#### MANSI DAHIYA

#### 190121136

DS(Vth Sem)

# Design and Analysis of Algorithms

## Tutorial\_1

# Q1: What do you understand by Asymptotic notations? Define different Asymptotic notation with examples.

**Ans:** Asymptotic Notations. Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

Big-O Notation (O-notation) Big-O notation represents the upper bound of the running time of an algorithm. Thus, it gives the worst-case complexity of an algorithm.

f(n) = O(g(n)) iff there are two positive constants c and n0 such that  $|f(n)| \le c * |g(n)|$  for all  $n \ge n0$ 

Example:

$$n^2 + n = O(n^3)$$
  
Here, we have  
 $f(n) = n^2 + n$ ,  
 $g(n) = n^3$ 

Omega Notation ( $\Omega$ -notation) Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best-case complexity of an algorithm.

 $f(n) = \Omega(g(n))$  if there are two positive constants c and n0 such that  $|f(n)| \ge c*|g(n)|$  for all  $n \ge n0$ .

1

Example:

• 
$$n^3 + 4n^2 = \Omega(n^2)$$

• Here, we have  

$$f(n) = n^3 + 4n^2,$$

$$g(n) = n^2$$

Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analysing the average-case complexity of an algorithm.

```
f(n) = \Theta(g(n)) if there are three positive constants c1, c2 and n0.
```

#### Example:

```
\begin{array}{l} n^2 + 5n + 7 = \Theta(n^2) \; Where \; f(n) = n^2 + 5n + 7 \\ g(n) = n^2 \\ When \; n \geq 1, \\ n^2 + 5n + 7 \leq n^2 + 5n^2 + 7n^2 \leq 13n^2 \\ When \; n \geq 0, \\ n^2 \leq n^2 + 5n + 7 \\ Thus, \; when \; n \geq 1 \\ 1n^2 \leq n^2 + 5n + 7 \leq 13n^2 \\ Thus, \; we \; have \; shown \; that \; n^2 + 5n + 7 = \Theta(n^2) \; (by \; definition \; of \; Big-\Theta, \; with \; n0 = 1, \\ c1 = 1, \; and \; c2 = 13.) \end{array}
```

## Q2. What should be time complexity of -

```
for (i=1 to n)
{
i=i*2;
}
Ans: T(n) = O(n)
```

## Q3. $T(n) = {3T(n-1) \text{ if } n>0, \text{ otherwise } 1}$

```
Ans: T(n) = 3T(n-1)

= 3(3T(n-2))

= 3^2T(n-2)

= 3^3T(n-3)

...

= 3^nT(n-n)

= 3^nT(0)
```

```
= 3^{n}
```

This clearly shows that the complexity of this function is  $O(3^n)$ .

## Q4. $T(n) = \{2T(n-1)-1 \text{ if } n>0, \text{ otherwise } 1\}$

Ans: 
$$T(n) = 2T(n-1) - 1$$
  
=  $2(2T(n-2)-1)-1$   
=  $22(T(n-2)) - 2 - 1$   
=  $22(2T(n-3)-1) - 2 - 1$   
=  $23T(n-3) - 22 - 21 - 20$   
.....  
=  $2nT(n-n) - 2n-1 - 2n-2 - 2n-3$   
.....  $22 - 21 - 20$   
=  $2n - 2n-1 - 2n-2 - 2n-3$   
.....  $22 - 21 - 20$   
=  $2n - (2n-1)$   
 $T(n) = 1$ 

## Q5. What should be time complexity of -

```
int i=1, s=1;
while(s<=n)
{ i++; s=s+i;
printf("#");
}
Ans: I will go on i = 1,2,3-----k
s = 3,6,10,15----</pre>
```

Time Complexity is O(1).

```
while loop will terminate if: 1+2+3+ -----+k
[k(k+1)/2] > n
So, k = O(sqrt n)
T(n) = O(sqrt n)
Q6. Time complexity of -
void function(int n)
{
int i, count= 0;
for (i=1; i*i<=n; i++)
count++
}
Ans: As the statement is valid for n/10 terms
So, T(n) = O(n/10)
ie: T(n) = O(n)
Q7. Time complexity of -
void function(int n){
int i, j, k, count=0;
for(i=n/2; i<=n; i++)
for(j=1; j<=n; j=j*2)
for(k=1; k<=n; k=k*2)
count++
}
Ans: T(n) = O(n\log^2 n)
Q8. Time complexity of -
function(int n){
if(n==1) return;
for(i=1 to n){
```

```
for(j=1 to n)
printf("*");
}
}
function(n-3);
}
Ans: T(n) = O(n)
Q9. Time complexity of -
void function(int n){
for(i=1 to n){
for(j=1; j<=n; j=j+i)
printf("*")
}
}
Ans: T(n) = O(n^2)
As n times for I loop and n times for j loop.
```

Q10. For the functions,  $n^k$  and  $c^n$ , what is the asymptotic relationship between these functions? Assume that k >= 1 and c > 1 are constants. Find out the value of c and n0 for which relation holds.

```
Ans: n^k is O(c^n)

Ie: n^k \le c^n

Taking log on both sides we get;

K \log n \le n \log c
```

In this case the relation will for values of k and c to be n.



	Date
	1+(n) k C C-1.
	$\frac{c}{a}$ $\frac{h}{a}$ $\frac{k}{a}$ $\frac{1}{2}$
200	$C7$ , $2+$ $n_0$ $\alpha=1.5$ .
	C 7/30+1. 3 C 7/4.
12	It is same as In: Ans = O(In).
32	T(n) = T(n-1) + T(n-2) + 1.  Solving using tree method.
	propried n-1 n-2.
. (*)	n-2 $n-3$ $n-4$
7.5 0	T. ( 3) 1+2+4+ 2n.
	This is $a \cdot 67$ . $a = 1$ , $a = 2$ .
	$n = \alpha(x^{\text{terms}} - 1) = 2^{n+1} - 1$

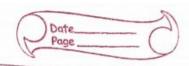


Ans	T.C = O(an+1) = O(an.2)
13	Space (omplexity = O(n).  O(log (logn)) , we get this facom  O(n <sup>3</sup> )
	O(nlogn) q we get this from Q. q.
14	Cn2. 1 4 95
(25)0	T(n/4) 00 T(n/2).
. 1-	On further breaking this down
	$T(n^2/16)$ $cn^2/4$ .
	T(n 16) $T(n 8)$ $T(n 8)$ $T(n 4)$ . $T(n) = Cn^2 + 3n^2/16. + 25n^2/256$
	This is a GP with ratio 5/16
	$\frac{n^2}{1-5}$ $\frac{7\cdot 1}{16}$ $\frac{7\cdot 1}{16}$



	Page C
15	This is same as question 9. O(nlogn)
10	O(nlogn)
16	1 3) 2, 2k (2k) k 2k logk (togn)
1	The last term 2 roge (rog n)
	equal to no need have
	equal to n nee have.  2 klog & (vogen) = 2 logen 7 n.
	= 2 mg 7 n.
18	0) 100 ( 100 (100 )
	100 stag (log n) < log n < In < n
	(n) < nlogn > n2 an < 4n
	a) 100 < log (log n) < log n < Jn < n < log (n1) < n log n ≥ n² < 2n < yn < n! < 2²n
b)	
100	(logn) < Nlogn < log(n) < log2n
	along n < 2n < 4n < log(n) <
	$1 \leq \log(\log n) \leq \log(n) \leq \log$
c)	10.5. 7
	10 Stogen ( log 2n ( Sn < log (n1)
	$\leq n \log_6(n) \leq n \log_2 n \leq 8n^2 \leq 7n^3$
	96 $< log_8 n < log_2 n < sn < log(n1)$ $< n log_6(n) < n log_2 n < 8n^2 < 7n^3$ < n   < 80n.
19	
2	int ls (int a[], int n, int d)
	for (1=0; )(n-1;i++)
	3
	y (a(i) = = d)
	retien i;
	else if (a[i] 7d)  Peturn -1;
	return -1;
	3
	9

Ans	Time Complexity: Best 3 0(1)
- (1)	Time Complexity: Best 3 O(1)
, 12	1 1 1 1 1 1 1 1
100 /x1	Space Complexity > O(1)
20	Salt Comments of the Comments
1	void insertion_s (int apprint ).
NEN	interior domation
1.811	int 1, j, temp; for (1=1; icn; iee)
(Chair	temp = a[i];
550	1=111-1: (1100) 000
*	while I j 7 = 0 kka[j] 7 temp)
0	a [jei] = a[j];
1 m p	1 10 0 10 10 10 10 10 10 10 10 10 10 10
	a grand of the second of the s
	a [j+1] = temp;
(h)	Ini 3, to 130 Jone 126 to 19
	This is the pseudo code for alexatrics
	This is the pseudo code for iterative insertion sort.
	void is (int a[], int n)
	( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )
	return.
	i-s (a, n-1).
	ent l = a a n-17.
	int j = n-2;



a[j+1] = a[j];	nohile (j 7=0 kk	arjj	72).	24
i j; in a property	T. Control of the con			14.
2. 6 100 6 100 0 000110	alythin = a	3)0	and a	0
4	24	1577	200110	7

An online algo is the one that process its input piece by piece in a serial fashion ie in the order that the input is fed to the algorithms without having the entire input avoida - ble from the beginning.

Insertion cont considers one input element per iteration and produces a partial solution without considering future elements. Thus, oit is online of

		Frank 1 7-11	1		9
21	Sorting Algo	Time	2 Comp	lexity	Space Complexity
-6	0 0	Best	Avg	Worst	The company
	Bubble	0(n2)	0(2)	O(n2)	0(1)
	Solection	0(n2)	O(n2)	D(n2)	0(1)
	mertiane	0(n)	0(02)	0(n2)	0(1)
	Merge	O(nlogn)	O(nlogn)	Oldogn)	0(n)
		O(nlogn)	O(nlogn)	Oln2)	0(n)
	Heap	O(nlogn)	O(nlogn)	Oldogn).	
			J /	0	



033	Stable sont 3 insertion, merges bubble sont
	bubble sort
	online sort & insertion sort
	inplace sort & bubble, selection
	enline sont à insertion sont inplace sont à bubble, selection insertion, heap sont.
	Herative pseudo code for binary search.
250	
183	wint b solington of 7 in 0 in
1000	int b_s (int a [], int l, int r, int x)
2005	sell and the de the sell.
1,100	while ( l < = g)
	S' principal of the
	S = 1 - (r+1)/2
<b>ブロ</b> 。	if (a[m] == 2)
0 23.5	if (a[m] == 2)
11. 12	if (a(m) (xx)
Sandy N	Contractor of the contractor o
	J = W-61.
diri-1	else die
0.14	90 = m-100
1 1 1	1 ( ( ( ( ( ( ( ( ( ( ( ( ( ( ( ( ( ( (
	2. return -1)
·	1000 (000 000000
-	Constitute Company of Constitutes
1	And complexity & Bust case 0(1)
	Time complexity & But case 0(1). And, worst 10 (log2n)
	Space Complexity ? O(1)
	11



	Page				
9	Recursive binary search.				
	Recursive binary search.  int be (int a P], int l, int n, intx)				
	1				
	4 (27=1)				
	mid = (ler)/2;				
	1				
	if (a[mid]==x)				
	schoon mid '				
,	else if (a[mid] 7x)				
	seturn b s (a, l, mid-1, x);				
	else				
	return b-s (a, mid+1, a, x);				
	3				
	return -1;				
	3.				
	- A . M '				
	Time Complexity 3 Best D(1), Aug, Worst				
	O(log2h)				
	O(log2n)  Space comploxity 3 Best O(1), Aug, Worst O(log2n).				
2	T(n) = T (n/2) + 1. is the securrence relation for binary recurrence search				
-0	appation los binary recursive search				
	Taxas o				