# Clase 11 MA3705.16 de octubre 2020.

**Tabla**

- Largos conservativos: Bellman Ford
- Largos positivos enteros pequeños: BFS
- Largos positivos: Dijkstra

# Largos conservativos: Bellman Ford

# Recuerdo: Largos conservativos, distancias y desigualdad triangular

$G = (V, E)$ digrafo, $\ell \colon E \to \mathbb{R}$ es conservativo si para cada ciclo $C$, $\ell(C) \geq 0$.

Si $\ell$ es conservativo, se tienen las siguientes propiedades:

$$d(s,t) = \min_{P \ s\text{-}t \text{ camino}} \ell(P) \qquad d(s,t) = \min_{P \ s\text{-}t \text{ paseo}} \ell(P).$$

$$d_{\leq k}(s,t) = \min_{P \ s\text{-}t \text{ camino con} \leq k \text{ arcos}} \ell(P) \qquad d_{\leq k}(s,t) = \min_{P \ s\text{-}t \text{ paseo con} \leq k \text{ arcos}} \ell(P).$$

$$d(s,t) \leq d(s,u) + d(u,t).$$

**Teorema: Subcaminos óptimos**

Sea $P$ $s$-$t$ camino mínimo. Todo subcamino de $P$ es de largo mínimo entre extremos.

Demostración:

# Consecuencia:

**Teorema: Si $\ell$ es conservativo**

$$d(s,t) = \min_{w \in N^-(t)} d(s,w) + \ell(wt) \tag{1}$$

$$d_{\leq k}(s,t) = \min\Big(d_{\leq k-1}(s,t), \min_{w \in N^-(t)} d_{\leq k-1}(s,w) + \ell(wt)\Big) \tag{2}$$

Demostración:

(1) del teorema anterior.

(2) vale siempre para paseos. Como $\ell$ es conservativo se traspasa a caminos.

# Bellman Ford: usa $d_{\leq k}(s,v) = \min\Big(d_{\leq k-1}(s,v), \min_{w\in N^-(v)} d_{\leq k-1}(s,w) + \ell(wv)\Big)$

## Algoritmo de Bellman-Ford (Bellman (1958), Ford (1956), Moore (1957))

**Entrada:** $G = (V, E)$ dirigido, $\ell\colon E \to \mathbb{R}$ conservativo, $s \in V$
**para** $v \in V$ **hacer** $d_{\leq 0}(s,v) = +\infty$, $\text{Padre}(v) \leftarrow \text{Null}$
$d_{\leq 0}(s,s) = 0$
**para** $k = 1, ..., n-1$ **hacer**
  **para** $v \in V$ **hacer** $d_{\leq k}(s,v) \leftarrow d_{\leq k-1}(s,v)$
  **para** $w \in N^-(v)$ **hacer**
    **si** $d_{\leq k-1}(s,w) + \ell(wv) < d_{\leq k}(s,v)$ **entonces**
      $d_{\leq k}(s,v) \leftarrow d_{\leq k-1}(s,w) + \ell(w,v)$
      $\text{Padre}(v) \leftarrow w$
    **fin**
  **fin**
**fin**
**devolver** $d_{\leq n-1}, \text{Padre}$

# Basta recordar el padre. Arborescencia (de entrada) de caminos mínimos.

Arborescencia:

- Cada nodo, excepto la raíz tiene grado de salida 1.
- La raíz es alcanzable desde cada nodo (equivalentemente: no hay ciclos)

# Complejidad de Bellman Ford

**Entrada:** $G = (V, E)$ dirigido, $\ell \colon E \to \mathbb{R}$ conservativo, $s \in V$
**para** $v \in V$ **hacer** $d_{\leq 0}(s, v) = +\infty$, $\text{PADRE}(v) \leftarrow \text{NULL}$
$d_{\leq 0}(s, s) = 0$
**para** $k = 1, ..., n-1$ **hacer**
  **para** $v \in V$ **hacer** $d_{\leq k}(s, v) \leftarrow d_{\leq k-1}(s, v)$
  **para** $w \in N^-(v)$ **hacer**
    **si** $d_{\leq k-1}(s, w) + \ell(wv) < d_{\leq k}(s, v)$ **entonces**
      $d_{\leq k}(s, v) \leftarrow d_{\leq k-1}(s, w) + \ell(w, v)$
      $\text{PADRE}(v) \leftarrow w$
    **fin**
  **fin**
**fin**
**devolver** $d_{\leq n-1}, \text{PADRE}$

# Largos positivos enteros pequeños: BFS

# Otra mirada a BFS

Si los largos son unitarios $\ell(e) = 1, \forall e \in E$, entonces la fórmula:

$$d(s,t) = \min_{w \in N^-(v)} d(s,w) + \ell(w,t)$$

se reduce a

$$d(s,t) = \min_{w \in N^-(v)} d(s,w) + 1$$

Luego las distancias posibles son $0, 1, 2, \ldots, n-1$.
Para un conjunto $I$, llamemos $L(I) = \{t \in V : d(s,t) \in I\}$. Luego:

$$t \in L(\{k\}) \iff t \in N^+(L([0, k-1]))$$

# BFS (menos eficiente - ilustrativo)

## BFS

**Entrada:** $G = (V, E)$ dirigido, $s \in V$
**para** $v \in V$ **hacer** $d(s, v) = +\infty$.
$d(s, s) \leftarrow 0$. VISITADOS $\leftarrow \{s\}$, $F \leftarrow \emptyset$.
RELOJ $\leftarrow 0$
**mientras** $\delta^+(\text{VISITADOS}) \neq \emptyset$ **hacer**
  RELOJ $\leftarrow$ RELOJ $+ 1$.
  ARCOS ACTIVOS $\leftarrow \delta^+(\text{VISITADOS})$
  **para** $e = uv \in$ ARCOS ACTIVOS **hacer**
    **si** $v \notin$ VISITADOS **entonces**
      $d(s, v) =$ RELOJ, $F \leftarrow F + e$
      VISITADOS $\leftarrow$ VISITADOS $+ v$.
    **fin**
  **fin**
**fin**
**devolver** $F, d$

# BFS para enteros pequeños

Si $\ell(e) = 1, \forall e$. BFS calcula distancias desde raíz en tiempo $O(n + m)$.

Supongamos ahora $\ell(e) \in \{1, \ldots, M\}$.
¿Cómo calcular distancias en tiempo $O(M(m + n))$.

# Mejor: Simular lo que haría BFS

## BFS-simulado para largos enteros (intuición)

**Entrada:** $G = (V, E)$ dirigido, $s \in V$
**para** $v \in V$ **hacer** $d(s, v) = +\infty$.

- Si un nodo $u$ es visitado por primera vez, activar todos los arcos de $\delta^+(v)$.
- Si un arco $e = uv$ es activado, mandar un mensaje a $v$ que llegará en el tiempo $\text{RELOJ} + \ell(uv)$.
- Si un arco termina de mandar su mensaje, desactivarlo.
- Si un nodo $v$ **no visitado** recibe un mensaje, fijar $d(s, v) = \text{RELOJ}$ y visitar $v$.
- Si no hay más acciones en tiempo RELOJ y quedan arcos activos, avanzar el reloj en 1 unidad.

ARCOS ACTIVOS $\leftarrow \emptyset$
VISITADOS $\leftarrow \emptyset$
RELOJ $\leftarrow 0$.
Visitar $s$.

# Ejemplo

# Largos positivos: Dijkstra

# No necesitamos avanzar el reloj en tiempos discretos

En la simulación anterior *basta* que cada vértice $v$ no visitado guarde en su memoria el momento $T(v)$ en el que recibirá el siguiente mensaje.

# Así:

**Entrada:** $G = (V, E)$ dirigido, $s \in V$
**para** $v \in V$ **hacer** $d(s, v) = +\infty$, $T(v) \leftarrow \infty$.

- Si un nodo $u$ es visitado por primera vez, activar todos los arcos de $\delta^+(v)$.
- Si un arco $e = uv$ es activado, ~~mandar un mensaje a $v$ que llegará en el tiempo $\textsc{reloj} + \ell(uv)$.~~ $T(v) \leftarrow \min\{T(v), \textsc{reloj} + \ell(uv)\}$
- Si un arco termina de mandar su mensaje, desactivarlo.
- ~~Si un nodo $v$ **no visitado** recibe un mensaje, fijar $d(s, v) = \textsc{reloj}$ y visitar $v$.~~ Si un nodo $v$ no visitado tiene $T(v) = \textsc{reloj}$, fijar $d(s, v)$ y visitar $v$.
- Si no hay más acciones en tiempo $\textsc{reloj}$ y quedan arcos activos, avanzar el reloj ~~en 1 unidad.~~ hasta $\min_{v \in V \setminus \textsc{Visitados}} T(v)$.

$\textsc{Arcos activos} \leftarrow \emptyset$. $\textsc{Visitados} \leftarrow \emptyset$. $\textsc{reloj} \leftarrow 0$.
Visitar $s$.

# Simplificado queda:

**Entrada:** $G = (V, E)$ dirigido, $s \in V$
**para** $v \in V$ **hacer** $d(s, v) = +\infty$, $T(v) \leftarrow \infty$.

- Si un nodo $u$ es visitado por primera vez, activar todos los arcos de $\delta^+(v)$.
- Si un arco $e = uv$ es activado, $T(v) \leftarrow \min\{T(v), \text{RELOJ} + \ell(uv)\}$
- Si un arco termina de mandar su mensaje, desactivarlo.
- Si un nodo $v$ no visitado, tiene $T(v) = \text{RELOJ}$, fijar $d(s, v)$ y visitar $v$.
- Si no hay más acciones en tiempo RELOJ y quedan arcos activos, avanzar el reloj hasta $\min_{v \in V \setminus \text{VISITADOS}} T(v)$.

ARCOS ACTIVOS $\leftarrow \emptyset$. VISITADOS $\leftarrow \emptyset$. RELOJ $\leftarrow 0$.
Visitar $s$.

# No necesitamos arcos activos:

**Entrada:** $G = (V, E)$ dirigido, $s \in V$
**para** $v \in V$ **hacer** $d(s, v) = +\infty$. $T(v) \leftarrow \infty$

- Cada vez que un nodo $u$ es visitado por primera vez.
  Fijar para cada $v \in N^+(v)$ no visitado, $T(v) \leftarrow \min\{T(v), \text{RELOJ} + \ell(uv)\}$
- Si un nodo $v$ no visitado tiene $T(v) = \text{RELOJ}$, fijar $d(s, v)$ y visitar $v$.
- Si no hay más acciones en tiempo $\text{RELOJ} < +\infty$ avanzar hasta $\min_{v \in V \setminus \text{VISITADOS}} T(v)$.

$\text{VISITADOS} \leftarrow \emptyset$. $\text{RELOJ} \leftarrow 0$.
Visitar $s$.

# Dijkstra: no necesitamos reloj

## Algoritmo de Dijkstra (Dijkstra 1959):

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

# Dijkstra: Arborescencia óptima y complejidad.

## Algoritmo de Dijkstra (Dijkstra 1959):

**Entrada:** $G = (V, E)$ dirigido, $s \in V$, $\ell \colon E \to \mathbb{R}^+$
**para** $v \in V$ **hacer** $d(s, v) = +\infty$. $T(v) \leftarrow \infty$. $\text{Padre}(v) \leftarrow \text{Null}$
$\text{No-Visitados} \leftarrow V$, $F \leftarrow \emptyset$.
$T(s) \leftarrow 0$.
**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** $d, \text{Padre}$

# Mejores implementaciones

Al igual que Prim, Dijkstra se puede implementar mejor con mejores estructuras de datos

En tiempo $O(n^2)$ usando arreglos y listas enlazadas.
En tiempo $O((n+m)\log n)$ usando Heaps Binarios.
En tiempo $O(m + n\log n)$ usando Heaps de Fibonacci.

# 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)!