

lecture 13

big O

Big O (Worst Case)

e.g. we say that an algorithm that requires at most

$$c_0 + c_1 n + c_2 n^2$$

operations is $O(n^2)$

Big O (Worst Case)

e.g. we say that an algorithm that requires at most

$$c_0 + c_1 n + c_2 n \log n$$

operations is $O(n \log n)$

power(x, n) } are $O(\log n)$
binary search(a, e) }

merge(l1, l2) } are $O(n)$
factorial(n) }

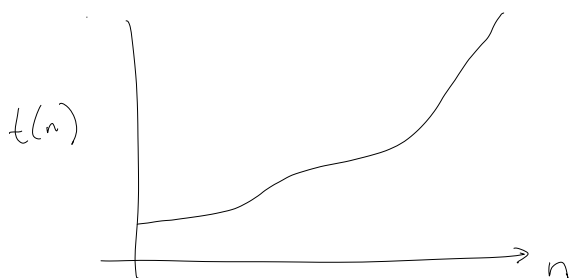
mergesort(list) is $O(n \log n)$

insertionsort(list) is $O(n^2)$

tower of Hanoi(n) is $O(2^n)$

⋮

More formal definition of $O()$



Suppose we have a sequence $t(n)$.

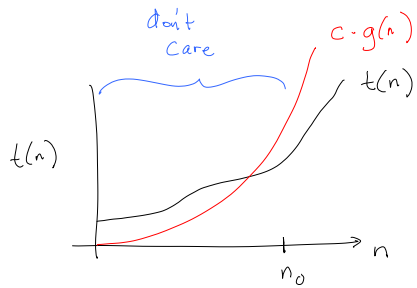
e.g. how the number of operations of an algorithm depends on n

$t(n)$ might be complicated

$$\text{e.g. } t(n) = c_0 + c_1 \log n + c_2 \sqrt{n} + c_3 n + c_5 n \log n + c_6 n^2$$

We want to talk about a simpler function $g(n)$ that defines an "asymptotic" upper bound on $t(n)$.

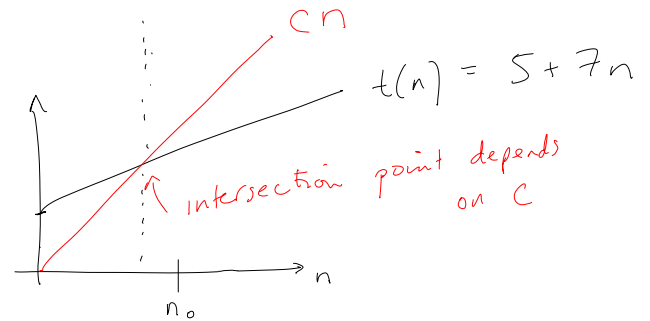
e.g. $g(n)$ would be just n^2 in the above example.



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

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

Example:
Let $t(n) = 5 + 7n$ and let $g(n) = n$.
Then $t(n)$ is $O(g(n))$.



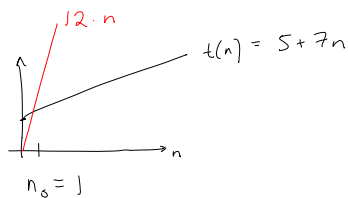
Example:

Let $t(n) = 5 + 7n$ and let $g(n) = n$.
Then $t(n)$ is $O(g(n))$.

Proof(1):

$$\begin{aligned} t(n) &= 5 + 7n \\ &\leq 5n + 7n, \text{ for } n \geq 1 \\ &= 12n \end{aligned}$$

Take $c = 12, n_0 = 1$



How NOT to do a proof

Claim:

Let $t(n) = 5 + 7n$ and let $g(n) = n$.
Then $t(n)$ is $O(g(n))$.

Proof (incorrect)

$$\begin{aligned} 5 + 7n &\leq cn \\ 5n + 7n &\leq cn, \text{ if } n \geq 1 \\ 12n &\leq cn \\ \therefore c &\geq 12 \end{aligned}$$

Q: Why is the above "proof" incorrect?

A: • Was the first statement ($5 + 7n \leq cn$) your hypothesis, or something you know is true because it was given?

• Is the first statement supposed to be true for all n , or for some n , or what?

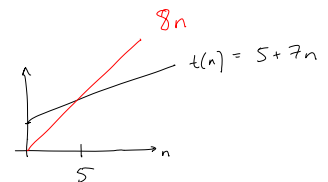
• What is the logical relationship between statements?

Let $t(n) = 5 + 7n$ and let $g(n) = n$.
Then $t(n)$ is $O(g(n))$.

Proof(2):

$$\begin{aligned} t(n) &= 5 + 7n \\ &\leq n + 7n, \text{ for } n \geq 5 \\ &= 8n \end{aligned}$$

Take $c = 8, n_0 = 5$



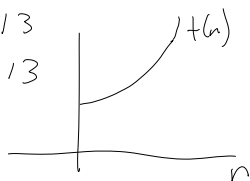
Example 2

Let $t(n) = 9n^2 - 15n + 46$.

Show $t(n)$ is $O(n^2)$.

Note: It is possible for real problems to have minus sign in coefficients.

$$\begin{aligned} t(n) &= 9(n-2)^2 + 3(n-1) + 13 \\ &= 9(n^2 - 2n + 4) + 3n - 3 + 13 \\ &= 9n^2 - 15n + 46 \end{aligned}$$



Let $t(n) = 9n^2 - 15n + 46$.

Show $t(n)$ is $O(n^2)$.

Proof (1)

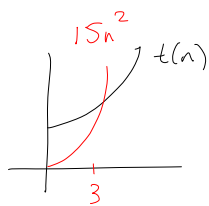
$$\begin{aligned} &9n^2 - 15n + 46 \\ &< 9n^2 + 46, \text{ if } n > 0 \\ &\leq 9n^2 + 46n^2, \text{ if } n \geq 1 \\ &= 55n^2 \end{aligned} \quad \text{So take } c = 55 \\ n_0 = 1$$

Let $t(n) = 9n^2 - 15n + 46$.

Show $t(n)$ is $O(n^2)$.

Proof (2)

$$\begin{aligned} &9n^2 - 15n + 46 \\ &< 9n^2 + 46, \text{ if } n > 0 \\ &< 9n^2 + 6n^2, \text{ if } n \geq 3 \end{aligned}$$



$$= 15n^2 \quad \text{So take } c = 15 \\ n_0 = 3$$

Let $t(n) = 9n^2 - 15n + 46$.

Show $t(n)$ is $O(n^2)$.

Proof (3):

$$\begin{aligned} &9n^2 - 15n + 46 < cn^2 \\ \text{if and only if } &\iff 9 - 15\frac{n}{n^2} + 46\frac{1}{n^2} < c \end{aligned}$$

"Letting $n \rightarrow \infty$ gives $9 < c$.
Therefore, there exists an n_0 that satisfies the big O definition."
Yes BUT this proof does not provide the n_0 .

"Taking the limit" proofs of big O statements are correct provided that:

1) you know what it means (it is more subtle than you think - see MATH 242 Real Analysis)

2) you are asked to show there exists a pair (c, n_0) but you are not asked for a **specific** c, n_0 values.

In COMP 250, taking limits not allowed.

What does $O(1)$ mean?

$t(n)$ is $O(1)$ if there exists positive constants c, n_0 such that, for all $n \geq n_0$,

$$t(n) \leq c$$

i.e. it means $t(n)$ is bounded above by a constant.