CSE 421

Divide and Conquer

Yin Tat Lee
Problem 4 (20 points).
Given an array of positive numbers $a = [a_1, a_2, \cdots, a_n]$ . Give an $O(n \log n)$ time algorithm that find $i$ and $j$ (with $i \leq j$) that maximize the subarray product $\prod_{k=i}^{j} a_k$. Prove the correctness and the runtime of the algorithm.

For example, in the array $a = [3, 0.2, 5, 7, 0.4, 4, 0.01]$, the sub-array from $i = 3$ to $j = 6$ has the product $5 \times 7 \times 0.4 \times 4 = 56$ and no other sub-array contains elements that product to a value greater than 56. So, the answer for this input is $i = 3, j = 6$.

Hints: Divide and Conquer.

Lecture 10 Quiz #95

Yin-Tat Lee [STAFF]
a day ago in Lectures

I was wrong about the maximum product subarray. You can solve using a sliding window.

Remark: I can be wrong. So feel free to debate with me after the lecture if you think you have a better algo!

Here is the algorithm provided by Adam Wang and Alan Wu:

$L = 1, R = 1$, Answer $= x_L$. AnswerIdx $= [L, R]$

For $R = 1, 2, \ldots, N$

- Let $P = \prod_{i=L}^{R} x_i$.
- if $P < x_R$
  - set $L=R$
- if Answer $< P$
  - set Answer $= P$. Set AnswerIdx $= [L, R]$

Output AnswerIdx

Runtime: $O(n)$ by maintaining the product.
Master Theorem
Proving Master Theorem

Problem size

\[ T(n) = aT\left(\frac{n}{b}\right) + cn^k \]

\begin{align*}
\text{Problem size} & \quad \text{# probs} & \quad \text{cost} \\
\frac{n}{b} & \quad 1 & \quad cn^k \\
\frac{n}{b^2} & \quad a & \quad a \cdot c\left(\frac{n}{b}\right)^k \\
1 & \quad a^2 & \quad a^2 \cdot c\left(\frac{n}{b^2}\right)^k \\
& \quad a^d & \quad a^d \cdot c\left(\frac{n}{b^d}\right)^k \\
\end{align*}

\[ T(n) = \sum_{i=0}^{d=\log_b n} a^i c \left(\frac{n}{b^i}\right)^k \]
Master Theorem

Suppose \( T(n) = a \, T \left( \frac{n}{b} \right) + cn^k \) for all \( n > b \). Then,

- If \( a < b^k \) then \( T(n) = \Theta(n^k) \) # of problems increases slower than the decreases of cost. First term dominates.
- If \( a = b^k \) then \( T(n) = \Theta(n^k \log n) \)
- If \( a > b^k \) then \( T(n) = \Theta(n^{\log_b a}) \) # of problems increases faster than the decreases of cost Last term dominates.
A Useful Identity

Theorem: \(1 + x + x^2 + \cdots + x^d = \frac{x^{d+1} - 1}{x-1}\)

Proof: Let \(S = 1 + x + x^2 + \cdots + x^d\)

Then, \(xS = x + x^2 + \cdots + x^{d+1}\)

So, \(xS - S = x^{d+1} - 1\)

i.e., \(S(x - 1) = x^{d+1} - 1\)

Therefore, \(S = \frac{x^{d+1} - 1}{x-1}\)

Corollary:

\[
1 + x + x^2 + \cdots + x^d = \begin{cases} 
O_x(1) & \text{if } x < 1 \\
 d + 1 & \text{if } x = 1 \\
O_x(x^{d+1}) & \text{if } x > 1
\end{cases}
\]

\(O_x\) means the hidden constant depends on \(x\)
Solve: $T(n) = aT\left(\frac{n}{b}\right) + cn^k$

Corollary:

$$1 + x + x^2 + \cdots + x^d = \begin{cases} \Theta(x(1)) & \text{if } x < 1 \\ \Theta(d) & \text{if } x = 1 \\ \Theta(x^{d+1}) & \text{if } x > 1 \end{cases}$$

Going back, we have

$$T(n) = \sum_{i=0}^{d=\log_b n} a^i c \left(\frac{n}{b^i}\right)^k = cn^k \sum_{i=0}^{d=\log_b n} \left(\frac{a}{b^k}\right)^i$$

Hence, we have

$$T(n) = \Theta(n^k) \begin{cases} 1 & \text{if } a < b^k \\ \log_b n & \text{if } a = b^k \\ \left(\frac{a}{b^k}\right)^{\log_b n} & \text{if } a > b^k \end{cases}$$

constant depends on $a, b, c$
Solve: \( T(n) = aT \left( \frac{n}{b} \right) + cn^k \)

\[
T(n) = \Theta(n^k) \begin{cases} 
1 & \text{if } a < b^k \\
\log_b n & \text{if } a = b^k \\
\left( \frac{a}{b^k} \right)^{\log_b n} & \text{if } a > b^k
\end{cases}
\]

For \( a < b^k \), we simply have \( T(n) = \Theta(n^k) \).

For \( a = b^k \), we have \( T(n) = \Theta(n^k \log_b n) = \Theta(n^k \log n) \).

For \( a > b^k \), we have \( T(n) = \Theta \left( n^k \left( \frac{a}{b^k} \right)^{\log_b n} \right) = \Theta(n^{\log_b a}) \).

\[
\begin{align*}
b^k \log_b n & = (b^{\log_b n})^k \\
& = n^k
\end{align*}
\]

\[
\begin{align*}
a^{\log_b n} & = (b^{\log_b a})^{\log_b n} \\
& = (b^{\log_b n})^{\log_b a} \\
& = n^{\log_b a}
\end{align*}
\]
Finding the Closest Pair of Points
Closest Pair of Points (1-dimension)

Given \( n \) points on the real line, find the closest pair, e.g., given 11, 2, 4, 19, 4.8, 7, 8.2, 16, 11.5, 13, 1 find the closest pair

Fact: Closest pair is adjacent in ordered list
So, first sort, then scan adjacent pairs.
Time \( O(n \log n) \) to sort, if needed, Plus \( O(n) \) to scan adjacent pairs

Key point: do not need to calculate distances between all pairs: exploit geometry + ordering
Closest Pair of Points (2-dimensions)

Given $n$ points in the plane, find a pair with smallest Euclidean distance between them.

**Fundamental geometric primitive.**

Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.

Special case of nearest neighbor, Euclidean MST, Voronoi.

**Brute force:** Check all pairs of points in $\Theta(n^2)$ time.

**Assumption:** No two points have same $x$ coordinate.
Closest Pair of Points (2-dimensions)

No single direction along which one can sort points to guarantee success!
Divide & Conquer

Divide: draw vertical line $L$ with $\approx n/2$ points on each side.
Conquer: find closest pair on each side, recursively.
Combine to find closest pair overall
Return best solutions
Suppose $\delta$ is the minimum distance of all pairs in left/right of $L$.

\[ \delta = \min(12,21) = 12. \]

**Key Observation**: suffices to consider points within $\delta$ of line $L$.

Almost the one-D problem again: Sort points in $2\delta$-strip by their $y$ coordinate.

Why the strip problem is easier?
Almost 1D Problem

Partition each side of $L$ into $\frac{\delta}{2} \times \frac{\delta}{2}$ squares

Claim: No two points lie in the same $\frac{\delta}{2} \times \frac{\delta}{2}$ box.

Proof: Such points would be within

$$\sqrt{\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2} = \delta \sqrt{\frac{1}{2}} \approx 0.7\delta < \delta$$

Let $s_i$ have the $i^{th}$ smallest $y$-coordinate among points in the $2\delta$-width-strip.

Claim: If $|i - j| > 11$, then the distance between $s_i$ and $s_j$ is $> \delta$.

Proof: only 11 boxes within $\delta$ of $y(s_i)$. 
**Closest Pair (2 dimension)**

```
Closest-Pair(p_1,p_2,\cdots,p_n) {
    if(n \leq 2) return |p_1 - p_2|

    Compute separation line L such that half the points are on one side and half on the other side.

    \[ \delta_1 = \text{Closest-Pair(left half)} \]
    \[ \delta_2 = \text{Closest-Pair(right half)} \]
    \[ \delta = \min(\delta_1, \delta_2) \]

    Delete all points further than \( \delta \) from separation line L

    Sort remaining points \( p[1] \cdots p[m] \) by y-coordinate.

    for \( i = 1,2,\cdots,m \)
        for \( k = 1,2,\cdots,11 \)
            if \( i + k \leq m \)
                \[ \delta = \min(\delta, \text{distance}(p[i], p[i+k])) \]

    return \( \delta \).
}
```
Closest Pair Analysis

Let $D(n)$ be the number of pairwise distance calculations in the Closest-Pair Algorithm

$$D(n) \leq \begin{cases} 
1 & \text{if } n = 1 \\
2D\left(\frac{n}{2}\right) + 11n & \text{o.w.} \Rightarrow D(n) = O(n \log n)
\end{cases}$$

BUT, that’s only the number of distance calculations

What if we counted running time?

$$T(n) \leq \begin{cases} 
1 & \text{if } n = 1 \\
2T\left(\frac{n}{2}\right) + O(n \log n) & \text{o.w.} \Rightarrow T(n) = O(n \log^2 n)
\end{cases}$$
Closest Pair (2 dimension) Improved

Closest-Pair\((p_1, p_2, \ldots, p_n)\) {
  if\((n \leq 2)\) return \(|p_1 - p_2|\)
  
  Compute separation line \(L\) such that half the points are on one side and half on the other side.

  \((\delta_1, p_1) = \text{Closest-Pair(}\text{left half})\)
  \((\delta_2, p_2) = \text{Closest-Pair(}\text{right half})\)
  \(\delta = \min(\delta_1, \delta_2)\)
  \(p_{\text{sorted}} = \text{merge}(p_1, p_2)\) (merge sort it by y-coordinate)

  Let \(q\) be points (ordered as \(p_{\text{sorted}}\)) that is \(\delta\) from line \(L\).

  for \(i = 1, 2, \ldots, m\)
    for \(k = 1, 2, \ldots, 11\)
      if \(i + k \leq m\)
        \(\delta = \min(\delta, \text{distance}(q[i], q[i+k]))\);

  return \(\delta\) and \(p_{\text{sorted}}\).
}

\[ T(n) \leq \begin{cases} 
 1 & \text{if } n = 1 \\
 2T\left(\frac{n}{2}\right) + O(n) & \text{o.w.} 
\end{cases} \]

\[ \Rightarrow T(n) = O(n \log n) \]
Quiz

How to solve closest pair in 3 dimension?
Closest-Pair\((p_1, p_2, \ldots, p_n)\) 
\[
\text{if} (n \leq 2) \text{ return } |p_1 - p_2|
\]

Compute separation line \(L\) such that half the points are on one side and half on the other side.

\[
\delta_1 = \text{Closest-Pair(left half)} \\
\delta_2 = \text{Closest-Pair(right half)} \\
\delta = \min(\delta_1, \delta_2)
\]

Delete all points further than \(\delta\) from separation line \(L\)

Put points into \(\frac{\delta}{2} \times \frac{\delta}{2} \times \frac{\delta}{2}\) cubes (via hash table)

for \(i = 1, 2, \ldots, m\)
   Let \((a, b, c)\) be the cube for \(p[i]\).
   for \(x, y, z = -3, -2, 1, 0, 1, 2, 3\)
      check the cube \((a+x, b+y, c+z)\)
      if there is a point \(q\) in the cube,
      \[
      \delta = \min(\delta, \text{distance}(p[i], q))
      \]
   return \(\delta\).

\[
\text{In } d \text{ dimension, the runtime is } \\
T(n) = 2^{O(d)}n \log n
\]
Median
Selecting k-th smallest

Problem: Given numbers $x_1, ..., x_n$ and an integer $1 \leq k \leq n$
output the $k$-th smallest number

\[
\text{Sel}(\{x_1, ..., x_n\}, k)
\]

A simple algorithm: Sort the numbers in time $O(n \log n)$ then
return the $k$-th smallest in the array.

Can we do better?

Yes, in time $O(n)$ if $k = 1$ or $k = 2$.

Can we do $O(n)$ for all possible values of $k$?
Choose a number $w$ from $x_1, \ldots, x_n$

Define

- $S_<(w) = \{x_i : x_i < w\}$
- $S_=(w) = \{x_i : x_i = w\}$
- $S_>(w) = \{x_i : x_i > w\}$

Solve the problem recursively as follows:

- If $k \leq |S_<(w)|$, output $Sel(S_<(w), k)$
- Else if $k \leq |S_<(w)| + |S_=(w)|$, output $w$
- Else output $Sel(S_>(w), k - |S_<(w)| - |S_=(w)|)$

Ideally want $|S_<(w)|, |S_>(w)| \leq n/2$. In this case ALG runs in $O(n) + O\left(\frac{n}{2}\right) + O\left(\frac{n}{4}\right) + \cdots + O(1) = O(n)$. 

Can be computed in linear time
How to choose w?

Suppose we choose w uniformly at random similar to the pivot in quicksort. Then, \( \mathbb{E}[|S_<(w)|] = \mathbb{E}[|S_>(w)|] = n/2 \). Algorithm runs in \( O(n) \) in expectation. Can we get \( O(n) \) running time deterministically?

- Partition numbers into sets of size 3.
- Sort each set (takes \( O(n) \))
- \( w = Sel(midpoints, n/6) \)
Assume all numbers are distinct for simplicity.

How to lower bound $|S_<(w)|, |S_>(w)|$?

• $|S_<(w)| \geq 2 \left( \frac{n}{6} \right) = \frac{n}{3}$

• $|S_>(w)| \geq 2 \left( \frac{n}{6} \right) = \frac{n}{3}$.

So, what is the running time?
Assume all numbers are distinct for simplicity.

Asymptotic Running Time?

- If $k \leq |S_{<}(w)|$, output $Sel(S_{<}(w), k)$
- Else if $k \leq |S_{<}(w)| + |S_{=} (w)|$, output $w$
- Else output $Sel(S_{>}(w), k - |S_{<}(w)| - |S_{=} (w)|)$

Where $\frac{n}{3} \leq |S_{<}(w)|, |S_{>}(w)| \leq \frac{2n}{3}$

$$T(n) = T \left( \frac{n}{3} \right) + T \left( \frac{2n}{3} \right) + O(n) \Rightarrow T(n) = O(n \log n)$$
An Improved Idea

Partition into n/5 sets. Sort each set and set \( w = \text{Sel(midpoints, n/10)} \)

- \( |S_{<}(w)| \geq 3 \left( \frac{n}{10} \right) = \frac{3n}{10} \) \quad \frac{3n}{10} \leq |S_{<}(w)|, |S_{>}(w)| \leq \frac{7n}{10} 
- \( |S_{>}(w)| \geq 3 \left( \frac{n}{10} \right) = \frac{3n}{10} \)

\[ T(n) = T\left( \frac{n}{5} \right) + T\left( \frac{7n}{10} \right) + O(n) \Rightarrow T(n) = O(n) \]
Can we do it even better?

Goal: Finding median.

Fix $\epsilon$.
Randomly select $T/\epsilon^2$ elements.
Output the median of these $T/\epsilon^2$ elements.

One can prove that it will gives an element with rank

$$(1/2 - \epsilon)n \text{ and } (1/2 + \epsilon)n$$

with probability at least $1 - \exp(-T)$.

Think $\epsilon = 0.1$ and $T = 30$.
Then, we have almost median with high prob in $O(1)$ time.
Integer Multiplication
**Integer Arithmetic**

**Add:** Given two $n$-bit integers $a$ and $b$, compute $a + b$.

$O(n)$ bit operations.

**Multiply:** Given two $n$-bit integers $a$ and $b$, compute $a \times b$.

The “grade school” method:

$O(n^2)$ bit operations.
Divide and Conquer

Let $x, y$ be two $n$-bit integers
Write $x = 2^{n/2}x_1 + x_0$ and $y = 2^{n/2}y_1 + y_0$
where $x_0, x_1, y_0, y_1$ are all $n/2$-bit integers.

$x = 2^{n/2} \cdot x_1 + x_0$
$y = 2^{n/2} \cdot y_1 + y_0$
$xy = (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0)$
$\quad \quad \quad = 2^n \cdot x_1y_1 + 2^{n/2} \cdot (x_1y_0 + x_0y_1) + x_0y_0$

Therefore,

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$$

So,

$$T(n) = \Theta(n^2).$$
Key Trick: 4 multiplies at the price of 3

\[ x = 2^{n/2} \cdot x_1 + x_0 \]
\[ y = 2^{n/2} \cdot y_1 + y_0 \]
\[ xy = (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \]
\[ = 2^n \cdot x_1 y_1 + 2^{n/2} (x_1 y_0 + x_0 y_1) + x_0 y_0 \]

\[ \alpha = x_1 + x_0 \]
\[ \beta = y_1 + y_0 \]
\[ \alpha \beta = (x_1 + x_0)(y_1 + y_0) \]
\[ = x_1 y_1 + (x_1 y_0 + x_0 y_1) + x_0 y_0 \]
\[ (x_1 y_0 + x_0 y_1) = \alpha \beta - x_1 y_1 - x_0 y_0 \]
Key Trick: 4 multiplies at the price of 3

Theorem [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in $O(n^{1.585...})$ bit operations.

$$x = 2^{n/2} \cdot x_1 + x_0 \Rightarrow \alpha = x_1 + x_0$$
$$y = 2^{n/2} \cdot y_1 + y_0 \Rightarrow \beta = y_1 + y_0$$
$$xy = \left(2^{n/2} \cdot x_1 + x_0\right)\left(2^{n/2} \cdot y_1 + y_0\right)$$
$$\quad = 2^n \cdot x_1 y_1 + 2^{n/2} \cdot \left(x_1 y_0 + x_0 y_1\right) + x_0 y_0$$

To multiply two n-bit integers:
- Add two $n/2$ bit integers.
- Multiply three $n/2$-bit integers.
- Add, subtract, and shift $n/2$-bit integers to obtain result.

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O\left(n^{\log_2 3}\right) = O(n^{1.585...})$$
Integer Multiplication (Summary)

- **Exercise**: generalize Karatsuba to do 5 size $n/3$ subproblems
  This gives $\Theta(n^{1.46\ldots})$ time algorithm

<table>
<thead>
<tr>
<th>Date</th>
<th>Authors</th>
<th>Time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;3000 BC</td>
<td>Unknown</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>1962</td>
<td>Karatsuba</td>
<td>$O(n^{\log 3/\log 2})$</td>
</tr>
<tr>
<td>1963</td>
<td>Toom</td>
<td>$O(n^{2^{5/2} \log n/\log 2})$</td>
</tr>
<tr>
<td>1966</td>
<td>Schöhage</td>
<td>$O(n^{2^{\sqrt{\log n/\log 2}} (\log n)^{3/2}})$</td>
</tr>
<tr>
<td>1969</td>
<td>Knuth</td>
<td>$O(n^{2^{\sqrt{\log n/\log 2}} \log n})$</td>
</tr>
<tr>
<td>1971</td>
<td>Schöhage–Strassen</td>
<td>$O(n \log n \log \log n)$</td>
</tr>
<tr>
<td>2007</td>
<td>Fürer</td>
<td>$O(n \log n 2^{O(\log^* n)})$</td>
</tr>
<tr>
<td>2014</td>
<td>Harvey-Hoeven-Lecerf</td>
<td>$O(n \log n 8^{\log^* n})$</td>
</tr>
<tr>
<td>2019</td>
<td>Harvey-Hoeven</td>
<td>$O(n \log n)$</td>
</tr>
</tbody>
</table>
Demonstration of multiplying $1234 \times 5678 = 7006652$ using fast Fourier transforms (FFTs). Number-theoretic transforms in the integers modulo 337 are used, selecting 85 as an 8th root of unity. Base 10 is used in place of base $2^w$ for illustrative purposes.
Matrix Multiplication
Multiplying Matrices

Let $A$ be an $n \times m$ matrix, $B$ be an $m \times p$ matrix.

\[
A = \begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1m} \\
a_{21} & a_{22} & \cdots & a_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nm}
\end{pmatrix}, \quad B = \begin{pmatrix}
b_{11} & b_{12} & \cdots & b_{1p} \\
b_{21} & b_{22} & \cdots & b_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
b_{m1} & b_{m2} & \cdots & b_{mp}
\end{pmatrix}
\]

Then, $C = AB$ is an $n \times p$ matrix

\[
C = \begin{pmatrix}
c_{11} & c_{12} & \cdots & c_{1p} \\
c_{21} & c_{22} & \cdots & c_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
c_{n1} & c_{n2} & \cdots & c_{np}
\end{pmatrix}
\]

such that

\[
c_{ij} = a_{i1}b_{1j} + \cdots + a_{im}b_{mj} = \sum_{k=1}^{m} a_{ik}b_{kj},
\]

Question: Why matrix multiplication is defined in such way?
Multiplying Matrices

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & b_{13} & b_{14} \\
b_{21} & b_{22} & b_{23} & b_{24} \\
b_{31} & b_{32} & b_{33} & b_{34} \\
b_{41} & b_{42} & b_{43} & b_{44}
\end{bmatrix}
= 
\begin{bmatrix}
a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} + a_{14}b_{43} & a_{11}b_{14} + a_{12}b_{24} + a_{13}b_{34} + a_{14}b_{44} \\
a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} + a_{24}b_{43} & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} + a_{24}b_{44} \\
a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} + a_{34}b_{43} & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} + a_{34}b_{44} \\
a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} + a_{44}b_{41} & a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32} + a_{44}b_{42} & a_{41}b_{13} + a_{42}b_{23} + a_{43}b_{33} + a_{44}b_{43} & a_{41}b_{14} + a_{42}b_{24} + a_{43}b_{34} + a_{44}b_{44}
\end{bmatrix}
\]
Multiplying Matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\begin{bmatrix}
  b_{11} & b_{12} & b_{13} & b_{14} \\
  b_{21} & b_{22} & b_{23} & b_{24} \\
  b_{31} & b_{32} & b_{33} & b_{34} \\
  b_{41} & b_{42} & b_{43} & b_{44}
\end{bmatrix}
\]

= 

\[
\begin{bmatrix}
  a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} + a_{14}b_{43} & a_{11}b_{14} + a_{12}b_{24} + a_{13}b_{34} + a_{14}b_{44} \\
  a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} + a_{24}b_{43} & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} + a_{24}b_{44} \\
  a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} + a_{34}b_{43} & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} + a_{34}b_{44} \\
  a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} + a_{44}b_{41} & a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32} + a_{44}b_{42} & a_{41}b_{13} + a_{42}b_{23} + a_{43}b_{33} + a_{44}b_{43} & a_{41}b_{14} + a_{42}b_{24} + a_{43}b_{34} + a_{44}b_{44}
\end{bmatrix}
\]
### Multiplying Matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22} \\
  a_{31} & a_{32} \\
  a_{41} & a_{42}
\end{bmatrix}
\begin{bmatrix}
  b_{11} & b_{12} \\
  b_{12} & b_{14} \\
  b_{21} & b_{22} \\
  b_{23} & b_{24}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31} + a_{14} b_{41} \\
  a_{21} b_{11} + a_{22} b_{21} + a_{23} b_{31} + a_{24} b_{41} \\
  a_{31} b_{11} + a_{32} b_{21} + a_{33} b_{31} + a_{34} b_{41} \\
  a_{41} b_{11} + a_{42} b_{21} + a_{43} b_{31} + a_{44} b_{41}
\end{bmatrix}
\begin{bmatrix}
  a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31} + a_{14} b_{41} \\
  a_{21} b_{11} + a_{22} b_{21} + a_{23} b_{31} + a_{24} b_{41} \\
  a_{31} b_{11} + a_{32} b_{21} + a_{33} b_{31} + a_{34} b_{41} \\
  a_{41} b_{11} + a_{42} b_{21} + a_{43} b_{31} + a_{44} b_{41}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  a_{11} b_{12} + a_{12} b_{22} + a_{13} b_{32} + a_{14} b_{42} \\
  a_{21} b_{12} + a_{22} b_{22} + a_{23} b_{32} + a_{24} b_{42} \\
  a_{31} b_{12} + a_{32} b_{22} + a_{33} b_{32} + a_{34} b_{42} \\
  a_{41} b_{12} + a_{42} b_{22} + a_{43} b_{32} + a_{44} b_{42}
\end{bmatrix}
\begin{bmatrix}
  a_{11} b_{12} + a_{12} b_{22} + a_{13} b_{32} + a_{14} b_{42} \\
  a_{21} b_{12} + a_{22} b_{22} + a_{23} b_{32} + a_{24} b_{42} \\
  a_{31} b_{12} + a_{32} b_{22} + a_{33} b_{32} + a_{34} b_{42} \\
  a_{41} b_{12} + a_{42} b_{22} + a_{43} b_{32} + a_{44} b_{42}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  a_{11} b_{14} + a_{12} b_{24} + a_{13} b_{34} + a_{14} b_{44} \\
  a_{21} b_{14} + a_{22} b_{24} + a_{23} b_{34} + a_{24} b_{44} \\
  a_{31} b_{14} + a_{32} b_{24} + a_{33} b_{34} + a_{34} b_{44} \\
  a_{41} b_{14} + a_{42} b_{24} + a_{43} b_{34} + a_{44} b_{44}
\end{bmatrix}
\begin{bmatrix}
  a_{11} b_{14} + a_{12} b_{24} + a_{13} b_{34} + a_{14} b_{44} \\
  a_{21} b_{14} + a_{22} b_{24} + a_{23} b_{34} + a_{24} b_{44} \\
  a_{31} b_{14} + a_{32} b_{24} + a_{33} b_{34} + a_{34} b_{44} \\
  a_{41} b_{14} + a_{42} b_{24} + a_{43} b_{34} + a_{44} b_{44}
\end{bmatrix}
\]

\[= \begin{bmatrix}
  a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31} + a_{14} b_{41} \\
  a_{21} b_{11} + a_{22} b_{21} + a_{23} b_{31} + a_{24} b_{41} \\
  a_{31} b_{11} + a_{32} b_{21} + a_{33} b_{31} + a_{34} b_{41} \\
  a_{41} b_{11} + a_{42} b_{21} + a_{43} b_{31} + a_{44} b_{41}
\end{bmatrix}
\begin{bmatrix}
  a_{11} b_{12} + a_{12} b_{22} + a_{13} b_{32} + a_{14} b_{42} \\
  a_{21} b_{12} + a_{22} b_{22} + a_{23} b_{32} + a_{24} b_{42} \\
  a_{31} b_{12} + a_{32} b_{22} + a_{33} b_{32} + a_{34} b_{42} \\
  a_{41} b_{12} + a_{42} b_{22} + a_{43} b_{32} + a_{44} b_{42}
\end{bmatrix} + \begin{bmatrix}
  a_{11} b_{14} + a_{12} b_{24} + a_{13} b_{34} + a_{14} b_{44} \\
  a_{21} b_{14} + a_{22} b_{24} + a_{23} b_{34} + a_{24} b_{44} \\
  a_{31} b_{14} + a_{32} b_{24} + a_{33} b_{34} + a_{34} b_{44} \\
  a_{41} b_{14} + a_{42} b_{24} + a_{43} b_{34} + a_{44} b_{44}
\end{bmatrix} + \begin{bmatrix}
  a_{11} b_{14} + a_{12} b_{24} + a_{13} b_{34} + a_{14} b_{44} \\
  a_{21} b_{14} + a_{22} b_{24} + a_{23} b_{34} + a_{24} b_{44} \\
  a_{31} b_{14} + a_{32} b_{24} + a_{33} b_{34} + a_{34} b_{44} \\
  a_{41} b_{14} + a_{42} b_{24} + a_{43} b_{34} + a_{44} b_{44}
\end{bmatrix} + \begin{bmatrix}
  a_{11} b_{14} + a_{12} b_{24} + a_{13} b_{34} + a_{14} b_{44} \\
  a_{21} b_{14} + a_{22} b_{24} + a_{23} b_{34} + a_{24} b_{44} \\
  a_{31} b_{14} + a_{32} b_{24} + a_{33} b_{34} + a_{34} b_{44} \\
  a_{41} b_{14} + a_{42} b_{24} + a_{43} b_{34} + a_{44} b_{44}
\end{bmatrix}
\]
Simple Divide and Conquer

\[
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{pmatrix}
= 
\begin{pmatrix}
A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\
A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22}
\end{pmatrix}
\]

- \( T(n) = 8T(n/2) + 4 \left( \frac{n}{2} \right)^2 = 8T(n/2) + n^2 \)

So, \( T(n) = \Theta(n^{\log_2 8}) = \Theta(n^3) \)
Strassen’s Divide and Conquer Algorithm

- **Strassen’s algorithm**

  Multiply $2 \times 2$ matrices using 7 instead of 8 multiplications (and 18 additions)

  \[ T(n) = 7T\left(\frac{n}{2}\right) + 18n \]

  Hence, we have $T(n) = O(n^{\log_2 7})$.

  Useful when $n \sim 500$.

One of the most important open problem:
Solve matrix multiplication in $O(n^2 \log^{O(1)} n)$ time
Strassen’s Divide and Conquer Algorithm

Naive

\[
\begin{align*}
C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1} \\
C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\
C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1} \\
C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2}
\end{align*}
\]

Strassen

\[
\begin{align*}
M_1 &= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}) \\
M_2 &= (A_{2,1} + A_{2,2})B_{1,1} \\
M_3 &= A_{1,1}(B_{1,2} - B_{2,2}) \\
M_4 &= A_{2,2}(B_{2,1} - B_{1,1}) \\
M_5 &= (A_{1,1} + A_{1,2})B_{2,2} \\
M_6 &= (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}) \\
M_7 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2})
\end{align*}
\]

\[
\begin{align*}
C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\
C_{1,2} &= M_3 + M_5 \\
C_{2,1} &= M_2 + M_4 \\
C_{2,2} &= M_1 - M_2 + M_3 + M_6
\end{align*}
\]

How did Strassen come up with his matrix multiplication method?

Stackexchange: I've been told no-one really knows, anything would be mainly speculation.