Describing:

- what
- how
- why
- how fast

comparing running times is not easy...

Everything fast if $n=2$

so focus on large n...

& ignore constant factors

→ asymptotic notation

Big-oh:

function $f(n) : \mathbb{N} \to \mathbb{R}_{>0}$

time on input size n

$g(n) : \mathbb{N} \to \mathbb{R}_{>0}$

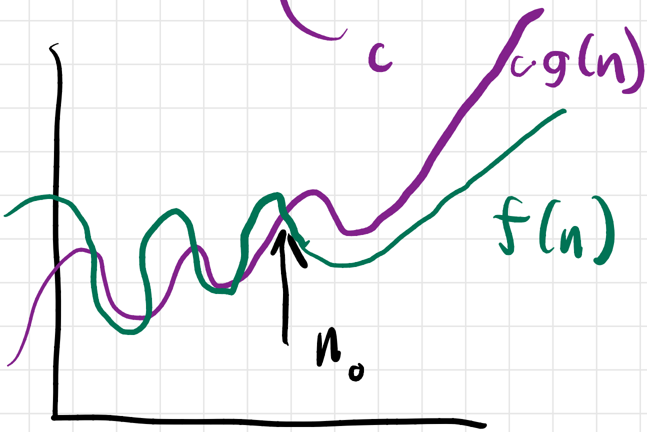
Define the <u>set</u>

$O(g(n)) = \{ f(n)$: there exist pos. constants c + n_0 s.t. $0 \le f(n) \le c\,g(n)$ for all $n \ge n_0 \}$

$O(g(n))$ is all functions that are $\leq$ a constant multiple of $g(n)$ when n is "large".

c

$\geq n_0$



big-Oh is like $\leq$

$256n \in O(n)$ $(n_0 = 0 \text{ \& } c = 256)$

$n \in O(n^2)$ $(n_0 = 256 \text{ \& } c = 1)$

claim a running time in $O(n^2 \log n)$ when it is in $O(n^2)$,

still "correct"

Lower bound: big-Omega:

$\Omega(g(n)) = \{f(n) :$ there exist pos. constants c & n_0 s.t. $0 \le c g(n) \le f(n)$ for all $n \ge n_0\}$

Exact: Theta

$$\Theta(g(n)) = O(g(n)) \wedge \Omega(g(n))$$

$f(n)$ is "asymptotically tight" to $g(n)$

$o(g(n))$: informally:

functions in $O(g(n))$

<u>but not</u> in $\Theta(g(n))$

Merge sort runs in $O(n \log n)$.

Is there one in $o(n \log n)$

"NO"

$\omega(g(n))$: tight lower bound

"Algebra"

Transitivity: If $f(n) \in O(g(n))$ & $g(n) \in O(h(n))$,
then $f(n) \in O(h(n))$

true for $O, \Omega, \Theta, o, \omega$

Reflexivity: $f(n) \in O(f(n))$

$f(n) \in \Omega(f(n))$

$f(n) \in \Theta(f(n))$

Symmetries: $f(n) \in \Theta(g(n))$ iff $g(n) \in \Theta(f(n))$

$f(n) \in O(g(n))$ iff $g(n) \in \Omega(f(n))$

$f(n) \in o(g(n))$ iff
$g(n) \in \omega(f(n))$

if and only
iff ($\Leftrightarrow$)

"Algebra": Suppose $f_1(n) \in \Theta(g_1(n))$

$+ f_2(n) \in \Theta(g_2(n))$...

$c \cdot f_1(n) \in \Theta(f_1(n))$ for constant $c > 0$

$f_1(n) + f_2(n) \in \Theta(g_1(n) + g_2(n))$

$f_1(n) + f_2(n) \in \Theta(\max\{g_1(n), g_2(n)\})$

$$f_1(n) \cdot f_2(n) \in \Theta(g_1(n) \cdot g_2(n))$$

If you have asymptotic notation in an algebraic expression, ...

then for any choices of functions for each "written" set on the left hand side (LHS) there should exist choices on RHS to make the expression true.

$f(n) = O(g(n))$

true iff $f(n) \in O(g(n))$

$5n^2 + 1000n = 5n^2 + O(n)$

then someone writes

$5n^2 + O(n) \subseteq$ [something]

FIBONACCIMULTIPLY($X[0..m-1], Y[0..n-1]$):

```
hold ← 0
for k ← 0 to n + m − 1
    for all i and j such that i + j = k
        hold ← hold + X[i] · Y[j]
    Z[k] ← hold mod 10
    hold ← ⌊hold/10⌋
return Z[0..m + n − 1]
```

What is the running time if..
each line takes $O(1)$ time &
$m = n$?

$k \le 2n$, so inner loop runs
at most $2n+1 = O(n)$ times.
Inner loop takes $O(n) \cdot O(1) = O(n)$.
Whole outer loop iteration

takes $O(n) + O(1) + O(1)$
$= O(n)$ time

whole alg takes

$O(1) + 2n \cdot O(n) + O(1) =$
$O(n^2)$ time

i.e., its a doubly nested for loop over $O(n)$ values

Favorite Functions

$f(n)$ is polynomially bounded if $f(n) = O(n^k)$ for some constant k

main first goal for times

$n^{k_1} = o(n^{k_2})$ when $k_1 < k_2$, so aim for small exponents exponential : a^n

constant

$n^k = o(a^n)$ for any constant k

$\forall$ constant $a > 1$.

$a^n = o(c^n)$ for constants

$c > a > 1$.

polylogarithmically bounded

$$(\log_b n)^\ell$$

$(\log_b n)^\ell = o(n^k)$ for constants $b > 1$, $\ell, k > 0$.

Ex: $n^2 = \omega(n^{1.9}(\log_5 n)^{100000})$

$\log n := \log_{10} n$

$\lg n := \log_2 n$

$\ln n : \log_e n$

Euler's number ≈ 2.72

$\log_b^{\ell} n : (\log_b n)^{\ell}$

$a^{\log_b n} = n^{\log_b a}$

write like this

$$\log_b n = \frac{\log n}{\log b}$$

so, if b is constant,

$$\log_b n = \Theta(\log n)$$

$$n^{\log_3 5} = o(n^{\lg 5})$$

oh no!