① Bubble Sort :

↳ Large Elements Come to End of Array by Swapping with Adjacent Elements. - For increasing order.

← Largest Elements will come into their position

arr = 5, 4, 1, 3, 2.

i=0

↳ 4 5 1 3 2 J=0
4 1 5 3 2 J=1
4 1 3 5 2 J=2
4 1 3 2 5 J=3

i=1

↳ 1 4 3 2 5 | 0
1 3 4 2 5 | 1
1 3 2 4 5 | 2

i=2

↳ 1 3 2 4 5 | 0
1 2 3 4 5 | 1

i=3

1 2 3 4 5 | 0

i → 0 : n-1

J → 0 : n - i - 1

check if curr elem is greater than next elem if yes swap.

---

③ Selection Sort :

↳ On every iteration, we will find the minimum element and swap it with ith index

arr = [5, 4, 3, 1, 2]

minimum = ∞

Algo:

```
for i → 0 : n-1 :
    min = i
    for j → i : n :
        if arr[j] < arr[min] :
            min = j

    Swap ( arr, i, min).
```

③ insertion Sort:

↳ pick an Element from unsorted part and place it in the right position in the sorted Part.

arr = [13, 46, 24, 52, 20, 9].

→ 13, 46, 24, 52, 20, 9 ⇒ 13, 46, 24, 52, 20, 9.

→ 13, 46, 24, 52, 20, 9 ⇒ 13, 46, 24, 52, 20, 9.

→ 13, 46, 24, 52, 20, 9 ⇒ 13, 24, 46, 52, 20, 9 ⇒ 13, 20, 24, 46, 52

→ 13, 24, 46, 52, 20, 9 ⇒ 13, 24, 46, 52, 20, 9

→ 13, 24, 46, 52, 20, 9 ⇒ 13, 24, 46, 20, 52 ⇒ 13, 24, 20, 46, 52

→ 13, 20, 24, 46, 52, 9 ⇒ 13, 20, 24, 46, 9, 52 ⇒ 13, 20, 20, 46, 52

9, 46

9 24

9 20

9 13.

Algo:

① for i → 1 → n :

j = i

while j > 0 { arr[j] < arr[j-1]

swap

j--;

Counting Sort:

it is used when the numbers are small, or if we know the range of numbers

ex: ↳ 0-100 (marks)

↳ only for +ve num.

arr = [1, 4, 1, 3, 2, 4, 3, 7]

count = [0 2 2 2 0 0 1]

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

⇒ 0 2 2 2 0 0 1

(update)

arr = [1, 1, 2, 3, 3, 4, 7]

for i = 0 to n.

freq

for i=0 to max num :

update.

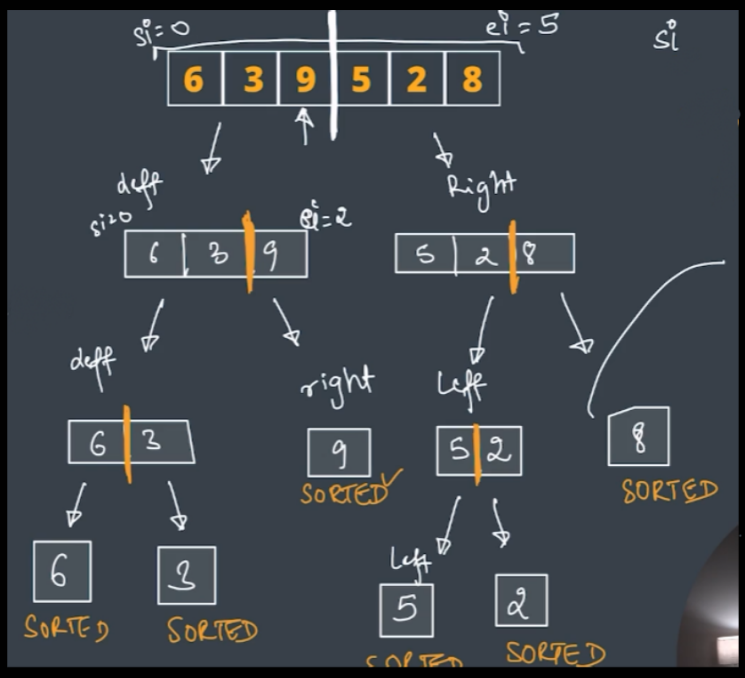
O (max (n, maxRank) , maxRange).

Divide & Conquer → Use those small problems and conquer huge problem..

↳ divide huge problem into small sub problems.

→ Merge Sort :

6, 3, 9, 5, 2, 8 ⇒ 2 3 5 6 8 9.



Approach :

① Divide Array using mid

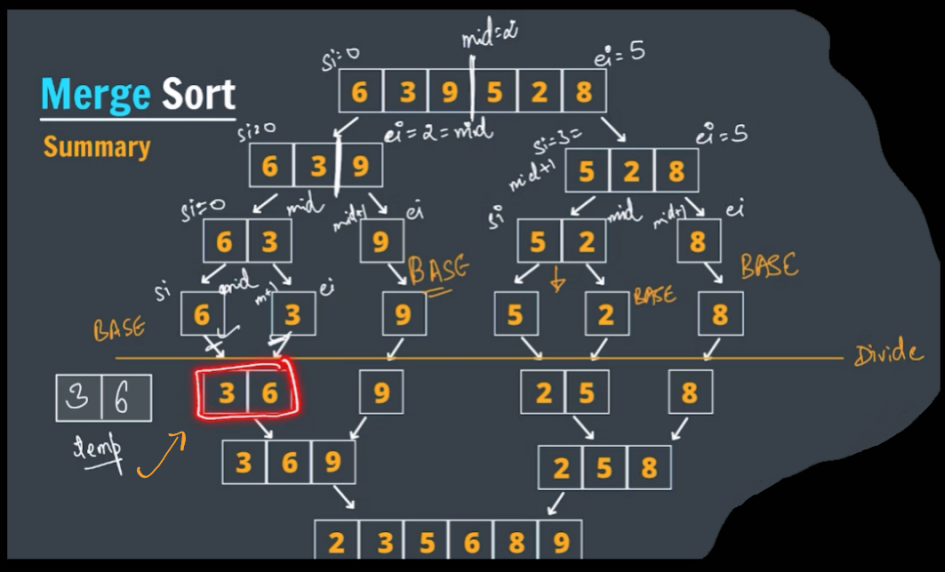
② merge Sort (Left)
merge Sort (Right)

③ merge

!!

369
258
⇒ 2 3 5 6 8 9 temp arr

↳ size = Left + Right elements



(nlogn , n)

Quick Sort

↳ avg cases => nlogn

Worst case => n^2

↳ When Pivot is Smallest / Largest of all elements.

↳ sorted array.

↳ in ascending / descending order.

=> (nlogn , 1)

Pivot & Partition

(1) It can be choosen in many ways } Picking a pivot

- First / Last
- Mid
- Random.

(2) based on condition. (partition).

____ < arr[pivot] < ____

1	2	3	4	5
6	3	9	8	2	5

=>

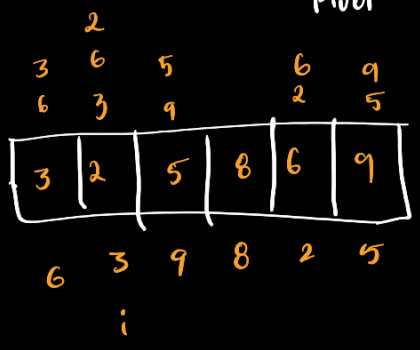
3,2 < 5 < 6,9,8

(3) Quick Sort (Left) Base Case: Single Element

Quick Sort (Right)

How partition happens:

pivot



6 3 9 8 2 5

i

j = -1

for i = 0 : n - 1

if arr[i] < arr[pivot]:

j += 1

swap(arr, i, j)

j = -1, 0, 1

last step:

j += 1

swap(arr, i, j)

return j

↳ using this we need to split array