

Merge Sort – 1

Lecture-31

Raghav Garg

COLLEGE
WALLAH

Already discussed sorting algorithms

Revisiting their time complexity!

Bubble Sort $O(n^2)$

Selection Sort $O(n^2)$

Insertion Sort $O(n^2)$

Merge Sort $O(n \cdot \log n)$

Quick Sort $O(n \cdot \log n)$

What if we had two sorted arrays?

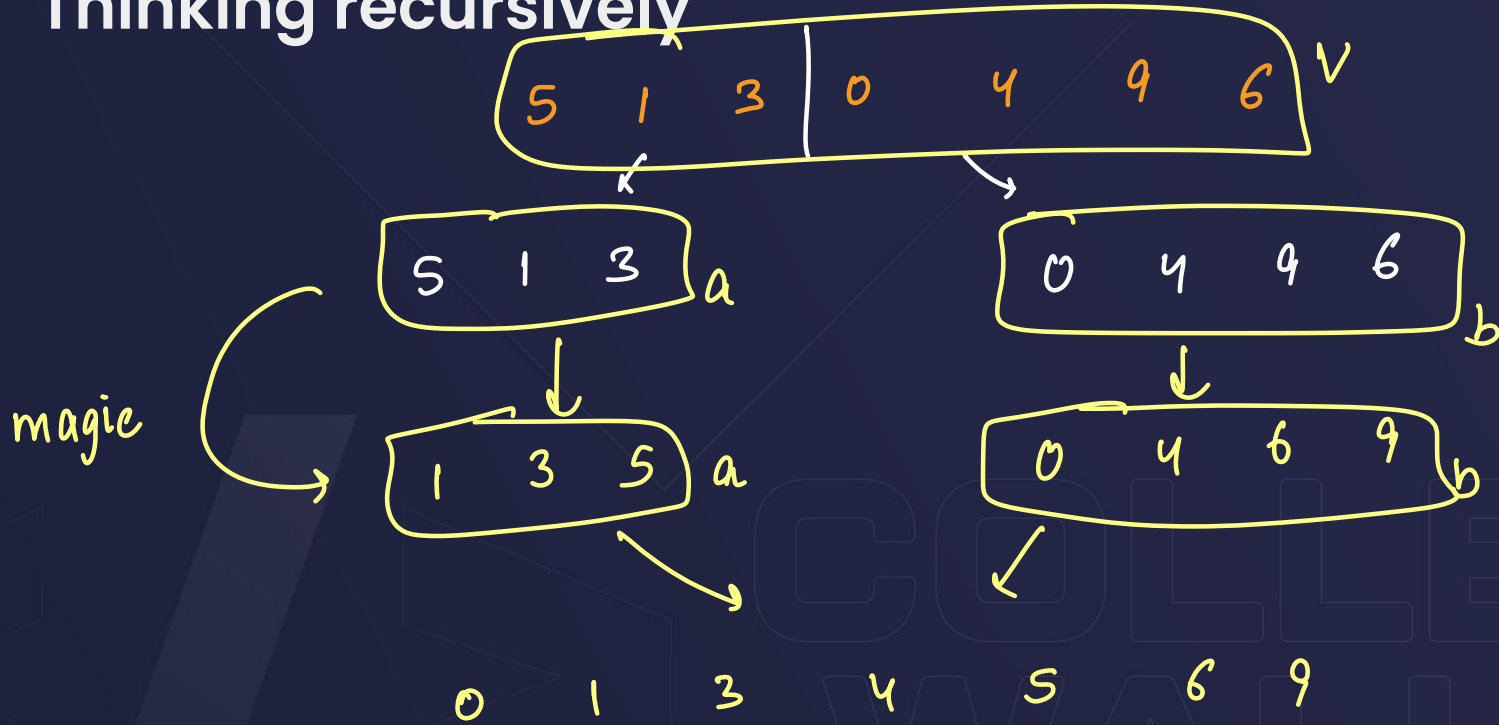
Building the intuition

COLLEGE
WALLAH

Introducing divide and conquer

Thinking recursively

mergeSort(v)



Merge sort algorithm

then apply
magic

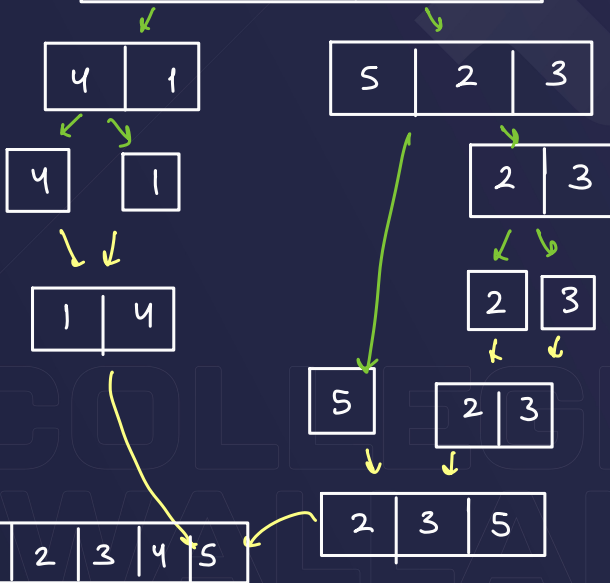
Dividing the array into two equal parts and then
merging them.

COLLEGE
WALLAH

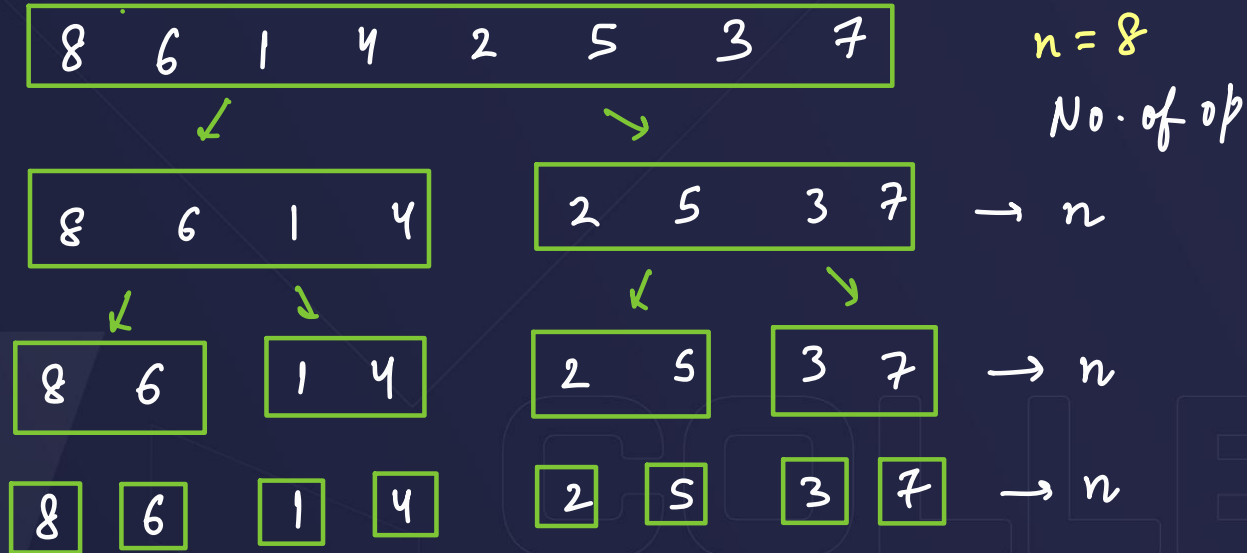
Code for Merge Sort

```
void mergeSort(vector<int>& v){  
    int n = v.size();  
    if(n==1) return;  
    int n1 = n/2, n2 = n - n/2;  
    vector<int> a(n1), b(n2);  
    // copy pasting  
    for(int i=0;i<n1;i++)  
        a[i] = v[i];  
    for(int i=0;i<n2;i++)  
        b[i] = v[i+n1];  
    // magic aka recursion  
    mergeSort(a);  
    mergeSort(b);  
    // merge  
    merge(a,b,v);  
}
```

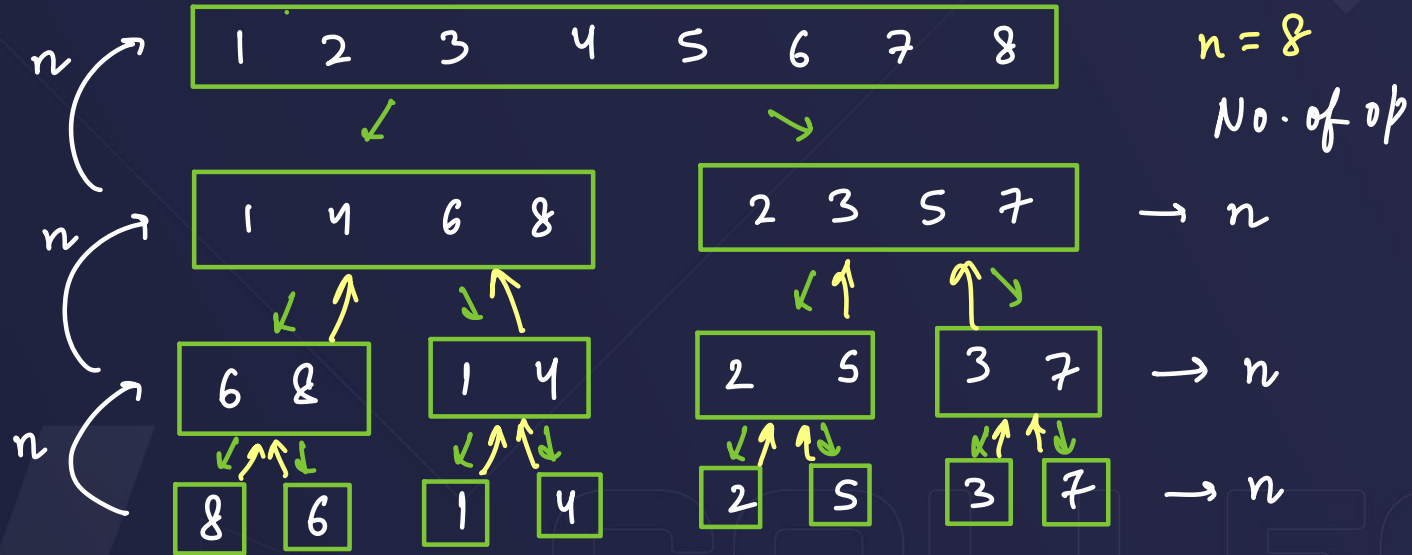
v =



Does it improve the time complexity?



Does it improve the time complexity?



Total no. of operations = $Kn + Kn = \times (2n)$

T.C. = $O(\times n) \rightarrow O(n \cdot \log n)$

Does it improve the time complexity?

$$n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \dots$$

$$\rightarrow n, \frac{n}{2^1}, \frac{n}{2^2}, \frac{n}{2^3}, \dots, \frac{n}{2^x}$$

x terms

$$\Rightarrow \frac{n}{2^x} = 1 \Rightarrow n = 2^x$$

$$\Rightarrow 2^x = n$$

$$\Rightarrow x = \log_2 n$$

$$n = 128$$

$$\downarrow$$

$$B.S. = \frac{n(n-1)}{2} = 64 \cdot 127$$

$$= 8128 \text{ ops}$$

$$\boxed{M.S.} \rightarrow 2n \log_2 n$$

$$\Rightarrow 2 \times 128 \times \log_2 128$$

$$= 256 \times 7$$

$$= 1792 \text{ operations}$$

Does it improve the time complexity?

→ Yes!!

$$n = 1024$$

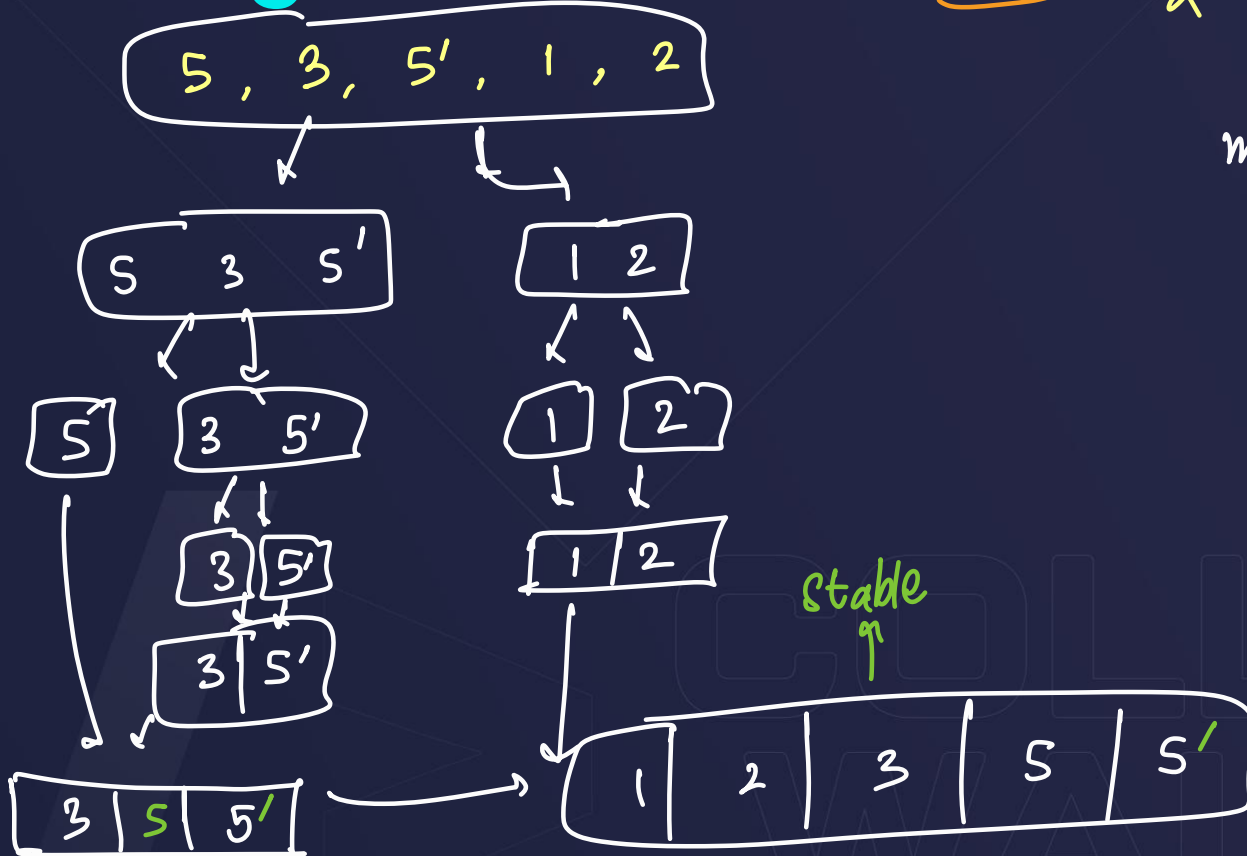
$$B.C. = \frac{n(n-1)}{2} = \frac{1024 \cdot 1023}{2} = 5,23,776 \text{ ops}$$

$$\begin{aligned} m.s. &= 2 \times n \times \log n = 2 \times 1024 \times \log_2 1024 \\ &= 2048 \times 10 \\ &= 20480 \text{ ops} \end{aligned}$$

Is merge sort stable? yes

2-3 din

↓
merge sort code
↓
video dekh



Applications of Merge Sort

- 1) It is used in sorting linked list
- 2) It is used in count inversion problem
- 3) External sorting

Drawbacks of Merge Sort

↳ Time Complexity $\rightarrow O(n \log n)$

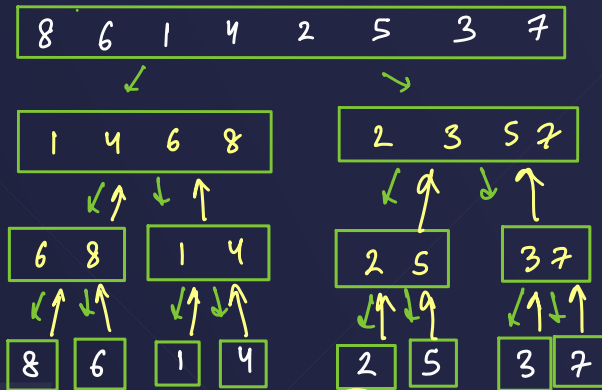
Space Complexity $\rightarrow O(n \log n) \rightarrow \text{improved}$

\downarrow
 $O(n)$

Drawbacks of Merge Sort



Extra Space Used: $n + \frac{n}{2} + \frac{n}{4} \rightarrow \boxed{2n} \rightarrow \boxed{O(n)}$



Extra Space Used :

$$\begin{aligned}
 & \left(n \right) + \left(\frac{n}{2} \right) + \left(\frac{n}{4} + \frac{n}{4} \right) + \left(\frac{n}{2} \right) + \left(\frac{n}{4} + \frac{n}{4} \right) \rightarrow 3n
 \end{aligned}$$

The diagram shows the recursive splitting of an array into sorted sub-arrays. The array [8, 6, 1, 4, 2, 5, 3, 7] is split into [1, 4, 6, 8] and [2, 3, 5, 7]. Each of these is further split into pairs: [6, 8] and [1, 4] from the first half, and [2, 5] and [3, 7] from the second half. These pairs are then sorted individually, resulting in [8, 6], [1, 4], [2, 5], and [3, 7]. The diagram also includes a handwritten formula for the extra space used, which is the sum of the sizes of all recursive calls, resulting in $3n$.

Practice

→ Homework

'Implement' merge sort algorithm to sort an array of elements in **decreasing** order.

COLLEGE
WALLAH

Count Inversions

Two elements of an array a , $a[i]$ and $a[j]$ form an **inversion** if $a[i] > a[j]$ and $i < j$. Given an array of integers. Find the Inversion Count in the array.

	0	1	2	3	4
$a \rightarrow$	5	1	8	2	3

$$a[0] > a[1]$$

$$0 < 1$$

$(5, 1)$

$(5, 2)$

$(5, 3)$

$(8, 2)$

$(8, 3)$

Total no. of
inversions = 5

M-I: Brute Force :

$$\text{total no. of ops} = \frac{n(n+1)}{2}$$

	0	1	2	3	4
a →	5	1	8	2	3

```

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

```

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

$$S.C. = O(1)$$

What if...?

We have an array made up of two subarrays, both sorted.

What can be said about the **inversions** including a certain element?

$\{ 3 \quad 5 \quad 6 \quad 7 \}$
 i

$\{ 2, 2, 3, 4 \}$
 j

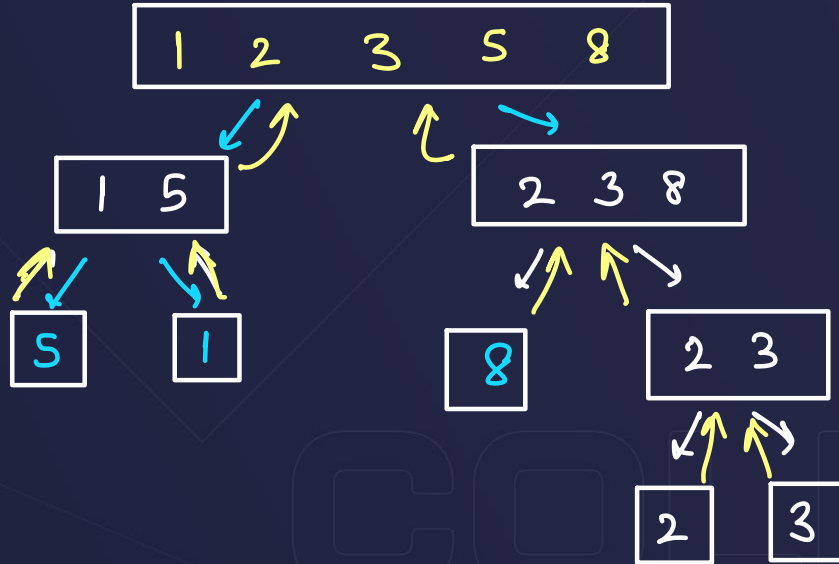
Count = 0 4 8 11 14 **inversions**

Recalling Merge Sort with this respect

The efficient solution

Algorithm

Count = \emptyset 1 2 3 4 5



Coding and dry run

Karli

COLLEGE
WALLAH

Time complexity



Merge Sort



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

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

Next Lecture

Next sorting algorithm: **Quick Sort** & *Quick Select*

COLLEGE
WALLAH