Feb. 14, 2018

P. TETALI

DIVIDE & CONQUER PARADIGM

– Solving Recurrences

A) DIVIDE the problem into subproblems

B) Solve the subproblems recursively

C) Merge or combine the solutions to subproblems to solve the larger problem.

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

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

1 6 10 12   2 3 7 8

↑ L   ↑ R

- 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 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 →

Start

1 6 10 12 2 3 7 8

L R

Compare: 1 < 2 ?

Combined List

1							

Step 2:

6 10 12 2 3 7 8

L R

Compare 6 < 2 ?

1	2						

Step 3:

6 10 12 3 7 8

L R

Compare 6 < 3 ?

1	2	3					

Step 4:

6 10 12 7 8

L R

6 < 7 ?

1	2	3	6				

Step 5.

10 12 7 8

L R

10 < 7 ?

1	2	3	6	7			

Step 6

10 12 8

↑ L ↑ R

compare 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. □

Merge Sort; Analysis

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

Merge Sort (a[1..n])

Input : array of n numbers : a[1..n]
Output : Sorted array

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 $T(n)$ = # comparisons done by Merge Sort. Then we have,

$$\begin{cases} T(2) = 1, \text{ and} \\ T(n) \leq 2T\left(\frac{n}{2}\right) + (n-1). \end{cases}$$

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

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

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

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

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

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

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

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

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

So $T(n) = \frac{n}{2} + (\log_2 n - 1)n = n\log_2 n - \frac{n}{2}$

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

§ <u>More recurrences</u>:

(I) First recall some basics:

<u>Exercise 0.2.</u> Let $c > 0$.

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

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

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

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

Solution: Recall: geometric series:

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

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

To check the above, simply cross multiply

$$(1+c+c^2+\cdots+c^n)(1-c) \overset{?}{=} 1-c^{n+1}$$

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

So (a) if $c<1$ then

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

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

(b) obvious

(c) if $c>1$ then $\dfrac{c^{n+1}-1}{c-1} = O(c^{n+1}) = O(c^n)$,

since c is a constant.

(II) Also continuing w/ basics.

$$\begin{cases} 2^{\log_2 n} = n \; ; \quad 3^{\log_2 n} = n^{\log_2 3} \\ a^{\log_2 n} = n^{\log_2 a} \\ a^{\log_b n} = n^{\log_b a}. \end{cases}$$

$$\begin{cases} (\log_a n)(\log_2 a) = \log_2 n \\ \dfrac{\log_2 n}{\log_a n} = \log_2 a \end{cases}$$

(III) ~~II~~ EXAMPLE RECURRENCES

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

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

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

$\ldots$

$= T\left(\frac{n}{2^k}\right) + k$, ~~where~~

$= T(1) + k$, if $n = 2^k$.

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

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

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

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

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

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

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

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

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

Now

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

$$\left[\left(\frac{3}{2}\right)^{k-2} + \left(\frac{3}{2}\right)^{k-3} + \cdots + \frac{3}{2} + 1\right] n = O\left(\left(\frac{3}{2}\right)^k n\right) = O(3^k),$$

since $n = 2^k$.

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

$\square$.

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

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

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

$$\vdots$$

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

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

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 <u>integer</u>.

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

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

Hence $T(n) = T(1) + O(n) = O(n)$.

This comes up in the <u>Median Search</u> algorithm by the divide & conquer strategy.

$\square$

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

$T(1) = 1$ let $n = 2^k$, $k \geq 1$.

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

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

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

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

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

$$\cdots$$

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

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

Now

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

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

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