Lecture 4: Integer and Matrix Multiplication

More complicated examples of divide-and-conquer

Version of Feb 13, 2019

Integer Arithmetic

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

- $\Theta(n)$ time

Multiply. Given two n-bit integers a and b, compute $a \cdot b$.

- Primary school method: $\Theta(n^2)$ time.
  - A.k.a. "long multiplication"

	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

							1	1	0	1	0	1	0	1
						×	0	1	1	1	1	1	0	1
							1	1	0	1	0	1	0	1
						0	0	0	0	0	0	0	0	
					1	1	0	1	0	1	0	1		
				1	1	0	1	0	1	0	1			
			1	1	0	1	0	1	0	1				
		1	1	0	1	0	1	0	1					
	1	1	0	1	0	1	0	1						
0	0	0	0	0	0	0	0							
1	1	0	1	0	0	0	0	0	0	0	0	0	0	1

Divide-and-Conquer Multiplication: First Attempt

Observation:

- **Let X,a, b, c, d be integers**
- **Simple algebra says**

$$(a\boldsymbol{X}+b)\,(c\boldsymbol{X}+d) = ac\boldsymbol{X}^2 + (ad+bc)\boldsymbol{X} + bd$$

- If $X = 2^{n/2}$ this becomes

$$\left(a\mathbf{2}^{\boldsymbol{n}/\mathbf{2}}+b\right)\left(c\mathbf{2}^{\boldsymbol{n}/\mathbf{2}}+d\right) = ac\mathbf{2}^{\boldsymbol{n}} + (ad+bc)\mathbf{2}^{\boldsymbol{n}/\mathbf{2}} + bd$$

- Example

$$163 \qquad 97 \qquad = \qquad 15{,}811$$

$$\left(10\cdot\mathbf{2}^{\mathbf{4}} + 3\right)\left(6\cdot\mathbf{2}^{\mathbf{4}}+1\right) = 60\cdot\mathbf{2}^{\mathbf{8}} + (10+18)\mathbf{2}^{\mathbf{4}} + 3$$

Divide-and-Conquer Multiplication: First Attempt

Recall:

- If $X = 2^{n/2}$

$$\left(a\mathbf{2}^{\boldsymbol{n}/\mathbf{2}} + b\right)\left(c\mathbf{2}^{\boldsymbol{n}/\mathbf{2}} + d\right) = ac\mathbf{2}^{\boldsymbol{n}} + (ad + bc)\mathbf{2}^{\boldsymbol{n}/\mathbf{2}} + bd$$

- Integers are stored in computers in binary format.
  - Multiplication by 2^k can be done in one time unit by performing a left shift of k bits

- Example 10 = 00001010
  - 10 X 2^3 = 80
  - is the same as left shift of 3
  - 00001010 << 3 = 01010000 = 80

Note: In the sequel, for simplicity, we write $\times 2^k$.
This should be read as an $O(1)$ time left shift **<< k**.

Divide-and-Conquer Multiplication: First Attempt

$$\begin{aligned}(75)(218) &= (4 \cdot 2^4 + 11)(13 \cdot 2^4 + 10) \\ &= 4 \cdot 13 \cdot 2^8 + (4 \cdot 10 + 11 \cdot 13) 2^4 + 11 \cdot 10 \\ &= 52 \cdot 2^8 + 183 \cdot 2^4 + 110 \\ &= 16{,}350\end{aligned}$$

$$\begin{aligned}\mathbf{0100\ 1011} \times \mathbf{1101\ 1010} &= (\mathbf{0100} \cdot 2^4 + \mathbf{1011}) \times (\mathbf{1101} \cdot 2^4 + \mathbf{1010}) \\ &= (\mathbf{0100} \times \mathbf{1101})\, 2^8 \\ &\quad + ((\mathbf{0100} \times \mathbf{1010}) + (\mathbf{1011} \times \mathbf{1101}))\ 2^4 \\ &\quad + \mathbf{1011} \times \mathbf{1010}\end{aligned}$$

In general:

- Let $a = a_1 2^{n/2} + a_0$, and $b = b_1 2^{n/2} + b_0$,
  where a_1, a_0, b_1, b_0 are all $(n/2)$-bit integers.

$$\Longrightarrow \quad ab = a_1 b_1 2^n + (a_1 b_0 + a_0 b_1) 2^{n/2} + a_0 b_0$$

The first divide-and-conquer algorithm for integer multiplication

Suppose the bits are stored in arrays $A[1..n]$ and $B[1..n]$, $A[1]$ and $B[1]$ are the least significant bits

<u>Multiply</u>(A, B)**:**
$n \leftarrow$ **size of** A
if $n = 1$ **then return** $A[1] \cdot B[1]$
$mid \leftarrow \lfloor n/2 \rfloor$
$U \leftarrow$ **Multiply**$(A[mid+1..n], B[mid+1..n])$ % $a_1 b_1$
$V \leftarrow$ **Multiply**$(A[mid+1..n], B[1..mid])$ % $a_1 b_0$
$W \leftarrow$ **Multiply**$(A[1..mid], B[mid+1..n])$ % $a_0 b_1$
$Z \leftarrow$ **Multiply**$(A[1..mid], B[1..mid])$ % $a_0 b_0$
$M[1..2n] \leftarrow 0$
$M[1..n] \leftarrow Z$ % $a_0 b_0$
$M[mid+1..] \leftarrow M[mid+1..] \oplus V \oplus W$ % + $(a_1 b_0 + a_0 b_1) \ll n/2$
$M[2mid+1..] \leftarrow M[2mid+1..] \oplus U$ % + $a_1 b_1 \ll n$
return M

$\oplus$: denotes the integer addition algorithm

Analysis (Expansion Method)

Recurrence.

For, $n > 1, \quad T(n) = 4T(n/2) + n. \qquad T(1) = 1$

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

$$= 4\left(4T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n \qquad = 4^2\ T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n$$

$$= 4^2\ \left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n \qquad = 4^3\ T\left(\frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n$$

$$= 4^3\ \left(4T\left(\frac{n}{2^4}\right) + \frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n \qquad = 4^4\ T\left(\frac{n}{2^4}\right) + \left(\frac{4^3}{2^3} + \frac{4^2}{2^2} + \frac{4}{2} + 1\right)n$$

....

$$= 4^i\ T\left(\frac{n}{2^i}\right) + \left(\frac{4^{i-1}}{2^{i-1}} + \frac{4^{i-2}}{2^{i-2}} + \cdots + \frac{4}{2} + 1\right)n$$

Analysis (Expansion Method)

$$T(n) = 4^i\, T\left(\frac{n}{2^i}\right) + \left(\frac{4^{i-1}}{2^{i-1}} + \frac{4^{i-2}}{2^{i-2}} + \cdots + \frac{4}{2} + 1\right) n$$

....

$$= 4^h\, T\left(\frac{n}{2^h}\right) + \left(\left(\frac{4}{2}\right)^{h-1} + \left(\frac{4}{2}\right)^{h-2} + \cdots + \left(\frac{4}{2}\right) + 1\right) n$$

$$= \boldsymbol{n^2}\, T\left(\frac{n}{\boldsymbol{n}}\right) + \left(2^{h-1} + 2^{h-2} + \ldots + 2 + 1\right) n$$

$$= n^2\, T(1) + \left(\frac{2^h - 1}{2 - 1}\right) n$$

$$= n^2 + \left(\frac{\boldsymbol{n} - 1}{1}\right) n$$

$$\boxed{= n^2 + n(n-1) = \Theta(n^2)}$$

$$h = \log_2 n$$

$$2^h = n$$

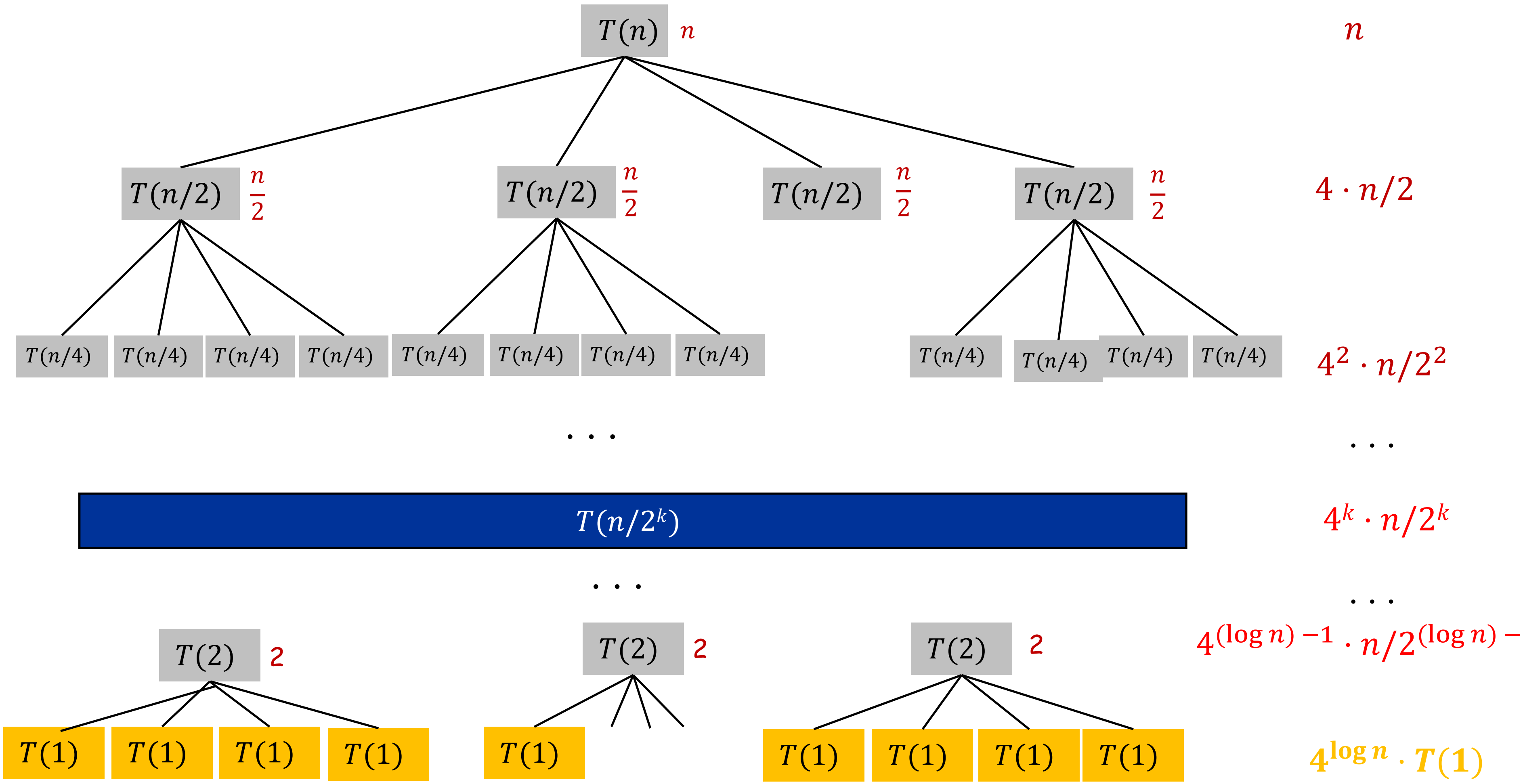
$$4^h = (2^2)^h = \left(2^h\right)^2 = n^2$$

Analysis (Tree Method)

Recurrence:

$$T(n) = 4T(n/2) + n; \quad T(1) = 1$$

Solve the recurrence:



Analysis (Tree Method)

$$n + \left(\frac{4}{2}\right) n + \left(\frac{4}{2}\right)^{2} n + \cdots + \ + \left(\frac{4}{2}\right)^{(\log n)-1} n \ + 4^{\log n}\, T(1)$$

$$= n(1 + 2 + 2^2 \ + \ 2^3 \ + \cdots + \ 2^{(\log n)\ -1}) + 4^{\log n}\, T(1)$$

$$= n\left(\frac{2^{\log n} \ - 1}{2 - 1}\right) + \ 4^{\log n}\, T(1) = \quad n(n-1) + 2^{2\log n}\, T(1)$$

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

- The divide-and-conquer algorithm is as bad as the primary school method
- Essentially, the algorithm still multiplies every bit of A with every bit of B.
- Compared with merge sort, the key difference is that one problem generates 4 subproblems of size $\boldsymbol{n/2}$.

Karatsuba Multiplication

New Observation:

- **Let X,a, b, c, d be integers**
- **Simple algebra said**

$$(a\boldsymbol{X} + b)\,(c\boldsymbol{X} + d) = ac\boldsymbol{X}^2 + (\boldsymbol{ad} + \boldsymbol{bc})\boldsymbol{X} + bd$$

- This used **4** multiplications to find the three coefficients ac, (ad +bc), bd

- We will now see how to find these 3 coefficients using only **3** multiplications

- Calculate ac, bd, and A= (a+b) (c+d)
- Notice that **ad + bc = A - ac -bd**

- So, we can calculate the three coefficients using only **3** multiplications (and one more addition and two subtractions)

Karatsuba Multiplication

- Let $a = a_1 2^{n/2} + a_0$, and $b = b_1 2^{n/2} + b_0$
  where a_1, a_0, b_1, b_0 are all $(n/2)$-bit integers.

- We already saw

$$ab = a_1 b_1 2^n + \boldsymbol{(a_1 b_0 + a_0 b_1)} 2^{n/2} + a_0 b_0$$

- Use the trick from previous page:

$$\boldsymbol{a_1 b_0 + a_0 b_1} = (a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0$$

Calculating ab now only requires performing 3 multiplication subproblems of size $n/2$!

Karatsuba's multiplication algorithm

<u>Multiply(A, B):</u>
$n \leftarrow$ **size of** A
if $n = 1$ **then return** $A[1] \cdot B[1]$
$mid \leftarrow \lfloor n/2 \rfloor$
$U \leftarrow$ **Multiply(**$A[mid+1..n], B[mid+1..n]$**)**
$Z \leftarrow$ **Multiply(**$A[1..mid], B[1..mid]$**)**
$A' \leftarrow A[mid+1..n] \oplus A[1..mid]$
$B' \leftarrow B[mid+1..n] \oplus B[1..mid]$
$Y \leftarrow$ **Multiply(**A', B'**)**
$M[1..2n] \leftarrow 0$
$M[1..n] \leftarrow M[1..n] \oplus Z$
$M[mid+1..] \leftarrow M[mid+1..] \oplus Y \ominus U \ominus Z$
$M[2mid+1..] \leftarrow M[2mid+1..] \oplus U$
return M

$\oplus$ $\ominus$: denotes the integer addition/subtraction algorithm

Analysis (Expansion Method)

Recurrence.

For, $n > 1$, $T(n) = 3T(n/2) + n.$ $T(1) = 1$

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

$$= 3\left(3T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n \qquad = 3^2\ T\left(\frac{n}{2^2}\right) + \frac{3}{2}n + n$$

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

$$= 3^3\ \left(3T\left(\frac{n}{2^4}\right) + \frac{n}{2^3}\right) + \frac{3^2}{2^2}n + \frac{3}{2}n + n \qquad = 3^4\ T\left(\frac{n}{2^4}\right) + \left(\frac{3^3}{2^3} + \frac{3^2}{2^2} + \frac{3}{2} + 1\right)n$$

....

$$= 3^i\ T\left(\frac{n}{2^i}\right) + \left(\frac{3^{i-1}}{2^{i-1}} + \frac{3^{i-2}}{2^{i-2}} + \cdots + \frac{3}{2} + 1\right)n$$

Analysis (Expansion Method)

$$T(n) = 3^i\, T\left(\frac{n}{2^i}\right) + \left(\frac{3^{i-1}}{2^{i-1}} + \frac{3^{i-2}}{2^{i-2}} + \cdots + \frac{3}{2} + 1\right)n$$

....

$$= 3^h\, T\left(\frac{n}{2^h}\right) + \left(\left(\frac{3}{2}\right)^{h-1} + \left(\frac{3}{2}\right)^{h-2} + \cdots + \left(\frac{3}{2}\right) + 1\right)n$$

$$= 3^h\, T\left(\frac{n}{n}\right) + \left(\frac{\left(\frac{3}{2}\right)^h - 1}{\frac{3}{2} - 1}\right)n$$

$$= 3^h\, T(1) + \; 2\left(\left(\frac{3}{2}\right)^h - 1\right)n$$

$$= 3^h \; + \; 2\left(\frac{3^h}{2^h} - 1\right)n$$

$$= 3^h \; + \; 2\left(\frac{3^h}{n} - 1\right)n$$

$$= 3 \cdot \mathbf{3^{\log_2 n}} - 2n$$

$$= 3 \cdot \boldsymbol{n^{\log_2 3}} - 2n$$

$$\boxed{= \Theta\left(\boldsymbol{n^{1.585\ldots}}\right)}$$

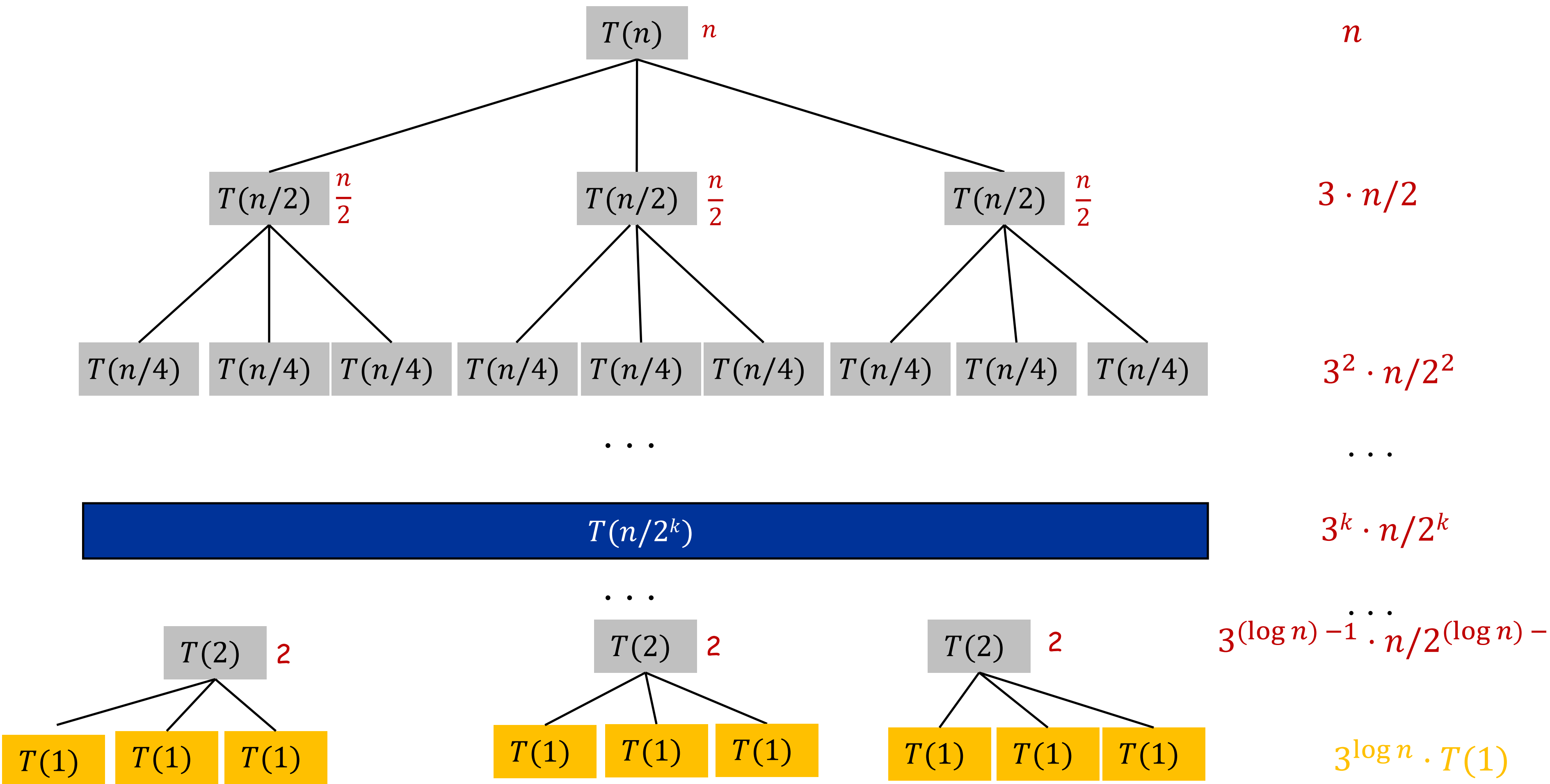
$h = \log_2 n$ $\quad$ $2^h = n$ $\quad$ *Recall* $3^{\log_2 n} = \left(2^{\log_2 3}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 3} = n^{\log_2 3}$

Analysis

Recurrence:

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

Solve the recurrence:



Analysis (continued)

$$T(n) = n + \left(\frac{3}{2}\right)^1 n + \left(\frac{3}{2}\right)^2 n + \cdots + \ + \left(\frac{3}{2}\right)^{(\log n)-1} n \ + 3^{\log n}\, T(1)$$

$$= n\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \left(\frac{3}{2}\right)^3 + \cdots + \left(\frac{3}{2}\right)^{(\log n)\,-1}\right) + 3^{\log n}\, T(1)$$

$$= \ n\left(\frac{\left(\frac{3}{2}\right)^{\log n} - 1}{\frac{3}{2} - 1}\right) + \ 3^{\log n}\, T(1)$$

$$= n\ \Theta\left(\frac{3^{\log n}}{2^{\log n}}\right) + \Theta\left(3^{\log n}\right)$$

Recall

$$3^{\log n} \ = \ \left(2^{\log 3}\right)^{\log n} \ = \ \left(2^{\log n}\right)^{\log 3} \ = n^{\log 3}$$

$$= n\ \Theta\left(\frac{n^{\log 3}}{n}\right) + \Theta\left(n^{\log 3}\right) = \ \Theta\left(n^{\log 3}\right)$$

$$\boxed{= \boldsymbol{\Theta}\left(\boldsymbol{n}^{\mathbf{1.585\ldots}}\right)}$$

Analysis (continued)

Recurrence For First D&C Algorithm

$$T(n) = 4T(n/2) + n; \quad T(1) = 1$$

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

Recurrence For Karatsuba Multiplication

$$T(n) = 3T(n/2) + n; \quad T(1) = 1$$

Solution: $T(n) = \Theta(n^{1.585\ldots})$

Analysis (continued)

Karatsuba Multiplication:

- Dividing each integer into 2 parts, and solve 3 subproblems
  - $T(n) = 3T(n/2) + n$, $T(n) = \Theta\left(n^{\log_2 3}\right) = \Theta(n^{1.585\ldots})$

Progressive improvements:

- Dividing each integer into 3 parts, and solve 5 subproblems
  - $T(n) = 5T(n/3) + n$, $T(n) = \Theta\left(n^{\log_3 5}\right) = \Theta(n^{1.465})$
- Dividing each integer into 4 parts, and solve 7 subproblems
  - $T(n) = 7T(n/4) + n$, $T(n) = \Theta\left(n^{\log_4 7}\right) = \Theta(n^{1.404})$
- …
- An $\Theta(n \log n \log \log n)$ algorithm (based on FFT)
- An $\Theta(n \log n \log \log \log n)$ algorithm
- The fastest algorithm runs in time $O\left(n \log n \, 2^{\Theta(log^* n)}\right)$
  - $log^* n$ is a VERY slow growing function
- The conjecture is that the problem can be solved in $\Theta(n \log n)$ time. This conjecture is still open.

Integer Multiplication in Practice

Work on the word level

- Example (using 16-bit words):
  - Decimal: `1316103040073424382`
  - Hexadecimal: `1243 BCBD EF63 5DFE`
  - Stored using an array of 4 words

In practice:

- Long multiplication: Best for < 20 words
- Karatsuba's algorithm: Best for 20 ~ 2000 words
- FFT based algorithm: Best for > 2000 words

The Master Theorem (proof coming soon)

Theorem: Let $a \geq 1, b > 1, c \geq 0$ be constants. The recurrence $T(n) = aT(n/b) + n^c$ have the following solutions.

- Case 1: $c < \log_b a$: $T(n) = \Theta\left(n^{\log_b a}\right)$.
- Case 2: $c = \log_b a$: $T(n) = \Theta(n^c \log n)$.
- Case 3: $c > \log_b a$: $T(n) = \Theta(n^c)$.

Examples: We have already seen Cases 1 & 2. Case 3 will arise later

- Case 1: $T(n) = 3T\left(\frac{n}{2}\right) + n$ => $T(n) = \Theta\left(n^{\log_2 3}\right)$
- Case 2: $T(n) = 2T\left(\frac{n}{2}\right) + n$ => $T(n) = \Theta(n \log n)$
- Case 3: $T(n) = 2T\left(\frac{n}{3}\right) + n$ => $T(n) = \Theta(n)$

Matrix Multiplication

Matrix multiplication. Given two n-by-n matrices A and B, compute $C = AB$.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

Brute force. $\Theta(n^3)$ time.

Fundamental question. Can we improve upon brute force?

Matrix Multiplication: First Attempt

Divide-and-conquer.

- Divide: partition A and B into $\frac{1}{2}n$-by-$\frac{1}{2}n$ blocks.
- Conquer: multiply 8 $\frac{1}{2}n$-by-$\frac{1}{2}n$ submatrices recursively.
- Combine: add appropriate products using 4 matrix additions.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

$$T(n) = 8\mathrm{T}(n/2) + O(n^2) \quad \Longrightarrow \quad T(n) = O(n^3)$$

Recursive calls

Add, form submatrices

Strassen's Matrix Multiplication Algorithm

Key idea. multiply 2-by-2 block matrices with only 7 multiplications.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_5 + P_1 - P_3 - P_7 \end{aligned}$$

$$\begin{aligned} P_1 &= A_{11} \times (B_{12} - B_{22}) \\ P_2 &= (A_{11} + A_{12}) \times B_{22} \\ P_3 &= (A_{21} + A_{22}) \times B_{11} \\ P_4 &= A_{22} \times (B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\ P_6 &= (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\ P_7 &= (A_{11} - A_{21}) \times (B_{11} + B_{12}) \end{aligned}$$

- 7 multiplications of $\frac{1}{2}n$-by-$\frac{1}{2}n$ submatrices.
- $\Theta(n^2)$ additions and subtractions.
- $T(n) = 7T(n/2) + n^2 \implies T(n) = \Theta\left(n^{\log_2 7}\right) = \Theta(n^{2.807})$

In practice: Used to multiply large matrices (e.g., $n > 100$)

Fast Matrix Multiplication in Theory

Q. Multiply two 2-by-2 matrices with only 7 multiplications?
A. Yes! $\Theta(n^{2.807})$ [Strassen, 1969]

Q. Multiply two 2-by-2 matrices with only 6 multiplications?
A. Impossible.

Q. Two 3-by-3 matrices with only 21 multiplications?
A. Also impossible.

Q. Two 70-by-70 matrices with only 143,640 multiplications?
A. Yes! $\Theta(n^{2.795})$

The competition continues...

- $\Theta(n^{2.376})$ [Coppersmith-Winograd, 1990.]
- $\Theta(n^{2.374})$ [Stothers, 2010.]
- $\Theta(n^{2.3728642})$ [Williams, 2011.]
- $\Theta(n^{2.3728639})$ [Le Gall, 2014.]
- Conjecture: close to $\Theta(n^{2})$