Name: km Parshvi Sahu
Uni. Roll no: 1918035
Sec : D4

DAA
Tutorial 1

Page: Date:

Ans 1- → Asymptotic notation are used do tell the complexity of an algorithm when the I/p is very large.

→ Asymptotic Notation are language to express the required time & space by an algorithm to solve a given problem or it is a function to describe the performance of an algorithm

Eg.

Time Complexity	Space Complexity
(n^2) approx. no. of instructions	(n) extra space that an algorithm takes except input

Ans 2:-

```
for(i=1 to n)
{   i = i * 2;
}
```

1, 2, 4, . . . n

$a = 1, \quad r = 2$

$t_K = ar^{K-1} = 1 \cdot 2^{K-1} \longrightarrow n = \frac{2^K}{2}$

$2^K = 2n \longrightarrow K = \log_2(2n)$

$\therefore K = \log_2(2n) + 1$

Time Complexity $= \log_2(n)$

Ans 3:- $T(n) = \{3T(n-1)$ if $n>0$; otherwise $1\}$

using substitution method

$T(n) = 3T(n-1)$

$= 3(3T(n-2))$

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

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

∴

$= 3^n \cdot T(n-n)$

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

$= 3^n$

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

Ans4:- $T(n) = \{ 2T(n-1) - 1 \text{ if } n>0, \text{ otherwise } 1 \}$

using substitution method:-

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

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

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

$= 2^2 (2T(n-3)-1) - 2 - 1$

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$= 2^n \cdot T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \cdots 2^0$

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

$= 1$

T.C. $= O(1)$

Ans 5:-

```
int i=1, s=1;
while(s<=n){
    i++;
    s=s+i;
    printf("#");
}
```

We can define the term 's' acc to relation $s_i = s_{i-1} + 1$. The value of 'i' increases by one for each iteration. The value contained in 's' at the i^{th} relation is the sum of the first 'i' the integers If K is total no. of iterations taken by the program. then while

loop terminates if:-

$1+2+3+\cdots K = \frac{K(K+1)}{2} > n$

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

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

Ans6

```
void function(int n){
int i, count=0;
for(i=1, i*i<=n; i++)
        count++;
}
```

loop ends if $i^2 > n$

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

An7

```
Void function(int n){
int i, j, k, count=0;
for(i=n/2; i<=n; i++)  ...  n
  for(j=1; j<=n; j=j*2)
    for(k=1; k<=n; k=k*2)   } execute log n time
        count++;
```

Time Complexity $= O(n \log^2 n)$

Ans8:

```
Void function(int n)
{
  if(n==1) return;  → constant time
  for(i=1 to n) {  → n times
    for(j=1 to n) {  → n times
      printf("*");
    }
  function(n-3);
}
```

Page:
Date:

Recurrence rel^n : $T(n) = T(n-3) + n^2$

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

Ans 9-

```
Void function (int n){
  for (i=1 to n{ → This loop execute n times
  for (j=1; j<=n; j=j+i) → This execute j-
    printf ("*");          times with i increase
                           by the role of i
  }
}
```

$\Rightarrow$ Inner loop executes n/i times for each value of i Its running times is $n \times \left(\sum_{i=1}^{n} n/i\right)$

$= O(n \log n)$

Ans 10- The asymptotic relationship b/w the functions n^k and a^n is

$n^k = O(a^n)$ $\quad k >= 1, a > 1$

$n^k \leq c \cdot a^n$ $\quad \forall$ $n \geq n_0$

$\rightarrow \frac{n^k}{a^n} \leq c$