Clase 12 MA3705. 26 de octubre 2020.

Tabla

- Dijkstra (revisitado)
- Calcular distancia en todos los pares

Dijkstra revisitado

Dijkstra.

Algoritmo de Dijkstra (Dijkstra 1959):

Entrada: $G = (V, E)$ dirigido, $s \in V$, $\ell \colon E \to \mathbb{R}^+$
para $v \in V$ **hacer** $d(s, v) = +\infty$. $T(v) \leftarrow \infty$. $\text{Padre}(v) \leftarrow \text{Null}$
$\text{No-Visitados} \leftarrow V$.
$T(s) \leftarrow 0$.
mientras $\text{Tmin} \leftarrow \min_{v \in \text{No-Visitados}} T(v) < +\infty$ **hacer**
  Elegir $v \in \text{No-Visitados}$ con $T(v) = \text{Tmin}$.
  $d(s, v) \leftarrow T(v)$
  $\text{No-Visitados} \leftarrow \text{No-Visitados} - v$
  para $w \in N^+(v) \cap \text{No-visitado}$ **hacer**
    si $T(v) + \ell(vw) < T(w)$ **entonces**
      $T(w) \leftarrow T(v) + \ell(vw), \text{Padre}(w) \leftarrow v$
    fin
  fin
fin
devolver d, Padre

Correctitud de Dijkstra

Inicialización

1 $(\forall v \in V) \quad d(s,v) \leftarrow +\infty,\ T_1(v) \leftarrow +\infty.$
2 $\text{NoVis} \leftarrow V.\ T_1(s) \leftarrow 0.$

Iteración $i+1$

$T_{i+1} \equiv T_i$
1 $\text{Tmin} \leftarrow \min_{v \in \text{NoVis}} T_i(v)$
2 **si** $\text{Tmin} = +\infty$ **entonces** terminar
3 Elegir $v^* \in \text{NoVis}$ con $T_i(v^*) = \text{Tmin}.$
4 $\text{NoVis} \leftarrow \text{NoVis} - v^*$
5 $d(s,v^*) \leftarrow T_i(v^*)$
6 $(\forall w \in N^+(v^*) \cap \text{NoVis})$
$T_{i+1}(w) \leftarrow \min(T_i(w), T_i(v^*) + \ell(vw))$

Teorema: Si H_i es el grafo de los arcos con colas visitadas al principio de la iteración i

$\forall v$ visitado:
$d(s,v) = \text{dist}_G(s,v) = \text{dist}_{H_i}(s,v)$
$\forall v$ No visitado:
$T(v) = \text{dist}_{H_i}(s,v)$

Demostración

Teorema

$\forall v$ visitado:
$d(s,v) = \text{dist}_G(s,v) = \text{dist}_{H_i}(s,v)$
$\forall v$ No visitado: $T(v) = \text{dist}_{H_i}(s,v)$

Cambios iteración $i+1$

$d(s,v^*) = T_i(v^*)$
$(\forall w \in N^+(v^*) \cap \text{NoVis})$
$T_{i+1}(w) \leftarrow \text{mín}(T_i(w), T_i(v^*) + \ell(vw))$

Demostración:

1. Si v visitado en $\leq i$.

$d(s,v) = T_i(v) = \text{dist}_{H_i}(s,v) \geq \text{dist}_{H_{i+1}}(s,v) \geq \text{dist}_G(s,v) = \text{dist}_{H_i}(s,v).$

2. Si $v = v^*$

$d(s,v^*) = T_i(v^*) = \text{dist}_{H_i}(s,v^*) = \text{dist}_{H_{i+1}}(s,v^*) \geq \text{dist}_G(s,v^*) = \ell(P)$
con P camino mínimo de s a v^*.
Si P no usa arcos fuera de H_i, entonces $\ell(P) = \text{dist}_{H_i}(s,v^*)$
Si P usa arcos fuera de H_i, entonces .. $\ell(P) \geq$

Demostración

Teorema

$\forall v$ visitado:
$d(s,v) = \mathsf{dist}_G(s,v) = \mathsf{dist}_{H_i}(s,v)$
$\forall v$ No visitado: $T(v) = \mathsf{dist}_{H_i}(s,v)$

Cambios iteración $i+1$

$d(s,v^*) = T_i(v^*)$
$(\forall w \in N^+(v^*) \cap \textsc{NoVis})$
$\quad T_{i+1}(w) \leftarrow \min(T_i(w), T_i(v^*) + \ell(vw))$

Demostración:

3. Si v no visitado, $v \notin N^+(v^*)$
$T_{i+1}(v) = T_i(v) = \mathsf{dist}_{H_i}(s,v) = \mathsf{dist}_{H_{i+1}}(s,v)$

4. Si v no visitado, $v \in N^+(v^*)$
$T_{i+1}(v) = \min(T_i(v), T_i(v^*) + \ell(v^*v)) = \min(\mathsf{dist}_{H_i}(s,v), d(s,v^*) + \ell(v^*v)) \geq \mathsf{dist}_{H_{i+1}}(s,v)$

Resumen de Caminos mínimos desde un nodo s en grafos dirigidos

Función de largo	Algoritmo	Complejidad
Cualquiera en DAG	Orden Topológico + PD	$O(n+m)$
Unitaria	BFS	$O(m+n)$
Enteros entre 1 y M	BFS	$O(M(m+n))$
No negativos	Dijkstra	$O(m+n\log n)$
Conservativos	Bellman Ford	$O(n(n+m))$

¡Ojo: Estos algoritmos también funcionan en grafos no dirigidos (basta bidirigir aristas)!

Distancias en todos los pares

Como calcular las distancias $d(u, v)$ para todos los pares u, v

Idea 1:

Función de largo	Algoritmo	Complejidad
Cualquiera en DAG	Orden Topológico + PD	
Unitaria	BFS	
Enteros entre 1 y M	BFS	
No negativos	Dijkstra	
Conservativos	Bellman Ford	

Para largos conservativos en grafos densos ($m = \Theta(n^2)$), esto es:
que es demasiado lento.

Programación dinámica de Bellman-Ford

Bellman Ford usa (llamando $D^{(k)}(a,b) = d_{\leq k}(a,b)$)

$$D^{(i+1)}(s,t) = \min\left(D^{(i)}(s,t), \min_{u \in N^-(t)} D^{(i)}(s,u) + \ell(u,t)\right)$$

Esto equivale a:

$$\begin{aligned} D^{(i+1)}(s,t) &= \min\left(D^{(i)}(s,t), \min_{u \in N^-(t)} D^{(i)}(s,u) + D^{(1)}(u,t)\right) \\ &= \min_{u \in N^-(t)+t} \quad D^{(i)}(s,u) + D^{(1)}(u,t) \\ &= \min_{u \in V} \quad D^{(i)}(s,u) + D^{(1)}(u,t). \end{aligned}$$

Producto de matrices en semianillo $(\mathbb{R}_+{}^{\infty}, \text{mín}, +)$

Si A, B son matrices en $\mathbb{F}^{V\times V}$ entonces

$$(A \cdot B)(s,t) = \sum_{u\in V} A(s,u)\cdot B(u,t)$$

Consideremos ahora matrices en $(\mathbb{R}_+^{+\infty})^{V\times V}$

$$(A \odot B)(s,t) = \min_{u\in V}(A(s,u) + B(u,t))$$

¡El producto recién definido es asociativo!
(No es anillo pues la primera operación (mín) no tiene inversa).

¡Calcular distancia es multiplicación de matrices!

Calcular todas las distancias

Entrada: $G = (V, E)$ dirigido, $\ell \colon E \to \mathbb{R}$ conservativo

$$D^{(1)}(a,b) = \begin{cases} 0 & \text{si } a = b \\ \ell(ab) & \text{si } ab \in E \\ +\infty & \text{en otro caso} \end{cases}$$

para $k = 2, ..., n-1$ **hacer**

$\quad D^{(k)} = D^{(k-1)} \odot D^{(1)}$

fin

devolver $D^{(n-1)}$

Complejidad:

Calcular todas las distancias

Entrada: $G = (V, E)$ dirigido, $\ell \colon E \to \mathbb{R}$ conservativo
Sea M el menor entero tal que $2^M \geq n - 1$.

$$D^{(1)}(a,b) = \begin{cases} 0 & \text{si } a = b \\ \ell(ab) & \text{si } ab \in E \\ +\infty & \text{en otro caso} \end{cases}$$

para $k = 1, ..., M$ **hacer**
$\quad D^{(2^k)} = D^{(2^{k-1})} \odot D^{(2^{k-1})}$
fin
devolver $D^{(2^M)}$

Propuesto: Modificar el algoritmo anterior para que guarde matrices de (argmins) $R^{(2^k)}_{k=1...M}$. Suponga acceso directo a todas las matrices calculadas y muestre como calcular un s-t camino mínimo en tiempo $O(n)$

Mejoras usando vértices internos

Roy (1959), Floyd(1952), Warshall(1962) se dieron cuenta que la idea de iterativamente aumentar el conjunto de arcos usables es más eficiente que la multiplicación de matrices.

Para $G = (V, E)$ dirigido, ℓ conservativo. Enumeremos los vértices como $v_1, \ldots, v_n$.

$$D(u, v, i) = \min\{\ell(P) \colon P \text{ es } u\text{-}v \text{ camino con vértices internos en } V_i\}$$

Con esto:

$$D(u, v, 1) = D^{(1)}(u, v)$$
$$D(u, v, i+1) = \min\Big($$

Algoritmo de Floyd-Warshall

Algoritmo de (Roy)-Floyd-Warshall

Entrada: $G = (V, E)$ dirigido, $\ell \colon E \to \mathbb{R}$ conservativo

$$D(a, b, 1) = \begin{cases} 0 & \text{si } a = b \\ \ell(ab) & \text{si } ab \in E \\ +\infty & \text{en otro caso} \end{cases}$$

para $k = 2, ..., n, a \in V, b \in V$ **hacer**

$$D(a, b, k) = \min(D(a, b, k-1), D(a, v_k, k-1) + D(v_k, b, k-1))$$

fin

devolver $D(a, b, n)$

Propuesto: Suponga acceso directo al arreglo D y muestre como calcular un s-t camino mínimo en tiempo $O(n)$. Más aún, muestre como calcular un árbol de distancias desde s en tiempo $O(n + m)$.

¿Cómo determinar si un largo es conservativo?

1. Bellman-Ford: Existe un ciclo negativo alcanzable desde s si y solo si no hay desigualdad triangular en $d_{\leq n-1}$. Es decir, existen $vw \in E$ tal que $d_{\leq n-1}(s,v) > d_{\leq n-1}(s,w) + \ell(w,v)$
2. Floyd-Warshall: Existe ciclo negativo si y solo si $D(a,a,n) < 0$ para algún $a \in V$.

Propuestos:

1. Modificar BF para que devuelva el ciclo negativo encontrado.
2. Modificar FW para que devuelva el ciclo negativo encontrado.