I Análisis de algoritmos

ALGORÍTMICA Y COMPLEJIDAD

Camilo Palazuelos Calderón

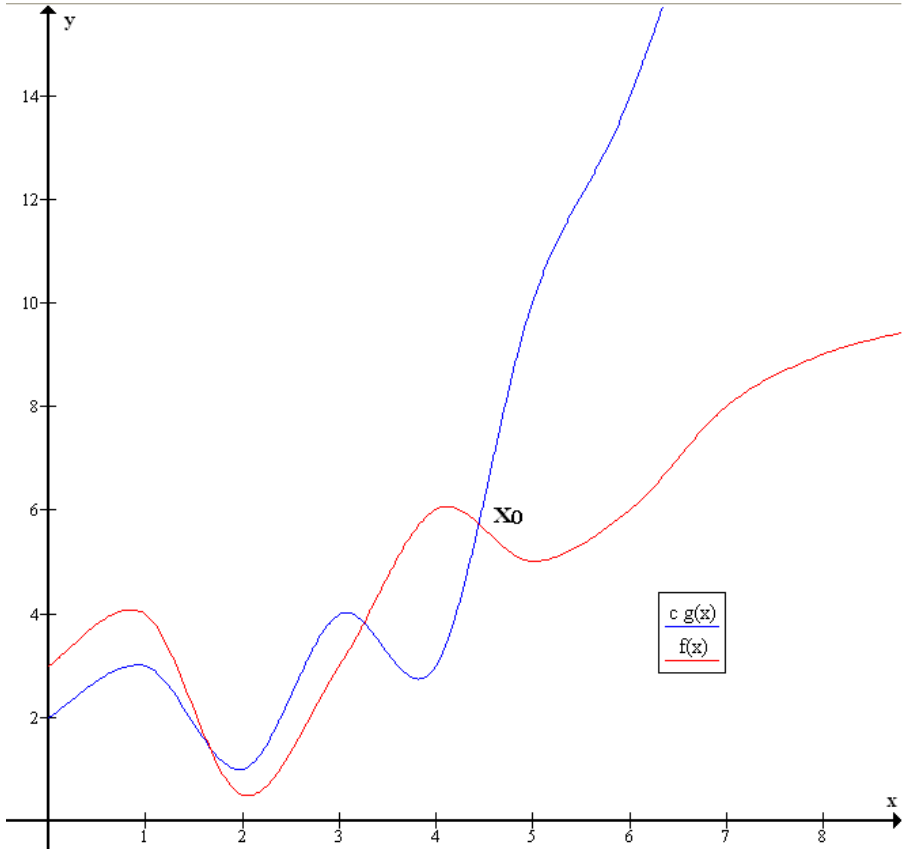
Curso 2023-2024

Notación asintótica, I

- $T(n)$ es $O(f(n))$ si existen constantes $c, n_0 > 0$ tales que $T(n) \leq c\, f(n)$ para todo $n \geq n_0$
  - $O(f(n))$ constituye una *cota superior asintótica* al ritmo de crecimiento de $T(n)$
- $T(n)$ es $\Omega(f(n))$ si existen constantes $c, n_0 > 0$ tales que $T(n) \geq c\, f(n)$ para todo $n \geq n_0$
  - $\Omega(f(n))$ constituye una *cota inferior asintótica* al ritmo de crecimiento de $T(n)$
- $T(n)$ es $\Theta(f(n))$ si $T(n)$ es tanto $O(f(n))$ como $\Omega(f(n))$
  - $\Theta(f(n))$ constituye una *cota ajustada asintótica* al ritmo de crecimiento de $T(n)$
  - Algunos de los resultados más interesantes que trataremos vienen dados en términos de $\Theta(f(n))$
- c, n y n_0 son enteros positivos y $T(n)$ y $f(n)$ son funciones reales con valores positivos

Notación asintótica, 2

- En la imagen, $f(x)$ es $O(g(x))$, ya que
  - Existe $c > 0$
  - Existe $n_0 = [x_0] = 5$
  - Tales que $f(n) > c\ g(n)$ para todo $n \geq n_0$



Cálculo de la complejidad

- Regla de la suma
  - Si $T_1(n)$ es $O(f_1(n))$ y $T_2(n)$ es $O(f_2(n))$, entonces $T_1(n) + T_2(n)$ es $O(\max\{f_1(n), f_2(n)\})$

- Regla del producto
  - Si $T_1(n)$ es $O(f_1(n))$ y $T_2(n)$ es $O(f_2(n))$, entonces $T_1(n)\ T_2(n)$ es $O(f_1(n)\ f_2(n))$
  - De esta regla, se deriva que $O(c\ f(n)) = O(f(n))$ para c constante distinta de 0

- Las instrucciones elementales son $O(1)$
  - Dependiendo del contexto, habrá que dejar claro cuáles son

Clases de complejidad importantes

- La clase NP
  - Problemas para los que podemos *verificar* si una solución es válida *en tiempo polinómico*
  - Ejemplo: En el problema SUBSETSUM podemos saber en $O(n)$ si el conjunto $\{-14, 6, 8\}$ es un subconjunto de suma 0 del conjunto $\{9, -14, 48, 6, 8, -24, 2\}$

- La clase P
  - Problemas que podemos *resolver en tiempo polinómico*
  - $P \subseteq NP$

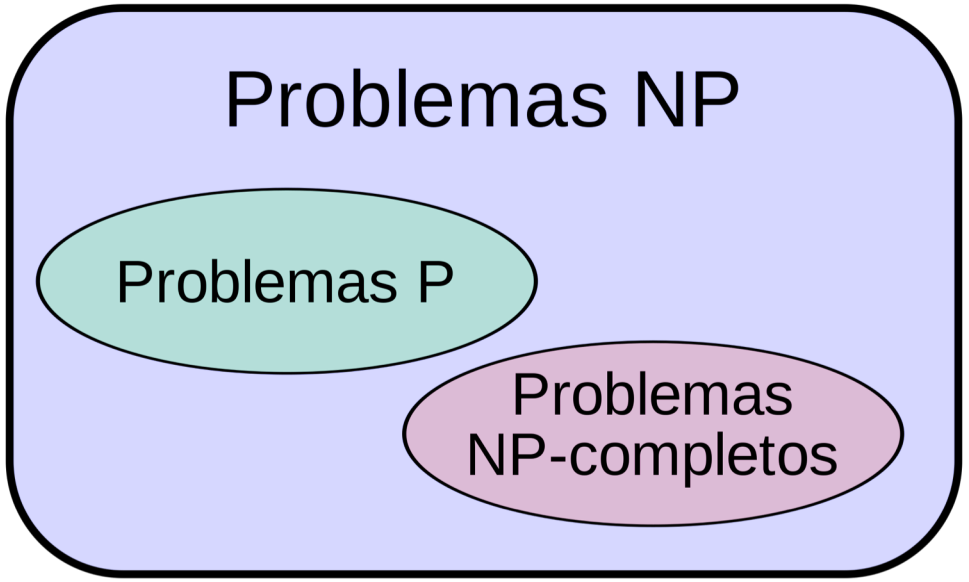
- La clase NP-completo
  - Problemas para los que *parece que no existe* un algoritmo que los resuelva en tiempo polinómico
  - Definiremos formalmente esta clase hacia el final de la asignatura

¿P = NP?

- Todo parece indicar que P ≠ NP
  - Pero continúa siendo una *conjetura*

- ¿Cómo demostrar que P = NP?
  - Resolviendo un problema NP-completo con un algoritmo de *ritmo de crecimiento polinómico*
  - Todos los problemas NP-completos podrían resolverse en tiempo polinómico si P = NP

Así creemos que es el universo en que vivimos:



FUENTE: Wikimedia Commons