Fundamentos de análisis y diseño de algoritmos

Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería de sistemas y computación

Agosto 2017

Notación O

Notación Ω

Notación Θ

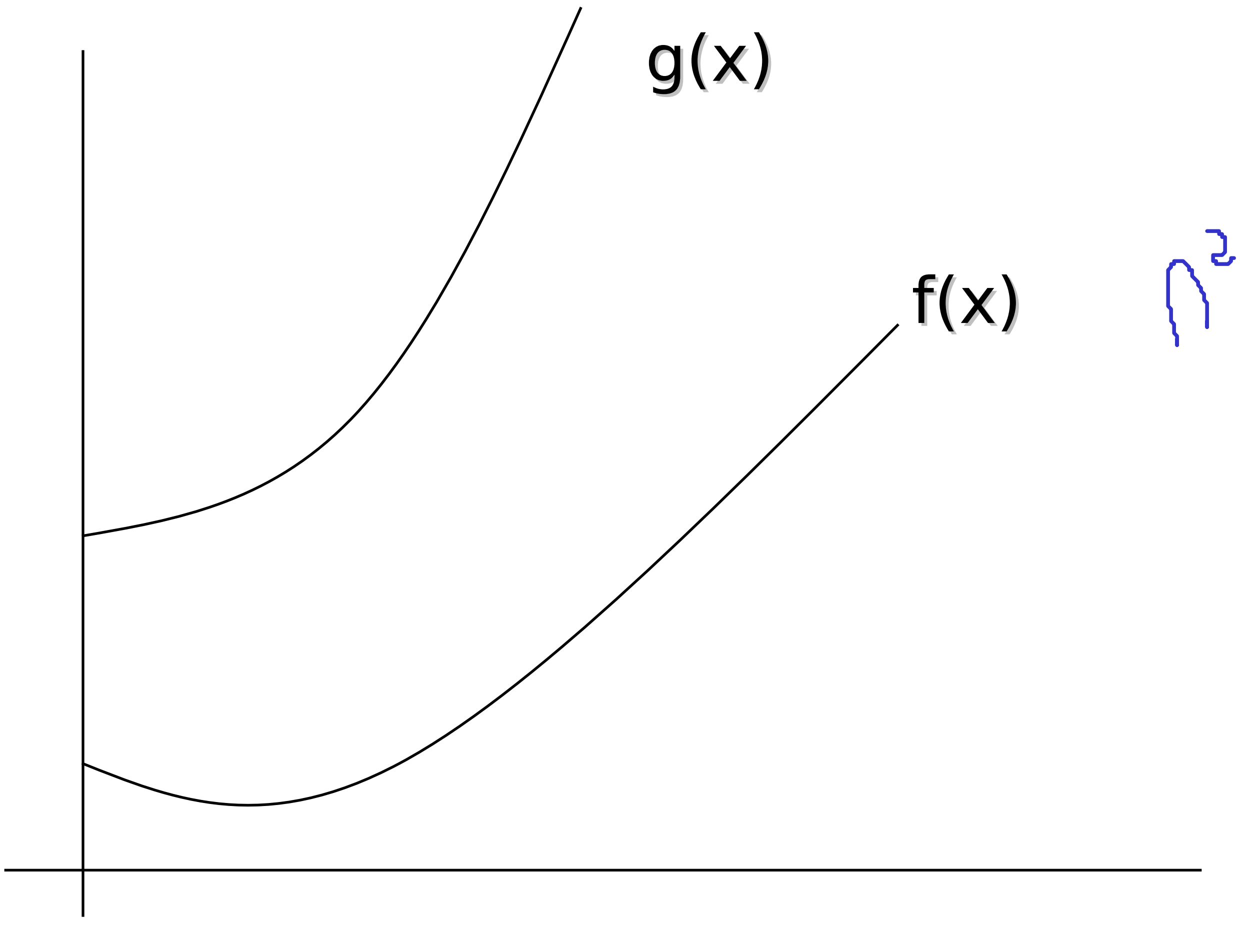
Notación o

Notación ω

Terminología de complejidades

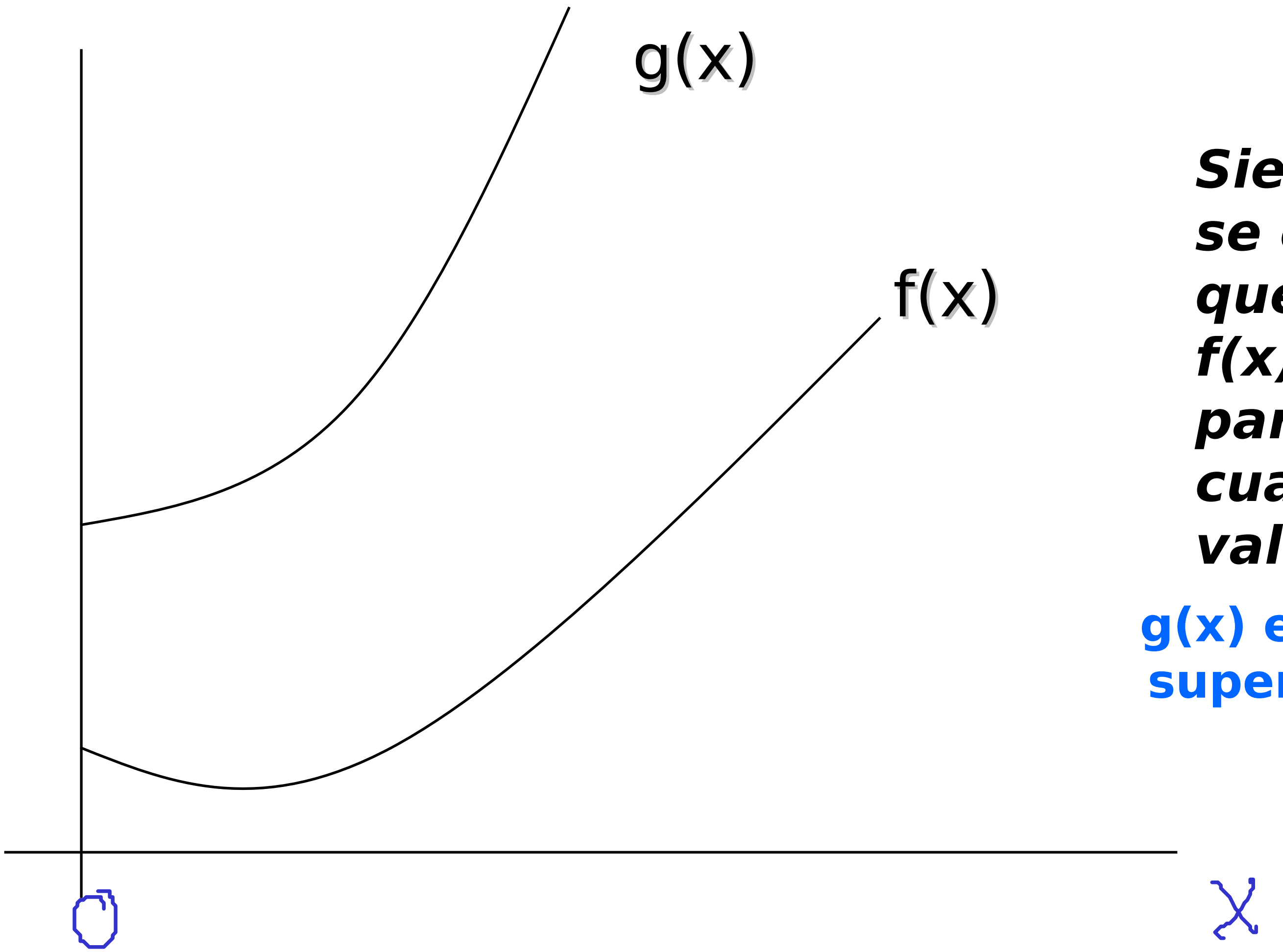
Clasificación de problemas

Crecimiento de funciones



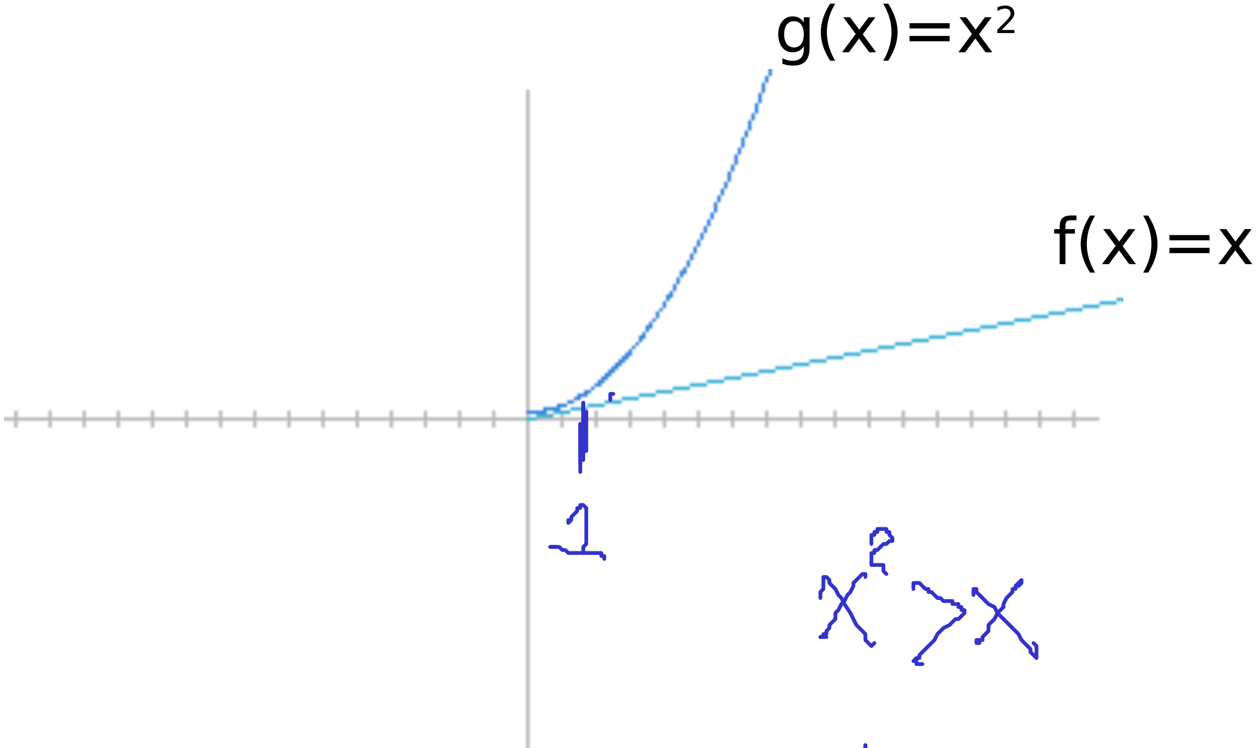
$n^2 + 4n + 5$

Crecimiento de funciones



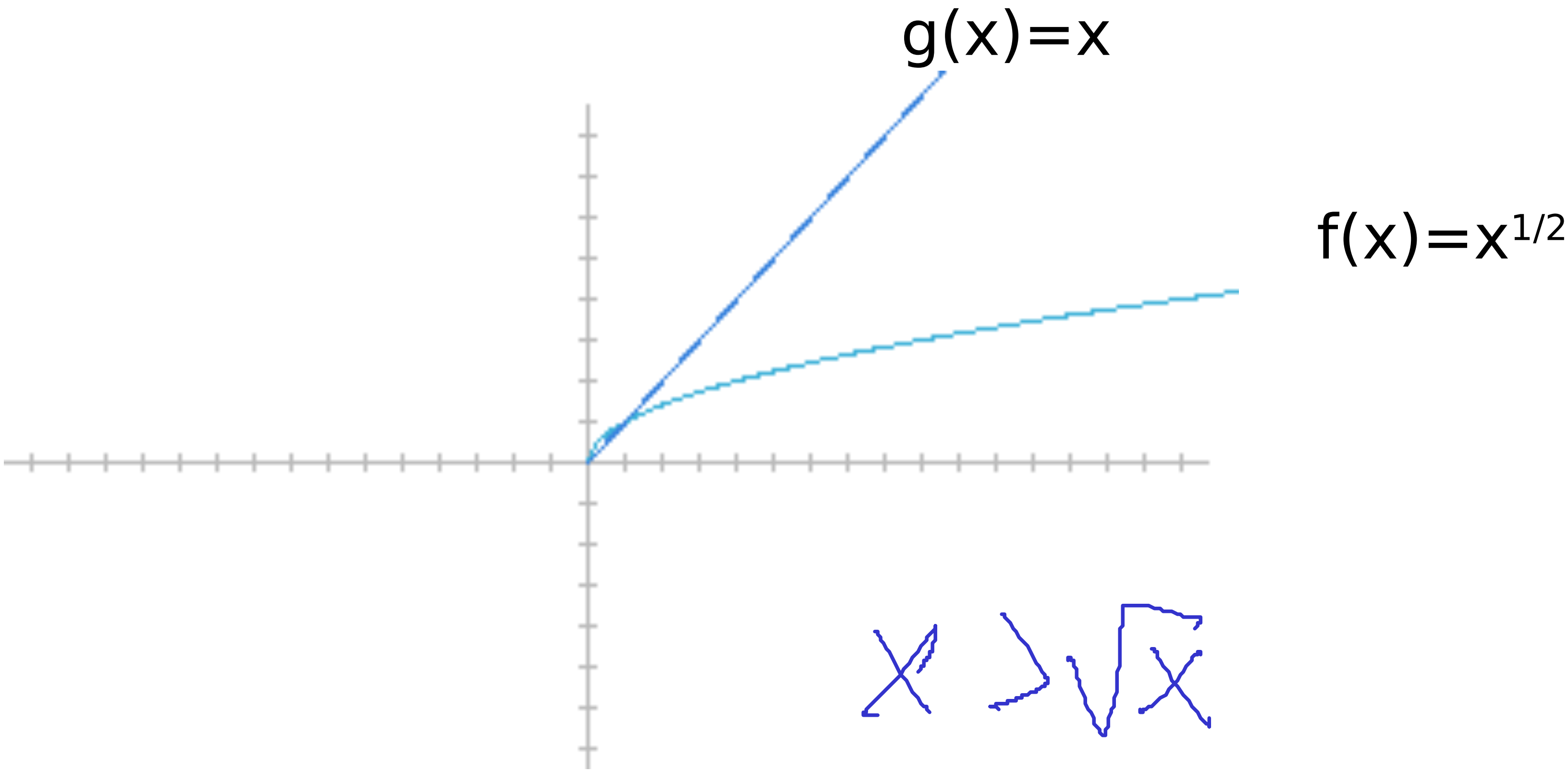
Siempre se cumple que f(x)<g(x) para cualquier valor x

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



$x^2 > x$

$x > 1$

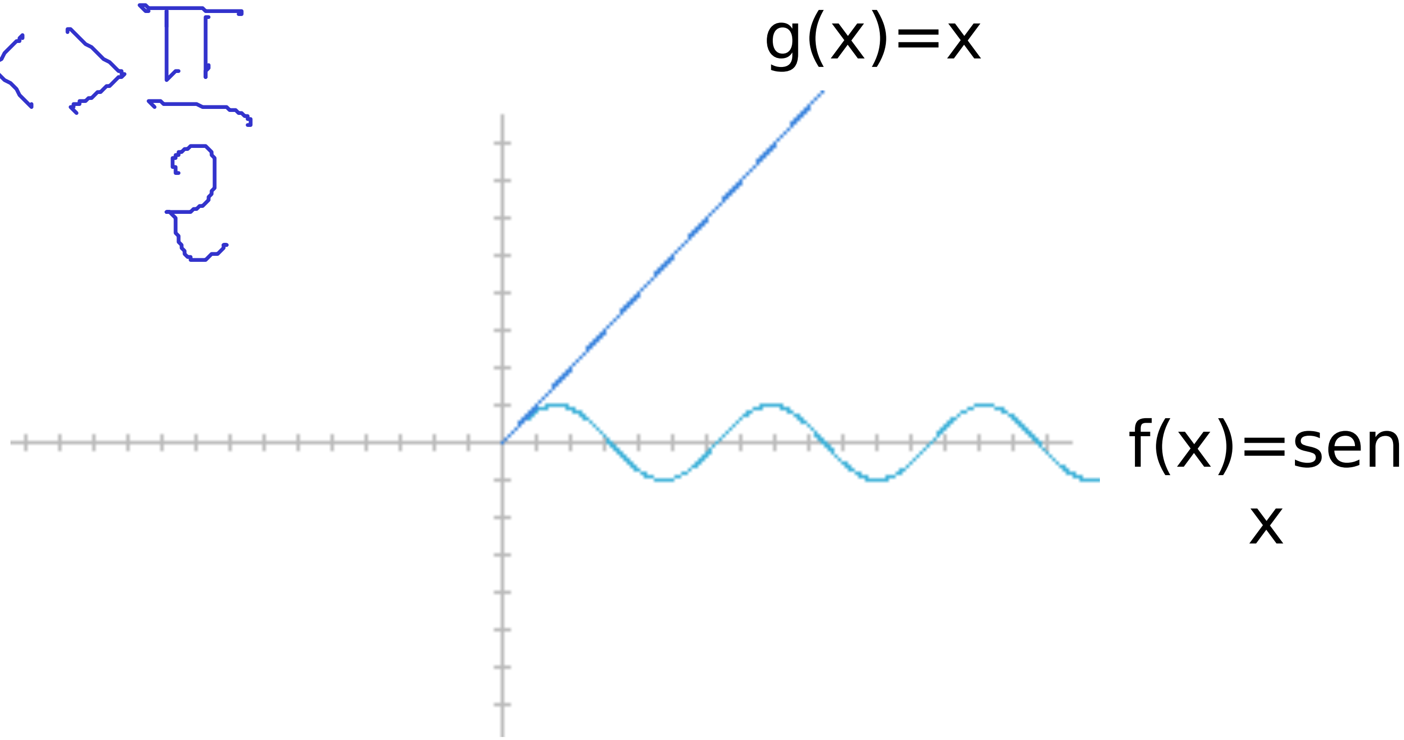


$x > \sqrt{x}$

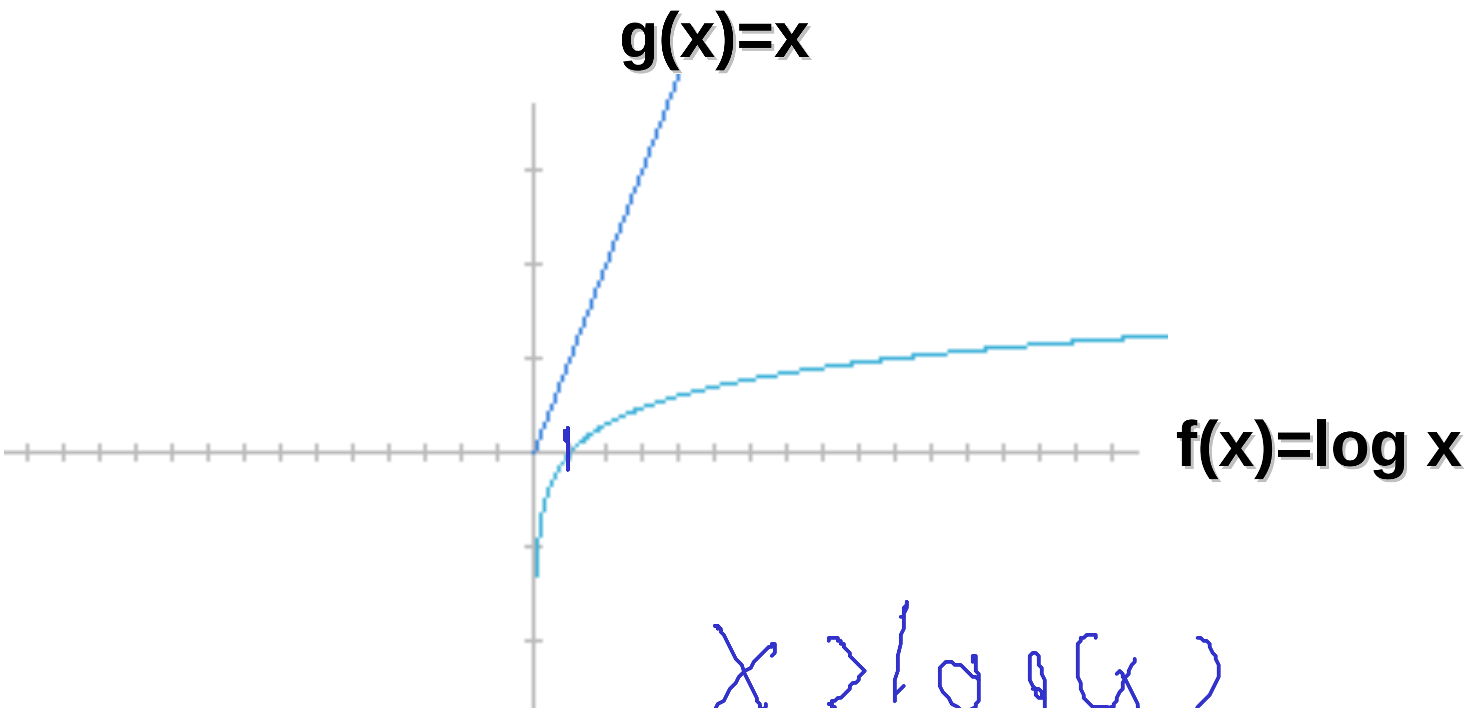
$x > 1$

$X > \mathrm{Sen}(x)$

$x > \frac{\pi}{2}$

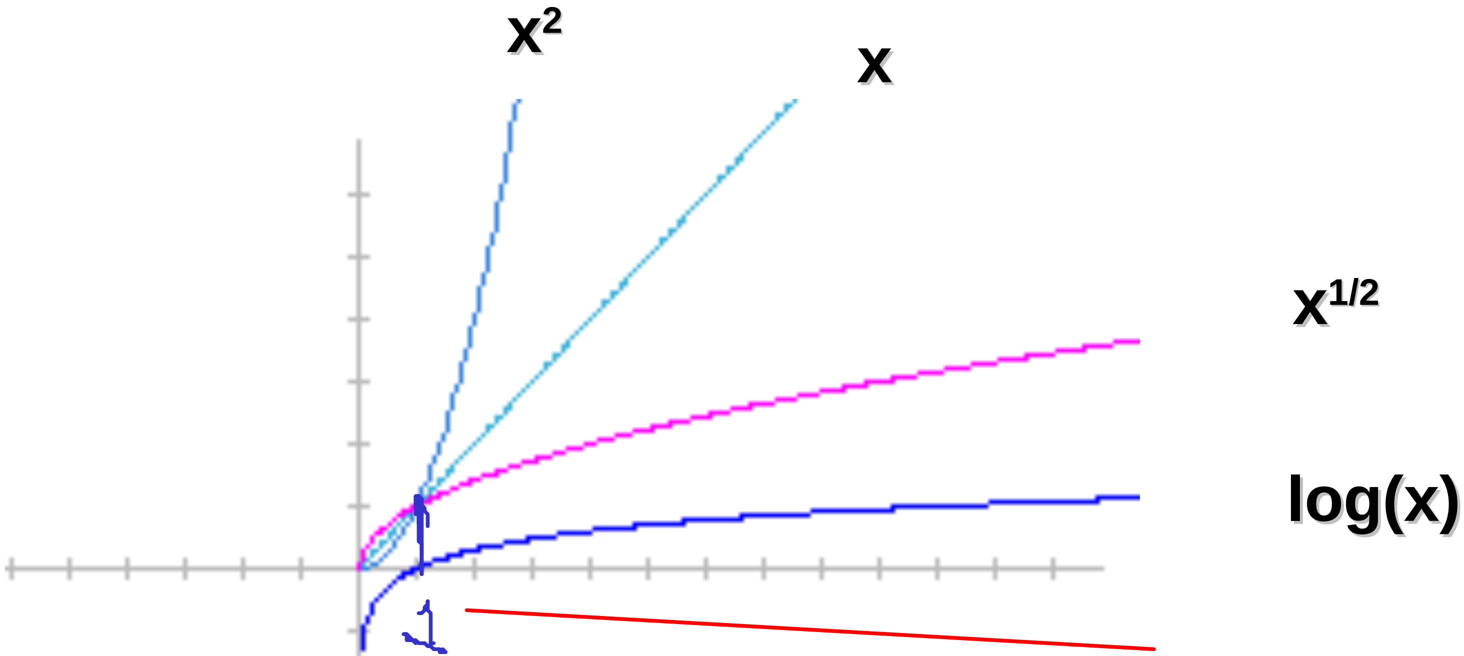


¿Está (sen x) por debajo de x^2?



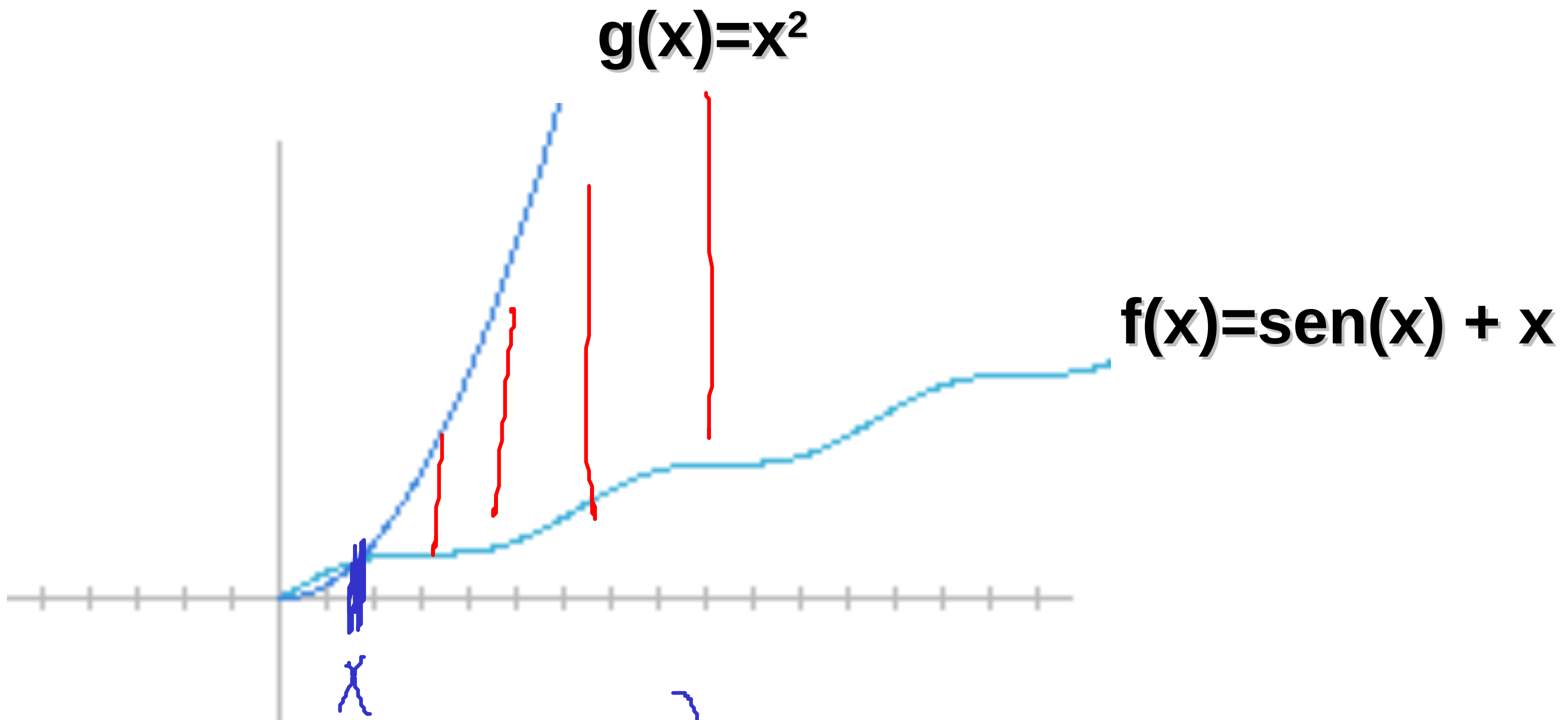
$x > \log(x)$

$x > 0$



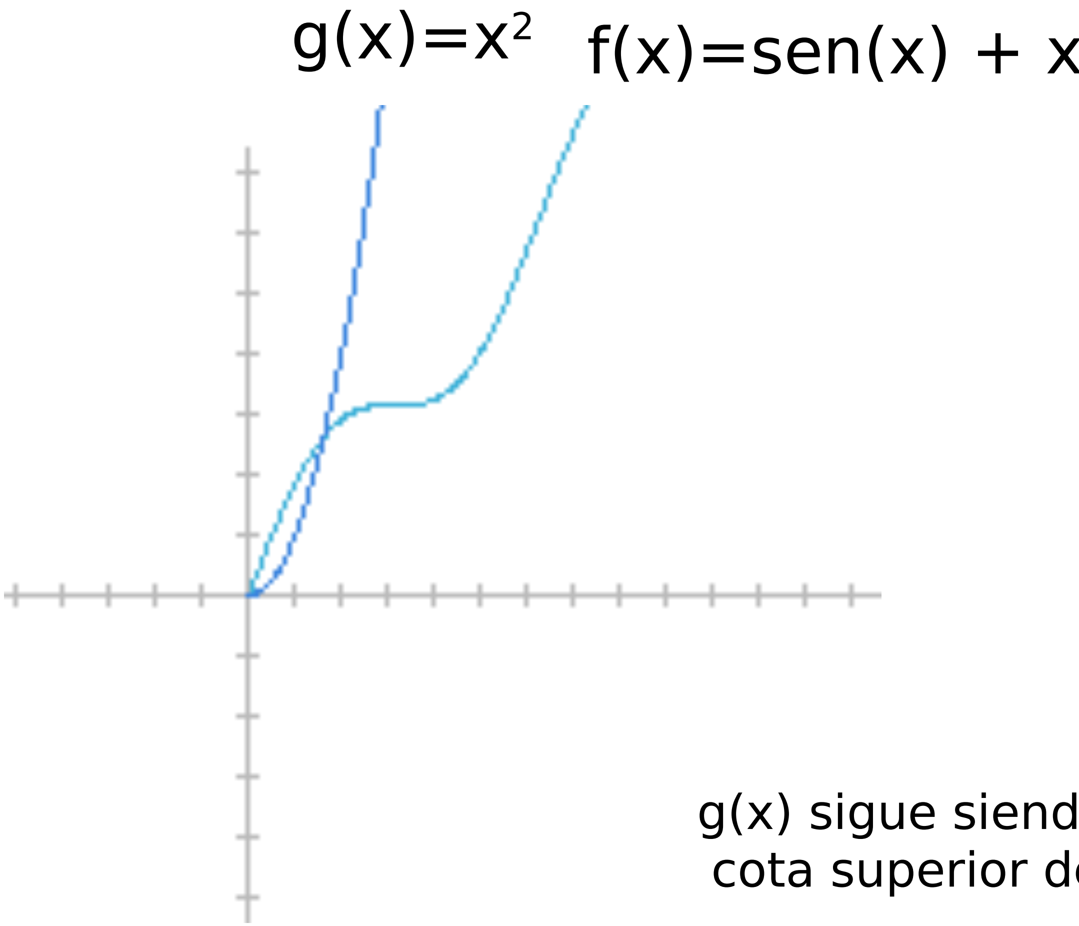
$x^2 > x \qquad x^2 > \log(x)$

$x^2 > \sqrt{x} \qquad x > 1$



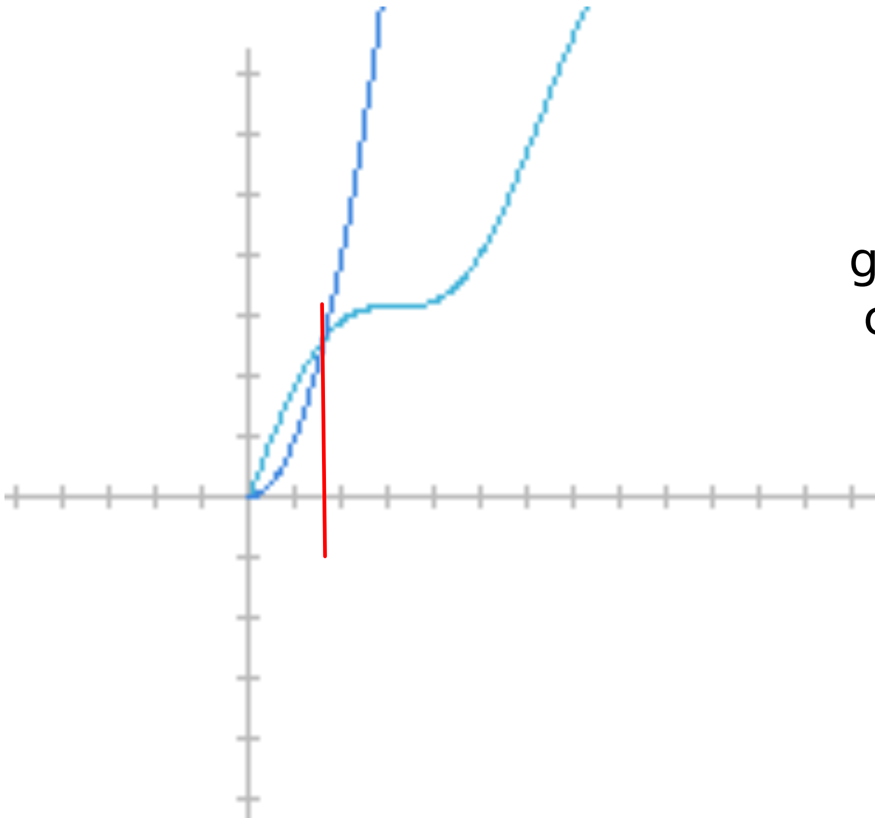
$$x^2 > \operatorname{sen}(x) + x$$

$$x > K$$



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

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

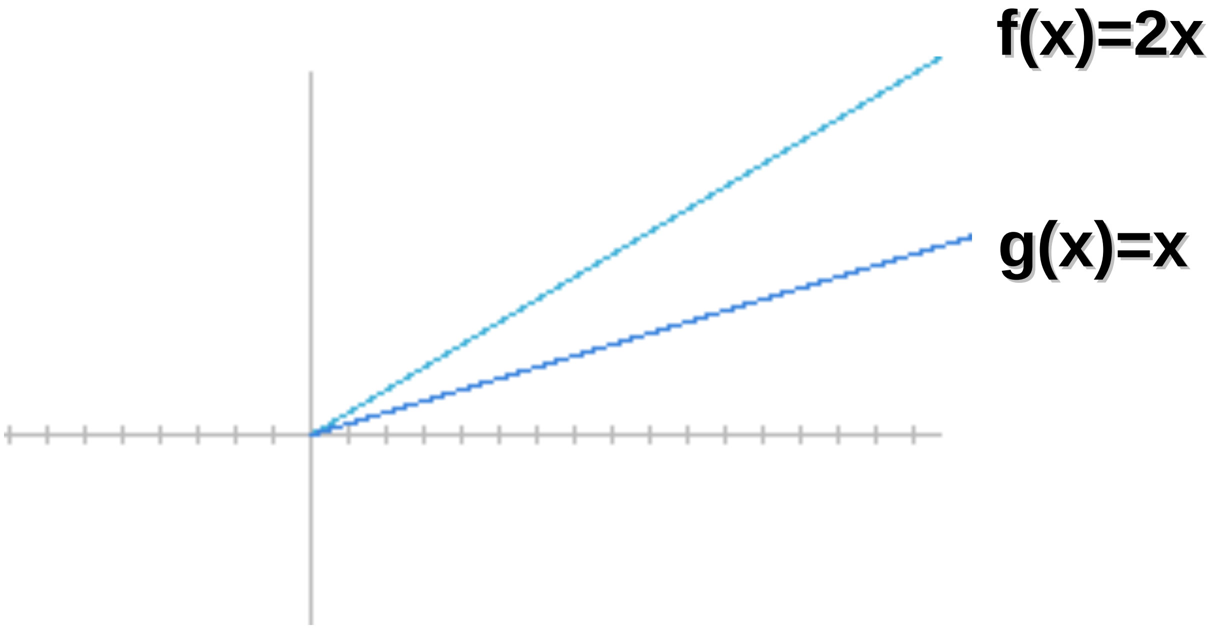


g(x) sigue siendo una cota superior de f(x)

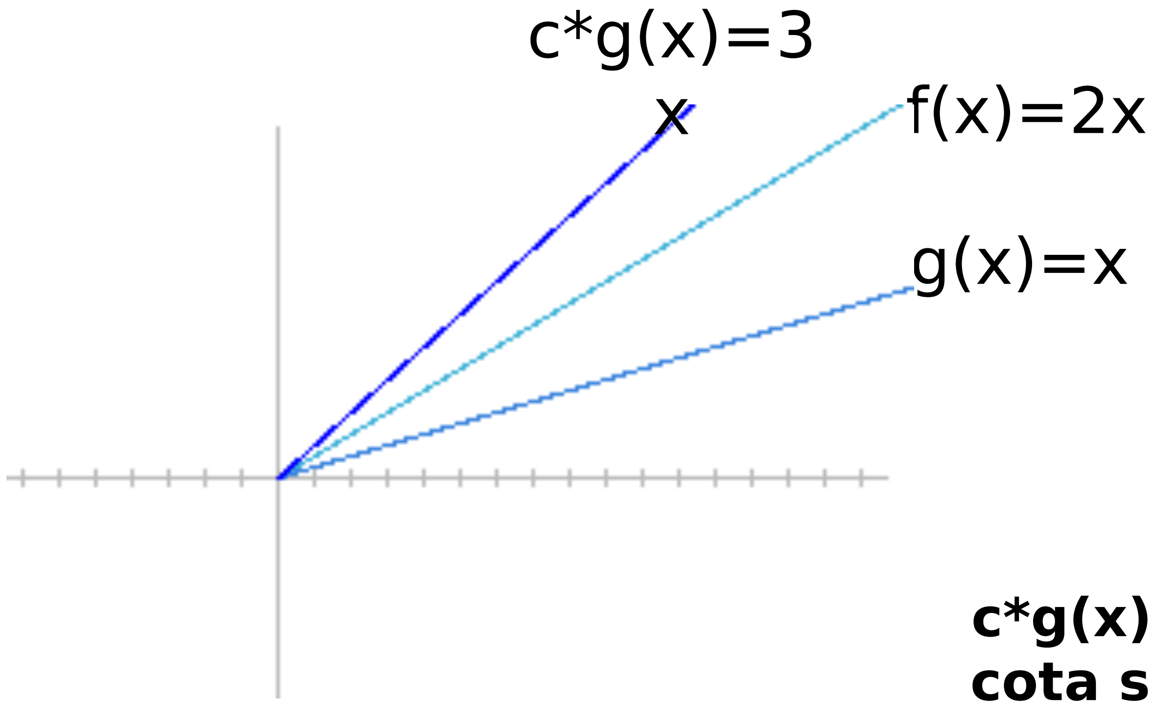
Para que una función sea cota superior de otra, debe serlo para <u>todos</u> los valores de **x≥k**

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)

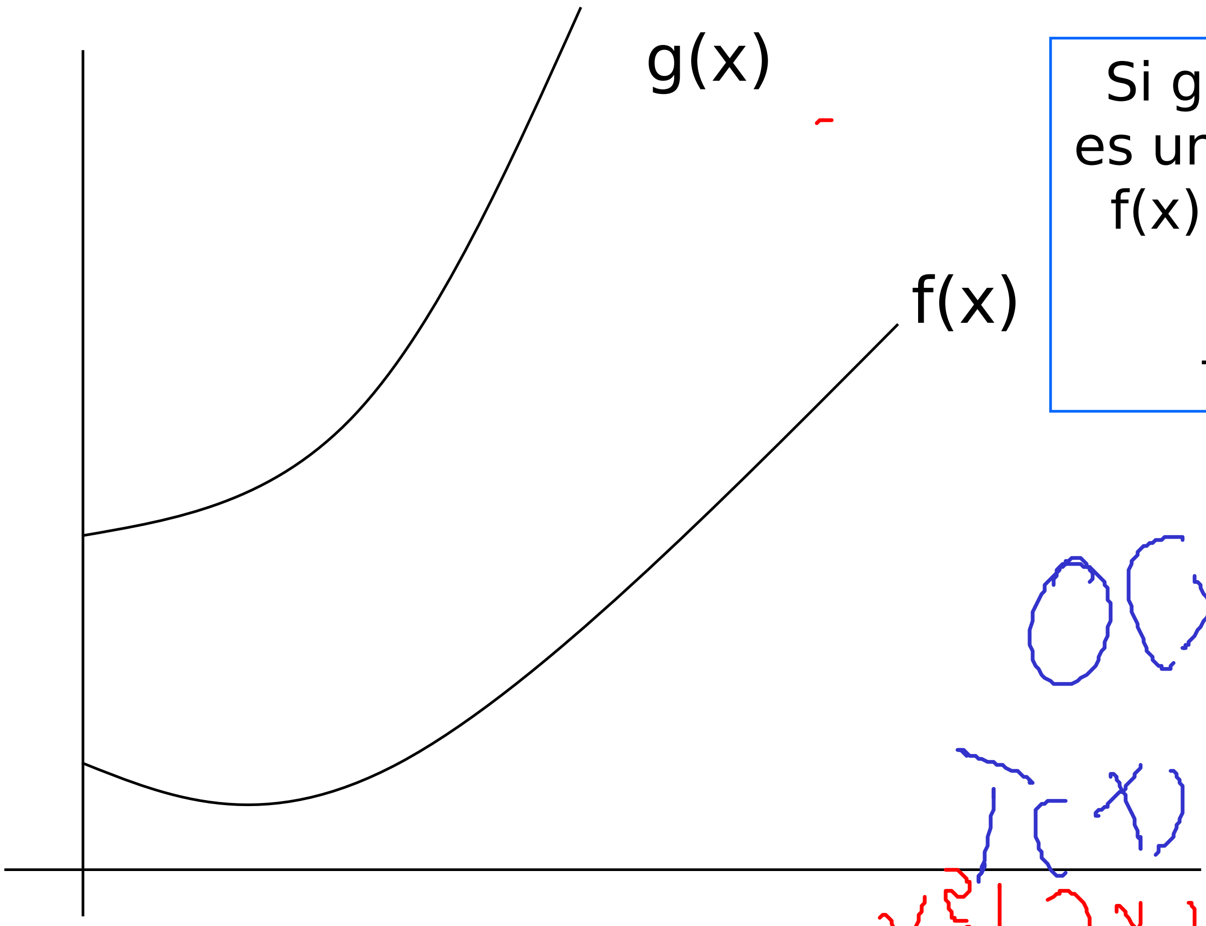


Crecimiento de funciones



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

Crecimiento de funciones



$c \geq 1$

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

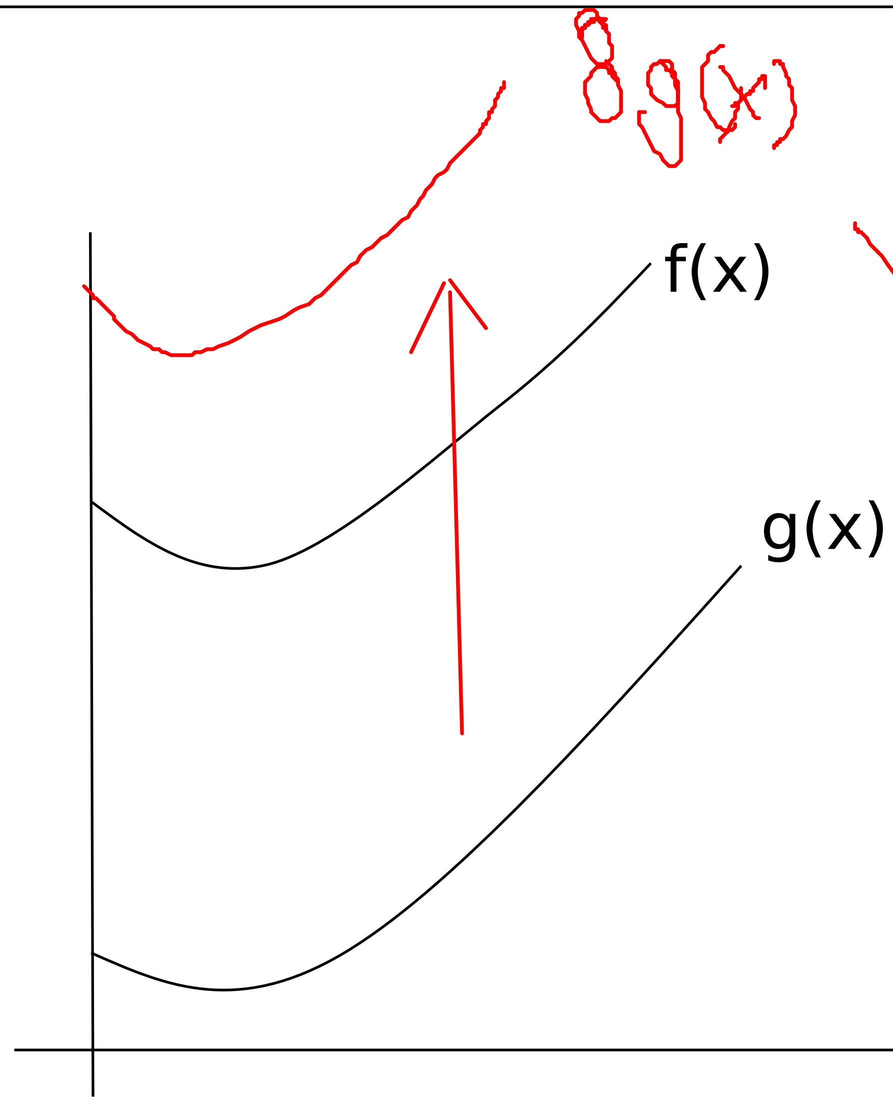
f(x) es O(g(x))

$O(x^2)$

$f(x) \leq c \times x^2$

$x^2 + 2x + 3$

Crecimiento de funciones

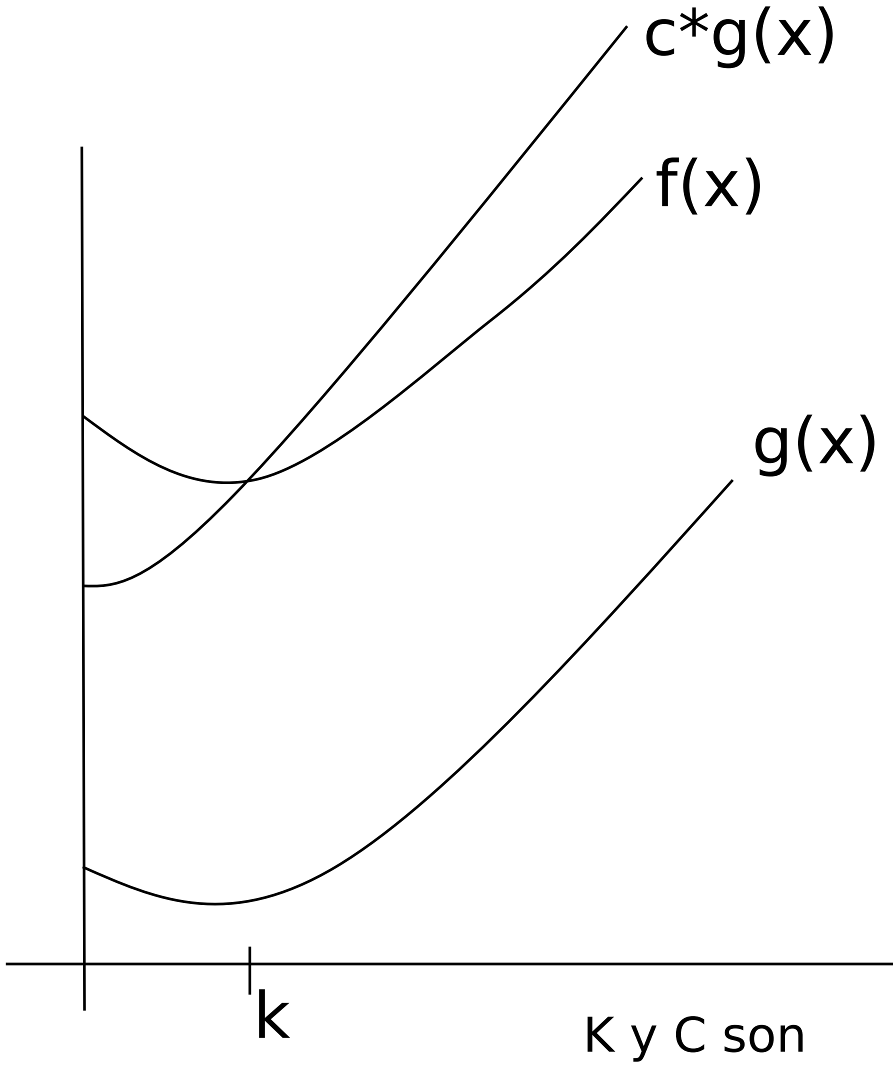


f(x) es O(g(x)) ?

$C \times g(x) > F(x)$

$x \geq k$

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>

K y C son constantes

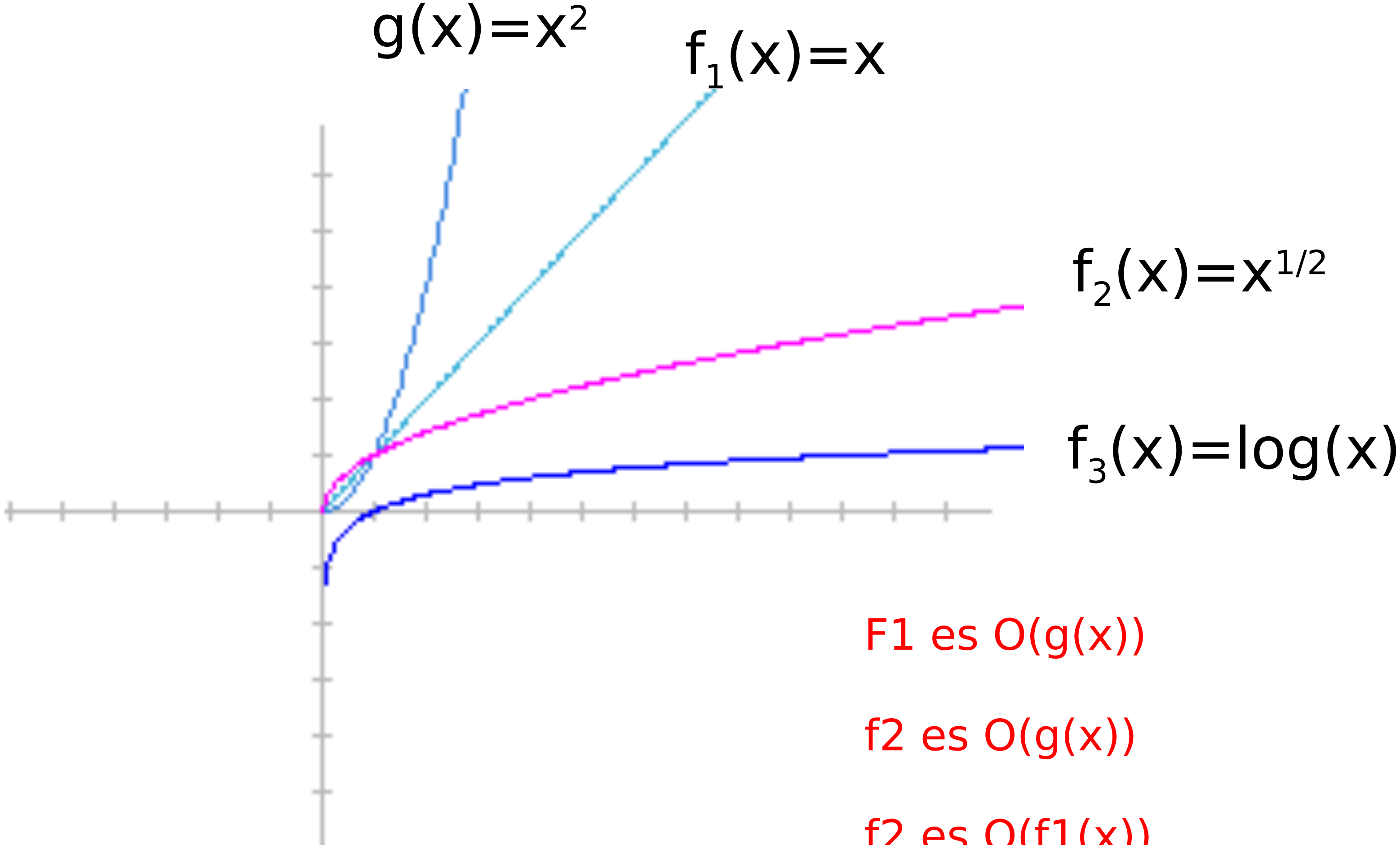
Crecimiento de funciones

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 $x \geq k$, $c \geq 1$

Ψ Donald Knuth(1938-) – Simbolo de Landau



F1 es O(g(x))

f2 es O(g(x))

f2 es O(f1(x))

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

$x \geq 7$

$7x^2 \leq Cx^3$

$C = 1$

$7x^2 \leq x^3$

$7 \leq x$

Crecimiento de funciones

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

$7x^2 \leq x^3$ para $x \geq 7$

Por lo tanto se cumple que

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

para $x \geq 7$

$C=1$, $k=7$

Crecimiento de funciones

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

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 $O(1)$

Crecimiento de funciones

Cota superior asintótica

Cuando se determina que el tiempo de computo de un algoritmo es O(g(x)) se establece una cota superior

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?

¿Que pueden esperar?

1) Solo esta garantizado que lo peor que podemos esperar de los algoritmos para T1 es O(n^2) y para T2 es O(nlogn)

¿T2 es mejor que T1?

2) En caso promedio T1 podría ser mejor que T2 o en el mejor de los casos

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.

Mejor caso $O(n)$, Caso promedio $O(n^2)$ y el peor es $O(n^2)$

Crecimiento de funciones

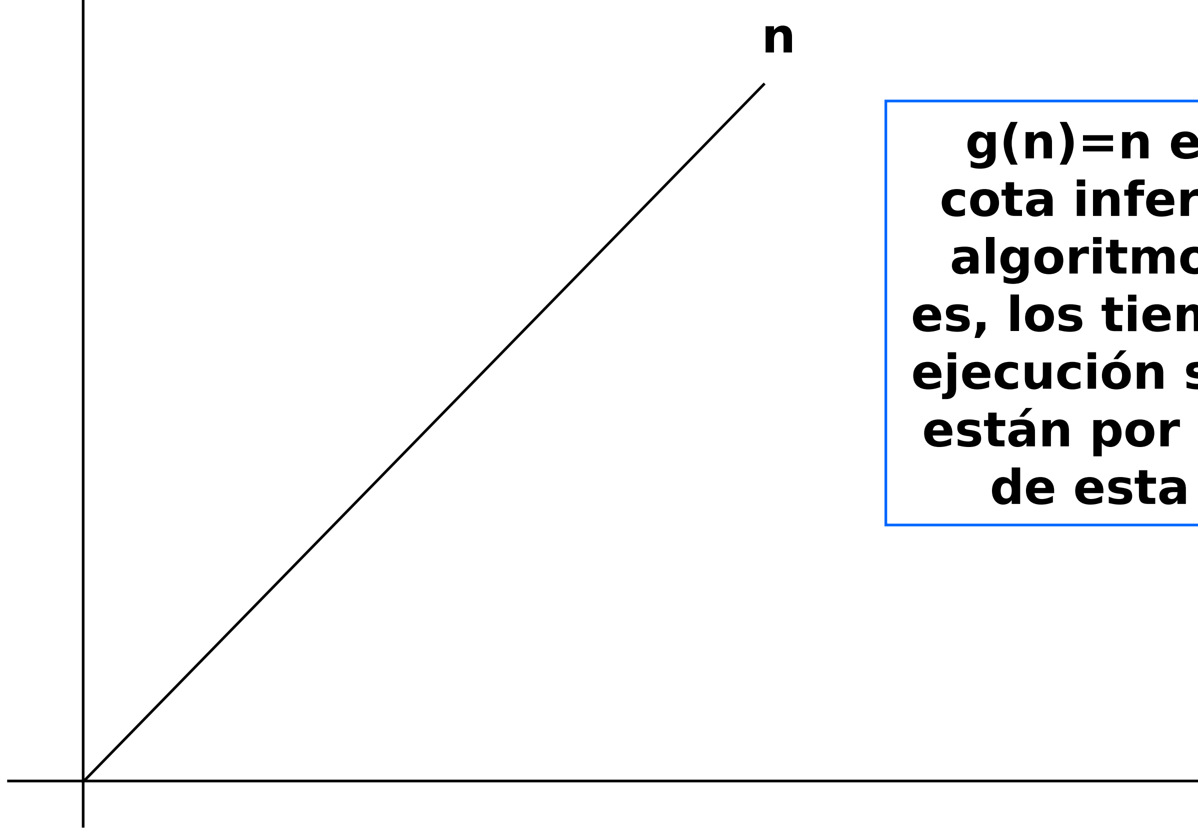
Cota inferior

$\Omega(g(x))$ = { 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?

En el mejor de los casos T2 es mejor que T1, pero no lo puedo asegurar para los demás

Crecimiento de funciones

Notación Θ(f(n))

f(n)=Θ(g(n)) sii f(n)=O(g(n)) y f(n)=Ω(g(n))

$O(n^2)$ $c=4$

$3n^2$

$\Theta(n^2)$

$\Omega(n^2)$ $c=1$

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
$O(n^b)$ $b > 1$	Complejidad polinomial
$O(b^n)$	Complejidad exponencial
$O(n!)$	Complejidad factorial

P

NP

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 se 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)

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

$V_1, V_2, V_3 \ldots$

$(V_1 \wedge V_2) \vee (\overline{V_3}) \ldots$

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