Scientific Computing

Fri, Feb 6

Announcements

* Homework 2 due Fri, Feb 13, 11:59pm
  pdf + zip file on D2L
  start early!!
  covers Greedy Algorithms

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

* Today - we will allocate half of class time for me troubleshooting your python installations - bring your laptop

Office Hours:

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

Cudahy 307

Topic 7 - Divide and Conquer

"Divide and Conquer" is an algorithmic paradigm that is roughly

1) Split the <u>input</u> in half
2) Solve the problem on each half separately (recursively)
3) Combine your two answers into one big answer.

Classic Example: Sorting a list (easy)

* You can phrase this as a constraint satisfaction problem.
* Input: n numbers
* Search space: All orderings of n things. These are called permutations, and the # of them is $n(n-1)(n-2)(n-3)\cdots 3\cdot 2\cdot 1 = n!$
* Goal: Find the rearrangement that puts things in the right order.

* Obvious optimal algorithm: (greedy-ish)
  - Pick the smallest thing, put it first
  - Pick the next smallest thing, put it second, etc.

How many steps does this take?
  - Finding the k^{th} smallest thing takes n steps (have to search the whole list)
  - We have to do this n times.

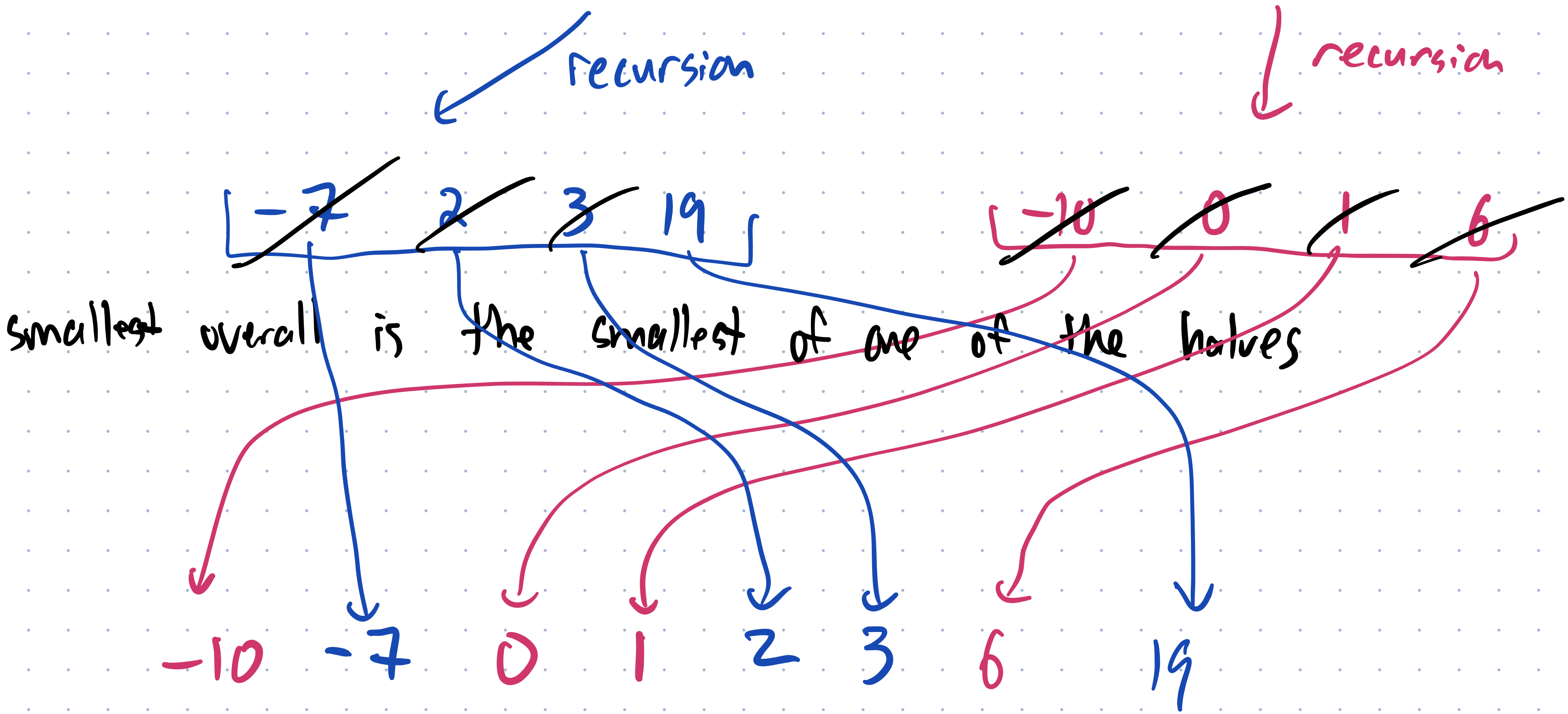
Thus, $O(n^2)$. Fine for a few thousand things, but not more.

1) Split your input elements in half (or close enough)

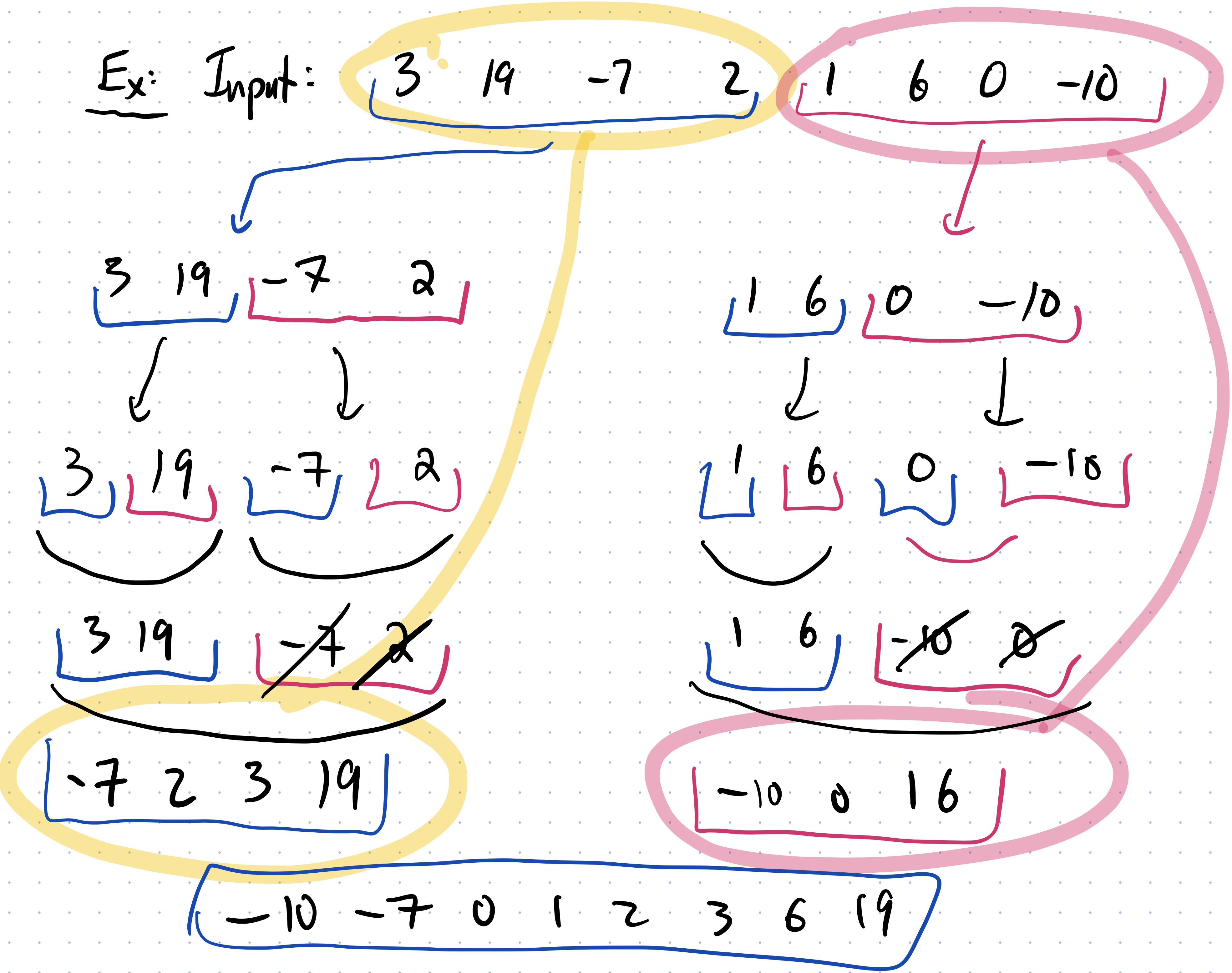
2) Sort each half (recursively, by dividing and conquering)

3) Combine the two sorted halves into one big sorted list

Ex: Input: 3 19 -7 2 1 6 0 -10



O(n) comparisons to recombine


Ex: Input: 3 19 -7 2 1 6 0 -10
3 19 -7 2
1 6 0 -10
3 19 -7 2
1 6 0 -10
3 19 -7 2
1 6 -10 0
-7 2 3 19
-10 0 1 6
-10 -7 0 1 2 3 6 19

Pseudocode

function merge_sort(Q):   Q = list of #s

if |Q| ≤ 1:
    return Q

If the input list has a single element, it's already properly sorted so return it

L = left half of Q
R = right half of Q
L = merge_sort(L)
R = merge_sort(R)

(at this point, we get to assume L and R are individually sorted)

recombine

new_list = []
while |L| + |R| > 0:
    take L[0] or R[0], whichever is smaller,
        remove it, and add to new_list
return new_list

What's the runtime? Harder, because it's recursive. What we can do is find a recurrence for the runtime. $a_n = a_{n-1} + a_{n-2}$

Suppose the runtime is $T(n)$ when the input has size n.

Steps:

Apply to left half	$T(n/2)$
Apply to right half	$T(n/2)$
Merge	n

Recurrence: $T(n) = 2T(\frac{n}{2}) + n$

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