Algorithms: Complexity Analysis (Examples)

Example #1

Claim : if $T(n) = a_k n^k + ... + a_1 n + a_0$ then

$$T(n) = O(n^k)$$

Proof : Choose $n_0 = 1$ and $c = |a_k| + |a_{k-1}| + .. + |a_1| + |a_0|$

Need to show that $\forall n \geq 1, T(n) \leq c \cdot n^k$

We have, for every $n \geq 1$,

$$\begin{aligned} T(n) &\leq |a_k| n^k + ... + |a_1| n + |a_0| \\ &\leq |a_k| n^k + ... + |a_1| n^k + |a_0| n^k \\ &= c \cdot n^k \end{aligned}$$

Example #2

<u>Claim</u> : for every $k \geq 1,\ n^k$ is not $O(n^{k-1})$

<u>Proof</u> : by contradiction. Suppose $n^k = O(n^{k-1})$

Then there exist constants c, n_0 such that

$$n^k \leq c \cdot n^{k-1} \quad \forall n \geq n_0$$

But then [cancelling n^{k-1} from both sides]:

$$n \leq c \quad \forall n \geq n_0$$

Which is clearly False [contradiction].

Example #1

Claim : $2^{n+10} = O(2^n)$

Proof : need to pick constants c, n_0 such that

$$(*) \quad 2^{n+10} \leq c \cdot 2^n \quad n \geq n_0$$

Note : $2^{n+10} = 2^{10} \times 2^n = (1024) \times 2^n$

So if we choose $c = 1024, n_0 = 1$ then (*) holds.

Q.E.D

Example #2

Claim : $2^{10n} \neq O(2^n)$

Proof : by contradiction. If $2^{10n} = O(2^n)$ then there exist constants $c, n_0 > 0$ such that

$$2^{10n} \leq c \cdot 2^n \quad n \geq n_0$$

But then [cancelling 2^n]

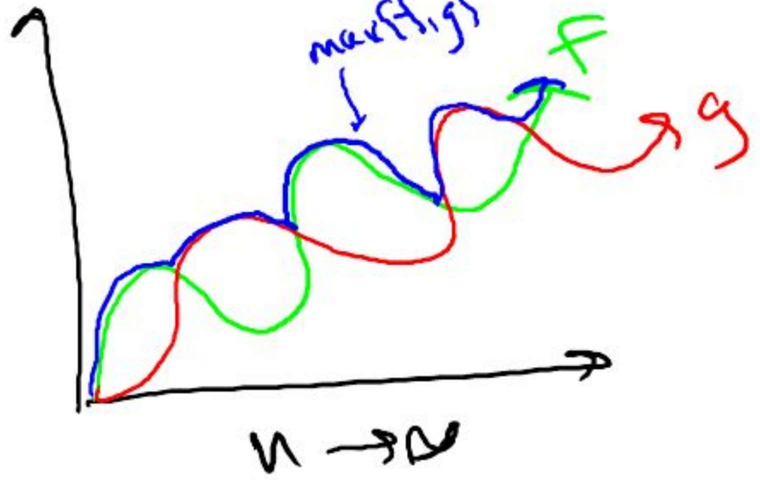
$$2^{9n} \leq c \quad \forall n \geq n_0$$

Which is certainly false. Q.E.D

Example #3

<u>Claim</u> : for every pair of (positive) functions f(n), g(n),

$$max\{f, g\} = \theta(f(n) + g(n))$$



Example #3 (continued)

Proof : $max\{f, g\} = \theta(f(n) + g(n))$

For every n, we have

$$max\{f(n), g(n)\} \leq f(n) + g(n)$$

And

$$2 * max\{f(n), g(n)\} \geq f(n) + g(n)$$

Thus $\frac{1}{2} * (f(n) + g(n)) \leq max\{f(n), g(n)\} \leq f(n) + g(n) \quad \forall n \geq 1$

$=> max\{f, g\} = \theta(f(n) + g(n))$ $[where\ n_0 = 1, c_1 = 1/2, c_2 = 1]$

Q.E.D

For example,
max(2,3) <= (2+3)
3 <= 5
But,

2*max(2,3) >= (2+3)
=> 2 * 3 >= (2+3)
=> .5 * (2*3) >= .5 *(2+3)
=> 3 >= .5 *(2+3)

=> max(2,3) >= .5*(2+3)
=> .5*(2+3) <= max(2,3)

Big O notation: properties

Reflexivity. f is $O(f)$.

Constants. If f is $O(g)$ and $c > 0$, then $c\,f$ is $O(g)$.

Products. If f_1 is $O(g_1)$ and f_2 is $O(g_2)$, then $f_1 f_2$ is $O(g_1 g_2)$.

Pf.

- $\exists\, c_1 > 0$ and $n_1 \geq 0$ such that $0 \leq f_1(n) \leq c_1 \cdot g_1(n)$ for all $n \geq n_1$.
- $\exists\, c_2 > 0$ and $n_2 \geq 0$ such that $0 \leq f_2(n) \leq c_2 \cdot g_2(n)$ for all $n \geq n_2$.
- Then, $0 \leq f_1(n) \cdot f_2(n) \leq \underbrace{c_1 \cdot c_2}_{c} \cdot g_1(n) \cdot g_2(n)$ for all $n \geq \underbrace{\max\{n_1, n_2\}}_{n_0}$. ▪

Sums. If f_1 is $O(g_1)$ and f_2 is $O(g_2)$, then $f_1 + f_2$ is $O(\max\{g_1, g_2\})$.

ignore lower-order terms

Transitivity. If f is $O(g)$ and g is $O(h)$, then f is $O(h)$.

Ex. $f(n) = 5n^3 + 3n^2 + n + 1234$ is $O(n^3)$.

Suggested Reading

- Algorithms (CLRS)
  - Chapter 3
    - Section 3.1
- Algorithm illuminated (Part 1) by Tim Roughgarden
  - Chapter 2
    - Section 2.3. 2.5