Fundamentos de análisis y diseño de algoritmos

Notación de complejidad y crecimiento de funciones

Notación de la complejidad

Notación O

Notación Ω

Notación Θ

Notación o

Notación ω

Terminología de complejidades

Clasificación de problemas

Notación de complejidad

$n + 8 + 5n + 6$ $\qquad$ $n^2 + 5n - 6 + 8n^3$

Hasta ahora hemos calculado la complejidad de los algoritmos directamente. Ejemplo:

Instrucción	Costo
1 i=1	1
2 **while** i<=n	n+1
3 $j \leftarrow 1$	n
4 **while** j<=n	n(n+1)
5 $j \leftarrow j+1$	n^2
6 $i \leftarrow i+1$	n
Total	$n^2+2n+(n+1)^2+1$

¿Es posible expresar de forma simple la complejidad $3n^2+2n+(n+1)^2+1$?

Notación de complejidad

La respuesta es sí, podemos utilizar una notación para describir el comportamiento del algoritmo, analizando cómo n crece.

$n^2 + \cancel{n}$

En la práctica, cobra importancia mirar cómo se comporta el algoritmo con valores muy grandes de n.

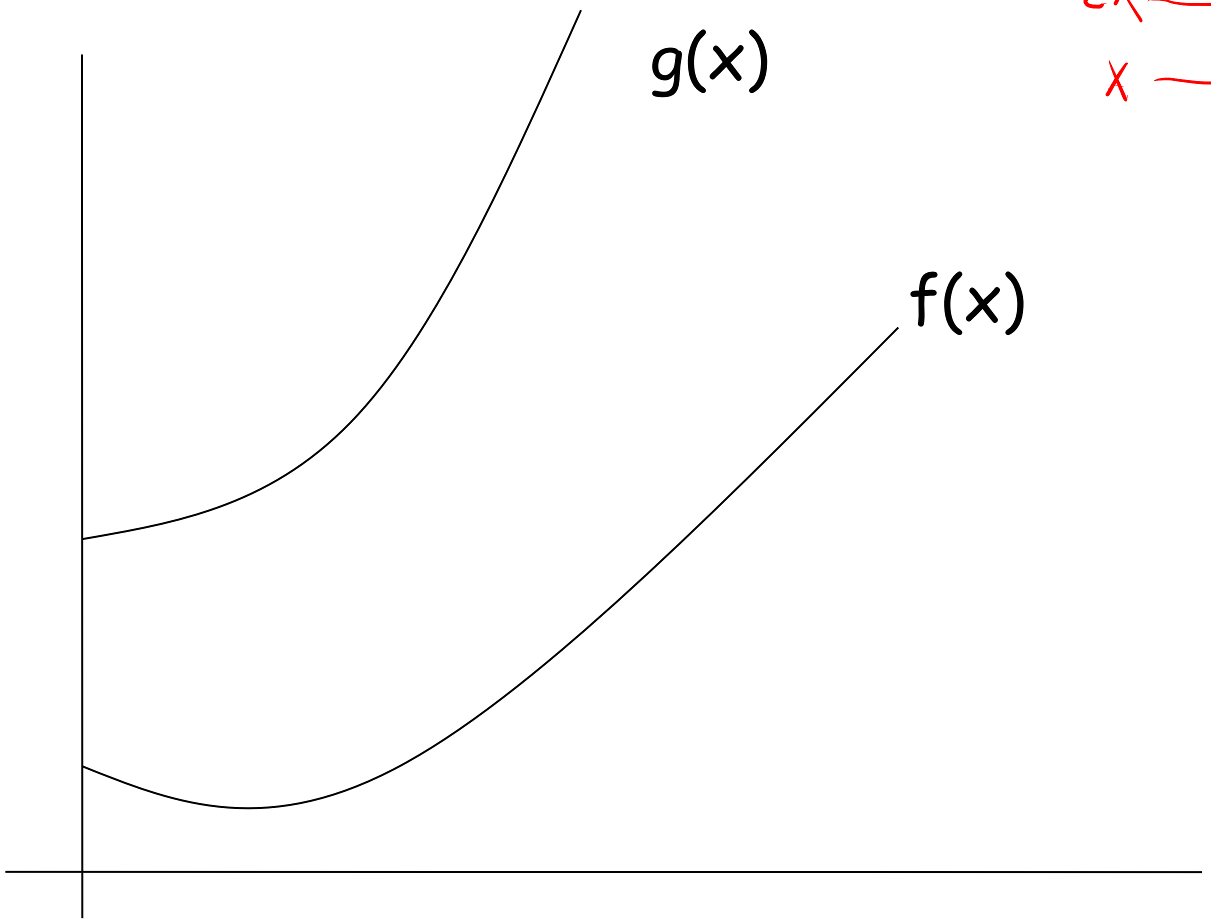
$3n^2+2n+(n+1)^2+1$ se puede decir que es $O(n^2)$ o complejidad cuadrática

Esto permite analizar y comparar más fácilmente algoritmos que solucionan el mismo problema, pero con diferente complejidad computacional.

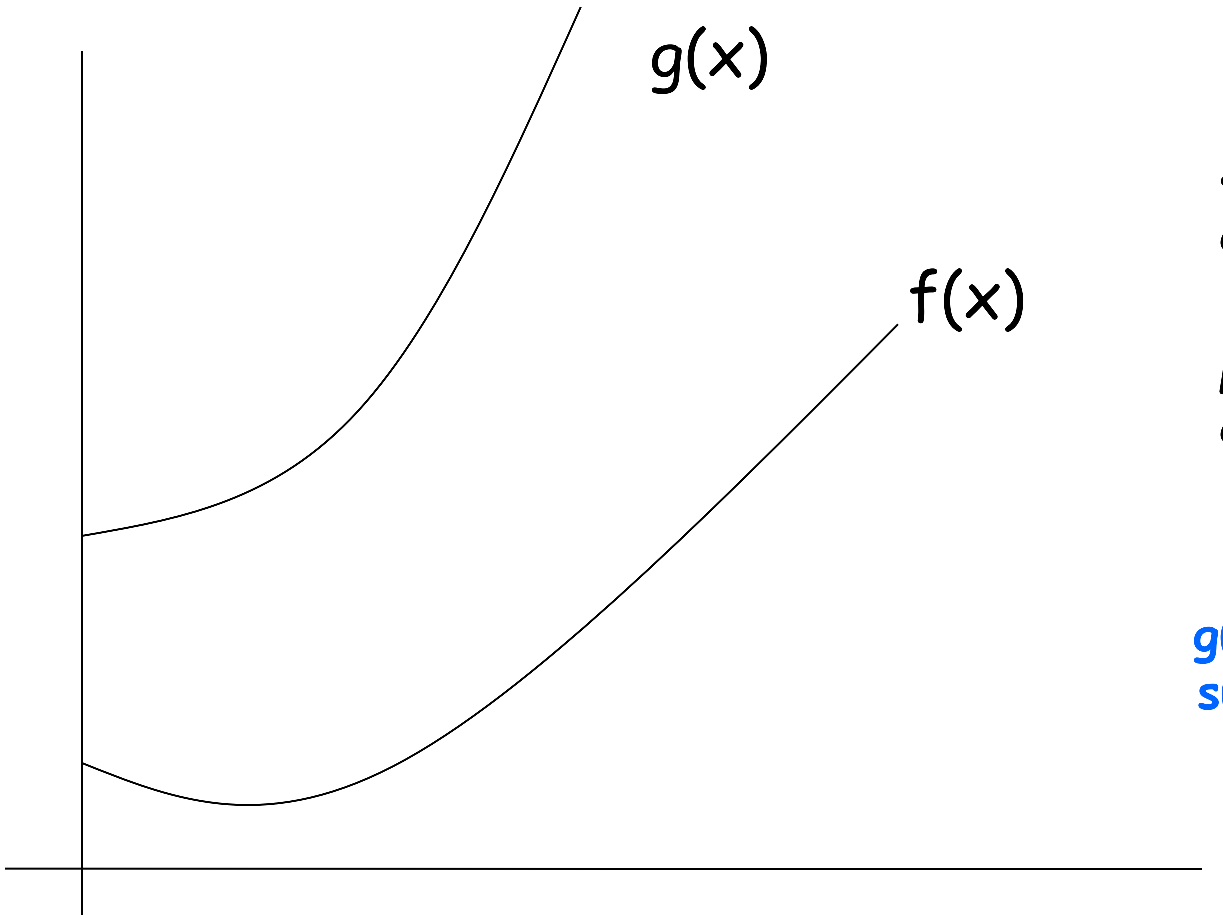
Crecimiento de funciones

$2x \longrightarrow 2$

$x \longrightarrow 1$

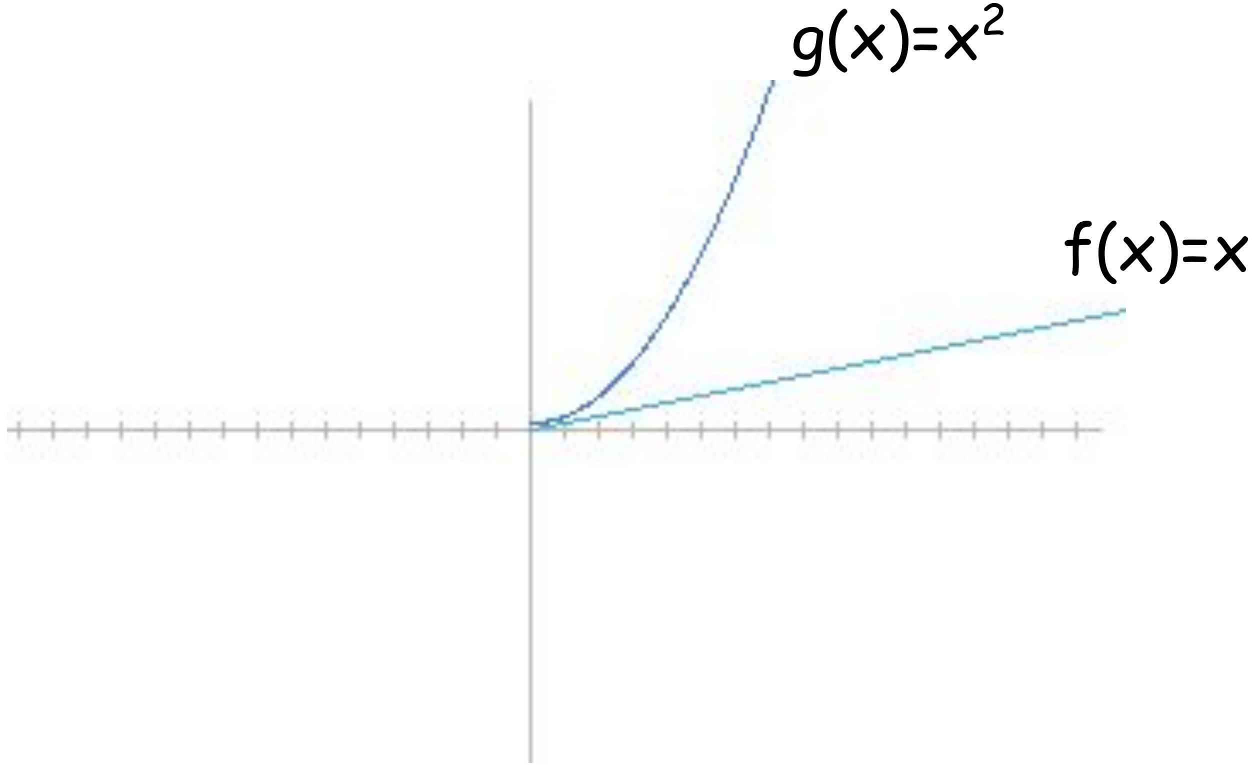


Crecimiento de funciones

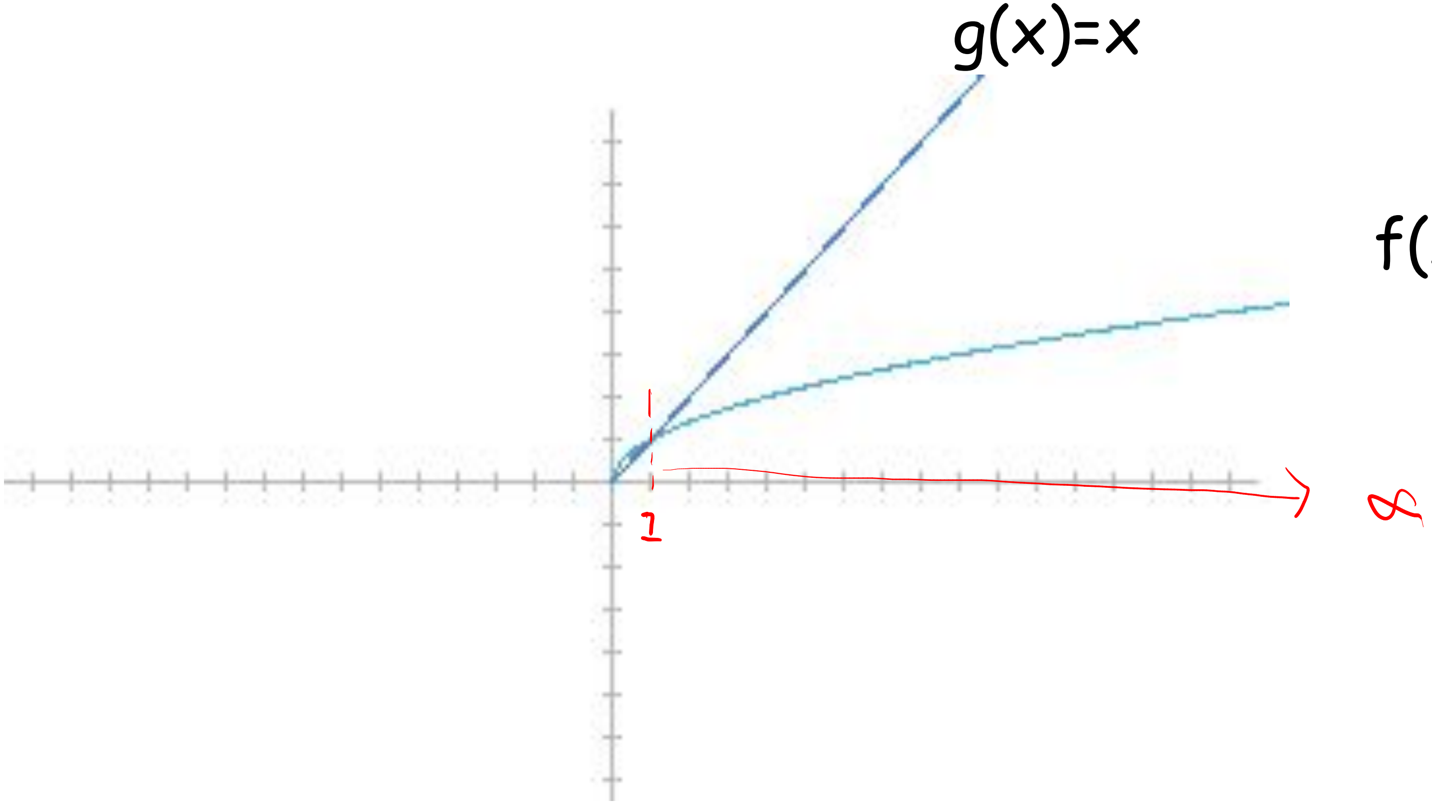


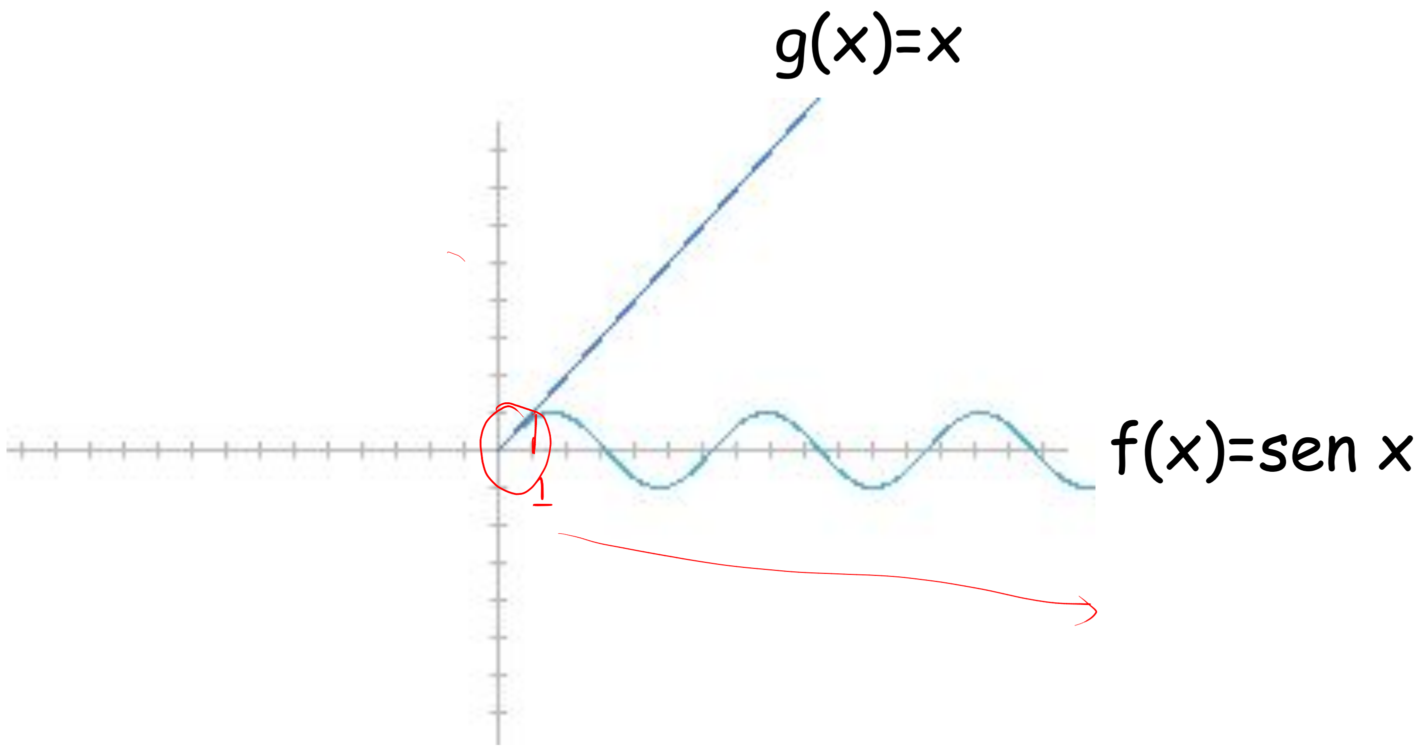
Siempre se cumple que f(x)<g(x) para cualquier valor x

g(x) es una cota superior de f(x)

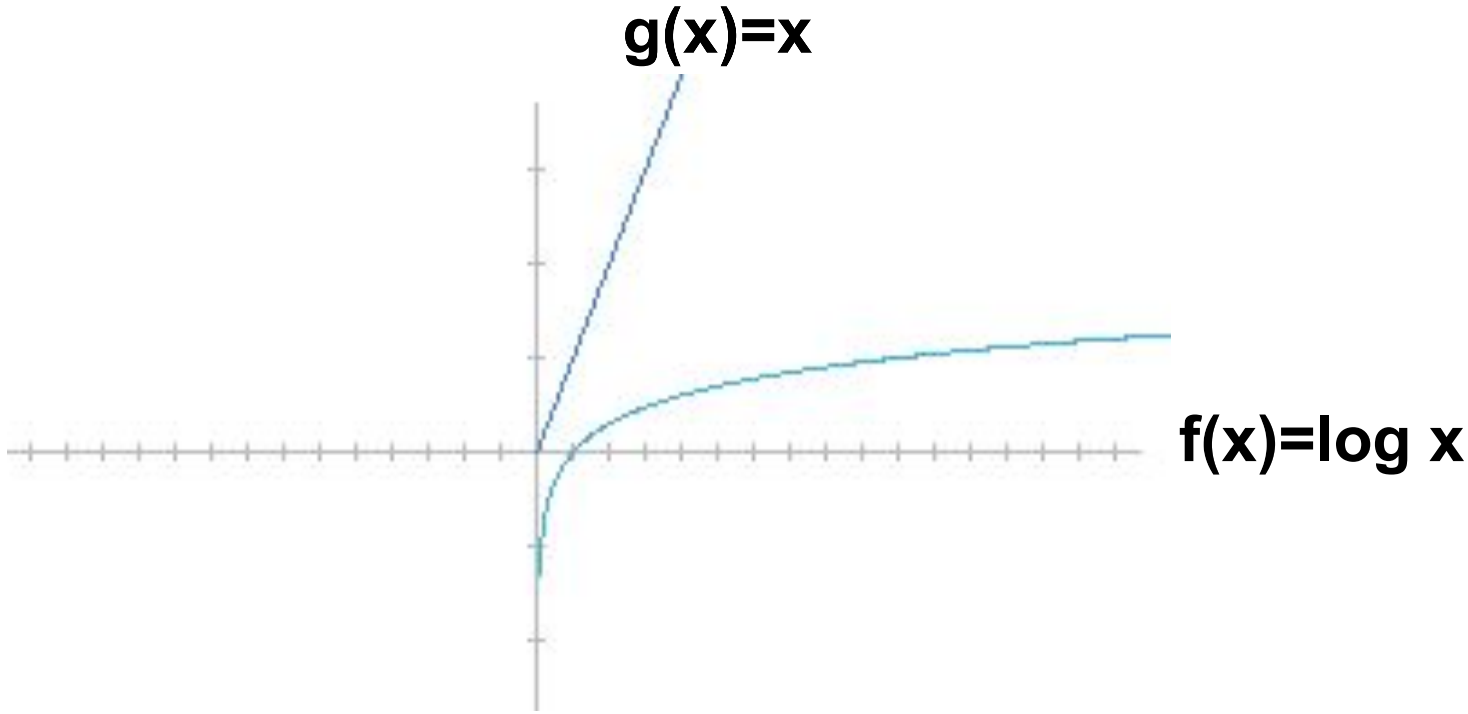

g(x)=x2
f(x)=x

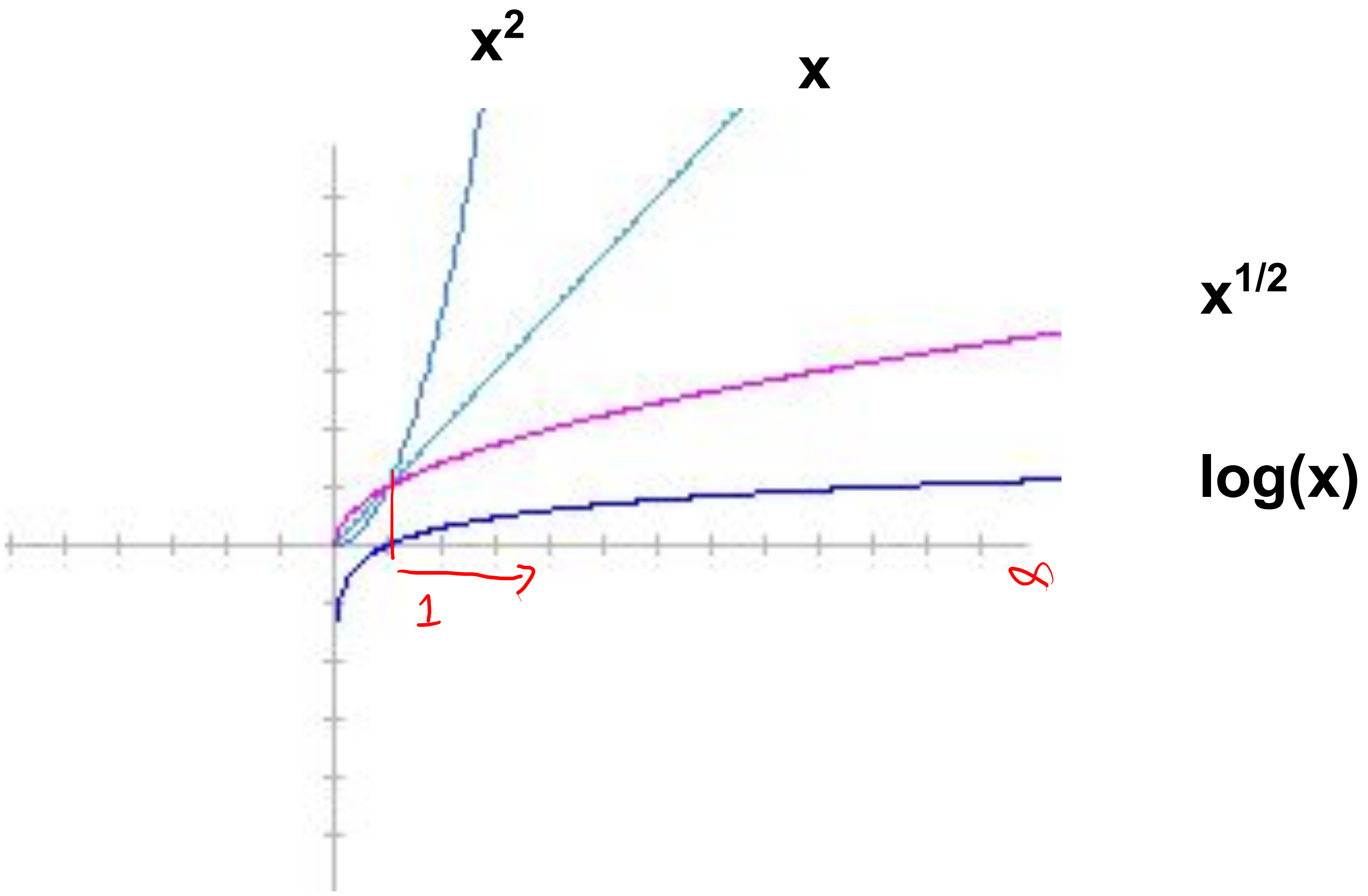
$\sqrt{0.25} = 0.5$

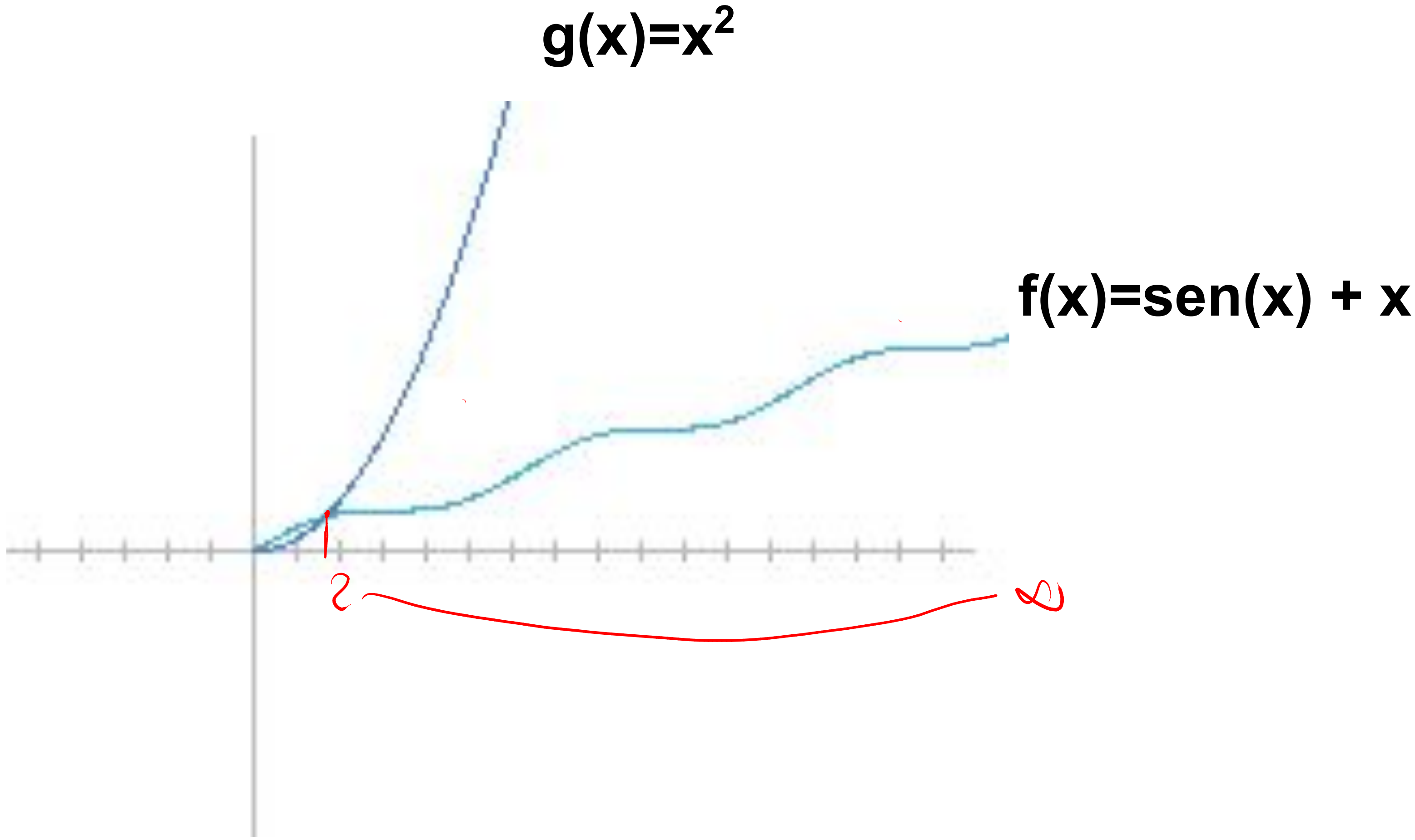




¿Está (sen x) por
debajo de x?

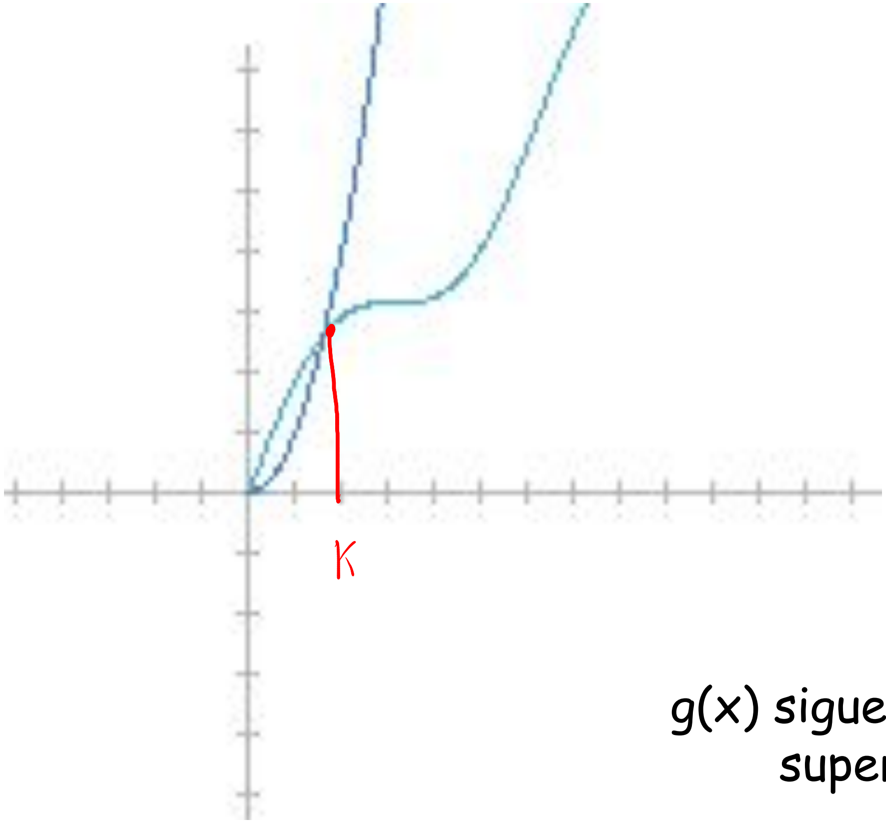

g(x)=x
f(x)=log x


x2
x
x1/2
log(x)
1
∞



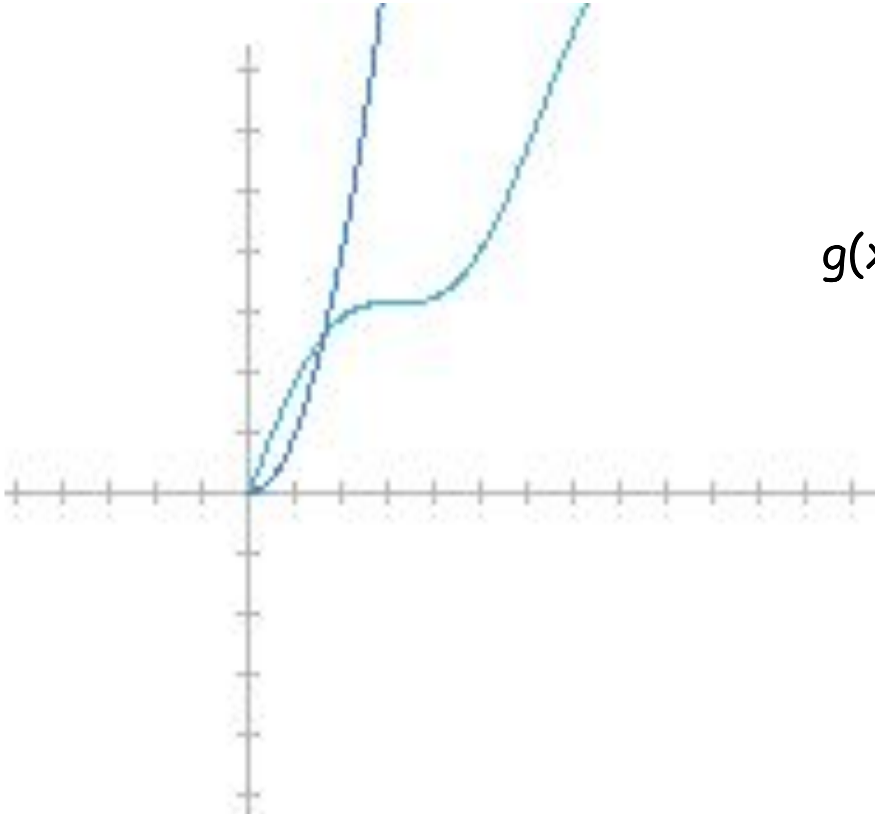
g(x)=x²
f(x)=sen(x) + x
2
∞

$g(x)=x^2$ $f(x)=sen(x) + x$



g(x) sigue siendo una cota superior de f(x)

$g(x)=x^2$ $f(x)=\text{sen}(x) + x$

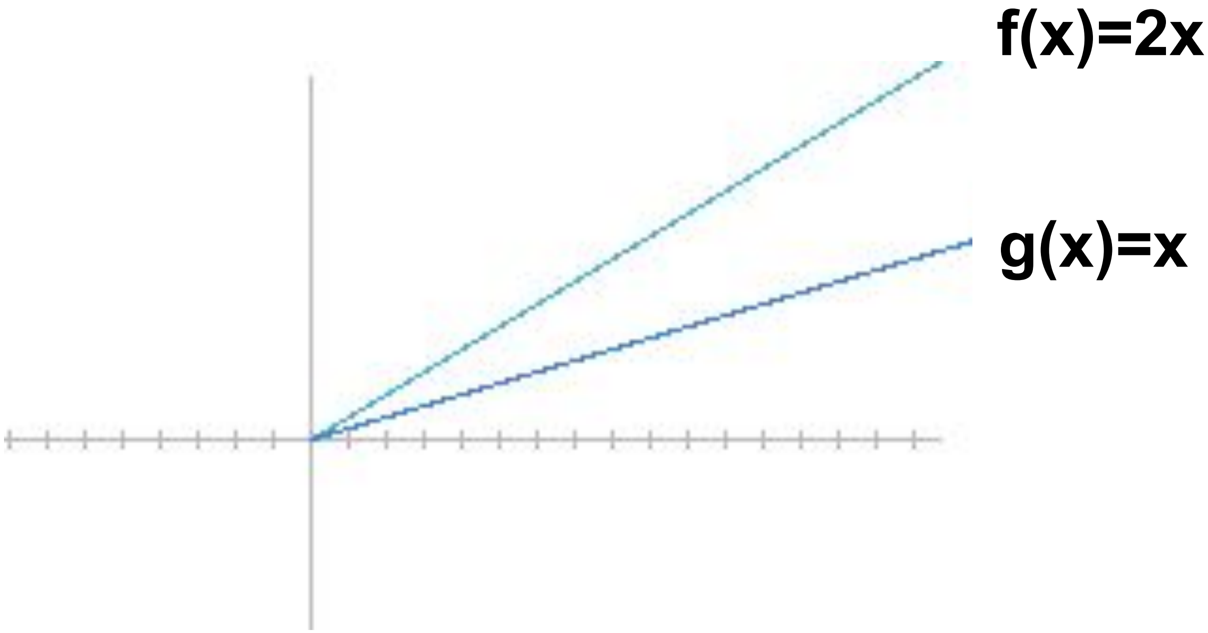


g(x) sigue siendo una cota superior de f(x)

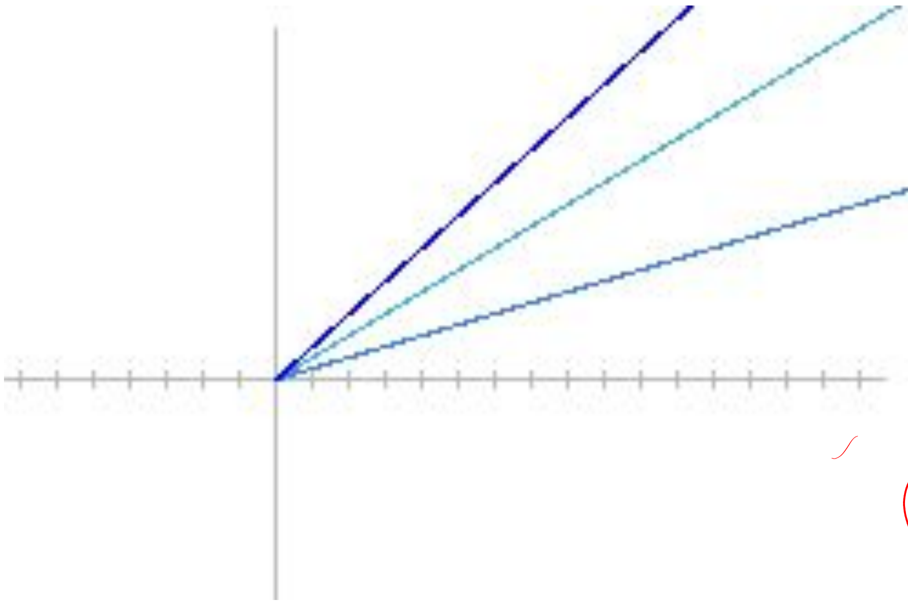
Para que una función sea cota superior de otra, debe serlo para todos los valores de **x≥k**

Pueden existir valores de x<k para los cuales no sea cota superior

Buscar una cota superior para f(x) en términos de g(x)

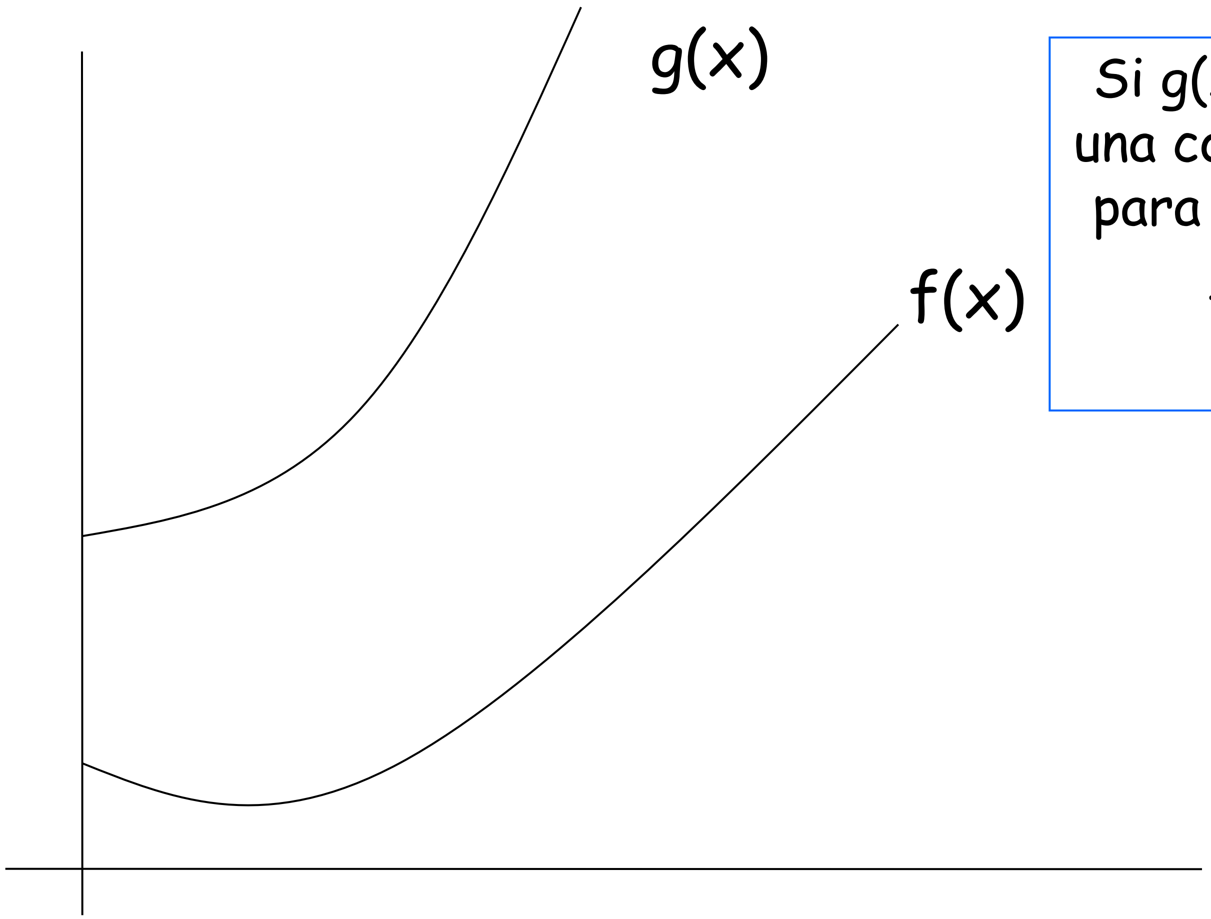


Crecimiento de funciones



c*g(x) es una cota superior para f(x)

Crecimiento de funciones

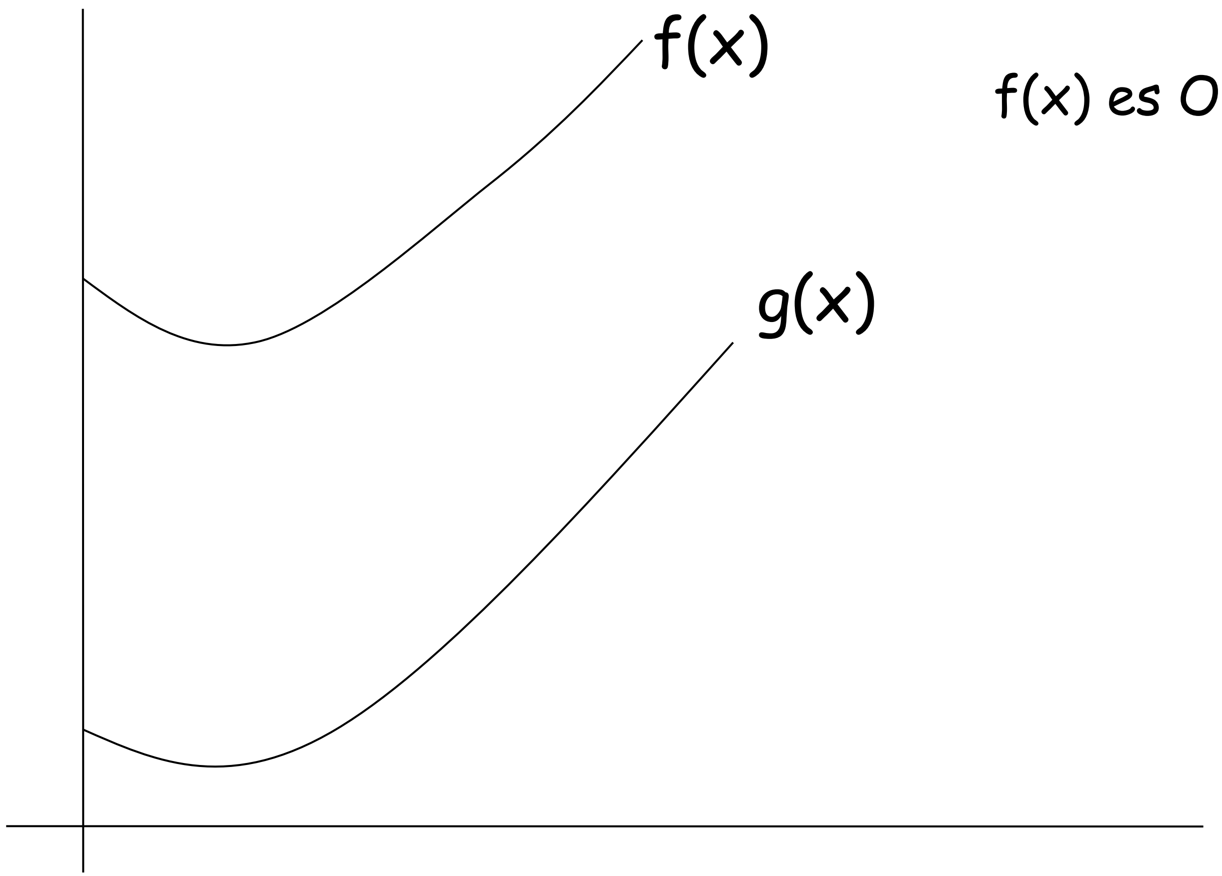


ctr

Si g(x) o algún c*g(x) es una cota superior de f(x), para los x>k, se dice que

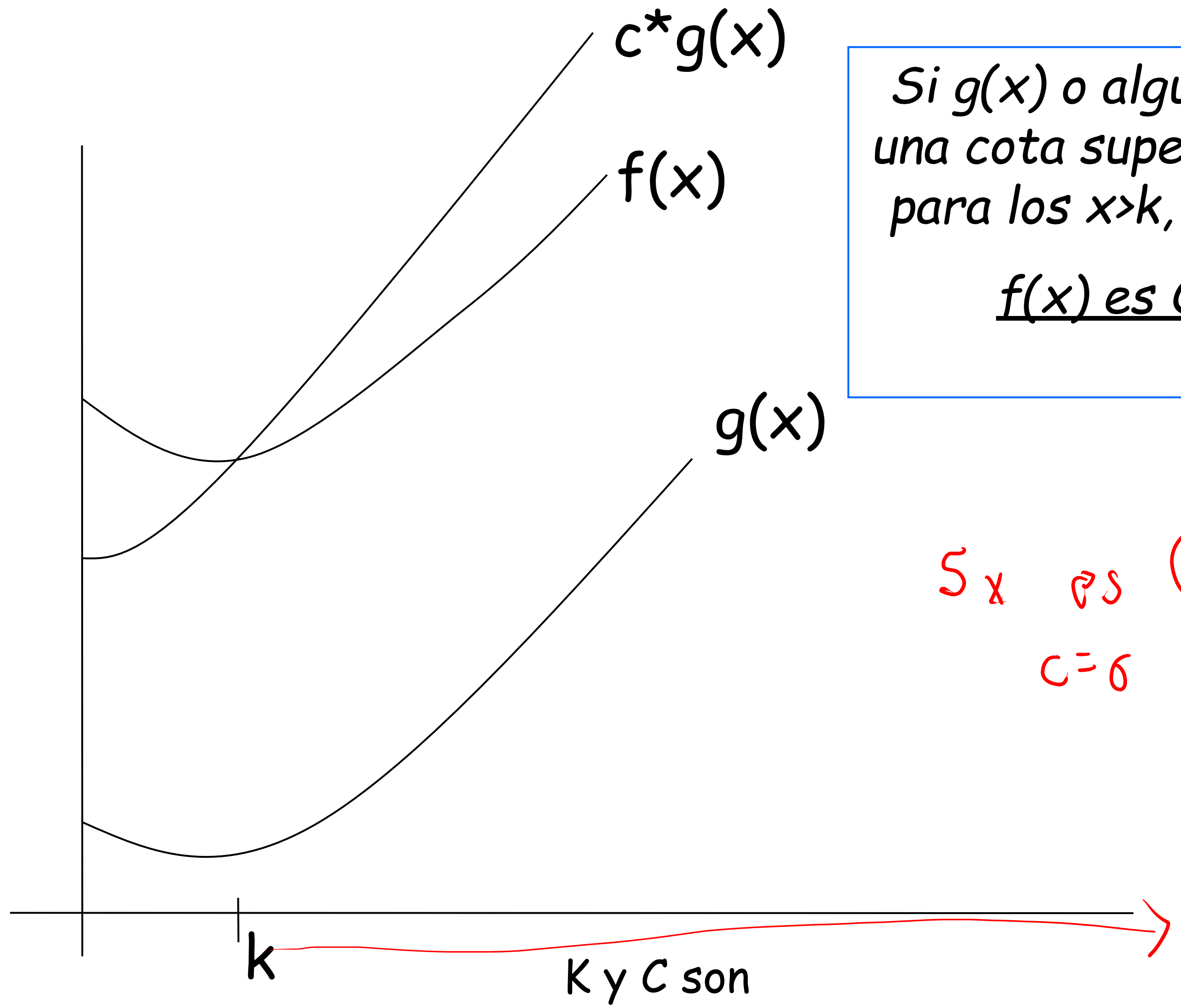
f(x) es O(g(x))

Crecimiento de funciones



$f(x)$ es $O(g(x))$?

Crecimiento de funciones



*Si g(x) o algún c*g(x) es una cota superior de f(x), para los x>k, se dice que*

<u>f(x) es O(g(x))</u>

5x es O(x)

c=6

K y C son constantes

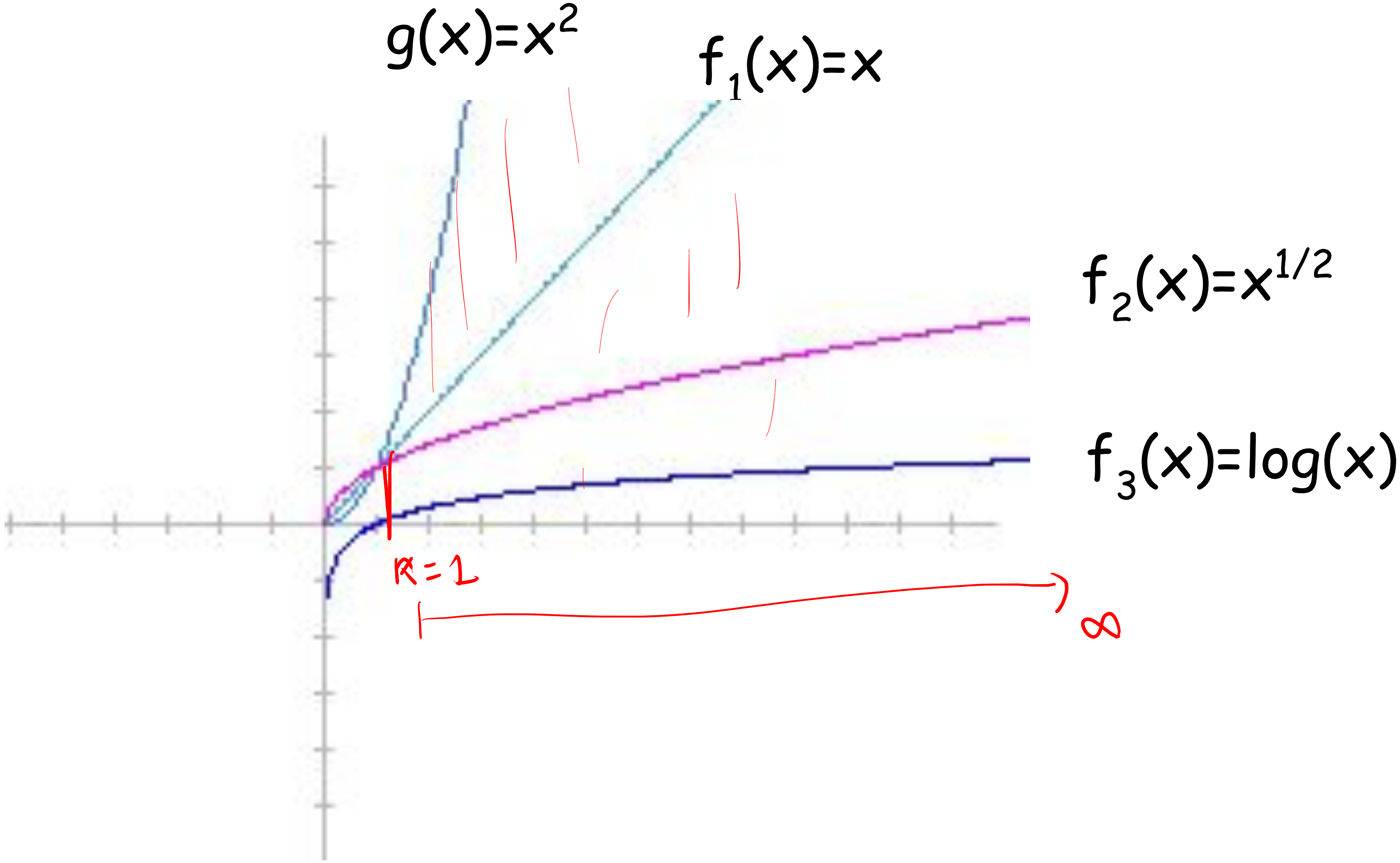
Crecimiento de funciones

Dadas dos funciones f y g, se dice que $f(x)$ es $O(g(x))$ si existen constantes c y k tales que:

$$f(x) \leq c * g(x)$$

se cumple para todos los $x \geq k$

Ψ Donald Knuth(1938-) – Simbolo de Landau



$g(x)$ es una cota superior de
$f_1(x)$, $f_2(x)$, $f_3(x)$

Crecimiento de funciones

Formalmente la notación O representa un conjunto de funciones

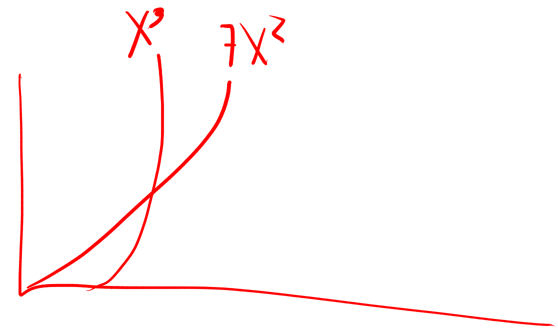
$O(g(x)) = \{ f(x) |$ existen constantes positivas c y k, tales que $0 \leq f(x) \leq c * g(x)$, para todos los $x \geq k\}$

Crecimiento de funciones

$O(x^2) = \{ x, x^{1/2}, \log(x), \ldots \}$

Crecimiento de funciones

Muestre que $7x^2 = O(x^3)$



$$7x^2 \le C \cdot x^3$$

$$7\cancel{x^2} \le \cancel{x^3}$$

$x > 0$

$$7 \le x$$

$x \ge 7$

$\boxed{k = 7}$

$$7x^2 \le C \cdot x^3$$

$$\lim_{x \to \infty} 7x^2 \le C \cdot x^3 = \frac{7x^2}{x^3} \le \frac{C \cdot x^3}{x^3} = \lim_{x \to \infty} \overset{0}{\cancel{\frac{7}{x}}} \le C \qquad \boxed{C \ge 0}$$

Crecimiento de funciones

Muestre que $7x^2 = O(x^3)$

$7x^2 < x^3$ para $x>7$

Por lo tanto se cumple que

$f(x) \leq 1\ g(x)$

para $x>7$

$C=1$, $k=7$

Crecimiento de funciones

Es x^3, $O(7x^2)$

$x \geq K$

$x^3 \leq c \cdot 7x^2$

$x \leq c \cdot 7$

$\boxed{x \leq 7}$ NO SE CUMPLE

$\lim_{x \to \infty} x^3 \leq c \cdot 7x^2 = 1 \leq \frac{c \cdot 7}{\cancel{x}}^{0} = 1 \leq 0 \rightarrow$ FALSO

Crecimiento de funciones

Es x^3, $O(7x^2)$

$x^3 < c\,7x^2$

$x < 7c$

Ya que no se cumple para todos los $x > k$, no es cierto que x^3 sea $O(7x^2)$

Crecimiento de funciones

Ejercicios

Demostrar que:

- x^2+2x+1 es $O(x^2)$
- $\log n$ es $O(n)$, parta sabiendo que $n<2^n$
- $x^2+4x+17$ es $O(x^3)$ pero que x^3 no es $O(x^2+4x+17)$
- $x \log x$ es $O(x^2)$

Explique qué significa que una función sea $\Omega(1)$

Explique qué significa que una función sea $O(1)$

$x^2 + 2x + 1$ es $O(x^2)$

$x^2 + 2x + 1 \leq c\,x^2$

$x^2 + 2x + 1 \leq 2x^2$

$x^2 - 2x - 1 \boxed{\geq} 0$

$x \geq 1 + \sqrt{2}$ ✓

$x^2 - 2x - 1 = 0$

$$\frac{2 \pm \sqrt{4+4}}{2} = \frac{2 \pm \sqrt{8}}{2}$$

$$\frac{\cancel{2} \pm \cancel{2}\sqrt{2}}{\cancel{2}} = 1 \pm \sqrt{2}$$

$1 + \sqrt{2}$ $\quad$ $1 - \sqrt{2}$

$$\lim_{x \to \infty} x^2 + 2x + 1 \leq c x^2 :$$

$$\lim_{x \to \infty} \frac{x^2}{x^2} + \overset{0}{\frac{2x}{x^2}} + \overset{0}{\frac{1}{x^2}} \leq c \frac{x^2}{x^2} = 1 \leq c$$

$\boxed{c \geq 1}$

$\boxed{c > 1}$

$$\log(n) \in O(n)$$

$$\log(n) \leq C \times n$$

$$\log(n) \leq 2n$$

$$2^{\log(n)} \leq 2^{2n}$$

$$n^{\log(2)} \leq 2^{n+n}$$

$$n \leq 2^{n} \times 2^{n}$$

$$n \leq (2^{n})^{2}$$

$$\log(n) \leq C \times n$$

$$\lim_{n \to \infty} \log(n) \leq C \times n$$

$$\lim_{n \to \infty} \cancel{\frac{\log(n)}{n}} \leq \frac{C \times n}{n}$$

$$0 \leq C$$

$$\boxed{C > 0}$$

$x^2 + 4x + 17$ es $O(x^3)$

$x > 3.5$

$x^2 + 4x + 17 \leq Cx^3$

$x^2 + 4x + 17 \leq x^3$

$x^3 - x^2 - 4x - 17 \geq 0$

$\lim_{x \to \infty} x^2 + 4x + 17 \leq Cx^3$

$\lim_{x \to \infty} \frac{x^2}{x^3} + \frac{4x}{x^3} + \frac{17}{x^3} \leq \frac{Cx^3}{x^3} = 0 \leq C$ $\boxed{C > 0}$

x^3 es $O(x^2+4x+17)$

$x^3 \le c(x^2+4x+17)$

$x^3 \le 3x^2+12x+51$

$-x^3+3x^2+12x+51 \ge 0$

$x \ge 6.235$

$$\lim_{x\to\infty} x^3 \le c(x^2+4x+17)$$

$$\lim_{x\to\infty} \frac{x^3}{x^3} \le \overset{0}{\frac{c(x^2+4x+17)}{x^3}}$$

$\boxed{1 \le 0}$ Falso

$x \log(x) \leq C_x x^2$

$x \log(x) \leq 2x^2$

$\boxed{\log(x) \leq 2x}$

$2^{\log(x)} \leq 2^{2x}$

$x^{\log(2)} \leq 2^{x+x}$

$x \leq 2^x 2^x$

$\boxed{x \leq (2^x)^2}$ ✓

$x > 0$

$\lim_{x \to \infty} x\log(x) \leq C_x x^2$

$\lim_{x \to \infty} \frac{x\log(x)}{x^2} \leq \frac{c x^2}{x^2}$

$\lim_{x \to \infty} \cancel{\frac{\log(x)}{x}}^{0} \leq c$

$0 \leq c \quad c \geq 0$

$\boxed{c > 0}$ OK ✓

1) $n^2 + 8n$ es $O(n^2)$

2) $n\log(n)$ es $O(n^2)$

1)

$n^2 + 8n \leq C \times n^2$

$\lim_{n\to\infty} n^2 + 8n \leq C n^2$

$\lim_{n\to\infty} \frac{n^2}{n^2} + \frac{8n}{n^2} \leq C \frac{n^2}{n^2}$

$\lim_{n\to\infty} 1 + \cancel{\frac{8}{n}} \leq C$ 0

$\boxed{C > 1}$

2)

$n\log(n) \leq C n^2$

$\lim_{n\to\infty} n\log(n) \leq C n^2$

$\lim_{n\to\infty} \frac{n\log(n)}{n^2} \leq \frac{C n^2}{n^2}$ 0

$\lim_{n\to\infty} \cancel{\frac{\log(n)}{n}} \leq C$

$\boxed{C > 0}$

Crecimiento de funciones

X

Cota superior asintótica

Cuando se determina que el tiempo de cómputo de un algoritmo es $O(g(x))$ se establece una <u>cota superior</u>

Dentro del análisis para obtener la cota superior, se debe considerar el peor caso, de esta forma se consigue tener una cota que no puede resultar peor

Crecimiento de funciones

Cota superior asintótica

Suponga que para el algoritmo 1 se encontró que $T_1(n)=O(n^2)$ y para el algoritmo 2 que $T_2(n)=O(n\lg n)$, qué representan estos resultados?

¿Qué se puede esperar en cuanto al tiempo de ejecución de los algoritmos?

¿Se puede asegurar que siempre es mejor el algoritmo 2 que el 1?

Crecimiento de funciones

Cota superior asintótica

Es un abuso decir que el tiempo de ejecución del insertion sort es $O(n^2)$, esto se puede asegurar, para el peor caso.

Crecimiento de funciones

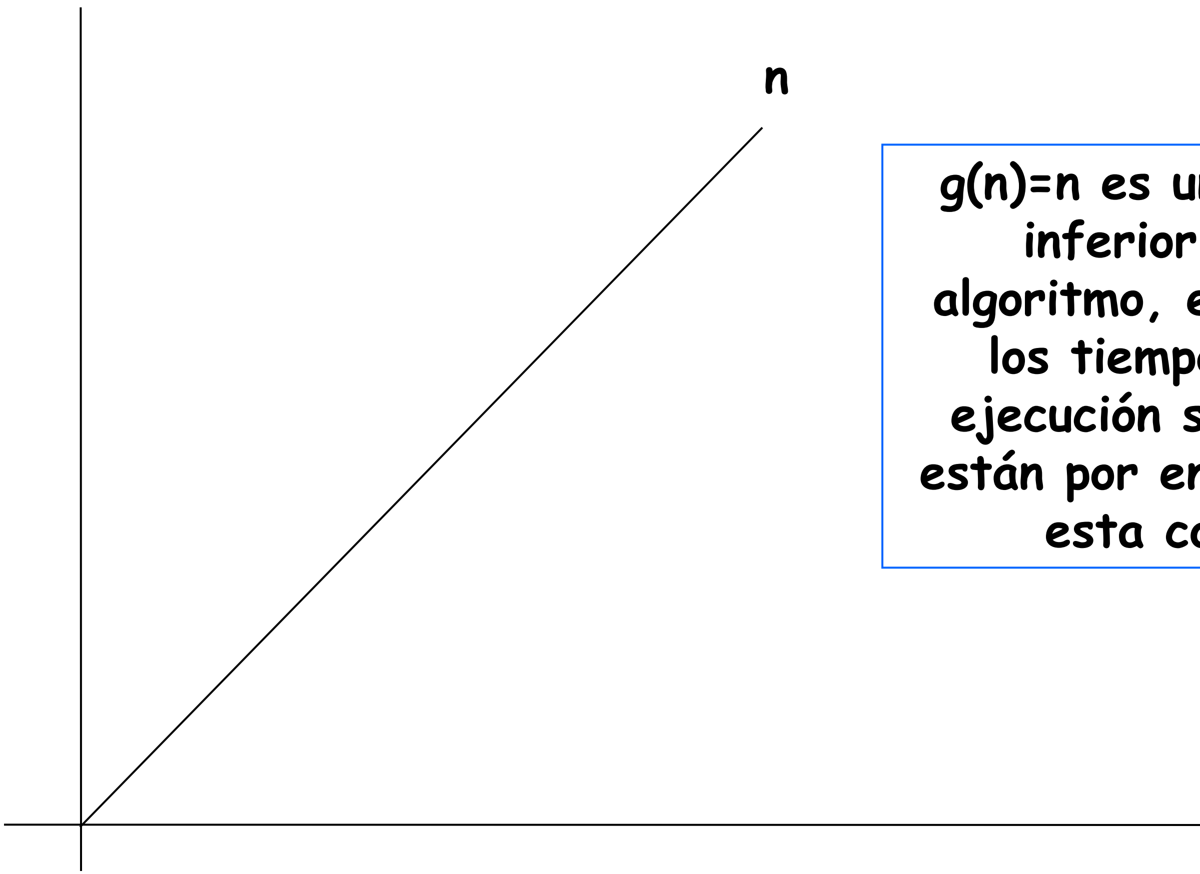
Cota inferior

$\Omega(g(x)) = \{ f(x) |$ existen constantes positivas c y k, tales que $0 \leq c * g(x) \leq f(x)$, para todos los $x \geq k\}$

Crecimiento de funciones

Cota inferior

Suponga que para un algoritmo se encontró que $T_1(n)=\Omega(n)$



g(n)=n es una cota inferior del algoritmo, esto es, los tiempos de ejecución siempre están por encima de esta cota

Crecimiento de funciones

Cota inferior

Suponga que para un algoritmo se encontró que $T_1(n)=\Omega(n^2)$ y para otro que $T_2(n)=\Omega(n\lg n)$, qué representan estos resultados?

¿Qué se puede esperar en cuanto al tiempo de ejecución de los algoritmos?

Crecimiento de funciones

Notación $\Theta(f(n))$

$f(n)=\Theta(g(n))$ sii $f(n)=O(g(n))$ y $f(n)=\Omega(g(n))$

$n^2 + 5n$ es $\Theta(n^2)$ $\left\{ \begin{array}{l} O(n^2) \\ \Omega(n^2) \end{array} \right.$

$n^2 + 5n$ es $O(n^2)$

$$n^2 + 5n \leq C \times n^2$$

$$\lim_{n \to \infty} \frac{n^2}{n^2} + \frac{5n}{n^2} \leq \frac{Cn^2}{n^2}$$

$1 \leq C$

$\boxed{C > 1}$

$C = 2$

$n^2 + 5n$ es $\Omega(n^2)$

$$n^2 + 5n \geq C \times n^2$$

$$\lim_{n \to \infty} \frac{n^2}{n^2} + \frac{5n}{n^2} \leq \frac{Cn^2}{n^2}$$

$C \leq 1$ $\quad C = 0.5$

$n^3 - 5n^2$ is $\Theta(n^3)$

$n^3 - 5n^2$ is $O(n^3)$

$$\lim_{n \to \infty} n^3 - 5n^2 \leq c \times n^3$$

$$\lim_{n \to \infty} \frac{n^3}{n^3} - \frac{5n^2}{n^3} \leq \frac{c \times n^3}{n^3}$$

$$\boxed{c > 1}$$

$n^3 - 5n^2$ is $\Omega(n^3)$

$$\lim_{n \to \infty} n^3 - 5n^2 \geq c \times n^3$$

$$\lim_{n \to \infty} \frac{n^3}{n^3} - \frac{5n^2}{n^3} \geq \frac{c \times n^3}{n^3}$$

$$\boxed{c < 1}$$

1) Dada la función n^2+8n de

a) Indique $O(g_1(n))$ y $O(g_2(n))$ $g_1=n^2$ $g_2=n^4$

b) Indique $\Omega(g_3(n))$ y $\Omega(g_4(n))$ $g_3=n$ $g_4=1$

c) Indique $\Theta(g_5(n))$ $g_5=n^2$

2) Tenemos dos algoritmos T1 y T2

T1: $O(n^3)$ $\Omega(n\log(n))$

T_2 estable tanto su O y Ω son n^2

T2: $\Theta(n^2)$

T_1 varía de acuerdo a la instancia

¿Que puede decir de ambos?

¿Siempre se garantiza que uno sea mejor que el otro? No, donde T_1 sea mejor que T_2 y viceversa

Hay situaciones donde T1 es nlogn(n) que es mejor que n^2, pero hay situaciones donde T1 es n^3 que es peor que n²

Recuerden

$O(f(n))$ cota superior, idea de la peor situación para un algoritmo

$\Omega(f(n))$ cota inferior, idea del mejor comportamiento

$\Theta(f(n))$ cota ajustada, o sea cota superior y cota inferior ← estabilidad

Crecimiento de funciones

Notación o

$o(g(x)) = \{ f(x) |$ existen constantes positivas c y k, tales que $0 \leq f(x) < c * g(x)$, para todos los $x > k \}$

Crecimiento de funciones

Notación o

La cota $2n^2=O(n^2)$ es asintóticamente ajustada pero $2n=O(n^2)$ no.

Se utiliza la notación o para denotar cota superiores que no son asintoticamente ajustadas

$2n=o(n^2)$ pero $2n^2=o(n^2)$ no.

Crecimiento de funciones

Notación ω

$\omega(g(x))$ = { $f(x)$| existen constantes positivas c y k, tales que $0 \leq c * g(x) < f(x)$, para todos los $x \geq k$}

Crecimiento de funciones

Complejidades de los algoritmos

Complejidad	Terminología
$O(1)$	Complejidad constante
$O(\log n)$	Complejidad logarítmica
$O(n)$	Complejidad lineal
$O(n \log n)$	Complejidad n log n
$O(n^b)$	Complejidad polinomial $b > 3$
$O(b^n)$	Complejidad exponencial
$O(n!)$	Complejidad factorial

$O(n^n)$

Crecimiento de funciones

Complejidades de los algoritmos

Un problema que se puede resolver utilizando un algoritmo con complejidad polinómica en el peor caso se llama tratable. De no ser así se llama intratable

Un problema para el cual no existe un programa que lo pueda solucionar se llama irresoluble. De no ser así se llaman resolubles. (Halting problem - Turing)

Ψ Alan Turing(1912-1954) Problema de la decision de Hilbert

Crecimiento de funciones

Complejidades de los algoritmos

Existen problemas que no se pueden resolver, en el peor caso en tiempo polinomial, pero que dada alguna solución, se puede comprobar que es efectivamente una solución. Estos problemas se llaman NP (Polinómico – No determinista)

Existe un conjunto de problemas NP que además, si se llegase a encontrar una solución en tiempo polinómico, daría solución a un conjunto de problemas relacionados. Este tipo de problema se conoce como NP-completos

Crecimiento de funciones

Complejidades de los algoritmos

$(a \wedge b) \vee (c \wedge \bar{a}) \vee (d \vee c)$

El problema de la satisfactibilidad es un problema NP-completo.

Dada una asignación de valores de verdad, se puede verificar en tiempo polinómico si tal asignación satisface, o no, una fórmula proposicional. Sin embargo, no existe un algoritmo que en tiempo polinómico pueda encontrar la asignación de valores de verdad para que una fórmula cualquiera se satisfaga.

Crecimiento de funciones

Complejidades de los algoritmos

Además, si se encontrara un programa que lograra hallar la solución en tiempo polinómico, se podría solucionar un conjunto de problemas relacionados con la lógica proposicional

Referencias

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. Chapter 3, Pages 43-64

Gracias

Próximo tema:

Ecuaciones de recurrencia