

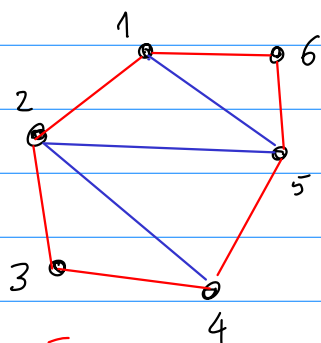
# ① Complejidad computacional.

Resumen. La teoría de complejidad computacional se ocupa principalmente de **PROBLEMAS DE DECISIÓN**, es decir preguntas con respuesta **Si** ó **NO**.  
1                      0

Y lo que queremos entender es qué tan difícil (en términos de recursos de cómputo) es responder una pregunta dada en una colección de instancias del problema de diferentes longitudes.

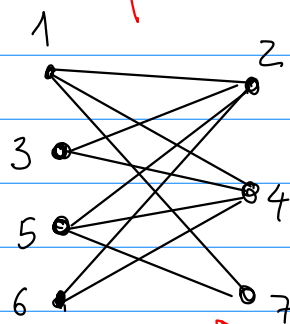
## Ejemplo:


Def: Un camino hamiltoniano en un grafo  $G$  es un ciclo simple (es decir una sucesión  $v_1, v_2, \dots, v_k \in V(G)$  con  $(v_i, v_{i+1}) \in E(G)$  con  $v_i \neq v_j$   $i \neq j$  que forma un ciclo  $(v_k, v_1) \in E(G)$ ) que contiene a todos los vértices de  $G$



Ejemplo

1, 2, 3, 4, 5, 6  
es un ciclo  
hamiltoniano



Ejemplo:  no  
contiene ningún ciclo  
Hamiltoniano

PROBLEMA: Contiene  $G$  a un ciclo Hamiltoniano?

input.  $G$   
muchos tamaños  
posibles, lo  
denotaremos por  
 $n$ .

Respuesta es SI o NO  
(para cada grafo  $G$ )

cuánto tiempo  
repite responder  
la pregunta?

$O(n)$ ?  $O(n^2)$ ? ...  $O(n^8)$ ?  
 $O(2^n)$ ?  $O(n^n)$ ?

Depende del algoritmo así que preguntamos:  
cuál es el tiempo necesario para el  
algoritmo más eficiente posible?  
(el mínimo tiempo)

Hay muchas maneras de codificar el input de un problema

es decir, escribirlo mediante un conjunto  
de 0's y 1's para que pueda ser  
procesado por un computador

La codificación escogida cambia el tiempo de ejecución  
como muestra el siguiente ejemplo:

A Algoritmo: Produce un entero  $x$  en representación UNARIA  
 $6 \mapsto 111111 \leftarrow$  así:

INPUT EN UNARIO  
111 111  
longitud  $n$

$A$

OUTPUT  
111111  
longitud  $n$

tiempo =  $O(n)$

INPUT EN BINARIO

6 = 110

longitud  $n$

A

OUTPUT

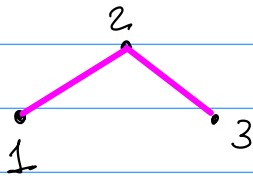
111111

longitud  $2^{n+1}$

tiempo =  $O(2^{n+1})$

Así que de ahora en adelante fixamos una codificación.

$\langle G \rangle$  para un grafo  $G$  usando enteros en binario y una descripción mediante paréntesis y listas de adyacencia



Codificado como

$[\{1,2\}, \{2,3\}] \mapsto$

$[\{1, 10\}, \{10, 11\}]$

binario

Buscar  
what is ASCII  
en internet

"0010....."

binario de  
cierta longitud.

determinada por  
 $n+m$

Usando la codificación nuestra pregunta  
se vuelve bastante precisa

[Ciclo-HAMILTONIANO]

Contiene  $G$  a  
un ciclo Hamiltoniano?

1 = SI

0 = NO

$\langle G \rangle$  codificación  
de un grafo  
no dirigido  $G$ .  
de longitud  $n$

Cuál es el tiempo mínimo

\* para un algoritmo que  
hace esto?

\* Para simplificar nuestro problema vamos a hacer una gran simplificación:

PROBLEMAS "FÁCILES"

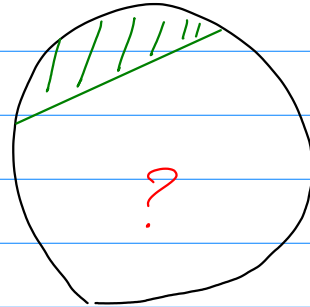
Aquellos que admiten un algoritmo que corre en tiempo polinomial

[Es decir  $\exists k \in \mathbb{N}$ :  
la pregunta se resuelve en tiempo  $O(n^k)$   
para todo input de longitud  $n$ ]

Clase de complejidad  $P$

PROBLEMAS "DIFÍCILES"

Todos los demás.



Hay muchas otras clases de complejidad

$$P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME \subseteq EXPSPACE$$

Motivación para pensar en "P" como una clase natural:

- Bastante independiente del encoding (problema más robusto)
- En ejemplos la complejidad típica de los problemas en  $P$  es  $O(n^k)$  para  $k$  pequeño...

Habiendo fijado una codificación para nuestro problema podemos pensar que los inputs codificados son solamente ciertas sucesiones de 0s y 1s. Pensar así nos permitirá describir PROBLEMAS DE DECISIÓN y CLASES DE COMPLEJIDAD de manera precisa; así:

## 1.1 Lenguajes y clases de complejidad:

Def:  $\{0,1\}^* = \{ \text{Secuencias de 0's y 1's de longitud finita (arbitraria)} \}$   
 Un lenguaje es un subconjunto (cualquiera)  $L \subseteq \{0,1\}^*$ .

Def: Si  $Q$  es un problema de decisión

$Q : \text{Grupos puros} \longrightarrow \{0,1\}$

(por ejemplo CICLO HAMILTONIANO)

el lenguaje de  $Q$  es

$$L_Q = \{ \langle G \rangle : G \text{ sí contiene un ciclo Hamiltoniano} \} \subseteq \{0,1\}^*$$

↑ Codificaciones de todas las instancias afirmativas.

Sea  $A$  una máquina de Turing <sup>determinista</sup> con estados finales en  $\{0,1\}$ . Escribimos  $A(x)$  para denotar una ejecución de la máquina iniciado con input  $x$ .

Def: Un algoritmo  $A$  acepta a una palabra  $x \in \{0,1\}^*$  ssi  $A(x) = 1$ .

Def: Sea  $L \subseteq \{0,1\}^*$  un lenguaje cualquiera.

(1)  $L$  es aceptado por  $A$  si  $\forall x \in L \ A(x) = 1$

(2)  $L$  es decidido por  $A$  si

(i)  $\forall x \in L \ A(x) = 1$

(ii)  $\forall x \in \{0,1\}^* \setminus L \ A(x) = 0$ .

## 1.2 LA CLASE DE COMPLEJIDAD Polinomial $P$

Lo anterior nos permite definir la clase de complejidad  $P$

$$P = \left\{ \begin{array}{l} \text{lenguajes } L \subseteq \{0,1\}^* : \\ \text{existe un entero } k_L \text{ y un algoritmo } A_L : \\ \begin{array}{l} \text{(i) } A_L \text{ decide a } L \\ \text{(ii) } \left\{ \begin{array}{l} \text{el tiempo de ejecución de } A_L \\ \text{en una palabra } x \in \{0,1\}^* \text{ de} \\ \text{longitud } n \text{ es } O(n^{k_L}) \end{array} \right\} \end{array} \end{array} \right.$$

← decidible en tiempo polinomial

Ejemplo: Sea  $G$  un grafo no orientado con costos enteros en las aristas  $(\langle G \rangle, \langle c \rangle, m) \rightarrow$  Existe un spanning tree de costo  $\leq m? \in P$  **HAMILTONIAN-CYCLE  $\in P$ ???**

Teorema I también puede definirse como la clase de lenguajes aceptados en tiempo polinomial

Dem: Dado un lenguaje  $L$  aceptado por  $A_L$  en tiempo  $\leq K n^{k_L}$  para todo input  $x \in L$  de longitud  $n$  crearemos un nuevo algoritmo  $A'_L$  que decida. Dado  $y \in \{0,1\}^n$  con longitud  $n$  ejecutamos  $K n^{k_L}$  pasos de  $A_L$ . Si  $A_L$  acepta  $\Rightarrow A'_L(y) := 1$  de lo contrario  $A'_L(y) := 0$ . El algo  $A'_L$  decide a  $L$ .