Rashi

2017[3]

DATE: / / 20

PAGE No

Tutorial 1

① Asymptotic Notation : These are the mathematical notation used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value

→ Θ Notation - Theta notation bounds a function from above & below so it defines exact asymptotic behaviour

eg: $3n^3+6n^2+6000 = \Theta n^3$

→ Big O notation : The big o notation defines an upper bound of an algorithm it bonds only from above

Eg: Two loops with iteration over n & m ~~technique~~ terms respectively will have time complexity as $O(n+m)$

→ Ω notation :- Just as Big O notation provides upper bound on a function Ω (omega) notation provides an asymptotic lower bound.

② for(i=1 to n) { i = i*2)

Time complexity for a loop means no. of times the loop has run for following values of i

$2^1, 2^2, 2^3 \dots 2^k$

$[1, 2, 3, \dots k]$ times

as per program

$2^k = n$

$k \log 2 = \log n$

$k = \log_2 n$

$T(n) = \log_2 n$

$O(\log_2 n)$

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

$T(1) = 3$

$T(n) = 3T(n-1)$ —— (1)

Let $n = n-1$

$T(n-1) = 3T(n-2)$ —— (2)

from (1) & (2)

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

$\vdots$

$T(n) = 3^n[T(n-n)] = 3^n T(0)$

$= 3^n$

$\therefore O(3^n)$

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

$T(n) = 2T(n-1) - 1$ —— (1)

Let $n = n-1$

$T(n-1) = 2T(n-2) - 1$ —— (2)

from (1) & (2)

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

$T(n) = 2^2 T(n-2) - 2 - 1$ — (3)

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

Rashi 2017532 DATE: / 20 PAGE No

let $k = n$

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

$= 2^n - 2^n + 1$

$= 1$

$O(1)$

(5) Time complexity

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

i	s	
1	1	1
2	1+2	3
3	1+2+3	6
,	1	
,	,	
,	,	
,	,	
	n	

$T(k) = 1+2+3 \cdots k$

$= \frac{1}{2}(k*1)$

for k iteration

$1+2+3+ \cdots k <= n$

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

$= \frac{k^2+k}{2} <= n \qquad O(k^2) <= n$

$\therefore k = O(\sqrt{n})$

$= O(\sqrt{n})$

Rashi 2017532

DATE : / 20
PAGE No

(6)
```
Void function (int n)
{
    int i, count = 0
    for(i=1; i*i <= n; i++)
        count++;                    // O(1)
}
```

$i * i <= n$

$i <= \sqrt{n}$

$i = 1, 2, 3 \dots = \sqrt{n}$

$\sum_{i=1}^{n} 1+2+3+\dots+\sqrt{n}$

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

$T(n) = \frac{n+\sqrt{n}}{2}$

$O(n)$

(7)
```
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++;
}
```

→ for $k = k * 2$

$k = 1, 2, 4, 8, \dots n$

Rashi

2017532

DATE: / / 20
PAGE No

$n = \frac{a(r^k - 1)}{r - 1}$ $[a=1, r=2]$

$n = 2^k - 1$

$\log n = k$

i	j	k
1	$\log n$	$\log n * \log n$
2	$\log n$	$\log n * \log n$
3	⋮	⋮
⋮	⋮	⋮
n	$\log n$	$\log n * \log n$

$\therefore O(n \log^2 n)$

(8)

```
function(int n)
{
    if(n==1) return;
    for(i=1 to n)
    {
        for(j=1 to n)
            print("*");
    }
    function(n-3);
}
```

$T(n) = T(n-3) + n^3$ — (1)

let $n = n-3$

$T(n-3) = T(n-6) + n^2$ — (2)

$T(n) = T(n-6) + n^2 + n^2$ — (3)

⋮

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

Rashi 2017532

DATE : / 20
PAGE No

Let $n-3k \geq 1$

$T(n) = T(1) + kn^2$

$\therefore O(n^2)$

(10) For the function, n^k & c^n, what is the asymptotic relation between those fun)

→ Assume that $k >= 1$ & $c > 1$ are constants Find out the value of c & n_0 for which relation holds.

As given n^k & c^n

relation between n^k & c^n is

$n^k \geq O(c^n)$

as $n^k \leq a c^n$

$\forall n \geq n_0$ & same constants $a > 0$

for $n_0 = 1$

$c = 2$

$1^k \leq a 2^1$

$n_0 = 1 \quad c \geq 2$