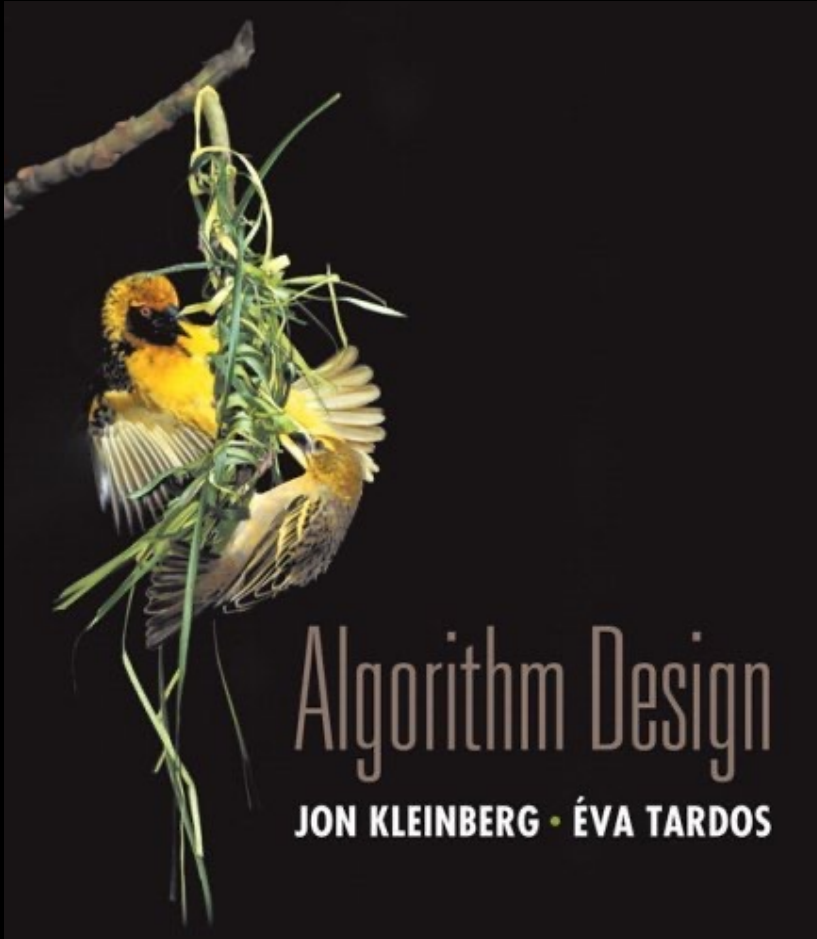

The slides are mostly coming from Kleinberg's textbook (made by Kevin Wayne). There might be slight modifications to the slides according to the class need, plan, and time restrictions. I have also added some slides mostly to add more explanations, examples and definitions for your better understanding of the concepts.

Instructor

Chapter 5

Divide and Conquer



Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

5.5 Integer Multiplication

Integer Arithmetic

Add. Given two n-digit integers a and b, compute $a + b$.

- How many bit operations?

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
<hr/>								
1	0	1	0	1	0	0	1	0

Add

Integer Arithmetic

Add. Given two n -digit integers a and b , compute $a + b$.

- $O(n)$ bit operations.

Multiply. Given two n -digit integers a and b , compute $a \times b$.

- Brute force solution: how many bit operations?

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Add

Diagram illustrating the long multiplication of two 10-bit numbers:

Multiplicand: 0 1 1 0 1 0 0 0 0 0

Multiplier: 1 1 0 1 0 1 0 1 0 1

The diagram shows the partial products generated by multiplying each bit of the multiplier by the multiplicand, shifted according to its position:

- 1 1 0 1 0 1 0 1 0 1 (Multiplier bit 1, shifted 0 positions)
- 0 1 1 0 1 0 1 0 1 0 (Multiplier bit 0, shifted 1 position)
- 1 1 0 1 0 1 0 1 0 0 (Multiplier bit 1, shifted 2 positions)
- 0 0 0 0 0 0 0 0 0 0 (Multiplier bit 0, shifted 3 positions)
- 1 1 0 1 0 1 0 1 0 0 (Multiplier bit 1, shifted 4 positions)
- 0 1 1 0 1 0 1 0 1 0 (Multiplier bit 0, shifted 5 positions)
- 1 1 0 1 0 1 0 1 0 0 (Multiplier bit 1, shifted 6 positions)
- 0 0 0 0 0 0 0 0 0 0 (Multiplier bit 0, shifted 7 positions)
- 1 1 0 1 0 1 0 1 0 0 (Multiplier bit 1, shifted 8 positions)
- 0 1 1 0 1 0 1 0 1 0 (Multiplier bit 0, shifted 9 positions)

The final result is: 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0

Integer Arithmetic

Add. Given two n -digit integers a and b , compute $a + b$.

- $O(n)$ bit operations.

Multiply. Given two n -digit integers a and b , compute $a \times b$.

- Brute force solution: $\Theta(n^2)$ bit operations.

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Add

[illegible]

Divide-and-Conquer Multiplication: Warmup

To multiply two n -digit integers:

- Multiply four $\frac{1}{2}n$ -digit integers.
- Add two $\frac{1}{2}n$ -digit integers, and shift to obtain result.

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \\xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) = 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0\end{aligned}$$

$$T(n) = \underbrace{4T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, shift}} \Rightarrow T(n) = \Theta(n^2)$$



assumes n is a power of 2

Karatsuba Multiplication

To multiply two n -digit integers:

- Add two $\frac{1}{2}n$ digit integers.
- Multiply **three** $\frac{1}{2}n$ -digit integers.
- Add, subtract, and shift $\frac{1}{2}n$ -digit integers to obtain result.

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \\xy &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \\&= \underbrace{2^n \cdot x_1 y_1}_A + 2^{n/2} \cdot \underbrace{((x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0)}_B + \underbrace{x_0 y_0}_C\end{aligned}$$

Theorem. [Karatsuba-Ofman, 1962] Can multiply two n -digit integers in $O(n^{1.585})$ bit operations.

$$\begin{aligned}T(n) &\leq \underbrace{T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(1 + \lceil n/2 \rceil)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, subtract, shift}} \\&\Rightarrow T(n) = O(n^{\log_2 3}) = O(n^{1.585})\end{aligned}$$

Karatsuba: Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n=1 \\ 3T(n/2) + n & \text{otherwise} \end{cases}$$

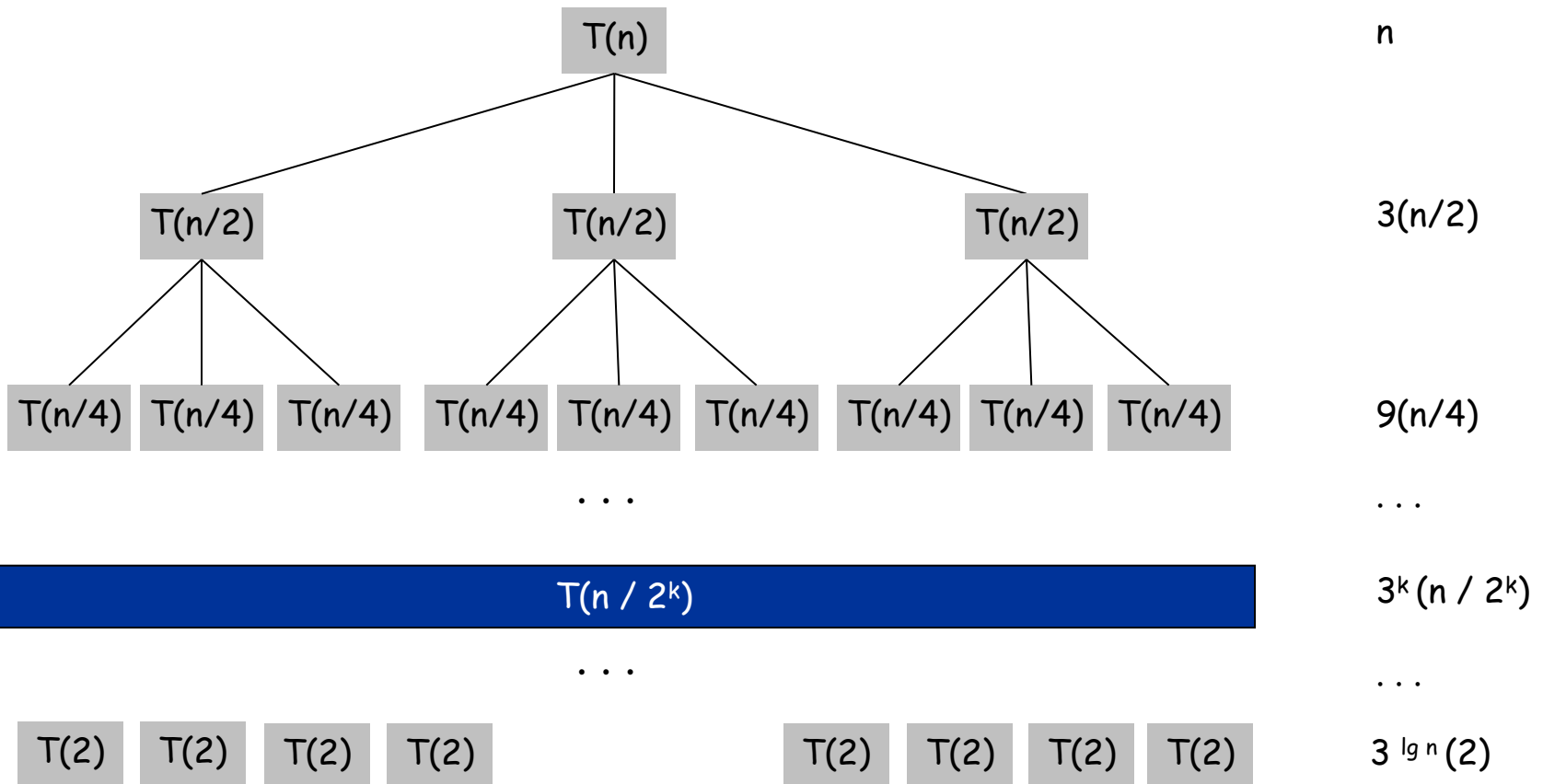
By Master Theorem,

$$n^{\log 3} > n \Rightarrow T(n) = O(n^{\log 3}) = O(n^{1.585})$$

Karatsuba: Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n=1 \\ 3T(n/2) + n & \text{otherwise} \end{cases}$$

$$T(n) = \sum_{k=0}^{\log_2 n} n \left(\frac{3}{2}\right)^k = n \frac{\left(\frac{3}{2}\right)^{1+\log_2 n} - 1}{\frac{3}{2} - 1} = 3n^{\log_2 3} - 2n$$



Example

Apply the algorithm to solve $x.y$ once :

$x = 11010011,$

$y = 01011001$