Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 5

Clase 5

- Mini-configuraciones
- Teorema de Cook-Levin
- Problemas **NP-completos**
- La clase **coNP**
- Las clases **ExpTime** y **NExpTime**

Mini-configuraciones

Mini-configuración

Para simplificar, consideremos una máquina determinística $M = (\Sigma, Q, \delta)$

- sin cinta de salida,
- con una cinta de entrada y
- con una sola cinta de trabajo (que funciona como salida también)

Todo lo que digamos a continuación se puede generalizar a cualquier cantidad de cintas de trabajo.

Mini-configuración

Supongamos un cómputo

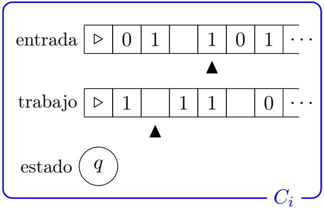
$$C_0, \dots, C_\ell$$

de M una con entrada y. La **mini-configuración en el paso** i es una tupla

$$z_i = (a_i, b_i, c_i) \in \Sigma \times \Sigma \times Q$$

tal que

- a_i es el símbolo leído por la cabeza de entrada en C_i
- b_i es el símbolo leído en la cinta de trabajo en C_i
- c_i es el estado de C_i



$z_i = (1, \square, q)$

Máquina *oblivious*

Supongamos que, además, M es *oblivious*: podemos calcular la posición de las cabezas de entrada y trabajo en el cómputo de $M(y)$ en función de $|y|$ y el número de paso (pero independiente de y).

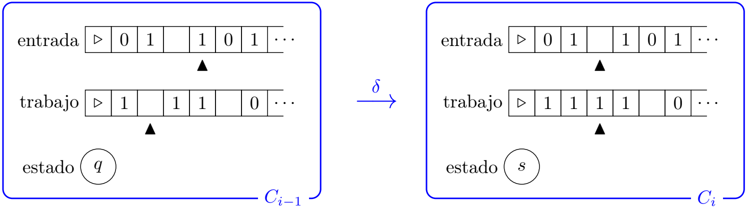
Máquina *oblivious*

Supongamos que, además, M es *oblivious*: podemos calcular la posición de las cabezas de entrada y trabajo en el cómputo de $M(y)$ en función de $|y|$ y el número de paso (pero independiente de y).

Estas tres funciones son computables en tiempo polinomial:

- $e(i, n) =$ posición de la cabeza de entrada en el paso i en el cómputo de M con entrada 0^n
- $t(i, n) =$ posición de la cabeza de trabajo en el paso i en el cómputo de M con entrada 0^n
- $\mathit{prev}(i, n) = \text{máx}\{j < i : t(j, n) = t(i, n)\} \cup \{1\}$, es decir $\mathit{prev}(i, n)$ es el máximo paso $j < i$ en el cómputo de M con entrada 0^n tal que la posición de la cabeza de trabajo en el paso j coincide con la posición en el paso i; o 1 si no existe tal j

Mini-configuración: del paso $i-1$ al paso i



$z_{i-1} = (1, \square, q)$ $\qquad$ $z_i = (\square, 1, s)$

Sea z_i la i-ésima mini-configuración en el cómputo de M con entrada x. $z_0 = (x(0), \square, q_0)$ y para $i > 0$ calculamos z_i con:

- el estado y los símbolos leídos en el paso $i-1$; esto está en z_{i-1}
- la función de transición δ de M
- el contenido de la cinta de entrada (solo lectura; no cambia a lo largo del cómputo) en la posición $e(i, |x|)$
- el contenido de la cinta de trabajo en la posición $prev(i, |x|)$; esto está en $z_{prev(i,|x|)}$

La función F representa la evolución en un paso

Supongamos el cómputo $C_0, \dots, C_\ell$ de $M = (\Sigma, Q, \delta)$ determinística, *oblivious*, sin cinta de salida con entrada y y la correspondiente secuencia de mini-configuraciones

$$z_0, \dots, z_\ell$$

La función F representa la evolución en un paso

Supongamos el cómputo $C_0, \ldots, C_\ell$ de $M = (\Sigma, Q, \delta)$ determinística, *oblivious*, sin cinta de salida con entrada y y la correspondiente secuencia de mini-configuraciones

$$z_0, \ldots, z_\ell$$

Podemos codificar cada mini-configuración $z \in \Sigma \times \Sigma \times Q$ con $\langle z \rangle \in \{0,1\}^k$ con $k = 4 + \lceil \log |Q| \rceil$ (k depende solo de M).

- $00, 11, 01, 10$ codifica cada símbolo de $\Sigma = \{0, 1, \triangleright, \square\}$.
- cada estado de Q se codifica con una cadena en $\{0,1\}^{\lceil \log |Q| \rceil}$; $0 \ldots 0$ codifica q_0; $0 \ldots 1$ codifica q_f

La función F representa la evolución en un paso

Supongamos el cómputo $C_0, \ldots, C_\ell$ de $M = (\Sigma, Q, \delta)$ determinística, *oblivious*, sin cinta de salida con entrada y y la correspondiente secuencia de mini-configuraciones

$$z_0, \ldots, z_\ell$$

Podemos codificar cada mini-configuración $z \in \Sigma \times \Sigma \times Q$ con $\langle z \rangle \in \{0,1\}^k$ con $k = 4 + \lceil \log |Q| \rceil$ (k depende solo de M).

- $00, 11, 01, 10$ codifica cada símbolo de $\Sigma = \{0, 1, \triangleright, \square\}$.
- cada estado de Q se codifica con una cadena en $\{0,1\}^{\lceil \log |Q| \rceil}$; $0 \ldots 0$ codifica q_0; $0 \ldots 1$ codifica q_f

Para $i > 0$, definimos $F : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^2 \to \{0,1\}^k$:

$$F\Big(\underbrace{\langle z_{i-1} \rangle}_{\substack{\text{configuración} \\ \text{anterior} \\ (k \text{ variables})}} \quad , \quad \underbrace{\langle z_{prev(i,|y|)} \rangle}_{\substack{\text{información} \\ \text{de la cinta de} \\ \text{trabajo} \\ (k \text{ variables})}} \quad , \quad \underbrace{\langle x(e(i,|y|)) \rangle}_{\substack{\text{codificación} \\ \text{del bit actual} \\ \text{leído} \\ (2 \text{ variables})}} \Big) = \langle z_i \rangle.$$

y $F(x) = 0^k$ para los otros casos (no nos importa).

La función F representada en CNF

Tenemos

$$F : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^2 \to \{0,1\}^k.$$

Existe una fórmula booleana

$$\varphi_F(\bar{p}, \bar{q}, \bar{r}, \bar{s})$$

con variables libres

- $\bar{p} = p_1, \ldots, p_k$, que codifica $\langle z_{i-1} \rangle$
- $\bar{q} = q_1, \ldots, q_k$ que codifica $\langle z_{prev(i,|y|)} \rangle$
- $\bar{r} = r_1, r_2$, que codifica $\langle y(e(i, |y|)) \rangle$
- $\bar{s} = s_1, \ldots, s_k$, que codifica $\langle z_i \rangle$

en CNF tal que para todo $\bar{a}, \bar{b}, \bar{d} \in \{0,1\}^k$ y $\bar{c} \in \{0,1\}^2$:

$$\bar{a}\bar{b}\bar{c}\bar{d} \models \varphi_F(\bar{p}, \bar{q}, \bar{r}, \bar{s}) \qquad \text{sii} \qquad \bar{d} = F(\bar{a}, \bar{b}, \bar{c})$$

Más aun, podemos computar φ_F a partir de $\langle F \rangle$ en tiempo polinomial y φ_F tiene tamaño $\leq (3k+2)2^{3k+2}$.

Fijada la máquina M, k es constante y luego φ_F tiene tamaño constante.

Teorema de Cook-Levin

SAT es **NP-hard**

Teorema

$\mathsf{SAT} \in \mathbf{NP\text{-}hard}$.

Demostración

Fijemos $\mathcal{L} \in \mathbf{NP}$ y veamos que $\mathcal{L} \leq_{\mathrm{p}} \mathsf{SAT}$.
Como $\mathcal{L} \in \mathbf{NP}$, existe una máquina determinística M tal que

- M corre en tiempo $t(n)$, con t un polinomio
- existe un polinomio p tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u \in \{0,1\}^{p(|x|)}\ M(xu) = 1$$

Definimos $F_x : \{0,1\}^{p(|x|)} \to \{0,1\}$ como $F_x(u) = M(xu)$.
Podemos computar $\varphi_x(q_1, \ldots, q_{p(|x|)}) \in \mathrm{CNF}$ tal que

$$u \models \varphi_x \qquad \text{sii} \qquad F_x(u) = 1 \qquad \text{sii} \qquad M(xu) = 1$$

Luego $x \in \mathcal{L}$ sii φ_x es satisfacible sii $\varphi_x \in \mathsf{SAT}$.

SAT es **NP-hard**

Teorema

SAT $\in$ **NP-hard**.

Demostración ¡incorrecta!

Fijemos $\mathcal{L} \in$ **NP** y veamos que $\mathcal{L} \leq_{\mathrm{p}}$ SAT.
Como $\mathcal{L} \in$ **NP**, existe una máquina determinística M tal que

- M corre en tiempo $t(n)$, con t un polinomio
- existe un polinomio p tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u \in \{0,1\}^{p(|x|)}\ M(xu) = 1$$

Definimos $F_x : \{0,1\}^{p(|x|)} \to \{0,1\}$ como $F_x(u) = M(xu)$.
Podemos computar $\varphi_x(q_1, \ldots, q_{p(|x|)}) \in \mathrm{CNF}$ tal que

$$u \models \varphi_x \qquad \text{sii} \qquad F_x(u) = 1 \qquad \text{sii} \qquad M(xu) = 1$$

Luego $x \in \mathcal{L}$ sii φ_x es satisfacible sii $\varphi_x \in$ SAT.
Problema: φ_x tiene tamaño exponencial: $O(p(|x|)2^{p(|x|)})$.

Demostración.

Fijemos $\mathcal{L} \in \mathbf{NP}$ y veamos que $\mathcal{L} \leq_{\mathrm{p}} \mathsf{SAT}$.
Como $\mathcal{L} \in \mathbf{NP}$, existe una máquina determinística M tal que

- M corre en tiempo $t(n)$, con t un polinomio
- M es *oblivious*, sin cinta de salida y con única cinta de trabajo
- existe un polinomio p tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u \in \{0,1\}^{p(|x|)}\ M(xu) = 1$$

Dado x construimos $\varphi_x \in \mathrm{CNF}$ en tiempo polinomial tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \varphi_x \in \mathsf{SAT}.$$

La máquina M está fija; x es variable.

$x \in \mathcal{L}$

sii $\exists u \in \{0,1\}^{p(|x|)}\ M(xu) = 1$

sii existe $u \in \{0,1\}^{p(|x|)}$ y existe un cómputo $C_0, \ldots, C_{t(|xu|)}$ de M a partir de la entrada xu tal que la salida en $C_{t(|xu|)}$ es 1

sii existe una codificación de la entrada $y \in \{0,1\}^{2n+4}$ con $n = |x| + p(|x|)$ y una secuencia de mini-configuraciones $z_0, \ldots, z_m$, $z_i \in \{0,1\}^k$ (k depende solo de M), con $m = t(n) = t(|x| + p(|x|))$, tal que:

$\psi_1 =$ "la entrada empieza con x y $u \in \{0,1\}^*$"
$\psi_2 =$ "z_0 es la configuración inicial"
$\psi_3 =$ "z_j evoluciona en z_{j+1} para $j = 0, \ldots, m-1$"
$\psi_4 =$ "z_m es una configuración final de M aceptadora"

$x \in \mathcal{L}$

sii $\exists u \in \{0,1\}^{p(|x|)}\ M(xu) = 1$

sii existe $u \in \{0,1\}^{p(|x|)}$ y existe un cómputo $C_0, \ldots, C_{t(|xu|)}$ de M a partir de la entrada xu tal que la salida en $C_{t(|xu|)}$ es 1

sii existe una codificación de la entrada $y \in \{0,1\}^{2n+4}$ con $n = |x| + p(|x|)$ y una secuencia de mini-configuraciones $z_0, \ldots, z_m$, $z_i \in \{0,1\}^k$ (k depende solo de M), con $m = t(n) = t(|x| + p(|x|))$, tal que:

$\psi_1 =$ "la entrada empieza con x y $u \in \{0,1\}^*$"

$\psi_2 =$ "z_0 es la configuración inicial"

$\psi_3 =$ "z_j evoluciona en z_{j+1} para $j = 0, \ldots, m-1$"

$\psi_4 =$ "z_m es una configuración final de M aceptadora"

Veamos que cada una de estas condiciones se expresa con una fórmula

$$\psi_j(\overbrace{\underbrace{p_0, \ldots, p_{2n+3}}_{\text{entrada } y}, \underbrace{q_1^0, \ldots q_k^0}_{z_0}, \ldots, \underbrace{q_1^m, \ldots, q_k^m}_{z_m}}^{\text{cantidad polinomial de variables}})$$

($j = 1 \ldots 4$) en CNF computable en tiempo polinomial a partir de x.

ψ_1 = "la entrada empieza con x y $u \in \{0,1\}^*$"
= y codifica la cinta con contenido "$\triangleright\ x\ u\ \Box$":

- recordar que $00, 11, 01, 10$ codifica cada símbolo de $\Sigma = \{0, 1, \triangleright, \Box\}$.
- $y(0)y(1) = 01$ (marca $\triangleright$)
- $y(2j+2)$ y $y(2j+3)$ codifican a $x(j)$ para $0 \le j \le |x| - 1$
- $y(2j)$ y $y(2j+1)$ tienen el mismo valor para $|x| + 1 \le j \le n$
- $y(2n+2)y(2n+3) = 10$ (marca $\Box$)

se expresa con

$$
\begin{aligned}
\psi_1 = &\neg p_0 \wedge p_1 \\
&\wedge \bigwedge_{j=0\ldots|x|-1} \begin{cases} p_{2j+2} \wedge p_{2j+3} & \text{si } x(j) = 1 \\ \neg p_{2j+2} \wedge \neg p_{2j+3} & \text{si } x(j) = 0 \end{cases} \\
&\wedge \bigwedge_{i=|x|+1\ldots n} p_{2i} \leftrightarrow p_{2i+1} \\
&\wedge p_{2n+2} \wedge \neg p_{2n+3}
\end{aligned}
$$

Notar que no se especifican los valores de las variables correspondientes a u. Observar que $|\psi_1| = O(n)$.

ψ_2 = "z_0 es la configuración inicial"

ψ_2 = "z_0 es la configuración inicial"
= "$z_0 = (x(0), \square, q_0)$" se expresa con

$$\psi_2 = \underbrace{\begin{cases} q_1^0 \wedge q_2^0 & \text{si } x(0) = 1 \\ \neg q_1^0 \wedge \neg q_2^0 & \text{si } x(0) = 0 \end{cases}}_{x(0)} \quad \wedge \quad \underbrace{q_3^0 \wedge \neg q_4^0}_{\square} \quad \wedge \quad \underbrace{\bigwedge_{i=5\ldots k} \neg p_i^0}_{q_0}$$

Observar que $|\psi_1| = O(1)$
(recordar que k depende solo de M pero M está fija)

ψ_3 = "z_j evoluciona en z_{j+1} para $j = 0, \ldots, m-1$". Para $0 < i \leq m$ la condición

$$\underbrace{\langle z_i \rangle}_{k} = F(\underbrace{\langle z_{i-1} \rangle}_{k}, \underbrace{\langle z_{prev(i,n)} \rangle}_{k}, \underbrace{\langle y(e(i,n)) \rangle}_{2})$$

se expresa con una $\psi_3^i \in \mathrm{CNF}$, con ayuda la fórmula φ_F que ya analizamos,

ψ_3 = "z_j evoluciona en z_{j+1} para $j = 0, \ldots, m-1$". Para $0 < i \leq m$ la condición

$$\underbrace{\langle z_i \rangle}_{k} = F(\underbrace{\langle z_{i-1} \rangle}_{k}, \underbrace{\langle z_{prev(i,n)} \rangle}_{k}, \underbrace{\langle y(e(i,n)) \rangle}_{2}))$$

se expresa con una $\psi_3^i \in \mathrm{CNF}$, con ayuda la fórmula φ_F que ya analizamos, en concreto $\psi_3^i =$

$$\varphi_F \left(\underbrace{q_1^{i-1}, \ldots, q_k^{i-1}}_{\langle z_{i-1} \rangle}, \underbrace{q_1^{prev(i,n)}, \ldots, q_k^{prev(i,n)}}_{\langle z_{prev(i,n)} \rangle}, \underbrace{p_{2e(i,n)}, p_{2e(i,n)+1}}_{\langle xu(e(i,n)) \rangle}, \underbrace{q_1^i, \ldots, q_k^i}_{\langle z_i \rangle} \right)$$

Podemos suponer que $e(i,n) \leq n+1$; primera celda de la cinta de entrada es posición 0; M con entrada xu no necesita leer más allá del primer blanco después de xu

ψ_3 = "z_j evoluciona en z_{j+1} para $j = 0, \ldots, m-1$". Para $0 < i \leq m$ la condición

$$\underbrace{\langle z_i \rangle}_{k} = F(\underbrace{\langle z_{i-1} \rangle}_{k}, \underbrace{\langle z_{prev(i,n)} \rangle}_{k}, \underbrace{\langle y(e(i,n)) \rangle}_{2})$$

se expresa con una $\psi_3^i \in$ CNF, con ayuda la fórmula φ_F que ya analizamos, en concreto $\psi_3^i =$

$$\varphi_F \left(\underbrace{q_1^{i-1}, \ldots, q_k^{i-1}}_{\langle z_{i-1} \rangle}, \underbrace{q_1^{prev(i,n)}, \ldots, q_k^{prev(i,n)}}_{\langle z_{prev(i,n)} \rangle}, \underbrace{p_{2e(i,n)}, p_{2e(i,n)+1}}_{\langle xu(e(i,n)) \rangle}, \underbrace{q_1^i, \ldots, q_k^i}_{\langle z_i \rangle} \right)$$

Podemos suponer que $e(i,n) \leq n+1$; primera celda de la cinta de entrada es posición 0; M con entrada xu no necesita leer más allá del primer blanco después de xu

Computamos ψ_3^i en tiempo polinomial y tiene tamaño $O(k2^{3k})$, pero k es constante.

Entonces $\psi_3 = \bigwedge_{i=1,\ldots,m} \psi_3^i$ se construye en tiempo polinomial y expresa $(\forall i = 1, \ldots, t(n))\ z_i = F(z_{i-1}, z_{prev(i,|y|)}, \langle xu(e(i,n)) \rangle)$

ψ_4 $=$ "z_m es una configuración final de M aceptadora"

$= z_m$ es de la forma $(*, 1, q_f)$, se expresa con ψ_4 análogo a ψ_2.

$|\psi_4| = O(1)$ (k depende solo de M pero M está fija)

$$
\begin{aligned}
\psi_4 &= \text{“}z_m \text{ es una configuración final de } M \text{ aceptadora”}\\
&= z_m \text{ es de la forma } (*, 1, q_f)\text{, se expresa con } \psi_4 \text{ análogo a } \psi_2.\\
&|\psi_4| = O(1) \ (k \text{ depende solo de } M \text{ pero } M \text{ está fija})
\end{aligned}
$$

Finalmente,

$$\varphi_x = \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$$

Observar que $|\varphi_x| = O(|x| + t(|x| + p(|x|)))$.

- φ_x se construye en tiempo polinomial en $|x| + p(|x|)$, o sea en tiempo polinomial en $|x|$
- $x \in \mathcal{L}$ sii φ_x es satisfacible sii $\varphi_x \in$ SAT

$\square$

Corolario

SAT $\in$ **NP-completo**.

3SAT es **NP-completo**

Ya vimos que 3SAT es **NP**. Para ver que 3SAT es **NP-hard** probamos

Ejercicio

SAT $\leq_{\mathrm{p}}$ 3SAT.

Corolario

SAT, 3SAT $\in$ **NP-completos**.

Problemas **NP-completos**

Clase 5

Mini-configuraciones

Teorema de Cook-Levin

Problemas **NP-completos**

La clase **coNP**

Las clases **ExpTime** y **NExpTime**

Ejemplo de problema **NP-completo**

Proposición

INDSET $\in$ **NP-completo**.

Demostración.

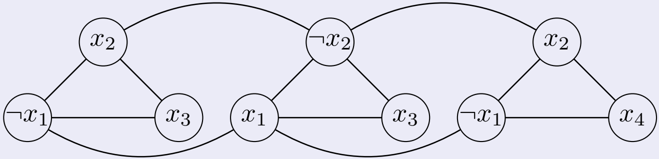
Ya vimos que INDSET es **NP**. Para ver que INDSET es **NP-hard**, probamos que 3SAT $\leq_{\mathrm{p}}$ INDSET. Supongamos una fórmula

$$\varphi = (l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{33}) \wedge \cdots \wedge (l_{m1} \vee l_{m2} \vee l_{m3})$$

en 3CNF con m cláusulas (l_{ij} son literales) y variables $x_1, \ldots, x_n$.

- Definimos un grafo G_φ con $3m$ vértices. Cada vértice corresponde a cada variable de cada cláusula. Supongamos que z es un vértice de G_φ correspondiente a l_{ij} y que z' es un vértice de G_φ correspondiente a $l_{i'j'}$. Definimos una arista entre z y z' si $i = i'$ o l_{ij} es la negación de $l_{i'j'}$ (o viceversa). G_φ es construible en tiempo polinomial en $|\varphi|$.
- Probamos que φ es satisfacible sii G_φ tiene un conjunto independiente de al menos m vértices.

$$\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

$G_\varphi =$ 

- Supongamos que $v \models \varphi$. Entonces para todo $i = 1, \ldots, m$ tenemos $v \models l_{i1} \vee l_{i2} \vee l_{i3}$, de modo que $v \models l_{ij}$ para algún j. Sea S el conjunto de vértices $z_1, \ldots, z_m$ tal que z_i corresponde a l_{ij} y $v \models l_{ij}$. S es independiente: si existiera arista entre z_i y $z_{i'}$ sería porque 1) $i = i'$ o 2) z_i corresponde a l_{ij}, $z_{i'}$ corresponde a $l_{i'j'}$ y l_{ij} es la negación de $l_{i'j'}$ (o viceversa). Ninguna puede pasar.
- Si S es un conjunto independiente en G_φ de m vértices, tenemos exactamente un vértice en cada "triángulo". Para cada $z \in S$:
  - si z corresponde a x_i definimos $v(x_i) = 1$
  - si z corresponde a $\neg x_i$ definimos $v(x_i) = 0$

  Para todas las otras variables x para las que no está definido v, definimos $v(x)$ de forma arbitraria. v está bien definida porque S es independiente; $v \models \varphi$. □

Camino hamiltoniano

Un **camino hamiltoniano** en un grafo dirigido G es un camino que visita todos los vértices de G_φ exactamente una vez.

Problema: CAMHAM (Camino hamiltoniano)

$$\mathsf{CAMHAM} = \{\langle G\rangle \colon G \text{ tiene un camino hamiltoniano}\}$$

Proposición

$\mathsf{CAMHAM} \in$ **NP-completo**.

Problema del viajante de comercio (*travelling salesman problem*)

Dadas n ciudades, representamos la distancia entre cada par de ciudades por medio de una matriz M de $n \times n$.

Problema: TSP (Problema del viajante de comercio o *Travelling Salesman Problem*)

$$\mathsf{TSP} = \left\{ \langle M, k \rangle : \begin{array}{l} \text{hay una ruta de distancia} \leq k \text{ (de acuerdo a } M\text{)} \\ \text{que visita todas las ciudades de } G \text{ exactamente} \\ \text{una vez y al finalizar vuelve a la ciudad de origen} \end{array} \right\}$$

Proposición

$\mathsf{TSP} \in$ **NP-completo**.

Problema de la mochila (*knapsack problem*)

Representamos una lista de n ´tems en una mochila con su valor y su peso por medio de una lista

$$M = [(v_1, p_1), (v_2, p_2), \ldots, (v_n, p_n)].$$

Problema: KNAPSACK (Problema de la mochila o *knapsack problem*)

$$\mathsf{KNAPSACK} = \left\{ \langle M, v, p \rangle : \begin{array}{l} \text{existe un conjunto de ítems por un valor} \\ \text{total} \geq v \text{ y con un peso} \leq p \end{array} \right\}$$

Proposición

$\mathsf{KNAPSACK} \in$ **NP-completo**.

La clase **coNP**

En general: problemas C-completos y C-hard

La noción de **NP-hard** y **NP-completo** se aplica a otras clases de complejidad.

En general, si **C** es una clase de complejidad, entonces

Clase de complejidad: C-hard, C-completo

$\mathcal{L}$ es **C-hard** si $\mathcal{L}' \leq_{\mathrm{p}} \mathcal{L}$ para todo $\mathcal{L}' \in \mathbf{C}$.
$\mathcal{L}$ es **C-completo** si $\mathcal{L} \in \mathbf{C}$ y $\mathcal{L}$ es **C-hard**.

- En realidad, estas son nociones de completitud y hardness para la reducción $\leq_{\mathrm{p}}$.
- Más adelante veremos clases para las que no tiene sentido usar $\leq_{\mathrm{p}}$ y necesitamos reducciones más débiles.

En general: problemas $\mathbf{coC}$

Notación: Complemento de un lenguaje

$\overline{\mathcal{L}} = \{0,1\}^* \setminus \mathcal{L}$ es el complemento de $\mathcal{L}$.

Clase de complejidad: $\mathbf{coC}$

Si $\mathbf{C}$ es una clase de complejidad, definimos

$$\mathbf{coC} = \{\mathcal{L} \colon \overline{\mathcal{L}} \in \mathbf{C}\}.$$

La clase coNP

Clase de complejidad: coNP

$\mathbf{coNP} = \{\mathcal{L} \colon \overline{\mathcal{L}} \in \mathbf{NP}\}$.

Es decir, **coNP** es la clase de lenguajes $\mathcal{L}$ tal que existe un polinomio $p : \mathbb{N} \to \mathbb{N}$ y una máquina determinística M tal que

- M corre en tiempo polinomial
- para todo x:

$$x \in \mathcal{L} \quad \text{sii} \quad \text{para todo } u \in \{0,1\}^{p(|x|)} \; M(\langle x, u \rangle) = 1$$

Relación de $\mathbf{coNP}$ con $\mathbf{P}$ y $\mathbf{NP}$

Ejercicio

$\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$.

Ejercicio

Si $\mathbf{P} = \mathbf{NP}$ entonces $\mathbf{NP} = \mathbf{coNP}$.

Ejemplo de problema **coNP-completo**

Problema: Tautología

$$\mathsf{TAUT} = \{\langle\varphi\rangle : \varphi \in \text{CNF es una tautología}\}$$

Observar que φ es una tautología sii $\neg\varphi$ es insatisfacible.

$$\langle\varphi\rangle \in \mathsf{TAUT} \qquad \text{sii} \qquad \langle\neg\varphi\rangle \notin \mathsf{SAT}$$

Ejercicio

$\mathsf{TAUT} \in$ **coNP-completo**.

Las clases **ExpTime** y **NExpTime**

Las clases ExpTime y NExpTime

Clase de complejidad: ExpTime y NExpTime

$$\textbf{ExpTime} = \bigcup_{c>0} \textbf{DTime}(2^{n^c}).$$

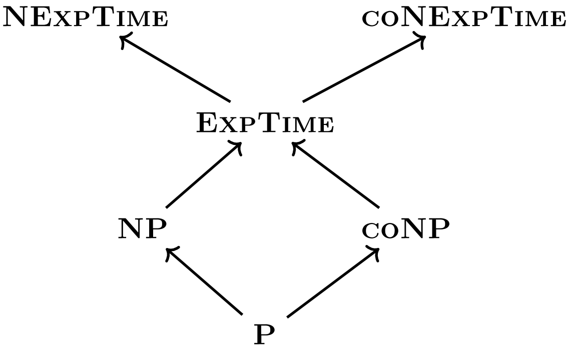
$$\textbf{NExpTime} = \bigcup_{c>0} \textbf{NDTime}(2^{n^c}).$$

Son los análogos de **P** y **NP** pero con tiempo exponencial.

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \textsc{ExpTime} \subseteq \textsc{NExpTime}$$

Ejercicio

$\mathbf{NP} \subseteq \textsc{ExpTime}$.


NExpTime
coNExpTime
ExpTime
NP
coNP
P

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Consideremos $\mathcal{L}_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}} \rangle \colon x \in \mathcal{L}\}$.

Veamos que $\mathcal{L}_{\text{pad}} \in \mathbf{NP}$.

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Consideremos $\mathcal{L}_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}}\rangle \colon x \in \mathcal{L}\}$.

Veamos que $\mathcal{L}_{\text{pad}} \in \mathbf{NP}$. Definimos una máquina no-determinística N' tal que con entrada y hace esto:

si no existe z tal que $y = \langle z, 1^{2^{|y|^c}}\rangle$, rechazar

si no, y es de la forma $\langle z, 1^{2^{|y|^c}}\rangle$

simular N con entrada z por $c \cdot 2^{|z|^c}$ pasos

devolver la salida de esta simulación

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Consideremos $\mathcal{L}_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}} \rangle : x \in \mathcal{L}\}$.

Veamos que $\mathcal{L}_{\text{pad}} \in \mathbf{NP}$. Definimos una máquina no-determinística N' tal que con entrada y hace esto:

si no existe z *tal que* $y = \langle z, 1^{2^{|y|^c}} \rangle$, *rechazar*

si no, y *es de la forma* $\langle z, 1^{2^{|y|^c}} \rangle$

simular N *con entrada* z *por* $c \cdot 2^{|z|^c}$ *pasos*

devolver la salida de esta simulación

N' corre en tiempo polinomial (en $|y|$) y por lo tanto $\mathcal{L}_{\text{pad}} \in \mathbf{NP} = \mathbf{P}$.

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Consideremos $\mathcal{L}_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}}\rangle \colon x \in \mathcal{L}\}$.

Veamos que $\mathcal{L}_{\text{pad}} \in \mathbf{NP}$. Definimos una máquina no-determinística N' tal que con entrada y hace esto:

si no existe z tal que $y = \langle z, 1^{2^{|y|^c}}\rangle$, rechazar

si no, y es de la forma $\langle z, 1^{2^{|y|^c}}\rangle$

simular N con entrada z por $c \cdot 2^{|z|^c}$ pasos

devolver la salida de esta simulación

N' corre en tiempo polinomial (en $|y|$) y por lo tanto $\mathcal{L}_{\text{pad}} \in \mathbf{NP} = \mathbf{P}$. Veamos que $\mathcal{L}_{\text{pad}} \in \textbf{DTime}(2^{n^c})$.

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Consideremos $\mathcal{L}_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}} \rangle : x \in \mathcal{L}\}$.

Veamos que $\mathcal{L}_{\text{pad}} \in \mathbf{NP}$. Definimos una máquina no-determinística N' tal que con entrada y hace esto:

si no existe z tal que $y = \langle z, 1^{2^{|y|^c}} \rangle$, rechazar
si no, y es de la forma $\langle z, 1^{2^{|y|^c}} \rangle$
  simular N con entrada z por $c \cdot 2^{|z|^c}$ pasos
  devolver la salida de esta simulación

N' corre en tiempo polinomial (en $|y|$) y por lo tanto $\mathcal{L}_{\text{pad}} \in \mathbf{NP} = \mathbf{P}$. Veamos que $\mathcal{L}_{\text{pad}} \in \textbf{DTime}(2^{n^c})$. Definimos una máquina determinística M tal que dada la entrada x:

computa $y = \langle x, 1^{2^{|x|^c}} \rangle$ *(tiempo $O(2^{|x|^c})$)*
decide si $y \in \mathcal{L}_{\text{pad}}$ (sii $x \in \mathcal{L}$) *(tiempo poli en $|y|$)*

Teorema

Si $\mathbf{P} = \mathbf{NP}$ entonces $\textbf{ExpTime} = \textbf{NExpTime}$.

Demostración.

Supongamos que $\mathbf{P} = \mathbf{NP}$. Tomemos $\mathcal{L} \in \textbf{NDTime}(2^{n^c})$ y N una máquina no-determinística que decide $\mathcal{L}$ en tiempo $c \cdot 2^{n^c}$.

Consideremos $\mathcal{L}_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}}\rangle : x \in \mathcal{L}\}$.

Veamos que $\mathcal{L}_{\text{pad}} \in \mathbf{NP}$. Definimos una máquina no-determinística N' tal que con entrada y hace esto:

si no existe z tal que $y = \langle z, 1^{2^{|y|^c}}\rangle$, rechazar
si no, y es de la forma $\langle z, 1^{2^{|y|^c}}\rangle$
simular N con entrada z por $c \cdot 2^{|z|^c}$ pasos
devolver la salida de esta simulación

N' corre en tiempo polinomial (en $|y|$) y por lo tanto $\mathcal{L}_{\text{pad}} \in \mathbf{NP} = \mathbf{P}$. Veamos que $\mathcal{L}_{\text{pad}} \in \textbf{DTime}(2^{n^c})$. Definimos una máquina determinística M tal que dada la entrada x:

computa $y = \langle x, 1^{2^{|x|^c}}\rangle$ *(tiempo $O(2^{|x|^c})$)*
decide si $y \in \mathcal{L}_{\text{pad}}$ (sii $x \in \mathcal{L}$) *(tiempo poli en $|y|$)*

M corre en tiempo $O(2^{|x|^{c+d}})$ y decide $\mathcal{L}$, luego $\mathcal{L} \in \textbf{DTime}(2^{|x|^c})$. □

$\mathbf{P}$ vs $\mathbf{NP}$

- ¿$\mathbf{P} = \mathbf{NP}$ o $\mathbf{P} \subsetneq \mathbf{NP}$? Es una pregunta abierta.
- ¿Reconocer la corrección de una solución es esencialmente más fácil que generarla?
    - Por ejemplo, dado un sistema axiomático $\mathcal{S}$ (= axiomas + reglas de inferencia) el lenguaje

      $$\{\langle \varphi, 1^n \rangle : \varphi \text{ tiene una demostración en } \mathcal{S} \text{ de longitud} \leq n\}$$

      es $\mathbf{NP}$. Pero verificar que una secuencia de pasos es una demostración es $\mathbf{P}$.
- En algunos casos, ¿lo mejor que podemos hacer es usar la fuerza bruta para llegar a la solución?