Matrix Multiplication

$$C = A \cdot B$$

$$\begin{bmatrix} 3 & 1 & 1 \\ 2 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 6 \\ 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 20 \\ 5 & 18 \end{bmatrix}$$

Algorithm for Matrix Multiplication

$$C = A \cdot B$$

$$\begin{bmatrix} 3 & 1 & 1 \\ 2 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 6 \\ 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 20 \\ 5 & 18 \end{bmatrix}$$

Write pseudocode

// input: A**, an** $n \times m$ **matrix and** B**, an** $m \times p$ **matrix**
// output: C**, an** $n \times p$ **matrix**
for $i = 1$ **to** n
    for $j = 1$ **to** p
        $C[i,j] = 0$
        for $k = 1$ **to** m
            $C[i,j] += A[i,k] \cdot B[k,j]$

Analysis

// input: A**, an** $n \times m$ **matrix and** B**, an** $m \times p$ **matrix**
// output: C**, an** $n\times$ **matrix**
for $i = 1$ **to** n
    for $j = 1$ **to** p
        $C[i, j] = 0$
        for $k = 1$ **to** m
            $C[i, j] += A[i, k] \cdot B[k, j]$

Running time

- **3 nested loops**
- $O(nmp)$ **time**
- **if** $n = m = p$ **, then** $O(n^3) time$
- **Lower bound of** $\Omega(n^2)$

Can we do better?

- **We are implementing the standard definition efficiently, what else could we do?**
- **You have to do n^3 operatation, each of n^2 entries of C, involves adding up the result of n multipications.**

Can we do better?

- We are implementing the standard definition efficiently, what else could we do?
- You have to do n^3 operatation, each of n^2 entries of C , involves adding up the result of n multipications.

Maybe divide and conquer can help

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

$$\begin{aligned} r &= ae + bf && (1) \\ s &= ag + bh && (2) \\ t &= ce + df && (3) \\ u &= cg + dh && (4) \end{aligned}$$

Maybe divide and conquer can help

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

$$\begin{aligned} r &= ae + bf && (5) \\ s &= ag + bh && (6) \\ t &= ce + df && (7) \\ u &= cg + dh && (8) \end{aligned}$$

Multiply 2 $n \times n$ matrices takes

- **8 multiplications of $n/2 \times n/2$ matrices**
- **4 additions of $n/2 \times n/2$ matrices**

- **Adding two $n \times n$ matrices takes $O(n^2)$ time**
- **Adding matrices seems easier than multiplying them**

Let's Analyze

Let $T(n)$ **be the time to multiply 2** n **by** n **matrices**

$$T(n) = \begin{cases} 8T(n/2) + 4(n/2)^2 & \textbf{if } n > 1 \\ 1 & \textbf{if } n = 1 \end{cases}$$

Let's Analyze

Let $T(n)$ **be the time to multiply 2** n **by** n **matrices**

$$T(n) = \begin{cases} 8T(n/2) + 4(n/2)^2 & \textbf{if } n > 1 \\ 1 & \textbf{if } n = 1 \end{cases}$$

By the master theorem, this solves to $O(n^3)$ **.**

But consider the following recurrence

$$T(n) = \begin{cases} 7T(n/2) + 18(n/2)^2 & \textbf{if } n > 1 \\ 1 & \textbf{if } n = 1 \end{cases}$$

As we will learn, this solves to $O(n^{\log_2 7}) = O(n^{2.81..})$ **.**

But can we multiply 2 $n \times n$ **matrices by doing** 7 **multiplications of** $n/2 \times n/2$ **matrices and** 18 **additions of** $n/2 \times n/2$ **matrices.**

Strassen's Algorithm

To Compute

$$
\begin{aligned}
r &= ae + bf && (9)\\
s &= ag + bh && (10)\\
t &= ce + df && (11)\\
u &= cg + dh && (12)
\end{aligned}
$$

Calculations

$$
\begin{aligned}
P_1 &= a(g-h) &&= ag - ah\\
P_2 &= (a+b)h &&= ah + bh
\end{aligned}
$$

$$s = P_1 + P_2$$

$$
\begin{aligned}
P_3 &= (c+d)e &&= ce + de\\
P_4 &= d(f-e) &&= df - de
\end{aligned}
$$

$$t = P_3 + P_4$$

$$
\begin{aligned}
P_5 &= (a+d)(e+h) &&= ae + ah + de + dh\\
P_6 &= (b-d)(h+f) &&= -dh - df + bh + bf
\end{aligned}
$$

$$r = P_5 + P_4 - P_2 + P_6$$

$$P_7 = (a-c)(e+g) = ae + ag - ce - cg$$

$$u = P_5 + P_1 - P_3 - P_7$$

p