

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

$$A = 2776548 \quad a_6 \quad a_5 a_4 a_3 a_2 a_1 a_0$$

[8 | 4 | 5 | 1 | 5 | 7 | 7 | 2]



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

$$\begin{array}{l} A \\ B \\ C \\ \hline C = A + B \end{array}$$

Diagram illustrating the addition of two numbers A and B. Each number is represented as a horizontal bar divided into segments labeled from 0 to n-1. The segments are aligned vertically, and the sum C is shown as a horizontal bar where each segment's value is the sum of the corresponding segments of A and B.

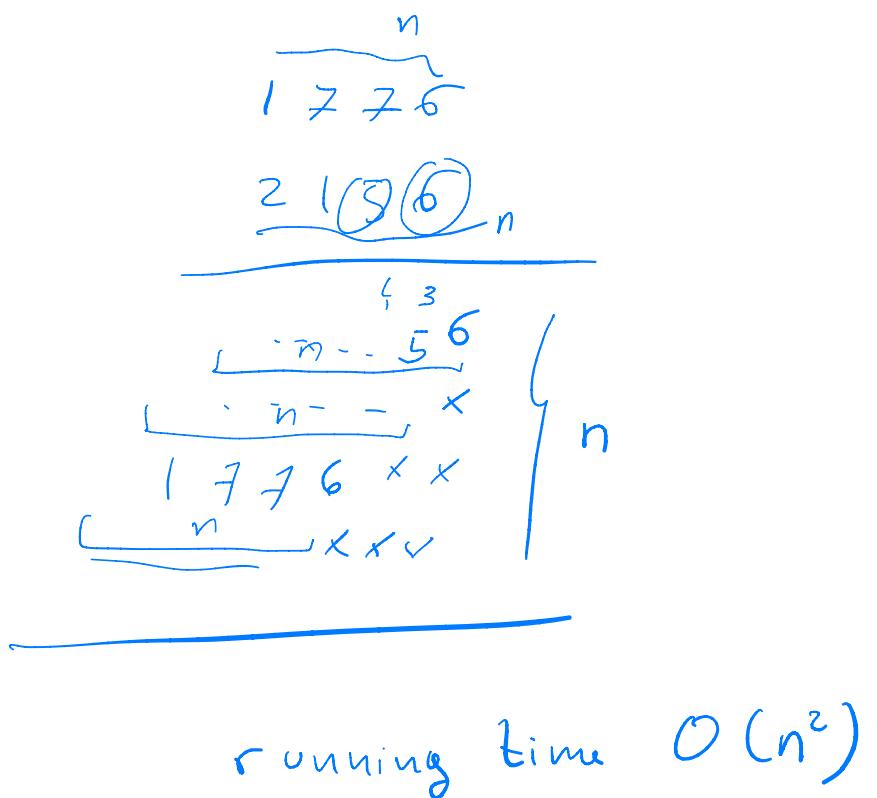
running time  $\mathcal{O}(n)$

$$T(n) = \mathcal{O}(f(n))$$

There is constant  $c$  such that

$$T(n) \leq c \cdot f(n) \quad \text{for all } n \geq 1$$

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



running time  $O(n^2)$

$$A \quad \begin{array}{c|cc|c} A_L & & A_H & \\ \hline 0 & \frac{n}{2}-1 & \frac{n}{2} & n-1 \end{array}$$

$$B \quad \begin{array}{c|cc|c} B_L & & B_H & \\ \hline 0 & \frac{n}{2}-1 & \frac{n}{2} & n-1 \end{array}$$

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

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

$$\begin{aligned} A \cdot B &= (A_L + 10^{\frac{n}{2}} \cdot A_H)(B_L + 10^{\frac{n}{2}} \cdot B_H) \\ &= A_L \cdot B_L + 10^{\frac{n}{2}} A_H B_L + 10^{\frac{n}{2}} A_L B_H + 10^n A_H B_H \end{aligned}$$

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

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

compute  $10^{\frac{n}{2}}$  & sums

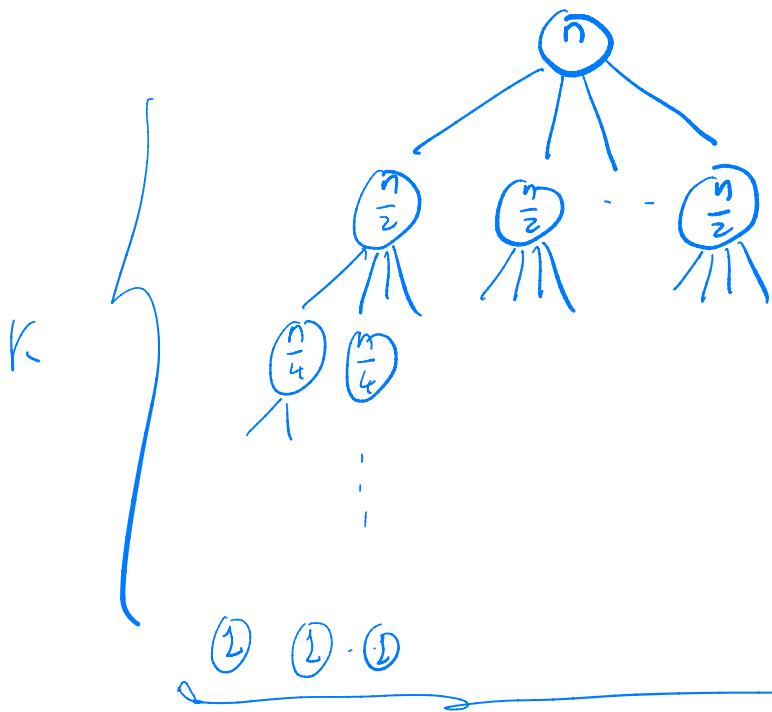
$$= 4 \cdot T\left(\frac{n}{2}\right) + O(n)$$

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

$$\boxed{\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + n \\ T(1) &= 1 \end{aligned}}$$

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

$$t(1) = 1$$



$$\begin{aligned} & n + 2 \cdot n + 4 \cdot n + 8n + \dots + n^2 \\ &= n \cdot \underbrace{(1+2+4+\dots+n)}_{n+} \\ &= n \cdot \underbrace{(2n-1)}_{n+} = \mathcal{O}(n^2) \end{aligned}$$

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

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

⋮

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

$$A = A_L + 10^{n_2} A_H \quad A \text{ } n\text{-digits}$$

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

$$B = B_L + 10^{n_2} B_H$$

$$A \cdot B = A_L B_L + 10^{n_2} (A_L B_H + A_H B_L) + 10^n A_H B_H$$

recursively compute

$$X = A_L \cdot B_L$$

$$Y = A_H \cdot B_H$$

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

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

$$\text{return } X + 10^{n_2} (Z - X - Y) + 10^n Y$$

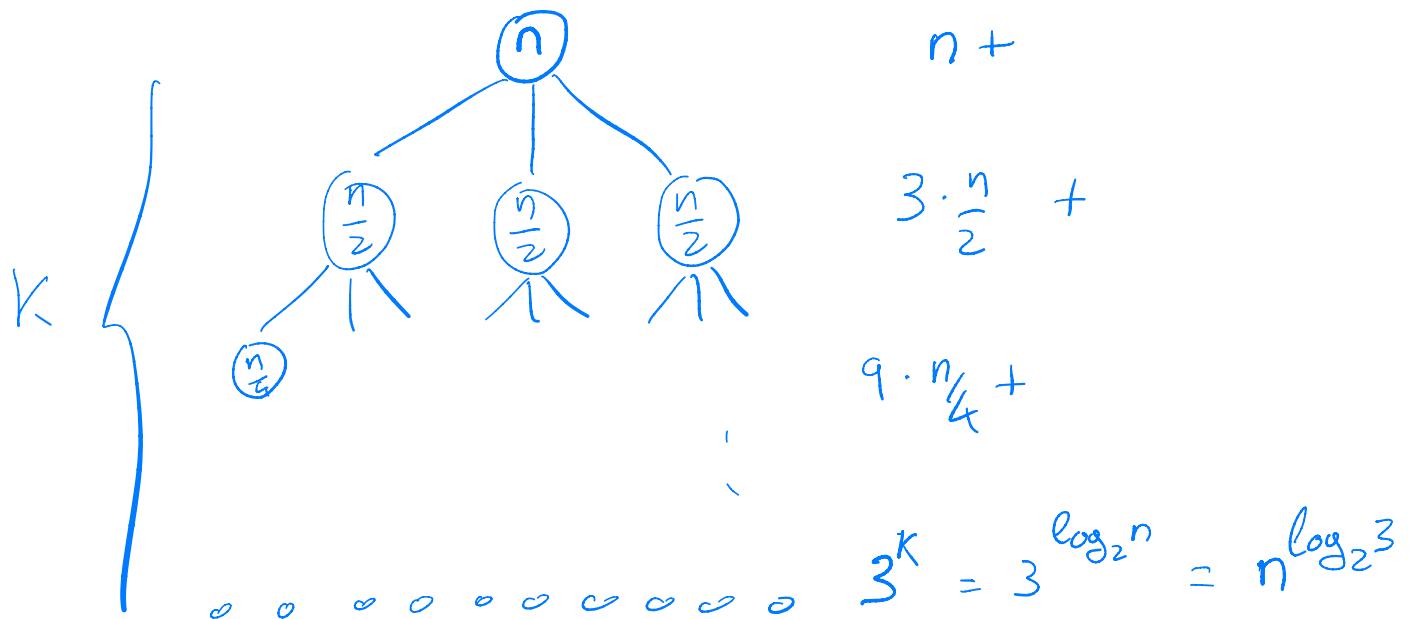
$$T(n) = 3T(n/2) + O(n)$$

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

$$t(1) = 1$$

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

$$t(1) = 1$$



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

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

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

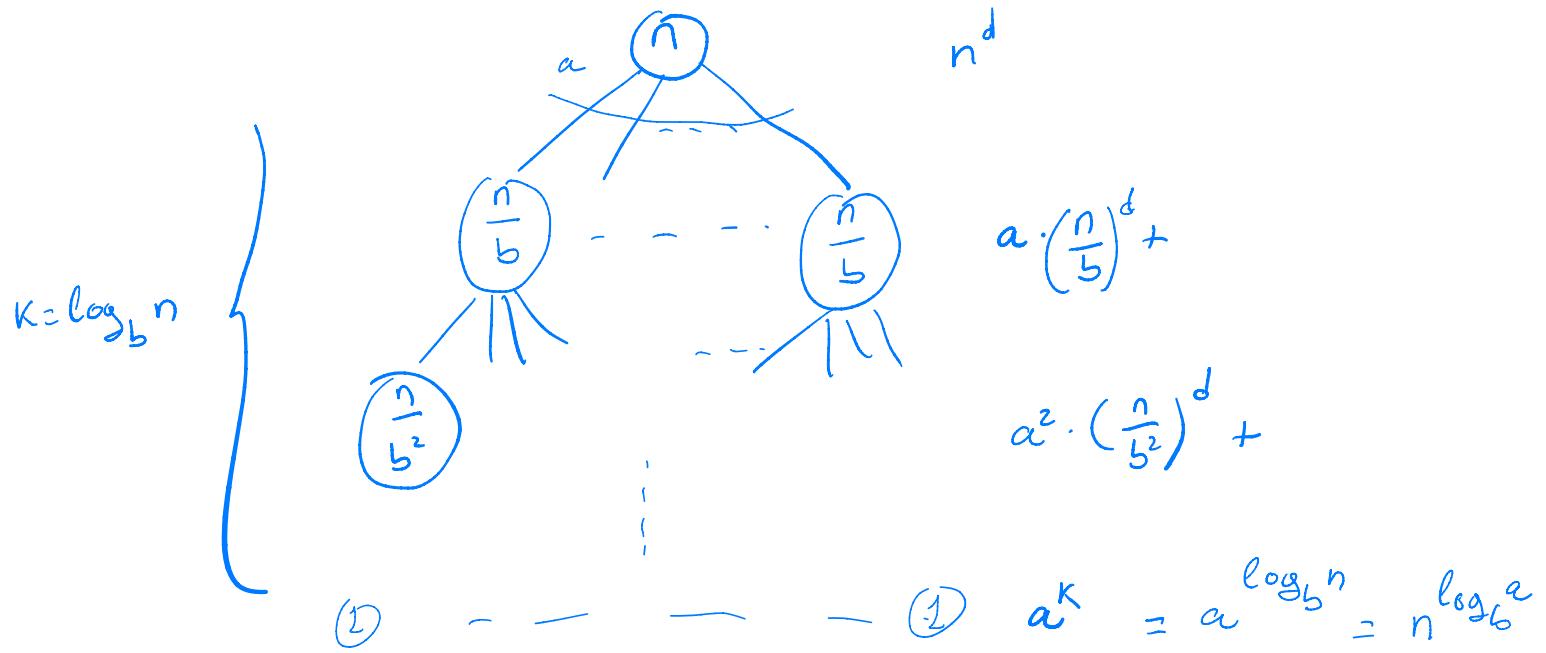
$$T(n) = a \cdot T\left(\frac{n}{b}\right) + n^d$$

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       $a=3$        $b=2$        $d=1$

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



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

$a = b^d$	$T(n) = n^d \cdot \log_b d$	$T(n) = O(n^d \log n)$
$a > b^d$		$T(n) = O(n^{\log_b a})$
$a < b^d$		$T(n) = O(n^d)$

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots + \frac{1}{2^k} + \dots$$

$$\begin{aligned} p < 1 & \quad 1 + p + p^2 + p^3 + \dots + p^n + \dots = \frac{1}{1-p} = O(1) \\ & \quad = 2 \end{aligned}$$

# 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

$O(n \log n)$

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