Fast Matrix Multiplication

Michael Schmid

31.05.2021

Big $\mathcal{O}$ notation

- Time complexity of an algorithm

Big $\mathcal{O}$ notation

- Time complexity of an algorithm
- How many multiplications in a function

Big $\mathcal{O}$ notation

- Time complexity of an algorithm
- How many multiplications in a function
- Drop Constants

Big $\mathcal{O}$ notation

Algorithm 1 Foo 1

```
function FOO(a, b)
    return a + b
```

Big $\mathcal{O}$ notation

Algorithm 2 Foo 1

function $\textsc{Foo}(a, b)$
    return $a + b$

$\mathcal{O}(1)$

Big $\mathcal{O}$ notation

Algorithm 3 Foo 2

```
function FOO(a, b)
    x ← a + b
    y ← a · b
    return x + y
```

Big $\mathcal{O}$ notation

Algorithm 4 Foo 2

```
function FOO(a, b)
    x ← a + b
    y ← a · b
    return x + y
```

$$\mathcal{O}(1) + \mathcal{O}(1) = 2\mathcal{O}(1) = \mathcal{O}(1)$$

Big $\mathcal{O}$ notation

Algorithm 5 Foo 3

function FOO($\mathbf{A}, \mathbf{B}$,n)
    $sum \leftarrow 0$
    for $i = 0, 1, 2 \ldots, n$ **do**
        $sum \leftarrow sum + A[i] \cdot B[i]$
    return sum

Big $\mathcal{O}$ notation

Algorithm 6 Foo 3

```
function FOO(A, B, n)
    sum ← 0
    for i = 0, 1, 2 . . . , n do
        sum ← sum + A[i] · B[i]
    return sum
```

$\mathcal{O}(n)$

Big $\mathcal{O}$ notation

Algorithm 7 Foo 4

function FOO($\mathbf{A}$, $\mathbf{B}$, n)
    $sum \leftarrow 0$
    for $i = 0, 1, 2 \ldots, n$ **do**
        for $j = 0, 1, 2 \ldots, n$ **do**
            $sum \leftarrow sum + A[i] \cdot B[j]$
    return sum

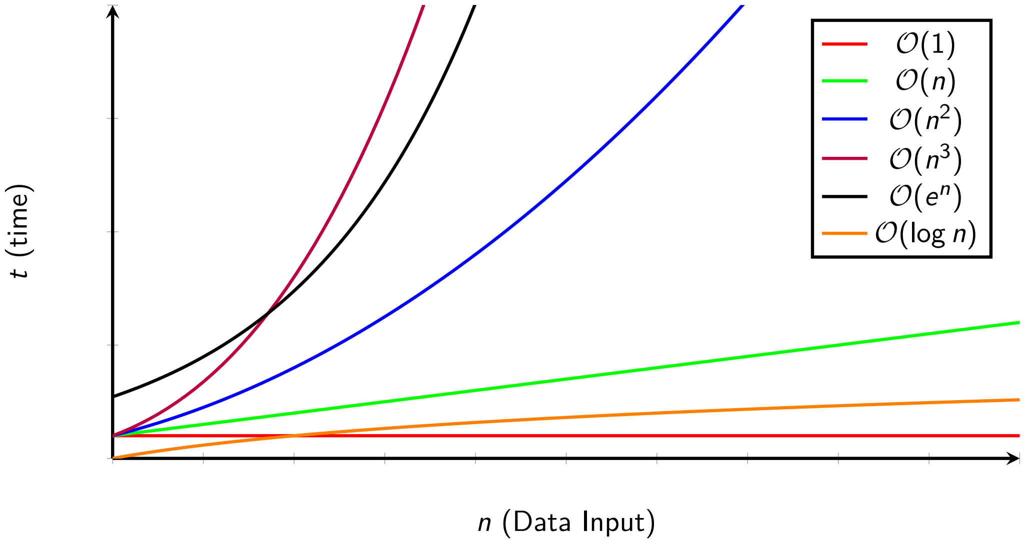
Big $\mathcal{O}$ notation

Algorithm 8 Foo 4

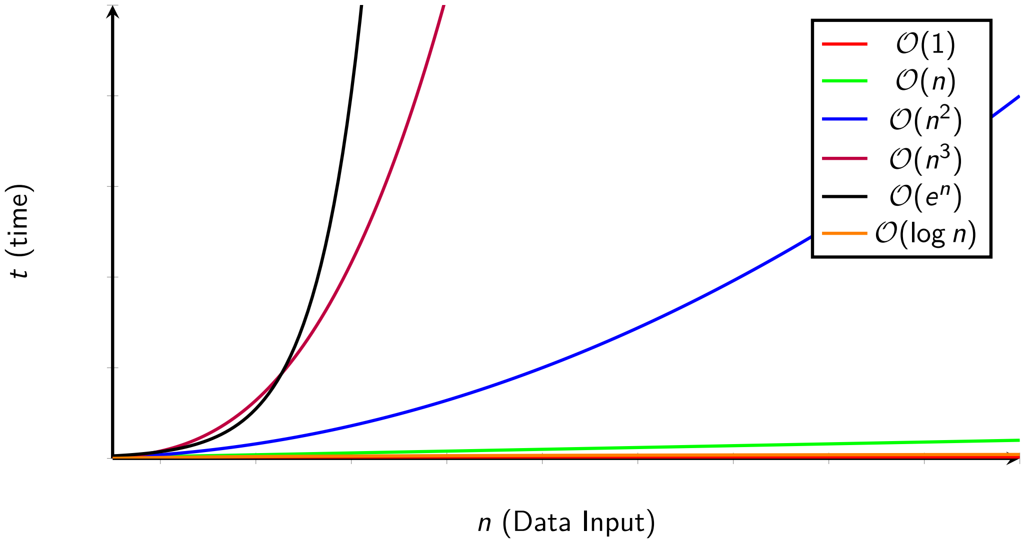
```
function FOO(A, B, n)
    sum ← 0
    for i = 0, 1, 2 ..., n do
        for j = 0, 1, 2 ..., n do
            sum ← sum + A[i] · B[j]
    return sum
```

$\mathcal{O}(n^2)$

Big $\mathcal{O}$ notation



Big $\mathcal{O}$ notation



Strassen's Algorithm

Numer. Math. 13, 354–356 (1969)

Gaussian Elimination is not Optimal

Volker Strassen*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{\log 7}$ arithmetical operations.

This fact should be compared with the result of Klyuyev and Kokovkin-Shcherbak [1] that Gaussian elimination for solving a system of linear equations is optimal if one restricts oneself to operations upon rows and columns as a whole. We also note that Winograd [2] modifies the usual algorithms for matrix multiplication and inversion and for solving systems of linear equations, trading roughly half of the multiplications for additions and subtractions.

It is a pleasure to thank D. Brillinger for inspiring discussions about the present subject and St. Cook and B. Parlett for encouraging me to write this paper.

2. We define algorithms $\alpha_{m,k}$ which multiply matrices of order $m2^k$, by induction on k: $\alpha_{m,0}$ is the usual algorithm for matrix multiplication (requiring m^3 multiplications and $m^2(m-1)$ additions). $\alpha_{m,k}$ already being known, define $\alpha_{m,k+1}$ as follows:

If A, B are matrices of order $m2^{k+1}$ to be multiplied, write

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

where the A_{ik}, B_{ik}, C_{ik} are matrices of order $m2^k$. Then compute

$$\begin{aligned}
\mathrm{I} &= (A_{11}+A_{22})(B_{11}+B_{22}), \\
\mathrm{II} &= (A_{21}+A_{22})B_{11}, \\
\mathrm{III} &= A_{11}(B_{12}-B_{22}), \\
\mathrm{IV} &= A_{22}(-B_{11}+B_{21}), \\
\mathrm{V} &= (A_{11}+A_{12})B_{22}, \\
\mathrm{VI} &= (-A_{11}+A_{21})(B_{11}+B_{12}), \\
\mathrm{VII} &= (A_{12}-A_{22})(B_{21}+B_{22}),
\end{aligned}$$

* The results have been found while the author was at the Department of Statistics of the University of California, Berkeley. The author wishes to thank the National Science Foundation for their support (NSF GP-7454).

$$\begin{aligned}
C_{11} &= \mathrm{I}+\mathrm{IV}-\mathrm{V}+\mathrm{VII}, \\
C_{21} &= \mathrm{II}+\mathrm{IV}, \\
C_{12} &= \mathrm{III}+\mathrm{V}, \\
C_{22} &= \mathrm{I}+\mathrm{III}-\mathrm{II}+\mathrm{VI},
\end{aligned}$$

using $\alpha_{m,k}$ for multiplication and the usual algorithm for addition and subtraction of matrices of order $m2^k$.

By induction on k one easily sees

Fact 1. $\alpha_{m,k}$ computes the product of two matrices of order $m2^k$ with $m^3 7^k$ multiplications and $(5+m)m^2 7^k - 6(m2^k)^2$ additions and subtractions of numbers.

Thus one may multiply two matrices of order 2^k with 7^k numbermultiplications and less than $6 \cdot 7^k$ additions and subtractions.

Fact 2. The product of two matrices of order n may be computed with $< 4.7 \cdot n^{\log 7}$ arithmetical operations.

Proof. Put

$$k = [\log n - 4],$$

$$m = [n 2^{-k}] + 1,$$

then

$$n \leqq m 2^k.$$

Imbedding matrices of order n into matrices of order $m2^k$ reduces our task to that of estimating the number of operations of $\alpha_{m,k}$. By Fact 1 this number is

$$\begin{aligned}
&(5+2m)m^2 7^k - 6(m2^k)^2 \\
&< \left(5 + 2(n2^{-k}+1)\right)(n2^{-k}+1)^2 7^k \\
&< 2n^3(7/8)^k + 12.03\, n^2 (7/4)^k
\end{aligned}$$

(here we have used $16 \cdot 2^k \leqq n$)

$$\begin{aligned}
&= \left(2(8/7)^{\log n - k} + 12.03\,(4/7)^{\log n - k}\right) n^{\log 7} \\
&\leqq \max_{4 \leqq t \leqq 5} \left(2(8/7)^t + 12.03\,(4/7)^t\right) n^{\log 7} \\
&\leqq 4.7 \cdot n^{\log 7}
\end{aligned}$$

by a convexity argument.

We now turn to matrix inversion. To apply the algorithms below it is necessary to assume not only that the matrix is invertible but that all occurring divisions make sense (a similar assumption is of course necessary for Gaussian elimination).

We define algorithms $\beta_{m,k}$ which invert matrices of order $m2^k$, by induction on k: $\beta_{m,0}$ is the usual Gaussian elimination algorithm. $\beta_{m,k}$ already being known, define $\beta_{m,k+1}$ as follows:

If A is a matrix of order $m2^{k+1}$ to be inverted, write

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

where the A_{ik}, C_{ik} are matrices of order $m2^k$. Then compute

$$\begin{aligned}
\mathrm{I} &= A_{11}^{-1}, \\
\mathrm{II} &= A_{21}\,\mathrm{I}, \\
\mathrm{III} &= \mathrm{I}\,A_{12}, \\
\mathrm{IV} &= A_{21}\,\mathrm{III}, \\
\mathrm{V} &= \mathrm{IV} - A_{22}, \\
\mathrm{VI} &= \mathrm{V}^{-1}, \\
C_{12} &= \mathrm{III} \cdot \mathrm{VI}, \\
C_{21} &= \mathrm{VI} \cdot \mathrm{II}, \\
\mathrm{VII} &= \mathrm{III} \cdot C_{21}, \\
C_{11} &= \mathrm{I} - \mathrm{VII}, \\
C_{22} &= -\mathrm{VI}
\end{aligned}$$

using $\alpha_{m,k}$ for multiplication, $\beta_{m,k}$ for inversion and the usual algorithm for addition or subtraction of two matrices of order $m2^k$.

By induction on k one easily sees

Fact 3. $\beta_{m,k}$ computes the inverse of a matrix of order $m2^k$ with $m2^k$ divisions, $\leqq \frac{6}{5} m^3 7^k - m2^k$ multiplications and $\leqq \frac{6}{5}(5+m)m^2 7^k - 7(m2^k)^2$ additions and subtractions of numbers. The next Fact follows in the same way as Fact 2.

Fact 4. The inverse of a matrix of order n may be computed with $< 5.64 \cdot n^{\log 7}$ arithmetical operations.

Similar results hold for solving a system of linear equations or computing a determinant (use $\operatorname{Det} A = (\operatorname{Det} A_{11}) \operatorname{Det}(A_{22} - A_{21} A_{11}^{-1} A_{12})$).

References

1. Klyuyev, V. V., and N. I. Kokovkin-Shcherbak: On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations. Translation by G. I. Tee: Technical Report CS 24, June 14, 1965, Computer Science Dept., Stanford University.
2. Winograd, S.: A new algorithm for inner product. IBM Research Report RC-1943, Nov. 21, 1967.

Prof. Volker Strassen
Seminar für angewandte Mathematik
der Universität
8032 Zürich, Freie Str. 36

Strassen's Algorithm

$$\mathbf{AB} = \mathbf{C}$$

Strassen's Algorithm

$$\mathbf{AB} = \mathbf{C}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Strassen's Algorithm

$$\mathbf{AB} = \mathbf{C}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Algorithm

Algorithm 9 Square Matrix Multiplication

```
function MM(A, B, C)
    sum ← 0
    n ← columns(A) == rows(B)
    m ← rows(A)
    p ← columns(B)
    for i = 0, 1, 2 ..., m − 1 do
        for j = 0, 1, 2 ..., p − 1 do
            sum ← 0
            for k = 0, 1, 2 ..., n − 1 do
                sum ← sum + A[i][k] · B[k][j]
            C[i][j] ← sum
    return C
```

$$
\begin{bmatrix}
B_{1,1} & \cdots & B_{1,j} & \cdots & B_{1,p} \\
\vdots & & \vdots & & \vdots \\
B_{k,1} & \cdots & B_{k,j} & \cdots & B_{k,p} \\
\vdots & & \vdots & & \vdots \\
B_{n,1} & \cdots & B_{n,j} & \cdots & B_{n,p}
\end{bmatrix}
$$

$$
\begin{bmatrix}
A_{1,1} & \cdots & A_{1,k} & \cdots & A_{1,n} \\
\vdots & & \vdots & & \vdots \\
A_{i,1} & \cdots & A_{i,k} & \cdots & A_{i,n} \\
\vdots & & \vdots & & \vdots \\
A_{m,1} & \cdots & A_{m,k} & \cdots & A_{m,n}
\end{bmatrix}
\begin{bmatrix}
C_{1,1} & \cdots & C_{1,j} & \cdots & C_{1,p} \\
\vdots & & \vdots & & \vdots \\
C_{i,1} & \cdots & C_{i,j} & \cdots & C_{i,p} \\
\vdots & & \vdots & & \vdots \\
C_{m,1} & \cdots & C_{m,j} & \cdots & C_{m,p}
\end{bmatrix}
$$

Algorithm

Algorithm 10 Square Matrix Multiplication

```
function MM(A, B, C)
    sum ← 0
    n ← columns(A) == rows(B)
    m ← rows(A)
    p ← columns(B)
    for i = 0, 1, 2 . . . , m − 1 do
        for j = 0, 1, 2 . . . , p − 1 do
            sum ← 0
            for k = 0, 1, 2 . . . , n − 1 do
                sum ← sum + A[i][k] · B[k][j]
            C[i][j] ← sum
    return C
```

$$\mathcal{O}(n^3)$$

Strassen's Algorithm

$$
\begin{aligned}
\mathrm{I} &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
\mathrm{II} &= (A_{21} + A_{22}) \cdot B_{11} \\
\mathrm{III} &= A_{11} \cdot (B_{12} - B_{22}) \\
\mathrm{IV} &= A_{22} \cdot (-B_{11} + B_{21}) \\
\mathrm{V} &= (A_{11} + A_{12}) \cdot B_{22} \\
\mathrm{VI} &= (-A_{11} + A_{21}) \cdot (B_{11} + B_{12}) \\
\mathrm{VII} &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

Strassen's Algorithm

$$
\begin{aligned}
\mathrm{I} &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
\mathrm{II} &= (A_{21} + A_{22}) \cdot B_{11} \\
\mathrm{III} &= A_{11} \cdot (B_{12} - B_{22}) \\
\mathrm{IV} &= A_{22} \cdot (-B_{11} + B_{21}) \\
\mathrm{V} &= (A_{11} + A_{12}) \cdot B_{22} \\
\mathrm{VI} &= (-A_{11} + A_{21}) \cdot (B_{11} + B_{12}) \\
\mathrm{VII} &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

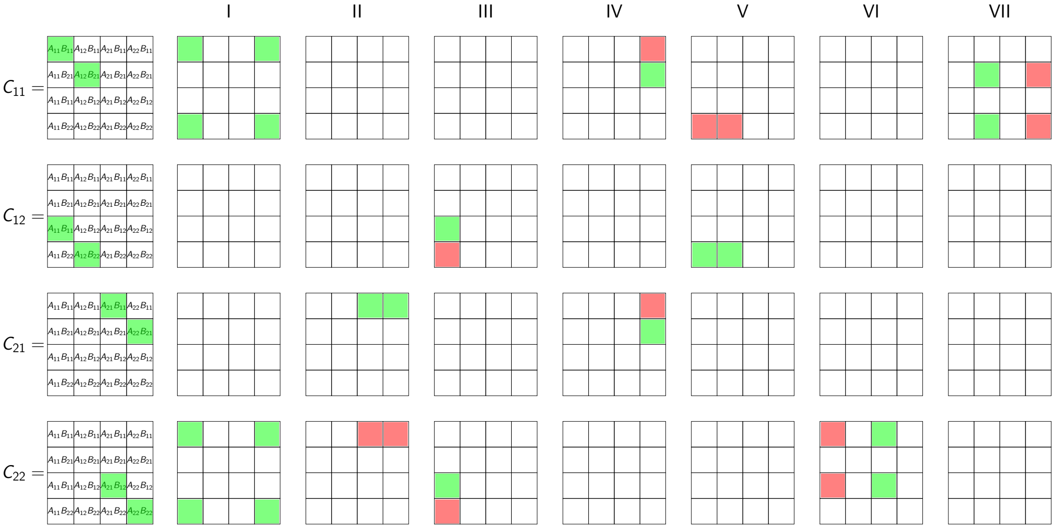
$$
\begin{aligned}
C_{11} &= \mathrm{I} + \mathrm{IV} - \mathrm{V} + \mathrm{VII} \\
C_{21} &= \mathrm{II} + \mathrm{IV} \\
C_{12} &= \mathrm{III} + \mathrm{V} \\
C_{22} &= \mathrm{I} + \mathrm{III} - \mathrm{II} + \mathrm{VI}
\end{aligned}
$$

Strassen's Algorithm

$$
\begin{aligned}
\mathrm{I} &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
\mathrm{II} &= (A_{21} + A_{22}) \cdot B_{11} \\
\mathrm{III} &= A_{11} \cdot (B_{12} - B_{22}) \\
\mathrm{IV} &= A_{22} \cdot (-B_{11} + B_{21}) \\
\mathrm{V} &= (A_{11} + A_{12}) \cdot B_{22} \\
\mathrm{VI} &= (-A_{11} + A_{21}) \cdot (B_{11} + B_{12}) \\
\mathrm{VII} &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

$$
\begin{aligned}
C_{11} &= \mathrm{I} + \mathrm{IV} - \mathrm{V} + \mathrm{VII} \\
C_{21} &= \mathrm{II} + \mathrm{IV} \\
C_{12} &= \mathrm{III} + \mathrm{V} \\
C_{22} &= \mathrm{I} + \mathrm{III} - \mathrm{II} + \mathrm{VI}
\end{aligned}
$$

$$
\begin{aligned}
C_{11} &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) + A_{22} \cdot (-B_{11} + B_{21}) - (A_{11} + A_{12}) \cdot B_{22} + (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \\
C_{11} &= A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{11} + A_{22}B_{22} - A_{22}B_{11} + A_{22}B_{21} - A_{11}B_{22} - A_{12}B_{22} + A_{12}B_{21} + A_{12}B_{22} - A_{22}B_{21} - A_{22}B_{22} \\
C_{11} &= A_{11}B_{11} + A_{12}B_{21}
\end{aligned}
$$


I
II
III
IV
V
VI
VII
$C_{11} =$
$A_{11}B_{11}$ $A_{12}B_{11}$ $A_{21}B_{11}$ $A_{22}B_{11}$
$A_{11}B_{21}$ $A_{12}B_{21}$ $A_{21}B_{21}$ $A_{22}B_{21}$
$A_{11}B_{11}$ $A_{12}B_{12}$ $A_{21}B_{12}$ $A_{22}B_{12}$
$A_{11}B_{22}$ $A_{12}B_{22}$ $A_{21}B_{22}$ $A_{22}B_{22}$
$C_{12} =$
$A_{11}B_{11}$ $A_{12}B_{11}$ $A_{21}B_{11}$ $A_{22}B_{11}$
$A_{11}B_{21}$ $A_{12}B_{21}$ $A_{21}B_{21}$ $A_{22}B_{21}$
$A_{11}B_{11}$ $A_{12}B_{12}$ $A_{21}B_{12}$ $A_{22}B_{12}$
$A_{11}B_{22}$ $A_{12}B_{22}$ $A_{21}B_{22}$ $A_{22}B_{22}$
$C_{21} =$
$A_{11}B_{11}$ $A_{12}B_{11}$ $A_{21}B_{11}$ $A_{22}B_{11}$
$A_{11}B_{21}$ $A_{12}B_{21}$ $A_{21}B_{21}$ $A_{22}B_{21}$
$A_{11}B_{11}$ $A_{12}B_{12}$ $A_{21}B_{12}$ $A_{22}B_{12}$
$A_{11}B_{22}$ $A_{12}B_{22}$ $A_{21}B_{22}$ $A_{22}B_{22}$
$C_{22} =$
$A_{11}B_{11}$ $A_{12}B_{11}$ $A_{21}B_{11}$ $A_{22}B_{11}$
$A_{11}B_{21}$ $A_{12}B_{21}$ $A_{21}B_{21}$ $A_{22}B_{21}$
$A_{11}B_{11}$ $A_{12}B_{12}$ $A_{21}B_{12}$ $A_{22}B_{12}$
$A_{11}B_{22}$ $A_{12}B_{22}$ $A_{21}B_{22}$ $A_{22}B_{22}$

Strassen's Algorithm

$$
\begin{aligned}
\mathrm{I} &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
\mathrm{II} &= (A_{21} + A_{22}) \cdot B_{11} \\
\mathrm{III} &= A_{11} \cdot (B_{12} - B_{22}) \\
\mathrm{IV} &= A_{22} \cdot (-B_{11} + B_{21}) \\
\mathrm{V} &= (A_{11} + A_{12}) \cdot B_{22} \\
\mathrm{VI} &= (-A_{11} + A_{21}) \cdot (B_{11} + B_{12}) \\
\mathrm{VII} &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

$$
\begin{aligned}
C_{11} &= \mathrm{I} + \mathrm{IV} - \mathrm{V} + \mathrm{VII} \\
C_{21} &= \mathrm{II} + \mathrm{IV} \\
C_{12} &= \mathrm{III} + \mathrm{V} \\
C_{22} &= \mathrm{I} + \mathrm{III} - \mathrm{II} + \mathrm{VI}
\end{aligned}
$$

Strassen's Algorithm

$$
\begin{aligned}
\mathbf{I} &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{II} &= (\mathbf{A}_{21} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{III} &= \mathbf{A}_{11} \cdot (\mathbf{B}_{12} - \mathbf{B}_{22}) \\
\mathbf{IV} &= \mathbf{A}_{22} \cdot (-\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{V} &= (\mathbf{A}_{11} + \mathbf{A}_{12}) \cdot \mathbf{B}_{22} \\
\mathbf{VI} &= (-\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{12}) \\
\mathbf{VII} &= (\mathbf{A}_{12} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{21} + \mathbf{B}_{22})
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{C}_{11} &= \mathbf{I} + \mathbf{IV} - \mathbf{V} + \mathbf{VII} \\
\mathbf{C}_{21} &= \mathbf{II} + \mathbf{IV} \\
\mathbf{C}_{12} &= \mathbf{III} + \mathbf{V} \\
\mathbf{C}_{22} &= \mathbf{I} + \mathbf{III} - \mathbf{II} + \mathbf{VI}
\end{aligned}
$$

Algorithm

Algorithm 11 Strassen Matrix Multiplication

function STRASSEN($\mathbf{A}, \mathbf{B}, n$)
  if $n = 2$ **then**
    $\mathbf{C} \leftarrow zeros((n, n))$
    $P \leftarrow (A[0][0] + A[1][1]) \cdot (B[0][0] + B[1][1])$
    $Q \leftarrow (A[1][0] + A[1][1]) \cdot B[0][0]$
    $R \leftarrow A[0][0] \cdot (B[0][1] - B[1][1])$
    $S \leftarrow A[1][1] \cdot (B[1][0] - B[0][0])$
    $T \leftarrow (A[0][0] + A[0][1]) \cdot B[1][1]$
    $U \leftarrow (A[1][0] - A[0][0]) \cdot (B[0][0] + B[0][1])$
    $V \leftarrow (A[0][1] - A[1][1]) \cdot (B[1][0] + B[1][1])$
    $C[0][0] \leftarrow P + S - T + V$
    $C[0][1] \leftarrow R + T$
    $C[1][0] \leftarrow Q + S$
    $C[1][1] \leftarrow P + R - Q + U$
  else
    $m \leftarrow n/2$
    $\mathbf{A11}, \mathbf{A12}, \mathbf{A21}, \mathbf{A22} \leftarrow \mathbf{A}[:m][:m], \mathbf{A}[:m][m:], \mathbf{A}[m:][:m], \mathbf{A}[m:][m:]$
    $\mathbf{B11}, \mathbf{B12}, \mathbf{B21}, \mathbf{B22} \leftarrow \mathbf{B}[:m][:m], \mathbf{B}[:m][m:], \mathbf{B}[m:][:m], \mathbf{B}[m:][m:]$
    $\mathbf{P} \leftarrow \text{strassen}((\mathbf{A11} + \mathbf{A22}), (\mathbf{B11} + \mathbf{B22}), m)$
    $\mathbf{Q} \leftarrow \text{strassen}((\mathbf{A21} + \mathbf{A22}), \mathbf{B11}, m)$
    $\mathbf{R} \leftarrow \text{strassen}(\mathbf{A11}, (\mathbf{B12} - \mathbf{B22}), m)$
    $\mathbf{S} \leftarrow \text{strassen}(\mathbf{A22}, (\mathbf{B21} - \mathbf{B11}), m)$
    $\mathbf{T} \leftarrow \text{strassen}((\mathbf{A11} + \mathbf{A12}), \mathbf{B22}, m)$
    $\mathbf{U} \leftarrow \text{strassen}((\mathbf{A21} - \mathbf{A11}), (\mathbf{B11} + \mathbf{B12}), m)$
    $\mathbf{V} \leftarrow \text{strassen}((\mathbf{A12} - \mathbf{A22}), (\mathbf{B21} + \mathbf{B22}), m)$
    $\mathbf{C11} \leftarrow \mathbf{P} + \mathbf{S} - \mathbf{T} + \mathbf{V}$
    $\mathbf{C12} \leftarrow \mathbf{R} + \mathbf{T}$
    $\mathbf{C21} \leftarrow \mathbf{Q} + \mathbf{S}$
    $\mathbf{C22} \leftarrow \mathbf{P} + \mathbf{R} - \mathbf{Q} + \mathbf{U}$
    $C \leftarrow vstack((hstack((C11, C12)), hstack((C21, C22))))$
  return C

Algorithm

Algorithm 12 Strassen Matrix Multiplication

```
function STRASSEN(A, B, n)
    if n = 2 then
        C ← zeros((n, n))
        P ← (A[0][0] + A[1][1]) · (B[0][0] + B[1][1])
        Q ← (A[1][0] + A[1][1]) · B[0][0]
        R ← A[0][0] · (B[0][1] − B[1][1])
        S ← A[1][1] · (B[1][0] − B[0][0])
        T ← (A[0][0] + A[0][1]) · B[1][1]
        U ← (A[1][0] − A[0][0]) · (B[0][0] + B[0][1])
        V ← (A[0][1] − A[1][1]) · (B[1][0] + B[1][1])
        C[0][0] ← P + S − T + V
        C[0][1] ← R + T
        C[1][0] ← Q + S
        C[1][1] ← P + R − Q + U
    else
        m ← n/2
        A11, A12, A21, A22 ← A[: m][: m], A[: m][m :], A[m :][: m], A[m :][m :]
        B11, B12, B21, B22 ← B[: m][: m], B[: m][m :], B[m :][: m], B[m :][m :]
        P ← strassen((A11 + A22), (B11 + B22), m)
        Q ← strassen((A21 + A22), B11, m)
        R ← strassen(A11, (B12 − B22), m)
        S ← strassen(A22, (B21 − B11), m)
        T ← strassen((A11 + A12), B22, m)
        U ← strassen((A21 − A11), (B11 + B12), m)
        V ← strassen((A12 − A22), (B21 + B22), m)
        C11 ← P + S − T + V
        C12 ← R + T
        C21 ← Q + S
        C22 ← P + R − Q + U
        C ← vstack((hstack((C11, C12)), hstack((C21, C22))))
    return C
```

Algorithm

Algorithm 13 Strassen Matrix Multiplication

```
function STRASSEN(A, B, n)
    if n = 2 then
        C ← zeros((n, n))
        P ← (A[0][0] + A[1][1]) · (B[0][0] + B[1][1])
        Q ← (A[1][0] + A[1][1]) · B[0][0]
        R ← A[0][0] · (B[0][1] − B[1][1])
        S ← A[1][1] · (B[1][0] − B[0][0])
        T ← (A[0][0] + A[0][1]) · B[1][1]
        U ← (A[1][0] − A[0][0]) · (B[0][0] + B[0][1])
        V ← (A[0][1] − A[1][1]) · (B[1][0] + B[1][1])
        C[0][0] ← P + S − T + V
        C[0][1] ← R + T
        C[1][0] ← Q + S
        C[1][1] ← P + R − Q + U
    else
        m ← n/2
        A11, A12, A21, A22 ← A[: m][: m], A[: m][m :], A[m :][: m], A[m :][m :]
        B11, B12, B21, B22 ← B[: m][: m], B[: m][m :], B[m :][: m], B[m :][m :]
        P ← strassen((A11 + A22), (B11 + B22), m)
        Q ← strassen((A21 + A22), B11, m)
        R ← strassen(A11, (B12 − B22), m)
        S ← strassen(A22, (B21 − B11), m)
        T ← strassen((A11 + A12), B22, m)
        U ← strassen((A21 − A11), (B11 + B12), m)
        V ← strassen((A12 − A22), (B21 + B22), m)
        C11 ← P + S − T + V
        C12 ← R + T
        C21 ← Q + S
        C22 ← P + R − Q + U
        C ← vstack((hstack((C11, C12)), hstack((C21, C22))))
    return C
```

$$\mathcal{T}(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ 7 \cdot \mathcal{T}\left(\frac{n}{2}\right) + n^2 & \text{if } n > 2 \end{cases} = \mathcal{O}(n^{2.81})$$

Algorithm

Algorithm 14 Strassen Matrix Multiplication

function $\mathrm{MM}(\mathbf{A}, \mathbf{B}, n)$
  if $n = 2$ **then**
    $\mathbf{C} \leftarrow zeros((n, n))$
    $C[0, 0] \leftarrow A[0][0] * B[0][0] + A[0][1] * B[1][0]$
    $C[0, 1] \leftarrow A[0][0] * B[0][1] + A[0][1] * B[1][1]$
    $C[1, 0] \leftarrow A[1][0] * B[0][0] + A[1][1] * B[1][0]$
    $C[1, 1] \leftarrow A[1][0] * B[0][1] + A[1][1] * B[1][1]$
  else
    $m \leftarrow n/2$
    $\mathbf{A11}, \mathbf{A12}, \mathbf{A21}, \mathbf{A22} \leftarrow \mathbf{A}[: m][: m], \mathbf{A}[: m][m :], \mathbf{A}[m :][: m], \mathbf{A}[m :][m :]$
    $\mathbf{B11}, \mathbf{B12}, \mathbf{B21}, \mathbf{B22} \leftarrow \mathbf{B}[: m][: m], \mathbf{B}[: m][m :], \mathbf{B}[m :][: m], \mathbf{B}[m :][m :]$
    $\mathbf{C11} \leftarrow \mathrm{MM}(\mathbf{A11}, \mathbf{B11}) + \mathrm{MM}(\mathbf{A12}, \mathbf{B21})$
    $\mathbf{C12} \leftarrow \mathrm{MM}(\mathbf{A11}, \mathbf{B12}) + \mathrm{MM}(\mathbf{A12}, \mathbf{B22})$
    $\mathbf{C21} \leftarrow \mathrm{MM}(\mathbf{A21}, \mathbf{B11}) + \mathrm{MM}(\mathbf{A22}, \mathbf{B21})$
    $\mathbf{C22} \leftarrow \mathrm{MM}(\mathbf{A21}, \mathbf{B12}) + \mathrm{MM}(\mathbf{A22}, \mathbf{B22})$
    $C \leftarrow vstack((hstack((C11, C12)), hstack((C21, C22))))$
  return C

Algorithm

Algorithm 15 Strassen Matrix Multiplication

```
function MM(A, B, n)
    if n = 2 then
        C ← zeros((n, n))
        C[0, 0] ← A[0][0] * B[0][0] + A[0][1] * B[1][0]
        C[0, 1] ← A[0][0] * B[0][1] + A[0][1] * B[1][1]
        C[1, 0] ← A[1][0] * B[0][0] + A[1][1] * B[1][0]
        C[1, 1] ← A[1][0] * B[0][1] + A[1][1] * B[1][1]
    else
        m ← n/2
        A11, A12, A21, A22 ← A[: m][: m], A[: m][m :], A[m :][: m], A[m :][m :]
        B11, B12, B21, B22 ← B[: m][: m], B[: m][m :], B[m :][: m], B[m :][m :]
        C11 ← MM(A11, B11) + MM(A12, B21)
        C12 ← MM(A11, B12) + MM(A12, B22)
        C21 ← MM(A21, B11) + MM(A22, B21)
        C22 ← MM(A21, B12) + MM(A22, B22)
        C ← vstack((hstack((C11, C12)), hstack((C21, C22))))
    return C
```

$$\mathcal{T}(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ 8 \cdot \mathcal{T}(\frac{n}{2}) + n^2 & \text{if } n > 2 \end{cases} = \mathcal{O}(n^{\log_2 8})$$

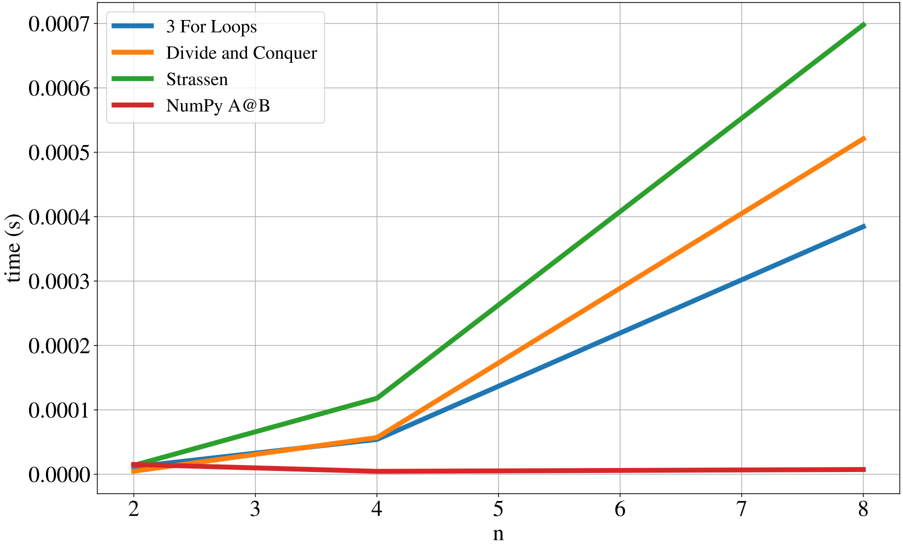
Algorithm

Algorithm 16 Strassen Matrix Multiplication

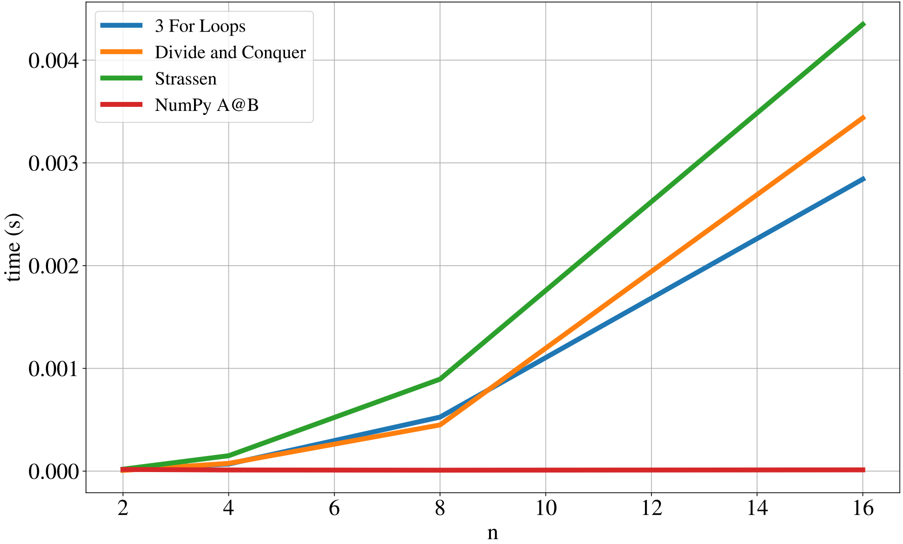
```
function MM(A, B, n)
    if n = 2 then
        C ← zeros((n, n))
        C[0,0] ← A[0][0] * B[0][0] + A[0][1] * B[1][0]
        C[0,1] ← A[0][0] * B[0][1] + A[0][1] * B[1][1]
        C[1,0] ← A[1][0] * B[0][0] + A[1][1] * B[1][0]
        C[1,1] ← A[1][0] * B[0][1] + A[1][1] * B[1][1]
    else
        m ← n/2
        A11, A12, A21, A22 ← A[: m][: m], A[: m][m :], A[m :][: m], A[m :][m :]
        B11, B12, B21, B22 ← B[: m][: m], B[: m][m :], B[m :][: m], B[m :][m :]
        C11 ← MM(A11, B11) + MM(A12, B21)
        C12 ← MM(A11, B12) + MM(A12, B22)
        C21 ← MM(A21, B11) + MM(A22, B21)
        C22 ← MM(A21, B12) + MM(A22, B22)
        C ← vstack((hstack((C11, C12)), hstack((C21, C22))))
    return C
```

$$\mathcal{T}(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ 8 \cdot \mathcal{T}\left(\frac{n}{2}\right) + n^2 & \text{if } n > 2 \end{cases} = \mathcal{O}(n^3)$$

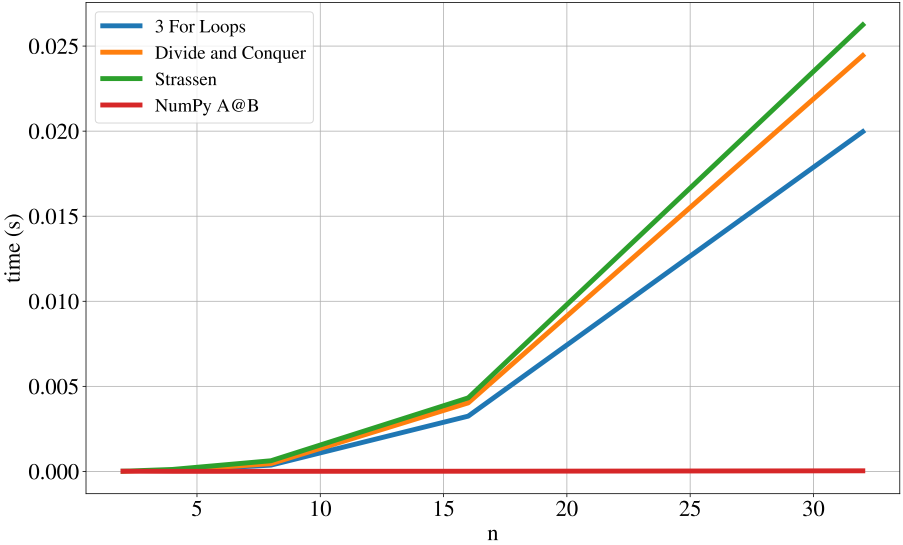
Measurements Python



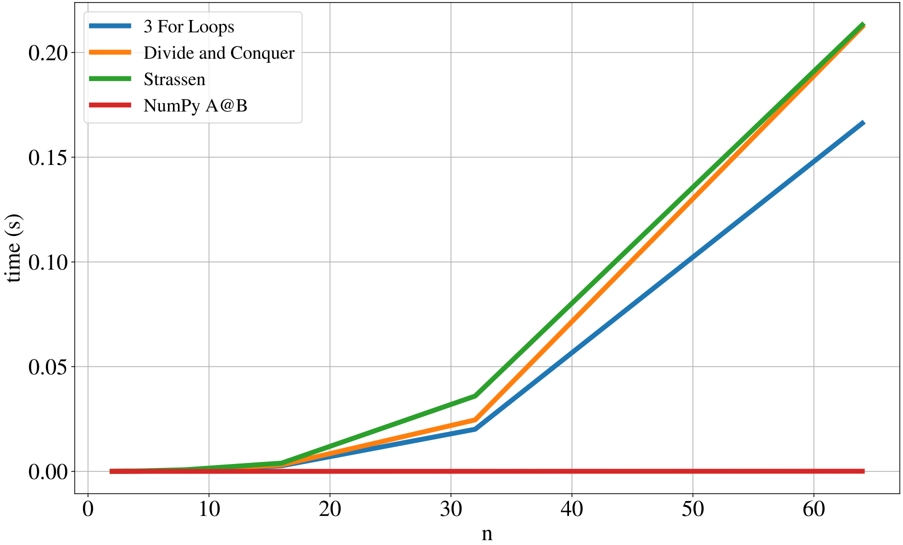
Measurements Python



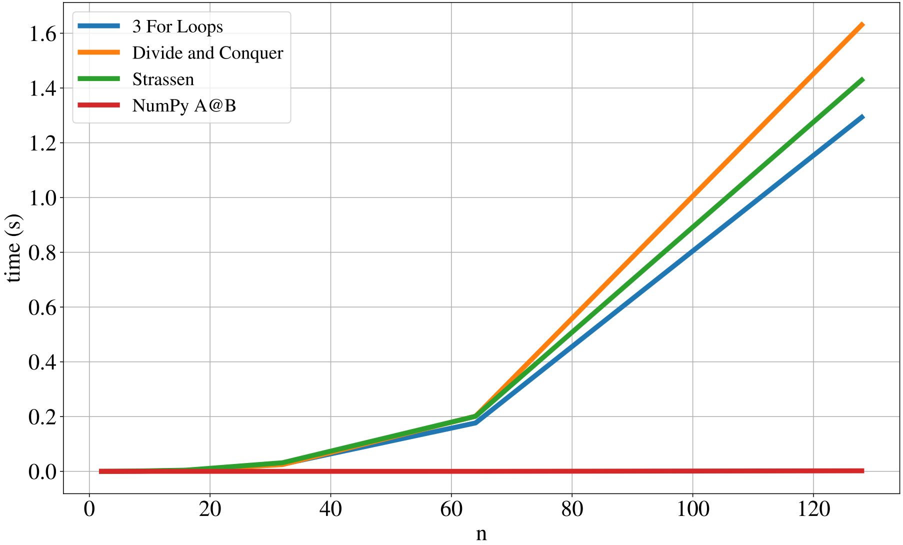
Measurements Python



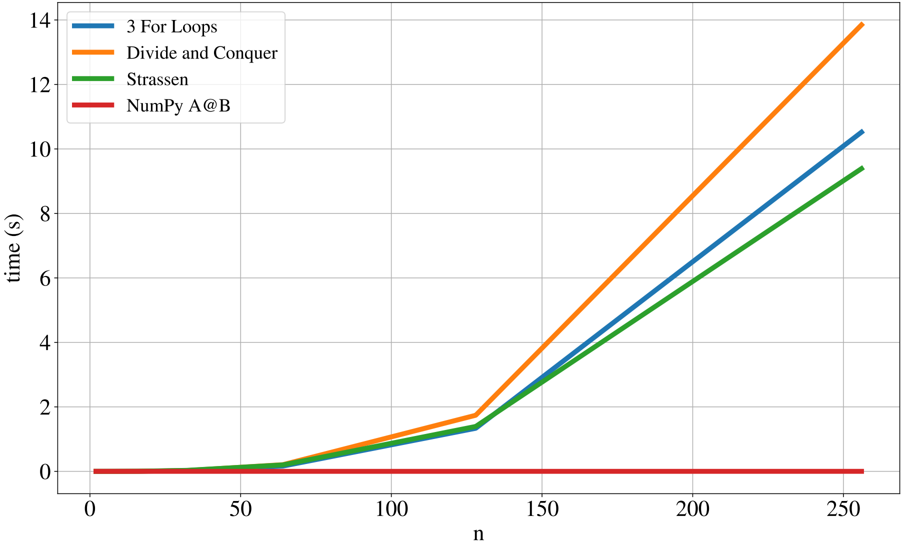
Measurements Python



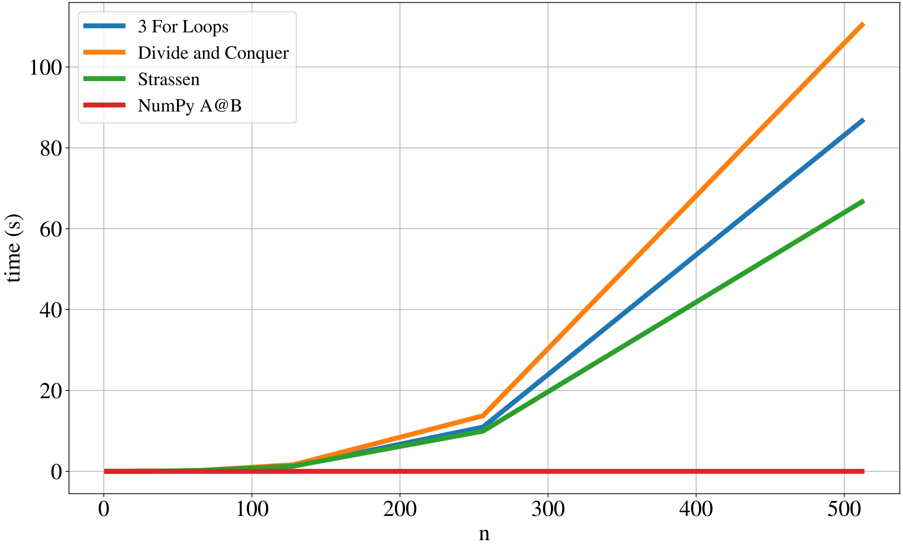
Measurements Python



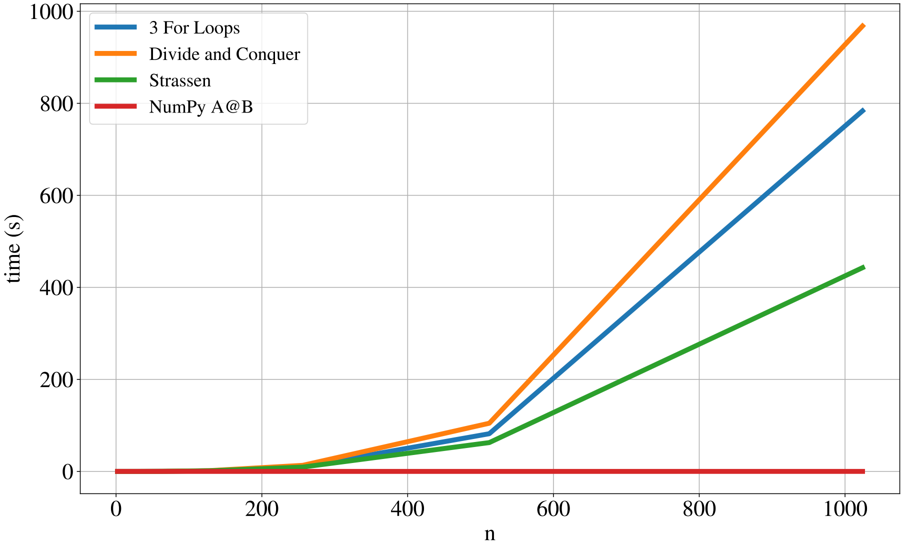
Measurements Python



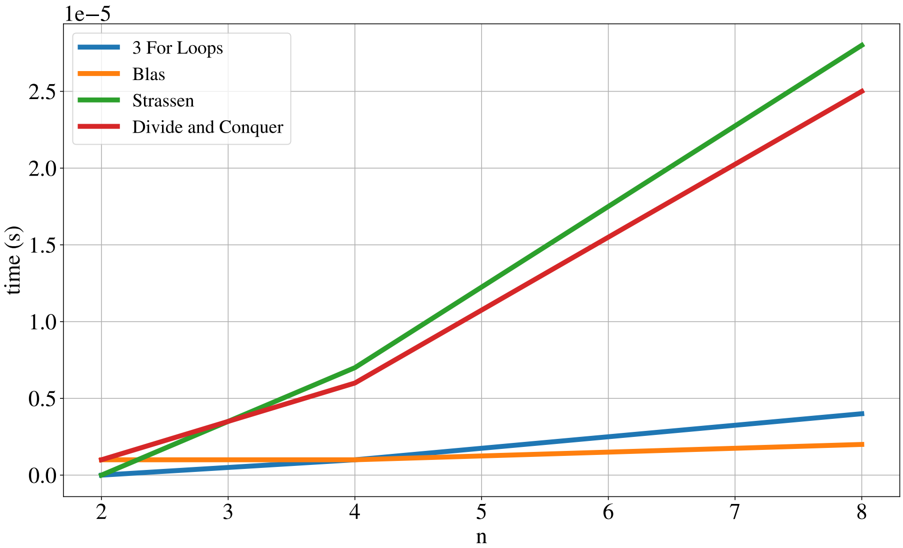
Measurements Python



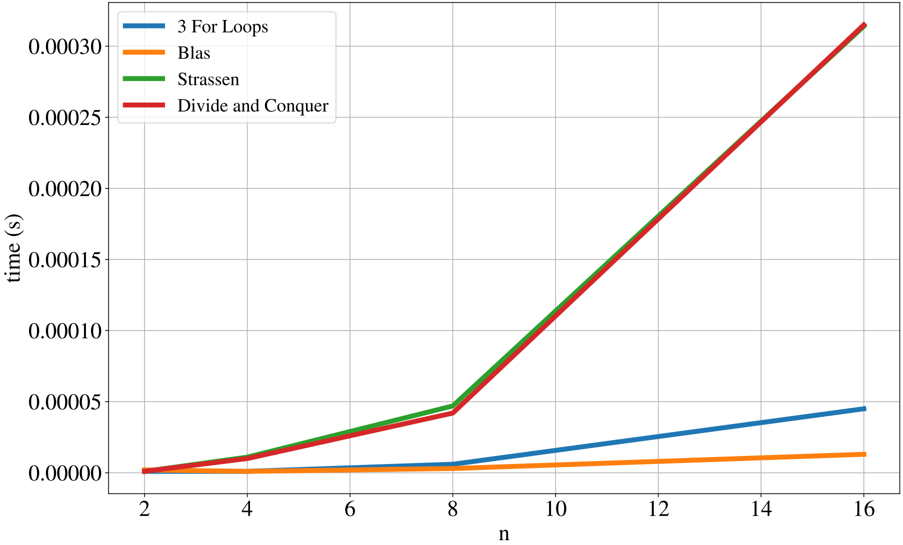
Measurements Python



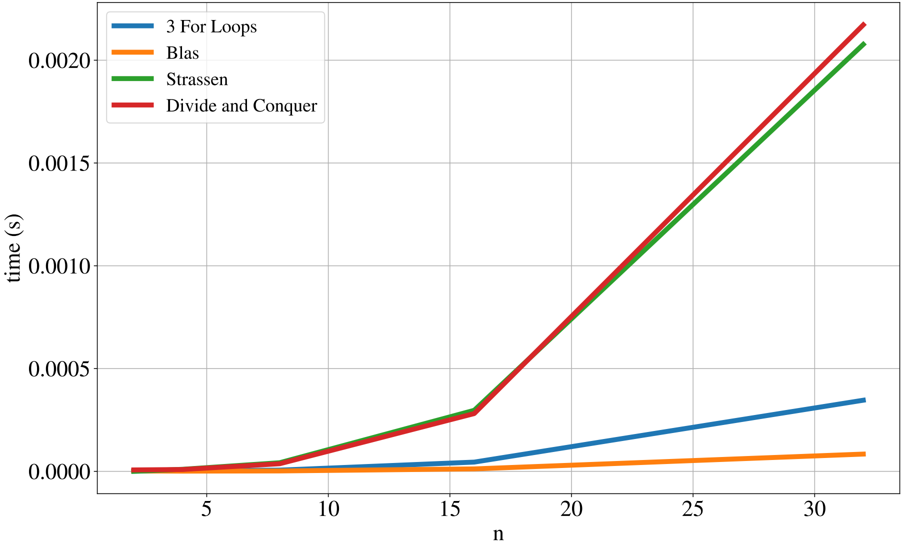
Measurements C



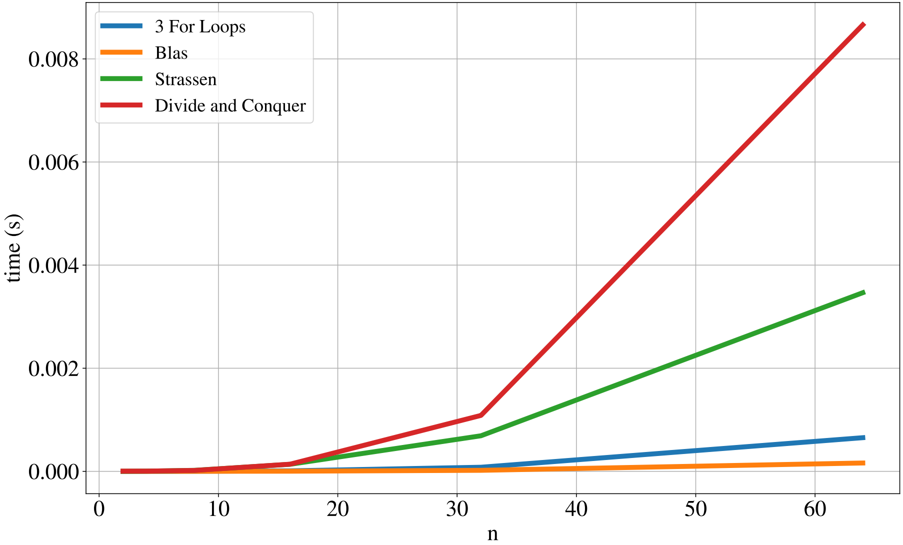
Measurements C



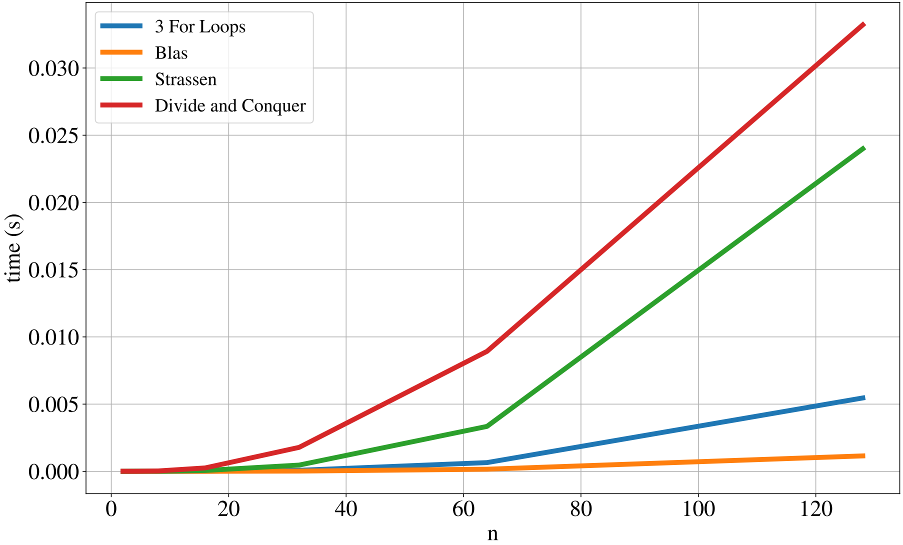
Measurements C



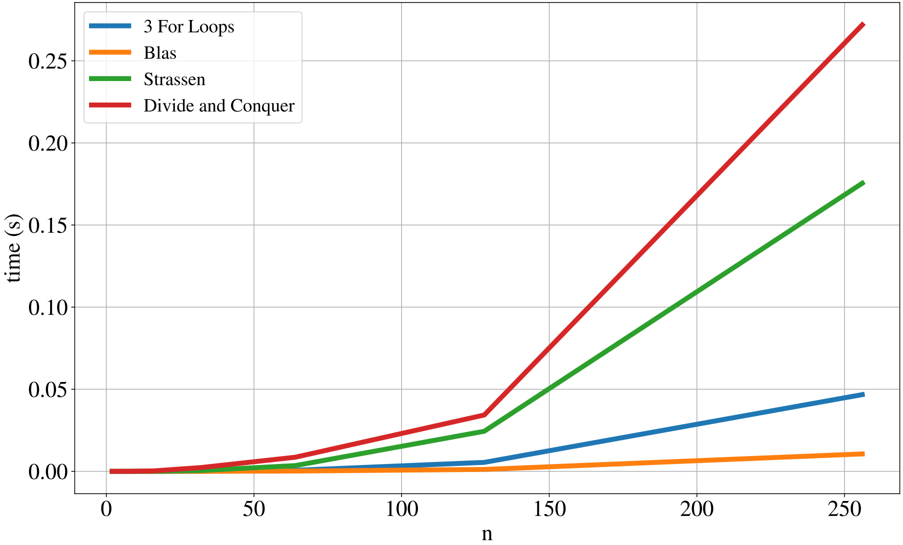
Measurements C



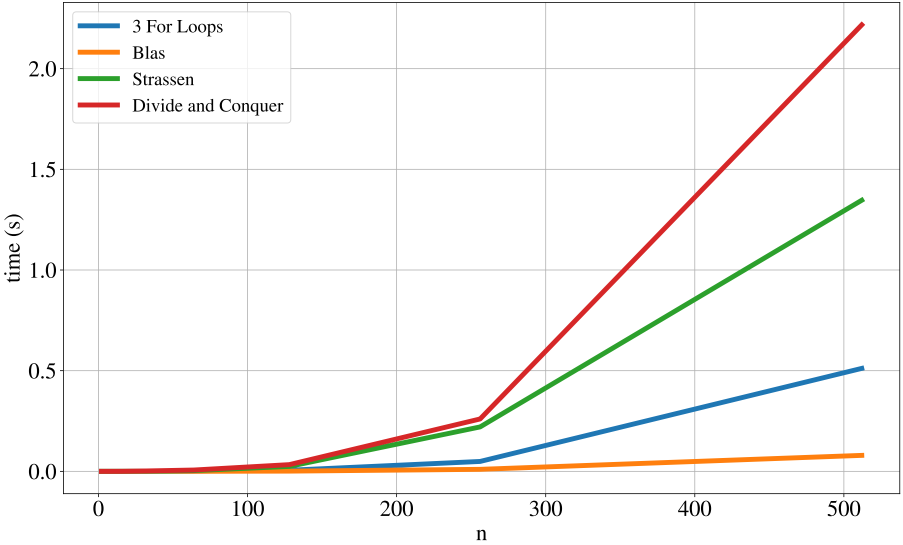
Measurements C



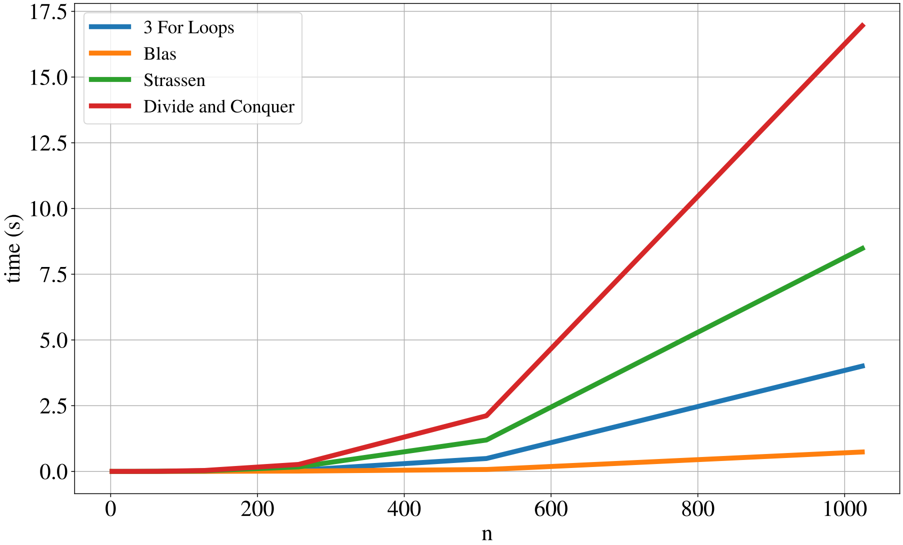
Measurements C



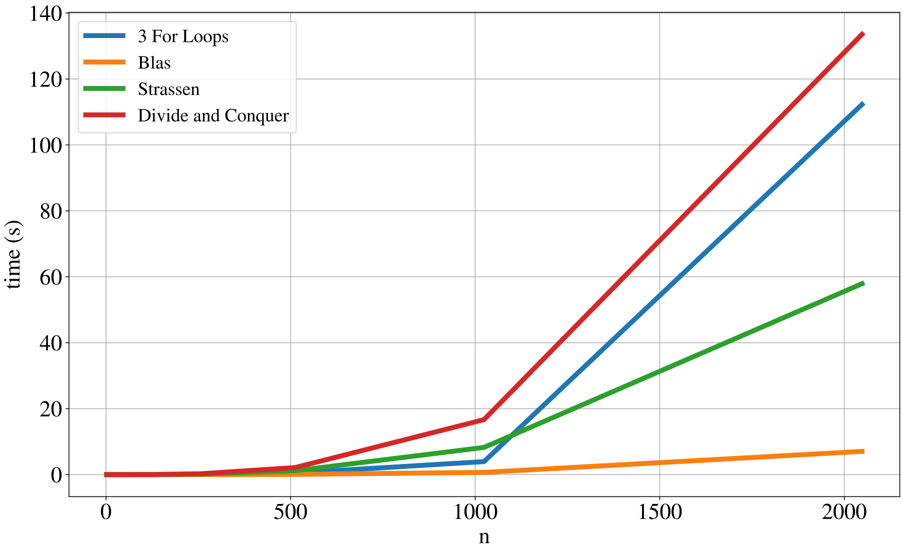
Measurements C



Measurements C



Measurements C



BLAS, LAPACK

- Basic Linear Algebra Subprograms
  - $\mathbf{y} = \alpha\mathbf{x} + \mathbf{y}$
  - $\mathbf{y} = \alpha\mathbf{A}\mathbf{x} + \beta\mathbf{y}$
  - $\mathbf{C} = \alpha\mathbf{A}\mathbf{B} + \beta\mathbf{C}$
- Linear Algebra Package
  - QR decomposition
  - Singular value decomposition
  - Eigenvalues