DAA ASSIGNMENT → 1

NAME → Kashish Mirza

COURSE → B.Tech

BRANCH → C.S.E

SECTION → A

U. Roll. No → 1961083

Ques → 1 → What do you understand by Asymptotic notations. Define different Asymptotic notations with example?

Ans → **Asymptotic NOTATION**

→ We use Asymptotic notation to represent order of growth of function.

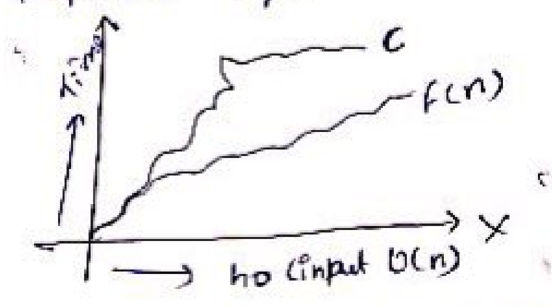
TYPES

→ O notation.
→ Θ notation.
→ Ω notation.

① **BIG O notation**

→ This is used to represent upper bound of a function.

→ Graphical representation

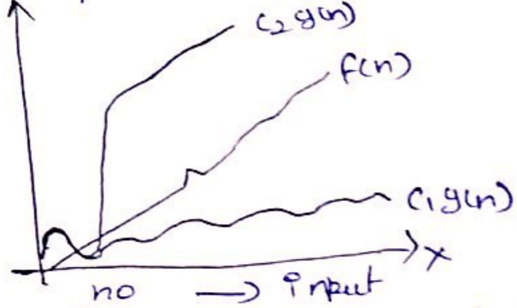


$O(gn) = \{ f(n) : 0 \le f(n) \le c(g(n))$ where c is a positive constant $\forall\ n \ge n_0 \}$.

② **Θ notation**

→ This is used to represent both upper as well as lower bound.

→ Graphical representation.



→ $\Theta(g n) = \{ f(n) : 0 \le c_1 g n \le f(n) \le c_2 g n \}$ where c_1 & c_2 are tive constant $\forall\ n \ge n_0 \}$.

→ <u>Big</u> $\underline{\underline{\Omega}}$

→ This is used to represent the lower bound of the function.

→ Graphical representation.

→ $\Omega(g n) = \{ f(n) : \le c\, g(n) \le f(n)$ where c is positive constant $\forall\ n \ge n_0 \}$.

Ans→2→ for(i = 1 to n)
{
(i = i*2);
}

1, 2, 4, 8, ... n

$T(n) = O(\log_2 n)$

Ans→3→ $T(n) = \begin{cases} 3T(n-1) & n>0 \\ 1 & n=0 \end{cases}$

$T(n) = 3T(n-1)$ — ①

$T(n-1) = 3T(n-2)$ —

$T(n) = 9T(n-2)$ — ②

$T(n) = 3^3 T(n-3)$ — ③

$T(n) = 3^k T(n-k)$ — ④

for $T(n-k) = T(0)$

$n-k = 0$

$n = k.$

$T(n) = 3^n T(0)$

$T(n) = 3^n.$

$T(n) = \Theta(3^n)$.

Ans→ 4→ $T(n) = \begin{cases} 2T(n-1) - 1 & , n>0 \\ 1 & , n=0 \end{cases}$

$T(n) = 2T(n-1) - 1$ — ①

$T(n-1) = 2T(n-2) - 1.$

$T(n) = 4T(n-2) - 1 - 2$ —(2)

$T(n) = 8T(n-3) - (1+2+4)$ —(3)

$T(n) = 2^k\, T(n-k) - [1+2+4+ \cdots 2^{k-1}]$

$T(n-k) = T(0)$

$n = k.$

$T(n) = 2^n\, T(0) - \underbrace{[1+2+4+ \cdots\cdots]}_{k \text{ terms}}$

Its an A.P.

$a = 1$

$r = 2$

$T(n) = 2^n - \left(\frac{1\,(2^{n-1} - 1)}{2-1}\right)$

$T(n) = 2^n - 2^n + 1.$

$T(n) = 1.$

$T(n) = \Theta(1).$

Ans-) 5-) int i = 1 , S = 1 ;

while (S ≤ n)

{

i++ ;

S = S + i

Print f ("#") ;

}

$\underbrace{1, 3, 6, 10, 15, \cdots n}_{k \text{ terms}}$

Its kth term is $\frac{k(k+1)}{2} = n.$

$k = \sqrt{n}.$

$T(n) = O(\sqrt{n})$.

Ans→6→ void function (int n) {
int i , count = 0 ;
for (int i = 1 ; i * i < n ; i++)
count ++
}
$T(n) = O(\sqrt{n})$

Ans→7→ $T(n) = O(n * \log_2 n * \log_2 n)$

$T(n) = O(n * (\log_2 n)^2)$

$T(n) = O(n (\log n)^2)$

Ans→8→ function (int n) { $T(n)$
if(n==1) return ;
for(i=1 to n)
{
for(j = 1 to n) {
{
printf("*");
}
}
function(n-3) ; $T(n-3)$
}

$T(n) = T(n-3) + n^2$ —①

$T(n-1) = T(n-4) + (n-1)^2$.

$T(n) = T(n-4) + n^2 + (n-1)^2$

$T(n) = T(n-5) + n^2 + (n-1)^2 + (n-2)^2$

$T(n) = T(n-k) + (n^2 + (n-1)^2 + (n-2)^2 + ---)$
(k-2) terms.

for $T(n-k) = 1$
$k = n-1$

Scanned with CamScanner

$T(n) = T(1) + (n^2 + (n-1)^2 + (n-2)^2 + \cdots\cdots)$
$(n-3)$ terms

$T(n) = T(1) + (4^2 + 5^2 + \cdots\cdots n^2)$

$T(n) = T(1) + \left(\frac{(n-3)(n-2)(2n-5)}{6}\right)$

$T(n) = 2 + \left(\frac{2n^3 + \cdots}{6}\right)$

$T(n) = n^3.$

$T(n) = O(n^3)$

Ans → 9 → void function (int n) {
for (i = 1 to n)
{
for (j = 1 ; j ≤ n ; j = j + i)
printf ("*");
}

$i=1$	n times	
$i=2$	$1, 3, 5, \cdots n$	n
$i=3$	$1, 4, 7 \cdots n$	$n/2$
⋮		
$i=n$	0	

$T(n) = \left[n + \frac{n}{2} + \frac{n}{3} \cdots \text{n-times}\right]$

$T(n) = O(n \log n)$

Ans → 10 → for the function n^k and a^n, what is the relation.
$k \geq 1$ & $a > 1$
relation is n^k is $O(c^n)$.

Scanned with CamScanner

Ans→11→ void fun (int n)

```
{
  int j=1 ; i=0;
  while( i<n )
   {
     i = i+j ;
     j++ ;
   }
}
```

$\underbrace{0, 3, 6, 10, 15 \text{ ------ } n}$

so for this series is k terms

k^{th} term is $\frac{k(k+1)}{2}$.

$$n = \frac{k^2+k}{2}.$$

$$k \simeq \sqrt{n}$$

$$T = O(\sqrt{n})$$

Ans→ 12→ Recurrence relation of fabonacci series is

$$T(n) = T(n-1) + T(n-2) + 1$$

$$T(n) = 2T(n-2) + 1$$

$$T(n) = 4T(n-4) + 3$$

$$T(n) = 8T(n-6) + 7$$

$$T(n) = 16T(n-8) + 15$$

$$T(n) = 2^k T(n-2k) + (2^k - 1)$$

for $T(n-2k) = T(0)$

$$n = 2k$$

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

$$T(n) = 2^{n/2} \; T(0) + (2^{n/2} - 1)$$

$$T(n) = 2^n - 1$$

$$T(n) = O(2^n)$$

Hence space complexity of fabinucci series is O(n) as it depends on height of recursive tree & it is equal to n in fabinacci series.

Ans→ 13 →O n (log n)

```
void fun( for(int j = 0; j < n; j++)
        {
          for ( int i=0 ; i<n ; i = i*2)
          {
           printf(" * ");
          }
        }
  }
void main ()
{
  fun ();
}.
```

(11) n3

```
# include <stdio.h>
void main()
{
 int n;
  cin >> n;
  for (int i=0; i<n; i++)
   {
    For(int j = 0; i<n ; j++)
     {
      for(int k=0;  k<n ; k++)
       {
         k++
       }
     }
   }
}
```

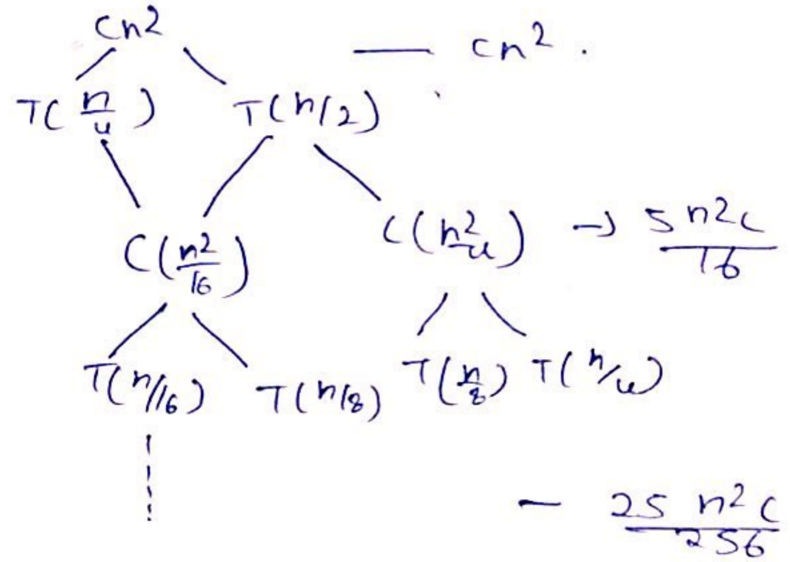
→ log (logn)

```
# include <bit/stdc++.h>
void fun(int n)
{
 if( n==2 )
  return 1;
 else
  fun(sqrt(n));
}
void main()
{
 fun(100);
}
```

Ans)14) $T(n) = T(n/4) + T(n/2) + cn^2$

$T(1) = c$

$T(0) = 0$



$T(n)$ = constant for each level

$T(n) = cn^2 + \frac{5cn^2}{16} + \frac{25(n^2)}{256} + \text{---}$

it is a G.P with $a = n^2$

$r = \frac{5}{16}$

so sum of G.P

$T(n) = cn^2 / (1 - \frac{5}{16}) = \frac{16cn^2}{11} = \frac{16cn^2}{11}$

$T(n) = O(n^2)$

Ans-)15-) for (int i to n)

```
{
for ( int j = 1 ; j < n ; j += i )
  {
   // O(1) .
  }
}
```

$$\underbrace{n , \frac{n}{2} , \frac{n}{3} , \frac{n}{4} , \frac{n}{5} , \text{------} 1}_{k \text{ times}}$$

$k = \log_2 n$

$n \cdot \left(1 , \frac{1}{2} , \frac{1}{3} , \frac{1}{4} , \text{---} \frac{1}{n} \right)$

$(n (\log n))$

$T(n) = O(n \log n)$.

Ans-)16-) for(int i = 2 ; i <= n ; i = Pow(i, k))

```
{
 // O(1)
}
```

$2 , 2^k , 2^{k^2} , 2^{k^3} , \text{---} n$

it G.P $a = 2$

$r = 2^k$

k^{th} term $= ar^{k-1}$

$n = 2(2^k)^{k-1}$

let $k^{(k-1)} = x$

$k \log_k k = \log x$ —

$k = \log x$ — ①

$n = 2^x$

$\log_2 n = x \log_2 2$

$x = \log_2 n$

$\log n = \log(\log n)$

from — ①.

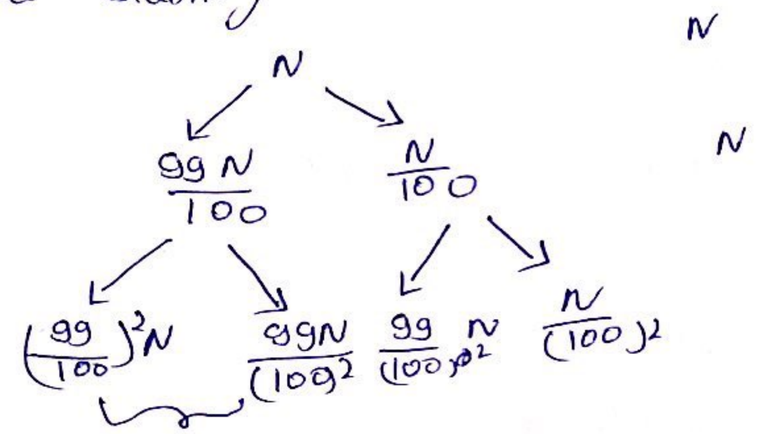
$k = \log(\log(n))$

$T(n) = O(\log(\log n))$.

Ans.17 → hence pivot is divided in 99% & 1%
so

$$T(n) = T\left(\frac{99}{100}N\right) + T\left(\frac{N}{100}\right) + N$$

Now as here we can use . 2 extreme of a tree
where starting point is N



$$N\left(\frac{99\times 99}{100\times 100} + \frac{99(1)}{100\times 100}\right) + \frac{\cancel{100}}{100\times \cancel{100}} N$$

$$= \frac{99}{100}N + \frac{N}{100}$$

$$= N$$

$$= N$$

so cost of each level is N only.

Total cost = height * cost of each level

so for 1st stream − $N, \frac{99}{100}N, \left(\frac{99}{100}\right)^2 N, \text{----}$

$$\left(\frac{99}{100}\right)^{h-1} N = 1$$

$$\left(\frac{99}{100}\right)^{h-1} = \frac{1}{N}$$

$$N = \left(\frac{100}{99}\right)^{h-1}$$

$$\log N = h \log (1)$$

$$h = \log N \quad \text{or}$$

$$h = \frac{\log N}{\log\left(\frac{100}{99}\right)} + 1$$

Scanned with CamScanner

height of 2^{nd} stream

$$N, \frac{N}{100}, \frac{N}{100^2}, \frac{N}{(100)^3}, ---- 1$$

$$N\left(\frac{1}{100}\right)^{h-1} = 1$$

$$N = (100)^{h-1}$$

$$(h-1)\log 100 = \log N$$

$$h = \frac{\log N}{\log 100} + 1 \quad \& \quad h = \log N \text{ (approx)}$$

$$T(n) = O(N \log N)$$

so time complexity is $O(N \log N)$

height of both extre is $\frac{\log N}{\log 100} + 1$ of $\left(\frac{1}{100}\right)$

and $\frac{\log N}{\log\left(\frac{100}{99}\right)} + 1$ of $\left(\frac{99}{100}\right)$

so we can conclude that if division is done more then height of tree will be more & when devision ratio is less ten height is less.

Ans→18→ⓐ n, n!, logn, loglogn, root(n), nlogn, 2n, 2^{2n}, 4^n, n^2, 100

Ans→ $O[100] < (\log\log n) < O(\log n) < O(\sqrt{n}) < O(n) < O(n\log n) < O(n^2) < O(2^n) < O(2^{2n}) < O(4^n)$

ⓑ $2(2^n)$, 4n, 2n, 1, log(n), log(log(n)), $\sqrt{\log(n)}$, log2n, 2log n, n, log(n!), n!, n2, nlog(n)

Ans→ $O(1) < O(\log(\log n)) < O(\log(n!)) \cancel{< O(2n) <} < O\log(2n) < O(2\log n) < O(n) < O(n\log(n)) < O(\log(n!)) < O(2n) < O(4n) < O(n^2) < O(n!) < O(2(2^n))$.

Scanned with CamScanner

© 2^{2n}, $\log_2 n$, $n \log_6(n)$, $n \log_2(n)$, $\log(n!)$, $\log_8(n)$, 96
$8n^2$, $7n^3$, $5n$.

Ans→ $O(96) < O(\log_8(n)) < O(\log_2 n) < O(\log(n)) < O(n \log_6(n))$
$< O(n \log_2(n)) < O(5n) < O(8n^3) < O(7n^3) < O(n!)$
$< O(8^{2n})$.

Ans→ 19→ void Linear Search (int arr[], int n, int key)

```
{
for (i=0 to i=n)
 if arr[i] = = key
   count << found ';
 else
   continue
}
```

Ans→ 20→ Iterative Insertion sort

```
→ void Insertionsort (arr ; n) {
   int i, temp, j
   for(i-1 to n
   {
    temp = arr[i]
    j = i - 1
    while j>= 0 && arr[j] >temp
    {
```

Scanned with CamScanner

```
arr [j+1] = arr[j]
      j--
  }
 arr[j+1] = temp
 }
}
```

RECURSIVE INSERTION SORT

```
→ insertion sort (arr, n)
  {
  if n <= 1
    return ;
    insertion sort (arr, n-1);
    last = arr[n-1] ;
     j = n-2
    while ( j >= 0 and arr[j] > last )
      {
       arr[j+1] = arr[j]
        j--
      }
    arr[j+1] = last ;
  }
```

Insertion sort is called online sorting because it don't know the whole input, it might make decision that later turn out to be not optimal.

Other algorithm are off-line algoritms ~~that~~ that are discussed in lectures.

Ans-> 21-> TIME COMPLEXITIES

	BEST	AVG	Worst	Space
~~Buffe~~ Bubble Sort	$O(n2)$	$O(n2)$	$O(n2)$	$O(1)$
selection sort	$O(n2)$	$O(n2)$	$O(n^2)$	$O(1)$
Insertion sort	(n)	$O(n2)$	$O(n^2)$	$O(1)$
merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n)$ {due to recursion}.
Quick sort	$O(n\log n)$	$O(n\log n)$	$O(n^2)$	$O(n)$
Heap sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(1)$

Ans-> 22->

	Implace	stable	online sorting.
Bubble sort	yes	yes	NO
Selection sort	yes	NO	NO
Insertion sort	yes	yes	yes
MERGE sort	NO	yes	NO
QUICK Sort	yes	NO	NO
HEAP sort	yes	NO	NO

Ans->23-> Binary search (arr, int n, key)

```
{
 beg =0
 end =n-1
 while ( beg <=end )
  {
   mid= ( beg+ end)/2
   if [arr [mid]  == key]
     found
   else if arr[mid] < key
      beg = mid +1
   else
      end= mid-1
  }
}
```

Time complexity of linear search - O(n)
Space complexity of linear search - O(1)

Time complexity of Binary search - O(log n)
Space complexity of Binary search - O(n)

Ans->24-> $T(n) = T(\frac{n}{2}) + 1$.

Scanned with CamScanner