CAMINHOS MÍNIMOS EM GRAFOS: BELLMAN-FORD (PARTE II)

Prof. Daniel Kikuti

Universidade Estadual de Maringá

22 de abril de 2015

Sumário

- Revisão
- Algoritmo de Bellman-Ford
- Propriedades
- Análise: correção e consumo de tempo
- Exercícios

Melhores momentos

- **Custo de um caminho** é a soma do peso das arestas no caminho.
- **Caminho mínimo** de u a v é o caminho de menor custo ($\delta(u, v)$) dentre todos os possíveis caminhos de u a v.

Melhores momentos

- **Custo de um caminho** é a soma do peso das arestas no caminho.
- **Caminho mínimo** de u a v é o caminho de menor custo $(\delta(u, v))$ dentre todos os possíveis caminhos de u a v.
- **Propriedade importante**. Qualquer subcaminho de um caminho mínimo é um caminho mínimo.

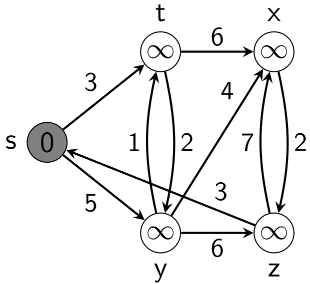
Melhores momentos

- **Custo de um caminho** é a soma do peso das arestas no caminho.
- **Caminho mínimo** de u a v é o caminho de menor custo ($\delta(u, v)$) dentre todos os possíveis caminhos de u a v.
- **Propriedade importante**. Qualquer subcaminho de um caminho mínimo é um caminho mínimo.
- Algoritmo de Dijkstra
  - Todas as arestas com pesos positivos
  - Estratégia gulosa

Melhores momentos

Funcionamento do Dijkstra

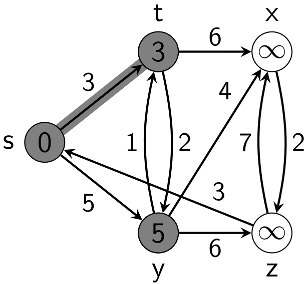
```
dijkstra(G, w, s)
initialize-single-source(G, s)
S = ∅
Q = G.V
enquanto Q != ∅
   u = extract-min(Q)
   S = S U u
   para cada vértice v em u.adj
      relax(u, v, w)
```



Melhores momentos

Funcionamento do Dijkstra

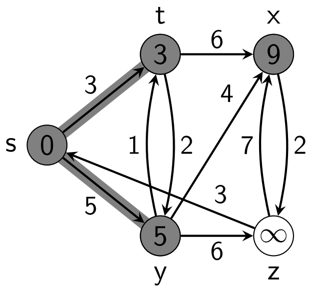
```
dijkstra(G, w, s)
initialize-single-source(G, s)
S = ∅
Q = G.V
enquanto Q != ∅
    u = extract-min(Q)
    S = S U u
    para cada vértice v em u.adj
        relax(u, v, w)
```



Melhores momentos

Funcionamento do Dijkstra

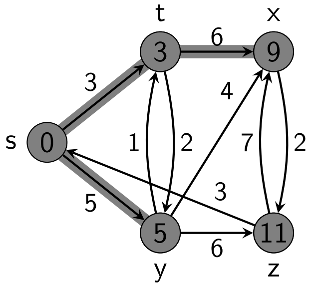
```
dijkstra(G, w, s)
initialize-single-source(G, s)
S = ∅
Q = G.V
enquanto Q != ∅
    u = extract-min(Q)
    S = S U u
    para cada vértice v em u.adj
        relax(u, v, w)
```



Melhores momentos

Funcionamento do Dijkstra

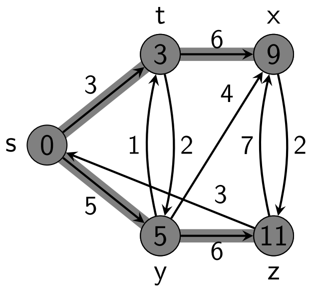
```
dijkstra(G, w, s)
initialize-single-source(G, s)
S = ∅
Q = G.V
enquanto Q != ∅
   u = extract-min(Q)
   S = S U u
   para cada vértice v em u.adj
      relax(u, v, w)
```



Melhores momentos

Funcionamento do Dijkstra

```
dijkstra(G, w, s)
initialize-single-source(G, s)
S = ∅
Q = G.V
enquanto Q != ∅
    u = extract-min(Q)
    S = S U u
    para cada vértice v em u.adj
        relax(u, v, w)
```



Complexidade de tempo

Tempo total $= O((V + E) \lg V)$ usando um heap.

Introdução

O algoritmo de Bellman-Ford

- Permite arestas com pesos negativos.
- Computa v.d e v.pred para todos os vértices $v \in V$.
- Devolve **verdadeiro** se nenhum ciclo de peso negativo é alcançável de s, **falso** caso contrário.

Introdução

O algoritmo de Bellman-Ford

- Permite arestas com pesos negativos.
- Computa v.d e v.pred para todos os vértices $v \in V$.
- Devolve **verdadeiro** se nenhum ciclo de peso negativo é alcançável de s, **falso** caso contrário.

Ideia do algoritmo

Relaxar todas as arestas $|V| - 1$ vezes. Por quê?

O Algoritmo

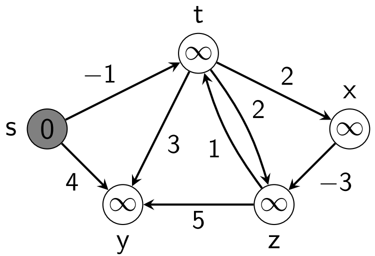
Algoritmo de Bellman-Ford

```
bellman-ford(G, w, s)
initialize-single-source(G, s)
para i ← 1 até |V| − 1 faça
   para cada aresta (u, v) ∈ E faça
      relax(u,v,w);
para cada aresta (u, v) ∈ E faça
   se v.d > u.d + w(u, v) então
      devolva falso
devolva verdadeiro
```

Um exemplo

Um grafo com $|V| = 5$ e $|E| = 8$

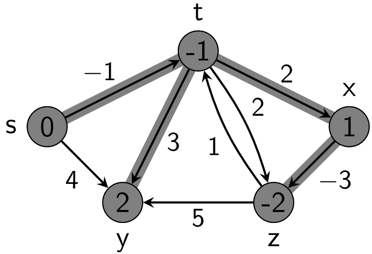
```
bellman-ford(G, w, s)
initialize-single-source(G, s)
para i ← 1 até |V| − 1 faça
    para cada aresta (u, v) ∈ E faça
        relax(u,v,w);
para cada aresta (u, v) ∈ E faça
    se v.d > u.d + w(u, v) então
        devolva falso
devolva verdadeiro
```



Um exemplo

Um grafo com $|V| = 5$ e $|E| = 8$

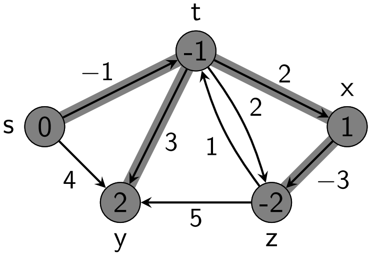
```
bellman-ford(G, w, s)
initialize-single-source(G, s)
para i ← 1 até |V| − 1 faça
    para cada aresta (u, v) ∈ E faça
        relax(u,v,w);
para cada aresta (u, v) ∈ E faça
    se v.d > u.d + w(u, v) então
        devolva falso
devolva verdadeiro
```



Um exemplo

Um grafo com $|V| = 5$ e $|E| = 8$

```
bellman-ford(G, w, s)
initialize-single-source(G, s)
para i ← 1 até |V| − 1 faça
   para cada aresta (u, v) ∈ E faça
      relax(u,v,w);
para cada aresta (u, v) ∈ E faça
   se v.d > u.d + w(u, v) então
      devolva falso
devolva verdadeiro
```



Complexidade

Tempo total $= \Theta(VE)$.

Propriedade

Relaxamento de caminho (Lema 24.15)

Seja $p = \langle v_0, v_1, \ldots, v_k \rangle$ um caminho mínimo de $s = v_0$ até v_k. Se a função `relax` for chamada na ordem $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ mesmo que intercalada com outros relaxamentos, então $v_k.d = \delta(s, v_k)$.

Propriedade

Relaxamento de caminho (Lema 24.15)

Seja $p = \langle v_0, v_1, \ldots, v_k \rangle$ um caminho mínimo de $s = v_0$ até v_k. Se a função `relax` for chamada na ordem $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ mesmo que intercalada com outros relaxamentos, então $v_k.d = \delta(s, v_k)$.

Demonstração (indução)

Após a i-ésima aresta do caminho p ser relaxada, temos que $v_i.d = \delta(s, v_i)$. [Hipótese]

- **Base:** $i = 0$ antes de qualquer aresta ter sido relaxada, da inicialização temos $v_0.d = s.d = 0 = \delta(s, v_0)$. O valor de $s.d$ nunca muda após a inicialização.
- **Passo:** Assumimos que $v_{i-1}.d = \delta(s, v_{i-1})$ e analisaremos o relaxamento da aresta (v_{i-1}, v_i).

Relaxamento de caminho (Lema 24.15)

Continuação da demonstração do passo indutivo

Se $v_{i-1}.d = \delta(s, v_{i-1})$ em algum ponto antes de relaxar a aresta (v_{i-1}, v_i), então

$$\begin{aligned} v_i.d &\leq v_{i-1}.d + w(v_{i-1}, v_i) && \text{(rever relaxamento)} \\ &= \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \\ &= \delta(s, v_i) && \text{(lema do subcaminho)} \end{aligned}$$

Pela propriedade do limite superior ($v.d \geq \delta(s, v)$ – ver demonstração desta propriedade no livro texto), concluímos que $v_i.d = \delta(s, v_i)$, e a igualdade é mantida a partir de então.

Por que funciona?

Teorema

Bellman-Ford computa corretamente os caminhos mínimos a partir de um vértice s.

Por que funciona?

Teorema

Bellman-Ford computa corretamente os caminhos mínimos a partir de um vértice s.

Demonstração

Seja v alcançável de s e $p = \langle v_0, v_1, \ldots, v_k \rangle$ um caminho de s a v, onde $v_0 = s$ e $v_k = v$.

- Como p é acíclico, ele possui menos que $|V| - 1$ arestas, portanto $k \leq |V| - 1$.
- Cada iteração do primeiro laço (linhas 2–4) relaxa todas as arestas. Na primeira iteração (v_0, v_1), na segunda (v_1, v_2) e na k-ésima (v_{k-1}, v_k). Pela propriedade de relaxamento de caminho, temos então que $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$.

Valores que o algoritmo devolve

Devolve verdadeiro

Suponha que não há um ciclo de peso negativo alcançável de s.
Ao fim do primeiro laço, para todas as aresta $(u, v) \in E$,

$$\begin{aligned} v.d &= \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \\ &= u.d + w(u, v) \end{aligned}$$

Então Bellman-Ford devolve verdadeiro.

Valores que o algoritmo devolve

Devolve falso

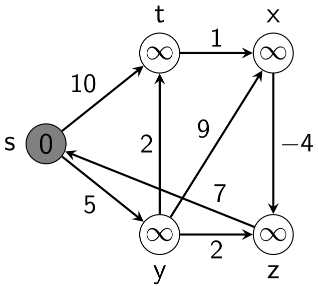
- Suponha que G contém um ciclo de custo negativo $c = \langle v_0, v_1, \ldots, v_k \rangle$, onde $v_0 = v_k$ e $\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$.
- Assuma com o propósito de contradição que Bellman-Ford devolve verdadeiro. Então, $v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$ para $i = 1, \ldots, k$.
- Somando as inequações do ciclo nos dá:

$$\begin{array}{rcl} \sum_{i=1}^{k} v_i.d & \leq & \sum_{i=1}^{k}(v_{i-1}.d + w(v_{i-1}, v_i)) \\ & = & \sum_{i=1}^{k} v_{i-1}.d + \sum_{i=1}^{k} w(v_{i-1}, v_i) \end{array}$$

- Cada vértice em c aparece exatamente uma vez e portanto $\sum_{i=1}^{k} v_i.d = \sum_{i=1}^{k} v_{i-1}.d$.
- O que no leva a concluir que $\sum_{i=1}^{k} w(v_{i-1}, v_i) \geq 0$ (contradição).

Exercício 1

Faça uma execução passo a passo do algoritmo de Bellman-Ford para a figura abaixo.



Exercício 2

Em um grafo direcionado acíclico (DAG) é possível encontrar os caminhos mínimos de tempo $\Theta(V + E)$. Desenvolva um algoritmo que encontre os caminhos mínimos para este tipo especial de grafos. Argumente por que seu algoritmo está correto.