CS350 ABET Objective 2

Use the Big Oh notation.

Textbook Section 2.2, ~5%.

The efficiency analysis of algorithms focuses on the order of growth of the basic operation count.

Introduce three notations that express the order of growth of functions and relate the efficiency of an algorithm being investigated to the efficiency of other algorithms that are known to be either feasible or unfeasible.

Big Oh

A function $t(n)$ is in $O(g(n))$ if there exist a positive constant c and non-negative integer n_0 such that:

$$t(n) \leqslant c\, g(n) \qquad for\, n \geqslant n_0$$

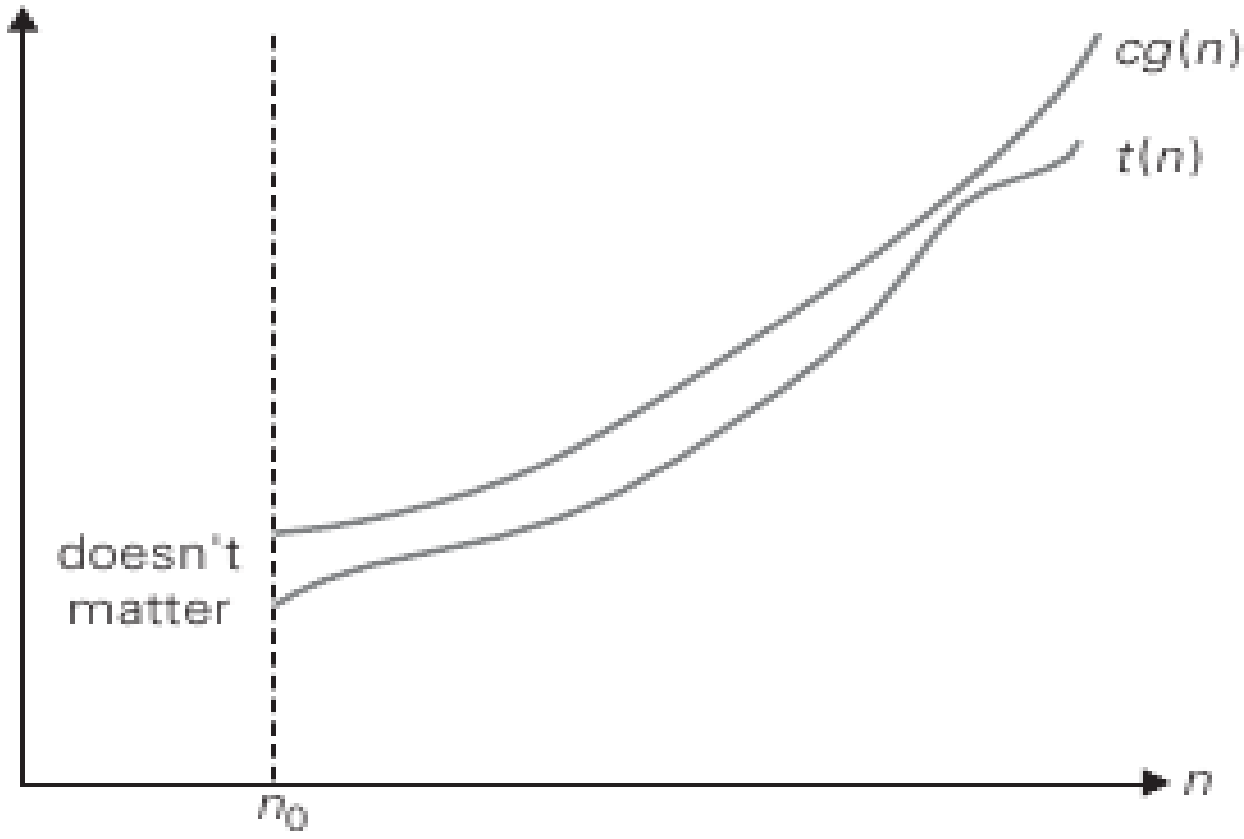
Question/example: is $100n+5 \in O(n^2)$?

Yes: verify for $c=100$ and $n_0=5$.

Often $t(n) \in O(g(n))$ is instead written as $t(n)=O(g(n))$ which is confusing!

Understanding Big Oh

Intuition: $t(n)$ grows no faster than $g(n)$ except perhaps for small values of the argument and apart from a constant multiplier.



If we run a $g(n)$ algorithm, most likely we can run a $t(n)$ algorithm as well.

Big Omega

A very similar to Big Oh, but now the function $t(n)$ grows no slower than $g(n)$.

A function $t(n)$ is in $\Omega(g(n))$ if there exist a positive constant c and non-negative integer n_0 such that:

$$t(n) \geqslant c\, g(n) \qquad for\ n \geqslant n_0$$

Question/example: is $n^3 \in O(n^2)$?

Yes: verify for $c = 1$ and $n_0 = 0$.

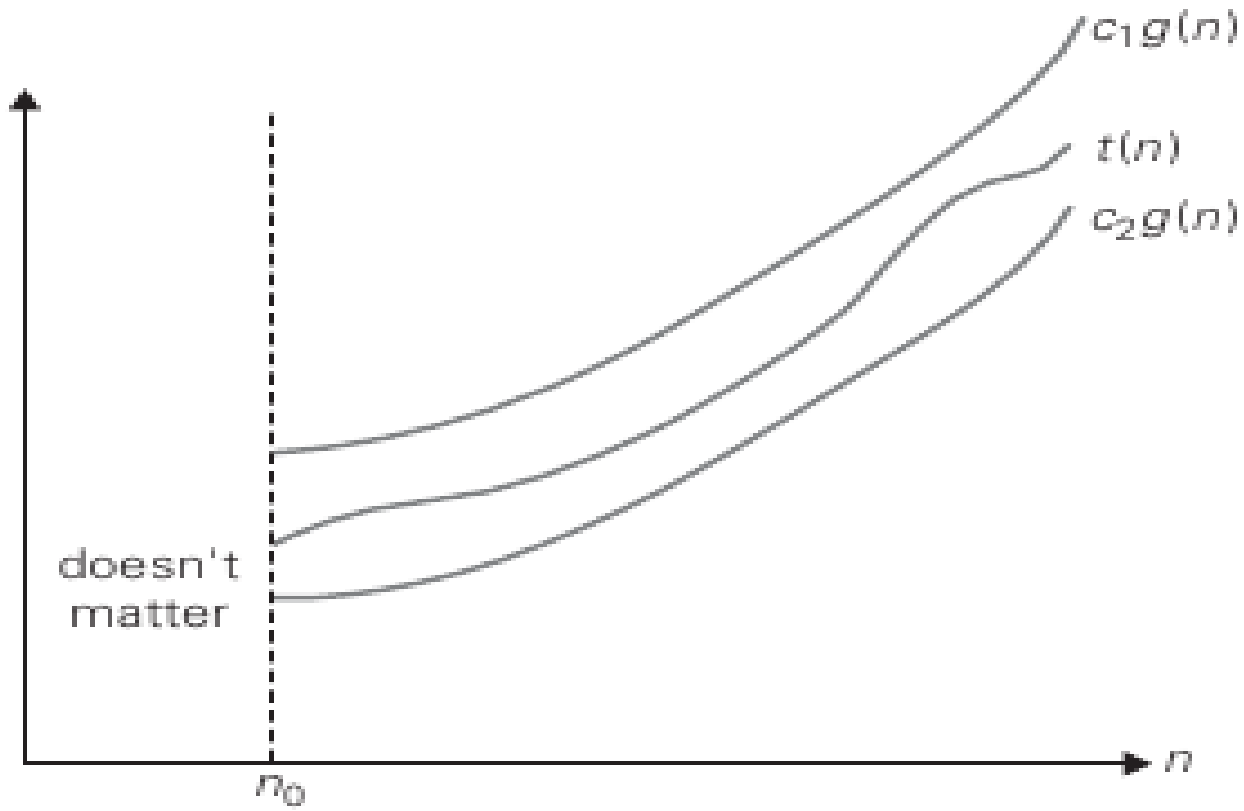
If we cannot run a $g(n)$ algorithm, most likely we cannot run a $t(n)$ algorithm as well. Negative result, but useful to know.

Big Theta

Combines Big Oh and Big Omega together.

A function $t(n)$ is in $\Theta(g(n))$ if there exist positive constants c_1 and c_2 and a non-negative integer n_0 such that:

$$c_2 g(n) \geqslant t(n) \geqslant c_1 g(n) \quad for\ n \geqslant n_0$$



Understanding Big Theta

Intuition: $t(n)$ grows as $g(n)$ except perhaps for small values of the argument and apart from constant multipliers.

Most likely, we run a $t(n)$ algorithm iff we run a $g(n)$ algorithm.

Question/example: is $\frac{n(n-1)}{2} \in \Theta(n^2)$?

Yes: verify for $c_1 = 1/2$ and $c_2 = 1/4$ and $n_0 = 2$.

Hint: let $n \geqslant 2$ then:

$$\frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} \geqslant \frac{n^2}{2} - \frac{n}{2}\frac{n}{2} = \frac{n^2}{4}$$

A property of Big Oh

Intuition: an algorithm is made of two consecutive parts of which we know the growth rate. What is the growth rate of the whole algorithm?

Theorem. If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$ then $t_1(n) + t_2(n) \in O(max(g_1(n), g_2(n)))$

Example. Tell whether an array contains a duplicated element.

Hint. There is an obvious $O(n^2)$ algorithm. There is also a two-part, more efficient algorithm whose analysis uses the theorem.

Theorem is adapted to Ω and Θ .

Limits and Growth

Let $t(n)$ and $g(n)$ be functions:

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}=\begin{cases}0 & t(n)\,grows\,slower\,than\,g(n),\\ c & t(n)\,grows\,as\,g(n),\\ \infty & t(n)\,grows\,faster\,than\,g(n).\end{cases}$$

Using limits is convenient for the following reasons:

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}=\lim_{n\to\infty}\frac{t'(n)}{g'(n)}$$

and

$$n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \qquad for\,large\,n$$

Example: compare growth of $n!$ and 2^n

References

Textbook Section 2.2

Web:

http://en.wikipedia.org/wiki/Big_O_notation