# Matrix multiplication Analysis

By: Shubham Arora

# Problem Formulation

- Matrices are used everywhere.
- Efficient matrix operations have been studied for quite some time
- I compare ways of matrix multiplications, and validate the theoretical results

# Algorithms Compared

- Naive
- Strassens' algorithm

Note: For simplicity, I consider only dense square matrices

# Naive Matrix

For two matrices *a* and *b* that can be multiplied

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj} .$$

# Naive Matrix multiplication

For two matrices of size n x n

Time complexity:

$$O(n^3)$$

# Strassens's Method of Matrix Multiplication

## Key insight

- Multiplication is computationally more expensive than addition
- Utilise Divide and Conquer

# Preparation for Divide and Conquer

Let us break down the problem into subparts as:

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

# Naive Algorithm

Now, the standard algorithm does the follows:

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

i.e. 8 multiplications, 4 Additions

# Strassens's Algorithm

Strassen's algorithm does a neat trick:

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

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

i.e. **7 multiplications**, 18 Additions

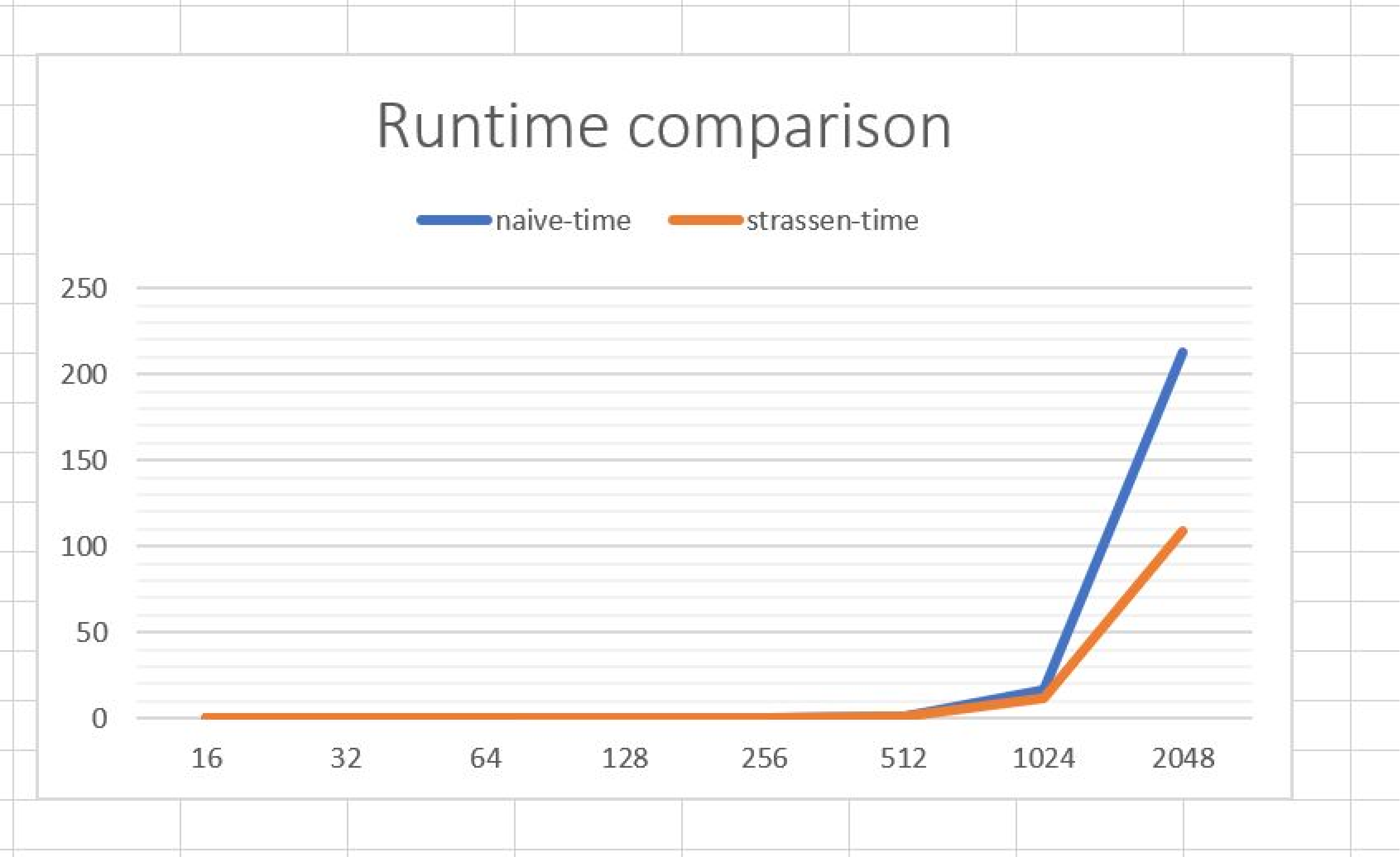
# Strassen Algorithm

For two matrices of size n x n

Time complexity:

$$O(n^{log(7)}) \approx O(n^{2.81})$$

# Results



Runtime comparison
naive-time
strassen-time
250
200
150
100
50
0
16
32
64
128
256
512
1024
2048

# Results

| size | naive-time | strassen-time | isEqual |
|---|---|---|---|
| 16 | 0 | 0 | TRUE |
| 32 | 0 | 0 | TRUE |
| 64 | 0 | 0 | TRUE |
| 128 | 0.03125 | 0.03125 | TRUE |
| 256 | 0.1875 | 0.25 | TRUE |
| 512 | 1.65625 | 1.671875 | TRUE |
| 1024 | 16.53125 | 11.625 | TRUE |
| 2048 | 213.098587 | 109.159874 | TRUE |

# Some notes

- Current fastest algorithms has performance : "Modified" Coppersmith–Winograd algorithm

  $$O(n^{2.373})$$

- However, algorithms better than strassens have very large overhead, and not used in practice
- The lower bound on Matrix Multiplication is proved to be:

  $$O(n^2)$$

# Future work

- Utilise parallelization
- Utilise distributed systems
- Use GPU's

# References


- Wikipedia -
- Introduction to Algorithms - CLRS - 3rd Edition