Selection Sort

→ sort the array by repeatedly finding the min element (ascending order) from the unsorted part and putting it at the beginning.

sorted | unsorted

arr → [10 25 12 20 15]

Find min [0, 4] → 11

[11 25 12 20 60] 1st iteration

sorted | unsorted

Find min [1, 4] → 15

[11 12 25 20 60] 2nd iteration

sorted | unsorted

Find min [2, 4] → 20

[11 12 20 25 60] 3rd iteration

sorted | unsorted

min → [3, 4] → 25

[11 12 20 25 60] 4th iteration

sorted | unsorted

If n elements,

iterations → (n-1)

For each iteration, find the min ele → O(n)

TC → $O(n^2)$

$(n-1) + (n-2) + (n-3) + (n-4) + \dots 1$

$= \frac{(n)(n-1)}{2} = \Theta(n^2)$

Time Complexity

Best → $O(n^2)$
Avg → $O(n^2)$
Worst → $O(n^2)$

Space Complexity → O(1)

Activity

Name	Age
Joe	25
Karan	29
Amit	21
Naman	28

Order by age →

name	age
Amit	21
Naman	28
Joe	25
Karan	29

(n)

name	age
Amit	21
Joe	25
Naman	28
Karan	29

→

name	age
Amit	21
Joe	25
Karan	29
Naman	28

3 5 1 4 5' 10

stable → 1 2 3 5 5' 10

unstable → 1 2 3 5' 5 10

The order in which element occurs is maintained even after sorting

Insertion sort → stable
Selection sort → unstable
Bubble sort → stable

selection sort

4 5 3 2 4' 1
1 5 3 2 4' 4
1 2 3 5 4' 4
1 2 3 5 4' 4
1 2 3 4' 5 4
1 2 3 4' 4 5

Merge Sort

→ Divide & conquer algorithm

Bigger problem

Smaller problem | Smaller problem

Given an array, find the max

[3 2 6 9 1 3 2 7 10]

Smaller problem → 9 | Smaller problem → 10

9 ← max → 10

1. Divide the problem into 2 halves
2. Merge the result of 2 smaller problems

Divide & conquer

→ Binary search
→ Ternary search
→ Quick sort
→ Merge sort

1. Find the middle point
   $m = (l + r)/2$
2. Call merge sort for first half
   ms(arr, l, m);
3. Call merge sort for second half
   ms(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3
   merge (arr, l, m, r)

1. Logic
2. Implementation
3. Time Complexity

$T(n) = 2T\left(\frac{n}{2}\right) + n$ → (1)

Recursive relation

→ substitution method

$n \to \frac{n}{2}$

$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$

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

$n \to \frac{n}{2^2}$

$T\left(\frac{n}{2^2}\right) = 2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}$

$T(n) = 2^2 T\left(\frac{n}{2^3}\right) + 2n$ → (3)

$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + kn$

$T\left(\frac{n}{2^k}\right) = T(1)$

$\therefore \frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log n$

$T(n) = 2^k T(1) + kn$

$T(n) = n \times 1 + n\log n$

$T(n) = n\log n$

Time Complexity

Worst → $O(n\log n)$
Avg → $O(n\log n)$
Best → $O(n\log n)$

Space Complexity → $O(n)$

Inversion Count

→ Inversion count for an array tells how far the array is from being sorted.

Formally speaking, two elements a[i] and a[j] form an inversion if

a[i] > a[j] & i < j

Eg (1) arr → [8 4 2 1]

output → (6)

(8, 4), (8, 2), (8, 1)
(4, 2), (4, 1) (2, 1)

arr → [3, 1, 2]

output → (2)

(3, 1) , (3, 2)