Selection Sort

arr → [60 25 12 20 11]

→ sort the array by repeatedly finding (selecting) the min ele from unsorted part and putting it at the beginning.

unsorted

60 25 12 20 11

Find min [0,4] → 11

11 | 25 12 20 60 — 1st iteration

sorted | unsorted

min [1,4] → 12

11 12 | 25 20 60 — 2nd iteration

sorted | unsorted

min [2,4] → 20

11 12 20 | 25 60 — 3rd iteration

sorted | unsorted

min [3,4] → 25

11 12 20 25 | 60 — 4th iteration

1 → 1 ele

If n elements,

Iterations → (n − 1)

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

TC → $O(n^2)$

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

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

Time Complexity

Best → $O(n^2)$

Avg → $O(n^2)$

Worst → $O(n^2)$

Space Complexity → $O(1)$

Stability

Eg:

Name	Age
Joe	25
Naman	24
Amit	21
Naman	28

sorted by age →

Name	Age
Amit	21
Naman	24
Joe	25
Naman	28

sorted by name

(a)

Name	Age
Amit	21
Joe	25
Naman	24 ✓
Naman	28 ✓

(b)

Name	Age
Amit	21
Joe	25
Naman	28
Naman	24

3 5 1 2 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 → HW]

3 3' 7 4

7 > 3'

3 = 3 smaller

selection sort

4 5 3 2 4' 1

1 | 5 3 2 4' 4

1 2 | 3 5 4' 4

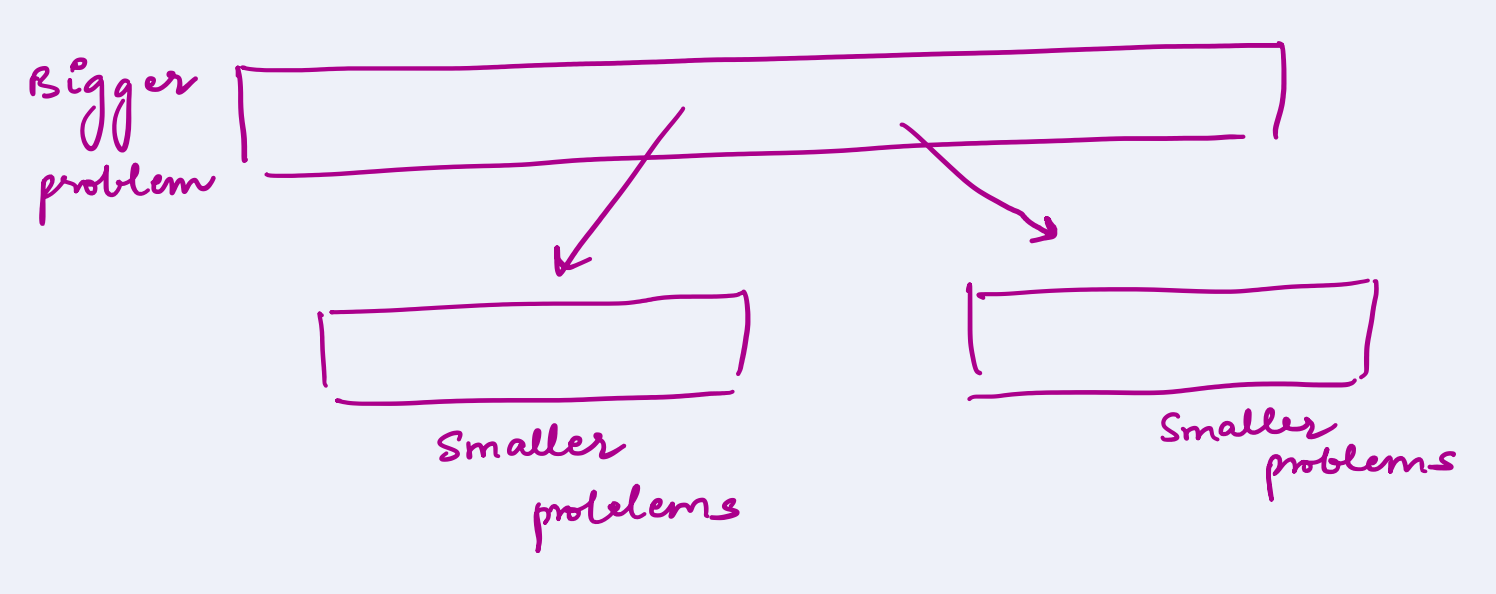
1 2 3 | 5 4' 4

1 2 3 4' | 5 4

1 2 3 4' 4 | 5

Merge Sort

→ Divide & Conquer Algorithm

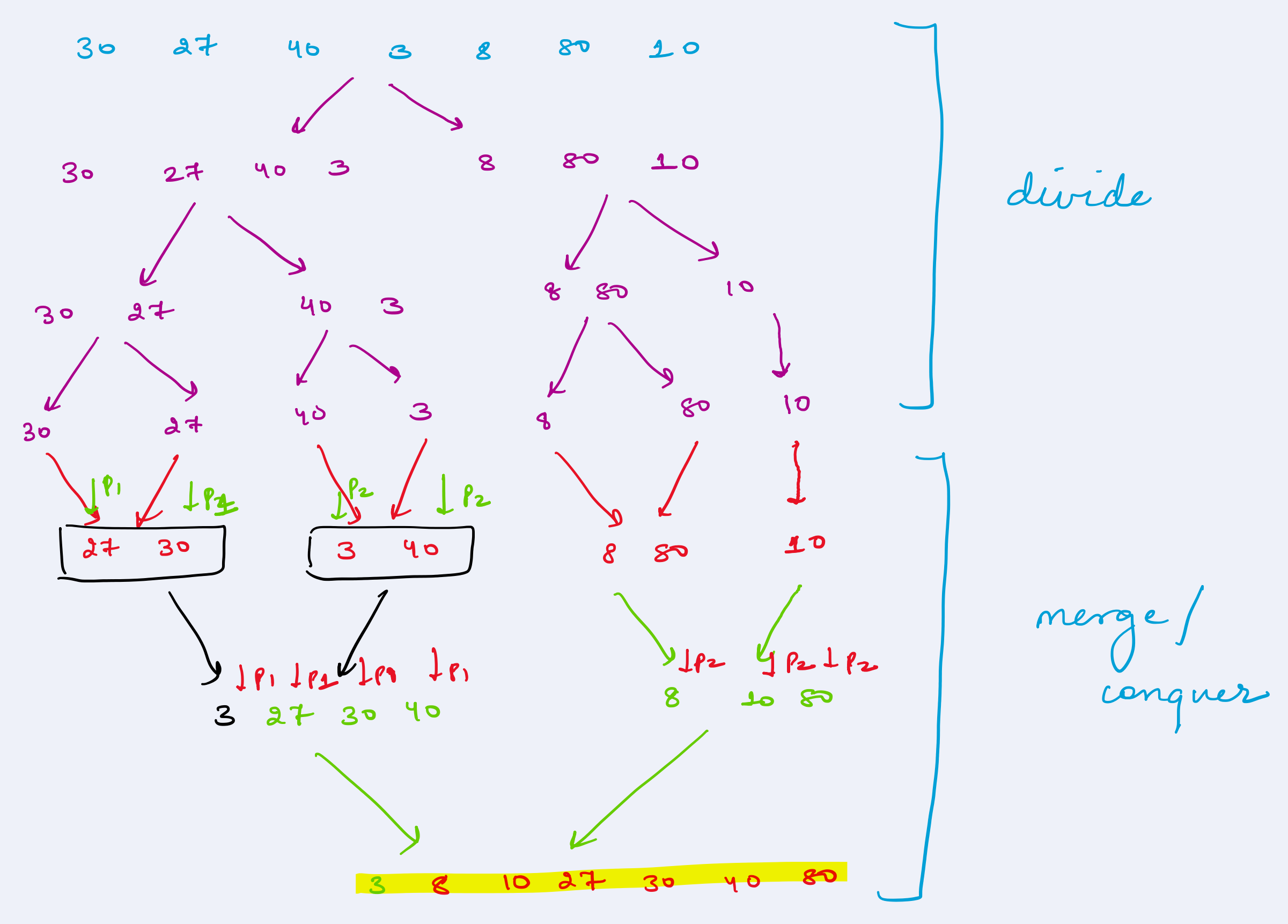


Q Given an array, find the max element.

Big → [3 2 6 7 | 1 3 2 −1 10]

smaller problem (7) < smaller problem (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

l mid r

① Find the middle part

$m = (l + (r-l)/2)$

② call merge sort for first half

ms(arr, l, m);

③ Call merge sort for second half

ms(arr, m+1, r)

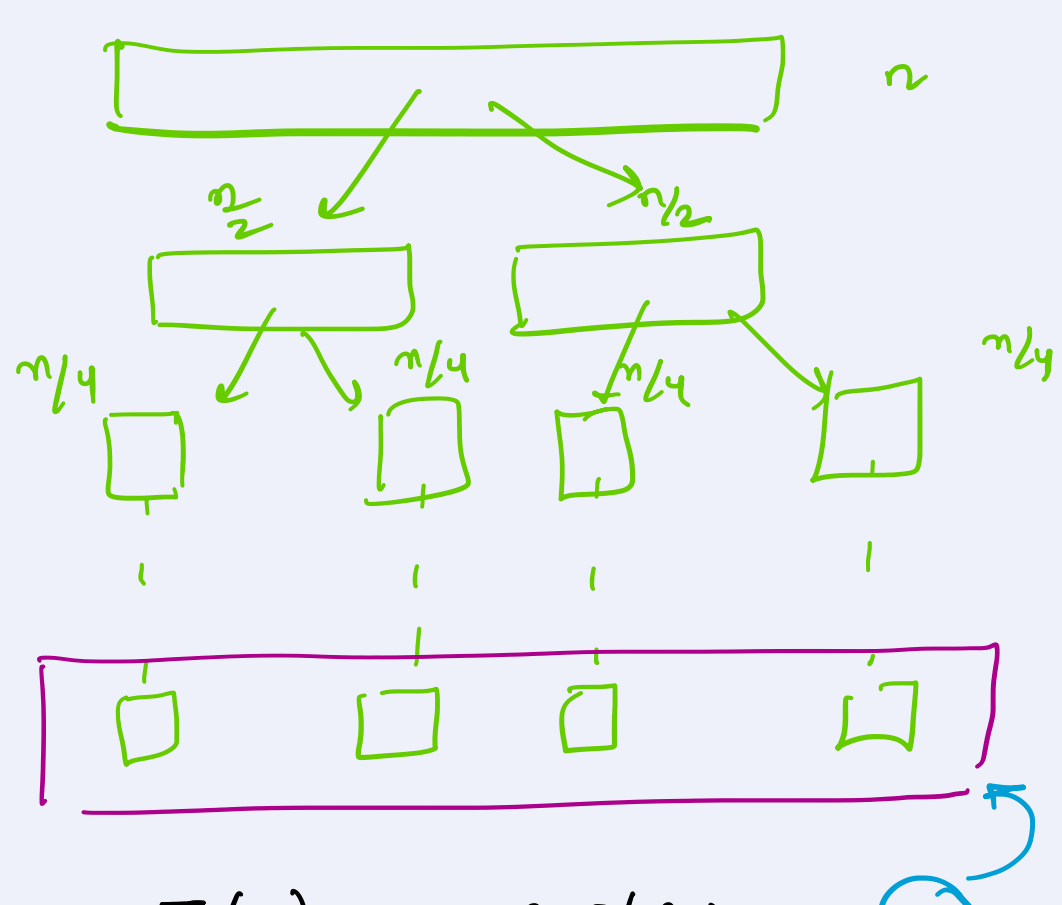
④ Merge the two halves sorted in step ② & ③.

merge (arr, l, m, r)

① logic

② Implementation

③ Time Complexity



$T(n) = 2T(\frac{n}{2}) + n$ — ①

Recurrence relation

→ Substitution method

$n \rightarrow \frac{n}{2}$

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

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

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

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

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

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

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

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

$\boxed{k = \log n}$

$2^{\log_2 n}$

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

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

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

Merge sort

TC

Worst → $O(n\log n)$

Best → $O(n\log n)$

Avg → $O(n\log n)$

SC → $O(n)$

Q 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] and i < j

Eg ① arr [8 4 2 1]

output → ⑥

(8, 4), (8, 2), (4, 2), [8, 1], [4, 1], [2, 1]

Eg ② arr = {3, 1, 2}

output → ②

(3, 1), (3, 2)

Merge sort