Scientific Computing

Wed, Feb 11

Announcements

* Homework 2 due this Friday, 11:59pm
  pdf + zip file on D2L

Don't forget to keep track of and cite any external resources you use - friends, websites, AI, etc.

Two kinds of things to cite:
(1) A resource helped me learn about a topic
(2) A resource wrote this line of code.
Be specific.

* Also, written explanations should be your own words.

Office Hours:

Mon, 9:30-10:30
Fri, 2:00-3:00

Cudahy 307

There is a theorem called The Master Theorem that tells you how to convert a recurrence into a formula.

See Wikipedia page

In this case, it tells us:

$$T(n) = O(n \log(n)).$$

⇝ Jupyter Notebook Sorting demo

For an $O(n^2)$ algorithm

Double the input size: $n \rightarrow 2n$

run time: $n^2 \rightarrow (2n)^2$

$= 4 \cdot n^2$

A few overall notes:

* We are splitting the <u>input</u> in half, not the search space.

* These algorithms are not obvious! Many times there isn't one.

* If there is, it's usually faster than brute force - the recombining function is always the hard part!

Ex #2 – The simplest divide-and-conquer algo. is "binary search".

* Guess the number

50 – lower

25 – lower

13 – higher

19 – higher

22 – lower

20 – ✓

$2^6 = 64 < 100$

$\boxed{2^7 = 128 > 100}$

Ex #2 – The simplest divide-and-conquer algo. is "binary search".

* Guess the number

2^{30}

In binary search, you just throw away half of your input each time.

$$\log_a(b) = \frac{\log_c(b)}{\log_c(a)}$$

Recurrence: $T(n) = T(n/2) + 1$

Solution: $T(n) = O(\log(n))$

Ex #3 - Counting Inversions (medium)

Consider a list of distinct #s.

$L = 3 \quad 19 \quad -7 \quad 2 \quad 1 \quad 6 \quad 0 \quad -10$

An <u>inversion</u> is a pair (L_i, L_j) where $i < j$ but $L_i > L_j$ (an out-of-order pair).

The list L has: $5 + 6 + 1 + 3 + 2 + 2 + 1 = 20$

Goal: compute the # of inversions in a list of n elements

Obvious algorithm: Check all pairs, $O(n^2)$.

Divide-and-conquer:

$L = \underbrace{3 \quad 19 \quad -7 \quad 2}\ \underbrace{1 \quad 6 \quad 0 \quad -10}$

recursively count inversions — 4

recursively count inversions — 5

So, 9 inversions <u>within</u> a half. How many <u>between</u> the lists? That would be a blue element that is larger than a red one.

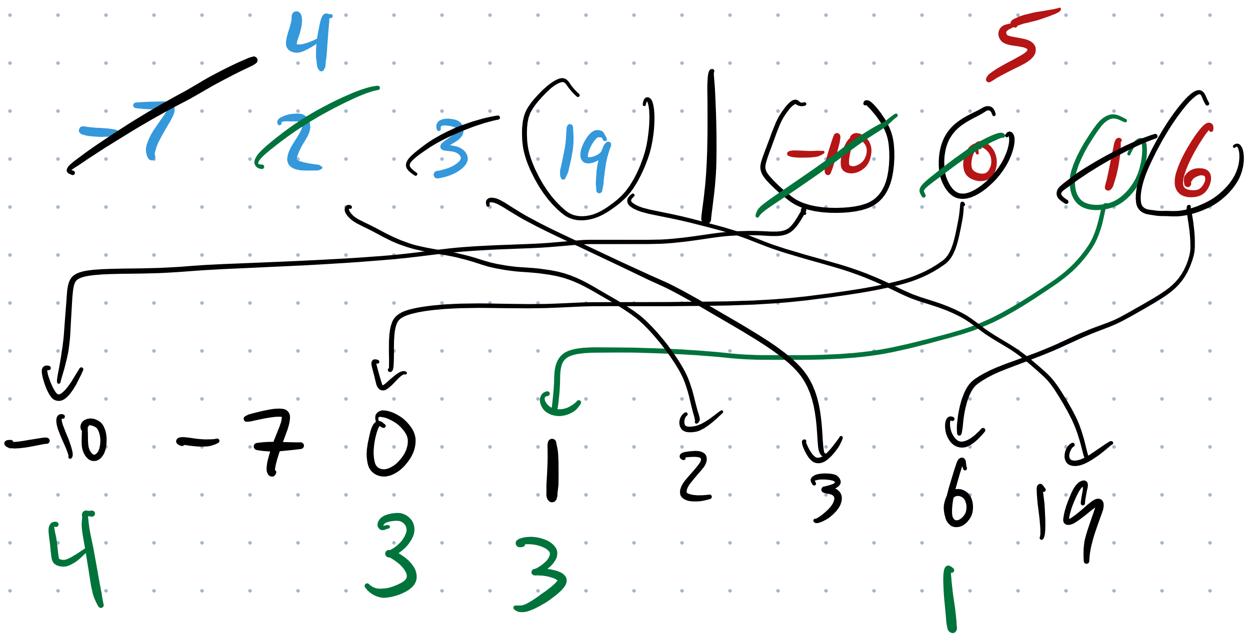
Right now, to do that, we'd have to go through all (blue, red) pairs, which takes $n^2/4$ time (still $O(n^2)$, not good!)

Here's the trick: While we're counting inversions, we'll *also* sort the lists, which we know takes $O(n\log(n))$ time.

$$L = \underbrace{3 \quad 19 \quad -7 \quad 2}_{4} \;\; \underbrace{1 \quad 6 \quad 0 \quad -10}_{5}$$

$$-7 \quad 2 \quad 3 \quad 19 \;\Big|\; -10 \quad 0 \quad 1 \quad 6$$

Now we recombine the lists just like the mergesort, and when do we detect an inversion? Anytime we take from the red list, there is an inversion for everything left in the blue list.



$$11 + 9 = 20$$

Time: $T(n) = 2T\left(\frac{n}{2}\right) + 2n$

$\rightsquigarrow T(n) = O(n\log(n))$