Name → Mohd Nasir
Section → F
Roll No → 14

Student Id → 20021595
University Roll → 2016855

Date
Page

DESIGN AND ANALYSIS OF ALGORITHM
TUTORIAL-01.

Ques→1 Asymptotic Notation :→ Asymptotic notations are the mathematical notations used to describe the complexity (runing time of an algorithm.
Different types of asymptotic Notation.

(i) Big-O :→ Big O notation specifically describes worst case scenario. It represents the tight upper bound running time complexity of an algorithm.

$$f(n) \leq c \cdot g(n). \qquad \forall \; n \geq n_0, \; c \geq 0$$

Ex :→ $O(1)$, $O(n)$, $O(n\log n)$.

```
for (i=0; i<n; i++)
    sum = sum+i;
```

The time complexity of above example is $O(n)$.

(ii) Omega (Ω):- omega Notation specifically describe best case scenario. It represents the tight lower bound running time of an algorithm.

$$f(n) \geq c \cdot g(n) \qquad \forall \; n \geq n_0, \; c \geq 0$$

Scanned with CamScanner

Example :- $\Omega(1)$, $\Omega(\log n)$.

```
void search (int size){
    int element, i;
    if (e' size == 1){
        printf("found");
        return;
    }.
    for (i=0; i< size; i++)
    {
        if (arr(i) = element){
            printf ("found");
            return;
        }
    }
    printf(" Not found");
}.
```

The Best Complexity of above Example is :- $\Omega(1)$.

(iii) Theta(Θ) :- This Notation describes both tight upper bound and tight lower bound of an algorithm. In real scenario the algorithm not always run on best and worst cases the avg running lies between best and worst And can be represent by 'Θ' Notation.

$$C_1\, g(n) \le f(n) \le C_2\, g(n).$$

$\forall\ n \ge \max(n_1, n_2)$
$C_1 \ge 0$ and $C_2 \ge 0$

Ques. 10:- For the function n^k and c^n, what is the asymptotic relationship b/w these function ?

assume that $k >= 1$ of $c > 1$ are constants find out the value of c. and no. of which relationship hold?

↳ as given n^k and c^n

Relationship between n^k and c^n is.

$n^k = O(c^n)$

$n^k \le a\,(c^n)$

$\forall\ n \ge n_0$ constant $a > 0$

for $n_0 = 1$; $c = 2$

$1^k < a^2$

$\Rightarrow$ $n_0 = 1$ and $c = 2$

Scanned with CamScanner

Date : Page :
Topic :

Qus→2 for ($i = 1$ to n)
{ $i = i*2$ }
$i \to 1, 2, 2^2, 2^3, 2^4, \ldots\ldots, 2^k$.
$2^k = n$
take $\log_2$ both side.
$\log_2 2^k = \log n$

$k \log_2 2 = \log n$

$\boxed{k = O(\log n)}$

Qus→3 $T(n) = 3T(n-1)$ $n>0$, otherwise 1.
$T(n) = 3T(n-1)$ — (1)
$T(n-1) = 3T(n-2)$
from equ (1) :→
$T(n) = 3[3T(n-2)]$
$= 3^2 T(n-2)$ — (2)
$T(n-2) = 3T(n-3)$
from equ (2) :→
$T(n) = 3^2[3T(n-3)]$
$= 3^3 T(n-3)$
⋮
$T(n) = 3^k T(n-k)$. — (3)
$n - k = 0$
$n = k$

Date : Page :
Topic :

from equ 3 :->

$T(n) = 3^K T(n-n)$

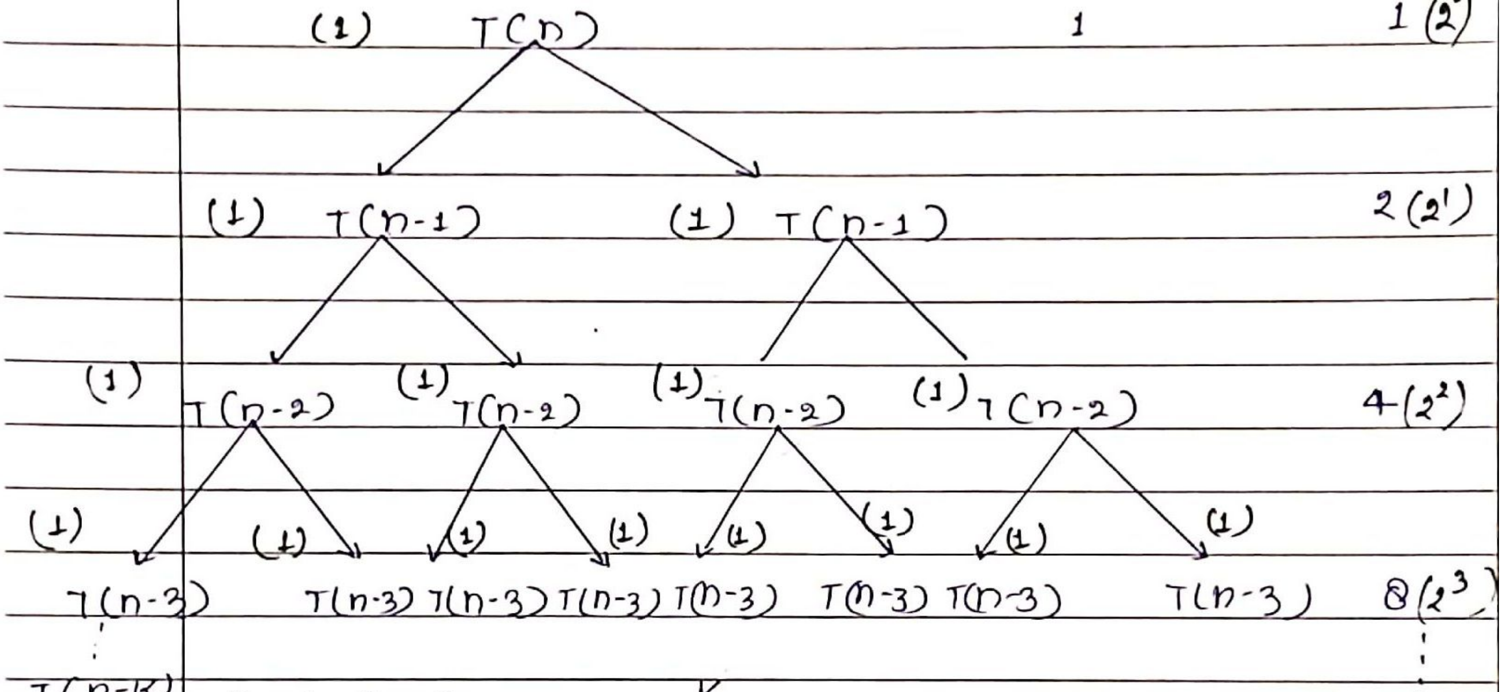
$= 3^K T(0)$

$T(0) = 1.$

$T(n) = 3^K$

$\boxed{T(n) = O(3^K)}$

Ques:- $T(n) = 2T(n-1) - 1$



$T(n-K)$ (2^K)

$2^0, 2^1, 2^2, 2^3 \ldots\ldots, 2^K$

$T(n) = T(n-K) * 2^K$

$n-K=0$

$n=K$

$T(n) = T(0) * 2^K$

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

Signature................................

Date : Page :
Topic :

Ans→5
```
int i=1, s=1;
while (s<=n){
    i++;
    s=s+i;
    Printf("#");
}.
```

s (Value of s):→

1, 3, 6, 10, ..., n

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

Termination Condition.

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

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

$K^2 + K = 2n$

$K^2 = n$

$K = \sqrt{n}$

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

Ans→6.
```
void Function (int n){
    int i, count=0;
    For (i=1; i*i<=n; i++)
        Count++;
}.
```

i→ 1, 2, 3, ..., K^2

$K^2 = n$ (terminate the program) $K^2 > n$

$K = \sqrt{n}$

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

STUDENT STYLE

Signature........................

Date

Topic :

Page

Ous→7
```
Void fun (int n) {
int i, j, K, count = 0;
for (i = n/2 ; i <= n ; i++) {          loop 1.
    for ( j=1; j<=n; j=j*2; ) {          loop 2
        for (K=1; K<=n; K=K*2) {         loop 3
            Count++;
        }
}.
```

loop 1 :→

$i \rightarrow \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3 \dots \frac{n}{2}+K$

$\hookrightarrow K = \frac{n}{2}$

$K > n$ (terminate pro).

$K = n$

$T(n) = O(n)$

for lop 1 and loop 2 we know time complexity.

$T(n) = \log n.$

total time complexity.

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

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

Date : Page :
Topic :

Ques→8
```
fun (int n){
  if(n==0)
     return;
  for (i=1 to n)
     for (j=1 to n)
       printf("#");

  fun (n-3)
}.
```

$T(n) = (n-3) + O(n^2)$

$= T(n-3) + n^2$ —— ①

$T(n-3) = T(n-6) + (n-3)^2$

From equ ①:→

$T(n) = T(n-6) + (n-3)^2 + n^2$

$T(n-6) = T(n-9) + (n-6)^2$

$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n^2$

$T(n) = T(n-3K) + (n-(3K-3))^2 + [n-(3K-6)]^2 + n^2$

$n-3K=0$

$n=3K$

$K=n/3$

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

$= T(0) + \frac{n(n+1)(n+2)}{6}$

$T(n) = n*n*n$

$\boxed{T(n) = O(n^3)}$

Signature..............................

Scanned with CamScanner

Date ________ Page ________
Topic ________

Ques 9. void fun(int n) {
    for (i=1 to n)
        for (j=1 ; j<=n ; j=j+i)
            printf("*");
}

$i=01, \quad j \to 1, 2, 3, 4, \ldots\ldots, n$

$i=2, \quad j \to 1, 3, 5, 7, \ldots\ldots, n/2$

$i=3, \quad j \to 1, 4, 7, 19, \ldots\ldots, n/3$

⋮

$i=n, \quad j \to 1, \ldots\ldots, n/n$

$$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n}$$

$$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{n}{n}\right)$$

→ logn.

$T(n) = n\log n$

$$\boxed{T(n) = O(\log n)}$$