

## Tutorial 2

A.1.) void fun(int n)

int j=1, i=0;

while(i < n)

{

    i = i + j

    j++

}

0 1 2 3 ... n-1

↑ 1 2 5 7 ... n(n-1)  
↓ 1 2 3 4 ... n

i = 1, 3, 5, 7, ..., n(n-1)

T(C) of  $\sum_{i=1}^n i = \frac{1}{2}n(n+1) = O(n)$

sum of AP i.e.  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$

$\Rightarrow \frac{n(n+1)}{2} + n(\text{for } j) \leq n$

$\Rightarrow \frac{n^2+n+2n}{2} \leq 2n$

$\Rightarrow n^2 \leq 2n \Rightarrow n^2 \leq n$

$\Rightarrow n < \sqrt{n}$

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

A.2.)

Fibonacci series program using Recursion:-

#include <iostream>

using namespace std;

int fib(int n)

{

    if(n <= 1)

```

    return n;
    return fib(n-1) + fib(n-2);
}

int main()
{
    int n=9;
    cout << fib(n);
    getch();
    return 0;
}

```

$$T(1) = T(n) + \text{constant}$$

Q.3) Write programs which have complexity  
 (i)  $n(\log n)$ ,  $n^3$ ,  $\log(\log n)$ .

A.3.)  $\Rightarrow n(\log n)$ :-

```

for(i=0; i<n; i++)
{
    for(j=1; j<n; j = j+2) n*logn
}

```

$\rightarrow n(\log n)$

ii)  $n^3$ :- Floyd Warshall Algorithm  
 for solving all shortest  
 path b/w vertices

i.e for  $k=1$  to  $n$

for  $i=1$  to  $n$

"  $j=1$  to  $n$

$$A^k[i, j] = \min(A^{k-1}[i, j], A^{k-1}[i, k] + A^{k-1}[k, j])$$

return  $A$

(ii) for ( $i=1$ ;  $i < n$ ;  $i=i+2$ )  $(T(c) = \log(\log(n)))$

{

$p++;$

}

for ( $j=1$ ;  $j < p$ ;  $j=j+2$ )

{

}

$$A.4.) T(c) \text{ or } T(n) = T(n/4) + T(n/2) + cn^2$$

assuming  $T(n/2) \geq T(n/4)$

on rearranging

$$T(n) \leq 2T(n/2) + cn^2$$

Now apply master's theorem to RHS

$$T(n) \leq \Theta(n^2) \Rightarrow T(n) = \Theta(n^2).$$

$$T(n) \geq cn^2 \Rightarrow T(n) = \Theta(n^2) \Rightarrow T(n) = \Omega(n^2)$$

$$\therefore T(n) = \Theta(n^2) \text{ & } T(n) = \Omega(n^2)$$

$$\Rightarrow T(n) = \Theta(n^2) \text{ by }$$

Q.5) What's time complexity of following fun()?

int fun(int n)

{

    for (int i=1; i<=n; i++)

{

        for (int j=1; j<n; j++)

{

            //some O(1) task

} }

A.5) for i=1, inner loop executes n times.

i=2,     "         "      $\frac{n}{2}$  times.

i=3,     "         "      $\frac{n}{3}$  times

i=n,     "         "      $\frac{n}{n}$  times.

T(C) of  $(n + \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n})$

$$\Rightarrow n * \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}\right)$$

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

Q.6) What should be time complexity of

for (int i=2; i<n; i) pow(i, k)

{

    //some O(1) expression / Statement

}

where K = constant

A.6.) It takes values  $2^k, 2^{k^2}, (2^k)^k = 2^{k^2}$

$$\Rightarrow (2^k)^k = 2^{k^2} \dots 2^{k \log k (\log(n))}$$

Last term must  $\leq n$

$$\Rightarrow 2^{k \log k (\log(n))} = 2^{\log(n)} = n.$$

$\downarrow$  eqv to last term

$\therefore$  there are  $\log k (\log(n))$

$$\therefore T(k) = O(\log(\log(n))) \text{ Ans}$$

Q.7) Write recursive relation . . .

what do you understand by analysis?

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + O(n).$$

taking one branch 99% & other 1%.

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

1<sup>st</sup> level = n.

$$2^{\text{nd}} \text{ level} = \frac{99}{100} + \frac{n}{100} = n$$

So 3<sup>rd</sup> level remain same for any kind of position.

$$\text{large branch} = O(n \log^{100} / ggn)$$

for shorter branch

$$\hookrightarrow = O(n \log_{10} n)$$

either base complexity of  $O(n \log n)$  remains.

A.8.)

$$\text{a.) } 100 < \sqrt{n} < \log(\log n) < \log(n) < n < n \log n < \log n! < n^2 < n!$$

$$< 2^n < 4^n < 6^n < 2^{2n}$$

$$\text{b.) } K \log(\log n) < \sqrt{\log n} < \log n < \log 2n < n < n \log n =$$

$$< 2n < 4n = 2(2^n) < n! < n^2$$

$$\text{c.) } 96 < \log n < \log n! < n \log n < n \log_2 n < 5n < n! < 8n^2 < 7n^3$$

$$< 8 \frac{n^2}{4} n$$