Lecture 25

Tuesday, 12 October 2021 10:01 AM

$a, d > 0 \quad b > 1$

Recurrence:

$$T(n) = \underbrace{a}_{\text{No of subproblems}} T\Big(\underbrace{\frac{n}{b}}_{\text{size of the subproblem}}\Big) + \underbrace{O(n^d)}_{\text{Divide + Combine step}}$$

Eg:

Merge/Quick Sort: $T(n) = 2T(n/2) + O(n)$

Min Pairwise Distance: $T(n) = 2T(n/2) + O(n)$

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

$$\underset{A}{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} \underset{B}{\begin{bmatrix} \; & \; \\ \; & \; \end{bmatrix}} = \begin{bmatrix} \; & \; \\ \; & \; \end{bmatrix}$$

Lemma 1 : For $x < 1$, $1 + x + x^2 + x^3 + \ldots \leq \frac{1}{1-x} = O(1)$ if x is constant

Lemma 2 : For $x > 1$, $1 + x + x^2 + \cdots + x^{n-1} = \frac{x^n - 1}{x-1} \leq \frac{x^n}{x-1} = O(x^n)$ if x is constant

Lemma 3 : $a^{\log_b(n)} = n^{\log_b(a)}$

(Proof : If you take $\log_b(\ )$ on both sides, then
LHS $= \log_b(n) \log_b(a) =$ RHS)

$$T(n) = a\,T(n/b) + c\,n^d$$

$$= a\left(a\,T\left(\frac{n}{b^2}\right) + c\left(\frac{n}{b}\right)^d\right) + c\,n^d$$

$$= a^2\,T\left(\frac{n}{b^2}\right) + c\,n^d\left(1 + \frac{a}{b^d}\right)$$

$$= a^2\, T\left(\frac{n}{b^2}\right) + c n^d \left(1 + \frac{a}{b^d}\right)$$

$$= a^2 \left(a\, T\left(\frac{n}{b^3}\right) + c\left(\frac{n}{b^2}\right)^d\right) + c n^d \left(1 + \frac{a}{b^d}\right)$$

$$= a^3\, T\left(\frac{n}{b^3}\right) + c n^d \left(1 + \frac{a}{b^d} + \frac{a^2}{b^{2d}}\right)$$

$\vdots$

$i = \log_b n$

$$= \underbrace{a^i\, T\left(\frac{n}{b^i}\right)}_{a^{\log_b n} \cdot T(1)} + c n^d \left(1 + \frac{a}{b^d} + \frac{a^2}{b^{2d}} + \dots + \frac{a^{i-1}}{b^{(i-1)d}}\right)$$

$$= \boxed{n^{\log_b a}} + \boxed{\text{—..||——}}$$

$$T(n) = \begin{cases} \underbrace{n^{\log_b a}} + n^d \log_b n & \text{if } \frac{a}{b^d} = 1 \\ \underbrace{n^{\log_b a}} + n^d \cdot 1 & \text{if } \frac{a}{b^d} < 1 \\ n^{\log_b a} + n^d \left(a^{\log_b n} \right. \dots \end{cases}$$

$$n^{\log_b a} + n^d \cdot \left(\frac{a}{b^d}\right)^{\log_b n} \quad \text{if} \quad \frac{a}{b^d} > 1$$

Case 1 & Case 2 : $a \le b^d$

$$\log_b a \le \log_b b = d$$

$$\underbrace{n^{\log_b a}}_{\text{LHS}} \le n^d$$

Case 3 :

$$\text{RHS} = \frac{n^d \cdot a^{\log_b n}}{b^{\log_b n^d}} = \frac{\cancel{n^d} \cdot n^{\log_b a}}{\cancel{(n^d)(\log_b b)}}$$

$$= \text{LHS}$$

$$T(n) = \begin{cases} O(n^d \log_b n) & \text{if } \frac{a}{b^d} = 1 \\ O(n^d) & \text{if } \frac{a}{b^d} < 1 \\ O(n^{\log_b a}) & \text{if } \frac{a}{b^d} > 1 \end{cases}$$

Master's Theorem.

Quick Sort : $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$a = 2 \quad b = 2 \quad d = 1 \qquad \frac{a}{b^d} = 1$

$T(n) = O(n \log_2 n)$

Matrix Product: $T(n) = 7\, T(n/2) + O(n^2)$

$a = 7 \quad b = 2 \quad d = 2 \qquad \frac{a}{b^d} = \frac{7}{4} > 1$

$O(n^{\log_2 7})$

Integer Product

Method I

$a = a_1 \, 2^{n/2} + a_0$

$b = b_1 \, 2^{n/2} + b_0$

$$b = b_1 2 + b_0$$

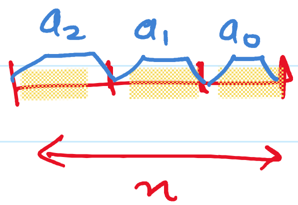
$$ab = \underbrace{(a_1 b_1)}_{\text{ii}} 2^{n} + (a_1 b_0 + a_0 b_1) * 2^{\frac{n}{2}} + \underbrace{a_0 b_0}_{\text{iii}}$$

$$\underbrace{(a_1 + a_0)(b_1 + b_0)}_{\text{i}} - \underbrace{a_1 b_1}_{\text{ii}} - \underbrace{a_0 b_0}_{\text{iii}}$$

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

$$= O\left(n^{\log_2 3}\right) \quad (\text{By Master's Thm})$$

Method II



$$a = a_2\, 2^{\frac{2n}{3}} + a_1\, 2^{\frac{n}{3}} + a_0$$

Assume $n = pow(3)$

$$b = b_2\, 2^{\frac{2n}{3}} + b_1\, 2^{\frac{n}{3}} + b_0$$

$$ab = (a_2 b_2)\, 2^{\frac{4n}{3}} + (a_2 b_1 + a_1 b_2)\, 2^{\frac{3n}{3}} + (\quad)\, 2^{\frac{2n}{3}} + (\quad)\, 2^{n/3} + a_0 b_0$$

$$ab = (a_2 b_2)\, 2^{\frac{4n}{3}} + (a_2 b_1 + a_1 b_2)\, 2^{\frac{3n}{3}} + (\quad)\, 2^{\frac{2n}{3}} + (\quad)\, 2^{n/3} + a_0 b_0$$

H. W. — Represent all five terms as add / sub of five products.

$$T(n) = 5\,T\left(\frac{n}{3}\right) + O(n)$$

$$= O\left(n^{\log_3 5}\right)$$

Method III
FFT

$$a = a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_1 2 + a_0 \Big] \leq n+1 \text{ bits}$$

$$a(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$$b = b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_1 2 + b_0 \Big] \leq n+1 \text{ bits}$$

$$b(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$$

Compute $c(x) = a(x) \cdot b(x)$ in $O(n \log n)$ time

Compute $C(x) = a(x) \cdot b(x)$ in $O(n \log n)$ time

$$= C_{2n} x^{2n} + \dots + C_1 x + C_0$$

$$C_i = a_i b_0 + a_{i-1} b_1 + \dots + a_1 b_{i-1} + a_0 b_i \leq i+1$$

2 binary bits

$$\text{Output } C(2) = C_{2n} 2^{2n} + C_{2n-1} 2^{2n-1} + \dots + C_2 2^2 + C_1 2 + C_0$$

$$\text{Trivial} = O(n) * O(n) = O(n^2)$$

<u>Algorithm</u>

① $ans = C_0$

② For $i = 1$ to $2n$: $ans = ans + C_i \cdot \underbrace{2^i}_{\text{bit-shifting}}$

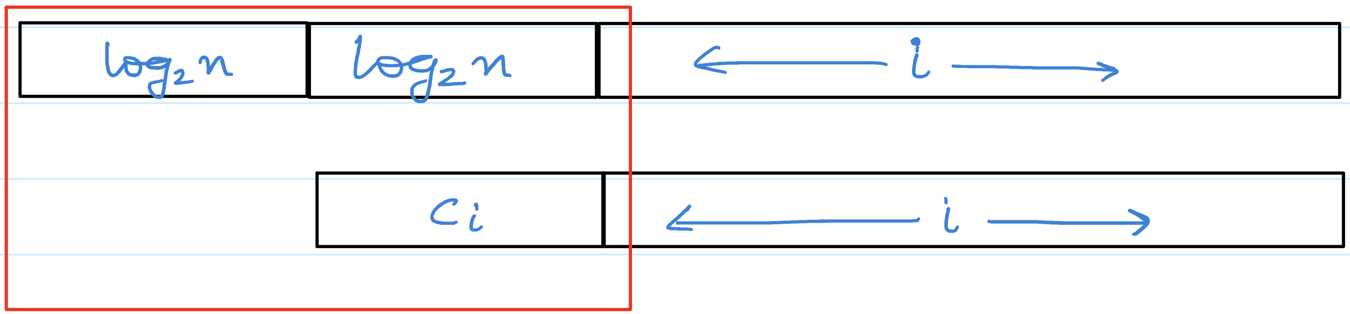
③ Return ans.

Before Round i : $ans = c_{i-1}2^{i-1} + \dots + c_1 2 + c_0 \leq i^2 \cdot 2^i$

Before Round i, number of bits taken by $ans \leq i + 2\log_2 n$

ans (before round i) →

c_i shifted by i bits →



Computing sum takes $O(\log_2 n)$ time

Time taken by step 2 – $O(\log n)$

$\Rightarrow$ Time to compute $C(2) = O(n \log_2 n)$

REMARK: Product of 2 n-bit numbers take $O(n \log n)$ time as long as n is small.

In practice, FFT involves rounding errors of complex numbers, so to tackle it there is extra overhead in time complexity.

See this for further reference:
https://en.wikipedia.org/wiki/F%C3%BCrer%27s_algorithm