# Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 3

Clase 3

- Halt y máquina universal
- La clase **P**
- La clase **NP**

# Halt y máquina universal

# El problema de la detención (*halting problem*)

Definimos $halt : \{0,1\}^* \to \{0,1\}$ como

$$halt(x) = \begin{cases} 1 & \text{si la máquina } x \text{ con entrada } x \text{ termina} \\ 0 & \text{si no} \end{cases}$$

**Teorema (Turing 1936)**

*halt* no es computable.

# El problema de la detención (*halting problem*)

Definimos $halt : \{0,1\}^* \to \{0,1\}$ como

$$halt(x) = \begin{cases} 1 & \text{si la máquina } x \text{ con entrada } x \text{ termina} \\ 0 & \text{si no} \end{cases}$$

**Teorema (Turing 1936)**

*halt* no es computable.

**Demostración.**

Supongamos que *halt* es computable. Definimos una máquina $M$ tal que $M(x)$ termina sii $halt(x) = 0$. Sea $y = \langle M \rangle$.

$$\begin{aligned} M(y) \text{ termina} \quad & \text{sii la máquina } y \text{ con entrada } y \text{ no termina} \\ & \text{sii } M(y) \text{ no termina} \end{aligned}$$

Absurdo. □

# La máquina universal

Llamemos $M_i$ a la máquina tal que $\langle M \rangle = i$. Definimos

$$u : \subseteq \{0,1\}^* \to \{0,1\}^*$$
$$u(\langle i, x \rangle) = M_i(x)$$

La función $u$ es parcial. En particular $u(\langle i, x \rangle) \downarrow$ sii $M_i(x) \downarrow$.

# La máquina universal

Llamemos $M_i$ a la máquina tal que $\langle M \rangle = i$. Definimos

$$u : \subseteq \{0,1\}^* \to \{0,1\}^*$$
$$u(\langle i, x \rangle) = M_i(x)$$

La función $u$ es parcial. En particular $u(\langle i, x \rangle) \downarrow$ sii $M_i(x) \downarrow$.

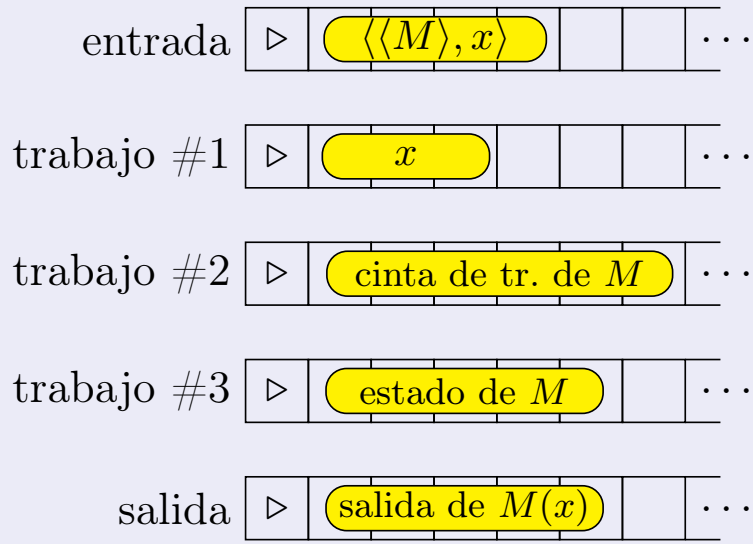
## Teorema

Existe una máquina $U$ que computa la función $u(\langle i, x \rangle) = M_i(x)$. Más aún, si $M_i$ con entrada $x$ termina en $t$ pasos, entonces $U$ con entrada $\langle i, x \rangle$ termina en $c \cdot t \cdot \log t$ pasos, donde $c$ depende solo de $i$.

## Idea de la prueba (un caso simple).

$U$ tiene 3 cintas de trabajo.
Supongamos $M = (\Sigma, Q, \delta)$ con *una sola* cinta de trabajo.

- $U$ recibe como entrada $\langle\langle M\rangle, x\rangle$. Objetivo: copiar $M(x)$
- $\langle M\rangle$ tiene la información sobre el comportamiento de $M$
- $U$ copia $x$ en la cinta de trabajo #1
- $U$ escribe $[q_0]$ en la cinta de trabajo #3
- mientras que lo que esté escrito en en la cinta #3 no sea $[q_f]$, busca en la cinta de entrada la información sobre $\delta$ que necesita y actualiza las cintas #2 y #3 de manera acorde



Si $M = (\Sigma, Q, \delta)$ con entrada $x$ termina en $t$ pasos, esta simulación usa $c \cdot t$ pasos, donde $c$ depende de $M$ pero no de $x$

- $U$ tiene que buscar la definición de $\delta$ en la cinta de entrada por cada instrucción que simula de $M$

Pero esto es una simplificación del caso general:

- $U$ está simulando una $M$ con 1 sola cinta de trabajo
- si $M$ tiene más cintas de trabajo, se puede transformar en una $M'$ que usa *una sola* cinta con el mismo comportamiento de $M$
- si $M$ llegaba al resultado en tiempo $t$, $M'$ lo hace en tiempo $O(t^2)$. Esto es demasiado para la cota $O(t \log t)$ del teorema.
- la demostración del caso general es más técnica

# La máquina universal con tiempo acotado

Llamemos $M_i$ a la máquina tal que $\langle M \rangle = i$.

$$\tilde{u} : \{0,1\}^* \to \{0,1\}^*$$

$$\tilde{u}(\langle i, t, x \rangle) = \begin{cases} 1\ M_i(x) & \text{si } M_i(x) \text{ termina en} \leq t \text{ pasos} \\ 0 & \text{si no} \end{cases}$$

La función $\tilde{u}$ es total.

# La máquina universal con tiempo acotado

Llamemos $M_i$ a la máquina tal que $\langle M \rangle = i$.

$$\tilde{u} : \{0,1\}^* \to \{0,1\}^*$$

$$\tilde{u}(\langle i, t, x \rangle) = \begin{cases} 1\ M_i(x) & \text{si } M_i(x) \text{ termina en} \leq t \text{ pasos} \\ 0 & \text{si no} \end{cases}$$

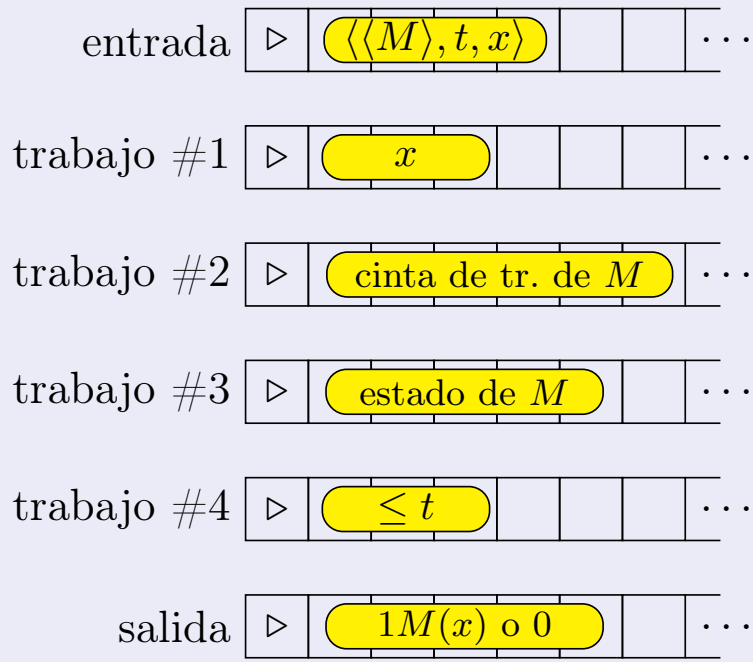
La función $\tilde{u}$ es total.

**Teorema**

Existe una máquina $\tilde{U}$ que computa la función $\tilde{u}(\langle i, t, x \rangle)$ en tiempo $c \cdot t \cdot \log t$, donde $c$ depende solo de $i$.

## Idea de la prueba.

Como antes, pero ahora $\tilde{U}$ tiene una cinta de trabajo más, que usa para llevar cuenta de la cantidad de pasos en la simulación



# La clase **P**

# Clases de complejidad y la clase $\mathbf{DTime}(T(n))$

Sea $\mathcal{L} \subseteq \{0,1\}^*$ un lenguaje (o problema) y sea $\Sigma$ el alfabeto estándar.

## Definición

Una máquina $M = (Q, \Sigma, \delta)$ **decide** $\mathcal{L}$ [en tiempo $T(n)$] si $M$ computa $\chi_{\mathcal{L}}$ [en tiempo $T(n)$]. Decimos que $M$ **acepta** $x$ cuando $M(x) = 1$ y que $M$ **rechaza** $x$ cuando $M(x) = 0$.

# Clases de complejidad y la clase $\mathbf{DTime}(T(n))$

Sea $\mathcal{L} \subseteq \{0,1\}^*$ un lenguaje (o problema) y sea $\Sigma$ el alfabeto estándar.

## Definición

Una máquina $M = (Q, \Sigma, \delta)$ **decide** $\mathcal{L}$ [en tiempo $T(n)$] si $M$ computa $\chi_{\mathcal{L}}$ [en tiempo $T(n)$]. Decimos que $M$ **acepta** $x$ cuando $M(x) = 1$ y que $M$ **rechaza** $x$ cuando $M(x) = 0$.

## Notación: $\mathcal{L}(M)$

El lenguaje decidido por $M$ es $\mathcal{L}(M)$.

# Clases de complejidad y la clase $\mathbf{DTime}(T(n))$

Sea $\mathcal{L} \subseteq \{0,1\}^*$ un lenguaje (o problema) y sea $\Sigma$ el alfabeto estándar.

## Definición

Una máquina $M = (Q, \Sigma, \delta)$ **decide** $\mathcal{L}$ [en tiempo $T(n)$] si $M$ computa $\chi_{\mathcal{L}}$ [en tiempo $T(n)$]. Decimos que $M$ **acepta** $x$ cuando $M(x) = 1$ y que $M$ **rechaza** $x$ cuando $M(x) = 0$.

## Notación: $\mathcal{L}(M)$

El lenguaje decidido por $M$ es $\mathcal{L}(M)$.

Una **clase de complejidad** es un conjunto de problemas que son decidibles por máquinas con ciertos recursos acotados.
Por ahora, nos centramos en el recurso *tiempo de cómputo*.

# Clases de complejidad y la clase $\mathbf{DTime}(T(n))$

Sea $\mathcal{L} \subseteq \{0,1\}^*$ un lenguaje (o problema) y sea $\Sigma$ el alfabeto estándar.

## Definición

Una máquina $M = (Q, \Sigma, \delta)$ **decide** $\mathcal{L}$ [en tiempo $T(n)$] si $M$ computa $\chi_{\mathcal{L}}$ [en tiempo $T(n)$]. Decimos que $M$ **acepta** $x$ cuando $M(x) = 1$ y que $M$ **rechaza** $x$ cuando $M(x) = 0$.

## Notación: $\mathcal{L}(M)$

El lenguaje decidido por $M$ es $\mathcal{L}(M)$.

Una **clase de complejidad** es un conjunto de problemas que son decidibles por máquinas con ciertos recursos acotados.
Por ahora, nos centramos en el recurso *tiempo de cómputo*.

## Clase de complejidad: $\mathbf{DTime}(T(n))$

$\mathbf{DTime}(T(n))$ es la clase de lenguajes $\mathcal{L}$ tal que existe una máquina $M$ que decide $\mathcal{L}$ en tiempo $O(T(n))$.

# La clase **P**

**Clase de complejidad: P**

$$\mathbf{P} = \bigcup_{c>0} \mathbf{DTime}(n^c)$$

Son los problemas que se resuelven por una máquina en tiempo polinomial respecto al tamaño de su entada.

# La clase **P**

## Clase de complejidad: **P**

$$\mathbf{P} = \bigcup_{c>0} \mathbf{DTime}(n^c)$$

Son los problemas que se resuelven por una máquina en tiempo polinomial respecto al tamaño de su entada.

## Observación

Por ahora, una 'máquina' $M = (\Sigma, Q, \delta)$ es un dispositivo *determinístico*. De ahí la 'D' en **DTime**. Más adelante veremos máquinas *no-deterministas*.

# La clase $\mathbf{P}$

## Clase de complejidad: $\mathbf{P}$

$$\mathbf{P} = \bigcup_{c>0} \mathbf{DTime}(n^c)$$

Son los problemas que se resuelven por una máquina en tiempo polinomial respecto al tamaño de su entada.

## Observación

Por ahora, una 'máquina' $M = (\Sigma, Q, \delta)$ es un dispositivo *determinístico*. De ahí la 'D' en $\mathbf{DTime}$. Más adelante veremos máquinas *no-determinísticas*.

Decimos "el problema $X$ está en $\mathbf{P}$" o "el problema $X$ se resuelve en tiempo polinomial" para referirnos a que el lenguaje de decisión que corresponde a $X$ está en $\mathbf{P}$.

$\mathbf{P}$ es considerada la clase de problemas "factibles".

# Ejemplo de problema en $\mathbf{P}$

Un grafo $G = (V, E)$ es *conexo* si todo par de nodos de $G$ está unido por un camino.

**Problema: Conectividad de un grafo**

$$\mathsf{CON} = \{\langle G \rangle : G \text{ es un grafo conexo}\}$$

# Ejemplo de problema en $\mathbf{P}$

Un grafo $G = (V, E)$ es *conexo* si todo par de nodos de $G$ está unido por un camino.

**Problema: Conectividad de un grafo**

$$\mathsf{CON} = \{\langle G \rangle : G \text{ es un grafo conexo}\}$$

- Representamos $G = (V, E)$ con su matriz de adyacencia
- Suponemos $V = \{0, \ldots, k-1\}$
- Hay una máquina $M$ que hace esto:

  *Explora $G$ usando* depth first search *desde $u$ y marca los nodos visitados. Si quedó un nodo sin visitar, escribe 0 en la salida; si no, escribe un 1. Luego termina (que quiere decir pasar a $q_f$).*

- Si $n = |\langle G \rangle|$ (siempre $n$ va a representar el tamaño de la entrada), $M$ llega al resultado en tiempo $O(n^2)$, entonces $\mathsf{CON} \in \mathbf{P}$.

# Simplificación de máquinas para problemas de decisión

Para calcular funciones valuadas en $\{0, 1\}$, no hace falta una cinta de salida (porque solo hay que devolver un 0 o un 1). Podemos trabajar con máquinas que solo tienen una cinta de entrada, una trabajo y

- dejar la salida (0 o 1) en la celda en la que termina la cabeza de trabajo cuando entra a $q_f$, o bien
- reemplazar $q_f$ por un $q_{\text{sí}}$ (que significa que devuelve 1) y un $q_{\text{no}}$ (que significa que devuelve 0); el estado de la configuración final es ahora $q_{\text{sí}}$ o bien $q_{\text{no}}$.

Llamemos a cualquiera de estas variantes una **máquina sin cinta de salida**.

# Simplificación de máquinas para problemas de decisión

Para calcular funciones valuadas en $\{0, 1\}$, no hace falta una cinta de salida (porque solo hay que devolver un 0 o un 1). Podemos trabajar con máquinas que solo tienen una cinta de entrada, una trabajo y

- dejar la salida (0 o 1) en la celda en la que termina la cabeza de trabajo cuando entra a $q_f$, o bien
- reemplazar $q_f$ por un $q_{\text{sí}}$ (que significa que devuelve 1) y un $q_{\text{no}}$ (que significa que devuelve 0); el estado de la configuración final es ahora $q_{\text{sí}}$ o bien $q_{\text{no}}$.

Llamemos a cualquiera de estas variantes una **máquina sin cinta de salida**.

## Proposición

Si $\mathcal{L}$ es decidible en tiempo $T(n)$ por una máquina estándar de 3 cintas (entrada, trabajo y salida), entonces $\mathcal{L}$ es decidible en tiempo $O(T(n))$ por una máquina sin cinta de salida.

# La clase **NP**

# La clase **NP**

## Clase de complejidad: **NP**

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

- $M$ corre en tiempo polinomial
- para todo $x$:

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

$M$ se llama el **verificador para** $\mathcal{L}$; $u$ se llama **certificado para** $x$.

# La clase **NP**

## Clase de complejidad: **NP**

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

- $M$ corre en tiempo polinomial
- para todo $x$:

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

$M$ se llama el **verificador para** $\mathcal{L}$; $u$ se llama **certificado para** $x$.

Notar que la segunda condición es equivalente a

$$x \in \mathcal{L} \quad \text{sii} \quad \text{existe } u \in \{0,1\}^{p(|x|)} \text{ tal que } M(xu) = 1$$

porque $M(y)$ puede primero contar la cantidad $m$ de celdas en la cinta de entrada hasta el primer blanco y luego buscar el primer $n$ tal que $m = n + p(n)$. Tenemos que

$$y(0), \ldots, y(n-1) = x \quad \text{y} \quad y(n), \ldots, y(n + p(n) - 1) = u$$

**Teorema**

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

## Teorema

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

## Demostración.

Supongamos que $\mathcal{L} \in \mathbf{P}$. Supongamos una máquina $M$ tal que $M$ decide $\mathcal{L}$ en tiempo polinomial. Tomamos $p(n) = 0$. Definimos la máquina $M'$ que, dado $\langle x, \epsilon \rangle$ como entrada, simula $M$ con entrada $x$. Como $M$ corre en tiempo polinomial, $M'$ también.

$$
\begin{array}{lll}
x \in \mathcal{L} & \text{sii} & M(x) = 1 \\
 & \text{sii} & M'(\langle x, \epsilon \rangle) = 1 \\
 & \text{sii} & \text{existe } u \in \{0,1\}^0 \text{ tal que } M'(\langle x, u \rangle) = 1
\end{array}
$$

□

# Ejemplo de problema en **NP**

Un conjunto $X$ de nodos de un grafo $G = (V, E)$ es **independiente** si no existen $u, v \in X$ tal que $(u, v) \in E$.

**Problema: Conjunto independiente**

$$\mathsf{INDSET} = \{\langle G, k\rangle \mid \begin{array}{l} G \text{ tiene un conjunto inde-} \\ \text{pendiente de} \geq k \text{ vértices} \end{array} \}$$

# Ejemplo de problema en **NP**

Un conjunto $X$ de nodos de un grafo $G = (V, E)$ es **independiente** si no existen $u, v \in X$ tal que $(u, v) \in E$.

## Problema: Conjunto independiente

$$\mathsf{INDSET} = \{\langle G, k\rangle \mid \begin{array}{l} G \text{ tiene un conjunto inde-} \\ \text{pendiente de} \geq k \text{ vértices} \end{array} \}$$

- suponemos que $V = \{[0], [1], \ldots, [|V| - 1]\}$
- certificado: lista de $k$ nodos distintos de $V$ que forman un conjunto independiente
- podemos codificar ese certificado en una palabra $u$ de tamaño $O(k\lceil \log |V| \rceil)$. Sea $n = |\langle\langle G\rangle, [k]\rangle|$. Como $k \leq |V|$, entonces $|u| = O(n \log n)$ (usar codificación de listas)
- $|u| = p(n)$ para un polinomio cuadrático $p$ (completar con 0s al final)
- $M$ recibe como entrada $x$. Si $x$ no es de la forma $\langle\langle G, k\rangle, u\rangle$ con $|u| = p(n)$, escribe 0 en la salida. Si no: si $u$ codifica un conjunto independiente de $G$ de $k$ nodos, escribe 1 en la salida; si no, escribe 0.
- $M$ corre en tiempo polinomial.
- $x \in \mathsf{INDSET}$ sii existe $u \in \{0, 1\}^{p(|x|)}$ tal que $M(\langle x, u\rangle) = 1$.