Divide & Conquer Algorithmic Approach

Recall: Mergesort is the exemplar

Recall: Under certain circumstances, the Master Theorem helps us analyze these algorithms.

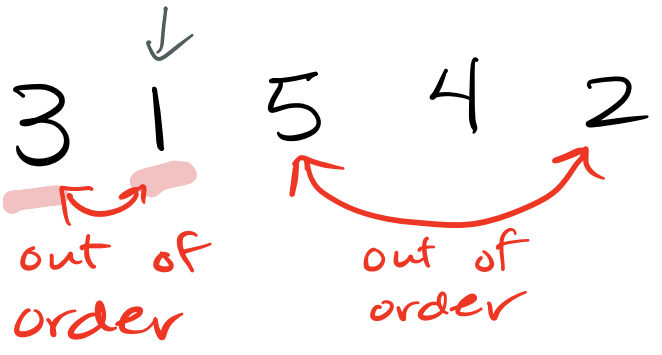
Divide & Conquer Paradigm

1. Divide the input into smaller subproblems
2. Conquer the subproblems, using recursion
3. Combine the solutions of subproblems into solution for original problem.

Problem: Counting Inversions

Input: array A of n distinct integers.

Output: the number of inversions of A



Inversions are:

1,3 2,3 2,5 2,4 4,5 #inversions = 5

Why are we interested?
- recommendation algorithms

Brute Force algorithm to count inversions:

```
BF_CountInversions(array A)
/* Output = number of pairs of elements
/* that are out of order in A */

numInv = 0
for i = 1 to n-1 do
    for j = i+1 to n do
        if A[i] > A[j]
            numInv ++
return numInv.
```

7 ... 3
i j

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ $\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$

Running time? $O\left(\sum_{i=1}^{n-1} i\right) = O(n^2)$

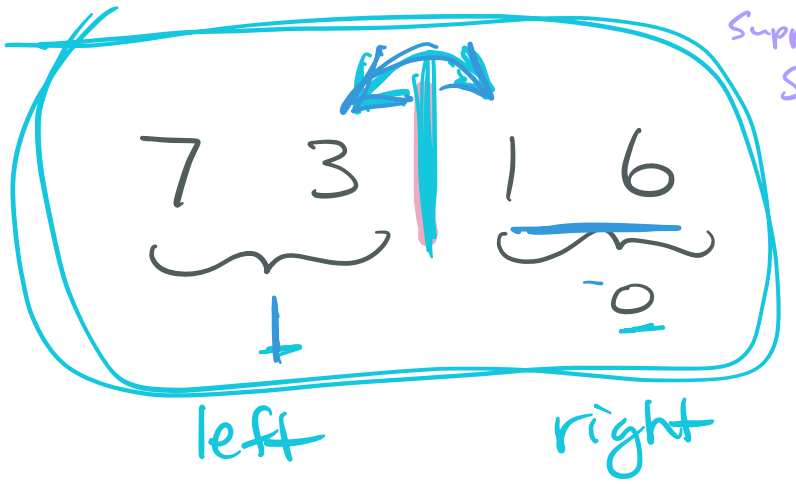
Counting Inversions using Divide & Conquer

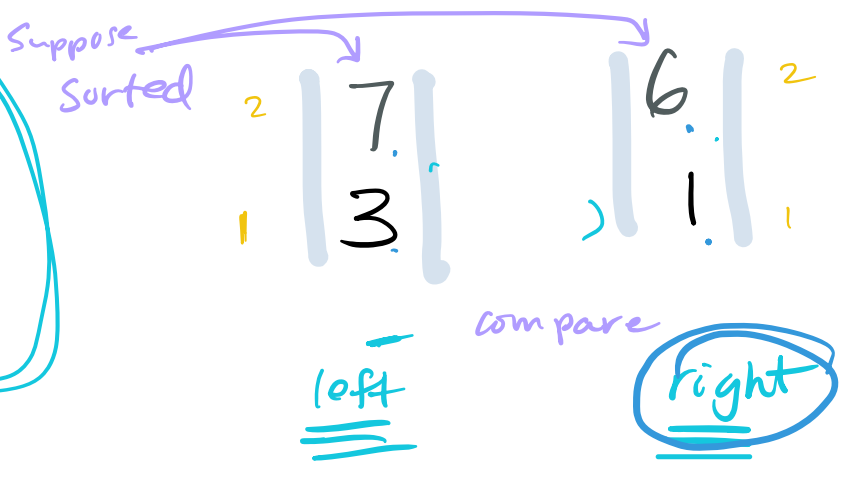
7 3 1 6 | 5 4 2 8

recursion: 4 recursion: 2+1=3

- Divide A into left-half and right half
- count inversions in left half
- count inversions in right half
- count inversions between the two halves ("split" inversions)

return sum of these 3.





element i	1	3	6	7
# split inversions with i		0	1	0

MergeAndCountSplitInversions(C, D)

/* C and D are sorted lists of length $\frac{n}{2}$

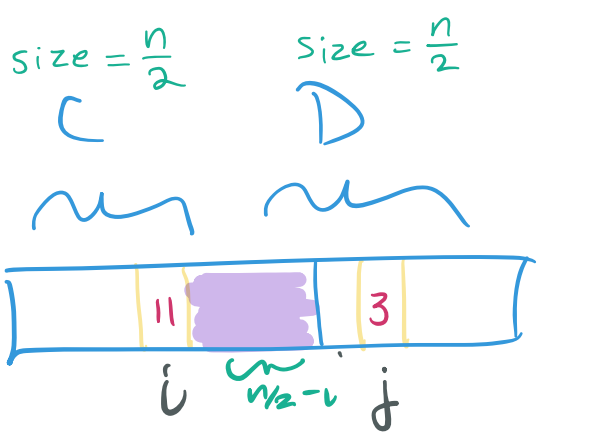
/* Sum, $\forall\ d \in D$, num elements $c \in C$ where

/* $d < c$, and output an array B of all elements

/* of $C \cup D$ in ~~sorted~~ order.

/* simplifying assumption: $|C| = |D|$

```
i = 1 ; j = 1 ; splitInv = 0
for K = 1 to n do
```




```
    if C[i] < D[j]
        B[K] = C[i]
        i = i + 1
```

What is the number of inversions involving D[j] in this scenario?

```
    else
        B[K] = D[j] ; j = j + 1
        splitInv += n/2 - i + 1
return (B, splitInv)
```

MergeSortCountInversions $(A[1..n])$

/* A is an array of n integers

/* simplifying assumption: n is even */

⋮

for you to do

⋮

Running Time:

MergeSort analysis using Master Theorem:

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad n \in \Theta(n^1)$$

$$\Rightarrow \Theta(n \lg n)$$

MergeSort CountInversions analysis using Master Theorem:

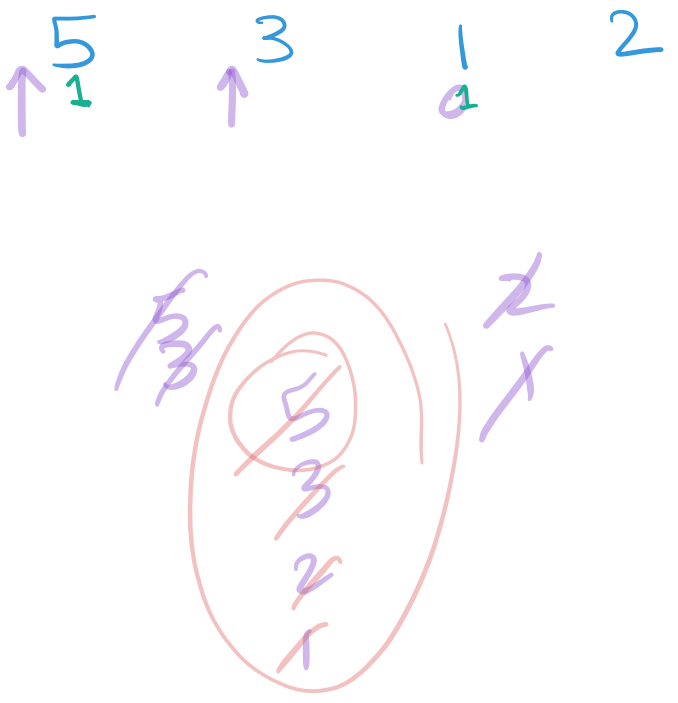
$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad \Theta(n \lg n)$$

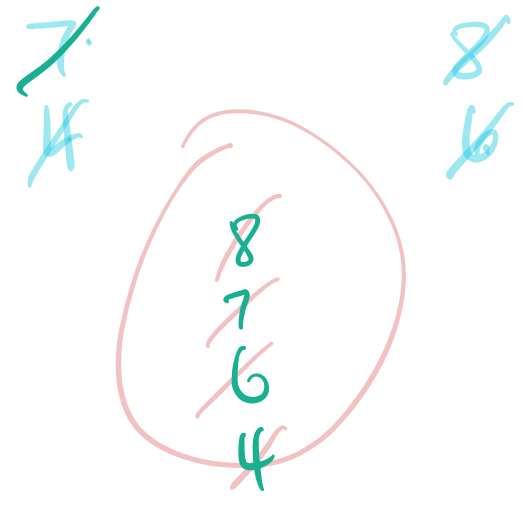
5 3 1 2 | 7 4 8 6

5 3 | 1 2 7 4 8 6

5 3 1 2 7 4 8 6

5 3 1 2 7 4 8 6





1 + 1 + 1 + 1

inv count 1 + 0 + 2 + 2 + = 9 inversions

7 4 8 6 | 5 3 1 2

$\left.\begin{array}{c}\cancel{8}\\7\\6\\\cancel{4}\end{array}\right\}$ $\begin{array}{c}\cancel{5}\\\cancel{3}\\\cancel{2}\\\cancel{1}\end{array}$

4 + 4 + 4 + 0 + 3

Matrix Multiplication

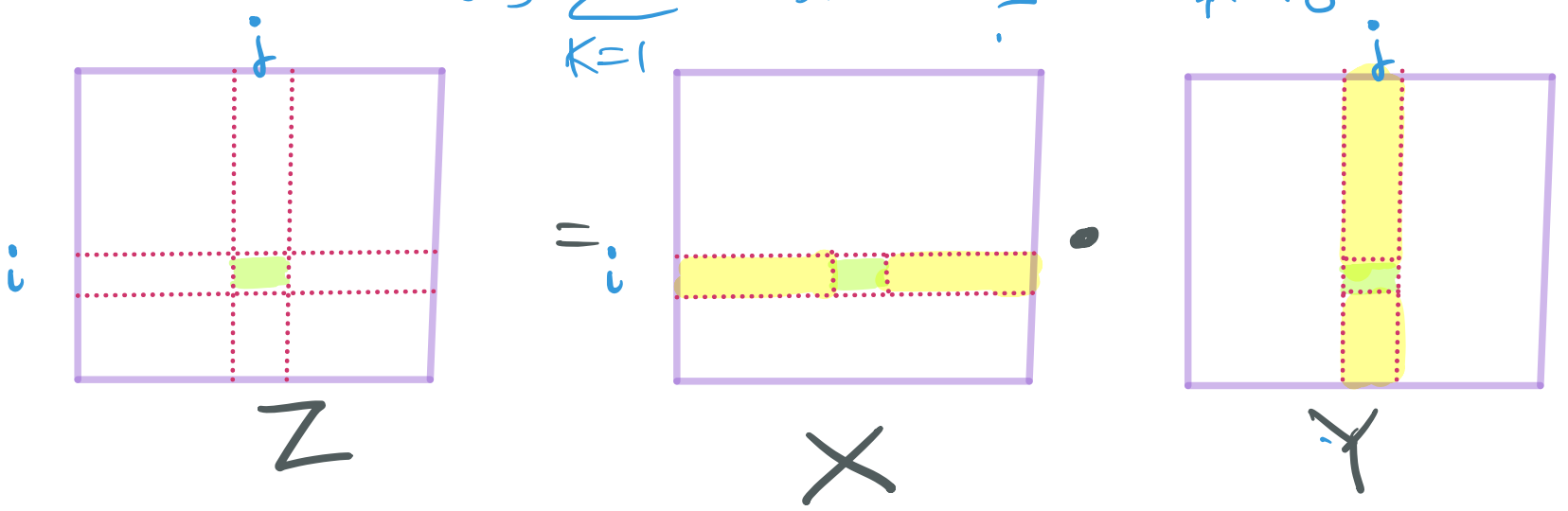
$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \cdot \begin{bmatrix} c_1 & c_2 \\ d_1 & d_2 \end{bmatrix} = \begin{bmatrix} a_1 \cdot c_1 + a_2 \cdot d_1 & a_1 \cdot c_2 + a_2 \cdot d_2 \\ b_1 \cdot c_1 + b_2 \cdot d_1 & b_1 \cdot c_2 + b_2 \cdot d_2 \end{bmatrix}$$

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \cdot \begin{bmatrix} c_1 & c_2 \\ d_1 & d_2 \end{bmatrix} = \begin{bmatrix} a_1 \cdot c_1 + a_2 \cdot d_1 & a_1 \cdot c_2 + a_2 \cdot d_2 \\ b_1 \cdot c_1 + b_2 \cdot d_1 & b_1 \cdot c_2 + b_2 \cdot d_2 \end{bmatrix}$$

Strassen's Matrix Multiplication

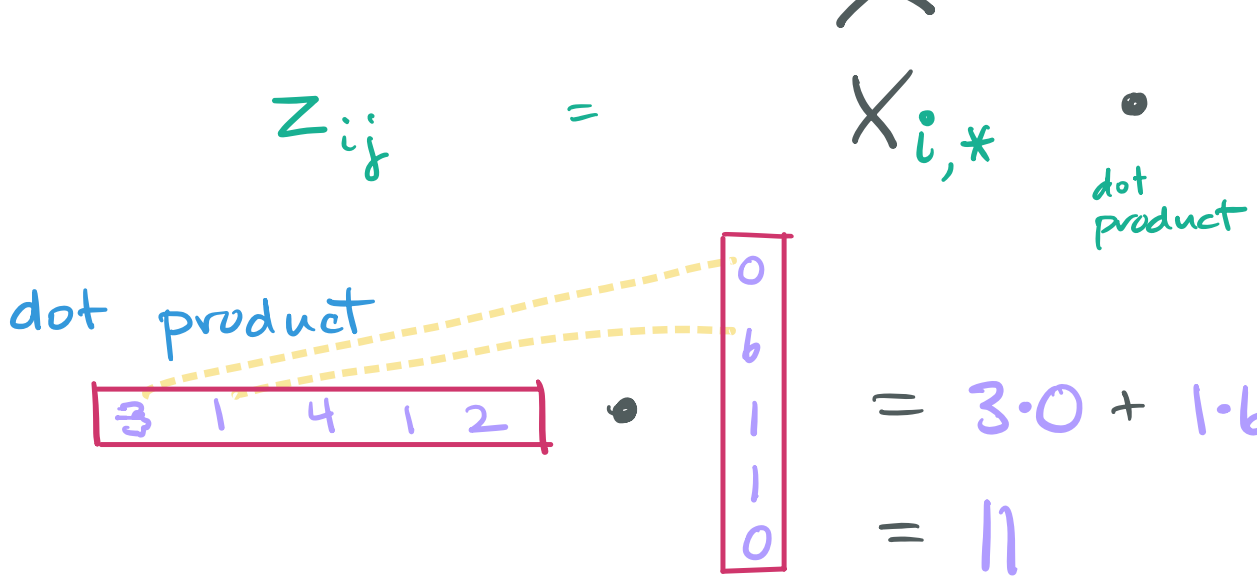
X and Y are n x n matrices (n^2 elements each)

$$Z[i][j] = \sum_{k=1}^{n} X[i][k] \cdot Y[k][j]$$



$$z_{ij} = x_{i,*} \cdot y_{*,j}$$

dot product



What is the time complexity of computing just one entry?

$\Theta(n)$. $\exists$ n^2 entries so this alg is $\Theta(n^3)$.

Illustrating with 2x2 matrices

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

Problem: Matrix Multiplication

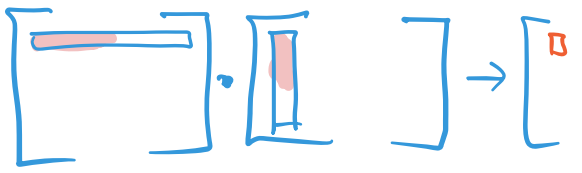
Input: 2 $n \times n$ matrices X and Y

Output: the matrix product $X \cdot Y$

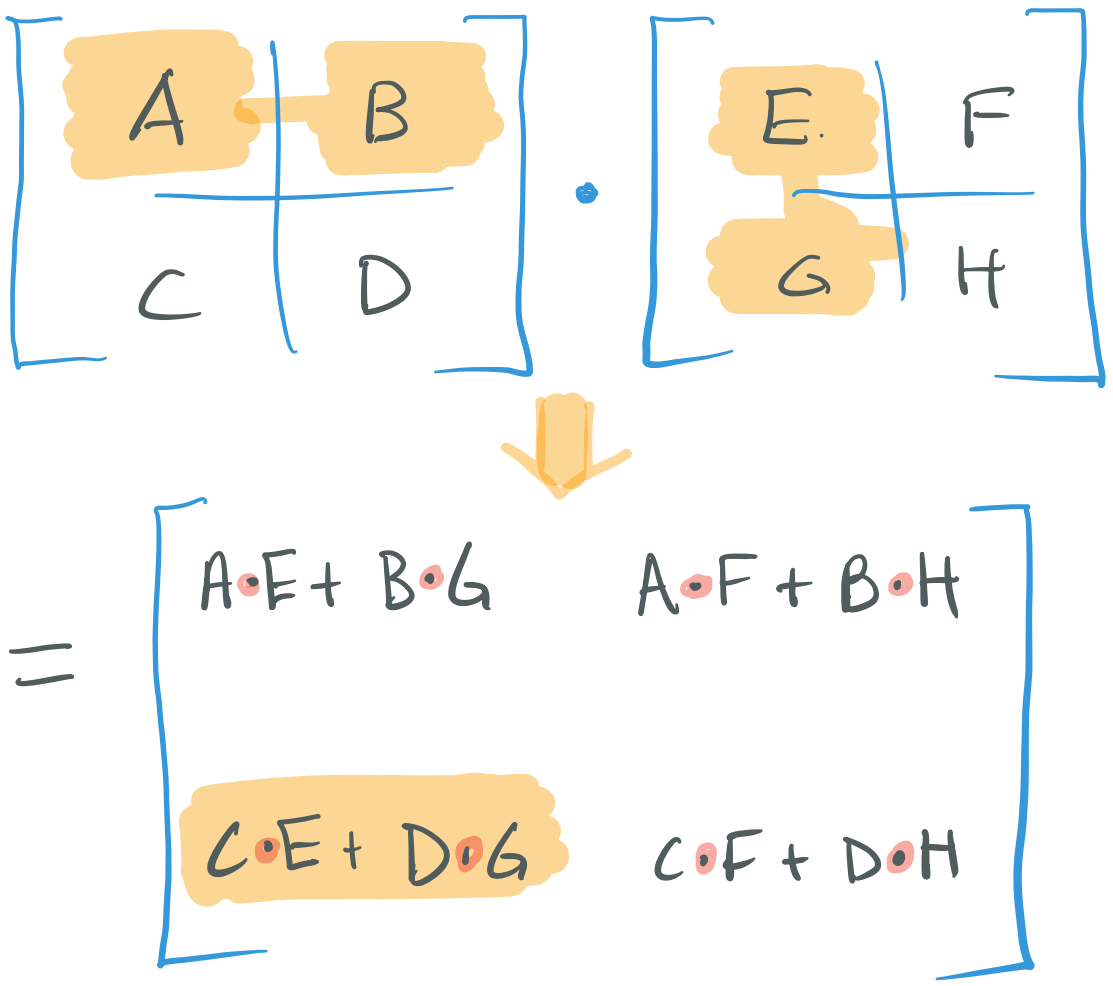
```
StraightforwardMatrixMult(X, Y)
// input: n x n matrices X and Y
// output: n x n matrix Z where Z = X · Y

for i = 1 to n do
    for j = 1 to n do
        Z[i, j] = 0
        for k = 1 to n do
            Z[i, j] += X[i, k]
                     * Y[k, j]
return Z.
```

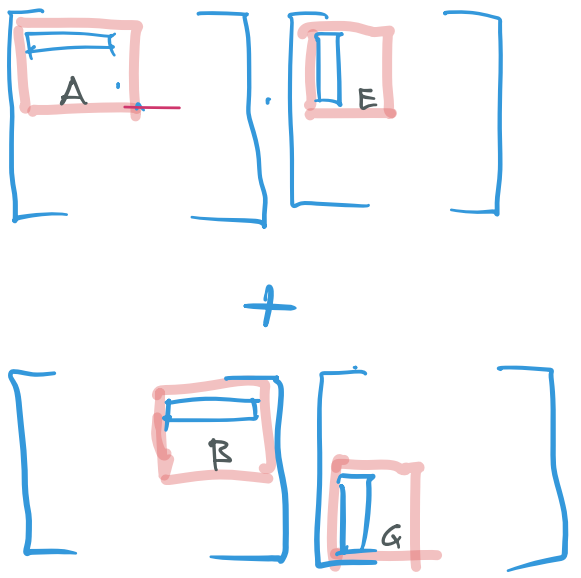
Running time? $\Theta(n^3)$

Strassen's Cool idea....



is given by





First insight : can do it recursively

```
RecMatMult
/* input: n×n matrices X and Y
/* output: Z = X·Y
/* assume: n is a power of 2

    if n=1
        return the 1×1 matrix X[1][1] *Y[1][1]
```

else

A, B, C, D = submatrices of X, as above

E, F, G, H = submatrices of Y, as above

recursively compute the **eight** matrix products as above

perform the necessary additions

return the result

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

Master Thm: $n^2 \in O(n^{\log_2 8 - \varepsilon})$ $\therefore T(n) \in \Theta(n^3)$

($n^{3-\varepsilon}$; n^3)

So why is it interesting?

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH. \end{bmatrix}$$

Strassen's crazy idea:

$\exists$ 8 $\frac{n}{2} \times \frac{n}{2}$ - sized matrix products to compute,

but they are **not independent**...

can we do fewer matrix mults and get all 8

Submatrices?

$$P_1 = A \cdot (F - H)$$

$$P_2 = (A + B) \cdot H$$

$$P_3 = (C + D) \cdot E$$

$$P_4 = D \cdot (G - E)$$

$$P_5 = (A + D) \cdot (E + H)$$

$$P_6 = (B - D) \cdot (G + H)$$

$$P_7 = (A - C) \cdot (E + F)$$

Compute the above. Can we now extract

$AE + BG$, $AF + BH$, $CE + DG$, $CF + DH$?

$$AE + BG = P_5 + P_4 - P_2 + P_6$$

$$= (A+D)\cdot(E+H) + D\cdot(G-E) - (A+B)\cdot H + (B-D)\cdot(G+H)$$

$$= AE + AH + DE + DH + DG - DE - AH - BH + BG + BH - DG - DH$$

$$AF + BH = P_1 + P_2$$

$$CE + DG = P_3 + P_4$$

$$CF + DH = P_1 + P_5 - P_3 - P_7$$

Running time of Strassen's Matrix Mult Algorithm ....

complexity? $\Theta(n^2)$.

... which uses 18 Matrix additions

and 7 matrix mults $\left(\frac{n}{2} \times \frac{n}{2}\right)$

as a function of n, for two $n \times n$ matrices:

$$T(n) = 7\,T\left(\frac{n}{2}\right) + \underline{n^2}$$

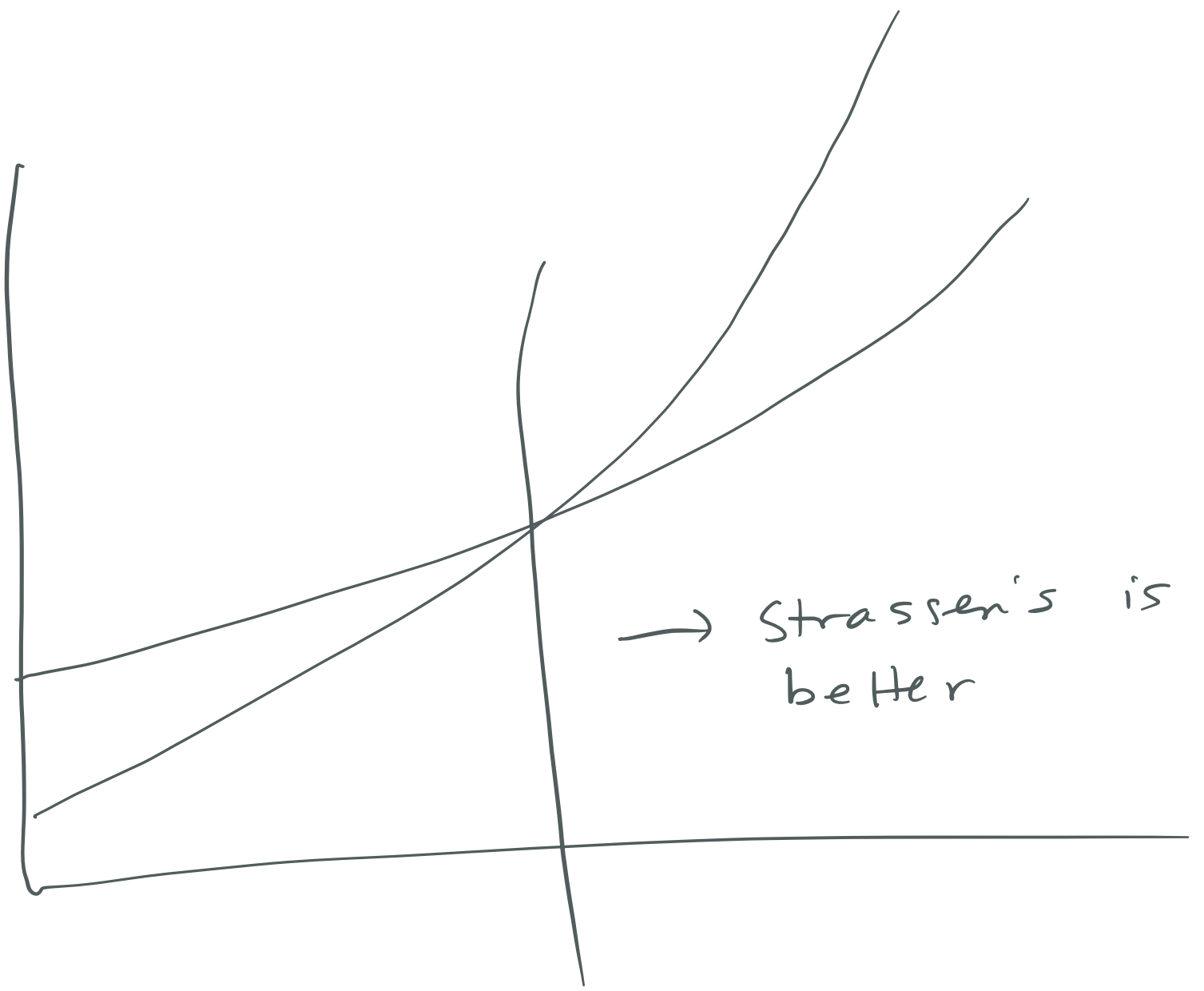
$$n^2 \overset{?}{\in} \_\_ \left(n^{\log_2 7 \overset{+}{?} \varepsilon}\right)$$

ie $n^2 \in \underline{O}\left(n^{2.8074-\varepsilon}\right)$

Hence by case _ of Master Theorem, Strassen's Matrix Mult algorithm runs in

$$\Theta\left(n^{2.8074}\right)$$





$n = ?$