Aditi Agarwal
CST-27

Page No.
Date

Tutorial-2

Q.1 What is the time complexity of below code

```
void fun (int n)
{
    int j=1, i=0;
    while (i<n)
    {
        i=i+j;
        j++;
    }
}
```

j	i
1	0
1	1
2	3
3	6
4	10
5	15

$S = 0+1+3+6+10+15 \ldots\ldots T_k$ ——— (1)

also, $S = 0+1+3+6 \ldots\ldots T_{k-1} + T_k$

$0 = 1+2+3+4 \ldots\ldots k - T_k$

$T_k = 1+2+3+4 \ldots\ldots k$

$T_k = \frac{1}{2} k(k+1)$, for k iterations

$1+2+3+6 \ldots\ldots k < n$

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

$\Rightarrow \frac{k^2+k}{2} < n$

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

$k \simeq O(\sqrt{n})$

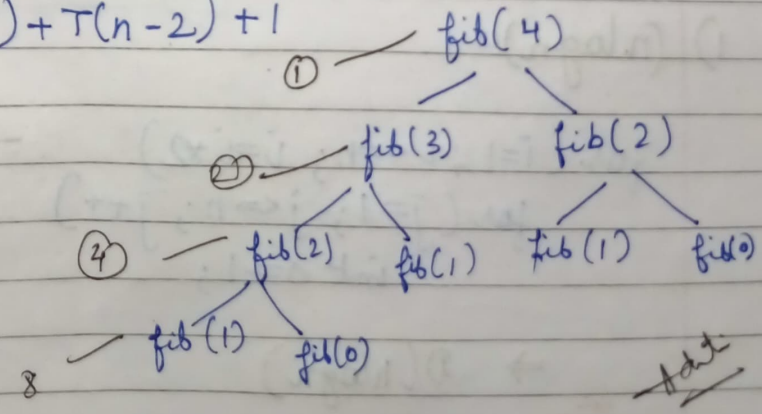
$\boxed{T(n) < O(\sqrt{n})}$

Q2 Write recurrence relation for recursive function that prints fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program & why ?

0 1 1 2 3 5 ..... n

```
int fib (int n)
{   if (n<=1)
        return n;                          —— O(1)
    return fib(n-1) + fib(n-2)             —— T(n-1) + T(n-2)
}
```

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



$$T(n) = 1+2+4+8 \cdots\cdots n$$

$$S(n) = \frac{a(r^{\text{terms}} - 1)}{r-1}$$

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

$$= 2^n \cdot 2 - 1$$

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

Space Complexity $(O(1))$

as recursive implementation doesn't store any values and calculates every value from scratch. so as complexity of 1 call is $O(1)$

$$\therefore \text{Total space complexity} = O(1)$$

Q.3 Program which have complexity :-

1) $(n \log n)$

```
for (i=1; i<=n; i=i*2)        ——— logn times
    for (j=1; j<=n; j++)      ——— n times
        int a=1;
```

$\Rightarrow O(n \log n)$

2) n^3 :

```
for (i=0; i<=n ; ++i)          ---- n calls
    for ( j=0; j<=n; ++j)      ---- n calls
        for (k=0; k<=n; ++k)   ---- n calls
            cout<<" OK";
```

$\Rightarrow O(n^3)$

3) $\log(\log n)$:

```
for (int i= 2; i<n ; i= pow (i,c) )   ⇒ O(loglogn)
    cout << " Hi";
```

// where c is any constant

Q.4 $T(n) = T(n/4) + T(n/2) + cn^2$

Neglecting the lower order term $T(n/4)$

$T(n) = T(n/2) + cn^2$

Let, $a=1, b=2$

$c = \log_2 1 \Rightarrow 0$

$n^c = n^0 = 1 < cn^2$

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

Aditi

Q.5

```
int fn(int n)
{
    for(int i=1; i<=n; i++)
        for (int j=1; j<n; j+=i)
            cout<< "Hi";
}
```

for $i=1$, $j = 1+2+3+4+5+6 \cdots\cdots n$
for $i=2$, $j = 1+3+5+7 \cdots\cdots n$
for $i=3$, $j = 1+4+7 \cdots\cdots n$

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

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

$$= n \int_1^n \frac{1}{x}$$

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

Q.6 Time Complexity : `for (i=2; i<=n ; i=pow(i,k)) { }`

where, k is constant

I iteration $i = 2$
II iteration $i = 2^k$
III iteration $i = (2^k)^k = 2^{k^2}$

$\vdots$

n^{th} Iteration $i = 2^{k^i} = n$

$$n = 2^{k^i}$$

$$\log n = \log 2^{k^i} = i \log 2^k = k^i \log 2$$

$$\log n = k^i$$

Taking log of base k

$$\log_k \log n = \log_k (k^i)$$

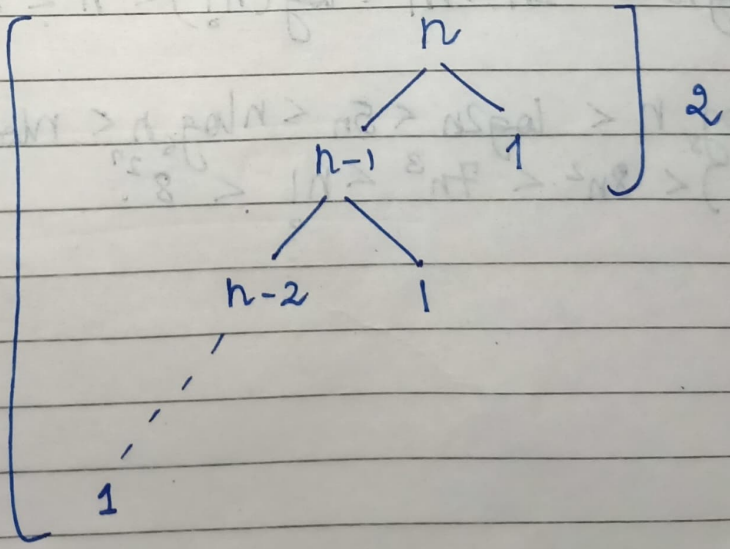
$$i = \log_k \log(n)$$

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

Ques7 Given algo divides array in

99% of 1% parts

$\therefore T(n) = T(n-1) + O(1)$



Ankit

$$h = [T(n-1) + T(n-2) + \cdots T(1) + O(1)] \times n$$

$$= n \times n$$

$$\because T(n) = O(n^2)$$

lowest height = 2
max height = n

$$\boxed{\because \text{diff} = n-2} \qquad n > 1$$

The given algorithm provides linear result.

Q.8 Arrange the following in increasing order of growth rate.

a) $100 < \log\log n < \log n < (\log n)^2 < \sqrt{n} < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2n}$.

b) $1 < \log\log n < \sqrt{\log n} < \log n < \log_2 n < 2\log n < n < n\log n < 2n < m < \log(n!) < n^2 < n! < 2^{2n}$.

c) $96 < \log_8 n < \log 2n < 5n < n\log_6 n < n\log_2 n < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$.