① Complejidad computacional.

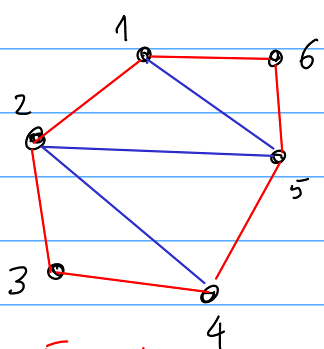
Resumen. La teoría de complejidad computacional se ocupa principalmente de PROBLEMAS DE DECISIÓN, es decir preguntas con respuesta SÍ ó NO.

1 0

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

Ejemplo:

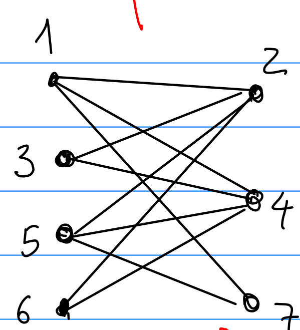
Def: Un camino hamiltoniano en un grafo G es un ciclo simple (es decir una sucesión $v_1, v_2, \ldots, v_k \in V(G)$ con $(v_i, v_{i+1}) \in E(G)$ con $v_i \neq v_j$ $i \neq j$ que forma un ciclo $(v_k, v_1) \in E(G)$) que contiene a todos los vértices de G



Ejemplo

1, 2, 3, 4, 5, 6

es un ciclo hamiltoniano



Ejemplo: NO contiene ningún ciclo Hamiltoniano

PROBLEMA: Contiene $\boxed{G}$ a un ciclo Hamiltoniano?

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

Respuesta es SÍ ó NO (para cada grafo G)

cuánto tiempo requiere 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:

$\boxed{A \text{ Algoritmo:}}$ Produce un entero x en representación UNARIA así:
$6 \longmapsto 111111$

INPUT EN UNARIO
111111
longitud n

$\longrightarrow \boxed{A} \longrightarrow$

OUTPUT
111111
longitud n

tiempo $= O(n)$

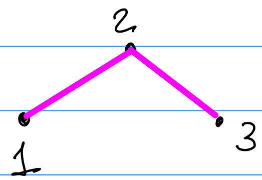
INPUT EN BINARIO

$6 = 110 \longrightarrow \boxed{A} \longrightarrow$ OUTPUT 111111 tiempo $= O(2^{n+1})$

longitud n longitud 2^{n+1}

Así que de ahora en adelante fijaremos una codificación.

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



$\longrightarrow$ Codificado como

$[\{1,2\}, \{2,3\}] \mapsto [\{1, 10\}, \{10, 11\}]$

binario

"0010....."

BUSCAR what is ASCII en internet

binario de cierta longitud.

determinada por $n + m$

Usando la codificación nuestra pregunta se vuelve bastante precisa

[CICLO - HAMILTONIANO]

$\boxed{\text{Contiene } G \text{ a un ciclo Hamiltoniano?}} \longrightarrow \begin{cases} 1 = \text{SI} \\ 0 = \text{NO} \end{cases}$

$\langle G \rangle$ 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"

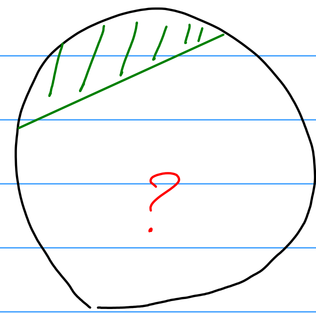
Aquellos que admiten un algoritmo que corre en tiempo polinomial

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

$\boxed{\text{Clase de complejidad } P}$

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 nuestros problemas podemos pensar que los inputs codificados son sólamente 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: $\{0,1\}^* = \{$ Sucesiones 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

$$Q : \text{Grafos finitos} \longrightarrow \{0,1\}$$

(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 determinista con estados finales en $\{0,1\}$. Escribimos $A(x)$ para denotar una ejecución de la máquina iniciando 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) $\forall x \in \{0,1\}^* \setminus L$ $A(x) = 0$.

Lo anterior nos permite definir la clase de complejidad $\mathcal{P}$

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

decidible en tiempo polinomial

Ejemplo: Sea G un grafo no orientado con costos enteros en las aristas $(\langle G \rangle, \langle c \rangle, m)$ ⇝ Existe un spanning tree de costo $\leq m$? $\in \mathcal{P}$

HAMILTONIAN-CYCLE $\in \mathcal{P}$??!

Teorema P 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 $y \in \{0,1\}^*$ 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.