CS & IT ENGINEERING

Algorithms

Introduction to Algorithms and Analysis

Lecture No.- [redacted]10

By- Dr. Khaleel Khan Sir

Recap of Previous Lecture

Topic Problem Solving with ASNs

Topic Analysis of Recursive Algorithms

Topic

Topic

Topic

Topics to be Covered

Topics

Framework for Analysing Recursive Algo

Loop Complexities

Topic: Asymptotic Notations

Let f(n) = n and g(n) = $n^{(1 + \sin n)}$, where n is a positive integer. Which of the following statements is/are correct?

I: $f(n) = O(g(n))$

II: $f(n) = \Omega(g(n))$

[GATE-2015: 2M]

(A) Only I

(B) Only II

(C) Both I and II

(D) Neither I nor II

Topic: Asymptotic Notations

$n < n^2$

MSQ

Let f and g be functions of natural numbers given by f(n) = n and g(n) = n^2.
Which of the following statements is/are TRUE?

[GATE-2023-CS:1M]

(A) $f \in O(g)$

(B) $f \in \Omega(g)$

(C) $f \in o(g)$

(D) $f \in \theta(g)$

Topic : Running Times of Program Segments with Loops:

4. for (i= 1; i < = n; + + i) : n
   for (j = 1; j < = n; + + j) : n
   for (k = n/2; k < = n; k + = n/2) : 2
   c = c + 1;

$K = n/2 + n/2 = n + \frac{n}{2} = \frac{3n}{2} > n$

$n \times n \times 2 = 2n^2 = O(n^2)$ ✓

5. i = 1; 2^K $n=16$
   while (i < = n)
   {
   i = i * 2;
   }

$i = 1 \times 2^0, 1 \times 2^1, 1 \times 2 \times 2 (2^2), 2^2 \times 2 (2^3), \ldots 2^K$

Time: $O(K)$

$: O(\log_2 n)$

for(i=1; i<=n; i=i*2) ✓
c=c+1;

$i = 2^K = n$

$K = \log_2 n$

6. i = n;
   while (i > 0)
   {
   i = i/2;
   }

$n=16$

$i = \boxed{16 ; 8 ; 4 ; 2 ; 1} ; 0$

$2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

$O(\log n)$

for(i=n; i>0; i=i/2) ✓
c=c+1;

1) for(i=1; i<=n; $\underline{\underline{i=i+1}}$) : O(n)
   c=c+1;
$n/1 \rightarrow O(n)$

Time : O(k)

2) for (i=1; i<=n; $\underline{\underline{i=i+2}}$) i=1 , 1+2 , 1+2+2 , 1+2+2+2
   c=c+1;
$n/2 \rightarrow O(n)$

$= \underline{\underline{1+2\times0}} ; \underline{1+2\times1} ; \underline{1+2\times2} ; \underline{1+2\times3} ; \dots$

$; 1+2\times K$

$1+2\times K = n$

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

3) for (i=1; i<=n; $\underline{\underline{i=i+5}}$) = 1 ; 1+5 ; 1+5+5
   c=c+1;
$n/5$

$= 1+5\times0 ; 1+5\times1 ; 1+5\times2 ; 1+5\times3 ; \dots$

$\underline{\underline{1+5\times K}} = n$

$K = \frac{n-1}{5} \sim \frac{n}{5}$

Generalized form:

for(i=1; i<=n; i=i+a) : n/a

1) for(i=1; i<=n; i=i*2) : $\log_2 n$

2) for (i=1; i<=n; i=i*3) : $\log_3 n$

$i = 1\times 3^0;\ 1\times 3^1;\ 1\times 3\times 3;\ 1\times 3^3, \ldots;\ 1\times 3^k$

1×3^2

Time: $O(k)$

$i = 1\times 3^k = n$

$\boxed{k = \log_3 n}$

$\boxed{\begin{array}{l}\text{for}(i=1; i<=n; i=i*a) \\ \quad : O(\log_a n)\end{array}}$

7.
```
k = 1; i = 1;
while (k < = n)
{
    i + + ;
    k = k + i;
}
```

i : 1 2 3 4 - - - t

K : 1 (1+2) (1+2+3) (1+2+3+4) (1+2+3+...+t)

$(1+2+3+\cdots+t) = n$

Time: $O(t)$

$\frac{t(t+1)}{2} \leq n$

$: O(\sqrt{n})$ ✓

$t^2 + \cancel{t} = \cancel{2}n$

$t^2 \sim n \quad \therefore t = \sqrt{n}$

8. for (i = 1; i < = n; ++i) : n $O(n * \log n)$

for (j = 1; j < n; j = 2 * j) : $\log n$

c + c + 1;

---

9. m = 2^n

for (i = 1; i < = n; + + i) : n : $n * \log_2 m$: $n * \log_2 2^n$

for (j = 1; j < = m; j = 2 * j) : $\log_2 m$: $n \times n = n^2$

c = c + 1 $\therefore O(n^2)$ ✓

$c = 0$

10. for i ← 1 to n $\quad$ ($n-1+1$)

    for j ← i to n $\quad$ ($n-i+1$)

    c= c+1;

Value of c =

$i = 1 \; ; \; 2 \; ; \; 3 \; \ldots \ldots \; ,$

$c = n \; ; \; (n + n-1) \; ; \; n + (n-1) + (n-2)$

$\boxed{n + (n-1) + (n-2) + \cdots + 1}$

Value of c & Total Iterations $= \frac{n(n+1)}{2} = O(n^2)$

11. $f(n) = \sum_{i=1}^{n} O(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} O(1)$

PYQ

$= O(n) * \sum_{i=1}^{n} 1$

$= O(n) * O(n)$

$= O(n^2)$ ✓

for i ← 1 to n
  for j ← 1 to n
    c=c+1;

$\boxed{f(n) = \sum_{i=1}^{n} 1 = O(n)}$

for i ← 1 to n
  c=c+1

Slide 10

$$f(n) = \sum_{i=1}^{K} O(n) \qquad (K > 0)$$

$$= O(n) \cdot \sum_{i=1}^{K} 1$$

$$= O(n) * K$$

$$= O(n)$$

12.
```
i = n;
while (i > 0)
{
    j = 1;
    while (j < = n)
    {
        j = 2 * j;
    }
    i = i/2;
}
```

$n = 16$

while: $\log_2 n$

1^{st} It	2^{nd} It	3^{rd}	$\log n$
$\log_2 n$	$\log n$	$\log n$	$\log n$

Time: $(\log n * \log n)$

$O\left((\log_2 n)^2\right) = O(\log_2^2 n)$

13. int fun (int n) : 2M

PYQ

```
{
    int i, j, p, q = 0;
    for (i = 1; i < = n ; + + i) : n
    {
        1. p = 0; q = 0;
        2. for (j = n; j > 1; j = j/2) : log n
               + + p;
        3. for (k = 1; k < p; k = k * 2)
               + + q;
    }
    return (q);
}
```

Total Time

n :

$n \cdot \log n$:

$n \cdot \log \log n$:

$q = \hookrightarrow \log P + \log P$

Q) The value of q returned by the function : $O(n * \log \log n)$

$q = n * \log P$

$P = \log n$

$= n * \log_2(\log_2 n)$

Q2) Time Complexity : $O(n \cdot \log_2 n)$

Value of $q = O(\log \log n)$

Time $= O(n \cdot \log_2 n)$

14. for (i = 1; i < = n; + + i) : n

{

1. j = 1; $\longrightarrow n$

$j = n;$

2. while (j < = n)

j = 2 * j; $\Big\}\ \log n \longrightarrow n * \log n$

3. for (k = 1; k < = n; + + k)

c = c + 1; $\Big\}\ n \longrightarrow n^2$

}

$n^2 + n\log n + n$

$O(n^2)$ ✓

Slide 13

15. $n = 2^{2^k}$

```
for (i = 1; i < = n; + + i)
{
    j = 2;
    while (j < = n)
    {
        j = j * j;
        pf("*");
    }
}
```

'*' : $n(k+1)$

: $n * \log\log n + n$

$K = 2$

$n = 2^4 = 16$

$i = 1$

$j = 2, 4, 16$

(* * *)

$i = 2$

$j = 2, 4, 16 \quad [2^{2^0}; 2^{2^1}; 2^{2^2}]$

(* * *)

$i = n$

$j = 2, 4, 16$

(* * *)

$K = 3$

$n = 2^8 = 256$

$i = 1$

$j = 2, 4, 16, 256 \quad [2^{2^0}, 2^{2^1}; 2^{2^2}; 2^{2^3}]$

* * * *

Per It: $O(K)$

Total Its: $O(n * K)$

: $O(n * \log\log n)$ ✓

$n = 2^{2^K}$

$\Rightarrow K = \log_2 \log n$

Slide 14

1) for (i=2; i<=n; $\underline{i=i*i}$)
c=c+1;

Time: $O(\log \log n)$

$\underline{n=16}$

$i = 2 ; 4 ; 16$

$= 2^{2^0} ; 2^{2^1} ; 2^{2^2}$

K = last It

$i = 2^{2^K} = n$

$\boxed{K = \log \log n}$

---

2) for (i=1; i*i<=n; i++)
c=c+1;

Time:

$i^2 <= n$ ✓

$i^2 = n \Rightarrow i = \sqrt{n}$

Time: $O(i) : O(\sqrt{n})$

3) for ($i = n$; $i >= 2$; $i = sqrt(i)$) $\Longrightarrow O(\log \log n)$

$c = c + 1;$

$n = 256$

$i = 256$; $(256)^{1/2}$; $\left[(256)^{1/2}\right]^{1/2}$; $\left[\left[(256)^{1/2}\right]^{1/2}\right]^{1/2} = \left(n^{1/2^K}\right) = 2$

$256 \quad 16 \quad 4 \quad 2$

$2^{2^3} \quad 2^{2^2} \quad 2^{2^1} \quad 2^{2^0}$

$K = \log \log n$

$2^{2^K} \leq n \Longrightarrow 2^{2^K} = n$

$\Longrightarrow K = \log \log n$

16. A: Array [1 ......n] of binary; (0/1)

** PYQ

$\boxed{f(m) = \theta(m);}$

count: integer;

count = 1;

for i = 1 to n

{

if (A[i] = = 1) count + +; $\frac{n}{2}$ $\hookrightarrow n/2$

else

{

f(count);

~~count = 1;~~

}

}

W.C

$$\left(\frac{n}{2} + \frac{n}{2} \times \frac{n}{2}\right)$$

$O(n^2)$

$O(n^2)$

Time - Complexity :

(i) Best Case :

(ii) Worst Case :

$\Big\}$ $O(n)$ ✓

1) $A[i] = 1, \quad i = 1, n$ Time : $O(n)$ | $O(n)$

2) $A[i] = 0, \quad i = 1, n$ Time : $O(n)$ | $O(n)$

3) $A[i] = 1, \quad i = 1, n-1$
$= 0, \quad i = n$

A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Time : $(n-1) + (n-1) + 1$
: $2n - 1 = O(n)$ ✓ | $O(n)$

4) $A[i] = 1, \quad i = 1, n/2$
$= 0, \quad i = (n/2 + 1), n$

$n/2$

A: | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

$n/2$ $\quad (n/2 - 1)$

$$: \frac{n}{2} + \left(\frac{n}{2} + 1\right) + \left(\frac{n}{2} - 1\right) O(1) = O(n)$$

Slide 15

H|W

Consider the following function:

```
int unknown (int n)
  {
    int i, j, k = 0;
    for (i = n/2; i < = n; i++)
        for(j = 2; j < = n; j = j*2)
            k = k + n/2;
    return(k);
  }
```

The return value of the function is ___.

[GATE-2013: 2M]

(A) $\Theta(n^2)$

(B) $\Theta(n^2 \log n)$

(C) $\Theta(n^3)$

(D) $\Theta(n^3 \log n)$

Consider the following C function.

H/W

```
int fun (int n)
{ int i, j;
  for (i = 1; i <= n; i ++){
     for (j = 1; j < n; j += i)
    {
     printf("%d %d", i, j);
    }
  }
}
```

Time complexity of fun in terms of Θ notation is [GATE-2017: 2M]

(A) $\Theta(n\sqrt{n})$

(B) $\Theta(n^2)$

(C) $\Theta(n \log n)$

(D) $\Theta(n^2 \log n)$

MSQ

<(A ; D)>

Consider functions Function_1 and Function_2 expressed in pseudocode as follows:

$n=16$

1 2 3 … K

$\frac{n}{2^0}$ $\frac{n}{2}$ $\frac{n}{4}$ $\frac{n}{2^K}$

$O(n)$

Function_1

```
while n > 1 do
    for i = 1 to n do
        x = x + 1;
    end for
    n = ⌊n/2⌋;
end while
```

Function_2

```
for i = 1 to 100 * n
do
    x = x + 1;
end for
```

Time : $100*n = O(n)$

$T(n) = 100 \times n$

Let $f_1(n)$ and $f_2(n)$ denote the number of times the statement "x = x + 1" is executed in Function_1 and Function_2, respectively.

Which of the following statement is/are TRUE? [GATE-2023-CS:2M]

$n = 2^K$

$K = \log_2 n$

(A) $f_1(n) \in \Theta(f_2(n))$ ✓

(B) $f_1(n) \in o(f_2(n))$ ✗

(C) $f_1(n) \in \omega(f_2(n))$ ✗

(D) $f_1(n) \in O(n)$ ✓

Total time $= n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots + \frac{n}{2^K}$

$= n\left[\sum_{i=1}^{K} \frac{1}{2^i}\right] = n\left(1 - \frac{1}{2^K}\right)$

$= n - 1 = O(n)$

SPACE - Complexity

Memory

Program / Algorithm

Instrn's

Data

Input

Space Complexity

Additional Space

Auxiliary Space

```
void SWAP(int a, int b)   Inputs
{
    int temp;   2B = Constant : O(1)
    1. temp = a;
    2. a = b;
    3. b = temp;
}
```

Algorithm Sum(A, n)

{

integer n, A[n], $\underline{i}$, $\underline{Sum}$ — 2B, 2B

1. Sum = 0;

2. for i ⟵ 1 to n

Sum = Sum + A[n];

}

1) Time : O(n) ✓

2) Space : O(1)

Input : n ; A[n]

Aux workspace : 4 Bytes

O(1)

Topic : Space Complexity

We define the space used by an algorithm to be the number of memory calls (or words) needed to carry out the computational steps required to solve an instance of the problem excluding the space allocated to hold the input. In other words, it is only the work space required by the algorithm.

Topic : Space Complexity

All definitions of order of growths and asymptotic bounds pertaining to time complexity carry over to space complexity. It is clear that the work space cannot exceed the running time of an algorithm, as writing into each memory call requires at least a constant amount of time.

Thus, if we let $T(n)$ and $S(n)$ denote, respectively, the time and space complexities of an algorithm, then $\boxed{S(n) = O(T(n))}$.

Topic : Space Complexity

Example: In Algorithm LINEARSEARCH, only one memory cell is used to hold the result of the search. If we add local variables, e.g. for looping, we conclude that the amount of space needed is $\Theta(1)$. This is also the case in algorithms BINARYSEARCH, SELECTION SORT and INSERTION SORT.

```
Algo LS(A, n, x)
{
  int n, A[n], x, i        2B
  for i <- 1 to n
  {
     if (x = A[i]) {
         print(i);
         return;
     }
  }
  return(0);
}
```

Time : $O(n)$

Space : $O(1)$

Topic : Space Complexity

Example: In Algorithm MERGE for merging two sorted arrays, we need an auxiliary amount of storage whose size is exactly that of the input, namely n (Recall that n is the size of the array A[p..r|). Consequently, its space complexity is $\Theta(n)$.

```
integer n, A[n];
Algorithm RSum(A, n)
{
    if (n = 1) return (A[n]);
    else
    {
        return (A[n] + RSum(A, n-1));
    }
}
```

Space: $O(n)$

n - records

Aux w.s

Sum of elems of Array

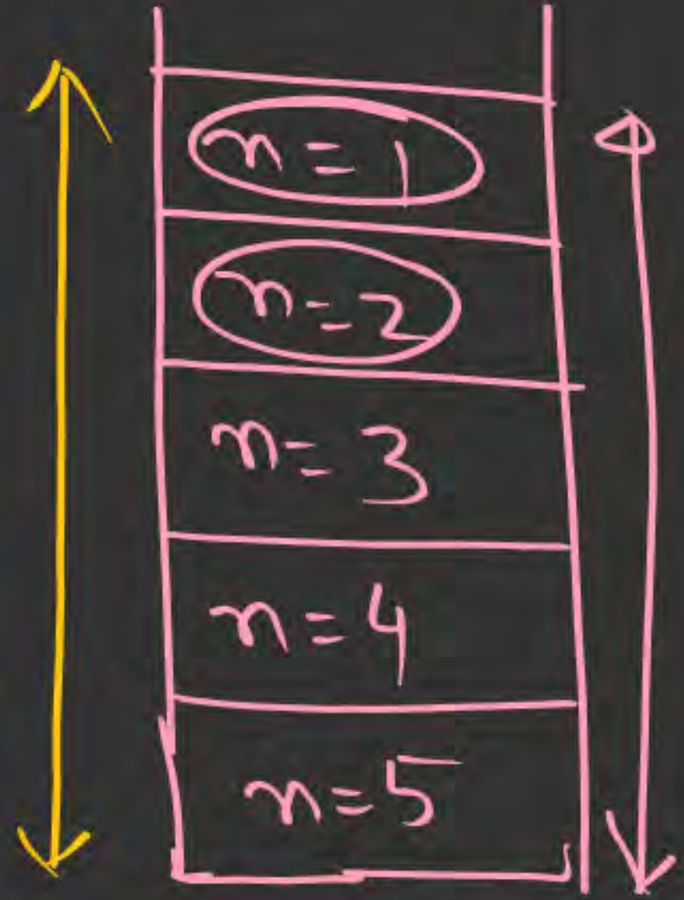
Time : $O(n)$

$T(n) = T(n-1) + c$

Space-Complexity :

$n = 5$

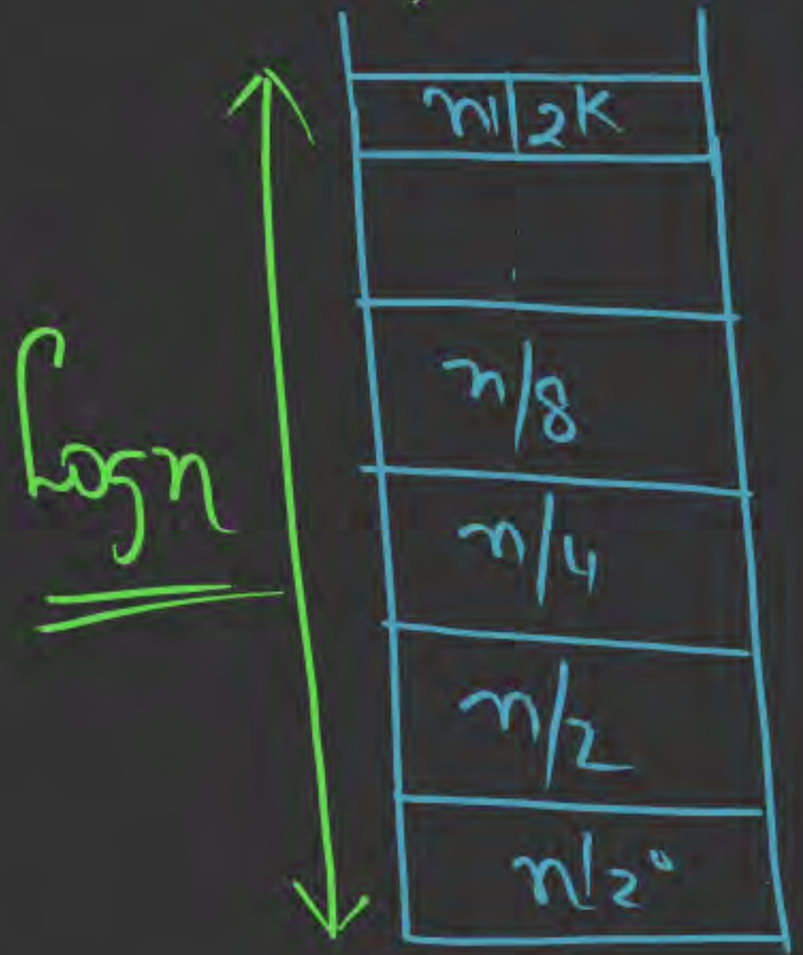
Size of Stack



No. of records

STACK(AUX)

```
Algorithm DO-it(n)
{
    if (n == 1) return;
    else
    {
        DO-it(n/2);
    }
}
```



Time : $O(\log n)$

$T(n) = T(n/2) + C$

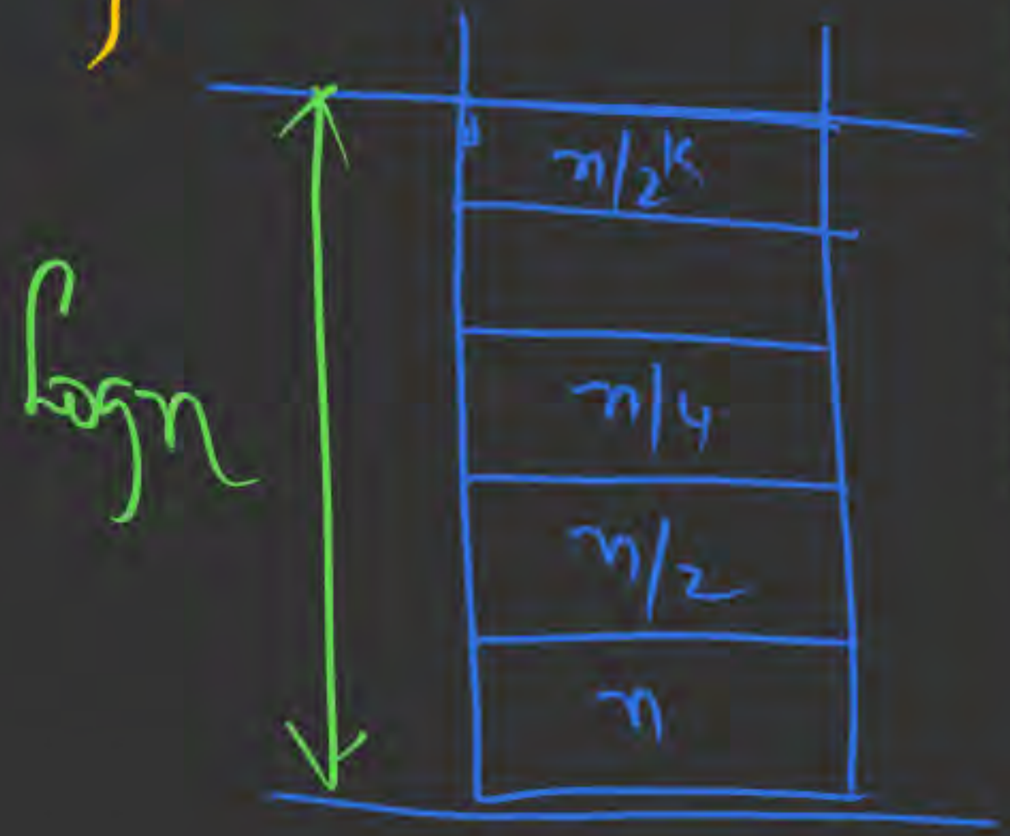
Space : $O(\log n)$

$1, 2, 4, 8, \cdots 2^K$

$2^K = n$

$\Rightarrow K = \log n$

```
Algorithm Test(n)
{
    if (n=1) return;
    else
    {
        Test(n/2);
        Test(n/2);
    }
}
```

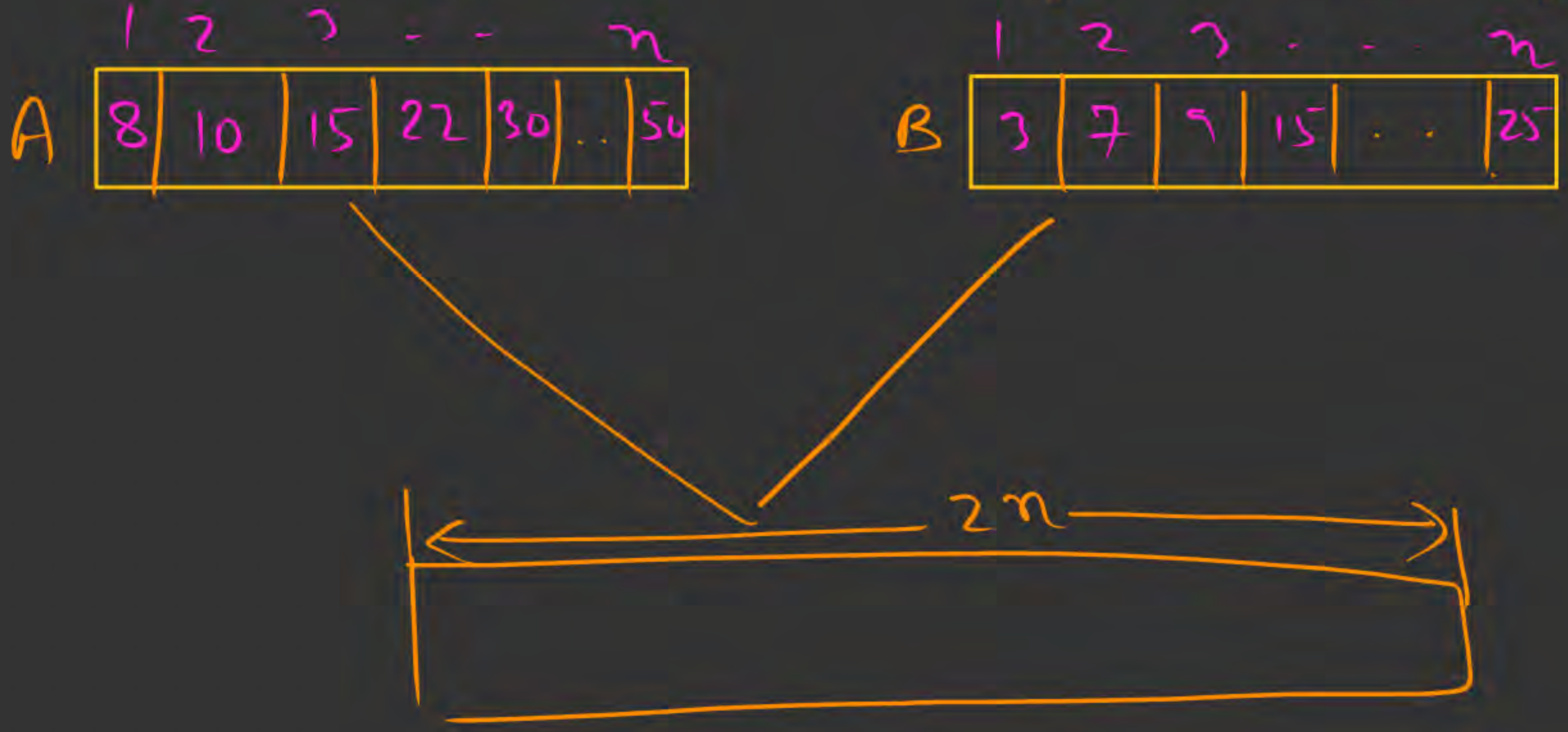


Time : $O(n)$

$$T(n) = 2T(n/2) + c$$

Space : $O(\log n)$

Q) Given Two Sorted arrays of Size 'n' each, The Space Complexity to merge The Two arrays is O(   )



THANK - YOU