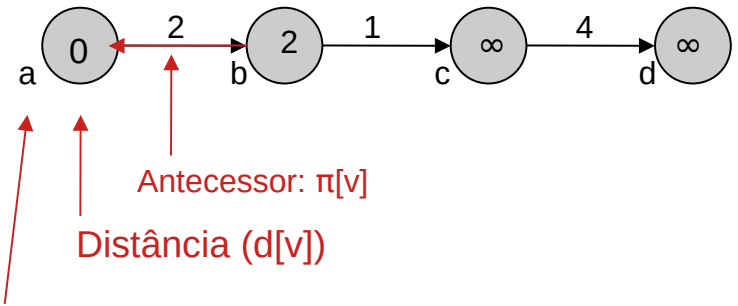
Como seria a execução do algoritmo?

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v]  = d[u] + w(u, v)
3      π[v]  = u
```

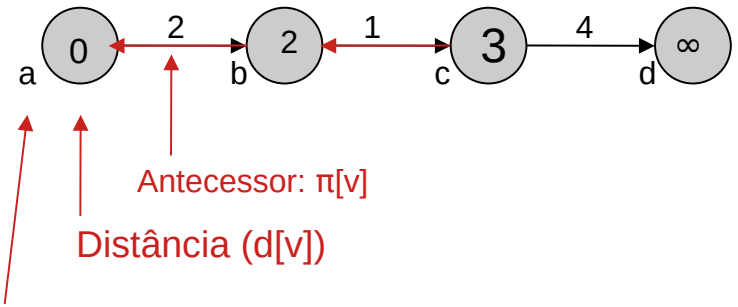


Como seria a execução do algoritmo?

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)




```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v] = d[u] + w(u, v)
3      π[v] = u
```

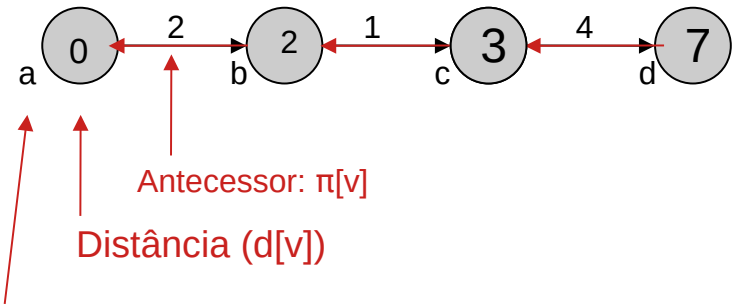
Como seria a execução do algoritmo?

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v] = d[u] + w(u, v)
3      π[v] = u
```



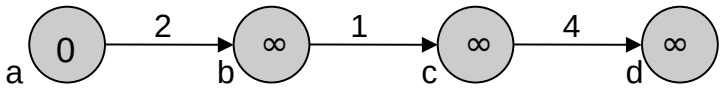
Como seria a execução do algoritmo?

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     d[v]  = d[u] + w(u, v)
3     π[v]  = u
```

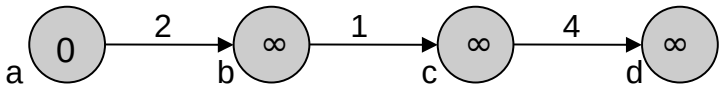


Como seria a execução do algoritmo?
Uma passada pelas arestas foi suficiente…
Precisa mesmo inspecionar as arestas V-1 vezes?

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)




```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     d[v]  = d[u] + w(u, v)
3     π[v]  = u
```

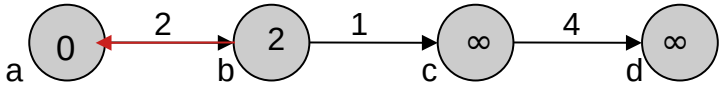
Como seria a execução do algoritmo?
Uma passada pelas arestas foi suficiente…
Precisa mesmo inspecionar as arestas V-1 vezes?
O que aconteceria se as arestas fossem inspecionadas, por azar, de trás para a frente?

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Como seria a execução do algoritmo?
Uma passada pelas arestas foi suficiente…
Precisa mesmo inspecionar as arestas V-1 vezes?
O que aconteceria se as arestas fossem inspecionadas, por azar, de trás para a frente?

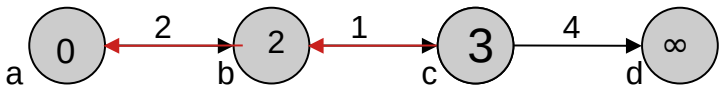
Passada 1:
Apenas d[b] seria atualizado

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Como seria a execução do algoritmo?
Uma passada pelas arestas foi suficiente…
Precisa mesmo inspecionar as arestas V-1 vezes?
O que aconteceria se as arestas fossem inspecionadas, por azar, de trás para a frente?

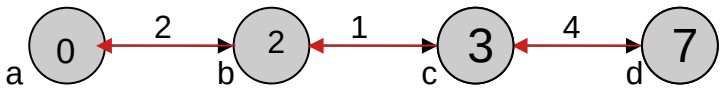
Passada 2:
Apenas d[c] seria atualizado

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Como seria a execução do algoritmo?
Uma passada pelas arestas foi suficiente…
Precisa mesmo inspecionar as arestas V-1 vezes?
O que aconteceria se as arestas fossem inspecionadas, por azar, de trás para a frente?

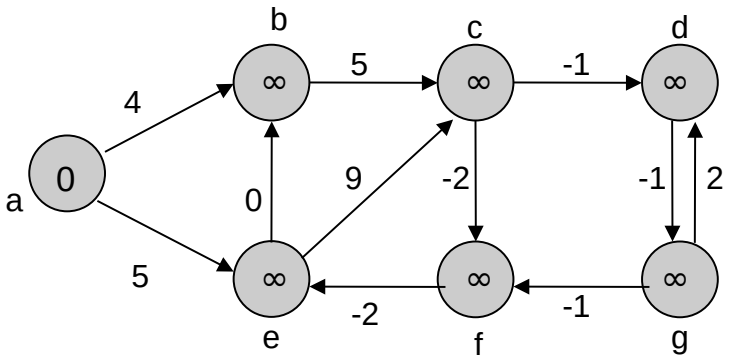
Passada 3:
Finalmente d[d] seria atualizado

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     d[v]  = d[u] + w(u, v)
3     π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

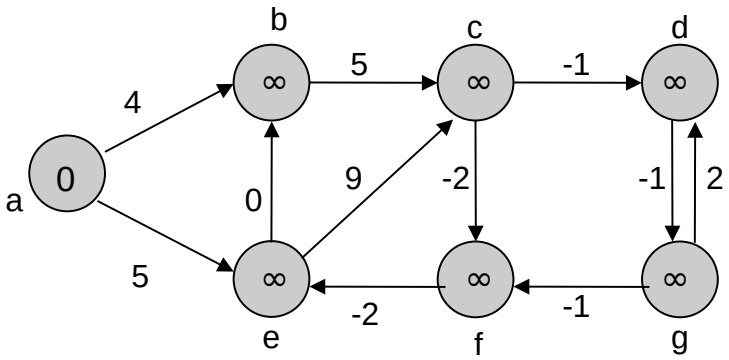
Vou usar uma fila aqui para fazer esse percurso em largura, mas note que isso **NÃO** é necessário para o algoritmo, apenas para o exemplo que mesmo percorrendo em largura 1 passada pelas arestas pode não ser suficiente...

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v] = d[u] + w(u, v)
3      π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

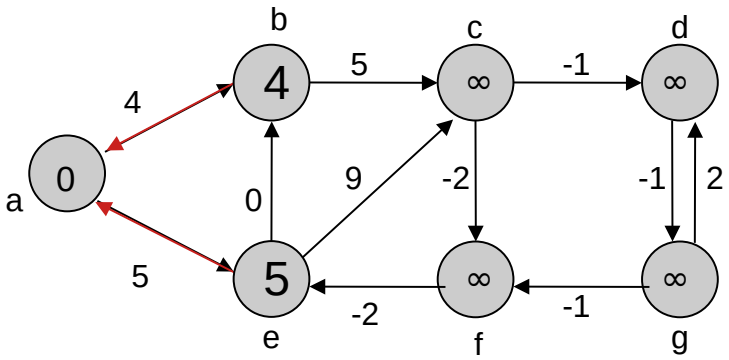
Passada 1:

"Fila": a

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 1:

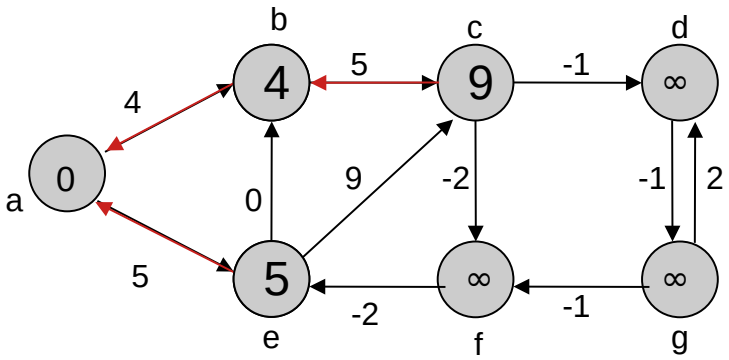
"Fila": a, b, e

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v] = d[u] + w(u, v)
3      π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 1:

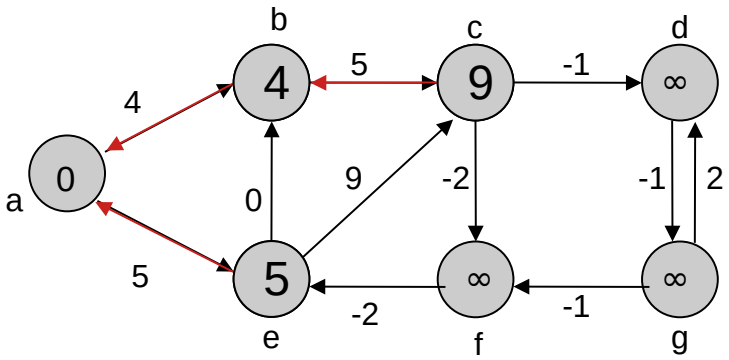
"Fila": a, b, e, c

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v] = d[u] + w(u, v)
3      π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

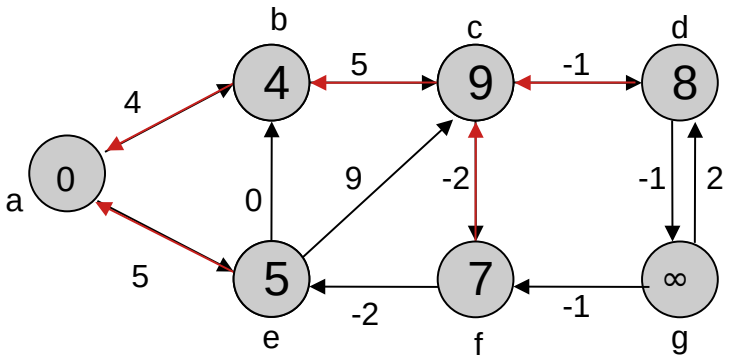
Passada 1:

"Fila": a, b, e, c

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

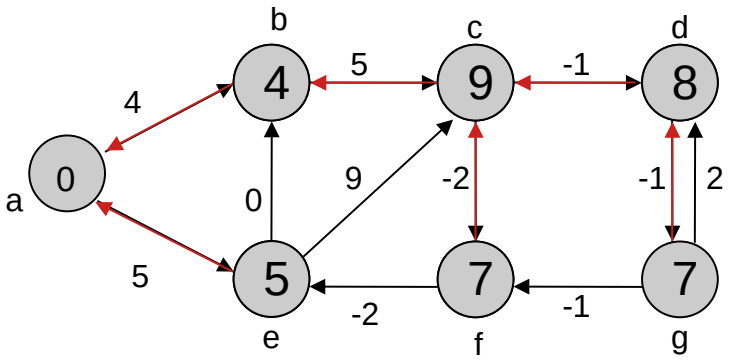
Passada 1:

"Fila": a, b, e, c, d, f

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 1:

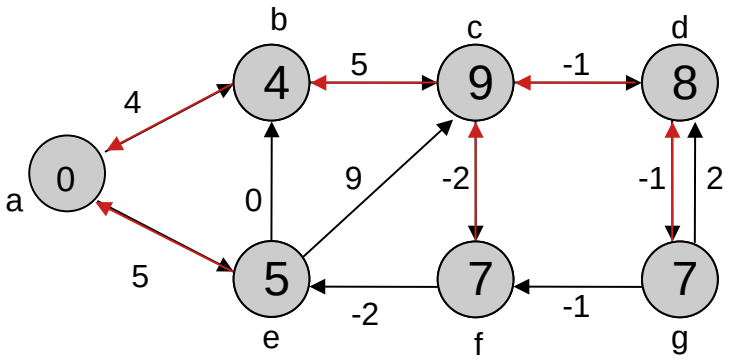
"Fila": a, b, e, c, d, f, g

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 1:

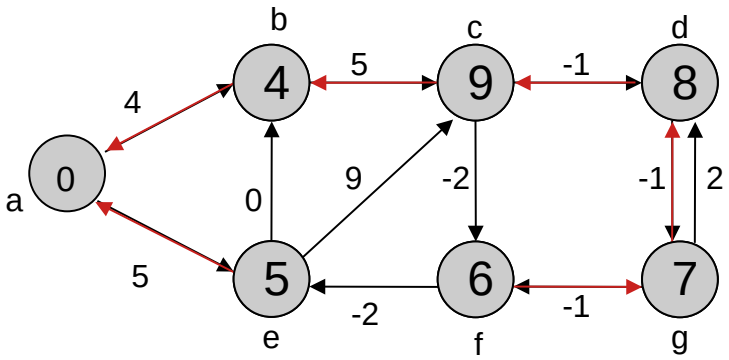
"Fila": a, b, e, c, d, f, g

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Passada 1:

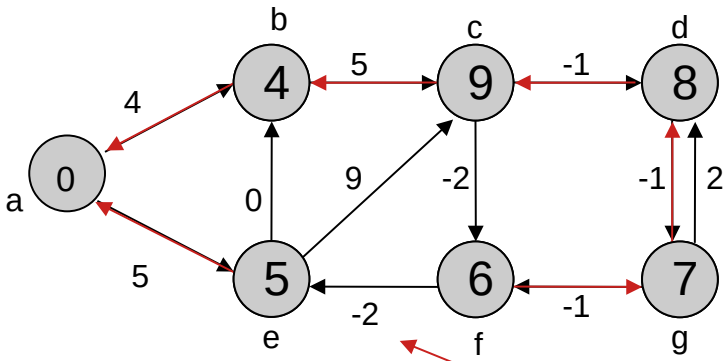
"Fila": a, b, e, c, d, f, g

Acabou a primeira passada. Está terminado?

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 1:

"Fila": a, b, e, c, d, f, g

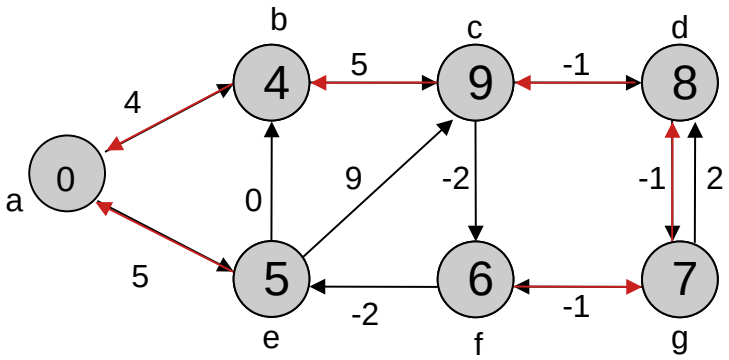
Acabou a primeira passada. Está terminado? NÃO!

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

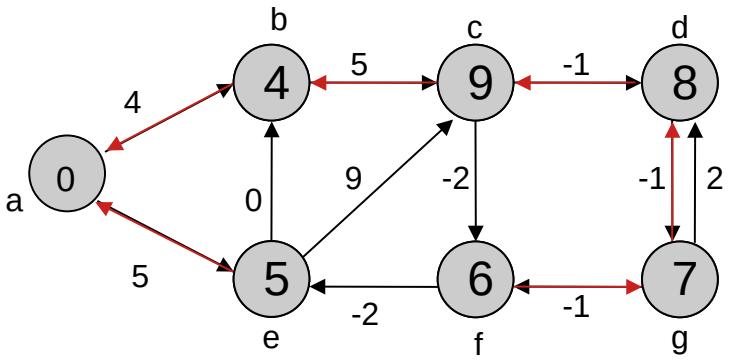
Passada 2:

"Fila": a, b, e

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 2:

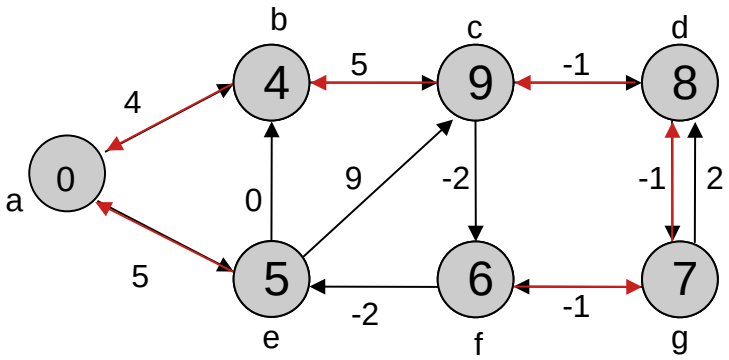
"Fila": a, b, e, c

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 2:

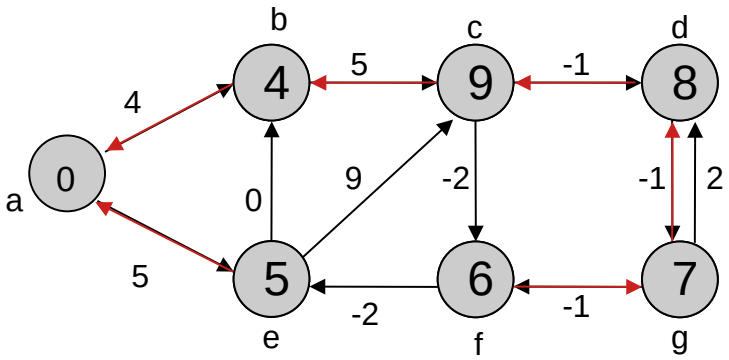
"Fila": a, b, e, c

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 2:

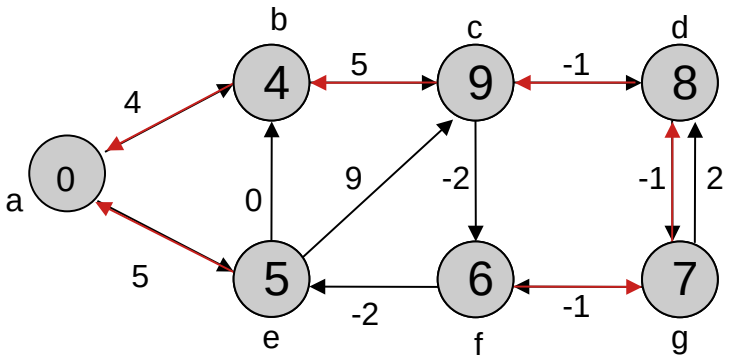
"Fila": a, b, e, c, d, f

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 2:

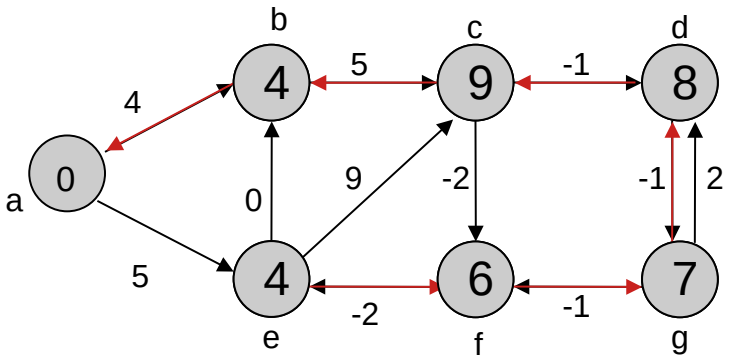
"Fila": a, b, e, c, d, f, g

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 2:

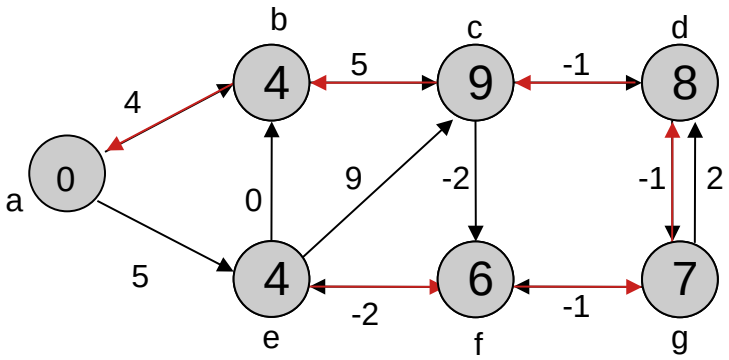
"Fila": a, b, e, c, d, f, g

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

Passada 2:

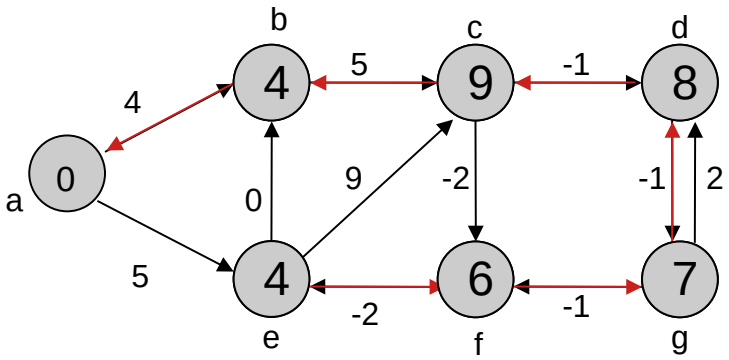
"Fila": a, b, e, c, d, f, g

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     d[v]    = d[u] + w(u, v)
3     π[v]    = u
```

Exemplo:(algoritmo retorna TRUE)



Mesmo fazendo uma inspeção das arestas em largura, não garantiria fazer tudo em um único passo (seguindo as arestas que saem de cada vértice, percorrendo os vértices em largura, passo por todas as arestas sem repeti-las em cada passada)

Ex:

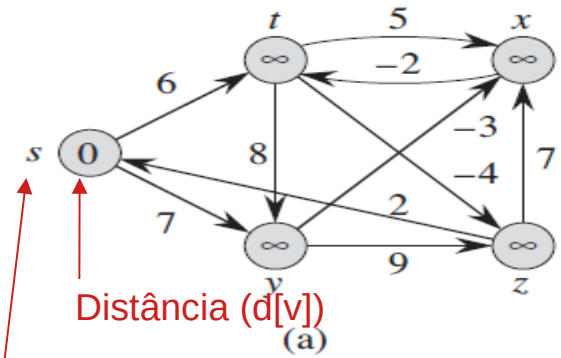
Passada 3:

Nada mais muda

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Exemplo:(algoritmo retorna TRUE)



Distância (d[v])

Rótulo do vértice

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

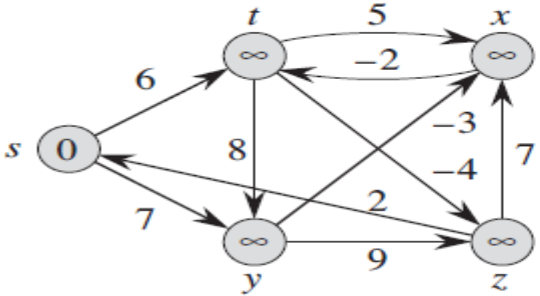
```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

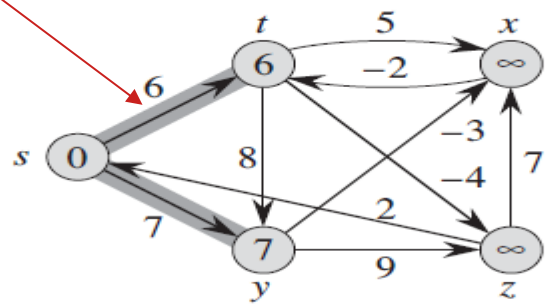
Outro exemplo (agora do Cormen):(algoritmo retorna TRUE)

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

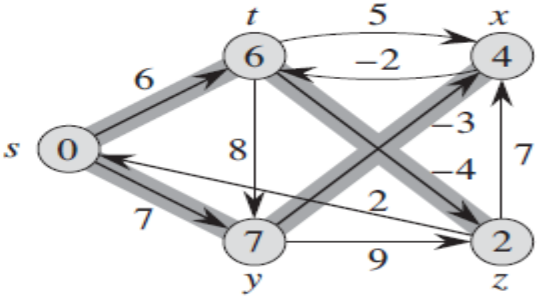
antecessor



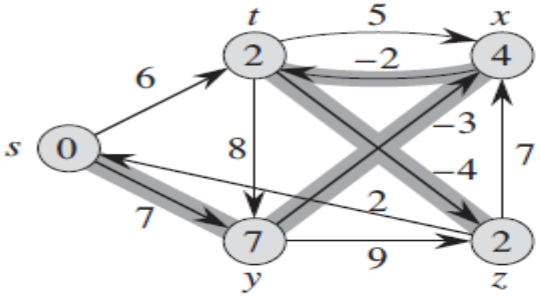
(a) (passada 1)



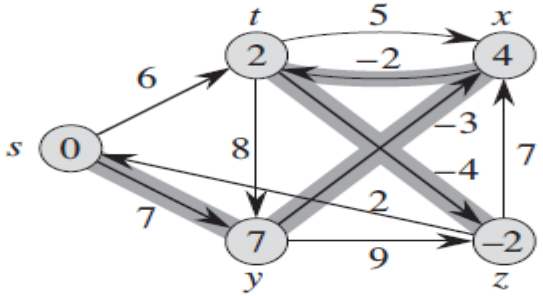
(b) (passada 1)



(c) (passada 1)



(d) (passada 1)



(e) (passada 2)

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

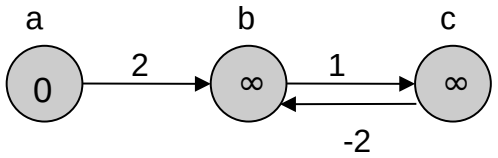
Como seria um exemplo que retornaria falso?

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v] = d[u] + w(u, v)
3       π[v] = u
```

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?

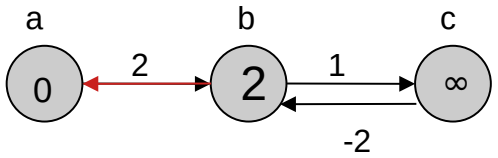



```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v]  = d[u] + w(u, v)
3      π[v]  = u
```

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?



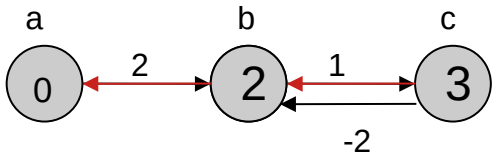
Passada 1:

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]   = d[u] + w(u, v)
3       π[v]   = u
```

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?



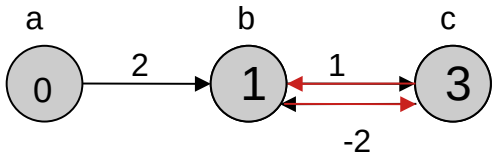
Passada 1:

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?



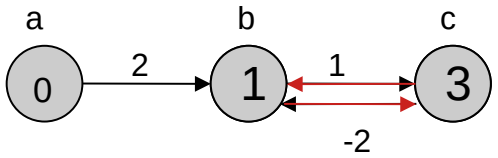
Passada 1:

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]   = d[u] + w(u, v)
3       π[v]   = u
```

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?



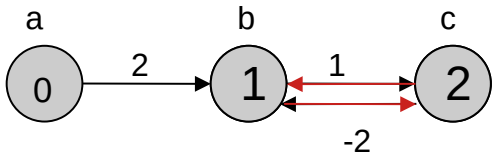
Passada 2:

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]   = d[u] + w(u, v)
3       π[v]   = u
```

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?



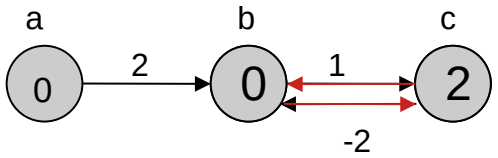
Passada 2:

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]   = d[u] + w(u, v)
3       π[v]   = u
```

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?




```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       d[v]  = d[u] + w(u, v)
3       π[v]  = u
```

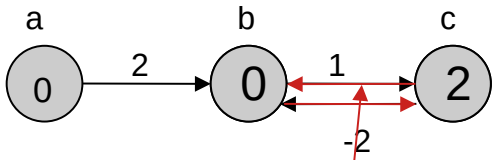
Passada 2:

Deu certo?

```
BELLMAN-FORD(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4           RELAX(u, v, w)
5   for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7           return FALSE
8   return TRUE
```

Algoritmo Bellman-Ford

Como seria um exemplo que retornaria falso?



Passada 2:

Deu certo? NÃO !!!

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2      d[v]  = d[u] + w(u, v)
3      π[v]  = u
```

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

Complexidade do Algoritmo Bellman-Ford

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Complexidade:

Complexidade do Algoritmo Bellman-Ford

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if d[v] > d[u] + w(u, v)
7            return FALSE
8  return TRUE
```

Complexidade:

L. 1: O(V)

L. 2-4: O(VA)

L. 5-7: O(A)

Total: O(VA)

Referências

Ziviani: seção 7.9 (cap 7)

Cormen: cap 24