

# Analysis of Algorithms

DIVIDE AND CONQUER

## Integer Multiplication

- ★ **Input:** integers  $I$  and  $J$  each represented by  $n$  bits.
- ★  $I + J$  and  $I - J$  can be computed with  $O(n)$  bit operations.
- ★  $I \cdot J$  can be computed with  $O(n^2)$  bit operations using the **direct** algorithm.
- ★ Using **divide-and-Conquer**,  $I \cdot J$  can be computed with  $O(n^{\log_2 3}) \approx O(n^{1.585})$  bit operations.
- ★ Using the **Fast-Transform-Fourier**,  $I \cdot J$  can be computed with  $O(n \log n)$  bit operations.
- ★  $I + J$ ,  $I - J$ , and  $I \cdot J$  need at least  $\Omega(n)$  bit operations.

## Ideas for Divide and Conquer

**Assumption:**  $n$ , the length of  $I$  and  $J$  is a power of 2.

**Input representation:**

$$\star I = I_h \cdot 2^{n/2} + I_\ell.$$

$$\star J = J_h \cdot 2^{n/2} + J_\ell.$$

**Claim:** Multiplying  $m$ -bit number by  $2^k$  can be done with  $\Theta(k + m)$  bit operations (**shifting**).

**Objective:** Compute the product  $I \cdot J$  using multiplications on  $I_h, I_\ell, J_h, J_\ell$  whose lengths are  $n/2$  bits.

## Divide and Conquer I

$$\star I = I_h \cdot 2^{n/2} + I_\ell.$$

$$\star J = J_h \cdot 2^{n/2} + J_\ell.$$

$$\star I \cdot J = I_h J_h 2^n + (I_h J_\ell + I_\ell J_h) 2^{n/2} + I_\ell J_\ell.$$

$$\star T(n) = 4T(n/2) + \Theta(n) \text{ bit operations.}$$

$$\star T(n) = \Theta(n^2) \text{ bit operations.}$$

**Result:** Not an **improvement**.

## Solving the Recursion

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

- ★  $a = 4$ .
- ★  $b = 2$ .
- ★  $\log_b(a) = 2$ .
- ★  $d = 1$ .
- ★  $d < \log_b(a)$ .

**Master Theorem Case 1:**  $T(n) = \Theta(n^2)$ .

## Divide and Conquer II

- ★  $(I_h - I_\ell)(J_\ell - J_h) = I_h J_\ell - I_h J_h - I_\ell J_\ell + I_\ell J_h.$
- ★  $I_h J_\ell + I_\ell J_h = (I_h - I_\ell)(J_\ell - J_h) + I_h J_h + I_\ell J_\ell.$
- ★  $A = I_h J_h.$
- ★  $B = I_\ell J_\ell.$
- ★  $C = (I_h - I_\ell)(J_\ell - J_h).$
- ★  $I \cdot J = I_h J_h 2^n + (I_h J_\ell + I_\ell J_h) 2^{n/2} + I_\ell J_\ell.$
- ★  $I \cdot J = A 2^n + (C + A + B) 2^{n/2} + B.$

## Divide and Conquer I and II

### Divide and Conquer I:

$$\star I \cdot J = I_h J_h 2^n + (I_h J_\ell + I_\ell J_h) 2^{n/2} + I_\ell J_\ell.$$

$$\star T(n) = 4T(n/2) + \Theta(n) = \Theta(n^2) \text{ bit operations.}$$

### Divide and Conquer II:

$$\star I \cdot J = A 2^n + (C + A + B) 2^{n/2} + B.$$

$$\star T(n) = 3T(n/2) + \Theta(n) = \Theta(n^{\log_2 3}) \text{ bit operations.}$$

**Result:**  $\Theta(n^{\log_2 3}) = \Theta(n^{1.585})$  is **better** than  $\Theta(n^2)$ .

## Solving the Recursion

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

- ★  $a = 3$ .
- ★  $b = 2$ .
- ★  $\log_b(a) \approx 1.585$ .
- ★  $d = 1$ .
- ★  $d < \log_b(a)$ .

**Master Theorem Case 1:**  $T(n) = \Theta(n^{\log_2 3})$ .

Arbitrary  $n \geq 1$

## Algorithm:

- ★  $2^{k-1} < n \leq 2^k \Rightarrow 2^k < 2n$ .
- ★ **Add**  $2^k - n$  zeros in front of both integers.
- ★ **Run** the algorithm with the new integers.
- ★ **Omit** zeros at the beginning of the product.

## Complexity:

- ★  $(2^k)^{\log_2 3} < (2n)^{\log_2 3} = 3n^{\log_2 3}$
- ★ The algorithm complexity  $\Theta((2^k)^{\log_2 3})$  is  $\Theta(n^{\log_2 3})$ .

# Matrix Multiplication

**Input:** 2 matrices  $A$  and  $B$  of **scalars** of size  $n \times n$  ( $n \geq 1$ ).

**Output:** An  $n \times n$  matrix  $C = A \times B$ .

**Definition:**  $c_{i,j} = \sum_{k=1}^n a_{i,k} \cdot b_{k,j}$ , for  $1 \leq i, j, \leq n$ .

**Operation:** An **addition** (**subtraction**) or a **multiplication** between 2 scalars.

## Matrix Multiplication – Algorithms

**Direct algorithm:**  $n^3$  multiplications and  $n^2(n-1)$  additions.

**Strassen algorithm:**  $O(n^{\log_2 7}) \approx O(n^{2.81})$  operations.

**Best known solution:**  $O(n^{2.376})$  operations.

**Lower bound:**  $\Omega(n^2)$  operations.

## Multiplying $2 \times 2$ Matrices

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

The values of  $r, s, t, u$ :

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

## The Magic Idea

★ Compute 7 help variables:

$$p_1 = a(g - h)$$

$$p_2 = (a + b)h$$

$$p_3 = (c + d)e$$

$$p_4 = d(f - e)$$

$$p_5 = (a + d)(e + h)$$

$$p_6 = (b - d)(f + h)$$

$$p_7 = (a - c)(e + g)$$

★ 7 multiplications and 10 additions.

## The Magic Idea

★ Compute  $r, s, t, u$ :

$$r = p_5 + p_4 - p_2 + p_6$$

$$s = p_1 + p_2$$

$$t = p_3 + p_4$$

$$u = p_5 + p_1 - p_3 - p_7$$

★ 8 more additions.

★ **Total** of 7 multiplications and 18 additions.

## Verifying the Value of $r$

$$\begin{aligned}r &= p_5 + p_4 - p_2 + p_6 \\&= (a + d)(e + h) + d(f - e) - (a + b)h + (b - d)(f + h) \\&= ae + ah + de + dh + df - de - ah - bh + bf + bh - df - dh \\&= ae + a/h + d/e + d/h + d/f - d/e - a/h - b/h + bf + b/h - d/f - d/h \\&= ae + bf\end{aligned}$$

## Verifying the Value of $s$

$$\begin{aligned} s &= p_1 + p_2 \\ &= a(g - h) + (a + b)h \\ &= ag - ah + ah + bh \\ &= ag - \cancel{a/h} + \cancel{a/h} + bh \\ &= ag + bh \end{aligned}$$

## Verifying the Value of $t$

$$\begin{aligned}t &= p_3 + p_4 \\&= (c + d)e + d(f - e) \\&= ce + de + df - de \\&= ce + \cancel{de} + df - \cancel{de} \\&= ce + df\end{aligned}$$

## Verifying the Value of $u$

$$\begin{aligned}u &= p_5 + p_1 - p_3 - p_7 \\&= (a + d)(e + h) + a(g - h) - (c + d)e - (a - c)(e + g) \\&= ae + ah + de + dh + ag - ah - ce - de - ae - ag + ce + cg \\&= \cancel{ae} + \cancel{ah} + \cancel{de} + dh + \cancel{ag} - \cancel{ah} - \cancel{ce} - \cancel{de} - \cancel{ae} - \cancel{ag} + \cancel{ce} + cg \\&= cg + dh\end{aligned}$$

## Divide-and-Conquer

**Lemma:** The magic idea works for sub-matrices as well.

**Algorithm:** (for  $n = 2^k$ )

- ★ **Partition**  $A$  and  $B$  into 4 sub-matrices of size  $\frac{n}{2} \times \frac{n}{2}$ .
- ★ **Recursively compute** the 7 sub-matrices multiplications.
- ★ **Do** the 18 matrices additions each with  $(n/2)^2$  addition operations.

**Time complexity:**

$$- T(n) = 7T(n/2) + \Theta(n^2) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.81}).$$

## Solving the Recursion

$$T(n) = 7T(n/2) + \Theta(n^2)$$

- ★  $a = 7$ .
- ★  $b = 2$ .
- ★  $\log_b(a) \approx 2.81$ .
- ★  $d = 2$ .
- ★  $d < \log_b(a)$ .

**Master Theorem Case 1:**  $T(n) = \Theta(n^{\log_2 7})$ .

Arbitrary  $n \geq 1$

### Algorithm:

- ★  $2^{k-1} < n \leq 2^k \Rightarrow 2^k < 2n$ .
- ★ Add  $2^k - n$  zero columns and rows to both  $A$  and  $B$ .
- ★ Run the algorithm for the new matrices.
- ★ Omit the zero columns and rows from  $C$ .

### Complexity:

- ★  $(2^k)^{\log_2 7} < (2n)^{\log_2 7} = 7n^{\log_2 7}$ .
- ★ The algorithm complexity  $\Theta((2^k)^{\log_2 7})$  is  $\Theta(n^{\log_2 7})$ .