Caminhos mais curtos

CLRS Secs 25.2

Caminhos mais curtos

$G = (V, E)$: grafo orientado
Função c que atribui um comprimento c_e para cada $e \in E$.

Caminhos mais curtos

$G = (V, E)$: grafo orientado
Função c que atribui um comprimento c_e para cada $e \in E$.

Para vértices u e v, a distância de u a v é o comprimento de um caminho de u a v de comprimento mínimo.

Caminhos mais curtos

$G = (V, E)$: grafo orientado
Função c que atribui um comprimento c_e para cada $e \in E$.

Para vértices u e v, a distância de u a v é o comprimento de um caminho de u a v de comprimento mínimo.

Problema 1: Dados G, c e um vértice s de G,
encontrar a distância de s a cada vértice de G.

Caminhos mais curtos

$G = (V, E)$: grafo orientado
Função c que atribui um comprimento c_e para cada $e \in E$.

Para vértices u e v, a distância de u a v é o comprimento de um caminho de u a v de comprimento mínimo.

Problema 1: Dados G, c e um vértice s de G,
encontrar a distância de s a cada vértice de G.

Problema 2: Dados G e c,
encontrar a distância entre todo par de vértices de G.

Caminhos mais curtos

$G = (V, E)$: grafo orientado
Função c que atribui um comprimento c_e para cada $e \in E$.

Para vértices u e v, a distância de u a v é o comprimento de um caminho de u a v de comprimento mínimo.

Problema 1: Dados G, c e um vértice s de G, encontrar a distância de s a cada vértice de G.

Problema 2: Dados G e c, encontrar a distância entre todo par de vértices de G.

Algoritmo de Dijkstra: comprimentos não negativos

Caminhos mais curtos

$G = (V, E)$: grafo orientado
Função c que atribui um comprimento c_e para cada $e \in E$.

Para vértices u e v, a distância de u a v é o comprimento de um caminho de u a v de comprimento mínimo.

Problema 1: Dados G, c e um vértice s de G,
encontrar a distância de s a cada vértice de G.

Problema 2: Dados G e c,
encontrar a distância entre todo par de vértices de G.

Algoritmo de Dijkstra: comprimentos não negativos

Algoritmo de Floyd-Warshall: sem circuitos negativos

Revisão da aula passada

P: caminho mais curto de s a t

Subestrutura ótima (sem circuitos negativos):
Subcaminhos de P são caminhos mais curtos.

Revisão da aula passada

P: caminho mais curto de s a t

Subestrutura ótima (sem circuitos negativos):
Subcaminhos de P são caminhos mais curtos.

Se isto não vale, então há circuito negativo no grafo!

Revisão da aula passada

P: caminho mais curto de s a t

Subestrutura ótima (sem circuitos negativos):
Subcaminhos de P são caminhos mais curtos.

Se isto não vale, então há circuito negativo no grafo!

Prova feita na aula.

Revisão da aula passada

P: caminho mais curto de s a t

Subestrutura ótima (sem circuitos negativos):
Subcaminhos de P são caminhos mais curtos.

Se isto não vale, então há circuito negativo no grafo!

Prova feita na aula.

Faltou argumentar porque o Dijkstra funciona.

Algoritmo de Dijkstra

π: representa os caminhos mínimos até s
d: guarda a distância de cada vértice a s.

DIJKSTRA (G, c, s)

para $v \in V(G)$ **faça** $d[v] \leftarrow \infty$
$d[s] \leftarrow 0$ $\quad \pi[s] \leftarrow$ NIL
$Q \leftarrow V(G)$ $\quad \triangleright$ chave de v é $d[v]$
enquanto $Q \neq \emptyset$ **faça**
    $u \leftarrow$ EXTRACT-MIN(Q)
    para cada $v \in \mathrm{adj}(u)$ **faça**
        se $v \in Q$ e $d[u] + c(uv) < d[v]$
            então $\pi[v] \leftarrow u$ $\quad d[v] \leftarrow d[u] + c(uv)$
devolva (π, d)

Algoritmo de Dijkstra

π: representa os caminhos mínimos até s

d: guarda a distância de cada vértice a s.

DIJKSTRA (G, c, s)

para $v \in V(G)$ **faça** $d[v] \leftarrow \infty$

$d[s] \leftarrow 0$ $\pi[s] \leftarrow$ NIL

$Q \leftarrow V(G)$ ▷ chave de v é $d[v]$

enquanto $Q \neq \emptyset$ **faça**

  $u \leftarrow$ EXTRACT-MIN(Q)

  para cada $v \in \mathrm{adj}(u)$ **faça**

    se $v \in Q$ e $d[u] + c(uv) < d[v]$

      então $\pi[v] \leftarrow u$ $d[v] \leftarrow d[u] + c(uv)$

devolva (π, d)

$d[u]$: comprimento de um caminho mínimo de s a u
cujos vértices internos estão fora de Q

Algoritmo de Dijkstra

π: representa os caminhos mínimos até s
d: guarda a distância de cada vértice a s.

```
DIJKSTRA (G, c, s)
   para v ∈ V(G) faça  d[v] ← ∞
   d[s] ← 0    π[s] ← NIL
   Q ← V(G)      ▷ chave de v é d[v]
   enquanto Q ≠ ∅ faça
       u ← EXTRACT-MIN(Q)
       para cada v ∈ adj(u) faça
           se v ∈ Q e d[u] + c(uv) < d[v]
               então  π[v] ← u  d[v] ← d[u] + c(uv)
   devolva (π, d)
```

Invariantes: $d[u] = \delta(s,u)$ se $u \notin Q$
$d[u] \geq \delta(s,u)$ se $u \in Q$

Problema 2

$G = (V, E)$: grafo orientado

Função c que atribui um comprimento c_e para cada $e \in E$.

Problema 2: Dados G e c,
encontrar a distância entre todo par de vértices de G.

Problema 2

$G = (V, E)$: grafo orientado

Função c que atribui um comprimento c_e para cada $e \in E$.

Problema 2: Dados G e c,
encontrar a distância entre todo par de vértices de G.

Hipótese:
Não há circuito de comprimento negativo em G.

Problema 2

$G = (V, E)$: grafo orientado

Função c que atribui um comprimento c_e para cada $e \in E$.

Problema 2: Dados G e c,
encontrar a distância entre todo par de vértices de G.

Hipótese:
Não há circuito de comprimento negativo em G.

Algoritmo de Floyd-Warshall: programação dinâmica

Subestrutura ótima

Para $k \in [n]$, seja P um caminho mais curto de s a t
cujos vértices internos estão todos em $[k]$.

Subestrutura ótima

Para $k \in [n]$, seja P um caminho mais curto de s a t
cujos vértices internos estão todos em $[k]$.

Se P não contém k como vértice interno,

então P é um caminho mais curto de s a t
cujos vértices internos estão todos em $[k-1]$.

Subestrutura ótima

Para $k \in [n]$, seja P um caminho mais curto de s a t
cujos vértices internos estão todos em $[k]$.

Se P não contém k como vértice interno,

então P é um caminho mais curto de s a t
cujos vértices internos estão todos em $[k-1]$.

senão P é a concatenação de dois caminhos,
um caminho mais curto de s a k, outro de k a t,
ambos com vértices internos em $[k-1]$.

Subestrutura ótima

Para $k \in [n]$, seja P um caminho mais curto de s a t
cujos vértices internos estão todos em $[k]$.

Se P não contém k como vértice interno,

então P é um caminho mais curto de s a t
cujos vértices internos estão todos em $[k-1]$.

senão P é a concatenação de dois caminhos,
um caminho mais curto de s a k, outro de k a t,
ambos com vértices internos em $[k-1]$.

Floyd-Warshall: Usa caminhos mínimos
com vértices intermediários em $[k-1]$ para obter
caminhos mínimos com vértices intermediários em $[k]$.

Recorrência

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

Recorrência

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

$$D^k[i][j] = \begin{cases} c_{ij} & \text{se } k = 0 \\ \min\{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]\} & \text{se } k \geq 1 \end{cases}$$

Recorrência

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

$$D^k[i][j] = \begin{cases} c_{ij} & \text{se } k = 0 \\ \min\{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]\} & \text{se } k \geq 1 \end{cases}$$

A matrix D^n tem a resposta do Problema 2.

Recorrência

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

$$D^k[i][j] = \begin{cases} c_{ij} & \text{se } k = 0 \\ \min\{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]\} & \text{se } k \geq 1 \end{cases}$$

A matrix D^n tem a resposta do Problema 2.

Algoritmo de Floyd-Warshall: calcula D^n pela recorrência.

Algoritmo de Floyd-Warshall

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

```
FLOYD-WARSHALL (G, c)
n ← |V(G)|
D^0 ← c
para k ← 1 até n faça
    para i ← 1 até n faça
        para j ← 1 até n faça
            D^k[i][j] = min{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]}
devolva D^n
```

Algoritmo de Floyd-Warshall

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

FLOYD-WARSHALL (G, c)

```
n ← |V(G)|
D^0 ← c
para k ← 1 até n faça
    para i ← 1 até n faça
        para j ← 1 até n faça
            D^k[i][j] = min{D^{k−1}[i][j], D^{k−1}[i][k] + D^{k−1}[k][j]}
devolva D^n
```

Consumo de tempo: $\Theta(n^3)$

Algoritmo de Floyd-Warshall

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k]$.

FLOYD-WARSHALL (G, c)

$n \leftarrow |V(G)|$
$D^0 \leftarrow c$
para $k \leftarrow 1$ **até** n **faça**
    para $i \leftarrow 1$ **até** n **faça**
        para $j \leftarrow 1$ **até** n **faça**
            $D^k[i][j] = \min\{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]\}$
devolva D^n

Consumo de tempo: $\Theta(n^3)$

Com Dijkstra: $\mathrm{O}(n^3)$
$\mathrm{O}(nm \lg n)$ com fila de prioridade
$\mathrm{O}(n(m + n \lg n))$ com Fibonacci heap.

Algoritmo de Floyd-Warshall

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G
com vértices intermediários em $[k-1]$.

```
FLOYD-WARSHALL (G, c)
n ← |V(G)|
D^0 ← c
para k ← 1 até n faça
    para i ← 1 até n faça
        para j ← 1 até n faça
            D^k[i][j] = min{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]}
devolva D^n
```

E os caminhos mais curtos?

Algoritmo de Floyd-Warshall

$D^k[i][j]$: comprimento de um caminho mínimo de i a j em G com vértices intermediários em $[k-1]$.

```
FLOYD-WARSHALL (G, c)
n ← |V(G)|
D^0 ← c
para k ← 1 até n faça
    para i ← 1 até n faça
        para j ← 1 até n faça
            D^k[i][j] = min{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]}
devolva D^n
```

E os caminhos mais curtos?

Guarde informação durante o processo acima para obter um caminho mais curto entre quaisquer dois vértices de G.

Simulação

$$D^0 = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^0 = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^1 = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^1 = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^1 = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^2 = \begin{pmatrix} 0 & 3 & 8 & \color{red}{4} & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \color{red}{5} & \color{red}{11} \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^2 = \begin{pmatrix} 0 & 3 & 8 & \color{red}{4} & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \color{red}{5} & \color{red}{11} \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^2 = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^3 = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^3 = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Simulação

$$D^3 = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^4 = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Simulação

$$D^4 = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Simulação

$$D^4 = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$D^5 = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Simulação

Distâncias:

$$D^5 = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Como obter os caminhos?

Simulação

Distâncias:

$$D^5 = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Como obter os caminhos?

Exercício!