Counting Inversion

Input: An array A of distinct integers

output: The number of inversion of A
The number of pairs (i, j) of array indices with
$\boxed{i < j \text{ and } A[i] > A[j]}$

1	3	5	2	4	6

Naive way of counting

Input: Array A
output: # of inversions

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

Divide & conquer (Basic)

Input :: Array A

Output: # inversion

---

if n = 0 or n = 1 then

    return 0

else

    left Inv := count Inv (first half of A)

    right Inv := count Inv (Second half of A)

    Split Inv := count split Inv (A)

    return left Inv + right Inv + split Inv

left Inv $(i, j \leq \frac{n}{2})$

right Inv $(i, j > \frac{n}{2})$

Split Inv $(i \leq \frac{n}{2} < j)$

1	3	5

2	4	6

Sort_and Count Inv

Input: Array A
Output: Sorted Array B & # Inversions

if $n=0$ or $n=1$ then
return (A, 0)
else
(C, left Inv) = Sort-and-Count Inv (first half of A)
(D, Right Inv) = " " " " (second " " ")
(B, split Inv) = Merge-and-count Split Inv (C, D)
return (B, left Inv. + right Inv + split Inv)

Merge-and-count Split Inv

Input: sorted array C & D
output:- Sorted array B (length n) and
split inversions.

assumption n is even
i:=1, J:=1, SplitInv:=0
for k=1 to n do
if C[i] < D[j] then
B[k] = C[i] i=i+1
else
B[k] = D[j] J = j+1
SplitInv = splitInv + $(\frac{n}{2} - i + 1)$
(# left in C)
return (B, splitInv)

Matrix Multiplication

Let X and Y are $n \times n$ Matrices of integers - n^2 entries in each.

In the product $Z = X \cdot Y$, the entry Z_{ij} in the ith row and jth column of Z is defined as the dot product of the ith row of X and jth column of Y

$$Z_{ij} = \sum_{k=1}^{n} x_{ik} y_{kj}$$

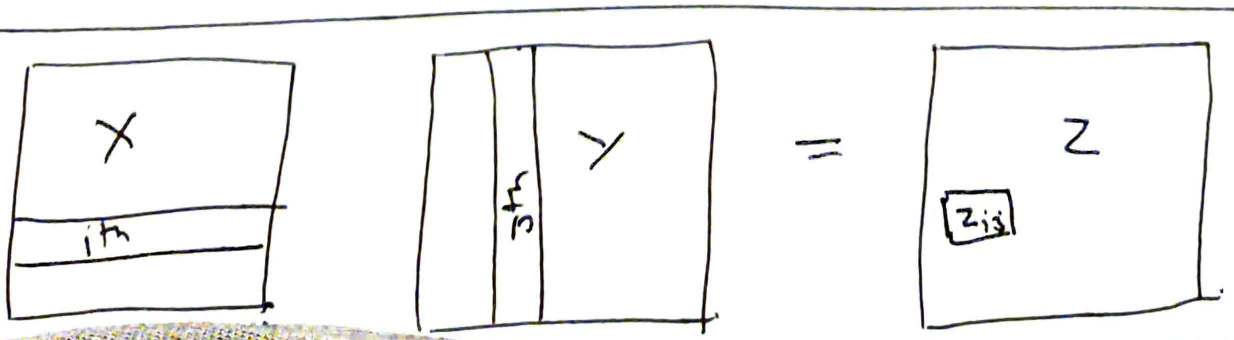
Basic Algo.

Input : $n \times n$ integer matrices X and Y.

Output : $Z = X \cdot Y$

```
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] := Z[i][j] + X[i][k] • Y[k][j]

return Z
```



Divid & Conquer

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \qquad Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

when A, B, C, D, E, F, G, H are all $\frac{n}{2} \times \frac{n}{2}$ matrices.

$$X \cdot Y = \begin{pmatrix} A \cdot E + B \cdot G & A \cdot F + B \cdot H \\ C \cdot E + D \cdot G & C \cdot F + D \cdot H \end{pmatrix}$$

Input $n \times n$ integer matrices X and Y

Output $Z = X \cdot Y$

Assumption n is power of 2

if $n = 1$ then

return 1×1 matrix with entry $X[1][1] \cdot Y[1][1]$

else

A, B, C, D = submatrices of X

E, F, G, H = submatrices of Y

recursively compute the eight matrix product

return the result.

Let

$$Z = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$C_{11} = \underset{1}{A \cdot E} + \underset{2}{B \cdot G}$$

$$C_{12} = \underset{3}{A \cdot F} + \underset{4}{B \cdot H}$$

$$C_{21} = \underset{5}{C \cdot E} + \underset{6}{D \cdot G}$$

$$C_{22} = \underset{7}{C \cdot F} + \underset{8}{D \cdot H}$$

Let $T(n)$ be the time to multiply two $n \times n$ matrices.

In the base case $(n=1)$ it perform on scalar multiplication $T(1) = \Theta(1)$

$T(n)$ = The partitioning time + Recursive calls + time to add the matrices

$$= \Theta(1) + 8T(n/2) + \Theta(n^2)$$

$$= 8T(n/2) + \Theta(n^2)$$

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