CS3000
5/6 - Tues.

Admin

- recitation 1 today!
- please say your name :)
- lecture questions
- please come in back door if late!

Agenda

1. Run-time + efficiency
2. Complexity Classes
3. Linear Search analysis

0. Math Reminder

↳ logarithms

$\log_2 b \quad x^? = b$

ex $\log_3 81 \quad 3^? = 81 \quad 4$

ex $\log_2 8 = \lg 8 \quad 3$

practice

	answers
$\log_4 64$	3
$\lg 64$	6
$\lg \lg 256$	$\lg 8 = 3$
$64 \lg 64$	$64 \cdot 6 = 384$
$\lg (2^{125})$	125

$\lg x \quad 2^? = x$

$\lg (2^{125}) \quad 2^? = 2^{125}$

1. Run-Time + Efficiency

"good" algorithm — correct
— efficient — space
— time

Q: How long will my code take to run?

- size of input
- code structure
- processing requirements
- complexity reqs
- termination conditions
- layers / flow of control
- repeating the input, subproblems
- hardware
- memory safety
- location of input, output
- length of code
- recursion / iteration
- programming language

Hardware — processor speed
size of memory
vs. Algorithm Design

1975
- processor 14MHz
- RAM 16KB

1995
- proc 144MHz (x10)
- RAM 2MB (x125)

2020
- proc 1.4GHz (x100)
- RAM 6GB (x375000)

meanwhile... amount of data!
- #websites 2008: 172 mil
now: 1.1 bil
- #tweets 2010: 25 mil
today: 500 mil
- language models: Brown corpus 1 mil words, 1 MB
chatGPT training data 300 bil words
570 GB

2. Complexity Classes

↳ same Q, new version: How many steps does my algo take on an input of size n?

independent of:
- prog lang
- hardware
- processing
- flow of control

↳ algorithm design only

Algorithm:
- n = size of array A
- step = roughly one line of pseudocode
- as n grows arbitrarily large

$T(n)$ = # steps on input of size n

which belong together?

ex $T(n) = 4n^2 + \frac{1}{2}n + 18$

$T'(n) = 3n^2 + \frac{1}{2}n + 19$

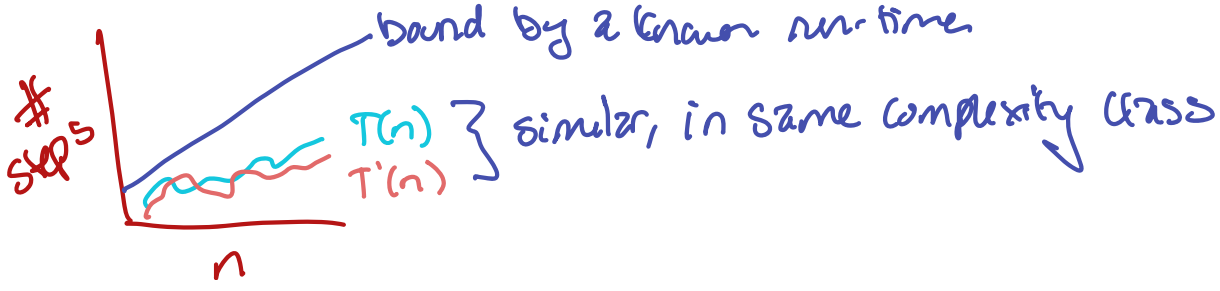
$n=1$ $T(n) = 22.5$ $T'(n) = 22.5$

$n=10$ $T(n) = 423$ $T'(n) = 324$

$n=100$ $T(n) = 40068$ $T'(n) = 30069$

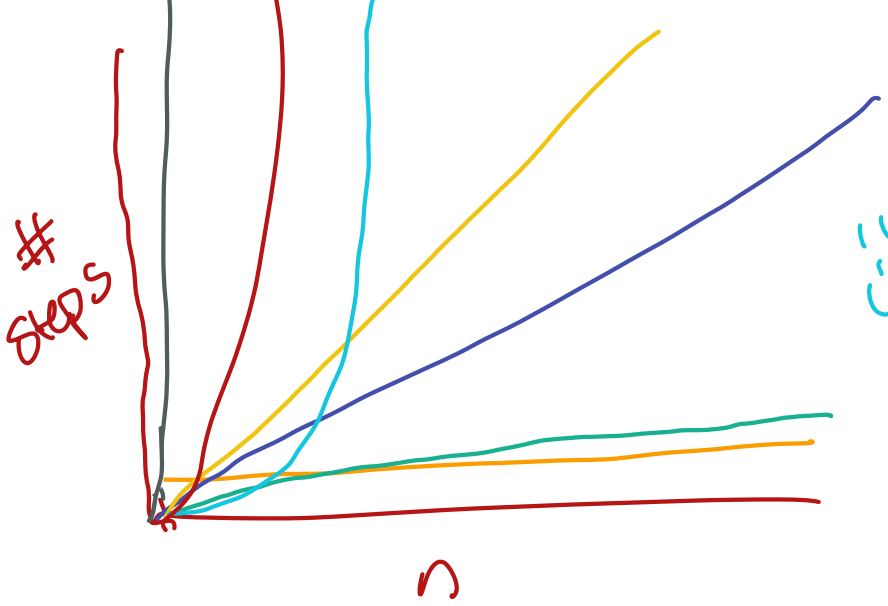
value of n has biggest impact!

put algorithms that are similar in the same bucket so we can compare to each other



Every $T(n)$ for an algo is assigned to a known complexity class

big-oh notation



$O(1)$ constant time

$O(\lg n)$ log time

$O(n)$ linear time

$O(n \lg n)$ sorting band

$O(n^k)$ k is const. (poly)

ex: $O(n^2)$ quadratic

$O(k^n)$ k is const (exp)

$O(n!)$ factorial

↳ intractable !!

12:50

3. Algorithm Analysis

 → pseudocode → $T(n)$ → complexity class

(ex) tractable vs. intractable

10,000 operations per second

$O(n^2)$

n	# seconds
100	1
1,000	100
5,000	2,500 (41 min)
10,000	10,000 (166 min)
15,000	22,500 (375 min)

$O(n!)$

n	# seconds	$O(n^n)$
4	.0024	.0025
8	4	27 min
10	362	11.5 days
12	47,900 (13 hours)	28.2 years
14	8.7 mil (100 days)	35.2 millenia
16	2.1 bil (66 years)	584 kilo

```
LINEARSEARCH(A, n, key)
1  for i = 1 to n
2      if A[i] == key
3          return i
4  return NIL
```

Array A, length n, key to look for
- returns position if key found
- returns NIL if not found

Best case:
- key is first!

Worst case:
- not there

T(n) for linear search

Assumptions:
- worst case (not found)
- cost of line i is c_i
- compute # times each line runs

LINEARSEARCH(A, n, key)	cost	# times	
1 for i = 1 to n	c_1	check loop condition:	$n+1$
2 if A[i] == key	c_2	body of loop runs:	n
3 return i	c_3	0	
4 return NIL	c_4	1	

- A function returns 0 or 1 thing
- only a max of 1 return statement executes

$$T(n) = c_1 \cdot (n+1) + c_2 \cdot n + c_3 \cdot 0 + c_4 \cdot 1$$

$$= c_1 \cdot n + c_1 + c_2 \cdot n + c_4$$

$$= (c_1 + c_2) n + (c_1 + c_4)$$

$$T(n) = \blacksquare n + \blacksquare$$