RESUMEN ALGORITMOS

BFS. (SEARCH)

DFS. (SEARCH)

TOPOLOGICAL SORT

KRUSKAL (AGM)

PRIM (AGM).

DIJKSTRA (SSSP).

BELLMAN-FORD (SSSP)

DAC (SSSP)

FLOYD-WARSHALL. (APSP)

DANTZIG (APSP).

JOHNSON (APSP).

DISCLAIMER: Estas son notas generadas como ayuda memoria para dar la clase, deberían funcionar como índice de estudio pero de ninguna manera como única fuente. No contiene material que se desarrollo en el pizarrón, conversaciones en clase, y puede tener algún error que se corrigió "en vivo".

Ch. 25

PROBLEMA TÍPICO: TENGO UN MAPA (COMO GOOGLE MAPS) Y QUIERO PRE-COMPUTAR TODAS LAS DISTANCIAS → LA CONSULTA ES $\Theta(1)$.

PAGANDO CON ESPACIO ⟶ $d = \begin{pmatrix} \ddots & & \\ & \delta_{ij} & \\ & & \ddots \end{pmatrix}$ $n \times n$

⟹ V × DIJKSTRA → $\Theta(V \cdot (V \log V + E)) = \Theta(V^2 \log V + VE)$

(si es sparse ⟹ $E \sim V$ ⟹ $\Theta(V^2 \log V)$.

si es densa ⟹ $E \sim V^2$ ⟹ $\Theta(VE) \sim \Theta(V^3)$.

V × BF ⟶ $\Theta(V \cdot VE)$ ⟶ sparse $\Theta(V^3)$

⟶ densa $\Theta(V^4)$

⟹ ∃ algo en tiempo polinomial pero queremos hacer algo mejor!

---

input

$G = (V, E)$, $W = \begin{cases} 0 & \text{si } i = j \\ w_{ij} & \text{si } i \neq j \ (i,j) \in E \\ \infty & \text{si } i \neq j \ (i,j) \notin E \end{cases}$

por ahora $w_{ij} < 0$ pero no contiene ciclos neg. (ya le sacamos el "alcanzable")

output $D = (d_{ij}) \longrightarrow$ al final deberíamos tener $d_{ij} = \delta(i,j)$.

P tq $P_{ij} \begin{cases} \text{NONE}, \text{ si } i = j \text{ ó } i \text{ no es alcanzable desde } j \\ \text{el pred en algún c.m. desde } i, \text{ en otro caso} \end{cases}$

↳ $G_{p,i}: \begin{cases} V_{p,i} = \{ j \in V \text{ tq } P_{ij} \neq \text{NONE} \} \cup \{ i \} \\ E_{p,i} = \{ (P_{ij}, j) \in E \text{ tq } j \in V_{p,i} - i \} \end{cases}$

G_{pi} ⇝ subgrafo generado a partir de i

⇝ árbol de C.M. c/raíz en i

ALGO. 1 APSP.

el algo va a hacer algo similar a una multiplicación de matrices en cada it. $\longrightarrow O(V^4)$ y la idea es bajarlo a $O(V^3 \log V)$.

RECAP D.P. $\rightsquigarrow$ ver MIT. Desmine.

1. Caracterizar la estructura de la sc. opt.
2. Definir el valor de la sc. opt. de forma recursiva.
3. Computar el valor de la sc opt. bottom-up
4. Construir la sc. opt. a partir del output.

---

1. (sabemos que C.M. tiene subestruct. opt.).

$W = (w_{ij})$.

considero $p: i \rightsquigarrow j$ C.M.

$|p| \leq m$

si $\nexists$ -ve. $\longrightarrow m < \infty$

~~si~~ si $i = j \longrightarrow w(p) = 0$

si $i \neq j \longrightarrow p: i \overset{p'}{\rightsquigarrow} k \longrightarrow j$

$|p'| \leq m$, p' es C.M.

$\delta(i,j) = \delta(i,k) + w_{kj}$

2. $l_{ij}^{(m)}$ es el peso mínimo de cualquier camino de i a j con como máx. m aristas.

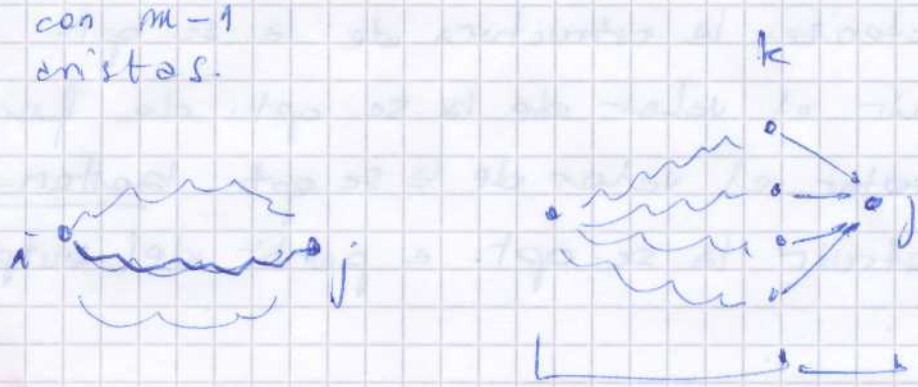
* si $m = 0 \longrightarrow \exists$ C.M. sii $i = j$

$$l_{ij}^{(0)} = \begin{cases} 0 & i = j \\ \infty & i \neq j \end{cases}$$

* $m \geq 1$

$$l_{ij}^{(m)} = \min\left(l_{ij}^{(m-1)} , \min_{1 \leq k \leq n} \left\{ l_{ik}^{(m-1)} + w_{kj} \right\} \right)$$

los caminos con $m-1$ aristas.



uso $m-1$ para ir hasta un vecino (k).

w_{kj} con k tq $(k,j) \in E$ (todos los vecinos).

$$\longrightarrow \boxed{l_{ij}^{(m)} = \min_{1 \leq k \leq n} \left\{ l_{ik}^{(m-1)} + w_{kj} \right\}}$$

w_{kj} puede ser $w_{jj} = 0$.

Si $|V| = n \longrightarrow \delta(i,j) = l_{ij}^{(n-1)} = l_{ij}^{(n)} = l_{ij}^{(n+1)}$

puede pasar por todos además de i

redundantes.

el valor opt. en m dep de $m-1$

(cual es la medida de opt no es trivial).

recursion!

③ $W = (w_{ij})$.

se calcula $L^{(1)}, L^{(2)}, \dots, L^{(n-1)}$ con $L^{(m)} = (l_{ij}^{(m)})$.

$l_{ij}^{(1)} = w_{ij}$

$l_{ij}^{(m-1)} = \delta(i,j)$; c.m!

E-SP (L, W)

$n = |L|$

$(L' = (l'_{ij})$ $n \times n$

for $i = 1 \dots n$

  for $j = 1 \dots n$

    $l'_{ij} = \infty$

    for $k = 1 \dots n$

      $l'_{ij} = \min(l'_{ij}, l_{ik} + w_{kj})$

return L'

$\Theta(V^3)$

if $l_{ij} > l_{ik} + w_{kj}$

$l_{ij} = l_{ik} + w_{kj}$

RELAX

---

multiplic. de matrices $C = A \cdot B$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

$$l_{ij}^{(m)} = \min_k \left(l_{ij}^{(m-1)}, l_{ik}^{(m-1)} + w_{kj} \right)$$

multipl_matr. (A, B)

$n = |A|$

$(C : n \times n)$

for $i = 1 : n$

  for $j = 1 : n$

    $c_{ij} = 0$

    for $k = 1 : n$

      $c_{ij} = c_{ij} + a_{ik} b_{kj}$

return C.

```
APSP(W)
    n = |W|
    L^(1) = W
    for m = 2 à n-1
        (L^(m) . n×n.)
        L^(mm) = E-SP(L^(m-1), W)   } O(V^3)
    return L^(n-1)                                  } O(V^4).
```

PERO no nos interesan todas las $L^{(m)}$ sólo $L^{(n-1)}$) sabemos que

$L^{(n-1)} = L^{(n)} = L^{(n+1)} = \dots L^{(nn)}$ con $m \geq n-1$

→ no nos interesa pesarnos

en multiplic. de matrices esto es

$L^{(1)} = W$

$L^{(2)} = W \cdot W = W^2$

$L^{(4)} = W^2 \cdot W^2 = W^4$

$L^{(8)} = W^4 \cdot W^4 = W^8$

$\dots$

$L^{(2^{\lceil \log(n-1) \rceil})} = \dots = W^{2^{\lceil \log(n-1) \rceil}}$

$n-1 \Rightarrow \log(n-1)$ op³.

⇒

```
APSP(W)
    n = |W|
    L^(1) = W
    m = 1
    while m < n-1
        L^(2m) : n×n
        E-SP(L^(m), L^(m)) ——→ W^m · W^m
        m = 2m
    return L^(m)
```

en c/ it se calc. $L^{(2m)} = \left(L^{(m)}\right)^2$ empezando en $m=1$

al final llega a m tq $n-1 \leq 2m < 2n-2$

$\rightsquigarrow$ hace $\dfrac{\log(n-1) \cdot \Theta(n^3)}{\Theta(n^3 \log(n))}$.

Floyd-Marshall $\rightsquigarrow O(V^3)$

antes era la arista $k-$ ahora el vértice.

[redacted], $\nexists$ -wc. / dyn. progr.

① estructura $p = \{v_1, v_2 \dots, v_{l-1}, v_l\}$

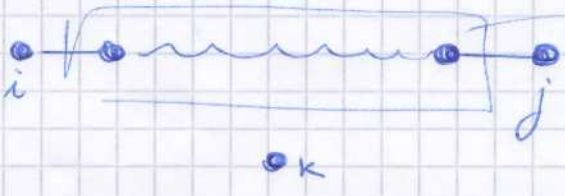
vértices internos $= \{v_2 \dots v_{l-1}\}$

dado G, $V = \{1, 2, \dots, i \dots, j \dots n\}$

dado k considero $\{1, 2, \dots, k\} = S_k$

$\Rightarrow i, j \in V$ considero todos los caminos formados por S_k tq [redacted] $\omega(p)$ es min. / p es simple.

* si $k \notin p$. $\Rightarrow$ todos los vértices de $p \in S_{k-1}$



* si $k \in p$.

$p_1, p_2 \in S_{k-1}$ → los extremos quedan afuera.

② recursión $d_{ij}^{(k)} = \omega(p \in S_k)$.

$d_{ij}^{(0)} = \omega_{ij}$ (no hay vértices intermedios $\Rightarrow$ tampoco en S_0)

siempre necesitamos un caso base

$$d_{ij}^{(k)} = \begin{cases} 0 & \text{si } k=0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{si } k \geq 1 \end{cases}$$

(3) bottom-up.

```
FLOYD-WARSHALL(W).
    n = |W|
    D^(0) = W
    for k = 1 a n
        (D^(k) . n x n)
        for i = 1 a n
            for j = 1 a n
                d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
    return D^(n)
```

$\Theta(V^3)$

RELAX.

if $d_{ij} > d_{ik} + d_{kj}$

$d_{ij} = d_{ik} + d_{kj}$

$P_{ij} = P_{kj}$

⇝ el algoritmo es simple y corto ⇒ la constante y otros terminos dentro de la Θ son peq.
⇒ es rápido inclusive para G pequeños

(4) construir G_p. (P). Se puede hacer aparte o dentro de los loops: $P^{(0)}, P^{(1)}, P^{(2)} \dots P^{(n)}$

* $P_{ij}^{(0)} = \begin{cases} \text{None} & \text{si } i = j \text{ o } w_{ij} = \infty \\ i & \text{si } i \neq j \text{ y } w_{ij} < \infty \end{cases}$

* $i \rightsquigarrow k \rightsquigarrow j$ con $k \neq j$,
⇒ pred j es igual en S_k y S_{k-1}

$$P_{ij}^{(k)} = \begin{cases} P_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ P_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

ver 25.4

TRANSITIVE CLOSURE

$G = (V, E)$, $V = \{1, 2, \ldots, n\}$ $\quad |V| = n$

queremos saber si $G \supset (i,j)$ $\forall i, j \in V$

$G^* = (V, E^*)$ tq $E^* = \{$ [redacted] tiene un 1 si $\exists$ un camino de $i \to j$. $\}$

↑ transitive closure of G.

idea 1: $w_{ij} = 1 \longrightarrow$ F-W $\longrightarrow$ si $d_{ij} < \infty \Rightarrow \exists p: i \rightsquigarrow j$

si $d_{ij} = \infty \Rightarrow \nexists p$.

$\longrightarrow \Theta(V^3)$ ✓

idea 2: $\rightsquigarrow$ menor tiempo y espacio en la práctica

sustituye $\min(\ ) \longrightarrow \vee$ (OR) en F-W

$+ \longrightarrow \wedge$ (AND)

$t_{ij}^{(k)} = 1$ si $\exists\ p \subset G$ $p: i \rightsquigarrow j$ c/vertices $\in S_k$

si no $\longrightarrow = 0$.

recursión

* $t_{ij}^{(0)} = \begin{cases} 0 & \text{si } i \neq j \wedge (ij) \in E^* \\ 1 & \text{si } i = j \vee (ij) \in E \end{cases}$

* $k \geq 1 \quad t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)} \right)$.

TRASITIVE-CLOSURE (G).

```
n = |V|
(T^(0) = (t_ij^(0))   n x n)
for i = 1:n
    for j = 1:n
        if (i == j) OR ((i,j) ∈ E)
            t_ij^(0) = 1
        else
            t_ij^(0) = 0
for k = 1:n
    (T^(k) = (t_ij^(k))  n x n))
    for i = 1:n
        for j = 1:n
            t_ij^(k) = t_ij^(k-1) v (t_ik^(k-1) ^ t_kj^(k-1))
return T^(n)
```

inic. con 1 si hay una arista. ? 0 si no.

$\Theta(n^2)$.

booleanos vs enteros / float

⟶ menos espacio!

en algunos compus ⟶ menos tiempo!

JOHNSON $\rightsquigarrow O(V^2 \log V + VE)$.

sparse V^2 denso V^3

- D.P. si no tiene $-wc$.
-  si tiene $-wc$.

1. Si no hay $w<0$ $\longrightarrow$ $V \times$ Dijkstra $O(V^2 \log V + VE)$
2. Si hay $w<0$ $\longrightarrow$ construye un nuevo set de pesos que sea >0.
   1) $p: u \overset{w}{\rightsquigarrow} r$ es C.M. sii $p: u \overset{w'}{\rightsquigarrow} r$ es C.M.
   2) $w' > 0$.

   $\rightsquigarrow V \times$ Dijkstra.

Lema (reasignación de pesos sin cambiar C.M.).

$G=(V,E)$, $w: E \longrightarrow \mathbb{R}$, $h: V \longrightarrow \mathbb{R}$

$w'(u,v) = w(u,v) + h(u) - h(v)$.

$p: v_0 \rightsquigarrow v_k$

p^* es C.M. de v_0 a v_k sii p' es C.M. de v_0 a v_k

$(w(p) = \delta(v_0, v_k)$ sii $w'(p) = \delta'(v_0, v_k))$

G tiene $-wc$ con w sii G tiene $-wc$ con w'

Demo (*) $w'(p) = w(p) + h(r_0) - h(r_k)$.

$$w'(r_0, r_k) = \sum_{i=1}^{k} w'(r_{i-1}, r_i)$$

$$= \sum_{i=1}^{k} \left(w(r_{i-1}, r_i) + h(r_{i-1}) - h(r_i) \right)$$

$$= \sum_{i=1}^{k} w(r_{i-1}, r_i) + h(r_0) + \cancel{\dots + h(r_{k-1}) - h(r_1) - \dots} \cancel{\dots - h(r_k)}$$

$$= \sum_{i=1}^{k} w(r_{i-1}, r_i) + h(r_0) - h(r_k)$$

$$\boxed{w'(p) = w(p) + h(r_0) - h(r_k)}$$

$\Big\downarrow$

como h no dep. del camino

$\Rightarrow$ si $\delta(r_0, r_k)$ $\exists$ si $\delta'(r_0, r_k)$. ✓

(*) $c: r_0 \dots r_k$ es un ciclo $\Rightarrow r_0 = r_k$

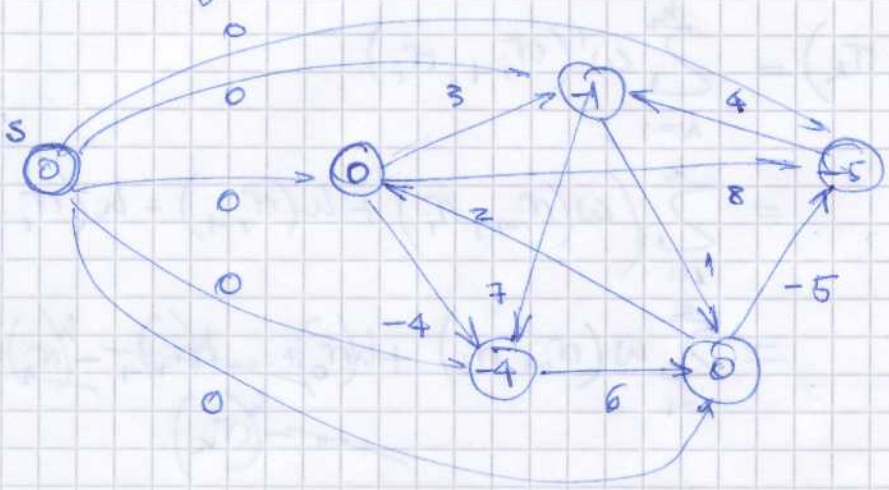
$$w'(c) = w(c) + \underbrace{h(r_0) - h(r_k)}_{r_0 = r_k \Rightarrow = 0}$$

$$w'(c) = w(c)$$

si $w(c) < 0 \longrightarrow w'(c) < 0$

✓

Reasignar tq $w \geq 0$.



Agrego s con aristas a todos los nodos con peso 0 ⟶ no modifica los C.M. porque nadie puede alcanzar s.
(similar a "diferencias")

definimos $h(v) = \delta(s, v)$ ← δ, δ' sí $-\infty$. para que δ tenga sentido

$h(x) \leq h(u) + w(u, v)$ ⟵ por desig triang

$w'(u, v) = w(u, v) + h(u) - h(v) \geq 0$

JOHNSON (G, w)

generar $G' \longrightarrow V' = V \cup \{s\}$

$E' = E \cup \{(s, v) : v \in V\}$

$w(s, v) = 0 \quad \forall v \in V$

$O(VE)$ { si B-F (G', w, s) == FALSO

| tiene — NC!

sólo se corre desde S. que alcanza a todos

si no.

$O(V)$ | for $v \in V$

| | $h(v) = \delta(s, v)$ ← calculado con B-F

$O(E)$ | for $(u, v) \in E$

| | $w'(u, v) = w(u, v) + h(u) - h(v)$.

| $D : n \times n$

| for $u \in V$

$O(V \cdot (V \log V + E))$ | | DIJKSTRA $(G, w', u) \longrightarrow \delta'(u, v)$

| | for $v \in V$

| | | $d_{uv} = \delta'(u, v) + h(v) - h(u)$

return D.

$\hookrightarrow O(VE + V + E + V^2 \log V + VE)$.

$O(V^2 \log V + VE)$

Si $E \sim V \longrightarrow O(V^2 \log V)$

$E \sim V^2 \longrightarrow O(V^3)$.