Welcome to CS II

Algorithms

Plan

Week 1 Divide-and-conquer algorithms

Weeks 2-3 Graph algorithms

Week 4 Greedy algorithms

Week 5 Dynamic programming algorithms

Week 6 NP-completeness

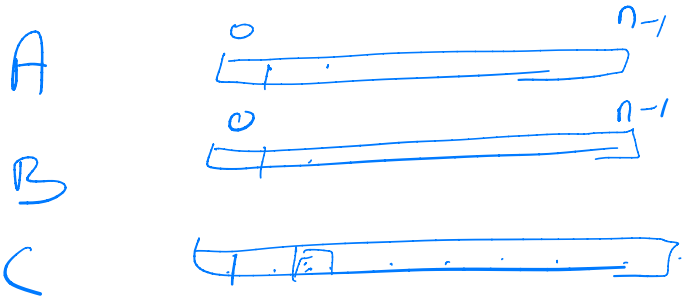
Integer multiplication

$A = 2\,7\,7\,6\,5\,4\,8$

$a_6 \quad a_5\, a_4\, a_3\, a_2\, a_1\, a_0$

8	4	5	6	7	7	2

$$\sum_{i=0}^{n-1} 10^i \cdot a_i$$



$C = A + B$

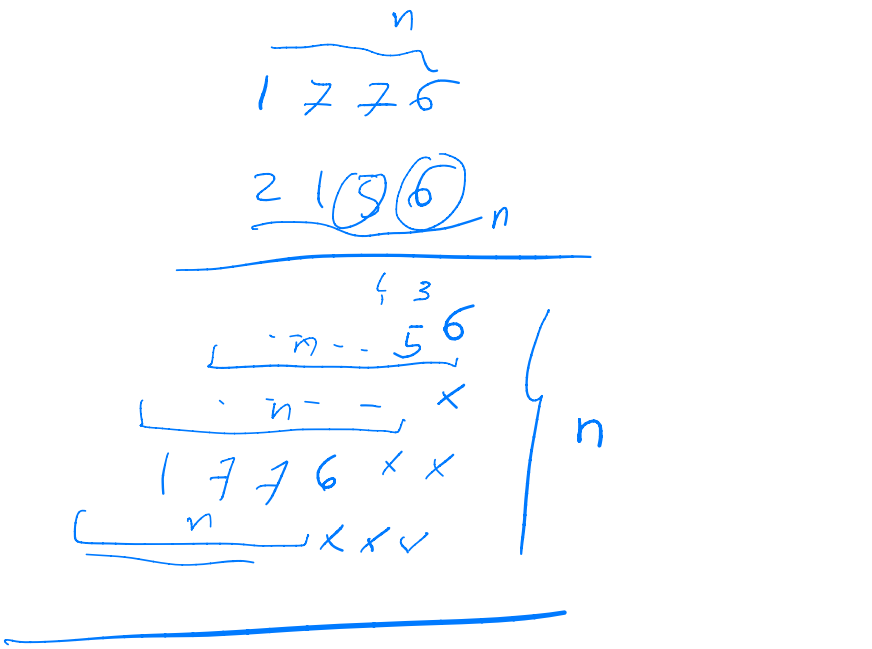
running time $O(n)$

$$T(n) = O(f(n))$$

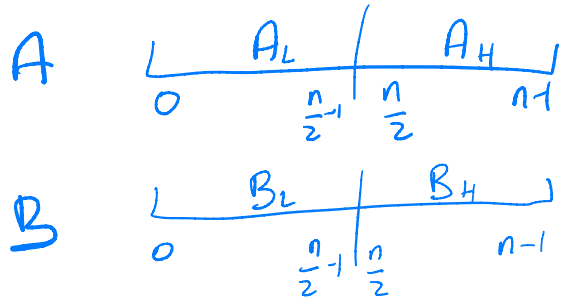
there is constant c such that

$T(n) \le c \cdot f(n)$ for all $n \ge 1$

$$\limsup_{n \to \infty} \frac{T(n)}{f(n)} < \infty$$



running time $O(n^2)$



$$A = A_L + 10^{n/2} \cdot A_H$$

$$B = B_L + 10^{n/2} \cdot B_H$$

$$A \cdot B = (A_L + 10^{n/2} \cdot A_H)(B_L + 10^{n/2} B_H)$$

$$= A_L \cdot B_L + 10^{n/2} A_H B_L + 10^{n/2} A_L B_H + 10^n A_H B_H$$

$T(n)$ time takes to multiply two n-digit numbers

computing $10^{n/2}$ ← $O(n)$ + $O(n)$ ← complex sums

$$T(n) = 4 \cdot T(n/2) + O(n) + O(n)$$

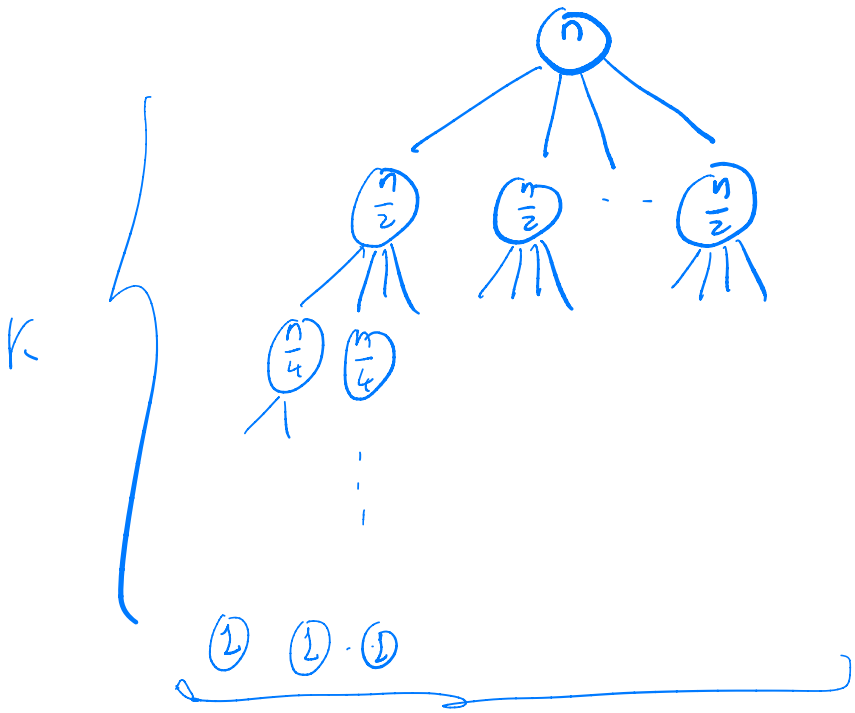
$$= 4 \cdot T(n/2) + O(n)$$

$$T(1) = O(1)$$

$$\boxed{\begin{aligned} t(n) &= 4\,t(n/2) + n \\ t(1) &= 1 \end{aligned}}$$

$T(n) = 4 \cdot T(n/2) + n$

$T(1) = 1$



$n + 2 \cdot n + 4 \cdot n + 8n + \cdots + n^2$

$= n \cdot (1 + 2 + 4 + \cdots n)$

$= n\,(2n-1) = O(n^2)$

$n +$

$4 \cdot \frac{n}{2} +$

$16 \cdot \frac{n}{4} +$

$\vdots$

n^2

$= n^{\log_2 4} = n^2$

$A = A_L + 10^{n/2} A_H$ $\quad A$ n-digits

A_L, A_H $\frac{n}{2}$-digits

$B = B_L + 10^{n/2} B_H$

$$A \cdot B = A_L B_L + 10^{n/2}(A_L B_H + A_H B_L) + 10^n A_H B_H$$

recursively compute

$X = A_L \cdot B_L$

$Y = A_H \cdot B_H$

$Z = (A_L + A_H) \cdot (B_L + B_H) = A_L B_L + A_H B_L + A_L B_H + A_H B_H$

$$Z - X - Y = A_H B_L + A_L B_H$$

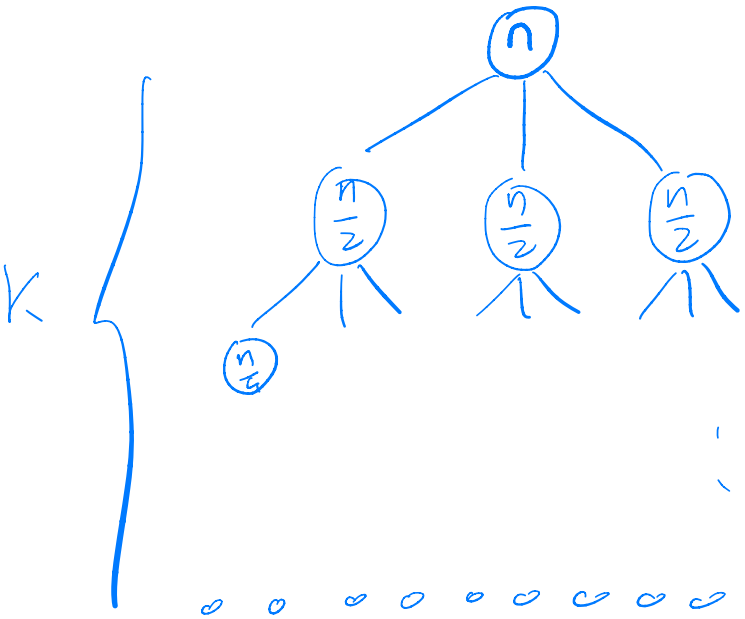
return $X + 10^{n/2}(Z - X - Y) + 10^n Y$

$$T(n) = 3T(n/2) + O(n)$$

$$t(n) = 3t(n/2) + n$$

$$t(1) = 1$$

$$t(n) = 3t(n/2) + n$$
$$t(1) = 1$$



$n +$

$3 \cdot \frac{n}{2} +$

$9 \cdot n/4 +$

$\vdots$

$3^K = 3^{\log_2 n} = n^{\log_2 3}$

$$t(n) = n\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \left(\frac{3}{2}\right)^{\log_2 n}\right)$$

$$\frac{n^{\log_2 3}}{n}$$

$$t(n) = O(n^{\log_2 3})$$

$$T(n) = O(n^{\log_2 3}) = O(n^{1.5849\ldots})$$

$$t(n) = a \cdot t(n/b) + n^d$$

divide-and-conquer algorithm that

given an input of size n

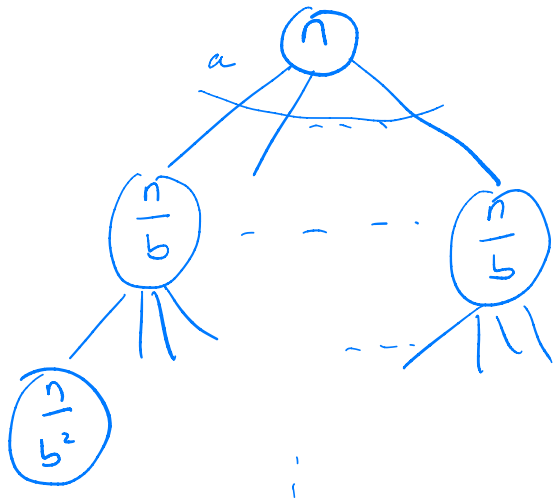
creates a recursive calls on

inputs of size n/b

and takes $O(n^d)$ extra time

naive	multiplication	$a = 4$	$b = 2$	$d = 1$
faster	multiplication	$a = 3$	$b = 2$	$d = 1$

$$t(n) = a \cdot t(n/b) + n^d$$



n^d

$a \cdot \left(\frac{n}{b}\right)^d +$

$a^2 \cdot \left(\frac{n}{b^2}\right)^d +$

$K = \log_b n$

① — — — — ① $\quad a^K = a^{\log_b n} = n^{\log_b a}$

$$t(n) = n^d \left(1 + a/b^d + (a/b^d)^2 + \cdots \frac{n^{\log_b a}}{n^d}\right)$$

$a = b^d \quad t(n) = n^d \cdot \log_b d \quad t(n) = O(n^d \log n)$

$a > b^d \quad t(n) = O(n^{\log_b a})$

$a < b^d \quad t(n) = O(n^d)$

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} - - - - - + \frac{1}{2^K} + \cdots$$

$$= 2$$

$p < 1 \qquad 1 + p + p^2 + p^3 + \cdots + p^n + \cdots = \frac{1}{1-p} = O(1)$

Next Time

Mergesort $O(n \log n)$

$t(n) = 2t(n/2) + O(n)$

Median randomized $O(n)$

$t(n) = t(n/2) + O(n)$

FFT $O(n \log n)$

$t(n) = 2t(n/2) + O(n)$