Fast Matrix Multiplication

Shusen Wang

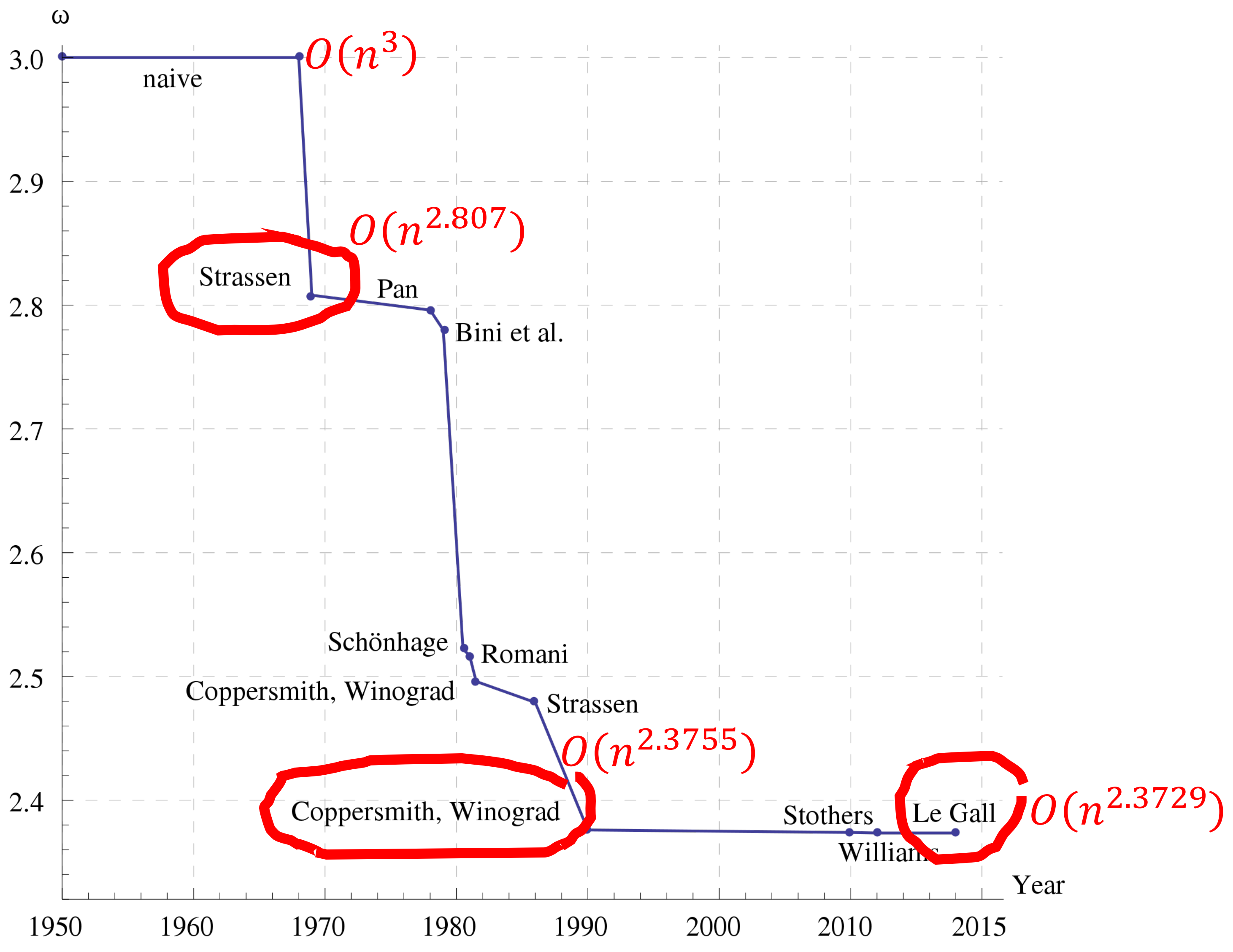
Matrix Multiplication

- Let $\mathbf{A}$ and $\mathbf{B}$ be $n \times n$ matrices.
- The multiplication $\mathbf{C} = \mathbf{AB}$ cost $O(n^3)$ time (nested for-loop).
- Can the time complexity be lower?

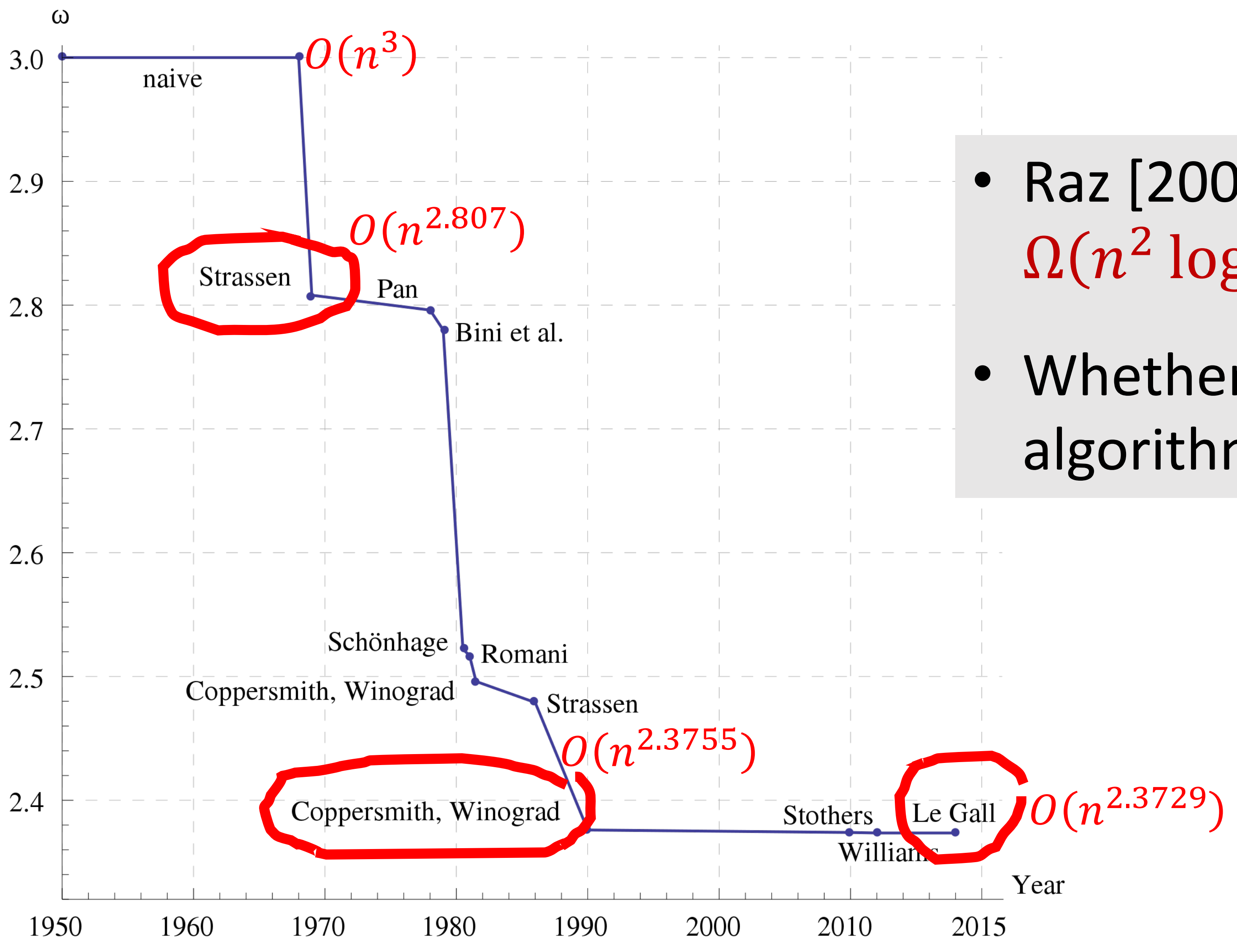
Fast Matrix Multiplication

- Strassen algorithm [1969] has $O(n^{2.807})$ time complexity.
- Coppersmith–Winograd algorithm [1990] has $O(n^{2.3755})$ time complexity.
- Le Gall [2014] improves the time complexity to $O(n^{2.3729})$.

Fast Matrix Multiplication



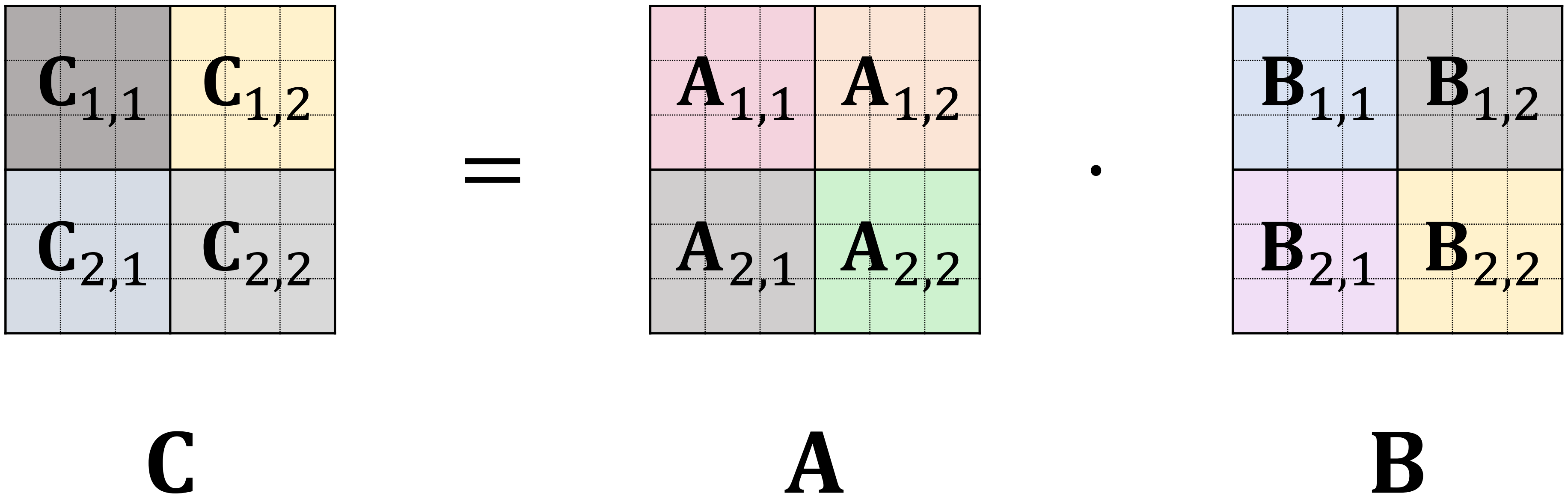
Fast Matrix Multiplication



- Raz [2002] proved a lower bound $\Omega(n^2 \log n)$.
- Whether there exists an $O(n^2 \log n)$ algorithm is yet unknown.

Divide-and-Conquer Matrix Multiplication

Block Matrix Multiplication



Block Matrix Multiplication

$$\begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \cdot \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

- $C_{1,1} =$

Block Matrix Multiplication

$$\begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \cdot \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

- $C_{1,1} = A_{1,1}B_{1,1} +$

Block Matrix Multiplication

$$\begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

- $\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}.$

Block Matrix Multiplication

$$\begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

- $\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}$.
- $\mathbf{C}_{1,2} = \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2}$.

Block Matrix Multiplication

$$\begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

- $\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}$.
- $\mathbf{C}_{1,2} = \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2}$.
- $\mathbf{C}_{2,1} = \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}$.

Block Matrix Multiplication

$$\begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

- $\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}.$
- $\mathbf{C}_{1,2} = \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2}.$
- $\mathbf{C}_{2,1} = \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}.$
- $\mathbf{C}_{2,2} = \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}.$

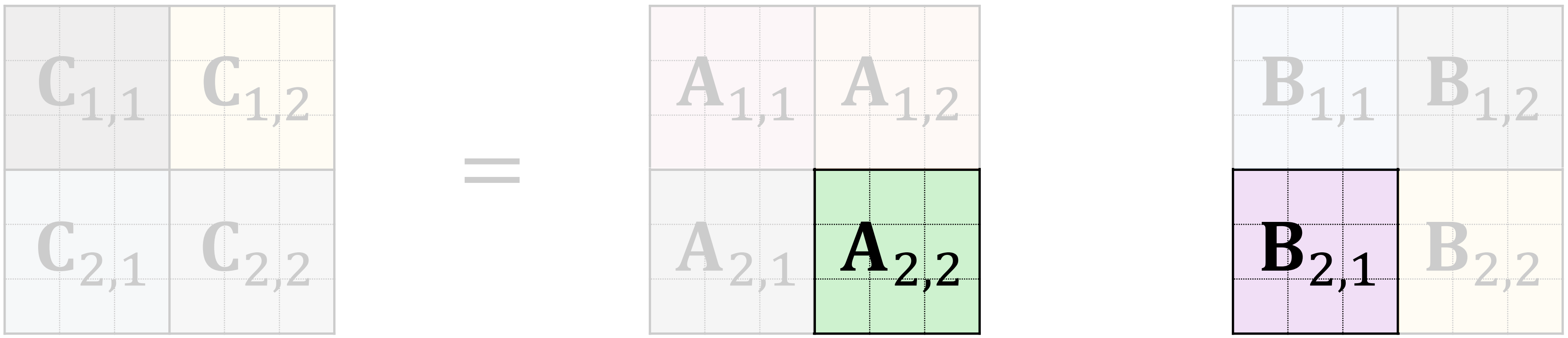
Block Matrix Multiplication

$$\begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

- $\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1}.$
- $\mathbf{C}_{1,2} = \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2}.$
- $\mathbf{C}_{2,1} = \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}.$
- $\mathbf{C}_{2,2} = \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}.$

- 8 matrix multiplications
- 4 matrix additions

Divide-and-Conquer



- Further partition $\mathbf{A}_{ij}$ and $\mathbf{B}_{jl}$ into sub-matrices.
- Recursively apply the block multiplication to compute $\mathbf{A}_{ij}\mathbf{B}_{jl}$.

Time Complexity

- Partition the $n\times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ into $\frac{n}{2}\times\frac{n}{2}$ sub-matrices.
- To compute the $n\times n$ matrix multiplication $\mathbf{C} = \mathbf{AB}$, perform
  - $\frac{n}{2}\times\frac{n}{2}$ matrix multiplications for 8 times,
  - $\frac{n}{2}\times\frac{n}{2}$ matrix additions for 4 times.

Time Complexity

- $T(n)$: time complexity for multiplying $n \times n$ matrices.
- Repeating $\frac{n}{2} \times \frac{n}{2}$ matrix multiplications for 8 times costs time:

$$8 \cdot T\left(\frac{n}{2}\right).$$

- Matrix additions cost $c \cdot n^2$ time. (c is a constant.)

Recurrence relation: $T(n) = 8 \cdot T(n/2) + c \cdot n^2.$

The Master Theorem

Recurrence relation: $T(n) = a \cdot T(n/b) + c \cdot n^d$.

- The master theorem:

$$T(n) = \begin{cases} O\left(n^d\right), & \text{if } a < b^d; \\ O\left(n^d \log n\right), & \text{if } a = b^d; \\ O\left(n^{\log_b a}\right), & \text{if } a > b^d. \end{cases}$$

- For block matrix multiplication, $a = 8$, $b = 2$, and $d = 2$.

The Master Theorem

Recurrence relation: $T(n) = a \cdot T(n/b) + c \cdot n^d$.

- The master theorem:

$$T(n) = \begin{cases} O(n^d), & \text{if } a < b^d; \\ O(n^d \log n), & \text{if } a = b^d; \\ O(n^{\log_b a}), & \text{if } a > b^d. \end{cases}$$

- For block matrix multiplication, $a = 8$, $b = 2$, and $d = 2$.
- Thus, $T(n) = O(n^{\log_2 8}) \approx O(n^3)$.
- No speedup at all.

Strassen Algorithm

Strassen Algorithm

$$\underbrace{\begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}}_{\mathbf{C}} = \underbrace{\begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}}_{\mathbf{B}}$$

Strassen Algorithm

- $\mathbf{M}_1 = (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2})$.
- $\mathbf{M}_2 = (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\,\mathbf{B}_{1,1}$.
- $\mathbf{M}_3 = \mathbf{A}_{1,1}\,(\mathbf{B}_{1,2} - \mathbf{B}_{2,2})$.
- $\mathbf{M}_4 = \mathbf{A}_{2,2}\,(\mathbf{B}_{2,1} - \mathbf{B}_{1,1})$.
- $\mathbf{M}_5 = (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\,\mathbf{B}_{2,2}$.
- $\mathbf{M}_6 = (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2})$.
- $\mathbf{M}_7 = (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})$.

- $\mathbf{C}_{1,1} = \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7$.
- $\mathbf{C}_{1,2} = \mathbf{M}_3 + \mathbf{M}_5$.
- $\mathbf{C}_{2,1} = \mathbf{M}_2 + \mathbf{M}_4$.
- $\mathbf{C}_{2,2} = \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6$.

- 7 matrix multiplications
- 18 matrix additions

Strassen Algorithm

- Divide-and-conquer: recursively compute $\mathbf{A}_{ij}\mathbf{B}_{kl}$.
- To compute the $n \times n$ matrix multiplication $\mathbf{C} = \mathbf{AB}$, perform
  - $\frac{n}{2} \times \frac{n}{2}$ matrix multiplications for 7 times,
  - $\frac{n}{2} \times \frac{n}{2}$ matrix additions for 18 times.
- Recursion relation: $T(n) = 7 \cdot T\left(\frac{n}{2}\right) + 18c\left(\frac{n}{2}\right)^2$.

The Master Theorem

Recurrence relation: $T(n) = a \cdot T(n/b) + c \cdot n^d$.

- The master theorem:

$$T(n) = \begin{cases} O\left(n^d\right), & \text{if } a < b^d; \\ O\left(n^d \log n\right), & \text{if } a = b^d; \\ O\left(n^{\log_b a}\right), & \text{if } a > b^d. \end{cases}$$

- For Strassen Algorithm, $a = 7$, $b = 2$, and $d = 2$.

The Master Theorem

Recurrence relation: $T(n) = a \cdot T(n/b) + c \cdot n^d$.

- The master theorem:

$$T(n) = \begin{cases} O(n^d), & \text{if } a < b^d; \\ O(n^d \log n), & \text{if } a = b^d; \\ O(n^{\log_b a}), & \text{if } a > b^d. \end{cases}$$

- For Strassen Algorithm, $a = 7$, $b = 2$, and $d = 2$.
- Thus, $T(n) = O\left(n^{\log_2 7}\right) \approx O(n^{2.807})$.

Summary

Block Matrix Multiplication

- Naively multiplying two $n \times n$ matrices costs $O(n^3)$ time.
- Block matrix multiplication:
  - Partition $n \times n$ matrices into four $\frac{n}{2} \times \frac{n}{2}$ sub-matrices.
  - Perform 8 multiplications and 4 additions.
- Divide-and-conquer does not help!
- The time complexity is still $O\left(n^{\log_2 8}\right) = O(n^3)$.

Strassen Algorithm

- Strassen algorithm also performs block matrix multiplication.
- It reduces the number of multiplications from 8 to 7.
- The time complexity is reduced to

$$O\left(n^{\log_2 7}\right) \approx O(n^{2.807}).$$

- There are better algorithms than Strassen algorithm.

Fast Matrix Multiplication

$\Omega(n^3)$ Naïve Algorithm

$\Omega(n^{2.807})$ Strassen [1969]

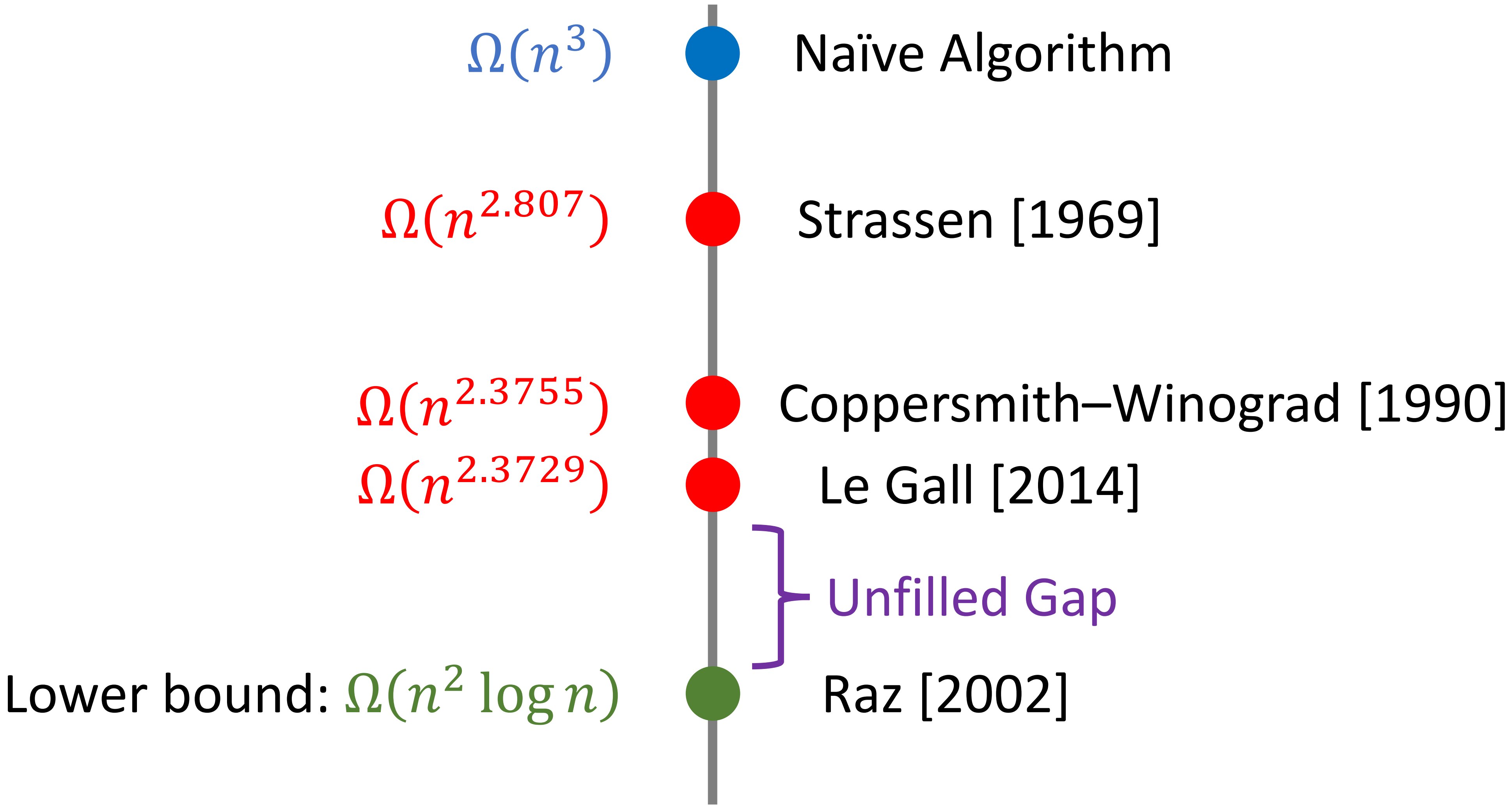
$\Omega(n^{2.3755})$ Coppersmith–Winograd [1990]

$\Omega(n^{2.3729})$ Le Gall [2014]

Fast Matrix Multiplication

$\Omega(n^3)$	Naïve Algorithm
$\Omega(n^{2.807})$	Strassen [1969]
$\Omega(n^{2.3755})$	Coppersmith–Winograd [1990]
$\Omega(n^{2.3729})$	Le Gall [2014]
	Any Worse Lower Bound?
Lower bound: $\Omega(n^2 \log n)$	Raz [2002]

Fast Matrix Multiplication



Thank You!