

COMPLEJIDAD COMPUTACIONAL

Capítulo 12 (12.5 y 12.6)

Br. Alvaro Hernández Orence

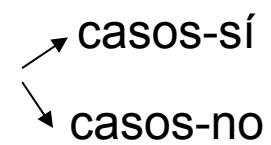
Contenido

- **12.5: Introducción a la NP-Complejidad**
 - 12.5.1 Clases P y NP
 - 12.5.2 Reducciones Polinómicas
 - 12.5.3 Problemas NP-Completos
 - 12.5.4 Algunos problemas NP-Completos
 - 12.5.5 Problemas NP-Difíciles
 - 12.5.6 Algoritmos no Deterministas
- **12.6: Clases de Complejidad**

12.5 Tratabilidad y NP-Complejidad

- Tipos de problemas: tratables, intratables (2 tipos), no resolubles en tiempo pol. ¿tratables?
- Teoría de la NP-Complejidad:
 - Problemas sin algoritmo eficiente, de dificultad intrínseca aún no demostrada. No se sabe si son fáciles o difíciles de resolver, pero son de complejidad parecida.
 - Se cree que algoritmos eficientes para ellos no existen. Lo que sí es eficiente es la validación de una supuesta solución.

12.5.1 P y NP

- ¿Qué es un algoritmo eficiente? $O(n \lg n)$, $O(n^2)$?
- Algoritmo eficiente: $\exists p(n) \mid$ él puede resolver cualquier caso de tamaño n en tiempo $O(p(n))$
 \Rightarrow Algoritmo de tiempo polinómico.
- ¿Qué es un problema de decisión?
 - La respuesta es “sí” o “no” \Rightarrow entre X casos 
- **Definición:** **P** es la clase de problemas de decisión resolubles mediante un algoritmo (determinista) de tiempo polinómico.

12.5.1 P y NP

- *Definición:* **NP** es la clase de problemas de decisión que tienen un sistema de pruebas eficiente, es decir, todo caso-sí debe poseer al menos un certificado “sucinto”, cuya validez pueda ser verificada rápidamente.

~~● **NP** = **No** Polinómico~~



$$P \subseteq NP$$

- **NP** = **N**on-deterministic **P**olynomial: tiempo polinómico no determinista.

12.5.1 P y NP

- ¿P = NP?
 - Para probar esto se tendría que encontrar un algoritmo polinómico \forall problema en NP
- ¿P \neq NP?
 - Para probarlo se tendría que encontrar un problema en NP que no esté en P
- Se conjetura P \neq NP. No ha sido confirmado ni rechazado.

12.5.2 Reducciones Polinómicas

- *Definición:* Sean A y B dos problemas. Se dice que A es **polinómicamente Turing reducible** a B si existe un algoritmo para resolver A en un tiempo que sería polinómico si se pudieran resolver casos arbitrarios de B con costo unitario. Esto se denota $A \leq_T^p B$. Cuando $A \leq_T^p B$ y $B \leq_T^p A$ son ciertos ambos, se dice que A y B son **polinómicamente Turing equivalentes**, esto es $B \equiv_T^p A$
- *Teorema:* $\text{HAM} \equiv_T^p \text{HAMD}$

12.5.2 Reducciones Polinómicas

- *Demostración (parte 1):*

- $\text{HAMD} \leq_T^P \text{HAM}$

5/6/05		
HamD(Grafo:G): Lógico		
pre: { $\exists G$ }		pos: { }
1	s = Ham(G)	-s:Lista. Sucesión de nodos.
2	Si (s define un ciclo hamilt. en G) ent. regresar verdadero sino regresar falso	Posible ciclo hamiltoniano.

$O(|A|)$, A: aristas del Grafo G

12.5.2 Reducciones Polinómicas

- *Demostración (parte 2):*

- $\text{HAM} \leq_T^P \text{HAMD}$

5/6/05

Ham(Grafo:G<N,A>): Lista

pre:{ $\exists G$ }

pos:{}

- | | |
|---|---|
| 1 | Si (HamD(G)=falso) entonces
regresar \emptyset
Fin_si |
| 2 | [Si (HamD(<N,A-{e}>)) entonces
A = A - {e}] e \in A |
| 3 | s = sucesión de nodos obtenida siguiendo
el único ciclo que queda en G |
| 4 | regresar s |

-e:Arista.
-s:Lista. Sucesión de nodos del ciclo.

$O(|A|)$

12.5.2 Reducciones Polinómicas

- *Teorema:* Se tienen dos problemas A y B . Si $A \leq_T^p B$, y si B se puede resolver en tiempo polinómico, entonces A también.
- *Teorema:* $TSP \equiv_T^p TSPD$
- *Teorema:* $HAMD \leq_T^p TSPD$

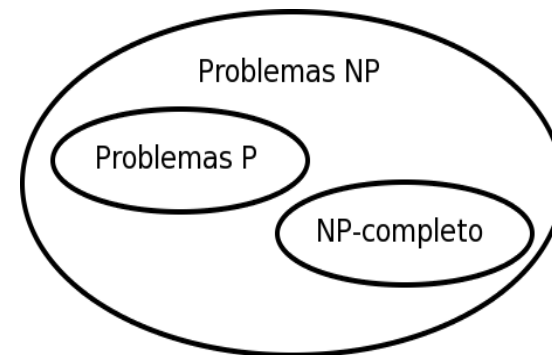
12.5.3 Problemas NP-Completo

- Problemas en NP de extrema complejidad. Son los “peores posibles” en NP.
- Son todos “iguales”, en el sentido que están relacionados computacionalmente
- Nadie ha encontrado un algoritmo polinómico para ninguno de ellos. Además, nadie ha probado que dicho algoritmo no exista.
- Si este algoritmo se descubriese, sería aplicable a todos los problemas NP \Rightarrow **NP**

12.5.3 Problemas NP-Completos

- **Definición:** Un problema de decisión X es NP-completo si:

- $X \in \text{NP}$
- $Y \leq_T^p X, \forall$ problema $Y \in \text{NP}$



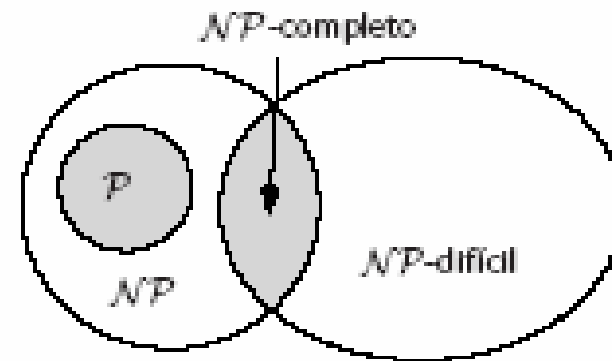
- **Teorema:** Sea X un problema NP-completo. Si existe un problema de decisión $Z \in \text{NP} \mid X \leq_T^p Z$, entonces Z también es NP-completo.

12.5.4 Ejemplos de NP-Complejidad

- *Teorema (Cook):* SAT-CNF es NP-completo.
- SAT
- 3COL
- TSP
- HAM
- Cliqué
- Mochila
- Ciclos eulerianos
- Buscaminas, Tetris
- SAT-3-CNF
- COLD
- TSPD
- HAMD
- Factorización
- Cobertura de vértices
- Suma de subconjuntos
- Envases

12.5.5 Problemas NP-Difíciles

- *Definición:* Un problema de decisión X es NP-difícil (hard, duro, @rr3c#Ø) si:
 - $Y \leq_T^P X$, $Y \in \text{NP-Completo}$



- Problemas al menos tan difíciles como un problema de NP. Un algoritmo eficiente para uno de ellos lo resolvería todo.

12.5.5 Problemas NP-Difíciles

- Estudio de la **NP-Dificultad** en vez de la **NP-Complejidad**:
 - Un problema NP-difícil no tiene por qué ser un problema de decisión, es decir, estar en NP.
Ej: COLO
COLC
 - Existen problemas de decisión de los cuales se sabe que son NP-difíciles, pero además se cree que no están en NP, y por lo tanto no son NP-completos
Ej: COLE

12.5.6 Algoritmos no Deterministas

- *Pre-Definición:* **NP** es la clase de problemas de decisión que se pueden resolver mediante un **algoritmo No determinista** en un tiempo **P** Polinómico.
- Algoritmo no determinista:
 - Finaliza su ejecución con: ***admitir*** o ***rechazar***
 - Puede utilizar: ***seleccionar n entre i y j***

- **Importante:**

Algoritmo no determinista \neq Algoritmo probabilista

\therefore seleccionar n entre i y j \neq $n \leftarrow \text{uniforme}(i..j)$

12.5.6 Algoritmos no Deterministas

5/6/05

HamND(Grafo:G<N,A>)

pre:{|N|>2}

pos:{|}

1	n, S, x = N , ∅, n ₁	-n:Entero. N° de nodos del grafo
2	[seleccionar i entre 2 y n Si (n _i ∈ S ∨ {x,n _i } ∉ A) entonces rechazar Fin_si x = n _i S = S ∪ {x}] k = 2,n	-S:Conjunto. Guarda los nodos del posible ciclo
3	Si ({x,n _i } ∈ A) entonces aceptar sino rechazar Fin_si	-x:Nodo. Posible nodo del ciclo -i,k:Entero. Indice del nodo escogido, contador

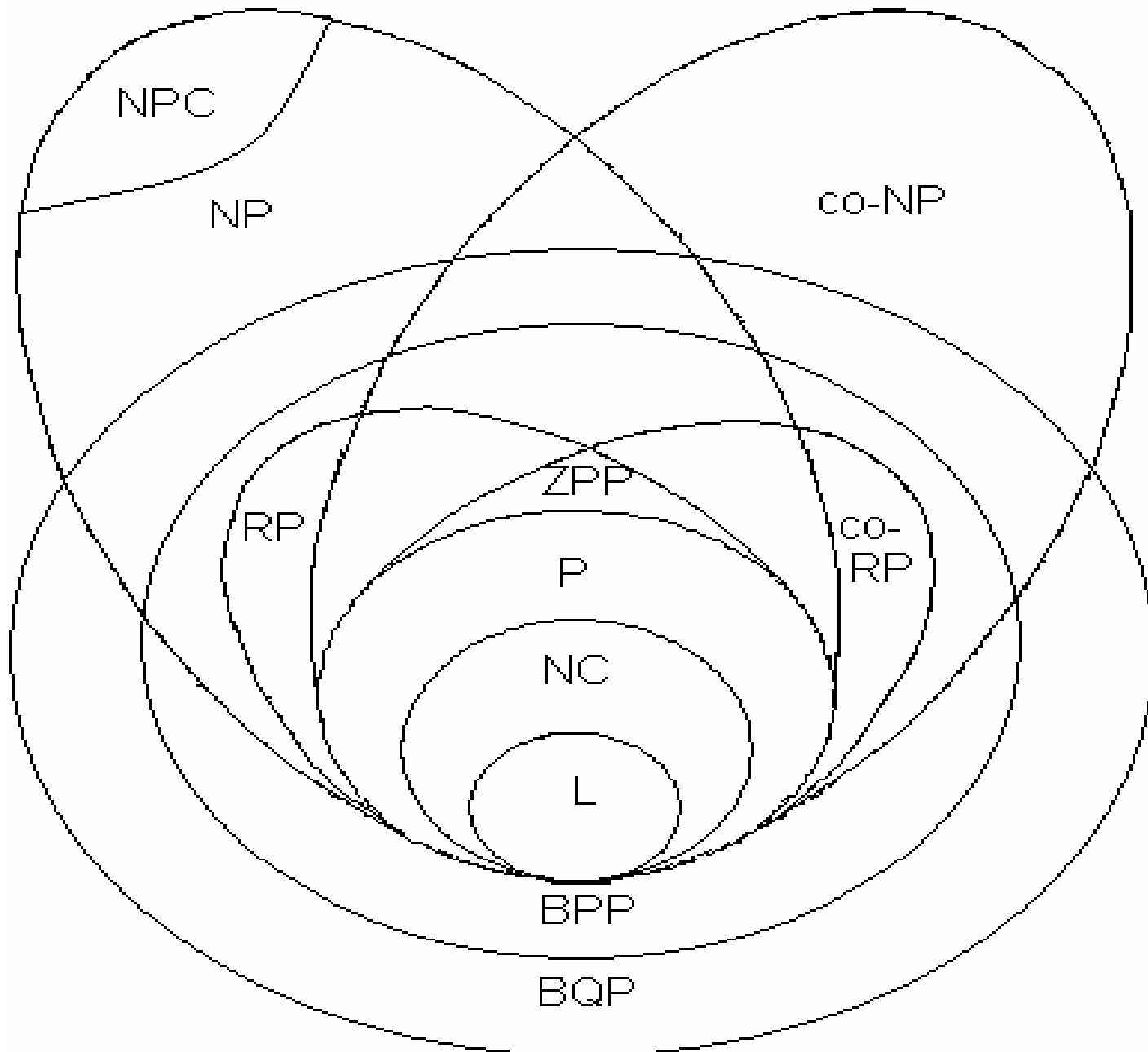
$O(|N|)$, N: nodos del Grafo G

12.6 Clases de Complejidad

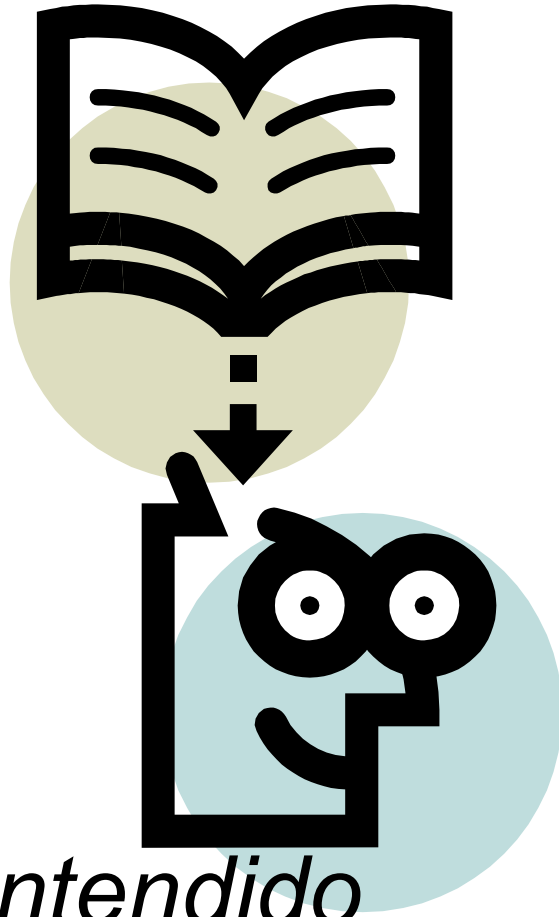
- *Clase de Complejidad.*

Conjunto de todos los problemas de algún tipo que se pueden resolver empleando un modelo dado de cómputo sin sobrepasar alguna cantidad dada de recursos.

Ej: P y NP



GRACIAS POR SU ATENCIÓN



*¿Entendido
todo?*



¿Preguntas?