MCQ

ISTE - HJT

Time and space complexity

MCQ

1) (b) O(log n)

↳ Explanation is given in the solution of (Q10)

2) (a) O(n)

3) (d) 500

ISTE - HJT

Time and Space Complexity

4) $i \rightarrow 1 \ldots n$ (length of array)

$j \rightarrow 1 \ldots n$ (length)

∴ Time complexity $= O(n * n)$ → No. of times the loop will be executed.

$= \boxed{O(n^2)}$

Space complexity:

$\boxed{O(1)}$ → input space for the length of array

(No extra space is used)

5) ~~Time complexity~~

sum = 0;

$1 \longrightarrow n$

(n times iteration)

∴ $\boxed{\text{Time complexity} = O(n)}$

only one variable is used which is constant

↳ $\boxed{\text{Space complexity} = O(1)}$

6) ⊙ i ⟶ length(array) ⟶ say (n)

n times ~~not~~ iterations, as it ⊙ is traversing through the entire array ~~without~~ irrespective of the condition

& ⊙ Space used is the ~~size~~ Size taken by the array. (n)

∴ **Time complexity = O(n)**

∴ **Space complexity = O(1)**

7) ⟶ could it understand the code snippet
↳ it is in C++

8)

(No of iterations before loop terminates = k+1)

i	i < n
1 (2^0)	True
2 (2^1)	True
4 (2^2)	True
8 (2^3)	True
⋮	- - -
2^k	false

2^k ~~> N

Condition breaks when $\rightarrow$ $2^k = n$

$k \log 2 = \log n$

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

Const. to be ignored

Constant can be ignored in time complexity

$k = \log_2 n$

$\therefore \log_2 n + 1$ is the total no. of approximate iterations

$\therefore$ **Time complexity = $O(\log n)$**

Space complexity = $O(1)$

input size of array

(Sum variable size being const. ignored)

9) **Time complexity = $O(n)$**

worst case

$i \rightarrow (1 - n)$

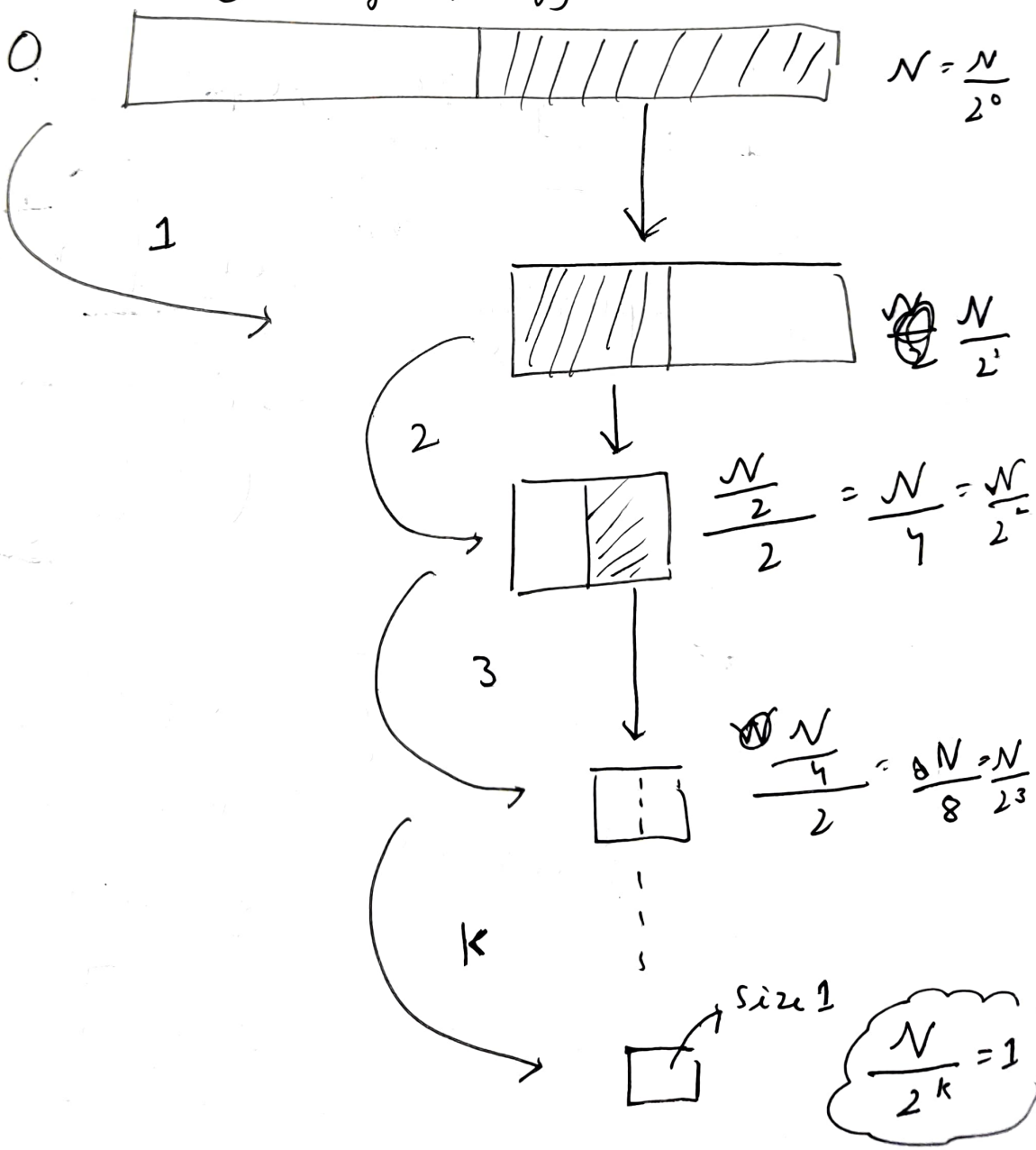
$\rightarrow$ n no. of iterations

Space complexity = $O(1)$ $\rightarrow$ one variable which is constant.

10) Binary Search algorithm

- the whole array gets divided in two equal parts at each iteration
- Based on the target element and middle element, the searching is done on either of two parts.

(N = length of array)



$N \frac{}{2^k} = 1$

$\frac{N}{2^k} = 1 \Rightarrow N = 2^k$

$\log N = \log 2^k$

$k = \frac{\log N}{\log 2}$ → Const. can be ignored in time complexity.

k → Total no. of levels in the worst case

Time complexity = $O(\log N)$

Space complexity = $O(1)$ → length of the array

→ the other variables "left", "right" & "mid" can be considered as auxillary space but since they are constant for every algorithm, they are ignored.

1 10 n = 10

11) → (Recursion)
↳ couldn't solve

12) outer loop → i ⇒ 1 — k
(k times)

inner loop → j ⇒ 1 - length(array) } n
(n times)

Total iterations = n * k

Time complexity = O(nk)

Space complexity = O(1)
↳ Size of array.

(9)

(X) (6)

13)

i	j	n = length of array.
0	1 ..... (n-1) times	
1	3 .... (n-3) times	
2	5 ... (n-5) times.	
⋮	⋮	
(n-1)	1 time	

Continued

13) Total no. of iterations

$\underbrace{1 + 3 + 5 \ldots\ldots (n-1)}$

↓

Sum of odd no. upto $(n-1)$ times

$= (n-1)^2$

$= n^2 - 2n + 1$ → constant.

↓

less dominating factor

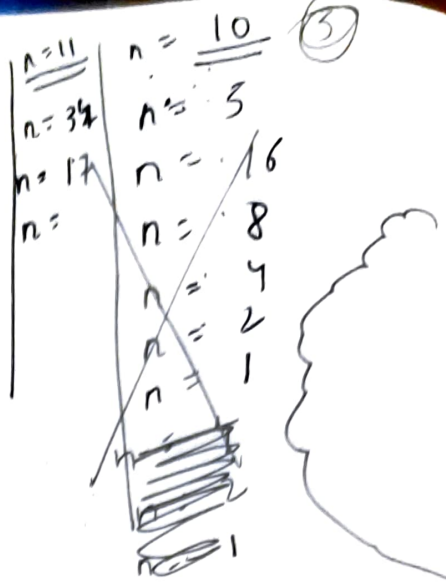
∴ Time complexity : $O(n^2)$

Space complexity : $O(n)$

→ array size

14) → Recursion

↳ (could't solve)



15) n = length of array

i	j
1	1 (1 time)
2	1, 2 (2 times)
⋮	
n	1, 2, 3, ... n (n times)

∴ Total no. of iterations

$$= 1 + 2 + 3 + \dots n$$

$$= \frac{n(n+1)}{2}$$

$$= \frac{n^2 + n}{2} \Rightarrow \begin{cases} n \to \text{less dominating factor} \\ 2 \to \text{const.} \end{cases}$$

Time complexity = $O(n^2)$

Space complexity = $O(1)$

16) Total no. of iteration = $n + n$

$= 2n$

Time complexity = $O(n)$ → 2 → Const. ignored

Space complexity = $O(n+n) = O(2n)$

$= O(1)$

17) → ~~It is with~~ .

Doubt

18) The ~~psu~~ pseudocode is of Multiplication of two matrices

Matrix 1 | Matrix 2

$2 \times 3 \quad 3 \times 1$

Product → 2×1

Product matrix

The no. of operations = No. of rows of first matrix × No. of rows of second matrix

considering both matrix as square of $n \times n$ size

∴ Time complexity = $O(n*n) = O(n^2)$

Space complexity = $O(n^2)$ $O(n*n)$

$= O(n^2)$

5^3 123

19)

i	j	k	
1	1	1	1
2	1, 2	1, 1, 2	2
3	1, 2, 3	1, (1,1), (1,1,1)	6
4	1, 2, 3, 4	1, (1,1), (1,1,1), (1,1,1,1)	10
5	1, 2, 3, 4, 5	1, (1,1), (1,1,1), (1,1,1,1), 1	15

∴ Total no. of iterations $= \left(\sum_{i=1}^{n} \left\{ \sum_{j=1}^{i} \left[\sum_{k=1}^{j} k \right] \right\} \right)$

$= j$ times

Doubt → couldn't solve further.

Space complexity = $O(1)$

20] outer loop → i ⇒ 0 − $(2^n - 1)$

2^n times

inner loop → j ⇒ (0 − (n−1)

n times

∴ Total no of iteration = $n * 2^n$

Time complexity = $O(n 2^n)$

Space complexity = $O(n 2^n)$

array subset

2^{15}

21] → Recursion (could solve)

22] Time complexity = $O(n * n)$

$= O(n^2)$

Space complexity = $O(\frac{1}{2})$

23] Time complexity = $O(n * n)$

$= O(n^2)$

Space complexity = $O(1)$

24) No. of iterations = $n/2$

Time complexity = $O\left(\frac{n}{2}\right) = O(n)$

Space complexity = $O(1)$

25) Transpose of a matrix

↳ Let rows = n = columns

↳ ~~columns = n~~

∴ No. of iteration = $n * n$

Time complexity = $O(n^2)$

Space complexity = $O(n^2)$

26) Time complexity = $O(n)$

Space complexity = $O(1)$

27) T.C. = $O(n)$

S.C. = $O(n)$

28) $T.C = O(\log n)$ → Explained in some of previous Q.

$S.C = O(1)$

29)

i	j
1	1
2	2
3	3
⋮	⋮
n	n

$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$

$T.C. = O\left(\frac{n^2+n}{2}\right) = \boxed{O(n^2)}$

$\boxed{S.C = O(1)}$

30) Outer loop → n iterations → array length

Inner loop → $(k-1)$ iterations

∴ Total iterations = $n*(k-1)$

$= nk - n$ → less dominating

$\boxed{T.C = O(n*k) \\ S.C = O(1)}$

$k = 5$

i	j	(i+k)
1	2	6
2	3	7

n!

Bonus - MCQs

n! × n

1) Outer loop → n! iteration

inner loop → n iterations for each outer loop iteration

∴ Total = n! × n

⊗ T.C = O(n! × n)

= O(n!)

less dominating factor

2) S.C. = O(n)

3) Total iterations = $2^n \times n$

T.C = O($n \times 2^n$)

less dominating

= O(2^n)

4) (Same as 3)