

CH-231-A

**Algorithms and Data Structures**

ADS

**Lecture 12**

Dr. Kinga Lipskoch

Spring 2024

# Matrix Multiplication: Strassen's Idea (1)

Strassen's idea:

Multiply matrices with 7 multiplications and 18 additions.

$$\begin{aligned}
 P_1 &= a \cdot (f - h) & r &= P_5 + P_4 - P_2 + P_6 \\
 P_2 &= (a + b) \cdot h & s &= P_1 + P_2 \\
 P_3 &= (c + d) \cdot e & t &= P_3 + P_4 \\
 P_4 &= d \cdot (g - e) & u &= P_5 + P_1 - P_3 - P_7 \\
 P_5 &= (a + d) \cdot (e + h) \\
 P_6 &= (b - d) \cdot (g + h) \\
 P_7 &= (a - c) \cdot (e + f)
 \end{aligned}$$

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

$$C = A \cdot B$$

# Matrix Multiplication: Strassen's Idea (2)

Strassen's idea:

$$\begin{aligned}
 P_1 &= a \cdot (f - h) & r &= P_5 + P_4 - P_2 + P_6 \\
 P_2 &= (a + b) \cdot h & &= (a + d)(e + h) \\
 P_3 &= (c + d) \cdot e & &+ d(g - e) - (a + b)h \\
 P_4 &= d \cdot (g - e) & &+ (b - d)(g + h) \\
 P_5 &= (a + d) \cdot (e + h) & &= ae + ah + de + dh \\
 P_6 &= (b - d) \cdot (g + h) & &+ dg - de - ah - bh \\
 P_7 &= (a - c) \cdot (e + f) & &+ bg + bh - dg - dh \\
 & & &= ae + bg
 \end{aligned}$$

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

$$C = A \cdot B$$

# Matrix Multiplication: Strassen's Algorithm

Strassen's algorithm:

1. **Divide:**

Partition  $A$  and  $B$  into  $(n/2) \times (n/2)$  submatrices.

Form terms to be multiplied using  $+$  and  $-$ .

2. **Conquer:**

Perform 7 multiplications of  $(n/2) \times (n/2)$  submatrices recursively.

3. **Combine:**

Form  $C$  using  $+$  and  $-$  on  $(n/2) \times (n/2)$  submatrices.

# Matrix Multiplication: Complexity

Complexity of Strassen's algorithm:

Recurrence:

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

$$n^{\log_b a} = n^{\log_2 7} = n^{2.81}$$

Case 1:  $T(n) = \Theta(n^{\lg 7})$

2.81 may not seem much smaller than 3, but the difference is in the exponent, therefore the impact on running time is significant.

Strassen's algorithm beats the standard algorithm for  $n \geq 32$  or so.

## Matrix Multiplication: More

### Best known algorithm:

Latest improvement in 2020 in the following publication:

*Alman, Josh; Williams, Virginia Vassilevska (2020), "A Refined Laser Method and Faster Matrix Multiplication", 32<sup>nd</sup> Annual ACM-SIAM Symposium on Discrete Algorithms*

$$T(n) = O(n^{2.3728596})$$

- ▶ Only of theoretical interest.
- ▶ Most approaches that are faster than Strassen's are not used in practice.
- ▶ They are only faster for very large  $n$ .
- ▶ One cannot get better than  $O(n^2)$ , cf. [Case 3](#).

## Intermediate Conclusion

- ▶ Definitions
- ▶ First example of an algorithm – Insertion Sort
- ▶ Asymptotic analysis
- ▶ First powerful concept – Divide & Conquer
- ▶ Solve recurrences for analysis

## Recall: Sorting Problem

- ▶ Input:
  - ▶ Sequence  $\langle a_1, a_2, \dots, a_n \rangle$  of numbers
- ▶ Output:
  - ▶ Permutation  $\langle a'_1, a'_2, \dots, a'_n \rangle$
  - ▶ Such that  $a'_1 \leq a'_2 \leq \dots \leq a'_n$



## Recall: Insertion & Merge Sort

Time complexity:

	Insertion Sort	Merge Sort
Best case	$\Theta(n)$	$\Theta(n \lg n)$
Average case	$\Theta(n^2)$	$\Theta(n \lg n)$
Worst case	$\Theta(n^2)$	$\Theta(n \lg n)$

Visualizations:

<http://www.sorting-algorithms.com/insertion-sort>

<http://www.sorting-algorithms.com/merge-sort>

What about storage space complexity?

# In-situ Sorting

- ▶ Definition:
  - ▶ In-situ algorithms refer to algorithms that operate with  $\Theta(1)$  memory
- ▶ In-situ sorting:
  - ▶ Sorting algorithms that need only a constant number of additional storings
- ▶ Insertion Sort:
  - ▶ In-situ sorting
- ▶ Merge Sort:
  - ▶ Not in-situ sorting