Sorting

Friday, October 10, 2025 3:20 PM

Sorting:

1) Bubble sort

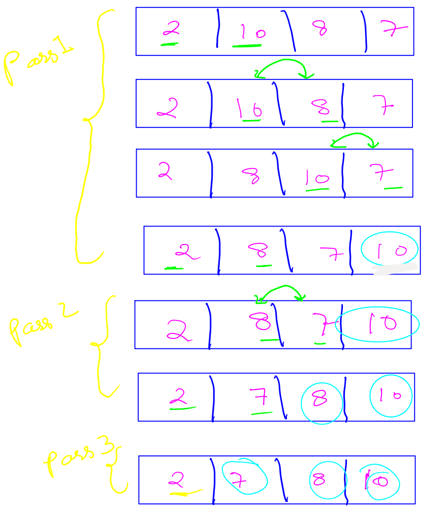
→ |Array| ←

Arrays → Merge

⇄

pass ✓ 1 ✓

→ Comparison based sorting



pass 1

| 2 | 10 | 8 | 7 |

| 2 | 10 | 8 | 7 |

| 2 | 8 | 10 | 7 |

| 2 | 8 | 7 | 10 |

pass 2

| 2 | 8 | 7 | 10 |

| 2 | 7 | 8 | 10 |

pass 3

| 2 | 7 | 8 | 10 |

→ sorted

n = 4

Pass → 1 element sort

n = 4 pass = 3

n − 1 = pass ✓

Brute force :-

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

Approach 2: optimized: sorted:

Drawback: Time complexity / comparison

| 1 | 2 | 3 | 4 | sorted

BubbleSort1 | Sort2 :- $O(n^2)$

swap = ~~[illegible]~~ false

1 2 3 4

pass 1

if(arr[i] > arr[i+1])

sw

swap = true

Co

Counting Sort:-

1	4	1	3	2	4	3	7

max = ✓

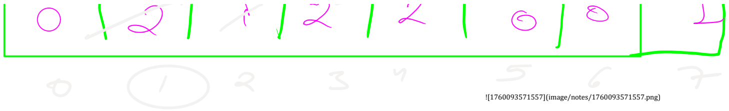
arr/freq[] = max+1

← Integer ~~[illegible]~~ Positive integer ✓

0	~~0~~ ~~1~~ 2	~~0~~ 1	~~0~~ ~~1~~ 2	~~0~~ 2 ~~1~~	0	0	~~0~~ 1
0	1	2	3	4	5	6	7

0	2	1	2	2	0	0	1

![1760093571557](image/notes/1760093571557.png)


0
2
1
2
2
6
8
4
0
1
2
3
4
5
6
7

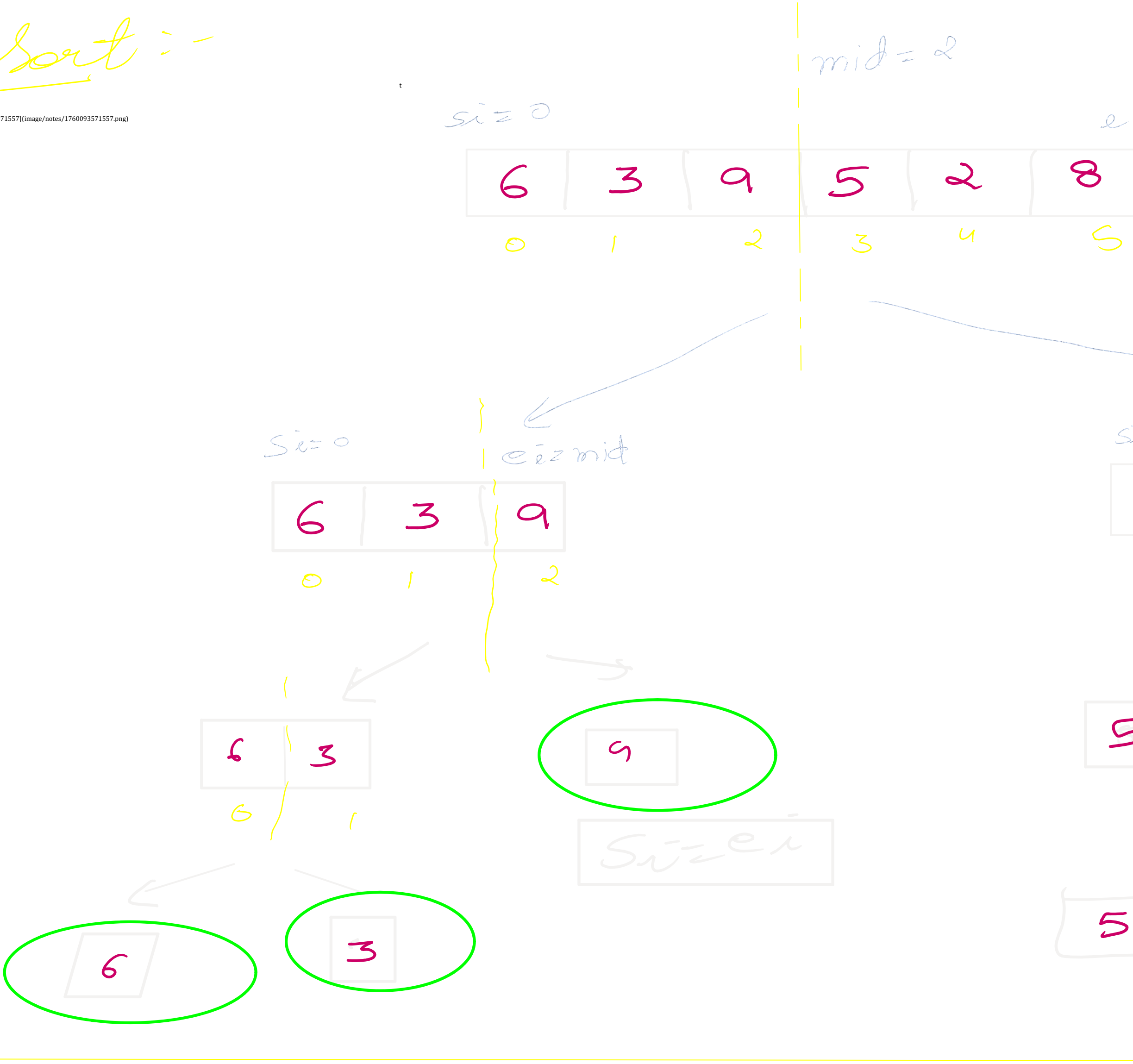
![1760093571557](image/notes/1760093571557.png)

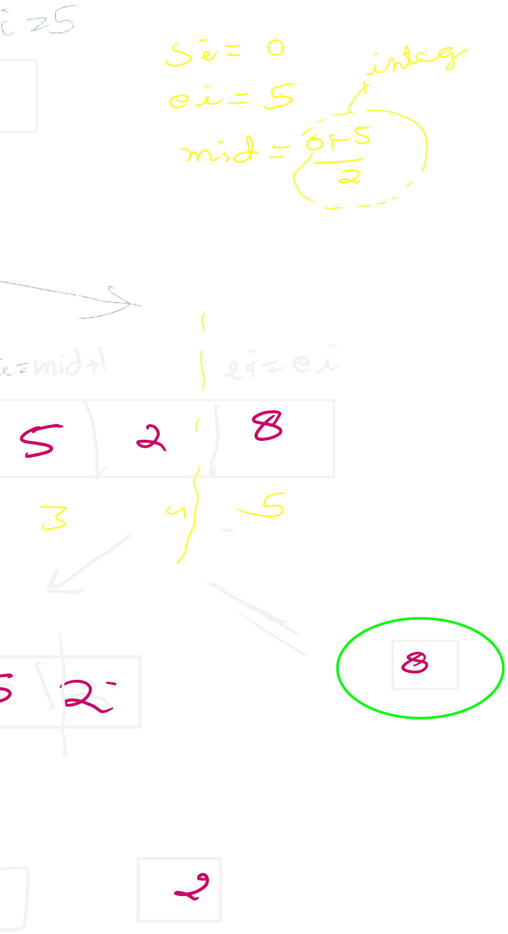
1
1
2
3
3
4
4
7

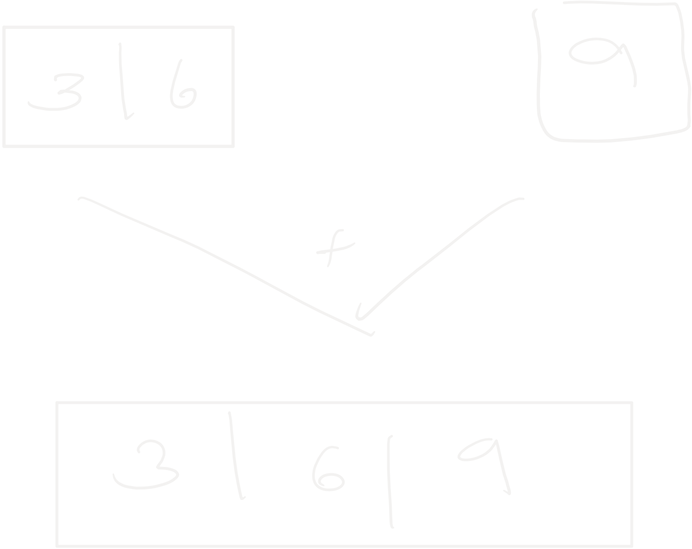
Merge Sort :-

![1760093571557](image/notes/1760093571557.png)

t


mid = 2
si = 0
e
6
3
9
5
2
8
0
1
2
3
4
5
si = 0
ei = mid
6
3
9
0
1
2
6
3
9
0
1
si = ei
6
3
5
5


i=5
si = 0
ei = 5
mid = (0+5)/2
integ
e=mid+1
ei=ei
5
2
8
3
4
5
5
2
8
2

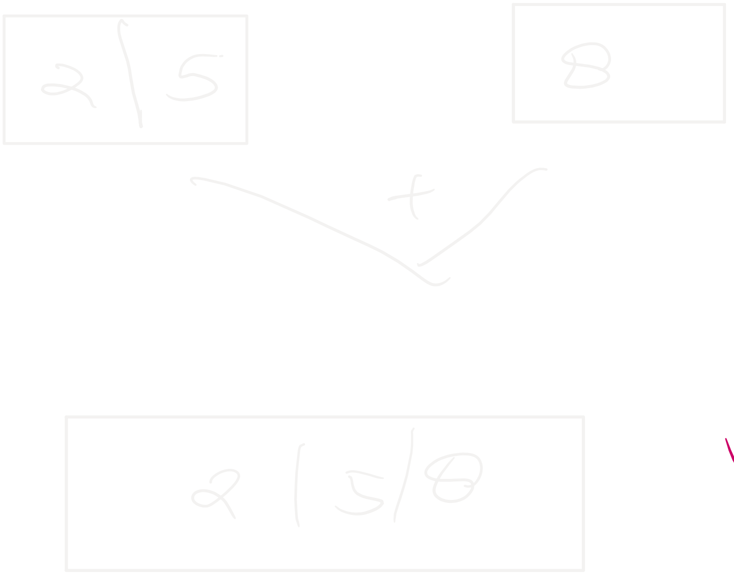


main() {

arr ✓

si = 0

ei = n-1



Addition
+
Conquer

~~mid = (si + ei)/2~~ ✓

MergeSort (arr , si

}

MergeSort (arr , ~~si~~

//Base Case

if(si == ei) return

mid ✓

MergeSort (arr,

Merge sort (arr,

Merge (arr, si, m

, ei)

, ei)

si, mid) //left ✓

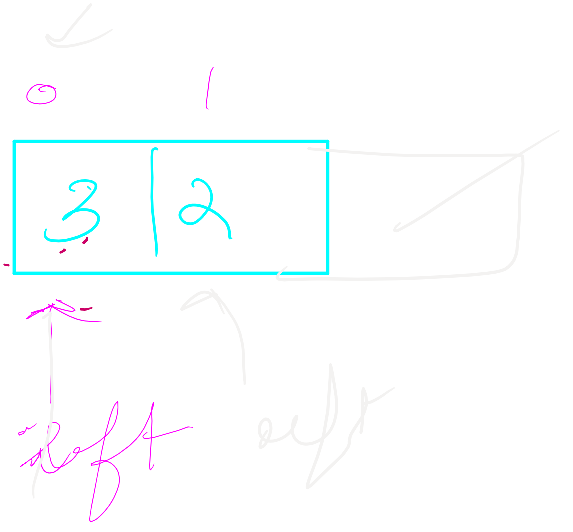
mid+1, ei) //righ ✓

mid, ei);

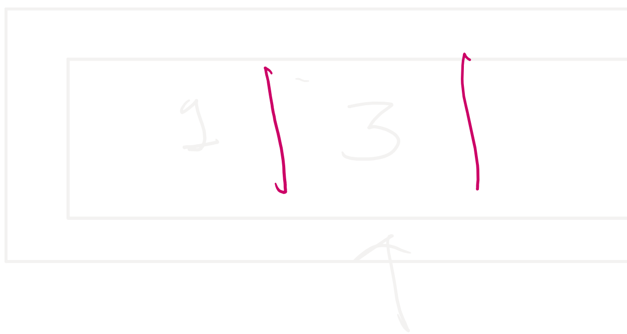
Merge (arr, si, m

Merge(arr, si, mi

temp() = 0 e

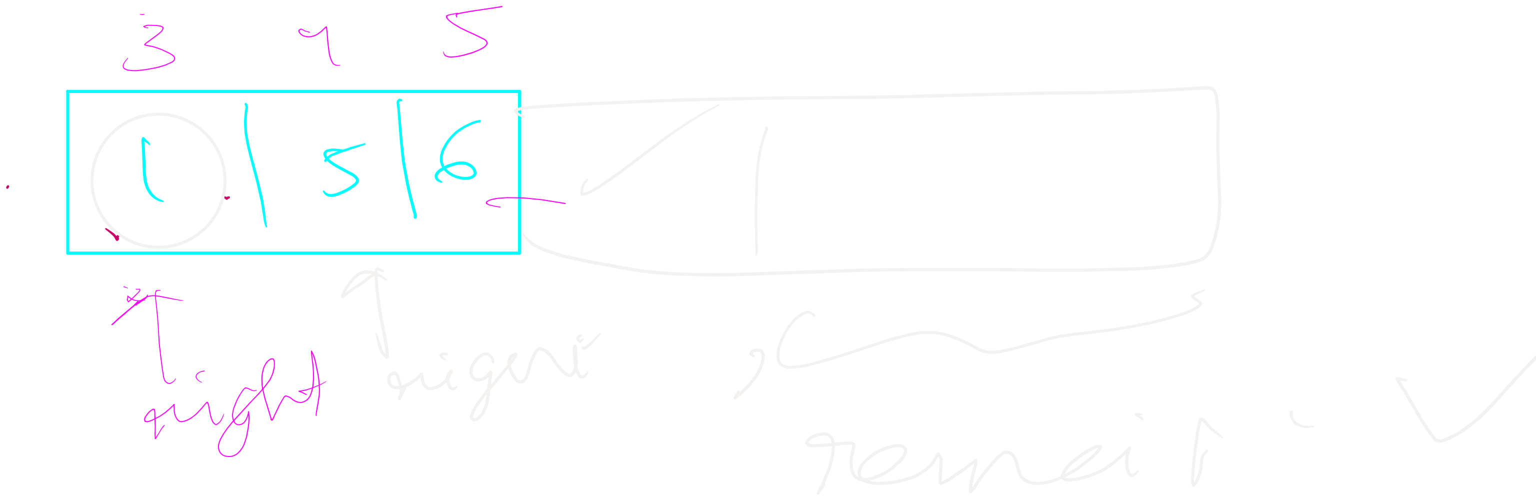


index



d, ci) {   // Important ✓

= Si + 1


3
4
5
1
5
6
right
right
remain