PW SKILLS

# Merge Sort

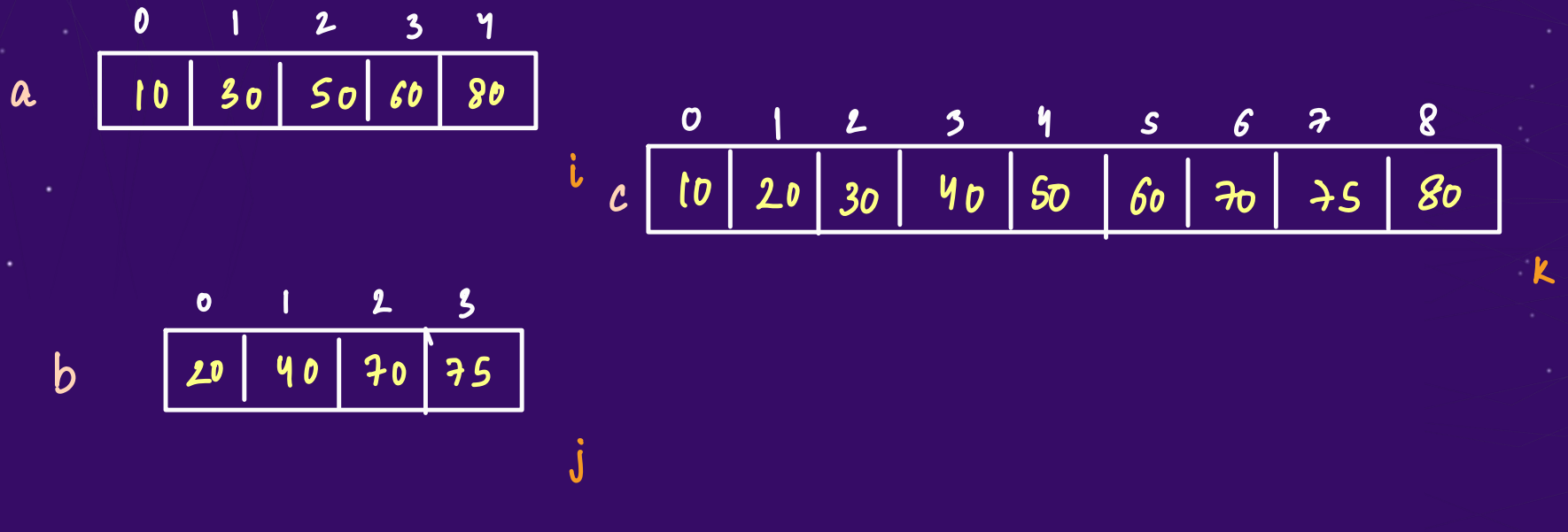
## & Inversion Count

# Today's checklist

PW SKILLS

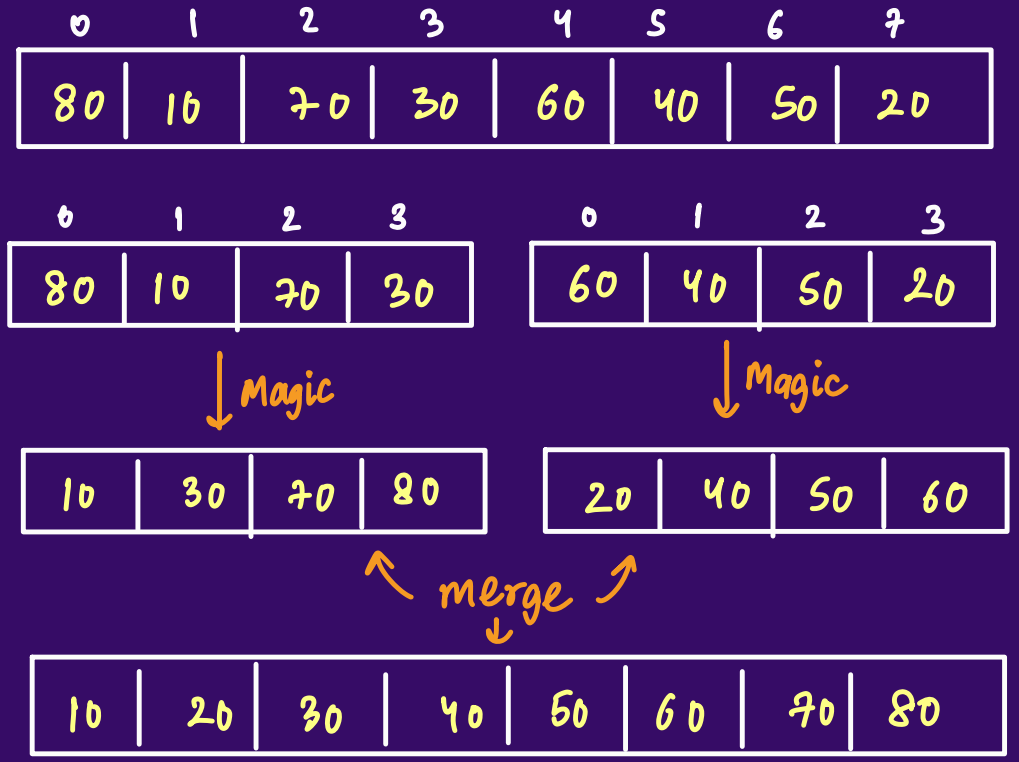
1. Merge 2 Sorted Arrays → Already Done
2. Merge Sort Algorithm
3. Merge Sort Time and Space Complexity
4. Stability
5. Applications of ~~Quick~~ Merge Sort
6. Count Inversion Problem

# Merge 2 Sorted Arrays
## (VV Important)

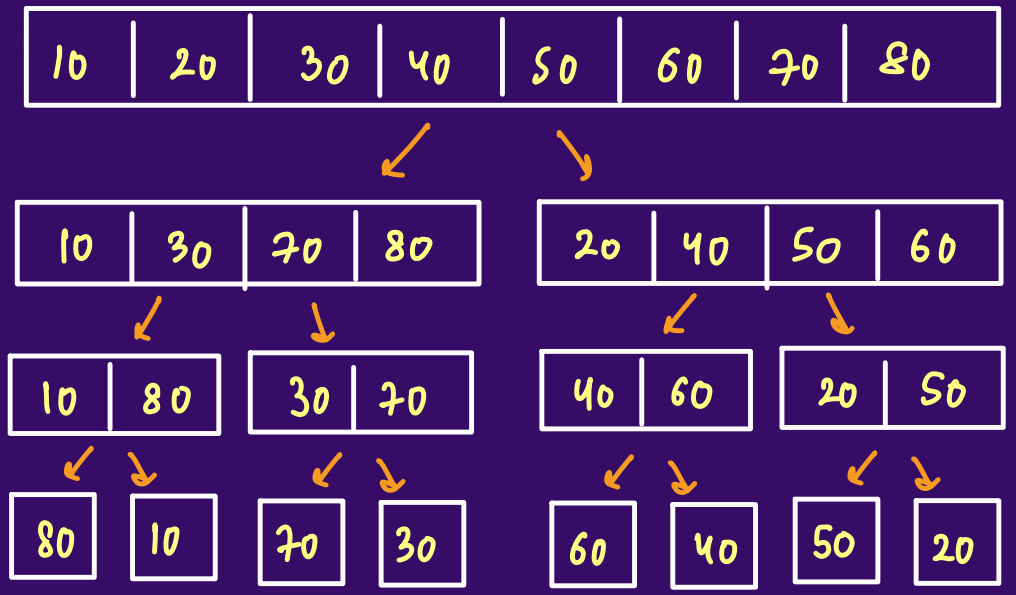



T.C. = O(m+n)

# Merge Sort Algorithm → Using Magic



# Merge Sort Algorithm




Divide & Conquer

$$T.n.o = \text{copy pasting}$$

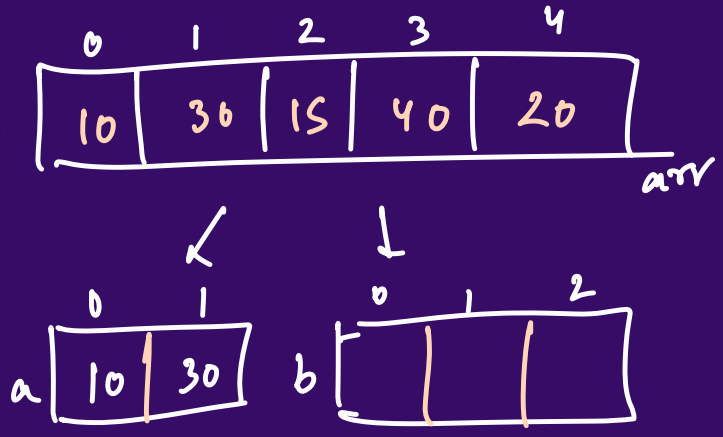
$$\left[ n + \left(\frac{n}{2} + \frac{n}{2}\right) + \left(\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4}\right) \right] + \left[ \left(\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4}\right) + \left(\frac{n}{2} + \frac{n}{2}\right) + n \right]$$

merging

$$\Rightarrow 2[n + n + n]$$

$$\Rightarrow 2(2-1) \cdot n$$

# Merge Sort Algorithm




$$b[i] = arr[i + n/2]$$

hm

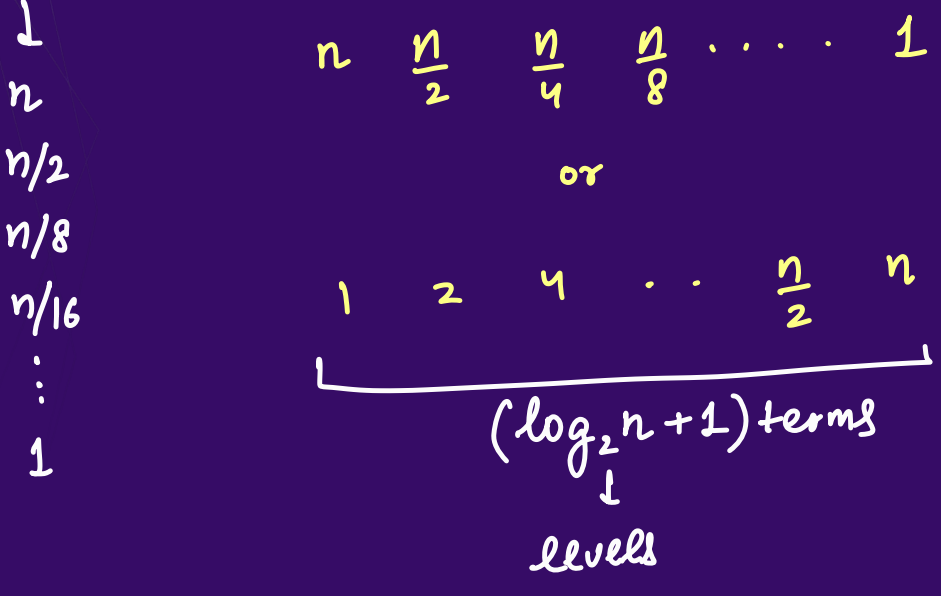
```java
public static void mergesort(int[] arr){
    int n = arr.length;
    if(n==1) return; // base case
    // create two arrays of n/2 size each
    int[] a = new int[n/2];
    int[] b = new int[n-n/2];
    // copy pasting
    for(int i=0;i<n/2;i++){
        a[i] = arr[i];
    }
    for(int i=0;i<n-n/2;i++){
        b[i] = arr[i+n/2];
    }
    // magic
    mergesort(a);
    mergesort(b);
    // merge these 'a' and 'b'
    merge(a,b,arr);
}
```

```java
public static void merge(int[] a, int[] b, int[] c){
    int i = 0, j = 0, k = 0;
    while(i<a.length && j<b.length){
        if(a[i]<=b[j]) c[k++] = a[i++];
        else c[k++] = b[j++];
    }
    while(j<b.length) c[k++] = b[j++];
    while(i<a.length) c[k++] = a[i++];
}
```

→ S.C. = O(nlogn)

# Time and Space complexity

PW SKILLS




Best Case : $O(n \log n)$

Avg Case : $O(n \log n)$

Worst Case : $O(n \log n)$

$$T.n.O = 2 \cdot (l-1) \cdot n = 2 \cdot (\log_2 n + 1 - 1)\, n$$

$$= 2\, n \log n$$

$$T.C. = O(n^* \log n)$$

# Time and Space complexity

PW SKILLS

In this basic code, the amount of space used = $(l-1) \cdot n$

$= n \log_2 n$

$S.C. = O(n \log n)$

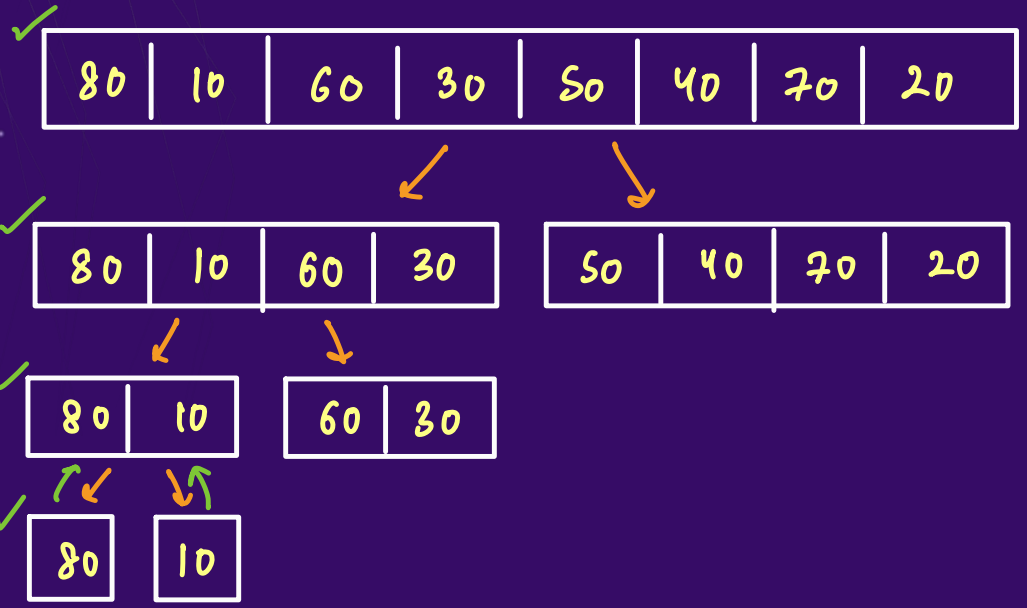
We can improve it by deletion of a & b after merging into arr

Best Case : $O(n)$

Avg Case : $O(n)$

Worst Case : $O(n)$

# Time and Space complexity




Maths

$1 + \frac{1}{2} + \frac{1}{4} \cdots < 2$

$n(1 + \frac{1}{2} + \frac{1}{4} \cdots) < 2 * n$

Extra Space Used : $n + \frac{n}{2} + \frac{n}{4} = n\left(1 + \frac{1}{2} + \frac{1}{4} \cdots \overset{\infty}{\underset{\text{if}}{}}\right) < 2n$

S.C. = O(n)

# Stability of Merge Sort

PW SKILLS

Yes. Merge Sort is stable

arr = 5 1 2 5* 3

↓ sorting

1 2 3 5 5*

Make sure to code like this

```
if (a[i] <= b[j])
    c[k++] = a[i++];
```

if I use '<' instead of '<=' then it won't be stable

# Applications of Merge Sort

PW SKILLS

1. O(nlogn) worst case time complexity
2. Custom Sorting
3. Sorting Linked Lists
4. Inversion Count & Related Problems

# Inversions Count Problem

PW SKILLS

arr = 8 2 5 3 1 4

Q) You have to count the no. of inversions

i < j

arr[i] > arr[j]

(8,2) (2,1) (5,3) (3,1)
(8,5) (5,1)
(8,3) (5,4)
(8,1)
(8,4)

ans = 10

# Inversions Count Problem

Basic Approach (Brute Force)

```
int count = 0;
for(int i = 0; i< n-1 ; i++){
    for(int j= i+1 ; j<n; j++){
        if(arr[i] > arr[j]) count ++;
    }
}
```

T. C. = $O(n^2)$

S.C. = $O(1)$

Definitely not the best

# Inversions Count Problem


a = 2 5 8 b = 1 3 4

i j

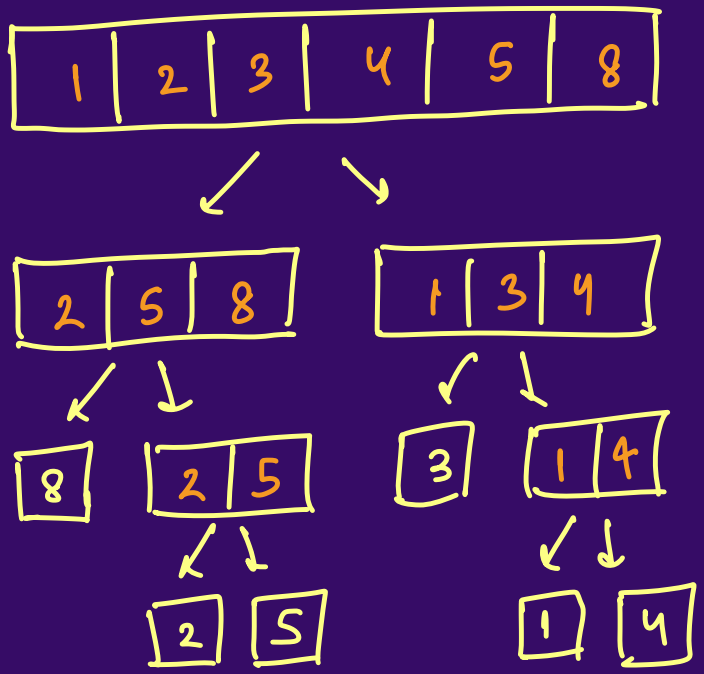
arr[i] > arr[j]

↓ ↳ → ele from b

ele from a

Count = +3 +2 +2
∅ 3 5 7

# Inversions Count Problem




Count

0
1
2
3
6
8
10

# Inversions Count Problem

Create two arrays a & b of n/2 size each & copy paste.

Sort (a)

Sort (b)

Count inversions in a & b

Merge (a, b, arr)

# Inversions Count Problem

PW SKILLS

```java
public static void inversion(int[] a, int[] b){
    int i = 0, j = 0;
    while(i<a.length && j<b.length){
        if(a[i]>b[j]){
            count += (a.length-i);
            j++;
        }
        else i++;
    }
}
```

T.C. = $O(n * \log n)$

```java
public static void mergesort(int[] arr){
    int n = arr.length;
    if(n==1) return; // base case
    // create two arrays of n/2 size each
    int[] a = new int[n/2];
    int[] b = new int[n-n/2];
    // copy pasting
    for(int i=0;i<n/2;i++) a[i] = arr[i];
    for(int i=0;i<n-n/2;i++) b[i] = arr[i+n/2];
    // magic
    mergesort(a);
    mergesort(b);
    inversion(a,b);
    // merge these 'a' and 'b'
    merge(a,b,arr);
    // delete a and b
    a = null; b = null;
}
```

Extra

# Homework :

**Q : Reverse Pairs**

[Leetcode 493]

# THANK YOU