Lecture 3 Divide and Conquer 2

Wednesday, January 15, 2020 12:52 PM

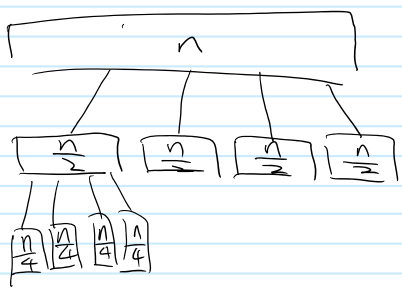
- Integer multiplication
  - First attempt

    Let $T(n)$ be the running time of multiplying two n-digit numbers.

$$T(n) = \underbrace{O(n)}_{\text{merge cost}} + \underbrace{4T\left(\frac{n}{2}\right)}_{\text{recursion cost.}}$$

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

  - use recursion tree



layer	#nodes	size
0	1	n
1	4	$\frac{n}{2}$
2	16	$\frac{n}{4}$
i	4^i	$\frac{n}{2^i}$
$\log_2 n$	$4^{\log_2 n}$	1

$$T(n) = \sum_{i=0}^{\log_2 n - 1} \text{total merge cost at level } i$$

$$= \sum_{i=0}^{\log_2 n-1} 4^i \times \frac{n}{2^i}$$

$$= \sum_{i=0}^{\log_2 n-1} 2^i \times n$$

$$= n \times \left(\sum_{i=0}^{\log_2 n-1} 2^i\right)$$

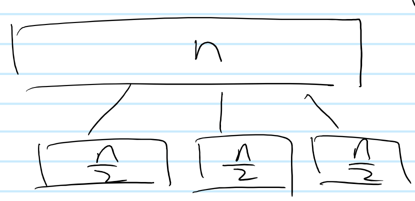
$$1+2+4+\cdots+\underbrace{2^{\log_2 n-1}}_{\frac{n}{2}}$$

$$= n\times(n-1) = \Theta(n^2)$$

  - Improved algorithm

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

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



layer	#nodes	size
0	1	n
1	3	$\frac{n}{2}$
2	9	$\frac{n}{4}$
i	3^i	$\frac{n}{2^i}$
$\log_2 n$	$3^{\log_2 n}$	1

$$T(n) = \sum_{i=0}^{\log_2 n-1} 3^i \times \frac{n}{2^i}$$

$$= n \times \sum_{i=0}^{\log_2 n-1} \left(\frac{3}{2}\right)^i$$

$$S = 1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^p$$

$$\frac{3}{2}S = \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^p + \left(\frac{3}{2}\right)^{p+1}$$

$$\frac{S}{2} = \left(\frac{3}{2}\right)^{p+1} - 1 \Rightarrow S = 2\left(\frac{3}{2}\right)^{p+1} - 2$$

$$\frac{S}{2} = \left(\frac{3}{2}\right) - 1 \Rightarrow S = 2\left(\frac{3}{2}\right) - 2$$

$$= n \times \left(2 \cdot \left(\frac{3}{2}\right)^{\log_2 n} - 2\right)$$

$$= n \times \left(2 \frac{3^{\log_2 n}}{\underbrace{2^{\log_2 n}}_{=n}} - 2\right)$$

$$= 2 \times 3^{\log_2 n} - 2n$$

$$\| $$

$$n^{\log_2 3} \approx n^{1.5??}$$

$$= \Theta(n^{\log_2 3}) \ll n^2$$

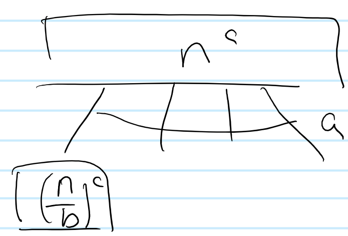
- Faster integer multiplication : FFT $O(n \log n \log \log^2 n)$

- Master Theorem

- $1 + c + c^2 + c^3 + \cdots + c^p = \begin{cases} c > 1 & \Theta(c^p) \\ \underline{c = 1} & \Theta(p) \\ c < 1 & \Theta(1) \end{cases}$

- case 1

cost for layer 0 $= n^c$



layer 1 $= a\left(\frac{n}{b}\right)^c$

$= n^c \cdot \frac{a}{b^c}$

$> n^c$

cost dominated by the cost of last layer

$\Theta(n^{\log_b a})$

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

$a = 4 \quad b = 2 \quad c = 1$

$c < \log_b a \qquad \Theta(n^{\log_b a})$

$= \Theta(n^2)$

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

$a=3 \quad b=2 \quad c=1 \qquad \Theta(n^{\log_2 3})$

- case 2   layers have similar cost

$T(n) = \text{merge cost for top layer} \times \underbrace{\#\text{ layers}}_{\Theta(\log n)}$

Example: merge sort $T(n) = 2T\left(\frac{n}{2}\right) + n^1$

$c=1 \quad a=b=2 \quad \log_b a = 1$

-

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

- case 3   cost dominated by top layer

Failed attempt for Counting Inversions

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

$c=2 \quad a=b=2 \quad \log_b a = 1$

$T(n) = \Theta(n^2)$