Cosc 241
Programming and Problem Solving
Practice with Big-O

Michael Albert

michael.albert@cs.otago.ac.nz

UNIVERSITY of OTAGO

SAPERE AUDE

Te Whare Wānanga o Otāgo

NEW ZEALAND

True or false?

a) $n^2 = O(3n^2)$
b) $n^2 = O(1000n)$
c) $1000n = O(n^2)$
d) $n^2 = O(n^2 - 1000n)$

▸ Answers

$O(1)$

a) Intuitively, what does it mean to say that $f = O(1)$?
b) What parts of an algorithm or process typically have running times bounded by $O(1)$?

▸ Answers

'Best' bounds

In each of the following give the 'best' O bound you can for the given expressions (here 'best' means - the simplest form that captures the essential growth rate of the given expression).

a) $24 + 12n^3 + 103n$
b) $1000n^2 + 2^n/1000$
c) $(n - 2)^3$

▸ Answers

Extension

Given two functions $f(n)$ and $g(n)$ is it necessarily the case that either $f(n) = O(g(n))$ or $g(n) = O(f(n))$? What if both functions are increasing?

Answers

a) $n^2 = O(3n^2)$ True.
In fact $n^2 \leq 3n^2$ for all positive integers n.

b) $n^2 = O(1000n)$ False.
No matter how big a constant A we choose, if $n > 1000A$ then $n^2 > A \times (1000n)$.

c) $1000n = O(n^2)$ True.
For $n > 1000$, $1000n < n^2$.

d) $n^2 = O(n^2 - 1000n)$ True.
If $n > 2000$ then $1000n < n^2/2$, so $n^2 - 1000n > n^2/2$. In particular if we choose $A = 2$ (or anything bigger), then for $n > 2000$, $n^2 \leq A(n^2 - 1000n)$.

Back

$O(1)$

a) Intuitively, what does it mean to say that $f = O(1)$?
Formally, it means that there is some constant A such that for sufficiently large n, $f(n) \leq A$. Since f takes on only finitely many values before n is "sufficiently large", we can say there is a constant C such that for all n, $f(n) \leq C$. So, the intuitive meaning, is that $f(n)$ is bounded above by some fixed constant.

b) What parts of an algorithm or process typically have running times bounded by $O(1)$?
Typically the preprocessing and postprocessing phases have such bounds. It should only take constant time to set up, e.g., the user interface for a program or to shut it down.

Back

'Best' bounds

In each of the following give the 'best' O bound you can for the given expressions (here 'best' means - the simplest form that captures the essential growth rate of the given expression).

a) $24 + 12n^3 + 103n = O(n^3)$

The largest of the three terms (for large n) is $12n^3$ *but we don't include any constants since O doesn't care.*

b) $1000n^2 + 2^n/1000 = O(2^n)$

For even moderate values of n, $2^n/1000$ *is much bigger than* $1000n^2$*, so this is the dominant term. Again we don't care about the constant multiplier.*

c) $(n-2)^3 = O(n^3)$

$(n-2)^3$ *and* n^3 *are not that different, so we ignore the constant offset. Or, expand*

$$(n-2)^3 = n^3 - 6n^2 + 12n - 8$$

and then argue as in the first example.

Back

Extension

Given two functions $f(n)$ and $g(n)$ is it necessarily the case that either $f(n) = O(g(n))$ or $g(n) = O(f(n))$?

No. Consider $f(n)$ defined as: if n is odd, $f(n) = 1$, and if n is even $f(n) = n$. Define $g(n)$ similarly: if n is odd, $g(n) = n$, and if n is even $f(n) = 1$. Then, for odd n, $g(n) = n \times f(n)$ so $g(n) \neq O(f(n))$, while the corresponding argument for even n shows that $f(n) \neq O(g(n))$.

What if both functions are increasing?

No, but constructing an example is a bit trickier. One way is to define $f(n)$ and $g(n)$ recursively: $f(1) = g(1) = 1$, and for even n, $f(n)$ is n times one more than the maximum of $f(n-1)$ and $g(n-1)$, while for odd n, $f(n)$ is equal to one more than the maximum of $f(n-1)$ and $g(n-1)$ plus 1. Define g similarly switching even and odd. Then, both are increasing, but otherwise the analysis for the previous example applies.

Back