Date:

Process of Interviews

Tier 1 companies:

1st Round → DSA coding Round + MCQ's
(2 Problems)
[Amazon, Google, Facebook]
(Easy - Moderate)

2nd Round → DSA (Technical Aspects)
↳ Fundamentals
(45 mins) BFS or Merge sort or time complexity
Interview Questions
↳ Easy → Medium → Hard
↳ Leetcode → Easy + Medium
↑
Direct Questions
System Design

3rd Round → Industry Projects,
Real time Projects → End to End
Pipeline.
4th Round → Behavioral Round
5th Round → HR Round

printo.in

Date:

▷ Google → Easy → Collinear points

$(x_1, y_1) = (1, 6)$ → P_1
$(x_2, y_2) = (1, 0)$ → P_2
$(x_3, y_3) = (1, 9)$ → P_3

Collinear → lie on the straight line.

Like slope?

↳ slope formula

$$\frac{y_2 - y_1}{x_2 - x_1} \overset{?}{=} \frac{y_3 - y_2}{x_3 - x_2}$$

$$(y_2 - y_1) * (x_3 - x_2) = (x_2 - x_1) * (y_3 - y_2)$$

LHS RHS

If LHS == RHS then collinear.

collinear → Area of triangle = 0

$$\text{Area} = \frac{1}{2}(x_1 * (y_2 - y_3) + x_2 * (y_3 - y_1) + x_3 (y_1 - y_2))$$

```
if area == 0:
    print('yes')
else:
    print('no')
```

printo.in

Date:

2) Amazon

Left Right

(-1, 1, 4, 2) → 0 1 2 3

↳ Sort (-1, 1, 2, 4)

Keep adding & updating the sum value
Use Greedy Approach.

Merge Sort

0	1	2	3	4	5	6	7
50	20	40	70	10	60	39	64

n = 8

Divide & Conquer

n == 1 → return arr(i)

n > 1 → D & C

We can take value as 2 also.

- We will divide array continoudy till we have single element

- mid = (i+j)/2 = (0+7)/2

= 3

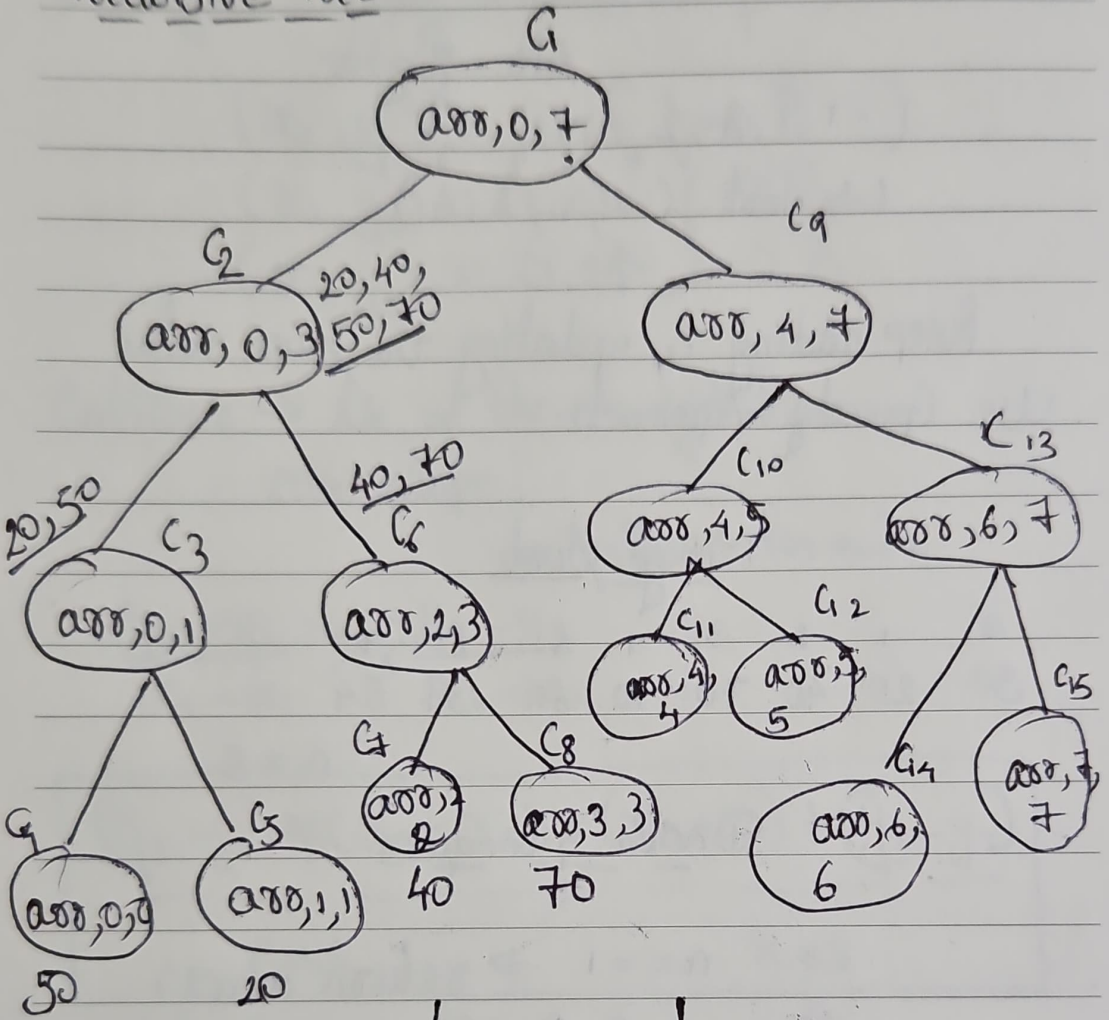
i, mid = 0, 3

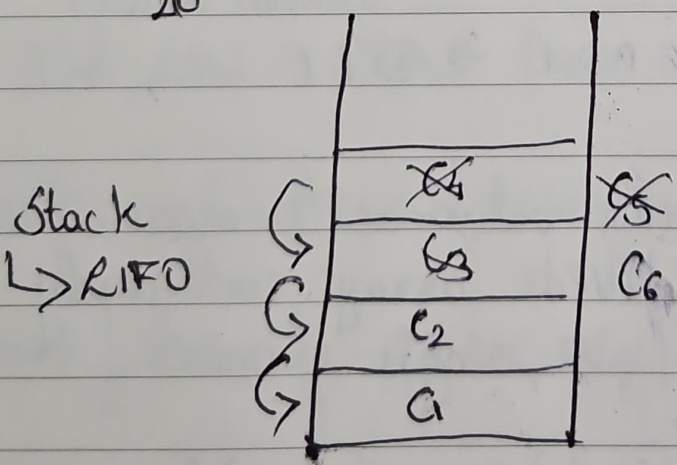
mid+1, j = 4, 7

printo.in

Date:

Recursive Tree





Merge Procedure → will do combine part

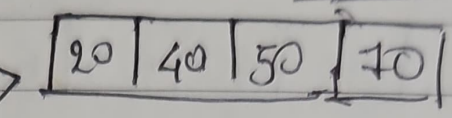
printo.in

Date:

Merge Procedure

Left Subarray: 20, 50 — i

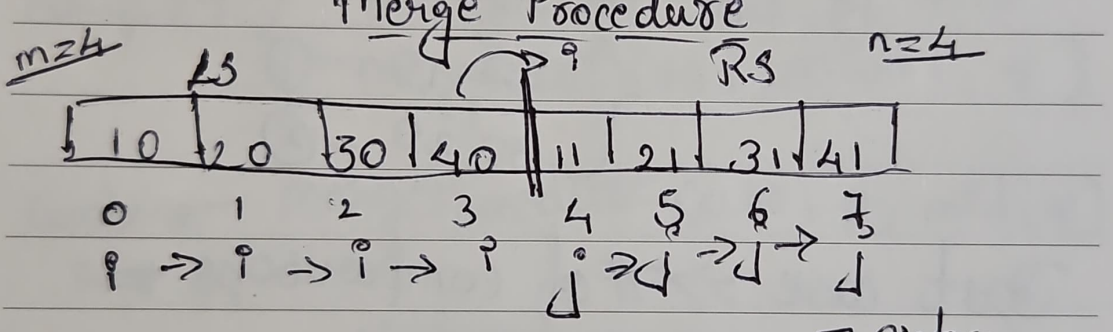
Right Subarray: 40, 70 — j

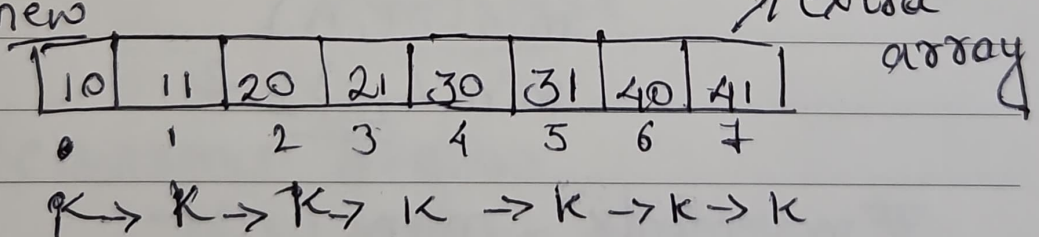


extra array

20, 40
50, 40
50, 70

Merge Procedure





Worst case = $m+n-1$

no. of comparisions

$LS = m$ $\quad m+n = N$

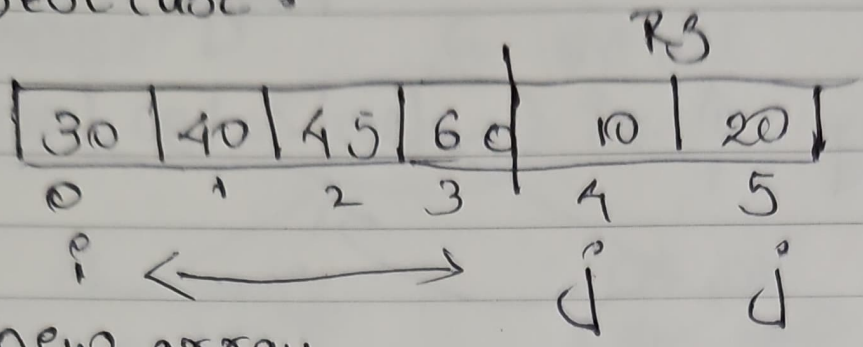
$RS = n$

10, 11 → (1)
20, 11 – (2)
20, 21 – (3)
30, 21 – (4)
30, 31 – (5)
40, 31 – (6)
40, 41 – (7)

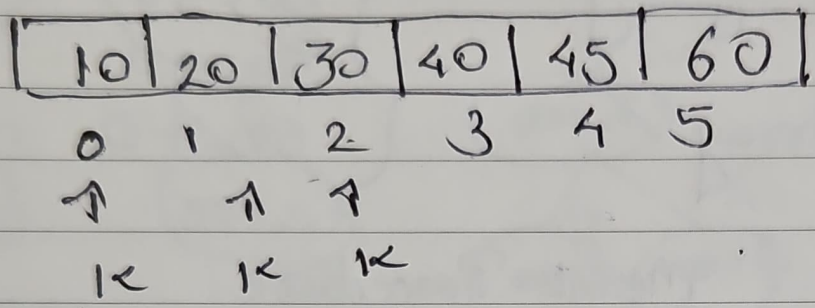
printo.in

Date:

Best case:



new array



{ 30, 10 - ①
{ 30, 20 - ②

Best case => # of comparisons = min(m, n)
min(4, 2) = 2

movements = m+n

Time complexity = # of comparisons + # of movements

Worst case
→ (m+n-1) + (m+n)
→ O(m+n)
→ O(N) => m+n=N

Date:

Best case → min(m, n) + (m+n)
⇒ O(N)

Pseudocode :-

```
mergesort(arr, p, q)
{ if p == q;
    return arr(p)
  else;
    mid = p + (q-p)/2
    mergesort(arr, p, mid)
    mergesort(arr, mid+1, q)
}
mergeProcedure(arr, p, mid, q)
return arr
```

Small problem → C
Divide → C
conquer
Combine → N

Recurrence Relation

$$T(n) = \begin{cases} C & ; N=1 \\ 2T(N/2) + N & ; N>1 \end{cases}$$

$a = 2$
$b = 2$
$k = 1$
$p = 0$

$\log_b a = \log_2 2 = 1$

$\log_b a = k$ case 2

$p > -1$

Every case ← $O(n \log n)$

printo.in

Date:

other cases → $O(n^2)$ → Sorting
Merge sort → $O(n \log n)$
↳ Better.

Space complexity → Outplace sorting
↳ extra space + stack algo space
↳ $O(N)$ ↳ $\log N$
$\Rightarrow O(N)$

⟶ Stable Algorithm
↳ Retain patter with index

Foomula → mid - i + 1
↳ To calculate no. of elements in array

Higher Index − Lower Index + 1

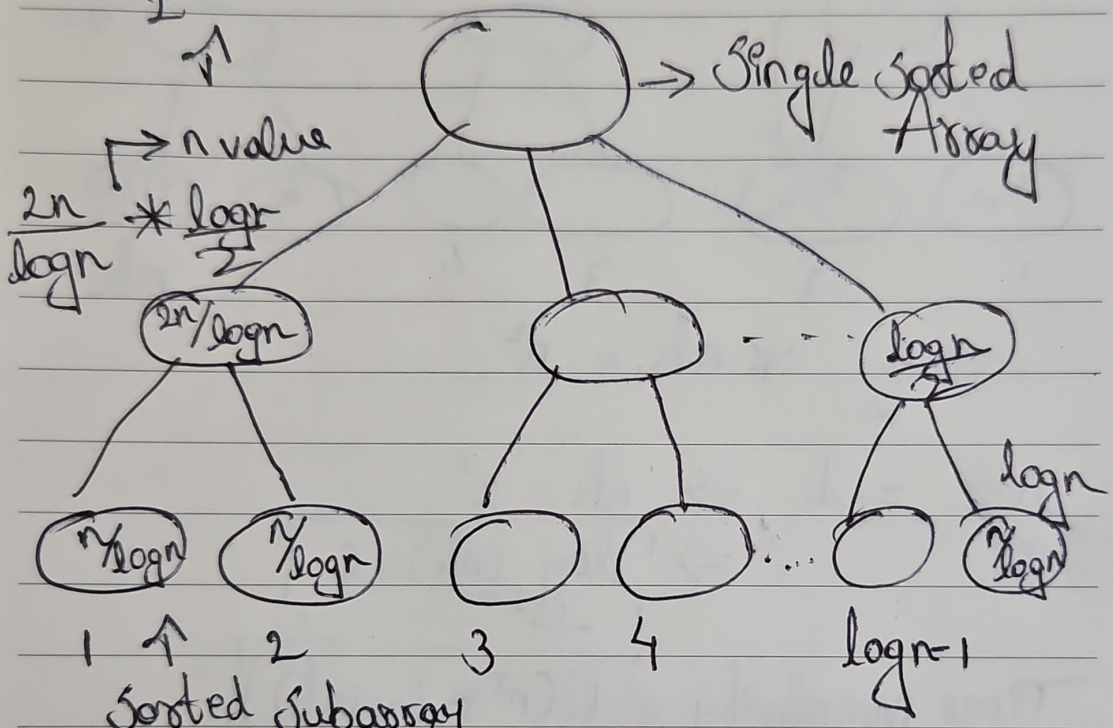
$\Rightarrow J + (m - 1) + 1$
$\Rightarrow J - mid - 1 + 1$
$\Rightarrow J = mid$

Date:

Merge Sort Questions:-

1) logn sorted subarrays each of size n/logn
⇒ time complexity of single sorted array?

$\frac{\log n}{2^k} = 1$ → Single sorted Array



Total no. of elements = n

$\frac{n}{\log n} * \log n = n$ → Total size of the array.

(n/logn: size; logn: subarrays)

Every level value of n is same.

$\frac{\log n}{2^k} = 1$

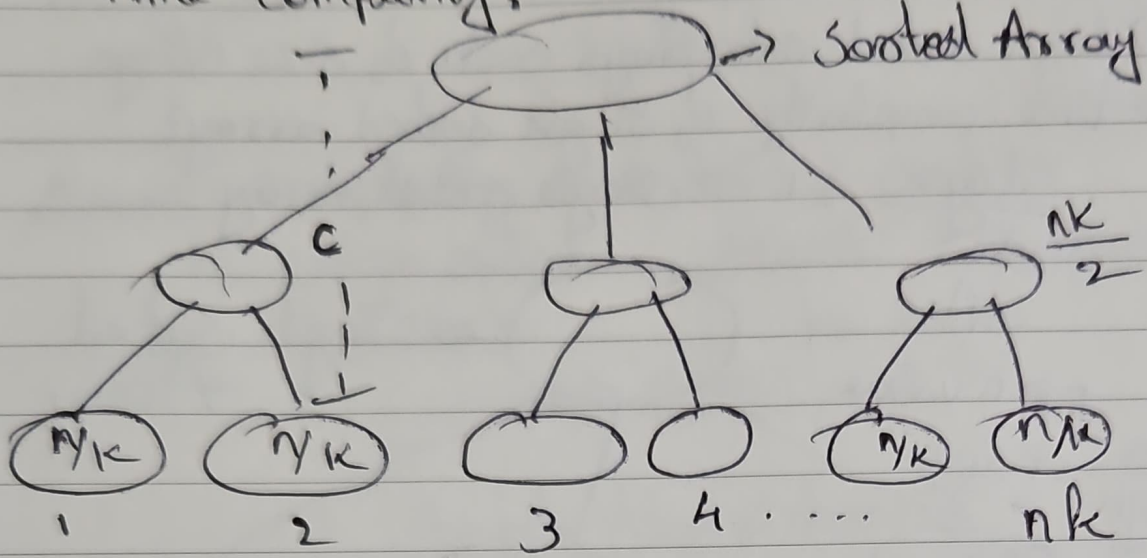
$\log n = 2^k \Rightarrow k = \log_2(\log n)$

Time Complexity = $O(n \log(\log n))$

printo.in

Date:

2) nk sorted subarrays each of size n/k
Time complexity?



$\frac{n}{k} * nk = n^2$

$\frac{nk}{2^c} = 1 \Rightarrow nk = 2^c$

$\Rightarrow \log_2(nk) = c$

Time complexity = $O(n^2 * \log(nk))$

printo.in