CS 173 Lecture 16a: Asymptotic Analysis, A Motivation

Running time of an algorithm
can usually be parametrized by the input size

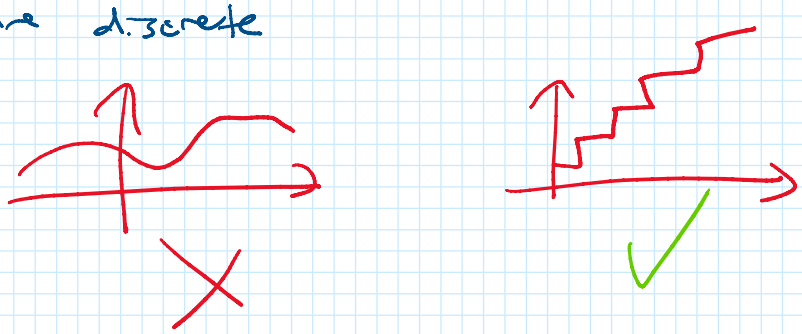
Some caveats

(a) for small inputs, there is noise/overhead

(b) multiplicative constants that depend on implementation/hardware

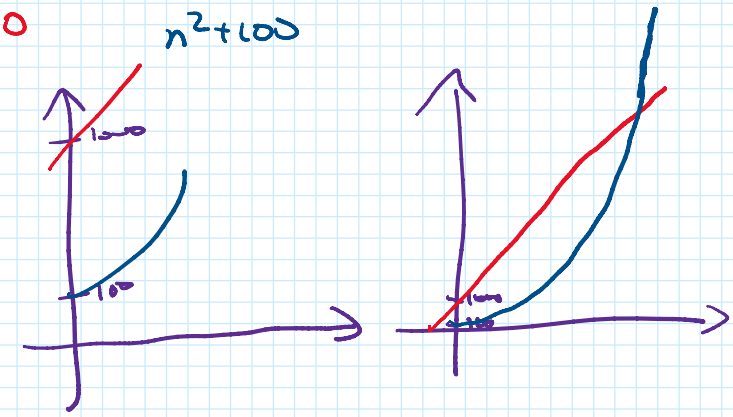
still reasonable to say, e.g.
doubling input size → 4x running time
(for sufficiently large inputs / all hardware).

(c) functions describing performance w.r.t. input size are discrete



Dealing w/ (a): consider limits.

e.g. $n+1000$ n^2+100



for large inputs, n^2+100 is much bigger than $n+1000$.

via limits: $\lim_{n\to\infty} \frac{n+1000}{n^2+100} = 0$

Define: $f(n) \ll g(n)$ iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. ("asymptotically smaller")

good for "much slower" / "much faster"

bad for "similar".

An attempt: $n+1000$ vs $n+100$

$$\lim_{n\to\infty} \frac{n+1000}{n+100} = 1.$$

Does not account for (b).

More limits: $f(n) \approx g(n)$ iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ for some $c \in \mathbb{R}$. ("asymptotically similar")

e.g. alg $n+1000$ on fast machine

$2(n+1000)$ on slower machine

$$\lim_{n\to\infty} \frac{n+1000}{2n+2000} = \frac{1}{2}$$

What about (c)?

Big-O (and Big-Θ (Theta))

Motivation: $\ll$ & $\approx$ for more "continuous" functions

Goal: $\lesssim$, $\lesssim$, $\approx$ for discrete functions

$\lesssim$: stay mostly. Big-O

$\lesssim$: Big-Θ.

Stay mostly. | Big O | Big-Θ.

$f(n) = O(g(n))$ iff $\exists\ k, c \in \mathbb{Z}$, such that

$\forall\ n \geq k,\quad f(n) \leq c\,g(n)$.

for "nice" fn's similar to

$\lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c$.

$f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ & $g(n) = O(f(n))$.

terrible notation

use them for historical reasons