# Adaptive Matrix Multiplication

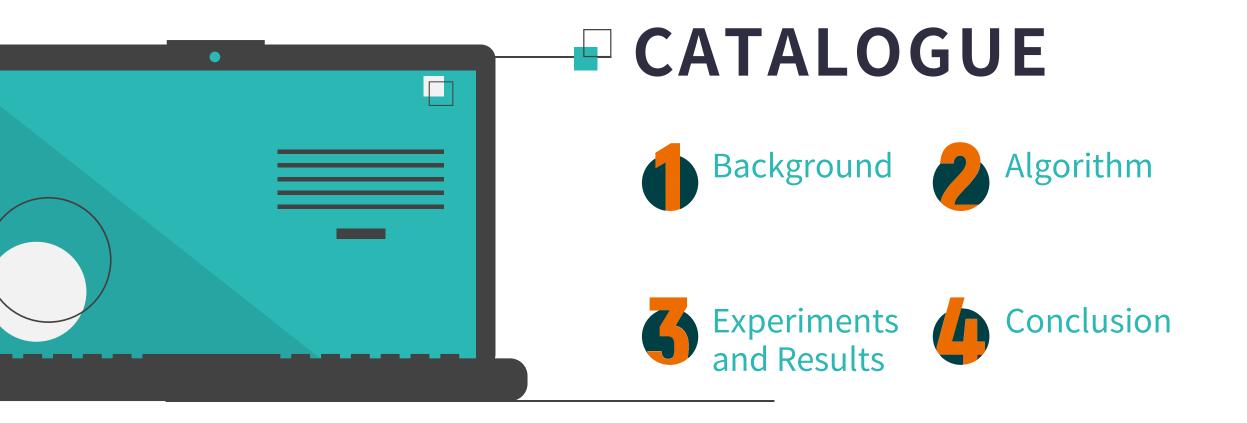
Data Structure and Algorithm Analysis

Group 21 WEI Jinqi LUO Peiyu HAN Xudong LI Xingze





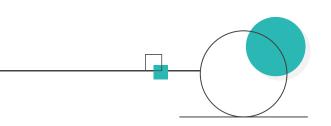








## Background





## Background



#### **Topic**

In this project we will implement and analyze, empirically and theoretically, a method for multiplying square matrices. The method will adapt to use algorithmic techniques depending on the system architecture on which it is run.



#### Goals



#### **Matrix Method**

Basic Variables Random Matrix Basic Method



#### **Multiply Algorithm**

Pseudocode Brute-force Algorithm Strassen Algorithm

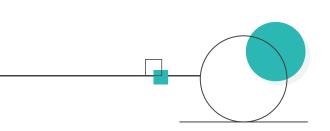


#### **Comparing and Optimizing**

Runtime Comparing Algorithm Optimizing Adaptive Algorithm











### Square Matrix Multiply Algorithm (Runtime: Θ(n³))

SQUARE-MATRIX-MULTIPLY(A, B)

```
1 n = A.rows

2 let C be a new n \times n matrix

3 for i = 1 to n

4 for j = 1 to n

5 c_{i,j} = 0

6 for k = 1 to n

7 c_{i,j} = c_{i,j} + a_{i,k} \cdot b_{k,j}

8 return C
```



#### Strassen Algorithm (Runtime: Θ(nlg7))

#### STRASSEN(A, B)

1 
$$n = A.rows$$

2 **if** 
$$n == 1$$

3 **return** 
$$A[1, 1] * B[1, 1]$$

4 let C be a new 
$$n \times n$$
 matrix

5 
$$A[1, 1] = A[1..n / 2][1..n / 2]$$

6 
$$A[1, 2] = A[1..n / 2][n / 2 + 1..n]$$

7 
$$A[2, 1] = A[n/2 + 1..n][1..n/2]$$

8 
$$A[2, 2] = A[n/2 + 1..n][n/2 + 1..n]$$

9 
$$B[1, 1] = B[1..n / 2][1..n / 2]$$

10 
$$B[1, 2] = B[1..n / 2][n / 2 + 1..n]$$

11 
$$B[2, 1] = B[n/2 + 1..n][1..n/2]$$

12 
$$B[2, 2] = B[n / 2 + 1..n][n / 2 + 1..n]$$

13 
$$S[1] = B[1, 2] - B[2, 2]$$

14 
$$S[2] = A[1, 1] + A[1, 2]$$

15 
$$S[3] = A[2, 1] + A[2, 2]$$

16 
$$S[4] = B[2, 1] - B[1, 1]$$

17 
$$S[5] = A[1, 1] + A[2, 2]$$

18 
$$S[6] = B[1, 1] + B[2, 2]$$

19 
$$S[7] = A[1, 2] - A[2, 2]$$

20 
$$S[8] = B[2, 1] + B[2, 2]$$

21 
$$S[9] = A[1, 1] - A[2, 1]$$

22 
$$S[10] = B[1, 1] + B[1, 2]$$

23 
$$P[1] = STRASSEN(A[1, 1], S[1])$$

24 
$$P[2] = STRASSEN(S[2], B[2, 2])$$

25 
$$P[3] = STRASSEN(S[3], B[1, 1])$$

26 
$$P[4] = STRASSEN(A[2, 2], S[4])$$

27 
$$P[5] = STRASSEN(S[5], S[6])$$

28 
$$P[6] = STRASSEN(S[7], S[8])$$

29 
$$P[7] = STRASSEN(S[9], S[10])$$

30 
$$C[1..n/2][1..n/2] = P[5] + P[4] - P[2] + P[6]$$

31 
$$C[1..n/2][n/2 + 1..n] = P[1] + P[2]$$

32 
$$C[n/2 + 1..n][1..n/2] = P[3] + P[4]$$

33 
$$C[n/2 + 1..n][n/2 + 1..n] = P[5] + P[1] - P[3] - P[7]$$

34 return C



#### Strassen Algorithm For Any Matrix (Runtime: $\Theta(n^{\lg 7})$ )

#### STRASSEN(A, B)

- n = A.rows
- **if** n == 1
- **return** A[1, 1] \* B[1, 1]
- **if** n % 2 == 1
- n = n 1
- 6 let C be a new  $n \times n$  matrix
- A[1, 1] = A[1..n / 2][1..n / 2]
- A[1, 2] = A[1..n / 2][n / 2 + 1..n]
- A[2, 1] = A[n/2 + 1..n][1..n/2]
- A[2, 2] = A[n/2 + 1..n][n/2 + 1..n]
- B[1, 1] = B[1..n / 2][1..n / 2]
- B[1, 2] = B[1..n / 2][n / 2 + 1..n]

- B[2, 1] = B[n/2 + 1..n][1..n/2]
- B[2, 2] = B[n/2 + 1..n][n/2 + 1..n]
- S[1] = B[1, 2] B[2, 2]
- S[2] = A[1, 1] + A[1, 2]
- S[3] = A[2, 1] + A[2, 2]
- S[4] = B[2, 1] B[1, 1]
- S[5] = A[1, 1] + A[2, 2]
- S[6] = B[1, 1] + B[2, 2]
- S[7] = A[1, 2] A[2, 2]
- S[8] = B[2, 1] + B[2, 2]
- S[9] = A[1, 1] A[2, 1]
- S[10] = B[1, 1] + B[1, 2]

- P[1] = STRASSEN(A[1, 1], S[1])
- P[2] = STRASSEN(S[2], B[2, 2])
- P[3] = STRASSEN(S[3], B[1, 1])
- P[4] = STRASSEN(A[2, 2], S[4])
- P[5] = STRASSEN(S[5], S[6])
- P[6] = STRASSEN(S[7], S[8])
- P[7] = STRASSEN(S[9], S[10])
- C[1..n/2][1..n/2] = P[5] + P[4] P[2] + P[6]
- C[1..n/2][n/2 + 1..n] = P[1] + P[2]
- C[n/2 + 1..n][1..n/2] = P[3] + P[4]
- C[n/2 + 1..n][n/2 + 1..n] = P[5] + P[1] P[3] P[7]
- 36 return C

B[2, 2] = B[n/2 + 1..n][n/2 + 1..n]



### Parallel Strassen Algorithm (Runtime: Θ(nlg<sup>7</sup>/lg<sup>2</sup>n))

26

#### P-STRASSEN(A, B)

13

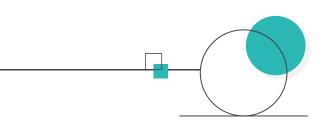
=A.rows	14	parallel	27	P[2] = STRASSEN(S[2], B[2, 2])
n == 1	15	S[1] = B[1, 2] - B[2, 2]	28	P[3] = STRASSEN(S[3], B[1, 1])
<b>return</b> $A[1, 1] * B[1, 1]$	16	S[2] = A[1, 1] + A[1, 2]	29	P[4] = STRASSEN(A[2, 2], S[4])
$C$ be a new $n \times n$ matrix	17	S[3] = A[2, 1] + A[2, 2]	30	P[5] = STRASSEN(S[5], S[6])
rallel	18	S[4] = B[2, 1] - B[1, 1]	31	P[6] = STRASSEN(S[7], S[8])
A[1, 1] = A[1n / 2][1n / 2]	19	S[5] = A[1, 1] + A[2, 2]	32	P[7] = STRASSEN(S[9], S[10])
A[1, 2] = A[1n / 2][n / 2 + 1n]	20	S[6] = B[1, 1] + B[2, 2]	33	parallel
A[2, 1] = A[n / 2 + 1n][1n / 2]	21	S[7] = A[1, 2] - A[2, 2]	34	C[1n/2][1n/2] = P[5] + P[4] - P[2] + P[6]
A[2, 2] = A[n / 2 + 1n][n / 2 + 1n]	22	S[8] = B[2, 1] + B[2, 2]	35	C[1n / 2][n / 2 + 1n] = P[1] + P[2]
B[1, 1] = B[1n / 2][1n / 2]	23	S[9] = A[1, 1] - A[2, 1]	36	C[n/2 + 1n][1n/2] = P[3] + P[4]
B[1, 2] = B[1n / 2][n / 2 + 1n]	24	S[10] = B[1, 1] + B[1, 2]	37	C[n/2 + 1n][n/2 + 1n] = P[5] + P[1] - P[3] - P[7]
B[2, 1] = B[n / 2 + 1n][1n / 2]	25	parallel	38	return C
	return $A[1, 1] * B[1, 1]$ C be a new $n \times n$ matrix rallel $A[1, 1] = A[1n / 2][1n / 2]$ $A[1, 2] = A[1n / 2][n / 2 + 1n]$ $A[2, 1] = A[n / 2 + 1n][1n / 2]$ $A[2, 2] = A[n / 2 + 1n][n / 2 + 1n]$ $B[1, 1] = B[1n / 2][1n / 2]$ $B[1, 2] = B[1n / 2][n / 2 + 1n]$	return $A[1, 1] * B[1, 1]$ 16  C be a new $n \times n$ matrix 17  rallel 18 $A[1, 1] = A[1n / 2][1n / 2]$ 19 $A[1, 2] = A[1n / 2][n / 2 + 1n]$ 20 $A[2, 1] = A[n / 2 + 1n][1n / 2]$ 21 $A[2, 2] = A[n / 2 + 1n][n / 2 + 1n]$ 22 $B[1, 1] = B[1n / 2][1n / 2]$ 23 $B[1, 2] = B[1n / 2][n / 2 + 1n]$ 24	return $A[1, 1] * B[1, 1]$ 16 $S[2] = A[1, 1] + A[1, 2]$ C be a new $n \times n$ matrix 17 $S[3] = A[2, 1] + A[2, 2]$ rallel 18 $S[4] = B[2, 1] - B[1, 1]$ A[1, 1] = A[1n/2][1n/2] 19 $S[5] = A[1, 1] + A[2, 2]A[1, 2] = A[1n/2][n/2 + 1n]$ 20 $S[6] = B[1, 1] + B[2, 2]A[2, 1] = A[n/2 + 1n][1n/2]$ 21 $S[7] = A[1, 2] - A[2, 2]A[2, 2] = A[n/2 + 1n][n/2 + 1n]$ 22 $S[8] = B[2, 1] + B[2, 2]B[1, 1] = B[1n/2][1n/2]$ 23 $S[9] = A[1, 1] - A[2, 1]B[1, 2] = B[1n/2][n/2 + 1n]$ 24 $S[10] = B[1, 1] + B[1, 2]$	return $A[1, 1] * B[1, 1]$ 16 $S[2] = A[1, 1] + A[1, 2]$ 29 $C$ be a new $n \times n$ matrix 17 $S[3] = A[2, 1] + A[2, 2]$ 30 $C$ rallel 18 $C[4] = B[2, 1] - B[1, 1]$ 31 $C[4] = A[1n / 2][1n / 2]$ 19 $C[5] = A[1, 1] + A[2, 2]$ 32 $C[6] = A[1, 1] + A[2, 2]$ 33 $C[6] = A[1, 1] + A[2, 2]$ 34 $C[6] = A[1, 1] + A[2, 2]$ 35 $C[6] = A[1, 1] + A[2, 2]$ 36 $C[6] = A[1, 1] + A[2, 1]$ 36 $C[6] = A[1, 1] + A[2, 1]$ 36 $C[6] = A[1, 1] + A[2, 1]$ 37 $C[6] = A[1, 1] + A[2, 1]$ 37 $C[6] = A[1, 1] + A[2, 1]$ 37

P[1] = STRASSEN(A[1, 1], S[1])





# **Experiments**and Results





## Timeline





	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	
Week 11	Team Formed	Topic and	d Requirements	Paper Reading and Reference Collection				
Week 12	Basic Matrix Method		Brute-force Algorithm	Weekly	Strassen Algorithm			
Week 13	Strassen O	Strassen Optimizing		Meeting for	Parallel Algorithm			
Week 14	Algorithm Comparing			Progress Reporting	Presentation Preparing			
Week 15	Final Presentation	Paner Prenaring						

### Contribution



**Brute-force and Strassen Algorithm** WEI **25%** Pseudocode and Parallel Algorithm **HAN 25%** Parallel Optimizing and Debugging LI **25% Basic Matrix Method and Paper** LUO 25%



## **Asymptotic Bounds**



#### **Square Matrix Multiply Algorithm**

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$
$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

where

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Then

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)$$

## **Asymptotic Bounds**



#### **Strassen Algorithm**

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$S_1 = B_{12} - B_{22}, \quad S_2 = A_{11} + A_{12}, \quad S_3 = A_{21} + A_{22}, \quad S_4 = B_{21} - B_{11}, \quad S_5 = A_{11} + A_{22},$$

$$S_6 = B_{11} + B_{22}, \quad S_7 = A_{12} - A_{22}, \quad S_8 = B_{21} + B_{22}, \quad S_9 = A_{11} - A_{21}, \quad S_{10} = B_{11} + B_{12}$$

$$P_1 = A_{11} \cdot S_1, \quad P_2 = S_2 + B_{22}, \quad P_3 = S_3 \cdot B_{11}, \quad P_4 = A_{22} \cdot S_4, \quad P_5 = S_5 \cdot S_6, \quad P_6 = S_7 \cdot S_8, \quad P_7 = S_9 \cdot S_{10}$$

$$C_{11} = P_5 + P_4 - P_2 + P_6, \quad C_{12} = P_1 + P_2$$

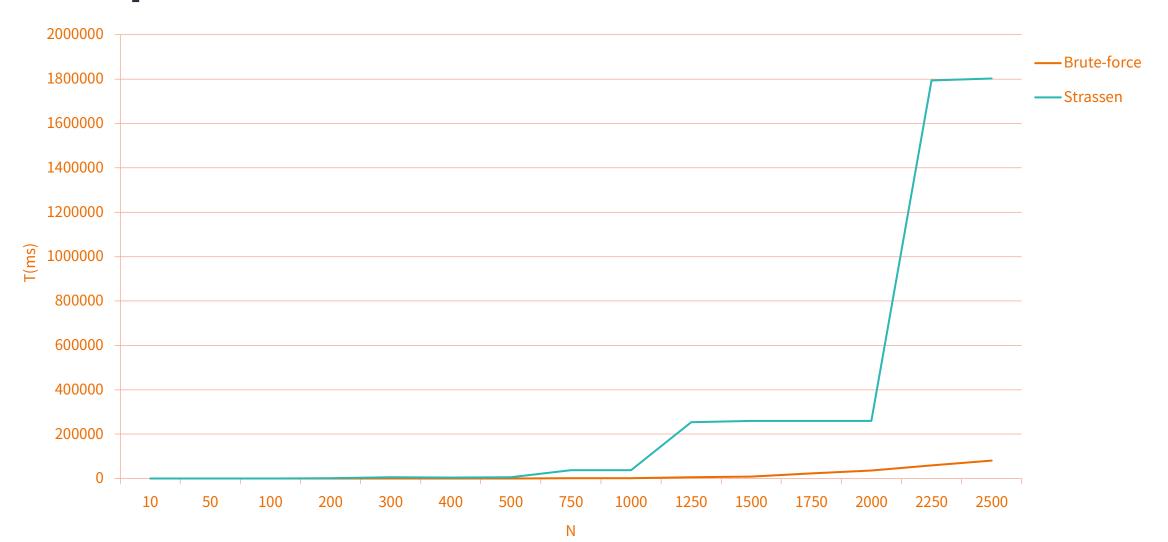
$$C_{21} = P_3 + P_4, \quad C_{22} = P_5 + P_1 - P_3 - P_7$$

Then

$$T(n) = \begin{cases} \Theta(1), & n = 1\\ 7T\left(\frac{n}{2}\right) + \Theta(n^2), & n > 1 \end{cases}$$

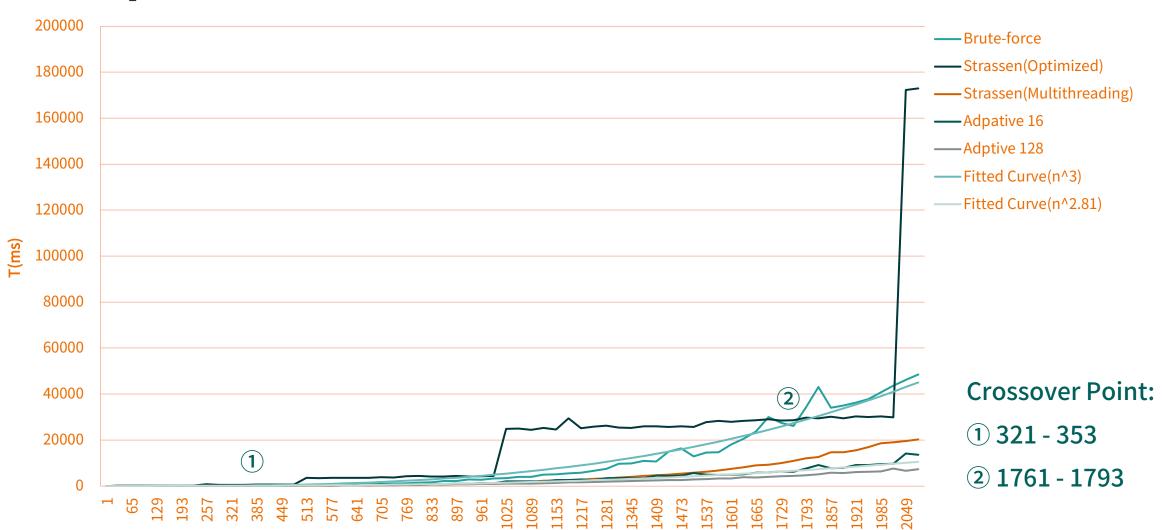
## **Compare Runtime**





## **Compare Runtime**

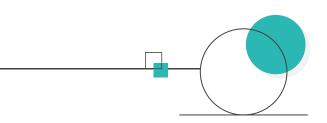








## Conclusion





## CONCLUSION

In this project, we first implemented the brute-force algorithm of matrix multiplication and Strassen algorithm. After that, the existing algorithms are optimized from two directions of multithreading and code structure, and more ideal results are obtained. After comparison, when the matrix size is small, the brute-force algorithm is faster. But when the matrix size is large, the Strassen algorithm has greater advantages. In addition, the parallel algorithm can improve the utilization of resources and speed up the computation significantly. Finally, we propose a method that can select different algorithms according to the matrix itself, so as to improve the operational efficiency of matrix multiplication.















# Thanks FOR LISTENING

December 21, 2020