
BIG O !
makeameme.org

Big-O notation Part I

COMP 250: Winter 2018

Lecture 11

Carlos G. Oliver

Slides adapted from M. Langer and M. Blanchette

Running time of selection sort

- We showed that running selection sort on an array of n elements takes in the worst case $T(n) = 1 + 15\,n + 5\,n^2$ primitive operations
- When n is large, $T(n) \approx 5\,n^2$
- When n is large,

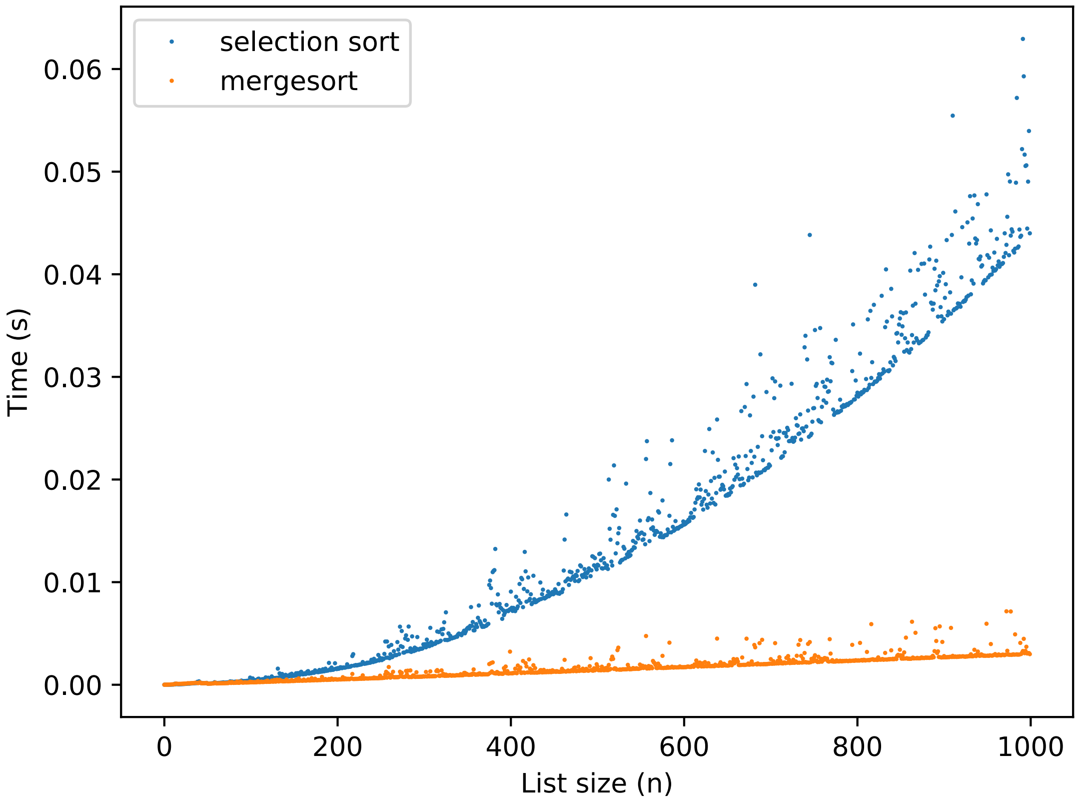
  $T(2n) / T(n) \approx 5\,(2n)^2 / 5\,n^2$

  ≈ 4

  Doubling n quadruples T(n)

  N.B. That is true for any coefficient of n^2 (not just 5)

n	T(n)
10	661
20	2301
30	4951
40	8601
...	...
1000	5015001
2000	20030001


selection sort
mergesort
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Time (s)
0
200
400
600
800
1000
List size (n)

Towards a formal definition of big O

Let $t(n)$ be a function that describes the time it takes for some algorithm on input size n.

We would like to express how $t(n)$ grows with n, as n becomes large i.e. *asymptotic* behavior.

Unlike with limits, we want to say that $t(n)$ grows like certain *simpler* functions such as

$log_2 n,\ \ n,\ \ n^2, \dots, 2^n$, etc.

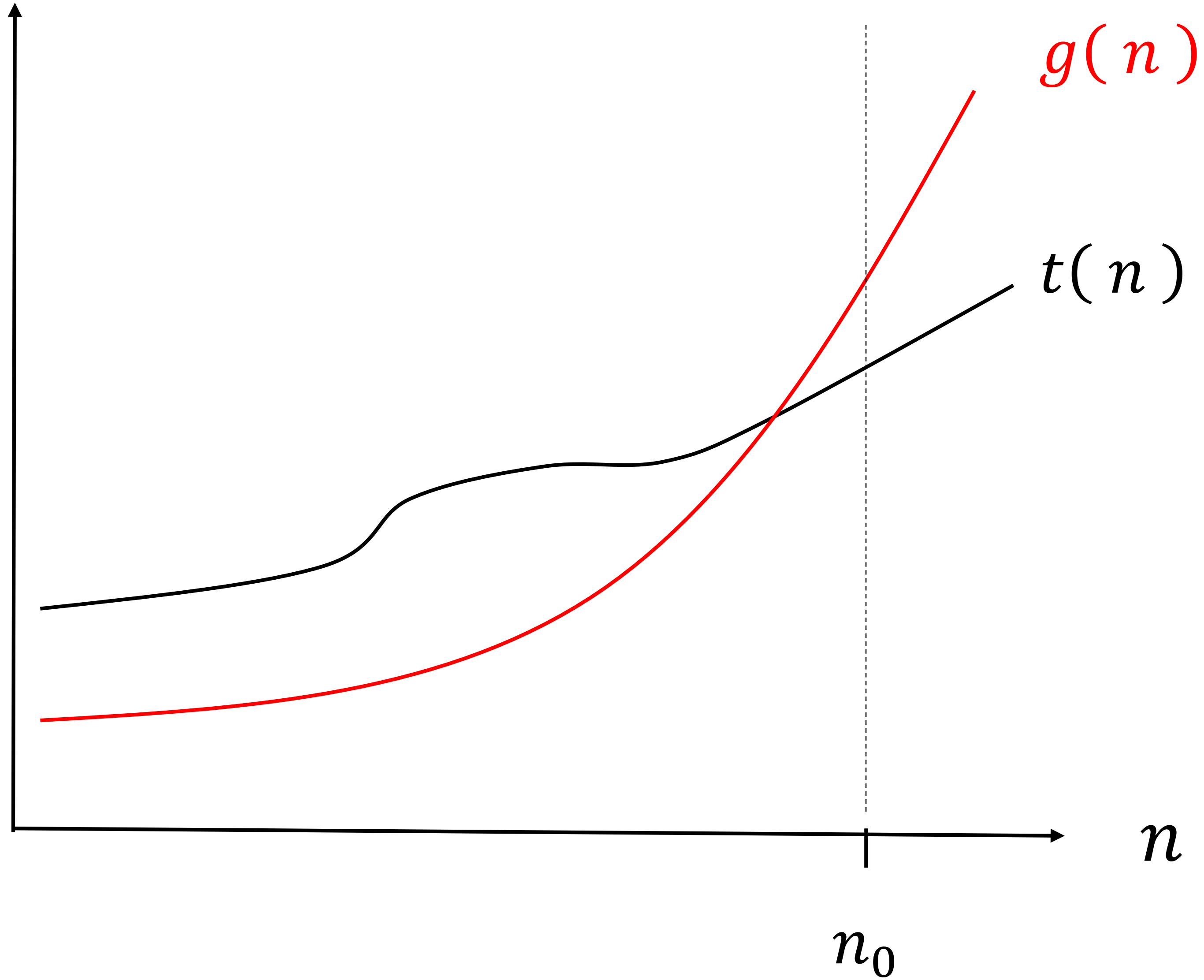
Preliminary Formal Definition

Let $t(n)$ and $g(n)$ be two functions, where $n \geq 0$.
We say $t(n)$ is *asymptotically bounded above* by $g(n)$
if there exists n_0 such that, for all $n \geq n_0$,

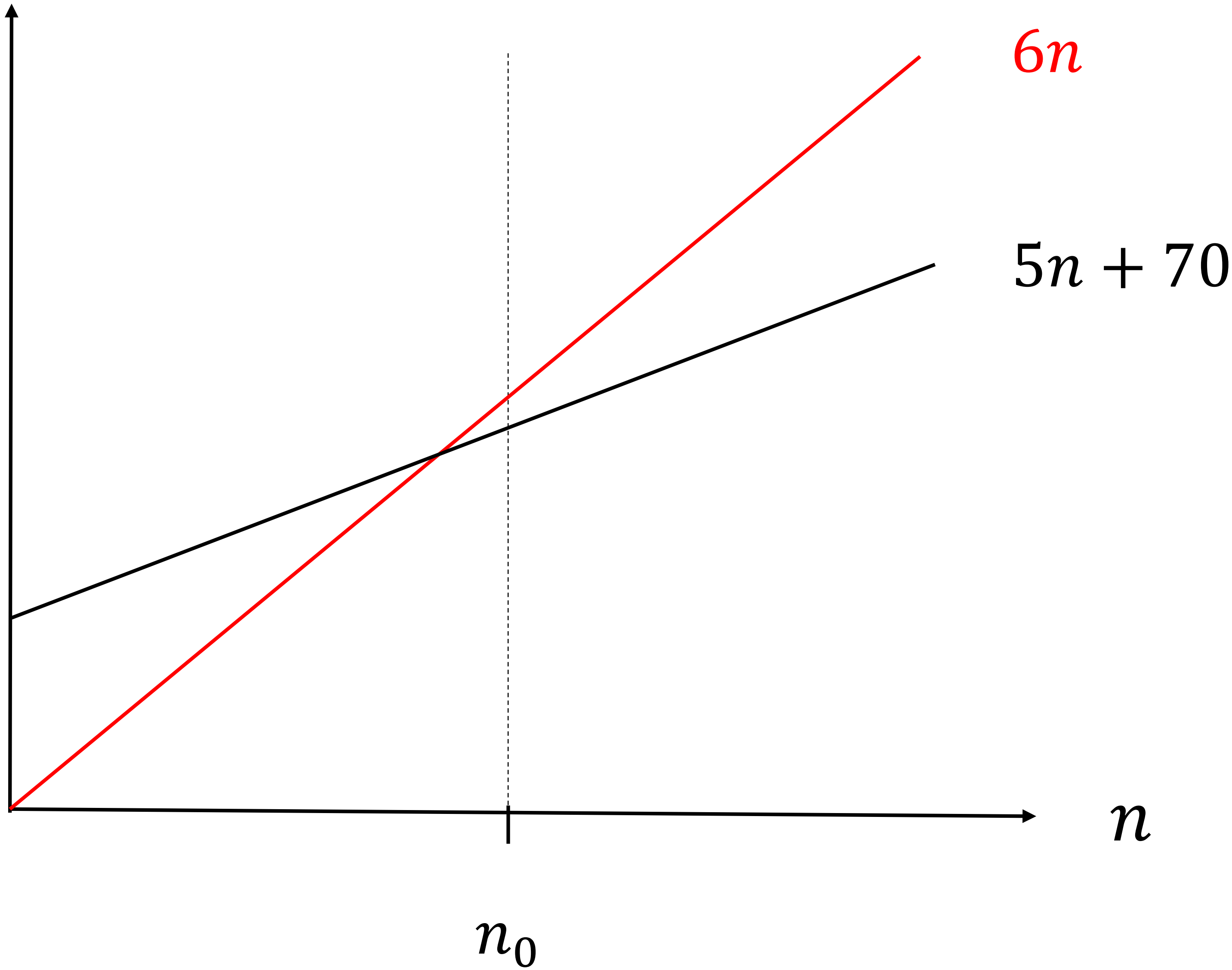
$$t(n) \leq g(n).$$

This is not yet a formal definition *of big O.*

for all $n_0 \geq n, \quad t(n) \leq g(n)$



Example



Claim: $5n + 70$ is *asymptotically bounded above* by $6n$.

Proof:

(State definition) We want to show there exists an n_0 such that, for all $n \geq n_0$, $5n + 70 \leq 6n$.

Claim: $5n + 70$ is asymptotically bounded above by $6n$.

Proof:

(State definition) We want to show there exists an n_0 such that, for all $n \geq n_0$, $5n + 70 \leq 6n$.

$$
\begin{aligned}
& 5n + 70 \leq 6n \\
\Longleftrightarrow \quad & 70 \leq n
\end{aligned}
$$

Symbol "$\Longleftrightarrow$" means "if and only if" i.e. logical equivalence

Claim: $5n + 70$ is asymptotically bounded above by $6n$.

Proof:

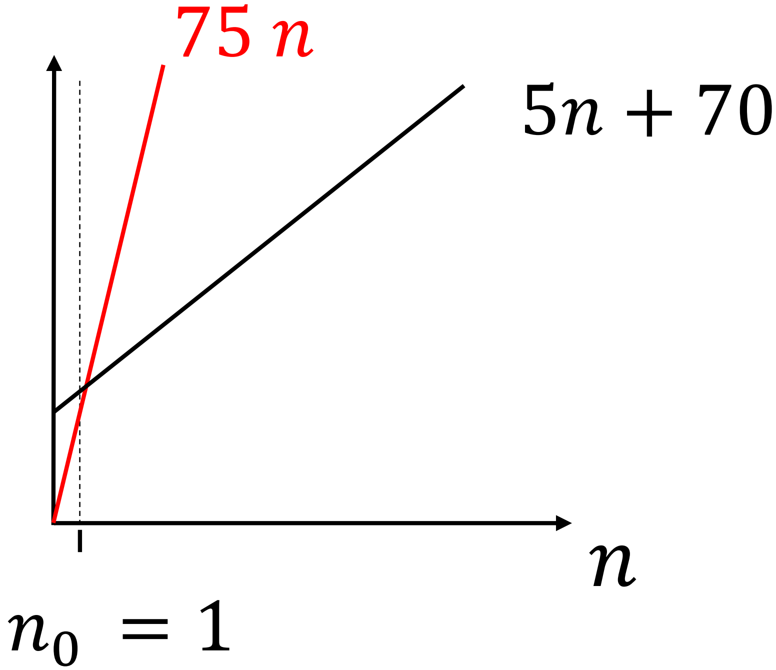
(State definition) We want to show there exists an n_0 such that, for all $n \geq n_0$, $5n + 70 \leq 6n$.

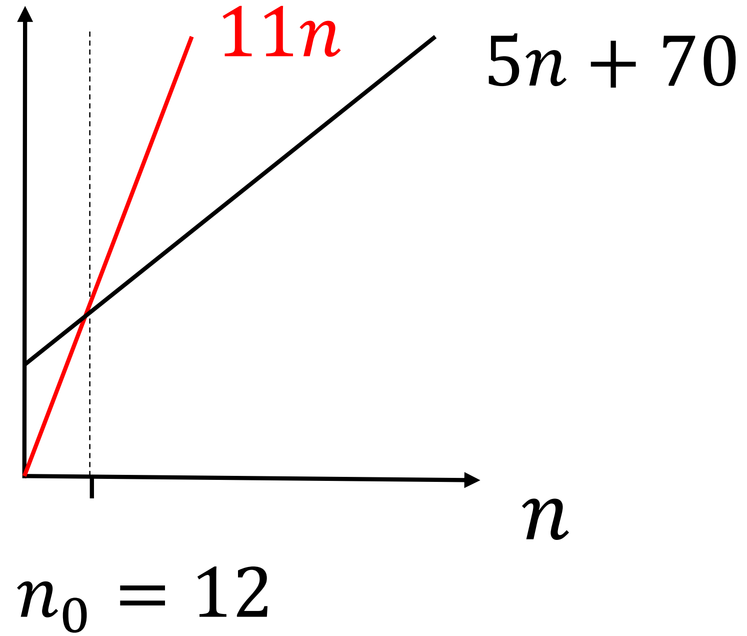
$$5n + 70 \leq 6n$$

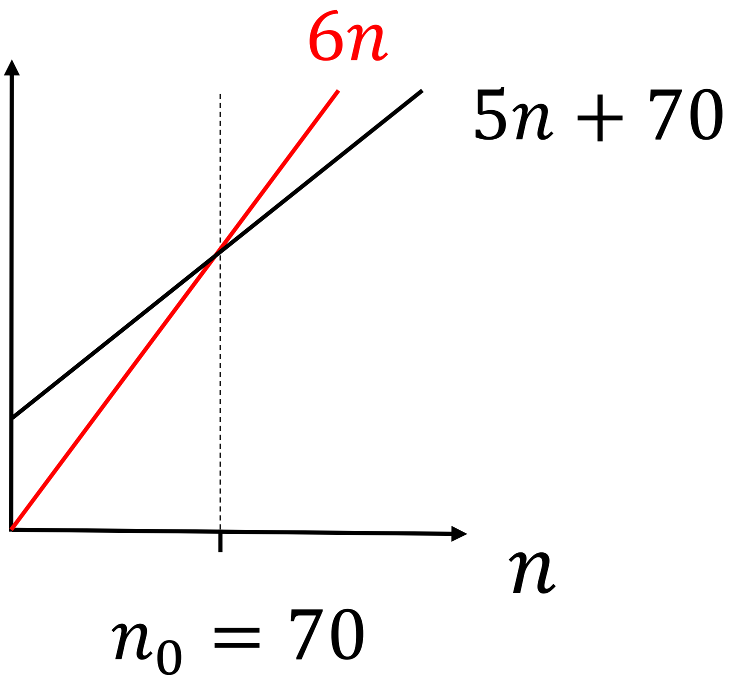
$$\Leftrightarrow \quad 70 \leq n$$

Thus, we can use $n_0 = 70$.

Symbol "$\Leftrightarrow$" means "if and only if" i.e. logical equivalence


$75\,n$
$5n + 70$
n
$n_0 = 1$


$11n$
$5n + 70$
n
$n_0 = 12$


$6n$
$5n + 70$
n
$n_0 = 70$

We would like to express formally how some function $t(n)$ grows with n, as n becomes large.

We would like to compare the function $t(n)$ with *simpler* functions , $g(n)$, such as $log_2 n,\ n,\ n^2, \dots, 2^n$, etc.

Formal Definition of Big O

Let $t(n)$ and $g(n)$ be two functions, where $n \geq 0$.

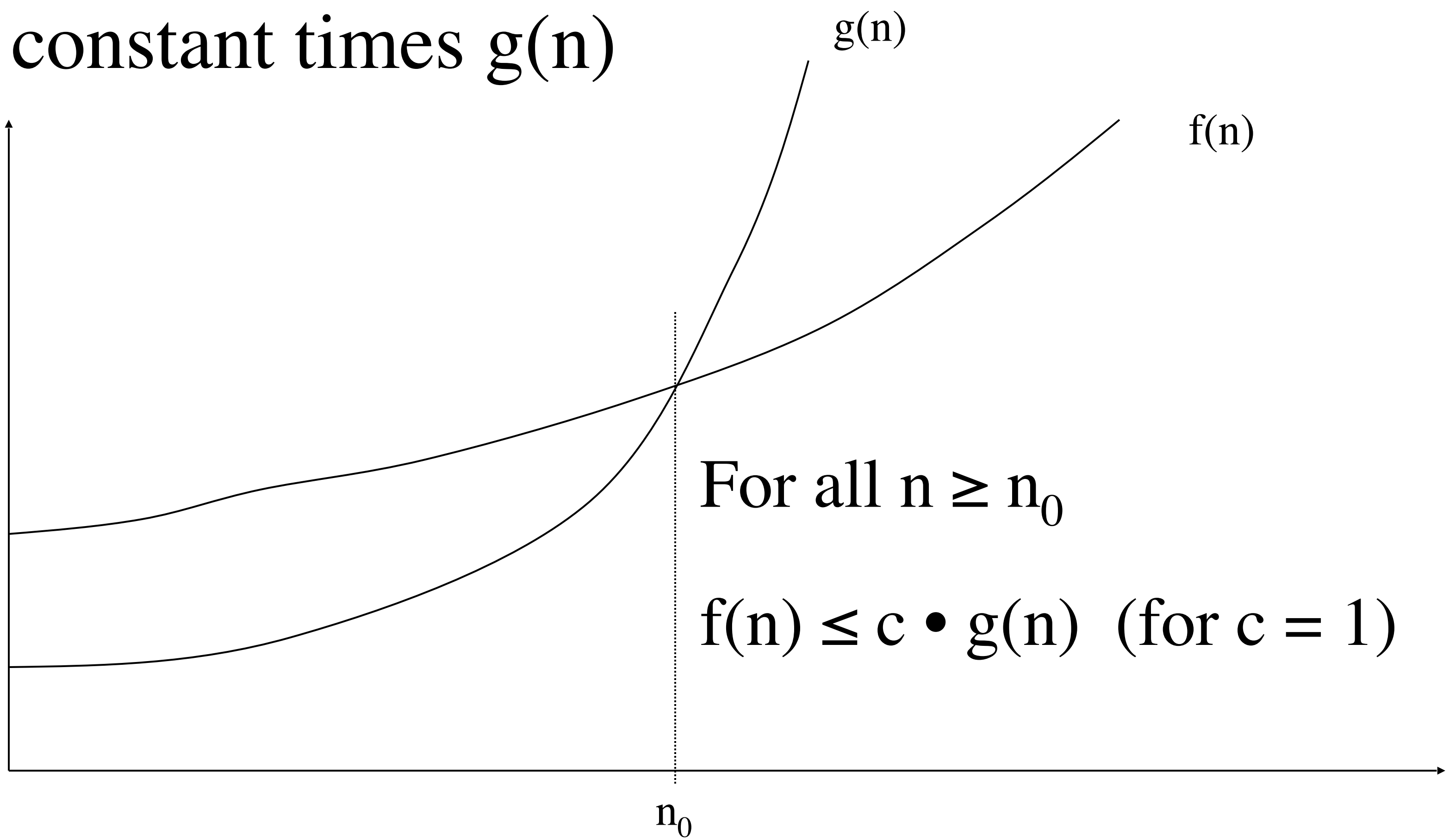
$g(n)$ will be a simple function, but this is not required in the definition.

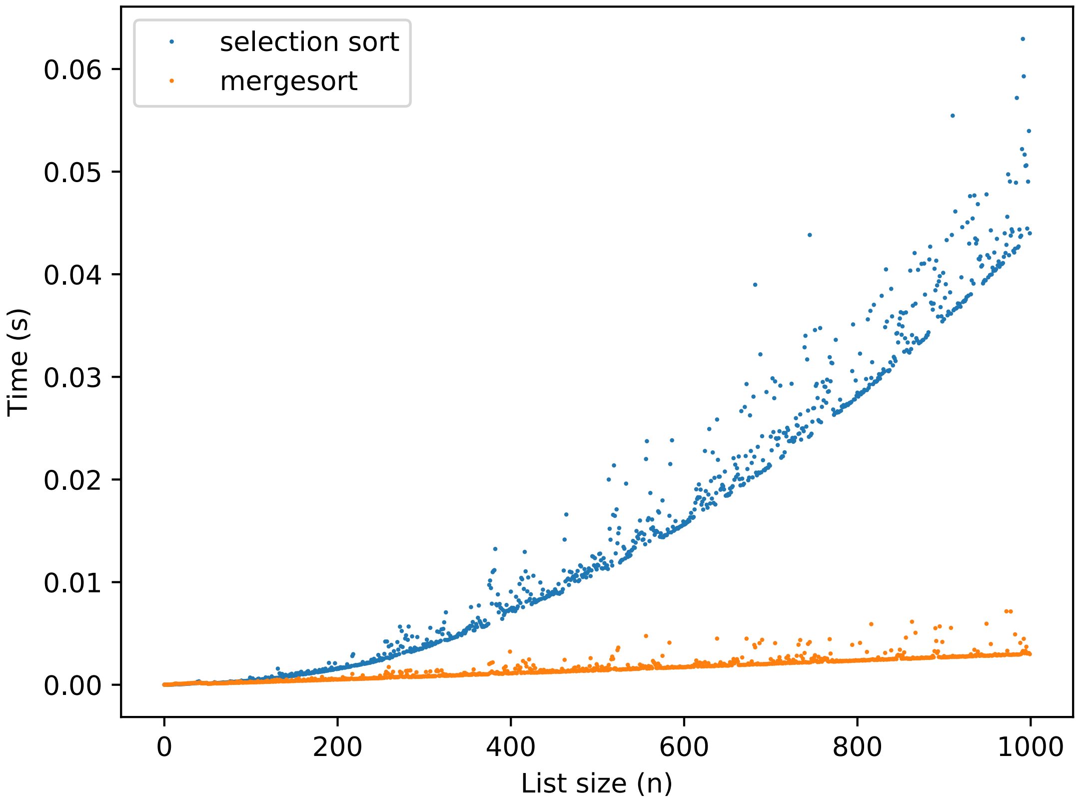
We say $t(n)$ is $O(\, g(n)\,)$ if there exist two positive constants n_0 and c such that, for all $n \geq n_0$,

$$t(\, n\,) \leq c \;\, g(\, n\,).$$

Intuition and visualization

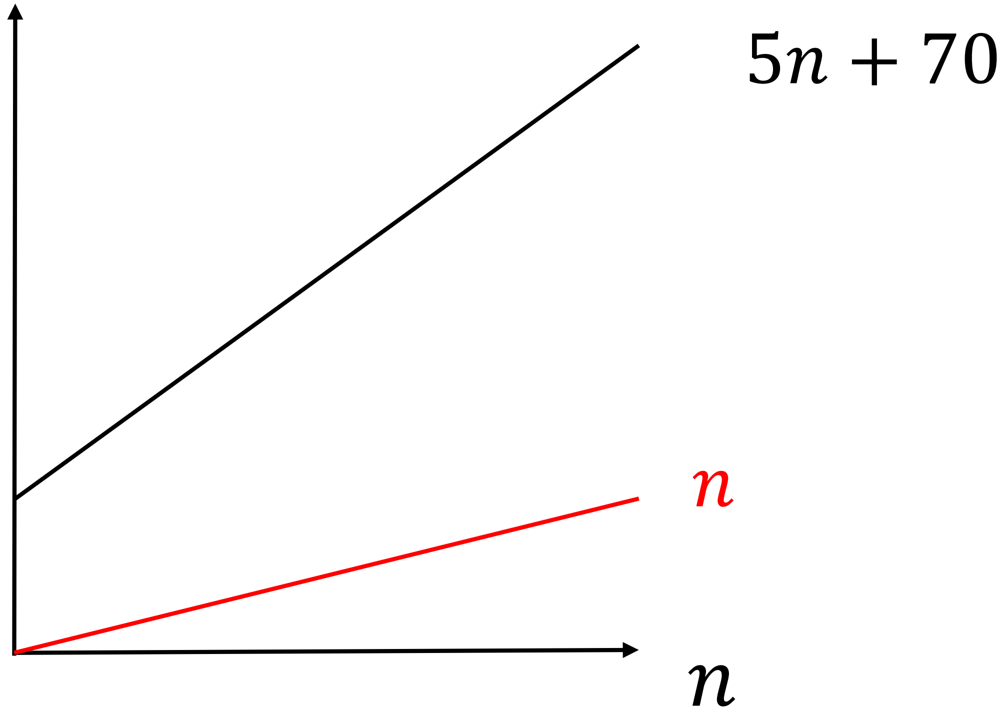
- "f(n) is O(g(n))" iff there exists a point n_0 beyond which f(n) is less than some fixed constant times g(n)





selection sort
mergesort
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Time (s)
0
200
400
600
800
1000
List size (n)

Claim: $5n + 70$ is $O(n)$.



Claim: $5\,n + 70$ is $O(\,n\,)$.

Proof 1:

$$5\,n + 70 \;\leq\; ?$$

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

$$t(\,n\,) \leq \; c\;\, g(\,n\,).$$

Claim: $5\,n + 70$ is $O(\,n\,)$.

Proof 1:

$$5\,n + 70 \;\leq\; 5\,n + 70n, \quad \text{if } n \geq 1$$

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

$$t(\,n\,) \leq \; c\;\, g(\,n\,).$$

Claim: $5\,n + 70$ is $O(\,n\,)$.

Proof 1:

$$5\,n + 70 \;\le\; 5\,n + 70n, \quad \text{if } n \ge 1$$

$$= \; 75\,n$$

So take $c = 75, \;\; n_0 = 1.$

Claim: $5\,n + 70$ is $O(\,n\,)$.

Proof 2:

$$5\,n + 70 \;\leq\; 5\,n + 6n, \quad \text{if}\, n \geq 12$$

Claim: $5\,n + 70$ is $O(\,n\,)$.

Proof 2:

$$5\,n + 70 \;\le\; 5\,n + 6n, \quad \text{if } n \ge 12$$

$$= \; 11\,n$$

So take $c = 11, \;\; n_0 = 12$.

Claim: $5\,n + 70$ is $O(\,n\,)$.

Proof 3:

$$5\,n + 70 \;\leq\; 5\,n + n, \quad n \geq 70$$

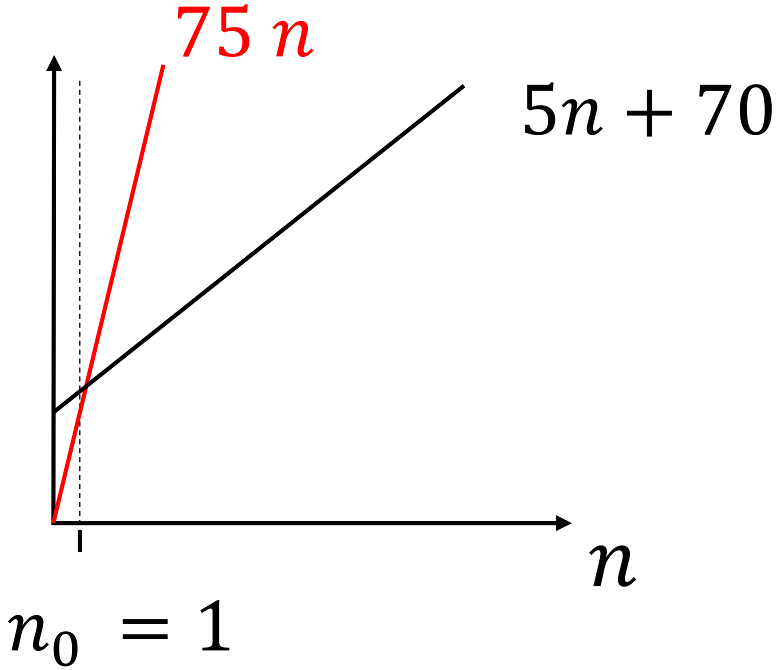
Claim: $5\,n + 70$ is $O(\,n\,)$.

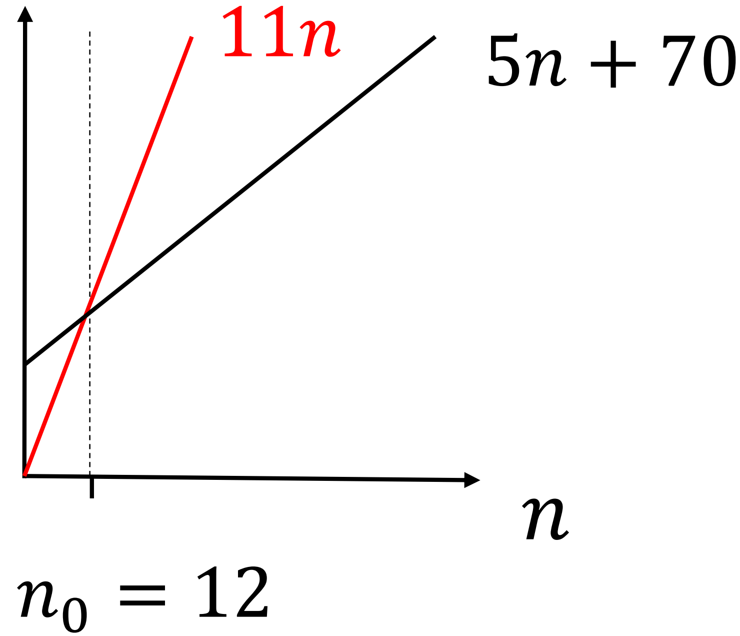
Proof 3:

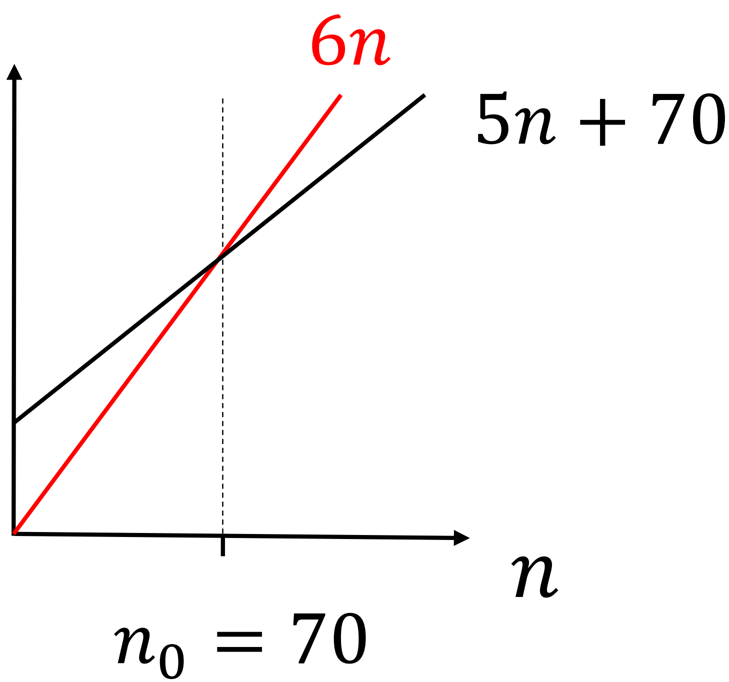
$$5\,n + 70 \;\le\; 5\,n + n, \quad n \ge 70$$

$$= \; 6\,n$$

So take $c = 6, \;\; n_0 = 70$.


$75\,n$
$5n + 70$
n
$n_0 = 1$


$11n$
$5n + 70$
n
$n_0 = 12$


$6n$
$5n + 70$
n
$n_0 = 70$

Claim: $5n + 70$ is $O(n)$.

Incorrect Proof:

$$\begin{aligned} 5n + 70 &\leq cn \\ 5n + 70n &\leq cn, \quad n \geq 1 \\ 75n &\leq cn \end{aligned}$$

$$\text{Thus,} \quad c > 75, \quad n_0 = 1$$

Q: Why is this incorrect ?

Claim: $5\,n + 70$ is $O(\,n\,)$.

Incorrect Proof:

$$\begin{aligned} 5n + 70 &\leq cn \\ 5n + 70n &\leq cn, \quad n \geq 1 \\ 75n &\leq cn \end{aligned}$$

$$\text{Thus,} \quad c > 75, \quad n_0 = 1$$

Q: Why is this incorrect ? A: Because we don’t know which line follows logically from which.

Claim: $8\,n^2 - 17n + 46$ is $O(n^2)$.

Proof (1):

$$8\,n^2 - 17n + 46$$

Claim: $8\,n^2 - 17n + 46$ is $O(n^2)$.

Proof (1):

$$8\,n^2 - 17n + 46$$

$$\leq\; 8\,n^2 \qquad + 46\,n^2, \qquad n \geq 1$$

Claim: $8\,n^2 - 17n + 46$ is $O(n^2)$.

Proof (1):

$$8\,n^2 - 17n + 46$$

$$\leq\ 8\,n^2 \qquad + 46\,n^2, \qquad n \geq\ 1$$

$$\leq\ 54\,n^2$$

Claim: $8\,n^2 - 17n + 46$ is $O(n^2)$.

Proof (1):

$$8\,n^2 - 17n + 46$$

$$\leq 8\,n^2 \qquad + 46\,n^2, \qquad n \geq 1$$

$$\leq 54\,n^2$$

So take $c = 54$, $n_0 = 1$.

Claim: $8\,n^2\ - 17n + 46$ is $O(n^2)$.

Proof (2):

$$8\,n^2\ - 17n + 46$$

Claim: $8\,n^2 \ - 17n + 46$ is $O(n^2)$.

Proof (2):

$$8\,n^2 \ - 17n + 46$$

$$\leq \ 8\,n^2 \ , \qquad\qquad n \geq \ 3$$

So take $c = 8,\ \ n_0 = 3.$

What does $O(1)$ mean?

We say $t(n)$ is $O(1)$, if there exist two positive constants n_0 and c such that, for all $n \geq n_0$,

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

So it just means that $t(n)$ is bounded.

Never write $O(3n)$, $O(5\, log_2 n)$, etc.

Instead, write $O(n)$, $O(log_2 n)$, etc.

Why? The point of big O notation is to *avoid dealing with constant factors*.

It is still *technically* correct to write the above. We just don't do it.

Considerations

- n_0 and c are not *uniquely* defined. For a given c and n_0 that satisfy $\mathcal{O}()$ we can increase one or both to again satisfy the definition. There is no "better" choice of constants.
- However, we generally want a "tight" upper bound, so smaller $\mathcal{O}()$ relations give us more information. (This is not the same as smaller c or n_0).
- e.g. any $f(n)$ that is $\mathcal{O}(n)$ is also $\mathcal{O}(n^2)$ and $\mathcal{O}(2^n)$. But $\mathcal{O}(n)$ is more informative.

Big O as a set

- When we show that a $t(n)$ *is* $\mathcal{O}(g(n)$ you will sometimes see this written as $g(n) = \mathcal{O}(g(n))$
- This is not strictly true given the standard definition of $=$ so instead we think of $\mathcal{O}(g(n))$ as a set of functions bounded by $g(n)$.
- We can then say that $t(n)$ is a member of this set as such: $t(n) \in \mathcal{O}(g(n))$

Example

Show that $n!$ is $\mathcal{O}((n+2)!)$

$$
\begin{aligned}
n! &\leq c(n+2)! \\
n! &\leq c(n+2)(n+1) \quad \text{divide by} n! \qquad (1) \\
1 &\leq c(n+2)(n+1)
\end{aligned}
$$

We choose $n_0 = 1$ and $c = 1$

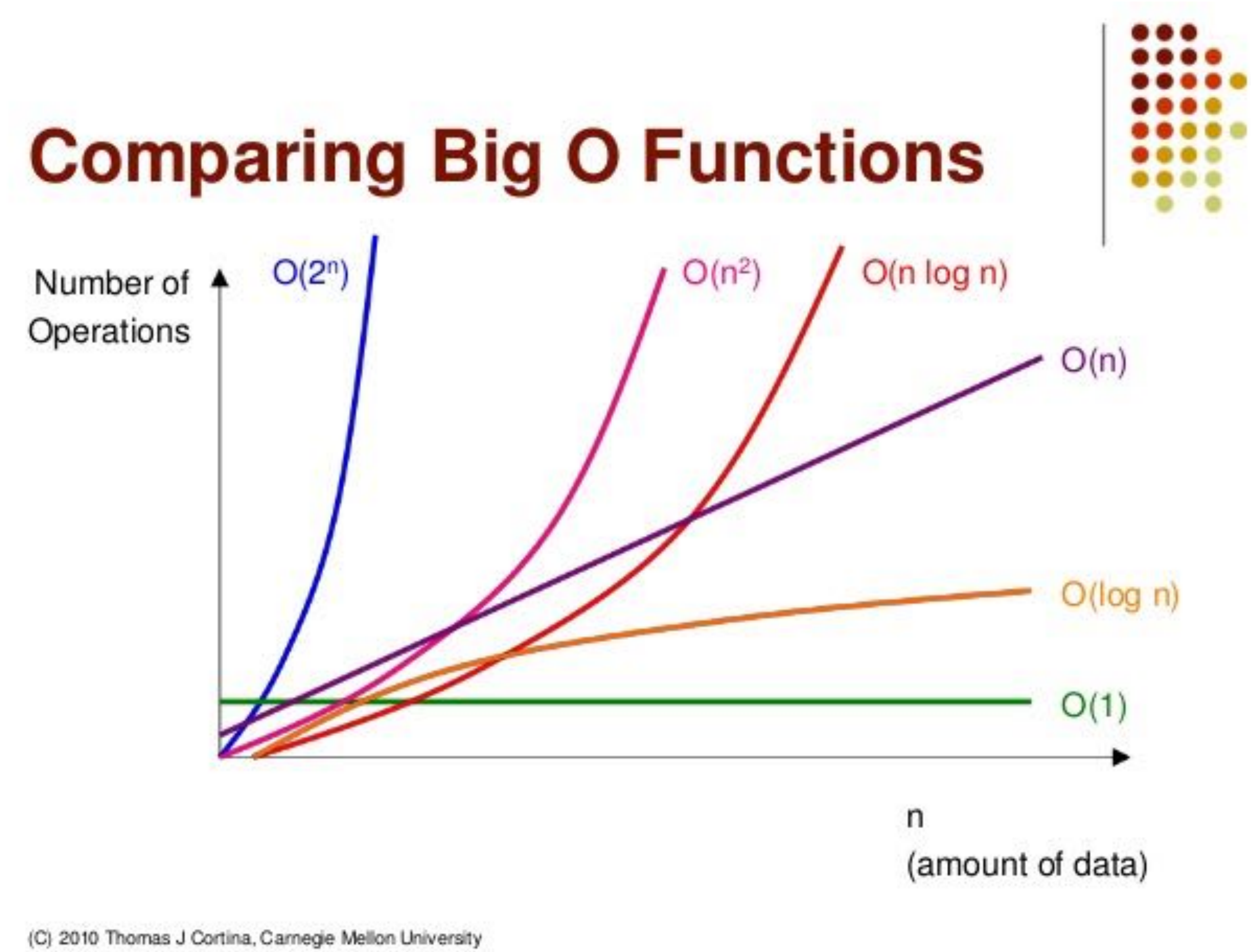
Example

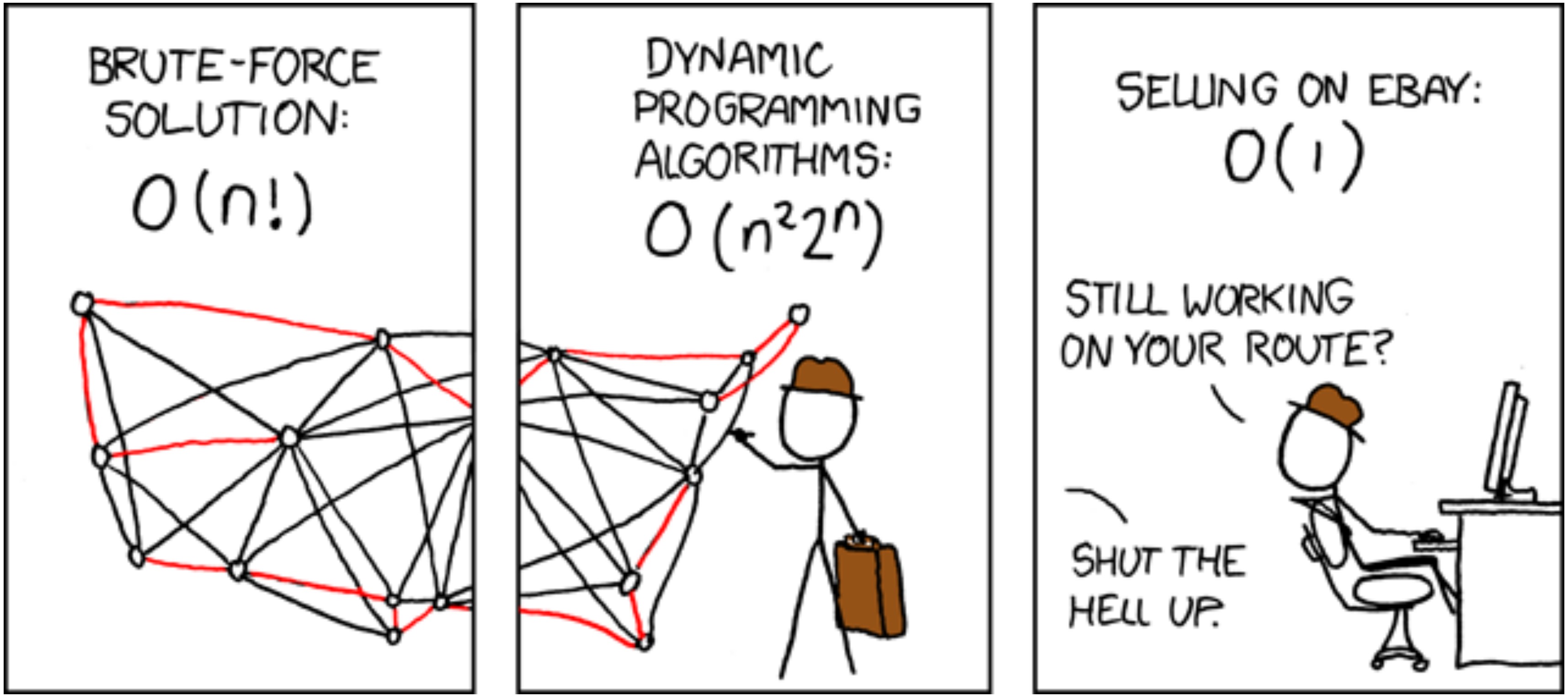
Show that $(n+2)!$ is $\mathcal{O}(\ n!\ )$
If this is true, I can write;

$$\begin{aligned} (n+2)! &\leq n! \qquad \text{for all } n \geq n_0 \\ (n+2)(n+1)n! &\leq cn! \\ (n+2)(n+1) &\leq c \end{aligned} \tag{2}$$

However, this is clearly not the case **for all** $n \geq n_0$ since c is constant (and $c < \infty$) and so it cannot be larger than an infinite number of increasing n

Complexity Classes




BRUTE-FORCE SOLUTION:
O(n!)
DYNAMIC PROGRAMMING ALGORITHMS:
O(n²2ⁿ)
SELLING ON EBAY:
O(1)
STILL WORKING ON YOUR ROUTE?
SHUT THE HELL UP.