Saturday, 27 February 2021 9:29 PM

→ Selection Sort

arr → [64 25 12 22 11]

→ sorts an array by repeatedly finding the min element from unsorted part and putting it at the beginning.

	64	25	12	22	11
①	Find min (0, 4) → 11				
	11	25	12	22	64
	sorted	unsorted			
②	Find min (1, 4)				
	11	12	25	22	64
	sorted	unsorted			
③	11	12	22	25	64
④	11	12	22	25	64

If → n elements

Iterations → n-1

For each iteration → find min etc → O(n)

TC → $O(n^2)$

n-1 iterations

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

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

SC → O(1)

Stability

Name	Age
Joe	25
Nanou	29
Anil	21
Nanou	28

Sorted by age →

Name	Age
Anil	21
Nanou	29
Joe	25
Nanou	28

Sorted by name

Name	Age
Anil	21
Joe	25
Nanou	29
Nanou	28

(✗)

Name	Age
Anil	21
Joe	25
Nanou	28
Nanou	29

(✓)

3 5 2 1 5' 10

same elements

stable sort

1 2 3 5 5' 10

unstable sort

1 2 3 5' 5 10

Insertion sort stable ?? (Yes)

Bubble sort ?? (Yes)

Selection sort ??

Merge Sort

→ Divide & Conquer algorithm

Divide the problem into smaller problems

smaller problem

smaller problem

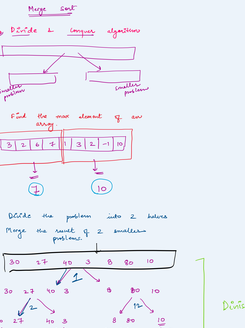
Q Find the max element of an array

3	2	6	7	1	3	2	-1	10

→ 7 → 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 part
$m = l + (r - l)/2$

② Call mergeSort for first half. ms(arr, l, m)

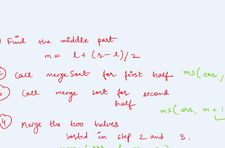
③ Call merge sort for second half. ms(arr, m+1, r)

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

Recurrence relation

→ Substitution method

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

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

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

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

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

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

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

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

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

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

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

$k = \log n$

$T(n) = 2^k T(1) + kn$ $\boxed{k = \log n}$

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

$\boxed{T(n) = n\log n}$

Merge Sort

TC

Worst case → $O(n\log n)$

Best case → $O(n\log n)$

Avg case → $O(n\log n)$

SC → O(n)

Q Inversion Count

Inversion Count for an array tells how far the array from being sorted.

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

Eg ① arr[8 4 2 1]

output → 6

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

② arr[3, 1, 2]

output → 2

(3, 1), (3, 2)