Most Important

Divide & Conquer

lecture-1

Merge Sort

7	3	2	16	24	4	11	9

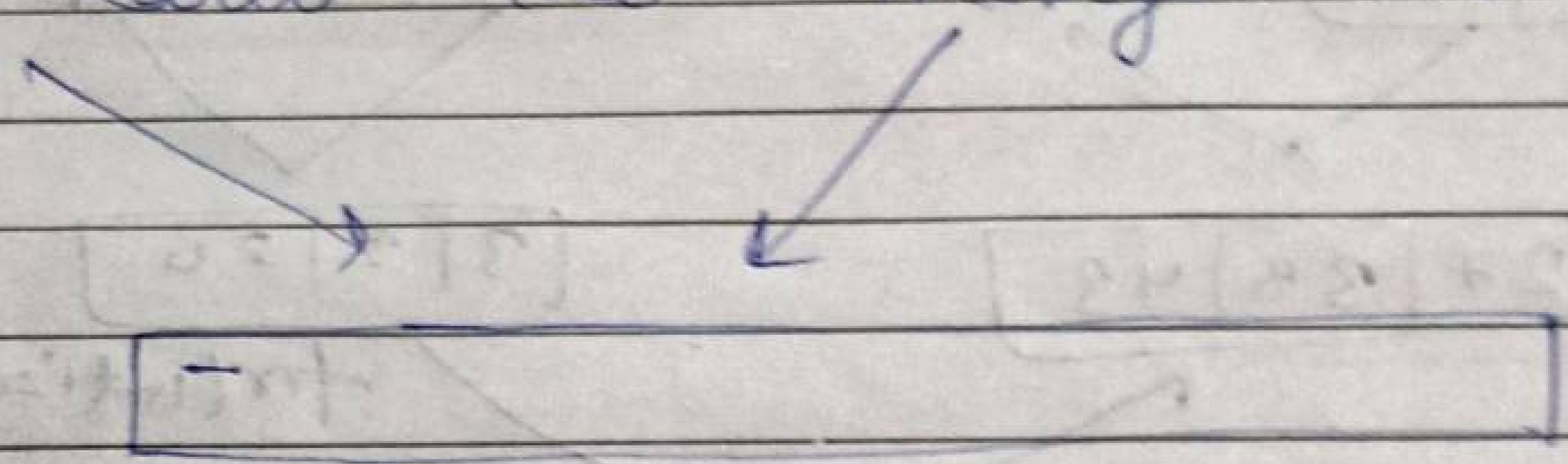
Divide karo and conquer karlo.

mid

7	3	2	16

24	4	11	9

Recursion se solve karlo i.e sort karlo the merge karlo



We don't know merging so.

→ Merge two sorted arrays

ip →

2	4	6

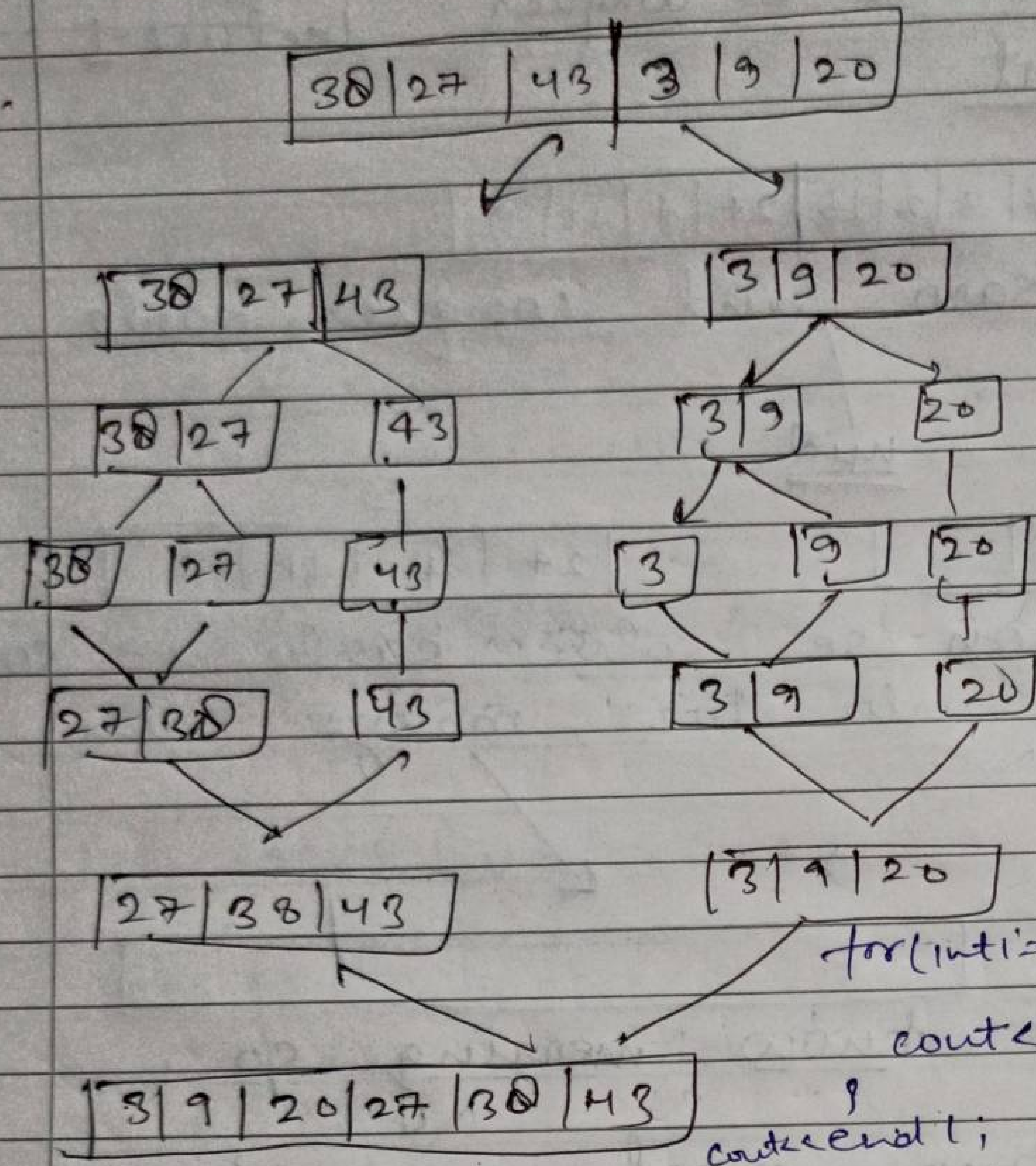
3	5	7	9	11

Use two pointer approach.

→ first loop me comparison hoga

→ If koe array end ho jaaye to jo bach array hai wo ans me aa jaega.

eg:



```
for(int i=0 ; i<n ; i++){
    cout<< arr[i]<<" ";
}
cout<<endl;
```

Code

```
int main(){
int arr[]={4,5,13,2,12};
int n=5;
int s=0;
int e=n-1;
mergesort(arr,s,e);
return 0;
}

void ~~int~~ mergesort(int *arr, int s, int e){
    if(s>=e)
        return;
int mid=(s+e)/2;
    mergesort(arr,s,mid);      // left part
 mergesort(arr,mid+1,e);      // right part
```

invalid

(s>e)

(s==e) → single element

```
merge (arr, s, e);
}

void merge (int * arr, int s, int e)
    int mid = (s+e)/2;
    int len1 = mid-s+1;
    int len2 = e- mid;
    int* ~~arr~~ left = new ~~left~~ int [len1];
    int* right = new int [len2];

//copy values.

 int k=s;
for(int i=0; i< len1; i++){
      left[i]= arr[k];
      k++;
  }

k= mid+1;
 for (int i=0; i<len2; i++){
      right[i] = arr[k];
      k++;
  }

int  leftIndex=0;
int  rightIndex=0;
int  mainArrayIndex=s;

while( leftIndex < len1 && rightIndex < len2){
  if (left[leftIndex]< right[rightIndex]) {
    arr[mainArrayIndex]= left[leftIndex];
    mainArrayIndex ++;
    leftIndex++;
  }
```

copy values

H.W. Inversion Count

```
else {
arr[mainArrayIndex] = right[rightIndex];
mainArrayIndex ++;
rightIndex ++;
}
}

while (leftIndex < len1) {
    arr[mainArrayIndex] = left[leftIndex]
    mainArrayIndex ++;
    leftIndex ++;
}

while (rightIndex < len2) {
   arr[mainArrayIndex] = right[rightIndex]
    mainarrayIndex ++;
    rightIndex ++;
}
```

bacha hua last me daal do bs

STEPS

A → Break karo

B → Recursion se bolo left sort kare

C → Recursion se bolo right sort kare

D → Merge karo (alag se function hoga)

Time Complexity => n(logn)

PAGE No
DATE

~~Quick sort~~

Time complexity of Merge sort

$$T(n) = \underbrace{k_1}_{\text{constant}} + \underbrace{T(n/2)}_{\text{Left call}} + \underbrace{T(n/2)}_{\text{Right call}} + \underbrace{n*k}_{\text{Merge karne}}.$$

$$T(n) = k_1 + 2T(n/2) + n*k$$

$$T(n) = 2T(n/2) + n*k \quad (\times 2)$$

$$T(n/2) = 2T(n/4) + n/2 * k \quad (\times 4.$$

$$T\left(\frac{n}{4}\right) = 2T(n/8) + \frac{n}{4} * k.$$

$|$

$$T(1) = k$$

~~Now 2T(n/2) +~~

$$\left.\begin{aligned} T(n) &= 2\cancel{T(n/2)} + n*k \\ 2\cancel{T(n/2)} &= 4\cancel{T(n/4)} + 2*\frac{n}{2} k \\ 4\cancel{T(n/4)} &= 8\cancel{T(n/8)} + 4*\frac{n}{4} * k \\ | \quad & \quad | \quad\quad | \quad | \end{aligned}\right\}$$

$$\Rightarrow T(n) = (a-1) * k + k$$

$$\Rightarrow T(n) = (\log n - 1) n * k + k$$

PAGE No

DATE

$= k * n * \log n$

→ $\boxed{T(C) = n \log n}$