CS1800

11/14 - Tues.

Admin

- exam#2! Fri 9AM-6pm 2hour window
- HW7 due Fri 11/17 (70% volume)
- this week's recitation - exam prep

Agenda

1. Growth of Functions
2. Complexity classes
3. Assigning a complexity class

1. Growth of Functions

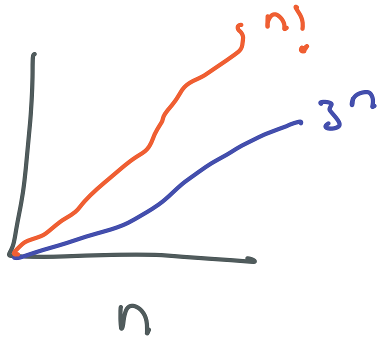
last week... we proved that $\boxed{3^n \leq n!}$

- for $n > 6$ this was true

$\boxed{3^n}$ grows more slowly than $\boxed{n!}$

- not focused on a specific value of n

- as n grows arbitrarily large, $n!$ gets bigger faster



(fake chart)

Context: real world!

$f(n) = 3^n$

n = input

3^n = output

$g(n) = n!$

n = input

$n!$ = output

n = size of an input

\# elements in set, graph, tree, list/array...

$f(n), g(n)$ = # steps an algorithm needs to solve a problem

$f(n)$ grows more slowly than $g(n)$ $\boxed{f(n) \leq g(n)}$

- as size of input grows arbitrarily large, $f(n)$ needs fewer steps than $g(n)$
- $f(n)$ is better (if all other things are equal)

ex

n	$f(n) = n^2$		$g(n) = 2n$
10	100		20
100	10,000	gap gets	200
1,000	1,000,000	wider	2,000
⋮		↓	
1,000,000	1,000,000,000,000		2,000,000

$g(n)$ grows more slowly than $f(n)$

Why this matter SO much in CS

- $f(n)$ # steps an algo needs on input of size n
- n size of input
- Choosing a slower-growing algorithm has more impact than prog. language, processor, memory space

n growing arbitrarily large ...

ex google search — # web sites

10 years ago: 600 m

now: 1.14 billion

ex tweets per day

10 years ago: 2.5 m

last year: 500 m

ex # insta users:

10 years ago: 15 m

now: > 1 billion

ex self-driving car

takes pics constantly

~ 4 Terabytes per day (1 TB = 1,000 GB)

huge "real life" implications to f(n) grows slower or faster than g(n)

$\boxed{2.}$ Complexity classes

If you have 100 algorithms,

then you'd have 100 different $f(n) = $ ~~~~

$3n+2$

$3n+3$

$3n+4$

$4n+1$

↳ basically the same!

We want to put run-times in $\boxed{\text{buckets}}$

- for a given algorithm, it has run-time $f(n)$
- put it into nearest $\boxed{\text{complexity class}}$

Basic complexity classes

$g(n) = k$ (k is constant)		constant time	$O(1)$
$g(n) = \lg n$		log time	$O(\lg n)$
$g(n) = n$	good :)	linear time	$O(n)$
$g(n) = n \cdot \lg n$		sorting time	$O(n \lg n)$
$g(n) = n^k$ (k is constant)		polynomial time	$O(n^k)$
- - - - - - - - - -			
$g(n) = k^n$ (k is constant)		exponential time	$O(k^n)$
$g(n) = n!$	bad :(intractable	factorial time	$O(n!)$

millenium prize P vs NP

- problems for which the only known solution is intractable
- come up w/ faster solution
- or, prove it can't be done

$\left.\begin{array}{l} \text{come up w/ faster solution} \\ \text{or, prove it can't be done} \end{array}\right\}$ \$1M

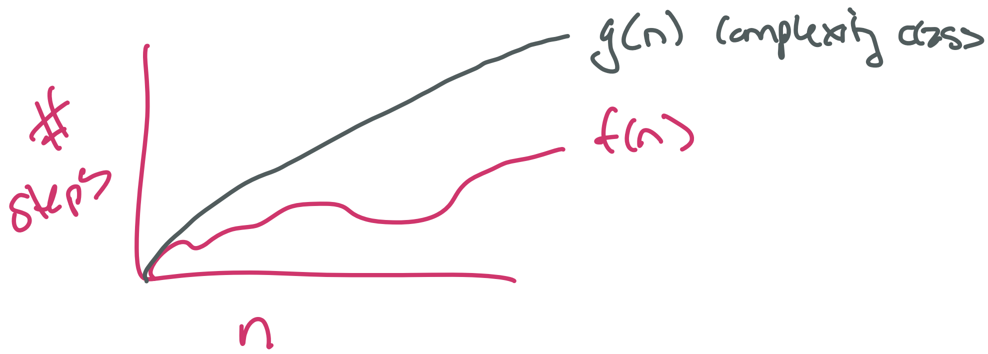
3. Assigning a Complexity Class

$f(n)$ = # steps an algo needs on input of size n

which of the 7 classes do you belong to?

↳ upper bound big-oh

$O(\sim\sim)$



How to put a function in a complexity class:

easy way :)

$f(n) = 3n + 2$

- drop coefficients
- drop lower-order terms

$f(n) = \cancel{3}n + \cancel{2}$

$= O(n)$

$f(n) = \cancel{3}n + \cancel{3}$

formal way

Defn of big-oh

$f(n)$ is $O(g(n))$ if $\exists$ positive c, k such that

$f(n) \leq c \cdot g(n)$ for $\forall n \geq k$

To formally show $f(n) = O(g(n))$ need to find c, k

$= O(n)$

$f(n) = \cancel{103}n + \cancel{18}$

$= O(n)$

$f(n) = \cancel{3}n^2 + \cancel{10n} + \cancel{\lg n}$

$= O(n^2)$

$f(n) = \cancel{2} + \cancel{2}n^2$

$= O(n^2)$

easy way ⟶ formal way

want to show:

$$\underbrace{2 + 2n^2}_{f(n)} \le c \cdot \underbrace{n^2}_{g(n)} \quad \forall n \ge k$$

the way to find c, k ... try candidates

n (possible k)	$f(n)$	$g(n)$	$\lceil f(n)/g(n) \rceil$ (possible c)
1	4	1	$\lceil 4/1 \rceil = 4$
→ 2	10	4	$\lceil 10/4 \rceil = 3$
3	20	9	$\lceil 20/9 \rceil = 3$
4	34	16	$\lceil 34/16 \rceil = 3$

Try: $c = 3, k = 2$ $\quad f(n) = 2n^2 + 2 \quad g(n) = n^2$

want: $f(n) \le c \cdot g(n) \quad \forall n \ge k$

$2n^2 + 2 \le 3 \cdot n^2 \quad \forall n \ge 2$

$n = 2$ $\quad 2 \cdot 4 + 2 = 10 \quad 3 \cdot 4 = 12$

$f(n) \leq c \cdot g(n)$ ✓

$$2 + 2n^2 \leq n + 2n^2$$
$$\leq n^2 + 2n^2$$
$$= 3n^2$$

$\rightarrow 2 + 2n^2 \leq 3 \cdot n^2$

$\boxed{\text{done!}}$