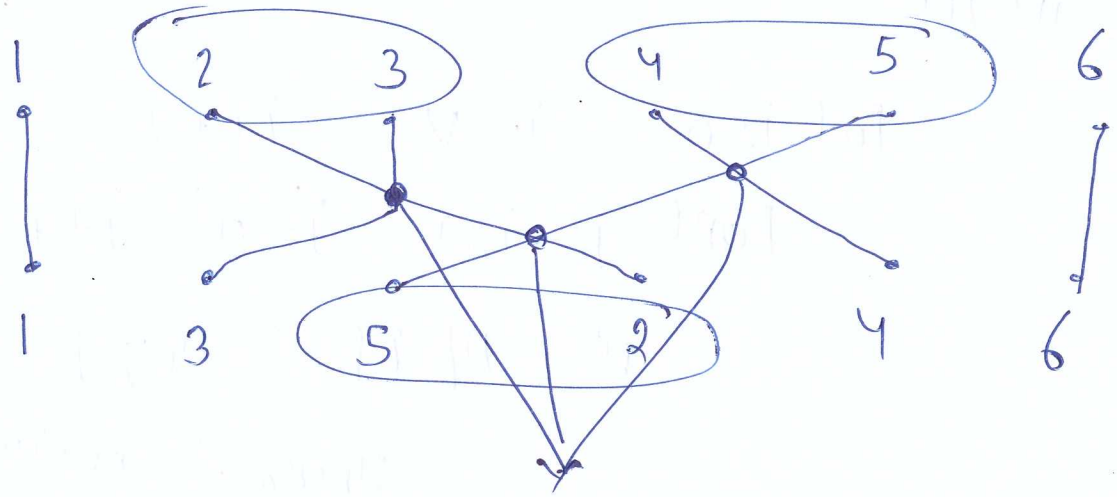
COUNTING Inversion Problem

I/p Array A containing numbers 1, 2, 3, -- n in some arbitrary order

o/p No. of inversions = No of pairs (i, j) of array indices with $i<j$ and $A[i] > A[j]$

Example 1, 3, 5, 2, 4, 6

Inversions = (3, 2) (5, 2) (5, 4)



3 inversions

→ The max. no. of inversions that a n element array can have?

* $\frac{n(n-1)}{2}$

When array is sorted in decreasy order

→ The min. no. of inversion = 0

When array is already sorted in increasy order

---

1) SIMPLE APPROACH

```
For( i=0    i<N    i++)
    FOR( j= i+1   j<N   j++)
        if  A[i] > A[j]
            Increment Inversion Count
```

D&C Approach

Types of inversion

1) Left if $i, j \leq n/2$
2) Right if $i, j > n/2$

] Can compute Recursively

3) Split if $i \leq \frac{n}{2} < j$

] Need seperate sub-routine for this

* In Array 1, 3, 5, 2, 4, 6 All are SPLIT inversion
* No left / Right Inversion

High level Approach

```
CountInv (array A, length n)
    if n = 1   Return 0
    else
        X = CountInv ( Ist half of A, n/2 )
        Y = CountInv ( IInd half of A, n/2 )
        Z = CountSplitInv ( A, n )
        Return X + Y + Z
```

→ Require special care

1	5	4	8	10	2	6	9	3	7

I^st Half = | 1 | 5 | 4 | 8 | 10 |

Inversion = (5-4)

II^nd Half = | 2 | 6 | 9 | 3 | 7 |

Inv = (6-3) (9-3) (9-7)

Split Inv

(5-2) (5-3) (4-2) (4-3) (8-2) (8-6) (8-3)
(8-7) (10-2) (10-9) (10-3) (10-7) (10-6)

Total = 1+3+13 = 17

$\frac{n}{2}+1$ —— n	[1 —— $\frac{n}{2}$]

* Quadratic no. of inversion will occur in this case

* Could Quadratic no. of inversion in Linear Time.

* An array as | 6 7 8 9 10 | 1 2 3 4 5 |

no of inv (all are split) = $\frac{n}{2} \times \frac{n}{2} = \frac{n^2}{4} = O(n^2)$

Idea Have Recursive calls for both count Inversion & Sort

Motivation Merge Subroutine naturally uncovers split inversion

High level Algo

```
Sort_and_CountInv ( array A, length n )
    if n = 1  Return 0
    else
        (B, X) ← Sort_and_CountInv ( 1st half, n/2 )
        (C, Y) ← Sort_and_CountInv ( 2nd half, n/2 )
        (D, Z) ← Sort_and_CountSplitInv ( A, n )
        Return X+Y+Z
```

Now we will pass (B, C, n)

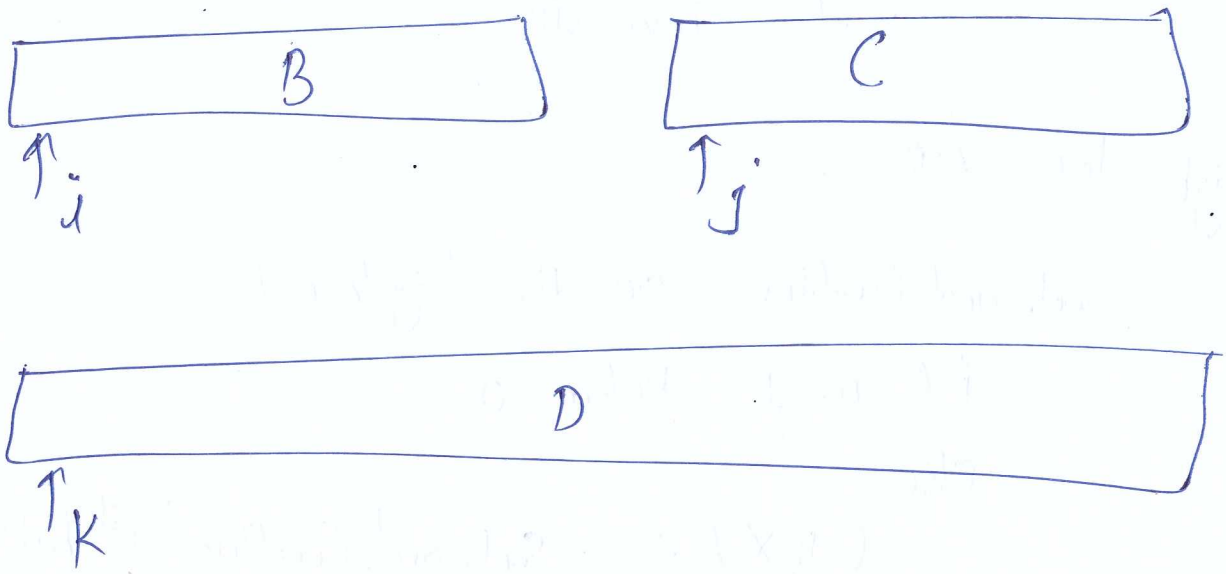
As we need to perform sorting also.

Merge B and C

B = First sorted Array

C = Second sorted Array

D = B+C in sorted



For K = 1 to n

if B[i] < C[j]

D[K] = B[i]

i++

else if C[j] < B[i]

D[K] = C[j]

j++

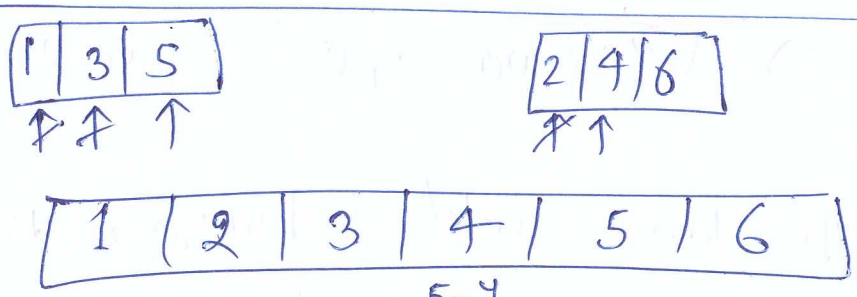
* Let array A has no split inversion then what is the relationship b/w sorted subarrays B and C.

⇒ All elements in B will be less than all elements in C

Proof by Contradiction

Let one elem in B is greater than an elem in C

then there will be split inversion.



5-4

3-2

5-2

Index 2 is copied then 2 is smaller than 3 and following so 3-2 and 5-2

Claim the split inversions involving an element y of 2nd array C are precisely the numbers left in the 1st array B when y is copied.

Proof Let x be an element in 1st array B and y be an element in IInd array C

Case 1 If x is copied to D before y, then $x < y \Rightarrow$ No inversion

Case 2 If y is copied to D before x, then $y < x \Rightarrow$ Split inversion

No of split inversion

See Algo

While merging the two sorted subarrays, keep Running total of number of split inversion

* When element of IInd array C gets copied to D, increment total no of elmt remining in 1st array B.

Run time $T(n) = 2T(n/2) + O(n)$

$\Rightarrow$ Sort $O(n) + O(n)$ inexact.