Monday, Feb 20, 2023
Lecture # 15
MSSC 6000

Announcements

* HW 2 due Wednesday, Feb 22, 11:59pm
* Normal Office Hours today, 1pm-2pm CU307
* Midterm Exam → Wed, March 8 or Fri, March 10 ?

Topic 6 - Divide and Conquer

Sorting a list (easy)

Ex #2 - The simplest divide-and-conquer algorithm is called "binary search".

I'm thinking of a # between 1 and 100. You have 7 guesses, and after each guess I'll tell you higher or lower.

#1) 50 - lower

#2) 25 - lower

#3) 10 - higher 11-24

#4) 16 - lower 11, 12, 13, 14, 15

#5) 13 - higher

#6) 14 - higher

#7) 15

It's always possible to win in 7 guesses.

$2^6 = 64 < 100$ $2^7 = 128 > 100$

#1-1000 how many guesses? 10 $(2^{10} = 1024)$

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

$T(n) = 1 \cdot T(\frac{n}{2}) + 1$

Solution: $T(n) = O(\log(n))$ $O(\log_2(n))$

these are the same in O-notation

$$\frac{\log_{10}(n)}{\log_{10}(2)} = \log_2(n)$$

This is how python answers if you ask if something is in a set OR if you look up a value in a dictionary. ③

Items in sets must be hashable (immutable)

Python keeps a sorted list of the hashes of elements in your set.

When you ask if something is in the set, it does a binary search on the list of hashes.

$2 \in [1,2,3]$ $2 \in \{1,2,3\}$

$O(n)$ $O(\log(n))$

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 inversion is a pair of entries (L_i, L_j), ~~$i \neq j$~~, where $i < j$ but $L_i > L_j$.

(an out-of-order pair)

Ex: (3, 6) is not an inversion

(3, 2) is

This list has 5 + 6 + 1 + 3 + 2 + 2 + 1 = 20 inversions

Goal: Compute the # of inversions in a list with n elements

Obvious Algorithm: check all pairs one-by-one

$O(n^2)$

Divide + Conquer:

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

recursively count inversions   recursively count inversions

4 inv.   5 inv.

We are successfully counting 9 inv. within the individual halves, but missing the 11 with one entry in one half and one in the other.

One way to do this that doesn't help is to loop over all (blue, red) pairs and check them. # of these is:

$$\left(\frac{n}{2}\right)\cdot\left(\frac{n}{2}\right) = \frac{n^2}{4} = O(n^2)$$

Here's the trick: While we're counting inversions, we're <u>also</u> going to sort the list. (sorting takes $O(n \cdot \log(n))$ but we're going to do it <u>in</u> the recursion, like merge sort.

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

$$-7 \quad 2 \quad 13 \quad 19 \qquad\qquad -10 \quad 0 \quad 1 \quad 6$$

Now we recombine the lists just like merge sort and when do we detect a (blue, red) inversion? Any time we take from the red list, that entry makes an inversion with every entry left in the blue list.

6

~~-7~~ ~~2~~ ~~3~~ ~~19~~ ~~-10~~ ~~0~~ ~~1~~ ~~6~~

-10 -7 0 1 2 3 6 19

+4 +3 +3 +1 = 11

(blue, red) inversions

$4 + 5 + 11 = 20$

Recurrence: $T(n) = 2T\left(\frac{n}{2}\right) + 2n$ ← rough sense

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

much faster than $O(n^2)$.

I'll send you Closest Points.

$O(n^2) \to O(n \cdot \log(n))$

Other famous divide-and-conquer examples

Integer Multiplication

Input: Two n-digit #s x and y

Output: $x \cdot y$

Simple algorithm:

$$
\begin{array}{r}
172 \\
\times\ 424 \\
\hline
688 \\
3440 \\
68800 \\
\hline
72928
\end{array}
$$

$O(n^2)$

D+C: Recurrence: $T(n) = 3 \cdot T\left(\frac{n}{2}\right) + n$

$\Rightarrow T(n) = O\left(n^{\log_2(3)}\right) = O\left(n^{1.59\ldots}\right)$

Matrix Multiplication: (wikipedia page)

Naive algo: $O(n^3)$ two $n \times n$ matrices

$\vdots$

$O(n^{2.7\ldots})$

$\vdots$

$O(n^{2.\ldots\ldots})$