

Feb. 14, 2018

①

P.TOTALI

DIVIDE & CONQUER PARADIGM

-Solving Recurrences

- A) DIVIDE the problem into sub problems
- B) Solve the sub problems recursively
or combine
- C) Merge the solutions to subproblems
to solve the larger ~~problem~~.

★ A canonical example here is the so-called Merge Sort which uses this technique to ~~sort~~ sort n numbers using $O(n \log n)$ comparisons; each comparison is of the form,
is $a_i < a_j$? where a_1, a_2, \dots, a_n are the given numbers.

★ Often step C is the key to the whole algorithm! [Sometimes, Step A might also require some creativity.] Let's first recall / see how Step C can be done for Mergesort.

Illustration of "Merge": sorted lists. (2)

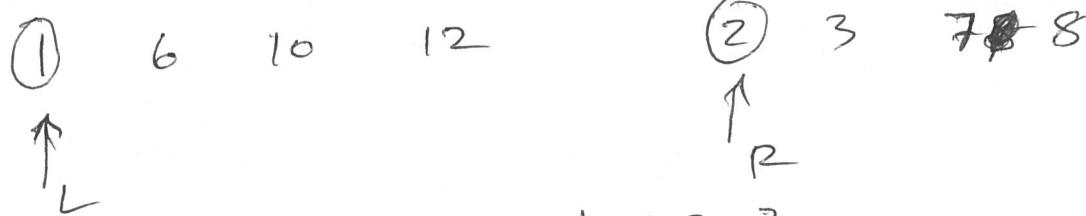
Example: combining two sorted lists, with distinct elements (say).



- Imagine a left pointer L and a right pointer R which will move to the right, based on the outcome of the comparisons.
- They both start at the left most on their lists.
- At each step if we compare the elements at the pointers L & R and the smaller element is added to the ~~final list~~ final combined list and the corresponding pointer moves one step to the right. (The other pointer stays put.)
- So at each step one element is added to the ~~list~~ combined list and at the very end, the last remaining elements of one ~~remaining~~ list are added to the list. [See →]

③

Start



Compare: $1 < 2 ?$

Combined list



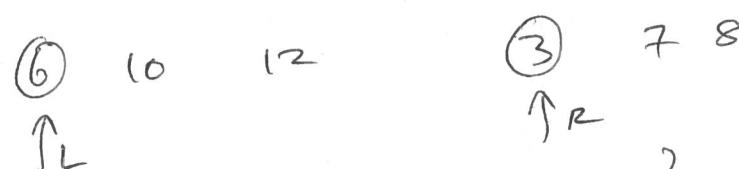
Step 2:



Compare $6 < 2 ?$



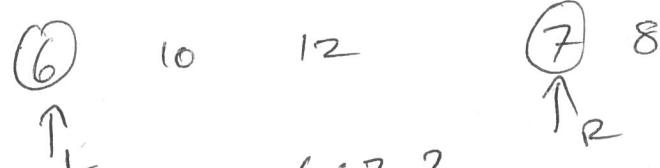
Step 3:



Compare $6 < 3 ?$



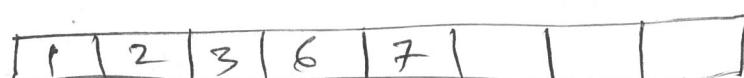
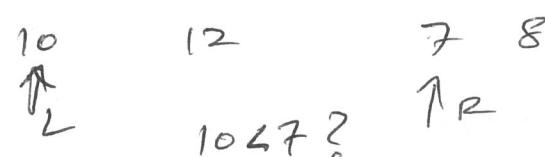
Step 4:



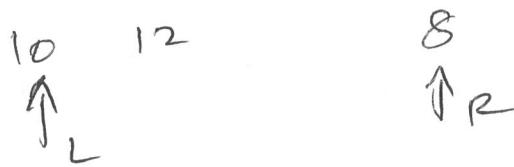
6 < 7 ?



Step 5:



(4)

Step 6compare $10 < 8$?

Combined List

1	2	3	6	7	8		
---	---	---	---	---	---	--	--

Now that ~~one~~ list is empty, we can simply attach the remaining list to the combined list!

Final:

1	2	3	6	7	8	10	12
---	---	---	---	---	---	----	----

.

Qn. How many comparisons total?

Ans.

- At each step one of the pointers moves one ~~place~~, after one comparison
- We stop in the worst case, when all but one element is left to be placed on the combined list.
- So if the total #elements is n , then the answer is $n-1$, since we place one element each time a pointer moves.

13.

Merge Sort; Analysis

(5)

The algorithm is described in Pages 50-51
of the textbook by Dasgupta et al.

Mergesort ($a[1..n]$)

Input : array of n numbers : $a[1..n]$
Output : sorted array ~~a~~

If $n > 1$:
return merge (merge sort ($a[1..n/2]$),
merge sort ($a[\frac{n}{2}+1..n]$))

else:
return array a .

(* The key is the "merge" operation explained before*

Let $T(n) = \# \text{ comparisons done by}$
 $\text{Merge Sort. Then we have,}$

$$T(2) = 1, \text{ and}$$

$$T(n) \leq 2 T\left(\frac{n}{2}\right) + (n-1).$$

Let us simplify by assuming $n = 2^k$, $k \geq 1$.
and replacing $(n-1)$ by n , so that
we get a cleaner recurrence -

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

$$= 2 \left\{ 2 T\left(\frac{n}{4}\right) + \frac{n}{2} \right\} + n$$

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

(6)

$$T(n) = 2^2 \left[2 T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} \right] + 2n$$

$$= 2^3 \left[T\left(\frac{n}{2^3}\right) \right] + \underbrace{n + 2n}_{= 3n}$$

... repeating it, we get

$$T(n) = 2^{k-1} \left[T\left(\frac{n}{2^{k-1}}\right) \right] + (k-1)n.$$

Recall $n = 2^k$, so that $T\left(\frac{n}{2^{k-1}}\right) = T(2) = 1$.

$$\text{So } T(n) = 2^{k-1} + (k-1)n.$$

what is k ? $2^k = n \Rightarrow k = \log_2 n$.

$$\begin{aligned} \text{So } T(n) &= \frac{n}{2} + (\log_2 n - 1)n = n \log_2 n - \frac{n}{2} \\ &= O(n \log n). \end{aligned}$$

□.

§ More recurrences:

(I) First recall some basics:

Exercise 0.2. Let $c > 0$.

Then $1 + c + c^2 + \dots + c^n$ is

(a) $\Theta(1)$ if $c < 1$

(b) $\Theta(n)$ if $c = 1$

(c) ~~$\Theta(n)$~~ $\Theta(c^n)$ if $c > 1$.

(7)

Recall:
Solution: ~~Geometric Series~~

$$1+c+c^2+\dots = \frac{1}{1-c} \text{ if } |c| < 1.$$

and $1+c+c^2+\dots+c^n = \frac{1-c^{n+1}}{1-c} = \frac{c^{n+1}}{c-1}$, if $c \neq 1$.

To check the above, simply cross multiply

$$(1+c+c^2+\dots+c^n)(1-c) \stackrel{?}{=} 1-c^{n+1}$$

$$\Leftrightarrow 1 + c + c^2 + \dots + c^n - c - c^2 - \dots - c^{n+1} = 1 - c^{n+1}. \checkmark$$

~~(a)~~ So @ if $c < 1$ then

$$1+c+c^2+\dots+c^n = \frac{1-c^{n+1}}{1-c} = O(1),$$

since c^{n+1} is exponentially small,
 for $c < 1$ and $1-c$ is a constant.

~~(b)~~ obvious

~~(c)~~ if $c > 1$ then $\frac{c^{n+1}-1}{c-1} = O(c^{n+1}) = O(c^n)$,
 since c is a constant.

————— * —————

(8)

(II) Also continuing w/ basics.

$$\left[\begin{array}{l} 2^{\log_2 n} = n \quad ; \quad 3^{\log_2 n} = n^{\log_2 3} \\ a^{\log_2 n} = n^{\log_2 a} \\ a^{\log_b n} = n^{\log_b a} \\ (\log_a n)(\log_2 a) = \log_2 n \\ \frac{\log_2 n}{\log_2 a} = \log_2 a \end{array} \right]$$

(III) EXAMPLE RECURSIONS

$$(i) T(n) = T\left(\frac{n}{2}\right) + 1$$

$$= \left[T\left(\frac{n}{4}\right) + 1 \right] + 1 = T\left(\frac{n}{4}\right) + 2$$

$$= \left[T\left(\frac{n}{8}\right) + 1 \right] + 2 = T\left(\frac{n}{8}\right) + 3$$

$$\dots = T\left(\frac{n}{2^k}\right) + k, \quad \text{~~RECURSION~~}$$

$$= T(1) + k, \text{ if } n = 2^k.$$

$$= O(\log n), \text{ since } T(1) = O(1).$$

(9)

(ii) $\left\{ \begin{array}{l} T(n) = 3T\left(\frac{n}{2}\right) + \cancel{O(n)} n, \text{ (say)} \\ n \geq 4 \\ \Phi \\ T(2) = 3 ; \text{ Let } n = 2^k ; k \geq 1. \end{array} \right.$

$$\begin{aligned} T(n) &= 3 \underbrace{T\left(\frac{n}{2}\right)}_{\rightarrow} + n \\ &= 3 \left[3T\left(\frac{n}{4}\right) + \frac{n}{2} \right] + n \\ &= 3^2 T\left(\frac{n}{2^2}\right) + \frac{3}{2}n + n \\ &= 3^2 \left[3T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} \right] + \frac{3}{2}n + n \\ &= 3^3 T\left(\frac{n}{2^3}\right) + \left(\frac{3}{2}\right)^2 n + \frac{3}{2}n + n \end{aligned}$$

So continuing this way (or formally by induction
 (or ...)) ,

$$T(n) = 3^{k-1} T\left(\frac{n}{2^{k-1}}\right) + \left(\frac{3}{2}\right)^{k-2} n + \left(\frac{3}{2}\right)^{k-3} n + \dots + \frac{3}{2}n + n$$

Now

$$T\left(\frac{n}{2^{k-1}}\right) = T\left(\frac{\cancel{n}}{2}\right) = 3. \quad \text{So } 3^{k-1} T\left(\frac{n}{2^{k-1}}\right) = 3^k.$$

$$\left\{ \left(\frac{3}{2}\right)^{k-2} + \left(\frac{3}{2}\right)^{k-3} + \dots + \frac{3}{2} + 1 \right\} n = O\left(\left(\frac{3}{2}\right)^k n\right), \quad \text{since } n = 2^k.$$

$$\text{So } T(n) = O(3^k) = O(3^{\log_2 n}) = O(n^{\log_2 3}).$$

B.

(10)

$$(iii) \quad T(n) = T\left(\frac{3n}{4}\right) + \underbrace{O(n)}_{=n}, \text{ say.}$$

$$= T\left(\left(\frac{3}{4}\right)^2 n\right) + \frac{3}{4}n + n$$

$$= T\left(\left(\frac{3}{4}\right)^3 n\right) + \left(\frac{3}{4}\right)^2 n + \left(\frac{3}{4}\right)n + n$$

$$= T\left(\left(\frac{3}{4}\right)^k n\right) + \left[\left(\frac{3}{4}\right)^{k-1} + \left(\frac{3}{4}\right)^{k-2} + \dots + 1\right]n.$$

We want to stop when $\left(\frac{3}{4}\right)^k n = 1$ say

$$\text{Then } n = \left(\frac{4}{3}\right)^k \Rightarrow k = \log_{4/3} n.$$

Also note that or $k = \lceil \log_{4/3} n \rceil$, an integer.

$$\left(\frac{3}{4}\right)^{k-1} + \dots + \left(\frac{3}{4}\right) + 1 = O(1), \text{ since } \frac{3}{4} < 1.$$

(recall geometric series,
with $c < 1$.)

$$\text{Hence } T(n) = T(1) + O(n) = O(n).$$

This comes up in the Median Search algorithm
by ^{the} divide & conquer strategy.

□

$$(IV) \quad T(n) = 7T\left(\frac{n}{2}\right) + n^2, \quad n \geq 2.$$

$$T(1) = 1 \quad \text{let } n = 2^k, k \geq 1.$$

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$= 7\left[7T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2\right] + n^2$$

$$= 7^2T\left(\frac{n}{2^2}\right) + 7\left(\frac{n}{2}\right)^2 + n^2$$

$$= 7^2\left[7T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^2\right] + \cancel{7\left(\frac{n}{2}\right)^2} \frac{7}{2}n^2 + n^2$$

$$= 7^3T\left(\frac{n}{2^3}\right) + \left(\frac{7}{4}\right)^2n^2 + \frac{7}{4}n^2 + n^2$$

$$= 7^kT\left(\frac{n}{2^k}\right) + \left(\frac{7}{4}\right)^{k-1} + \left(\frac{7}{4}\right)^{k-2} + \dots + \frac{7}{4} + 1]n^2$$

$$= 7^k T(1) + O\left(\left(\frac{7}{4}\right)^k n^2\right).$$

Now

$$7^k = 7^{\log_2 n} = n^{\log_2 7}; \quad \left(\frac{7}{4}\right)^k n^2 = 7^k \cdot \frac{n^2}{4^k} = 7^k.$$

$$\text{So } T(n) = O(n^{\log_2 7}), \text{ smaller than } n^{\log_2 8} = n^3.$$

This comes up on the matrix multiplication algorithm using divide & conquer!