classmate
Date
Page

Assignment - 1

Q1| What do you understand by asymptotic notations define different asymptotic notation with the .exc

→ It is a mathematical notation that describes the behaviours of a function as its input size approaches infinity. It is used to analyse the time & space complexity of algorithm.

Diff types of asymptic notations are

1) · Big O(n) = It is used to describe the upper bound of the running time or space complexity of algorithm. It is worst case senario of an algorithm

$f(n) = O(g(n))$

if $f(n) < g(n)$

$\forall\ n > n_0$ some constant $c > 0$

ii)

iii)

ii) Big Omega (Ω) → It is used to describe the lower space complexity of an algo. It is the best case scenario of an algo.

$f(n) = \Omega(g(n))$

if $f(n) \geq g(n)$

$\forall\ n \geq n_0$ some constant $(c > 0)$

iii) Theta (Θ) → It is used to describe the tight bound of the remaining time or space complexity of an algorithm. It is average case scenario of an algo.

$f(n) = \Theta(g(n))$

if $c_1 g(n) \leq f(n) \leq c_2 g(n)$

$\forall\ n > \max(n_1, n_2)$ some constant $c_1 > 0$ & $c_2 > 0$

classmate
Date
Page

✓) Small omega (ω) → It is used to describe the strict lower bound of running time or space complexity of an algorithm. It is a more strict version of Big omega notation.

$f(n) = \omega(g(n))$

if $f(n) > 8\$ \cdot c(g(n))$

$n > 0$; & $c > 0$

3)

2) What should be time complexity of

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

Sol - $i = 1, 2, 4, 8 \quad n$

k - times

There it is a G P

$a_n = ar^{n-1}$

$n = 1 \cdot 2^{k-1}$

$2^k = n$

$\log_2(2n) = \log_2(2^k)$

$\log_2(2n) = k \log_2(2)$

$k = \log_2(2) + \log_2(n)$

$k = 1 + \log_2(n)$

$O(\log_2 10)$

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

By otherwise substitution

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

$T(0) = 1$

$T(1) = 3T(1-1) = 3T(0) = 3$

$T(2) = 3T(2-1) = 3T(1) = 3 \cdot 3$

$T(3) = 3T(3-1) = 3T(2) = 3 \cdot 3 \cdot 3$

$O(3n) = O(n)$

Q5) What should be time complexity

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

Let us assume the input size is n

On first iteration

i	s
2	3
3	6
4	10
5	15
6	21
.	
.	
.	
k^{th}	

After k^{th} iteration s will become $\frac{k(k+1)}{2}$

Q6)

loop will stop after

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

$$k^2 + k > 2n$$

$$k^2 > 2n$$

$$k = \sqrt{2n}$$

off after ignoring constants

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

Q6) Time complexity of

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

Let us assume the input size is n

when	i	after loop i becomes
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

after kth time i becomes k^2

So loop will terminate at

$k^2 > n$

$k \geq \sqrt{n}$

$O(\sqrt{n})$

Q7)

```
void function (int n)
{
  for 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++;
        }
      }
    }
}
```

The complexity of inner most loop

1, 2, 8, 16

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

$2n = 2^k$

taking log on both sides

$\log_2 2n = \log_2 2^k$

$\log_2 2^n = k \log_2 2$

$\therefore k = \log_2 2 + \log_2 n$

It means for each value of j this inner loop uses $1 + \log_2 n$ times

complexity of middle loop

$j = 1$ to $j = j * 2$

$1, 2, 4, 8, 10 \cdots k^{th}$ time

It means for each value of this loop runs ($1 + \log_2$ times)

Complexity of outermost loop

$j = n/2$ to n $i++$

$n/2 ; \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3 \cdots k^{th}$

k^{th} term $= \frac{n}{2} + k$

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

- this loop will run $\frac{n}{2}$ times

Total complexity

$= \frac{n}{2}(1 + \log_2 n) + (1 + \log_2 n)$

$= \frac{n}{2} + \frac{n}{2}\log_2 n + \frac{n}{2}\cdot(\log_2 n)^2 + \frac{n}{2}\log_2 n$

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

Q4) $T(n) = 2T(n-1) - 1$ if $(n > 0)$
otherwise 1

Here $a = 2$

$b = 1$

$c = \log_b(2) = 1$

$f(n) = 1$

$n^c = f(n)$

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

Q8

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

i	j
1	1 →
2	1 → n
3	1 → n
4	1 → n
n	

n times running

for function (n-3)

$n, n-3, n-6, n-9 \ldots$ k^{th} term

$n, n-1\cdot3, n-2\cdot3, n-3\cdot3 \ldots$ k^{th} term

k^{th} term $= n-(k-1)\cdot3 = n-3k-3$

$1 = n-3k-3$

$\Rightarrow n-3k-3-1=0 \Rightarrow n-3k-4=0$

$k = \frac{n-4}{3}$

Inner most loop will execute
$n \times n \times \frac{n-4}{3}$ times $= n^3 - 4n^2$

~~for~~
~~loop~~
Complexity $= O(n^3)$ ignore constant & smaller value

Q9) Void function (int n)
{
for (i=1 to n)
{
for (j=1; j<=n; j=j++)
{
print ("*")
}
}
}

// Outer loop will n times (i)

for $i = 1$ will run n times
$i = 2$ will " $n/2$ "
$i = 3$ " " $n/3$ times
$i = n$ " " n/n "

classmate
Date
Page

Inner loop will run

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

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

$= n \cdot \log n$

- Sum of $\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots \frac{1}{n}\right)$ is $\log n$

Complexity $= O(n \cdot \log n)$