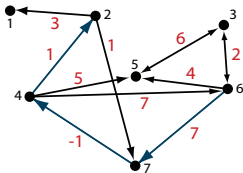


## Problemas de camino mínimo

- Dado un grafo orientado  $G = (V, E)$  con longitudes asociadas a sus aristas ( $\ell : E \rightarrow \mathbb{R}$ ), la **longitud** (o peso o costo) de un camino es la suma de las longitudes de sus aristas.



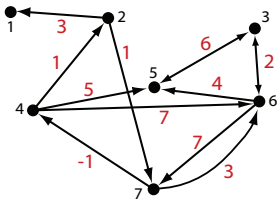
En el digrafo de la figura, la longitud del camino es 7.

- El problema del **camino mínimo** consiste en encontrar el camino de menor longitud...
  - de un vértice a otro
  - de un vértice a todos los demás
  - entre todos los pares de vértices.

## Problemas de camino mínimo

- La **distancia** de  $v$  a  $w$ ,  $d(v, w)$ , es la longitud de un camino mínimo entre  $v$  y  $w$ ,  $+\infty$  si no existe ningún camino de  $v$  a  $w$ , y  $-\infty$  si existen caminos de  $v$  a  $w$  pero no uno mínimo.
- Observemos que en un grafo orientado **no** siempre vale  $d(v, w) = d(w, v)$ .

Ej:



En el digrafo de la figura,  $d(7, 4) = -1$  y  $d(4, 7) = 2$ .

## Problemas de camino mínimo

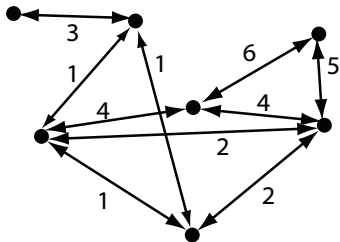
- Si  $G$  tiene un ciclo orientado de longitud negativa, no va a existir camino mínimo entre algunos pares de vértices, es decir, va a haber vértices a distancia  $-\infty$ .



- Si  $G$  no tiene ciclos orientados de longitud negativa (aunque pueda tener aristas de longitud negativa), para todo camino  $P$  existe  $P'$  simple con  $\ell(P') \leq \ell(P)$ . Como la cantidad de caminos simples en un grafo finito es finita, si existe un camino entre  $v$  y  $w$ , existe uno mínimo.

## Problemas de camino mínimo

- Si  $G$  es no orientado, el problema de camino mínimo en  $G$  va a ser el mismo que en el digrafo que se obtiene reemplazando cada arista por dos orientadas una para cada lado y con la misma longitud que la original. En ese caso, aristas de longitud negativa implicarían ciclos negativos.



# Problemas de camino mínimo

## Principio de optimalidad

Todo subcamino de un camino mínimo es un camino mínimo.

**Demo:** Sea  $P$  un camino mínimo entre  $s$  y  $t$  y sea  $P_1$  un subcamino de  $P$ , con extremos  $v$  y  $w$ . Si existe un camino  $P'_1$  en  $G$  entre  $v$  y  $w$  tal que  $\ell(P'_1) < \ell(P)$  entonces reemplazando  $P_1$  por  $P'_1$  en  $P$ , obtendríamos un camino entre  $s$  y  $t$  de longitud menor que la de  $P$ , absurdo. □

# Arbol de caminos mínimos

## Propiedad

Si  $G$  no tiene aristas de costo negativo y  $s$  es un vértice de  $G$  tal que para todo  $v \in V(G)$  existe un camino de  $s$  a  $v$ , entonces existe un árbol orientado  $T$  con raíz  $s$  tal que para todo  $v \in V(G)$ , el camino de  $s$  a  $v$  en  $T$  es un camino mínimo de  $s$  a  $v$  en  $G$ .

**Demo:** Por inducción. Si  $V(G) = \{s\}$ , vale. Si  $|V(G)| \geq 2$ , para cada  $x$ , sea  $P_x$  un camino mínimo de  $s$  a  $x$  que con cantidad de aristas mínima. Sea  $v$  tal que para todo  $v' \in V(G)$ ,  $d(s, v') \leq d(s, v)$ . Ante empates en la distancia,  $P_v$  es el que tiene más aristas. Como no hay aristas de costo negativo y por la forma de elegir  $v$  y  $P_x$ , para todo  $v'$ ,  $v$  no pertenece a  $P_{v'}$ . Por lo tanto, en  $G - v$  existen caminos de  $s$  a todos los vértices. Por hipótesis inductiva, existe  $T'$  árbol de caminos mínimos de  $G - v$ . Sea  $w$  el anteúltimo vértice en  $P_v$ . Entonces  $d(s, v) = d(s, w) + \ell(wv)$ . Luego, agregando a  $T'$  el vértice  $v$  y la arista  $wv$  obtenemos un árbol de caminos mínimos de  $G$ . □

## Arbol de caminos mínimos

La propiedad vale igual para grafos con aristas negativas pero sin ciclos negativos, aunque la demostración es distinta.

## Algoritmo de Dijkstra & Moore

- Sirve para calcular un árbol de caminos mínimos desde un vértice  $s$  a todos los vértices alcanzables desde  $s$ .
- Funciona cuando el grafo **no tiene aristas negativas**.
- **Input del algoritmo:** grafo  $G = (V, E)$ ,  $V = \{1, \dots, n\}$ , longitudes  $\ell : E \rightarrow \mathbb{R}^+$ .
- **Output del algoritmo:** árbol de caminos mínimos desde el vértice 1.



# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

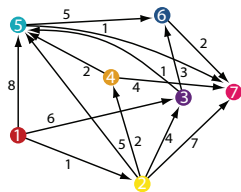
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \emptyset$

$\pi = [0, \infty, \infty, \infty, \infty, \infty, \infty]$

$P = [1, 0, 0, 0, 0, 0, 0]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

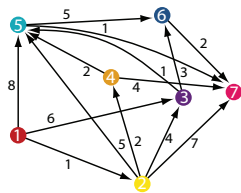
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1\}$

$\pi = [0, \infty, \infty, \infty, \infty, \infty, \infty]$

$P = [1, 0, 0, 0, 0, 0, 0]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

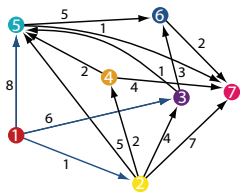
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1\}$

$\pi = [0, 1, 6, \infty, 8, \infty, \infty]$

$P = [1, 1, 1, 0, 1, 0, 0]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

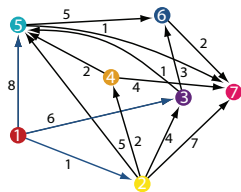
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2\}$

$\pi = [0, 1, 6, \infty, 8, \infty, \infty]$

$P = [1, 1, 1, 0, 1, 0, 0]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

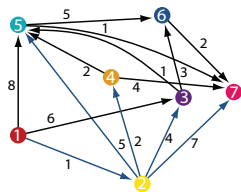
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2\}$

$\pi = [0, 1, 5, 3, 6, \infty, 8]$

$P = [1, 1, 2, 2, 2, 0, 2]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

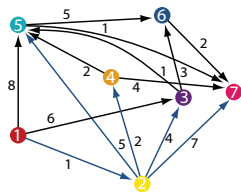
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4\}$

$\pi = [0, 1, 5, 3, 6, \infty, 8]$

$P = [1, 1, 2, 2, 2, 0, 2]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

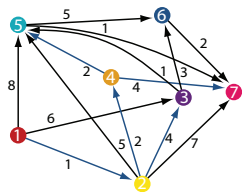
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4\}$

$\pi = [0, 1, 5, 3, 5, \infty, 7]$

$P = [1, 1, 2, 2, 4, 0, 4]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

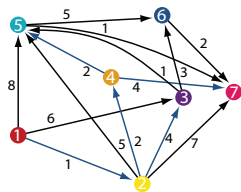
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4, 3\}$

$\pi = [0, 1, 5, 3, 5, \infty, 7]$

$P = [1, 1, 2, 2, 4, 0, 4]$



# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

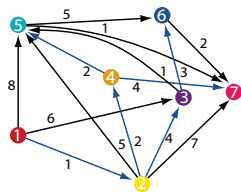
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4, 3\}$

$\pi = [0, 1, 5, 3, 5, 8, 7]$

$P = [1, 1, 2, 2, 4, 3, 4]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

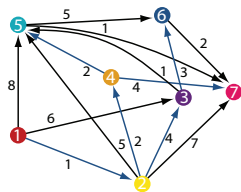
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4, 3, 5\}$

$\pi = [0, 1, 5, 3, 5, 8, 7]$

$P = [1, 1, 2, 2, 4, 3, 4]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

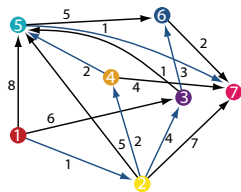
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4, 3, 5\}$

$\pi = [0, 1, 5, 3, 5, 8, 6]$

$P = [1, 1, 2, 2, 4, 3, 5]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

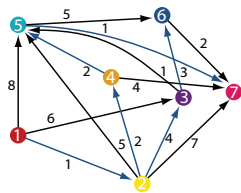
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4, 3, 5, 7\}$

$\pi = [0, 1, 5, 3, 5, 8, 6]$

$P = [1, 1, 2, 2, 4, 3, 5]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

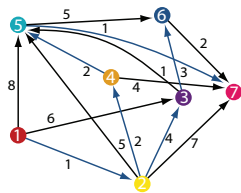
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = \{1, 2, 4, 3, 5, 7\}$

$\pi = [0, 1, 5, 3, 5, 8, 6]$

$P = [1, 1, 2, 2, 4, 3, 5]$

# Algoritmo de Dijkstra & Moore

conjunto  $S$ ; vectores  $\pi$  y  $P$  de longitud  $n$ ;

$S = \emptyset$ ;

$\pi(1) = 0$ ;  $\pi(i) = \infty$  para  $2 \leq i \leq n$ ;

$P(1) = 1$ ;  $P(i) = 0$  para  $2 \leq i \leq n$ ;

Mientras  $S \neq V$

Elegir  $j \notin S$  tq  $\pi(j) = \min_{i \notin S} \pi(i)$

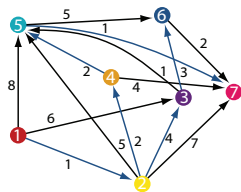
$S = S \cup \{j\}$ ;

Para cada  $i \in \text{suc}(j)$  tq  $i \notin S$

Si  $\pi(i) > \pi(j) + \ell(ji)$

$\pi(i) = \pi(j) + \ell(ji)$

$P(i) = j$



$S = V$

$\pi = [0, 1, 5, 3, 5, 8, 6]$

$P = [1, 1, 2, 2, 4, 3, 5]$

# Algoritmo de Dijkstra & Moore

- En cada iteración agrega un nuevo vértice a  $S$ , por lo tanto termina en  $n$  iteraciones.
- El costo de cada iteración es  $O(n + d(j))$  u  $O(\log n + d(j))$  según como se implemente  $S$ . Como  $j$  recorre  $V$ , el costo total del algoritmo es  $O(n^2)$  u  $O(m + n \log n)$ , respectivamente.
- **Notación:**  $\pi^*(i) = d(1, i)$ ;  $\pi_S(i) =$  longitud del camino mínimo de 1 a  $i$  que sólo pasa por vértices de  $S$ .
- **Lema:** Al terminar cada iteración, para todo  $i \in V$  vale:
  1.  $i \in S \Rightarrow \pi(i) = \pi^*(i)$ .
  2.  $i \notin S \Rightarrow \pi(i) = \pi_S(i)$ .
- **Teorema:** El algoritmo funciona.
  1. Termina.
  2. Cuando termina  $S = V$ , y por el lema,  $\forall i \in V \ \pi(i) = \pi^*(i)$ .

## Demostración del lema

Por inducción en la cantidad de iteraciones. Como inicialmente  $\pi(1) = 0$  y  $\pi(i) = \infty$  para  $i \neq 1$ , luego de la primera iteración,  $S = \{1\}$ . Además, por como se actualizó  $\pi$ , vale  $\pi(1) = 0 = \pi^*(1)$ ,  $\pi(i) = \ell(1i) = \pi_S(i)$  para los sucesores de 1 y  $\pi(i) = \infty = \pi_S(i)$  para el resto de los vértices.

Supongamos por hipótesis inductiva que el lema vale para  $S$  y veamos que vale para  $S \cup \{j\}$  luego de completar la siguiente iteración. Como  $\pi$  no se modifica para los elementos de  $S$ , vale  $\pi(i) = \pi^*(i)$  para  $i \in S$ .

Falta probar entonces

1.  $\pi(j) = \pi^*(j)$  y
2.  $\pi(i) = \pi_{S \cup \{j\}}(i)$  para  $i \notin S \cup \{j\}$ .



## Demostración del lema

1.  $\pi(j) = \pi^*(j)$

Por HI sabíamos que  $\pi(j) = \pi_S(j)$ . Tomemos un camino  $P$  de 1 a  $j$ .<sup>1</sup> Sea  $i$  el primer vértice fuera de  $S$  en  $P$  (podría ser  $j$ ), y sea  $P_i$  el subcamino de  $P$  que va de 1 a  $i$ . Como no hay aristas negativas,  $\ell(P) \geq \ell(P_i) \geq \pi_S(i) \geq \pi_S(j)$  por HI y la forma de elegir  $j$ . Luego  $\pi^*(j) = \pi_S(j) = \pi(j)$ .

2.  $\pi(i) = \pi_{S \cup \{j\}}(i)$  para  $i \notin S \cup \{j\}$

Sea  $i \notin S$ ,  $i \neq j$ . Claramente,  $\pi_{S \cup \{j\}}(i) \leq \min\{\pi_S(i), \pi^*(j) + \ell(ji)\}$ . Veamos la desigualdad inversa. Sea  $P$  un camino de 1 a  $i$  que sólo pasa por  $S \cup \{j\}$ . Sea  $v$  el último vértice antes de  $i$  en  $P$ . Si  $v = j$ , entonces  $\ell(P) \geq \pi^*(j) + \ell(ji)$  (\*). Si  $v \in S$ , sea  $P'$  el camino que se obtiene de reemplazar en  $P$  el subcamino de 1 a  $v$  por un camino de 1 a  $v$  de longitud  $\pi^*(v)$  que sólo pase por  $S$ , que por HI sabemos que existe. Entonces  $\ell(P) \geq \ell(P') \geq \pi_S(i)$  (\*\*).

Por (\*) y (\*\*), vale  $\pi_{S \cup \{j\}}(i) \geq \min\{\pi_S(i), \pi^*(j) + \ell(ji)\}$ . □

---

<sup>1</sup>Podemos considerar que existen todas las aristas pero algunas pesan  $\infty$ .

# Algoritmo de Bellman & Ford

# Algoritmo de Dantzig

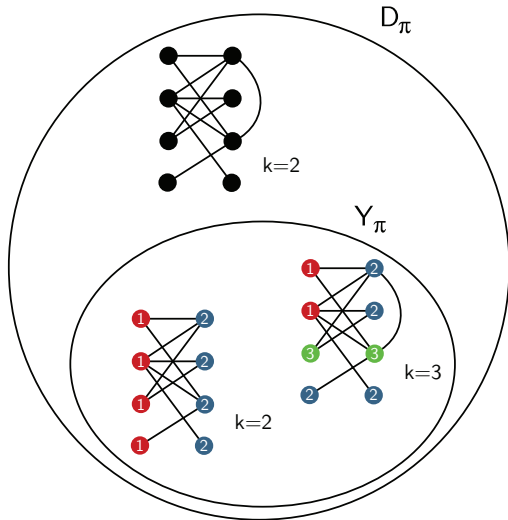
# Algoritmo de Floyd

## La teoría de NP-completitud

- Se aplica a **problemas de decisión**, o sea problemas que toman ciertos parámetros y tienen como respuesta SI o NO (aunque es sencillo ver que sus implicancias pueden extenderse a problemas de optimización).
- En el caso del problema de coloreo óptimo de un grafo, la variante de decisión se podría formular como: “Dado un grafo  $G$  y un número  $k$ , ¿existe un coloreo de  $G$  que utilice a lo sumo  $k$  colores?”
- Una **instancia** de un problema es una especificación de sus parámetros. Un problema de decisión  $\pi$  tiene asociado un conjunto  $D_\pi$  de instancias y un subconjunto  $Y_\pi \subseteq D_\pi$  de instancias cuya respuesta es SI.

## Ejemplo: coloreo

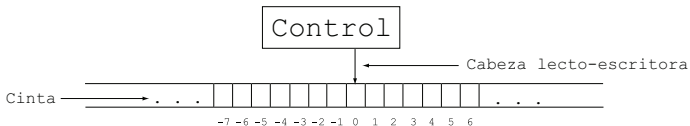
Dado un grafo  $G$  y un número  $k$ , ¿existe un coloreo de  $G$  que utilice a lo sumo  $k$  colores?



# Modelos de Computadoras: DTM

Recordemos la noción de Máquina de Turing Determinística (DTM)

- Consiste de un control finito, una cabeza lecto-escritora y una cinta con el siguiente esquema.



- $\Sigma$  finito, el alfabeto;  $\Gamma = \Sigma \cup \{*\}$ ;
- $Q$  finito, el conjunto de estados;
- $q_0 \in Q$ , estado inicial;  $Q_f \subseteq Q$ , estados finales ( $q_{si}$  y  $q_{no}$  para problemas de decisión)

## Modelos de Computadoras: DTM

- Sobre la cinta tengo escrito el input que es un string de símbolos de  $\Sigma$  a partir de la celda 1, y el resto de las celdas tiene \* (blancos).
- Definimos un programa  $S$  como un conjunto de quintuplas  $S \subseteq Q \times \Gamma \times Q \times \Gamma \times M$ , donde  $M = \{+1, -1\}$  son los movimientos de la cabeza a derecha o izquierda.
- **Para todo par  $(q_i, s_j)$ , existe exactamente una quintupla que comienza con ese par (máquina determinística).**

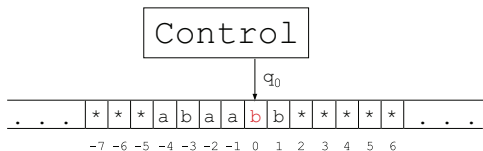


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

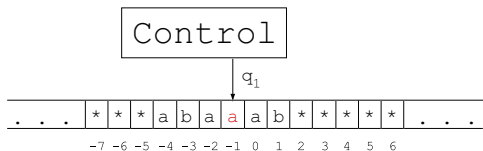


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

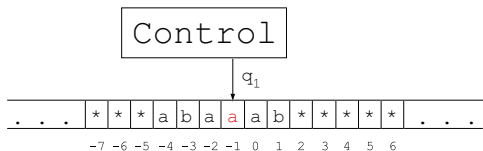


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

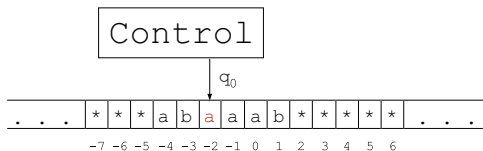


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

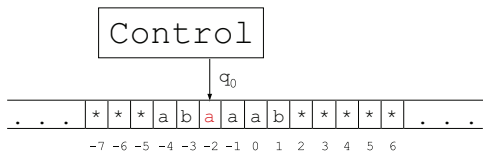


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

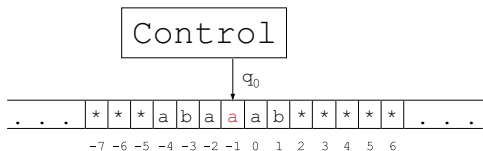


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

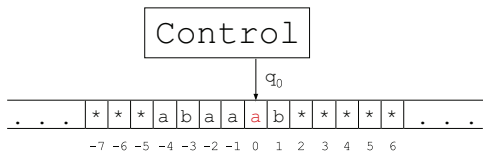


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

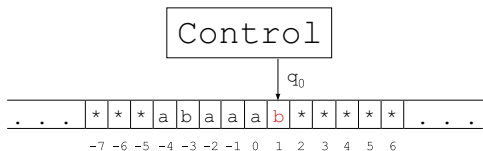


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$



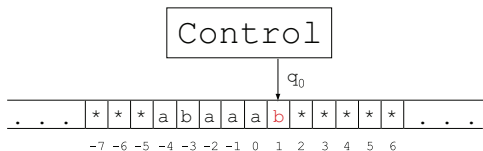


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

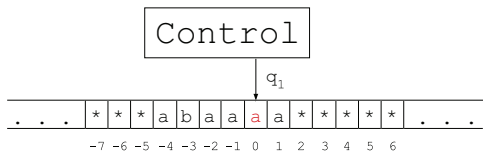


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

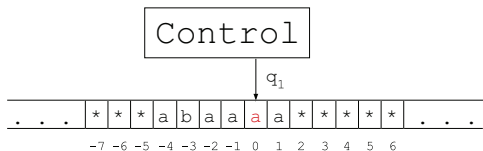


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

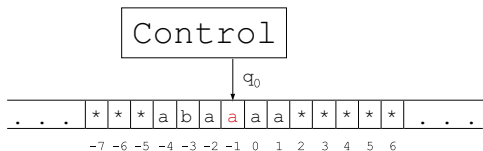


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\}$ ;  
 $\Gamma = \Sigma \cup \{*\}$ ;
- $Q = \{q_0, q_1, q_{si}, q_{no}\}$ ;  
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

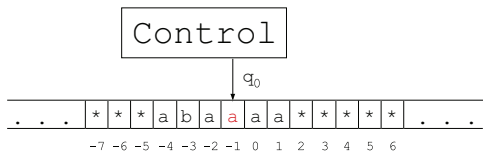


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

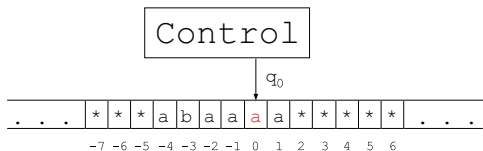


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

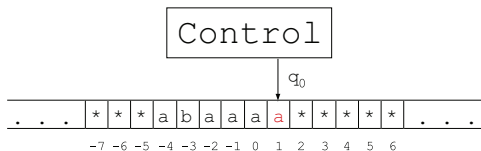


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

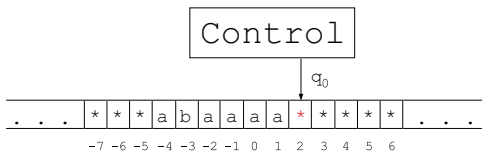


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$



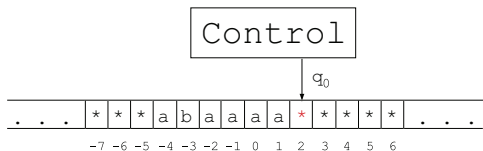


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$

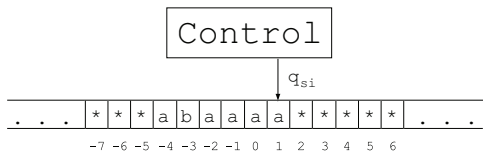


# Modelos de Computadoras: DTM

¿Qué significa la quintupla  $(q_i, s_h, q_j, s_k, +1)$ ? Significa que si estando en el estado  $q_i$  la cabeza lee  $s_h$ , entonces escribe  $s_k$ , se mueve a la derecha y pasa al estado  $q_j$ .

Ej:

- $\Sigma = \{a, b\};$   
 $\Gamma = \Sigma \cup \{*\};$
- $Q = \{q_0, q_1, q_{si}, q_{no}\};$   
 $Q_f = \{q_{si}, q_{no}\}$
- $S = (q_0, a, q_0, a, +1),$   
 $(q_0, b, q_1, a, -1),$   
 $(q_0, *, q_{si}, *, -1),$   
 $(q_1, a, q_0, a, -1),$   
 $(q_1, b, q_{no}, a, -1),$   
 $(q_1, *, q_0, b, +1)$



# Modelos de Computadoras: DTM

- Una máquina  $M$  resuelve el problema  $\pi$  si para toda instancia empieza, termina y contesta bien (o sea, termina en el estado final correcto).
- La complejidad de una DTM está dada por la cantidad de movimientos de la cabeza, desde el estado inicial hasta alcanzar un estado final, en función del tamaño de la entrada.  
$$T_M(n) = \text{máx}\{m \text{ tq } \exists x \in D_\pi, |x| = n \text{ y } M \text{ con input } x \text{ tarda } m\}$$

## La clase P

- **Un problema está en P si existe una DTM de complejidad polinomial que lo resuelve.**

$P = \{\pi \text{ tq } \exists M \text{ DTM tq } M \text{ resuelve } \pi \text{ y } T_M(n) \in O(p(n)) \text{ para algún polinomio } p\}$

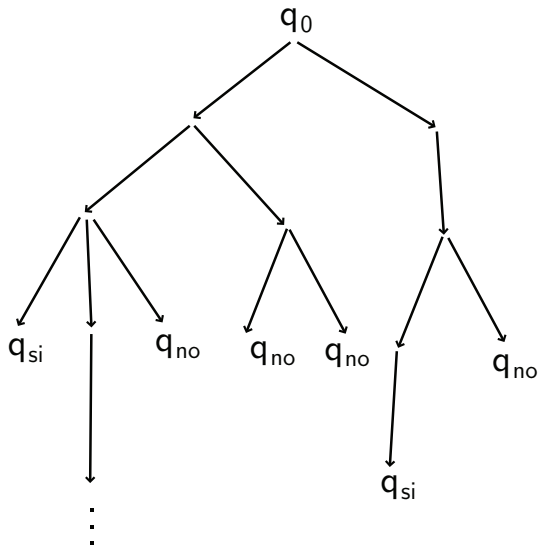
- Existen otros modelos de computadoras determinísticas (máquina de Turing con varias cintas, Random Access Machines, etc.) pero puede probarse que son equivalentes en términos de la polinomialidad de los problemas a la DTM.

# Modelos de Computadoras: NDTM

## Máquinas de Turing No Determinísticas (NDTM)

- **No se pide unicidad de la quintupla que comienza con cualquier par  $(q_i, s_j)$ .**
- En caso de que hubiera más de una quintupla, la máquina se replica continuando cada una por una rama distinta.
- Decimos que una NDTM resuelve el problema  $\pi$  si para toda instancia de  $Y_\pi$  existe una rama que llega a un estado final  $q_{si}$  y para toda instancia en  $D_\pi \setminus Y_\pi$  ninguna rama llega a un estado final  $q_{si}$ .

## Modelos de Computadoras: NDTM



# Modelos de Computadoras: NDTM

- Una NDTM es **polinomial** para  $\pi$  cuando existe una función polinomial  $T(n)$  de manera que para toda instancia de  $Y_\pi$  de tamaño  $n$ , alguna de las ramas termina en estado  $q_{si}$  en a lo sumo  $T(n)$  pasos.
- **Un problema  $\pi \in \text{NP}$  si existe una NDTM polinomial que resuelve  $\pi$ .**
- Claramente,  $P \subseteq \text{NP}$ .
- **Conjetura:  $P \neq \text{NP}$ .**  
Todavía no se demostró que exista un problema en  $\text{NP} \setminus P$ . Mientras tanto, se estudian clases de complejidad “relativa”, es decir, que establecen orden de dificultad entre problemas.

# NP-completitud

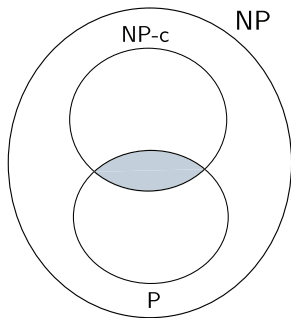
- **Reducción polinomial:** Sean  $\pi$  y  $\pi'$  dos problemas de decisión. Decimos que  $f : D_{\pi'} \rightarrow D_{\pi}$  es una reducción polinomial de  $\pi'$  en  $\pi$  si  $f$  se computa en tiempo polinomial y para todo  $d \in D_{\pi'}$ ,  $d \in Y_{\pi'} \Leftrightarrow f(d) \in Y_{\pi}$ . Notación:  $\pi' \preceq \pi$ .
- Notemos que si  $\pi'' \preceq \pi'$  y  $\pi' \preceq \pi$  entonces  $\pi'' \preceq \pi$ , ya que la composición de dos reducciones polinomiales es una reducción polinomial.
- Un problema  $\pi$  es **NP-completo** si:
  1.  $\pi \in \text{NP}$ .
  2. Para todo  $\pi' \in \text{NP}$ ,  $\pi' \preceq \pi$ .
- Si un problema  $\pi$  verifica la condición 2.,  $\pi$  es NP-Hard (es al menos tan “difícil” como todos los problemas de NP).



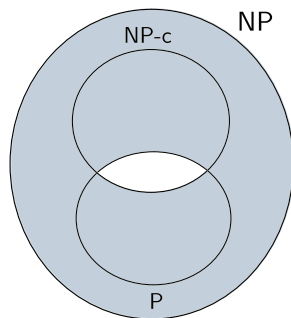
## ¿P ≠ NP? La pregunta del millón...

- **Si existe un problema en  $\text{NP-c} \cap \text{P}$ , entonces  $\text{P}=\text{NP}$ .**
  - Si  $\pi \in \text{NP-c} \cap \text{P}$ , existe un algoritmo polinomial que resuelve  $\pi$ , por estar  $\pi$  en  $\text{P}$ . Por otro lado, como  $\pi$  es NP-completo, para todo  $\pi' \in \text{NP}$ ,  $\pi' \preceq \pi$ .
  - Sea  $\pi' \in \text{NP}$ . Apliquemos la reducción polinomial que transforma instancias de  $\pi'$  en instancias de  $\pi$  y luego el algoritmo polinomial que resuelve  $\pi$ . Por definición de reducción polinomial, es fácil ver que lo que se obtiene es un algoritmo polinomial que resuelve  $\pi'$ .
- Hasta el momento no se conoce ningún problema en  $\text{NP-c} \cap \text{P}$ , así como tampoco se ha demostrado que un problema esté en  $\text{NP} \setminus \text{P}$ . En ese caso, obviamente, se probaría que  $\text{P} \neq \text{NP}$ .

## Esquema de clases



si  $P = NP \dots$



si  $P \neq NP \dots$

## ¿Cómo se prueba que un problema es NP-completo?

El problema SAT consiste en decidir si, dada una fórmula lógica  $\varphi$  expresada como conjunción de disyunciones (ej:  $\varphi = x_1 \wedge (x_2 \vee \neg x_1) \wedge (x_3 \vee \neg x_4 \vee x_1)$ ), existe una valuación de sus variables que haga verdadera  $\varphi$ .

### **Teorema de Cook (1971): SAT es NP-completo.**

La demostración de Cook es directa: considera un problema genérico  $\pi \in \text{NP}$  y una instancia genérica  $d \in D_\pi$ . A partir de la hipotética NDTM que resuelve  $\pi$ , genera en tiempo polinomial una fórmula lógica  $\varphi_{\pi,d}$  en forma normal (conjunción de disyunciones) tal que  $d \in Y_\pi$  si y sólo si  $\varphi_{\pi,d}$  es satisfactible.

## ¿Cómo se prueba que un problema es NP-completo?

A partir del Teorema de Cook, la técnica standard para probar que un problema  $\pi$  es NP-completo aprovecha la transitividad de  $\preceq$ , y consiste en lo siguiente:

1. Mostrar que  $\pi$  está en NP.
2. Elegir un problema  $\pi'$  apropiado que se sepa que es NP-completo.
3. Construir una reducción polinomial  $f$  de  $\pi'$  en  $\pi$ .

La segunda condición en la definición de problema NP-completo sale usando la transitividad: sea  $\pi''$  un problema cualquiera de NP. Como  $\pi'$  es NP-completo,  $\pi'' \preceq \pi'$ . Como probamos que  $\pi' \preceq \pi$ , resulta  $\pi'' \preceq \pi$ .

## Coloreo es NP-completo

A partir de un algoritmo de backtracking para coloreo es fácil construir una NDTM para el problema de coloreo, por lo tanto está en NP.

**Para probar que coloreo es NP-completo, vamos entonces a reducir SAT a coloreo.**

Tomemos una instancia genérica de SAT  $\varphi = C_1 \wedge \dots \wedge C_m$ . Vamos a construir un grafo  $G$  y determinar un número  $k$  de manera que  $\varphi$  sea satisfactible si y sólo si  $G$  se puede colorear con  $k$ -colores.

# Reducción de SAT a coloreo

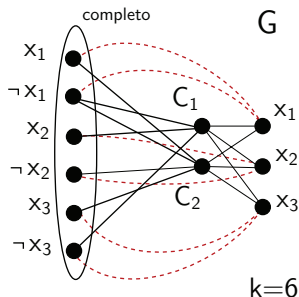
$G$  tiene:

- $V_1$ : un vértice por cada variable negada y afirmada, todos adyacentes entre si.
- $V_2$ : un vértice por cada cláusula, adyacente a los literales de  $V_1$  que no aparecen en la cláusula.
- $V_3$ : otro vértice por cada variable, adyacente a todo  $V_2$  y a los literales de  $V_1$  correspondientes a otras variables.

$k =$  dos veces la cantidad de variables.

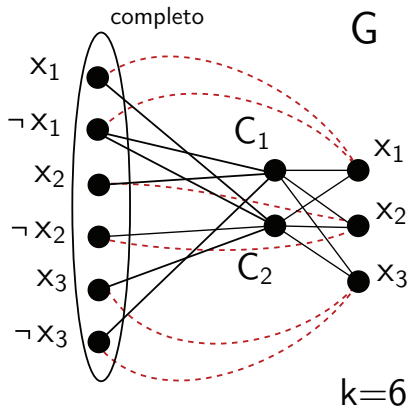
Queda como ejercicio escribir formalmente la reducción y demostrar que es una reducción polinomial de SAT a coloreo.

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$$



## Reducción de SAT a coloreo

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$$



## Reducción de SAT a 3-SAT

El problema 3-SAT es una variante del problema SAT, en el cual cada cláusula tiene exactamente tres literales. Como es una restricción del dominio de SAT, está en NP, y en principio es “no más difícil” que SAT.

Para probar que 3-SAT es NP-completo, vamos entonces a reducir SAT a 3-SAT.

Tomemos una instancia genérica de SAT  $\varphi = C_1 \wedge \cdots \wedge C_m$ .

Vamos a reemplazar cada  $C_i$  por una conjunción de disyunciones  $\varphi'_i$ , donde cada disyunción tenga tres literales, y de manera que  $\varphi$  sea satisfactible si y sólo si  $\varphi_1 \wedge \cdots \wedge \varphi_m$  lo es.



## Reducción de SAT a 3-SAT

- Si  $C_i$  tiene tres literales, queda como está.
- $C_i$  tiene menos de tres literales, agregamos nuevas variables como en el ejemplo:

$$(x_1 \vee \neg x_2) \rightarrow (x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee \neg y)$$

- Si  $C_i$  tiene cuatro o más literales, agregamos nuevas variables como en el ejemplo:

$$(x_1 \vee \neg x_2 \vee x_3 \vee x_4 \vee \neg x_5) \rightarrow \\ (x_1 \vee \neg x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge (\neg y_2 \vee x_4 \vee \neg x_5)$$

Queda como ejercicio escribir formalmente la reducción y demostrar que es una reducción polinomial de SAT a 3-SAT.