Caminhos mais curtos

CLRS Secs 24.3 e 25.2

Caminhos mais curtos

Dados:

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

- O comprimento de um caminho é a soma dos comprimentos de suas arestas.
- Para vértices u e v, a distância de u a v é o menor comprimento de um caminho de u a v.

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

Circuitos negativos

Dados:

$G = (V, E)$: grafo dirigido
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 entre u e v de comprimento mínimo.

Quando há um circuito de comprimento negativo no grafo, a distância entre certos vértices pode ficar mal-definida.

Poderíamos dar "voltas" num circuito negativo, cada vez obtendo um "caminho" de comprimento menor.

Assim definimos a distância $\delta(u, v)$ como
$-\infty$, caso exista circuito negativo alcançavel de u,
e o comprimento de um caminho mais curto de u a v c.c.

Propriedades

P: caminho mais curto de s a t

Subestrutura ótima:
Subcaminhos de P são caminhos mais curtos.

Lema: Dados G e c, seja $P = \langle v_1, \ldots, v_k \rangle$ um caminho mais curto em G de v_1 a v_k. Para todo $1 \leq i \leq j \leq k$, $P_{ij} := \langle v_i, \ldots, v_j \rangle$ é um caminho mais curto de v_i a v_j.

Corolário: Para G e c, se o último arco de um caminho mais curto de s a t é o arco ut, então $\delta(s, t) = \delta(s, u) + c(ut)$.

Lema: Para G, c e s,
$\delta(s, v) \leq \delta(s, u) + c(uv)$ para todos os arcos uv.

Algoritmo de Dijkstra

π: representa os caminhos mínimos até s
d: guarda estimativa a distância de s ao vértice

DIJKSTRA (G, c, s)
1 **para** $v \in V(G)$ **faça** $\quad v.d \leftarrow \infty \quad v.\pi \leftarrow \mathrm{nil}$
2 $\quad s.d \leftarrow 0$
3 $\quad Q \leftarrow V(G)$ ▷ fila de prioridade: chave de v é $v.d$
4 **enquanto** $Q \neq \emptyset$ **faça**
5 $\quad\quad u \leftarrow$ EXTRACT-MIN(Q)
6 $\quad\quad$ **para cada** $v \in \mathrm{adj}(u)$ **faça**
7 $\quad\quad\quad$ **se** $v \in Q$ e $v.d > u.d + c(uv)$
8 $\quad\quad\quad\quad$ **então** $v.\pi \leftarrow u \quad v.d \leftarrow u.d + c(uv)$
9 π e d são a informação desejada.

u d $\quad$ s u
Invariantes:
u d δ s u u
u d δ s u u

Algoritmo de Dijkstra

Complexidade:

DIJKSTRA (G, c, s)
1 **para** $v \in V(G)$ **faça** $\quad v.d \leftarrow \infty \quad v.\pi \leftarrow \mathrm{nil}$
2 $\quad s.d \leftarrow 0$
3 $\quad Q \leftarrow V(G)$ ▷ fila de prioridade: chave de v é $v.d$
4 **enquanto** $Q \neq \emptyset$ **faça**
5 $\quad\quad u \leftarrow$ EXTRACT-MIN(Q)
6 $\quad\quad$ **para cada** $v \in \mathrm{adj}(u)$ **faça**
7 $\quad\quad\quad$ **se** $v \in Q$ e $v.d > u.d + c(uv)$
8 $\quad\quad\quad\quad$ **então** $v.\pi \leftarrow u \quad v.d \leftarrow u.d + c(uv)$
9 π e d são a informação desejada.

Se Q for uma lista simples:
Linha 3 e EXTRACT-MIN : $\mathrm{O}(n)$

Consumo de tempo do Dijkstra: $\mathrm{O}(n^2)$

Algoritmo de Dijkstra

Complexidade:

DIJKSTRA (G, c, s)
1 **para** $v \in V(G)$ **faça** $\quad v.d \leftarrow \infty \quad v.\pi \leftarrow \mathrm{nil}$
2 $\quad s.d \leftarrow 0$
3 $\quad Q \leftarrow V(G)$ ▷ fila de prioridade: chave de v é $v.d$
4 **enquanto** $Q \neq \emptyset$ **faça**
5 $\quad\quad u \leftarrow$ EXTRACT-MIN(Q)
6 $\quad\quad$ **para cada** $v \in \mathrm{adj}(u)$ **faça**
7 $\quad\quad\quad$ **se** $v \in Q$ e $v.d > u.d + c(uv)$
8 $\quad\quad\quad\quad$ **então** $v.\pi \leftarrow u \quad v.d \leftarrow u.d + c(uv)$
9 π e d são a informação desejada.

Se Q for implementada com um heap:
Inicialização: $\mathrm{O}(n)$ $\quad$ EXTRACT-MIN e DECREASE-KEY : $\mathrm{O}(\lg n)$

DECREASE-KEY : linha 8

Consumo de tempo do Dijkstra: $\mathrm{O}(m \lg n)$
Consumo de tempo com Fibonacci heap: $\mathrm{O}(m + n \lg n)$

Algoritmo A^*

Dados G, c, s, t, achar um caminho mínimo de s a t.
Suponha que exista, para cada v uma estimativa

$$h(v) \leq \delta(v, t).$$

Organize a fila de prioridade Q pelo mínimo de

$$f(v) = \delta(s, v) + h(v).$$

Consistência: se para cada aresta $u \to v$

$$h(u) \leq c(u \to v) + h(v),$$

o algoritmo termina com um caminho ótimo de s a t.

Algoritmo A^*

Quando se aplica:

G é subgrafo de um grafo no qual é "fácil" calcular distâncias.

Por exemplo, num mapa, a distância euclidiana é uma estimativa.

Problema 2

Dados:

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

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

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

Propriedades

P: caminho mais curto de s a t

Subestrutura ótima:
Subcaminhos de P são caminhos mais curtos.

Lema: Dados G e c, seja $P = \langle v_1, \ldots, v_k \rangle$ um caminho mais curto em G de v_1 a v_k. Para todo $1 \le i \le j \le k$, $P_{ij} := \langle v_i, \ldots, v_j \rangle$ é um caminho mais curto de v_i a v_j.

Vamos identificar $V = [n] = \{1, 2, \ldots, n\}$.
Para $k \in [n]$, seja P um caminho mais curto de s a t cujos vértices internos estão todos em $[k]$.

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

Algoritmo de Floyd-Warshall

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

Seja P um caminho mínimo de i a j em G
com vértices intermediários em $[k]$.

Se P não usa k como vértice intermediário,
então P é um caminho mínimo de i a j em G
com vértices intermediários em $[k-1]$.
senão $P = P' \cdot P''$ onde
P' é um caminho mínimo de i a k em G
com vértices intermediários em $[k-1]$ e
P'' é um caminho mínimo de k a j em G
com vértices intermediários em $[k-1]$.

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 ao 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 $[k-1]$.

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
```

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

Dijkstra: $\mathrm{O}(nm \lg n)$
$\mathrm{O}(n(m + n \lg n))$ 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?

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

Simulação

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

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

Simulação

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

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

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

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

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

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?

Exercício!