(1)

For binary search of a sorted array,

$\because T(N) = 1 \cdot T\left(\frac{N}{2}\right) + O(1) \rightarrow (C \rightarrow \text{constant})$

$\therefore a_1 = 1 \;;\; b_1 = \frac{1}{2} \;;\; g(u) = O(1) = C$

Now, $a_1 b_1^{\,p} + a_2 b_2^{\,p} + \cdots\cdots = 1$

$\Rightarrow 1 \times \left(\frac{1}{2}\right)^p = 1$

$\Rightarrow \frac{1}{2^p} = 1 \Rightarrow 1 = 2^p \Rightarrow p = 0$

Now, By Akra bazzi's formula,

$\therefore T(N) = O\left(N^p + N^p \int_1^N \frac{g(u)}{u^{p+1}}\, du\right)$

$= O\left(1 + 1 \times \int_1^N \frac{C \rightarrow 1}{u^1}\, du\right)$

$= O\left(1 + \left[\log u\right]_1^N\right)$

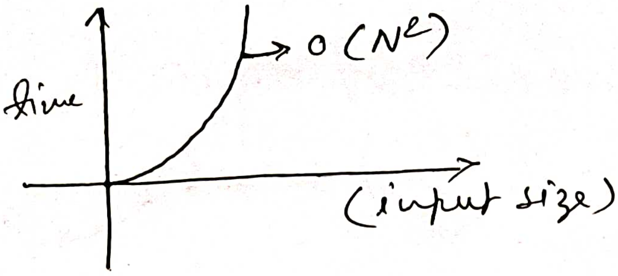
$= O\left(1 + \log N - \log 1\right)$

$= O(\log N) \longrightarrow$ Time complexity of Binary Search.

And, space complexity $= O(1)$

($\because$ no new/extra space is created)

③ (d) 500



(∵ here time increases exponentially as the input size is increased)

④ $i \Rightarrow 1 \longrightarrow N$; $j \Rightarrow 1 \longrightarrow N$

∴ Time complexity $= O(N \times N) = O(N^2)$

As, no extra space is being created so,

Space complexity $= O(1)$ (i.e, constant)

⑤ $i \Rightarrow 1 \longrightarrow n$

∴ Time complexity $= O(N)$

As, no extra space is being created,

∴ Space complexity $= O(1)$

⑥ $i \Rightarrow 1 \longrightarrow N$ (worst case)

∴ Time complexity $= O(N)$

As, no extra space is being created,

∴ Space complexity $= O(1)$

(here as the loop consists of both if & else part so, any one of them will definitely be triggered)

(7) Recursion not started yet.

(8) $i \Rightarrow 1 \rightarrow (i_x < n)$; Cond^n for updatation $\Rightarrow i = i \times 2$

$\therefore i = 1, 2, 4, 8, \ldots\ldots, \cancel{k}$ (last term) $\rightarrow P$

As, the above progression seems like a G.P.,

~~$\therefore i_x =$~~ $\therefore$ Const. of G.P $= \frac{2}{1} = \frac{4}{2} = \frac{8}{4} = \ldots = 2$

$\therefore i_x = 2^P \rightarrow$ (last term.)

And provided that,

~~$\therefore k < n$~~ $\therefore$ (last term) $< n$

$\Rightarrow 2^P < n$

$\Rightarrow P \log 2 < \log n$

$\Rightarrow P < \frac{\log n}{\log 2}$

$\therefore$ Time Complexity $= O\left(\frac{\log n}{\log 2}\right) = O(\log n)$

$\therefore$ Space Complexity $= O(1)$ ($\because$ no extra space is created)

(9) $i \Rightarrow 1 \longrightarrow n$

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

And as no extra space is created / used,

$\therefore$ Space complexity $= O(1)$

(10) from (Q1), Time Complexity $= O(\log N)$

& Space Complexity $= O(1)$

(12) $i \Rightarrow 1 \rightarrow K$ & $j \Rightarrow 1 \rightarrow N$ (worse case)

$\therefore$ Time complexity $= O(K \times N) = O(KN)$ ~~$= O(N)$~~

As, no extra space is created,

$\therefore$ Space complexity $= O(1)$

(13) $i \Rightarrow 0 \rightarrow (N-1)$ & $j = (i+1) \rightarrow (N-1)$
(i.e. 1, 2, 3, ...)

$\therefore$ Time complexity $= O(N \times N) = O(N^2)$

& space complexity $= O(1)$

~~(14)~~

(15) $i \Rightarrow 1 \rightarrow N$; $j \Rightarrow 1 \rightarrow i$ ~~(let, time taken for 1 iteration (of j = t)~~
(1, 2, 3, 4)

~~$\therefore$ time taken $= t + 2t + 3t + 4t + \ldots$~~
~~by j~~
~~$\therefore$ time taken $= t + 2t + 3t + \ldots = \frac{N(N+1)}{2} \pm Nt$~~

$\therefore$ Time complexity ~~$= O(N \times N)$~~ ~~($\because$ t is a constant)~~

$= O(N^2)$

$\therefore$ Space complexity $= O(1)$

(16) $i \Rightarrow 1 \rightarrow N$

& since ~~it is~~ there are two similar processes one after another,

$\therefore$ Time complexity $= O(N+N) = O(2N)$

$= O(N)$

& Space complexity $= O(1)$

(20) $i \Rightarrow 0 \longrightarrow \cancel{A}\ 2^N$; $j \Rightarrow 0 \longrightarrow \cancel{A}(N-1)$

$\therefore$ Time complexity $= O((2^N+1) \times (N \cancel{-1} \cancel{+1}))$

$= O(2^N \times N)$

$= O(N \cdot 2^N)$

$\therefore$ Space complexity $= O(N)$
(as, the size of array "subset" is being increased dynamically.)

(22) $i \Rightarrow 1 \longrightarrow N$; $j \Rightarrow 1 \longrightarrow N$

$\therefore$ Time complexity $= O(N^2)$

$\therefore$ Space complexity $= O(1)$

(23) $i \Rightarrow 1 \longrightarrow N$; $j \Rightarrow 1 \longrightarrow N$

$\therefore$ Time complexity $= O(N^2)$

$\therefore$ Space complexity $= O(1)$

(24) $i \Rightarrow 1 \longrightarrow (N/2)$

$\therefore$ Time Complexity $= O(N/2) = O(N)$

$\therefore$ Space complexity $= O(1)$

(25) $i \Rightarrow 1 \longrightarrow N$ & $j \Rightarrow 1 \longrightarrow K$ (let, length(Matrix[0]) $= K$ in worst case)

$\therefore$ Time Complexity $= O(N \times K)$
$= O(NK)$

$\therefore$ Space complexity $= O(NK)$

(26) $i \Rightarrow 1 \longrightarrow N$ (in worst case),

$\therefore$ Time complexity $= O(N)$

$\therefore$ Space complexity $= O(1)$

(27) $i \Rightarrow 1 \longrightarrow N$.

$\therefore$ Time complexity $= O(N)$

$\therefore$ Space complexity $= O(N)$

(28) $i \Rightarrow 1 \longrightarrow N$ & $i = (i \times 2)$

$\therefore$ Time Complexity $= O(\log N)$

$\therefore$ Space Complexity $= O(1)$

(29) $i \Rightarrow 1 \rightarrow N$; $j \Rightarrow i \longrightarrow 1$
$(1,2,3,\ldots N)$

$\therefore$ Time Complexity $= O(N^2)$

$\therefore$ Space Complexity $= O(1)$

(30) $i \Rightarrow 1 \rightarrow (N-K)$ & $j \Rightarrow (i+1) \rightarrow (i+K)$

where, $i = 1, 2, \ldots, (N-K)$

$\therefore$ Time Complexity $= O((N-K)\cdot K)$ $\quad j = 2, 3, \ldots, (N-K+1) < (N-\cancel{K}+\cancel{K})$

$\therefore$ Space Complexity $= O(1)$

BONUS MCQS :-

2.> $O(n)$ ($\because$ for 'n' times the array size is being increased per iteration)

~~3.>~~

3&4.> $O(n \cdot 2^n) = O(2^n)$ [$\because$ n has a very low contribution to the complexity as compared to 2^n here]

1.> $i \Rightarrow 0 \rightarrow (N!-1)$

$j \Rightarrow N \rightarrow 1$

$\therefore$ Time complexity $= O((N! \cancel{-1} \cancel{+1}) \times N)$

$= O(N \cdot N!)$ ($\because N <<< N!$ for worst cases where, $(N \rightarrow \infty)$ or very large input size)

$= O(N!)$