Friday, 18 June 2021 8:25 PM

Selection Sort

Eg:

| 62 23 10 20 9
sorted | unsorted

Pick the min ele from unsorted region and put it in the unsorted region.

i = 0

6 2 23 10 20 9 (62, 9)

① find min → n-1 comparisons

min: 1 2 3 4 5

i = 1

I 9 | 23 10 20 62 — n-2 comp. (min)

i = 2 unsorted

II 9 10 | 23 20 62 — n-3
sorted | unsorted

i = 3

III 9 10 20 | 23 62 — n-4
sorted | unsorted

IV 9 10 20 23 | 62 — n-5
sorted

Total comparisons → (n-1) + (n-2) + (n-3) + ... 1

→ $\frac{n \times (n-1)}{2} = O(n^2)$

SC → O(1) [In-place algorithm]

TC
Best → $O(n^2)$
Worst → $O(n^2)$
Average → $O(n^2)$

for n sorting n elements
↓
n-1 iterations
↓
for each iteration, 1 swap.

TS → (n-1)

Bubble Sort

→ works by repeatedly swapping the adjacent elements if they are in wrong order

I iteration

6 2 5 3 9
2 6 5 3 9
2 5 6 3 9
2 5 3 6 9

(n-1)

II iteration

2 5 3 6 9
2 5 3 6 9
2 3 5 6 9

boolean flag ✓
(n-2) swaps

III iteration

2 3 5 6 9
2 3 5 6 9
2 3 5 6 9

(n-3) swaps

IV iteration

2 3 5 6 9
2 3 5 6 9

→ n-4 swaps

n-1 iterations

TC → (n-1) + (n-2) + (n-3) + . . 1
= n(n-1)/2

1 2 3 4 5

TC
Worst → $O(n^2)$
Best → $O(n)$
Average → $O(n^2)$

SC

SC → O(1) [In-place sorting algo]

Applications:

1. simple & easy to understand
↳ short code

2. complexity doesn't matter

Stability of sorting algo's

Eg:

Name	Age
Joe	25
Naman	24
Amit	21
Naman	28

sort by age →

Name	Age
Amit	21
Naman	24
Joe	25
Naman	28

sort 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 2 1 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.

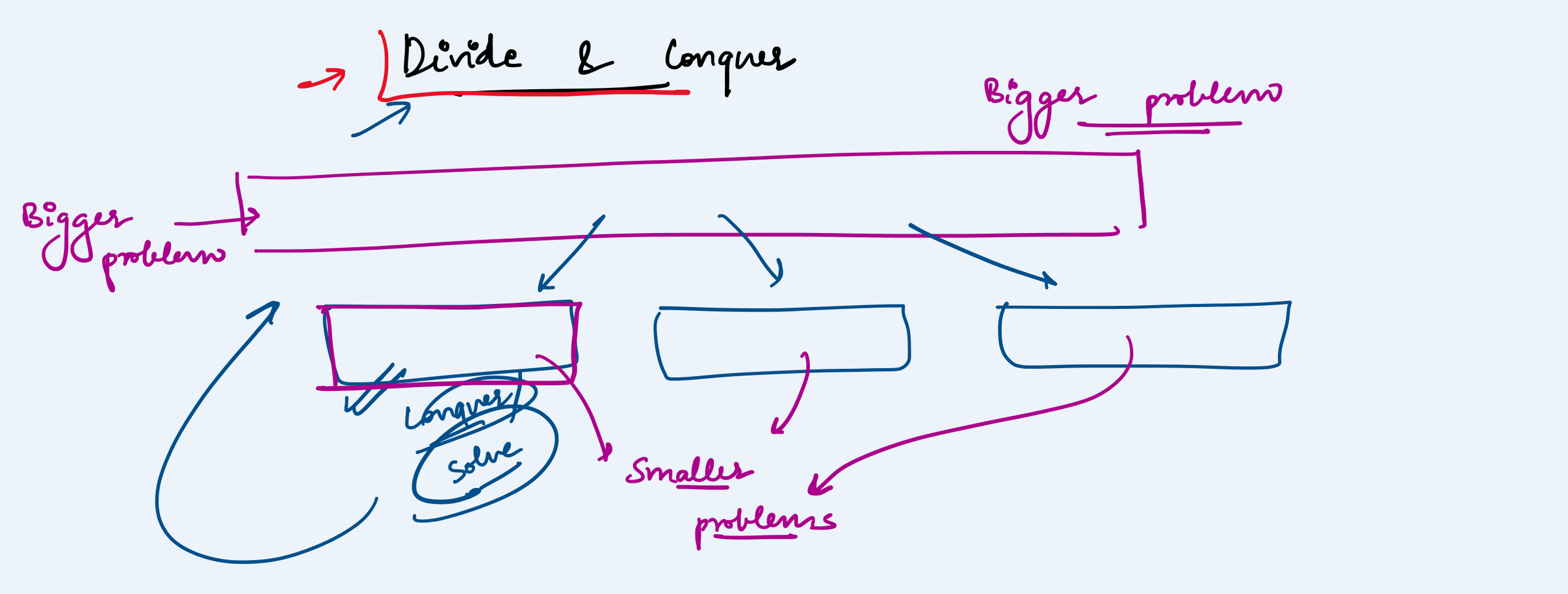
Insertion → Yes (stable)
Selection → Unstable NO
Bubble sort → (stable)

7 3 3' 4
3 7 3' 4
3 3' 7 4

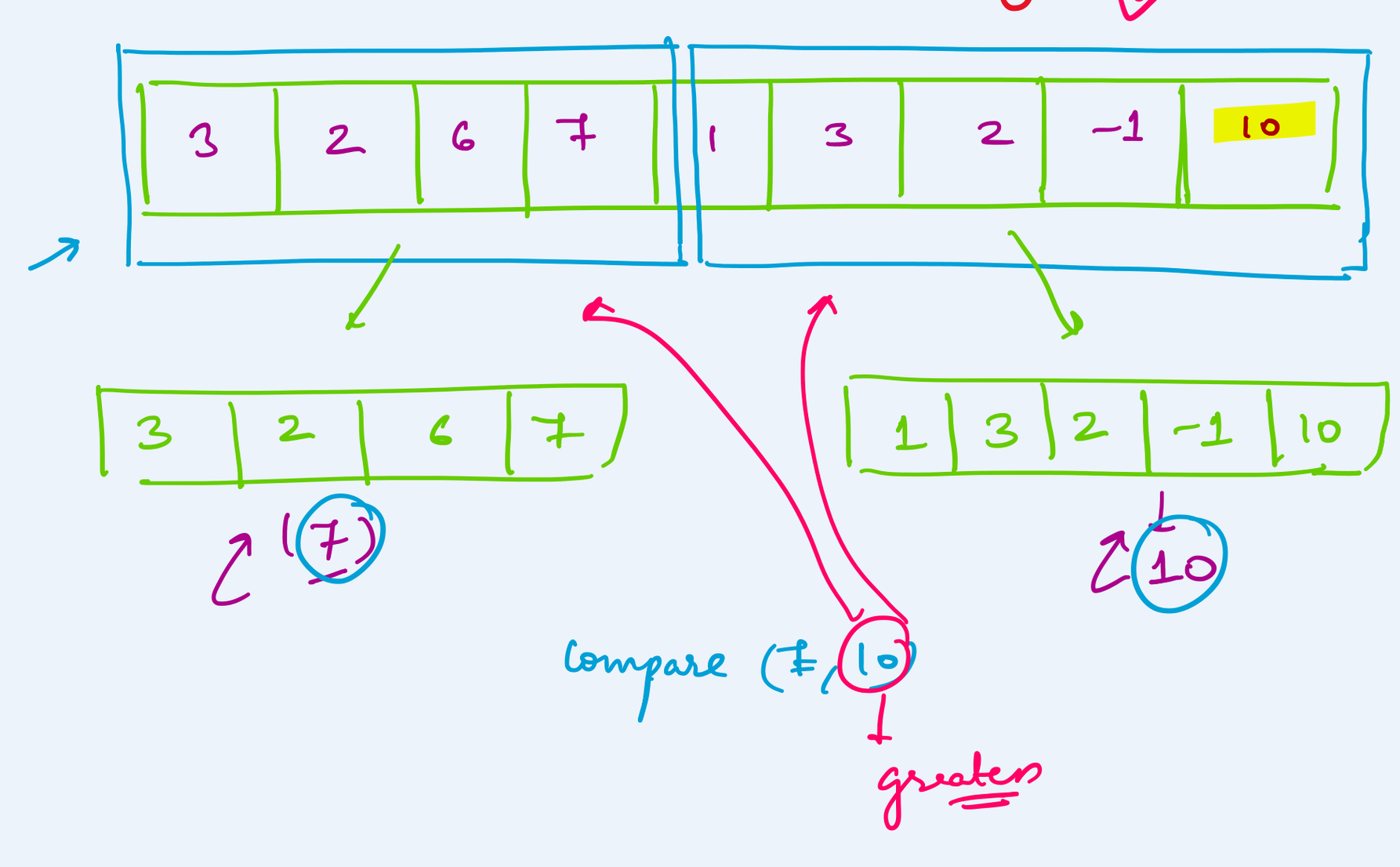
4 5 3 2 4' 1
1 | 5 3 2 4' 4
1 2 | 3 5 4' 4
1 2 3 | 5 4' 4
1 2 3 4' | 5 4
1 2 3 4' 4 | 5

_ _ 4 4' _

Divide & Conquer



Q Find the max element of an array.

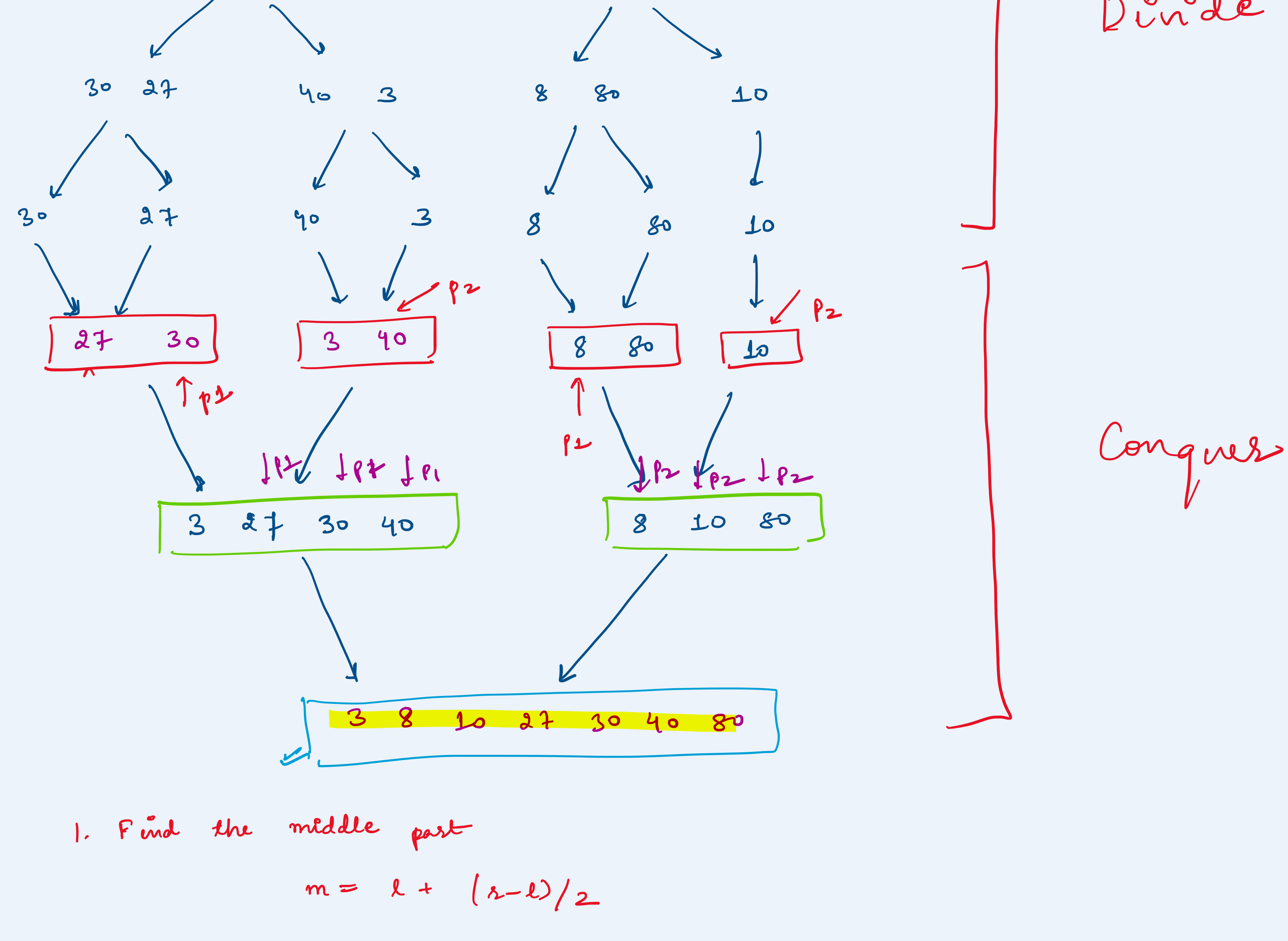


① Binary Search
② Ternary search
③ Quick sort
④ Merge sort

Merge Sort

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

(sort)



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 and step 3

merge(arr, l, m, r);