Addition and Multiplication on Big O notations

Let f and g be two functions. And they are used in other function which adds the output of two function and in other case multiply the output of two function. i.e. if f = O(n) and g = O(n^3), then f + g = ?. How about f*g ?

algorithm

asked Apr 5 '15 at 18:18

user539
45 4

2 f+g is O(n^3). f*g is O(n^4). – Asad Saeeduddin Apr 5 '15 at 18:20

3 This seems like a better question for Mathematics Stack Exchange. – mmking Apr 5 '15 at 18:24

@Asad There's no reason to think that multiplication would have an O(n^4) without knowing more about the result of f and g. Generally, operators have their own order costs which are added together with the costs of functions, not multiplied. – Pi Marillion Apr 5 '15 at 18:38

@PiMarillion Can you provide an example of two functions f and g such that f is O(n^3), g is O(n), and fg!=O(n^4)? – Asad Saeeduddin Apr 5 '15 at 22:10

@Asad I'm not familiar with any common algorithms with an asymptotic time cost of O(n^4), though they do exist. Take the function g(x) = [1, x[1 : n - 1]] * x[0 : n], which has time cost O(n). if f(x) = h(h(h(x))) with time complexity O(n^3), then f(x) * g(x) takes O(n^3) time to calculate f(x), O(n) to calculate g(x), and O(n) to perform the multiplication between them. The result is then O(n^3 + n + n) which is O(n^3). – Pi Marillion Apr 6 '15 at 22:34

1 Answer

O(n) works by taking the largest order term and neglecting the others. So in your example:

`f` = O(n)

`g` = $O(n^3)$

Assuming that `*` and `+` are of O(n) order themselves...

`f + g` = $O_f + O_+ + O_g = O(n) + O(n) + O(n^3) = O(n^3 + 2n) = O(n^3)$

Note that in the last step you just neglect all but the highest order term, and remove any multiple. For `*`, the result is the same.

Another example:

`f + f` = $O_f + O_+ + O_f = O(n) + O(n) + O(n) = O(3n) = O(n)$

Just remember that operators also have a "cost" that's added in with the rest of the operations.

Big O notation always adds, but make sure to take into account if the number of elements changes. (E.g. if you do an opperation that increases the number of elements, then the future instances of `n` are increased. This is important if you do something like matrix multiplication.)

answered Apr 5 '15 at 18:34

Pi Marillion
2,129 8 12

You don't state it explicitly (not entirely sure what "the result is the same" means), you seem to be claiming that `f(x)*g(x)=O(x^3)` . This is demonstrably incorrect: let `f(x)=x` , `g(x)=x^3` (both of these satisfy the given presumptions), then `f(x)*g(x)=x^4` , which is **not** `O(x^3)` . – Asad Saeeduddin Apr 5 '15 at 22:20

I've never seen this approach of an operator "having its own cost", and am fairly certain it is wrong. Could you link to a paper or source where this approach is described? – Asad Saeeduddin Apr 5 '15 at 22:22

@Asad f(x) = x has a time cost of O(1) if done by reference, or O(n) if done by replication. g(x) = x^3 has time cost O(n), like all scalar math operations... Big O notation measures the asymptotic time complexity of an algorithm or operation (most frequently applied to sorting algorithms). It is not the same as the exponent in a polynomial. Apples and oranges. – Pi Marillion Apr 6 '15 at 22:39

A function doesn't have a time cost. It is merely a correspondence between elements in sets. Similarly, big O is simply a mathematical formalism that is useful when applied to functions mapping the input size of an algorithm to the amount of steps it involves or the amount of space it takes. A function f(x)=O(g(x)) if and only if there exist any positive reals M and x_min such that |f(x)|<=|Mg(x)| for all x>=x_min. That is **all** it means. – Asad Saeeduddin Apr 6 '15 at 23:10

You're muddling up the asymptotic *value* of a product of two functions with the asymptotic complexity of some unspecified algorithm that computes it. They don't have the same domains: the former is simply an arbitrary function that maps some real number to some other real number, for example x->x^4 in the example of f(x)*g(x)=x^4. On the other hand, the algorithmic complexities of addition or multiplication aren't a function of the numeric value of the operand, they're functions of the number of digits. – Asad Saeeduddin Apr 6 '15 at 23:20