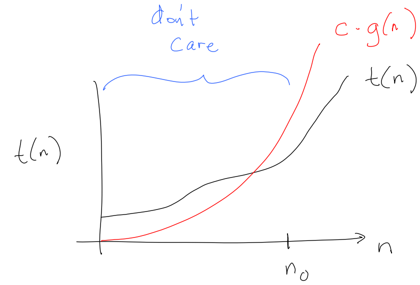
lecture 14

big O (quick review) – upper bound

not big O – not upper bound

big Omega (Ω) – lower bound



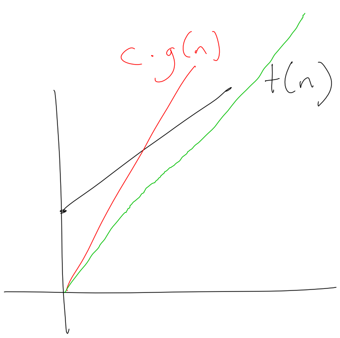
Definition: $t(n)$ is $O(g(n))$ if there exist two positive constants c and n_0 such that, for all $n \geq n_0$,

$$t(n) \leq c\, g(n)$$

Example 1

$5 + 7n$ is $O(n)$

c	n_0
12	1
8	5



Note: you need $c > 7$.

Example 2

$t(n) = 9^n$ is $O(12^n)$.

Proof: (trivial)

$9^n < 12^n$ for all $n \geq 1$.
(Take $c = 1$, $n_0 = 1$).

Question:

12^n is $O(9^n)$?

What does it mean to say "$t(n)$ is not $O(g(n))$" ?

i.e. "$t(n)$ is $O(g(n))$" is false

That is,

"there exist two positive constants c and n_0 such that, for all $n \geq n_0$, $t(n) \leq c\, g(n)$" is false

There do not exist positive constants c, n_0 such that, for all $n \geq n_0$, $t(n) \leq c g(n)$.

Equivalently....

For any positive constants n_0, c, there exists an $n \geq n_0$ such that $t(n) > c \cdot g(n)$.

Predicate Logic (MATH 318, PHIL 210)
Some of you may have seen this ?

$t(n)$ is $O(g(n))$ is written

$$\exists\, n_0, c > 0 \;\; \forall n \geq n_0 \, (t(n) \leq c\, g(n))$$

not ($t(n)$ is $O(g(n))$) is written :

$$\neg \exists\, n_0, c > 0 \, (\forall n \geq n_0 \, (t(n) \leq c\, g(n)))$$

or equivalently

$$\forall\, n_0, c > 0 \;\; \exists\, n \geq n_0 \, (t(n) > c\, g(n))$$

Example 3

$t(n) = 12^n$ is not $O(9^n)$.

Proof : Take any $c, n_0 \geq 1$

Show there exists $n \geq n_0$ such that $12^n > c\, 9^n$.

Take n such that $\left(\frac{12}{9}\right)^n > c$.

e.g. $n > \dfrac{\log_2 c}{\log_2\left(\frac{12}{9}\right)}$.

Example

$t(n) = 3n^2 + 5n + 2$ is not $O(n)$.

Proof Take any $c, n_0 \geq 0$.

We want to show that there exists n, such that $n \geq n_0$ and

$$3n^2 + 5n + 2 > cn.$$

$\therefore$ Divide by n. We want to find $n \geq n_0$ such that

$$3n + 5 + \frac{2}{n} > c.$$

Since $3n + 5 + \frac{2}{n} > 3n + 5$, if $n > 0$;

it is sufficient to find n such that

$3n + 5 > c$ and $n \geq n_0$.

Take $n > \max\left(\frac{c-5}{3}, n_0\right)$

... and I'm done.

big O (quick review) – upper bound

not big O – not upper bound

big Omega (Ω) – lower bound

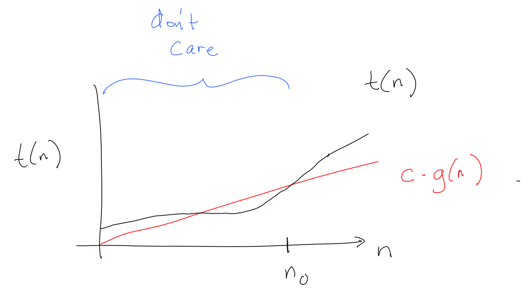
Often we want to make statements about lower bounds such as :

a particular algorithm takes at least a certain amount of time to run.

- finding the max of n numbers takes at least n steps (if the numbers are not sorted)

- searching for an element in a sorted list takes at least $\log n$ steps in the worst case

- any sorting algorithm takes at least $n \cdot \log n$ steps in the worst case (Proved in COMP 251.)



Definition: $t(n)$ is $\Omega(g(n))$ if there exist two positive constants c and n_0 such that, for all $n \geq n_0$,

$$t(n) \geq c\, g(n)$$

Example 4

Show
$$t(n) = \max\left(5, \tfrac{1}{2}(n-30)\right) \text{ is } \Omega(n)$$

Proof: Try $c = \frac{1}{4}$, and find an n_0 such that, for all $n \geq n_0$.

$$\max\left(20, \tfrac{1}{2}(n-30)\right) \geq \frac{n}{4}$$

$$\frac{1}{2}(n-30) \geq \frac{n}{4}$$
$$\iff \frac{1}{4} n \geq 30$$
$$\iff n \geq 60 .$$

see lecture notes for more details

Take $n_0 = 60$, $c = \frac{1}{4}$

Example 5

$$n! \text{ is } \Omega(2^n)$$

Proof We want to show there exists $n_0, c > 0$ such that for all $n \geq n_0$, $n! \geq c\, 2^n$,

that is,

$$\frac{n(n-1)(n-2) \cdots 3 \cdot 2 \cdot 1}{2 \cdot 2 \cdot 2 \cdots 2 \cdot 2 \cdot 2} \geq c, \text{ for all } n \geq n_0$$

Try $c = \frac{1}{2}$

$$\underbrace{\frac{n(n-1)(n-2) \cdots 3 \cdot 2 \cdot 1}{2 \cdot 2 \cdot 2 \cdots 2 \cdot 2 \cdot 2}}_{\text{product of terms which are each } \geq 1} \geq \frac{1}{2}, \text{ for all } n \geq n_0$$

$\therefore$ Take $n_0 = 2$ and we are done.

Example 6

if $t(n)$ is $O(g(n))$

then $g(n)$ is $\Omega(t(n))$

Proof: There exists $c, n_0 \geq 0$ such that $t(n) \leq c\, g(n)$ for all $n \geq n_0$

Thus, $g(n) \geq \frac{1}{c} t(n)$, for all $n \geq n_0$