PAGE NO. 1
DATE: / /

Q1 What is the time complexity of below code & how?

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

Sol

$j = 1$	$i = 1$
$j = 2$	$i = 1+2;$
$j = 3$	$i = 1+2+3;$

m level

for (i)

$\because 1+2+3+ \cdots < n$

$\because 1+2+3+m \cdots < n$

$\because \frac{m(m+1)}{2} < n$

$m \approx \sqrt{n}$

$\because$ by loop method

$\Rightarrow \sum_{i=1}^{m} 1 = 1+1+\cdots \sqrt{n}$ times

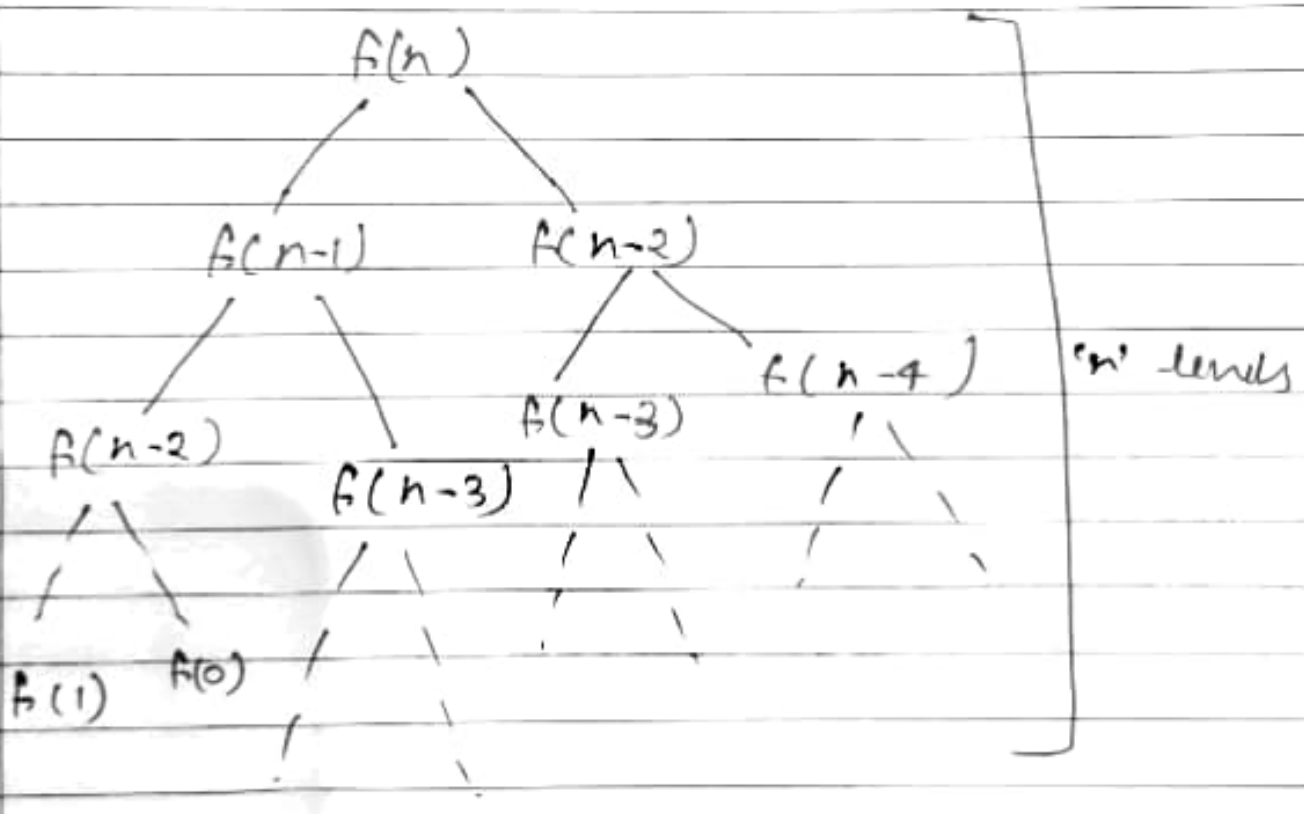
$\because \boxed{T(n) = \sqrt{n}}$

ques 2 Fibonacci Series

sol^n $f(n) = f(n-1) + f(n-2)$

$f(0) = 0 \quad f(1) = 1$

by forming tree :-



∵ At every function we get 2 function call

∵ for n levels :-

we have $= 2 \times 2 \cdots n$ times

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

maximum space :-

we consider recursive stack :-

no. of calls maximum = n

for each call we have space complexity O(1)

∴ $\boxed{T(n) = O(n)}$

without cosidering recursive stack :

ar each can we have time complexity O(1)

∴ $\boxed{T(n) = O(1)}$

Q3 write a programe which have complexity n(log n), n^3, log(log n)

i) for n(log n)

quick sort

PAGE NO. 4
DATE: / /

```
void quicksort (int arr[], int low, int high)
{
  if (low < high)
  {
  int pi = partition (arr, low, high);

  quicksort (arr, low, pi-1);
  quicksort (arr, pi+1, high);

  }
}

int partition (int arr[], int low, int high)
{
   int partition = arr [high];
   int i = (low -1);

for (int j = low; j <= high -1; j++)
  {
    if (arr [i] <          )
       {
          i++ }
          swap (&arr[i], &arr[j]);
```

```
    }
    swap (& arr[i+1], & arr[high]);
    return (i+1);

}
```

ii) for n^3

multiplication of two square matrix

```
for (i = 0; i < r1; i++)
    for (j = 0; j < c2; j++)
        for (k = 0; k < c1; k++)
        {
            res[i][j] += a[i][k] * b[k][j];
        }
```

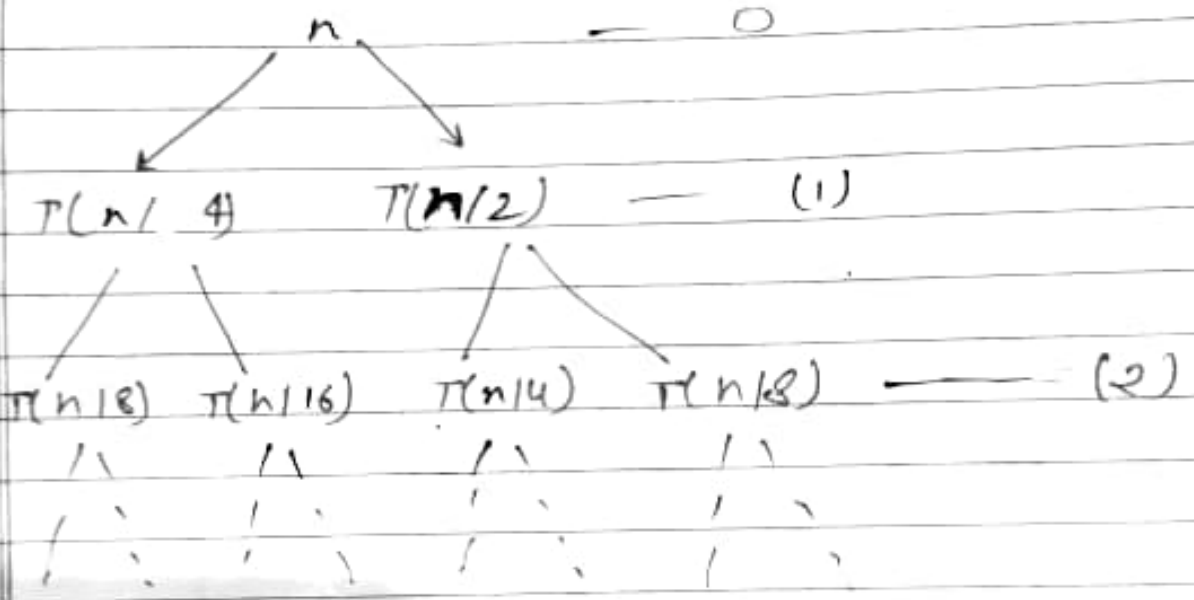
iii) for log(log n)

```
for (i = 2; i < n; i = i * i)
{
    count++;
}
```

Ques 4 Solve the following recurrence relation

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

Soln 4



At level :-

$0 \rightarrow cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{5n^2}{16}$

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 cn^2$

⋮

max level $= \frac{n}{2^k} = 1 \Rightarrow k = \log_2 n$

$$\therefore T(n) = c\left(n^2 + \left(\frac{5}{16}\right)n^2 + \left(\frac{5}{16}\right)^2 n^2 \cdots + \left(\left(\frac{5}{16}\right)^{\log n} n^2\right)\right)$$

$$T(n) = cn^2\left[1 + \left(\frac{5}{16}\right) + \left(\frac{5}{16}\right)^2 + \cdots \left(\frac{5}{16}\right)^{\log n}\right.$$

$$T(n) = cn^2 \times 1 \times \left(\frac{1 - (5/16)^{\log n}}{1 - (5/16)}\right)$$

$$= cn^2 \times \frac{11}{5}\left(1 - \left(\frac{5}{16}\right)^{\log n}\right)$$

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

$\downarrow$

$\boxed{O(Cn^2)}$

ques 5 what is the complexity (time) of

```
int fun (int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j < n; j += i) {
            // some O(1) tasks
        }
    }
}
```

sol 5

i	j
1	1
2	1+3+5
3	1+4+7
1	1+5+9
|	|
|	|
n	

j = (n-1)/i times

$$\sum_{i=1}^{n} \frac{(n-1)}{i}$$

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

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

$$= n \log n - \log n$$

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

Ques What should be the time complexity of

```
for (int i = 2 ; i <= n ; i = pow (i, k))
{
    // some O(1) expression or statements
}
```

when, k is const

Sol

```
for (i = 2 ; i <= n ; i = pow (i, k))
    {
        O(1)
    }
```

for

2^1
2^k
2^{k^2}
2^{k^3}
.
.
.
2^{k^m}

where, k^m

$2 \cdot 2 = n$

$k^m = \log 2^n$

$m = \log_k \log_2 n$

$\because \sum_{c^2-1}^{m} 1$

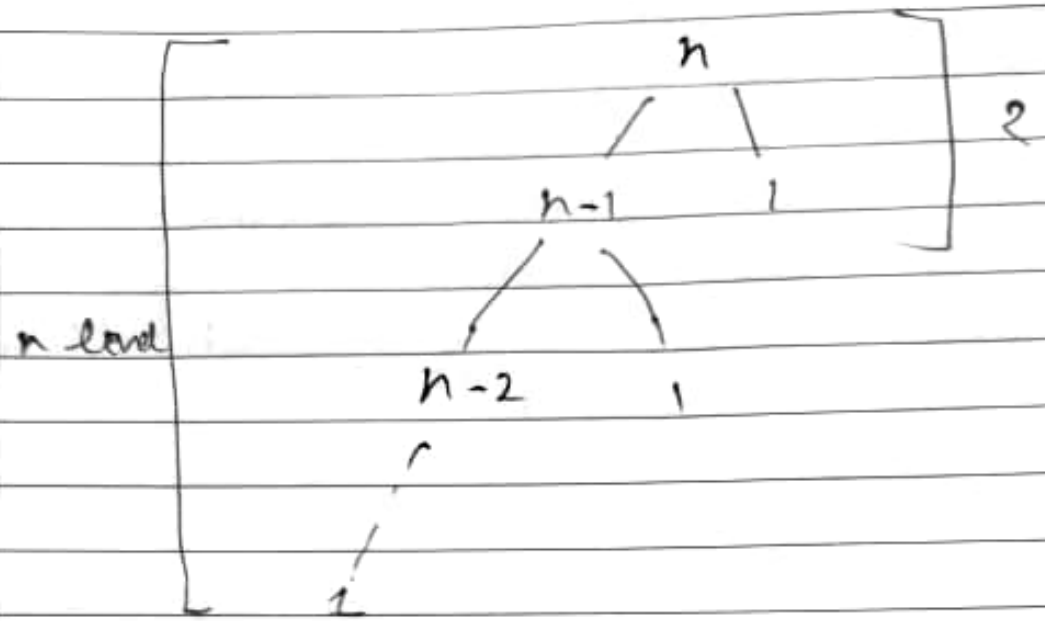
$\Rightarrow 1+1+1 \cdots \cdots$ m times

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

Ques 7 Given algo divides array in

Sol 7 99% & 1% parts

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



$$T(n) = \left(T(n-1) + T(n-2) + \cdots T(1) + O(1) \right) \times n$$

$$= n \times n$$

$$\therefore \quad T(n) = O(n^2)$$

lowest height = 2

height height = n

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

The given algo provides linear result.

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

Sol 8 a) $100 < \log\log n < \log n < (\log n)^2 < \sqrt{n}$
$< n < n\log n < \log(n!) < n^2 < 2^n < 4^n$
$< 2^{2^n}$

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

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