PAGE NO.
DATE: / /

Tutorial - 2

DAA

Name : Sakshi Sahu

Section : CST SPL -1

University Roll no. : 20022369

Class Roll no. : 03

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)

∵ $1+2+3+ \cdots < n$

∵ $1+2+3+m \cdots < n$

∴ $\frac{m(m+1)}{2} < n$

$m \simeq \sqrt{n}$

∵ by lw method

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

∵ $\boxed{T(n) = \sqrt{n}}$

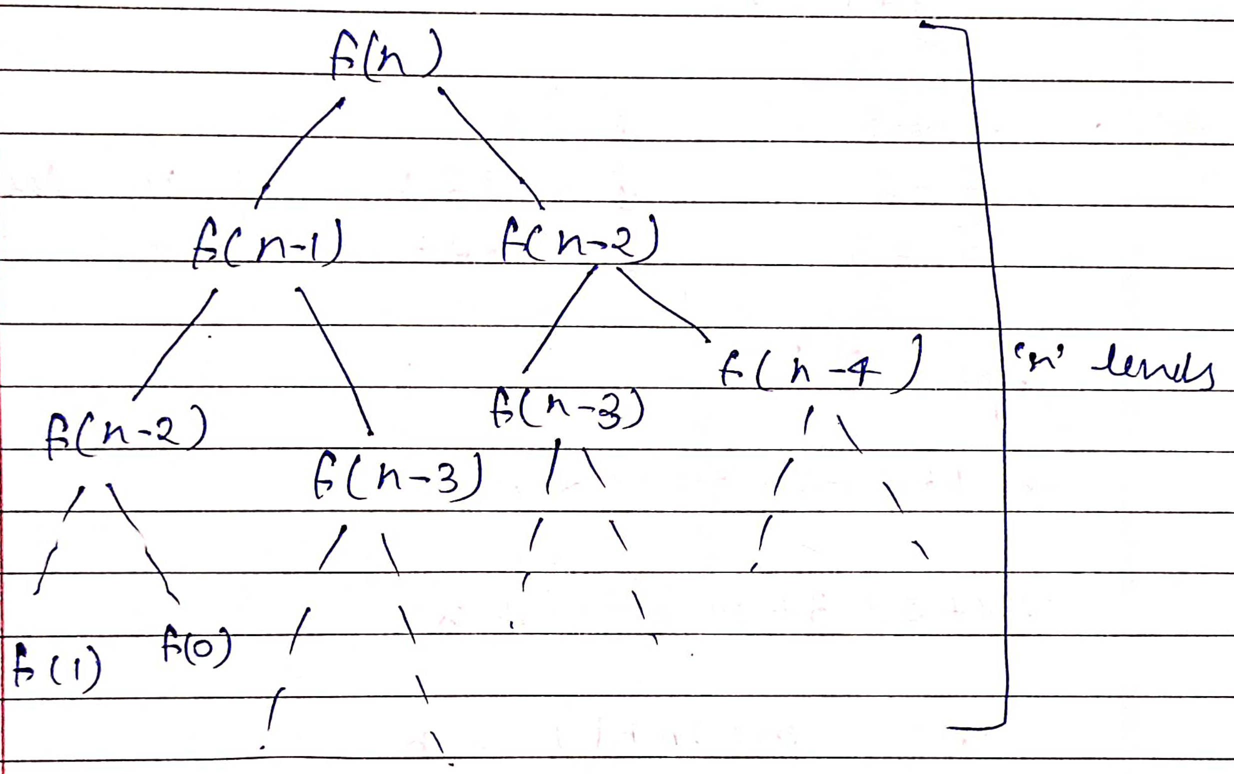
Sakshi Saw

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 × 2 ---- n times

∴ $\boxed{T(n) = 2^n}$

Sakshi Sahu

maximum space :-

we Consider recursive stack :-

no. of calls maximum = n

for each call we have space complexity $O(1)$

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

without cosidering recursive stack :

as each can we have time complexity $O(1)$

$\therefore$ $\boxed{T(n) = O(1)}$

Qs3 write a programe which have complexity

$n(\log n)$, n^3, $\log(\log n)$

i) for $n(\log n)$

quick sort

Sakshi Sahu

```
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]);
       }
```

Sayeshi Sahu

```
{
    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)

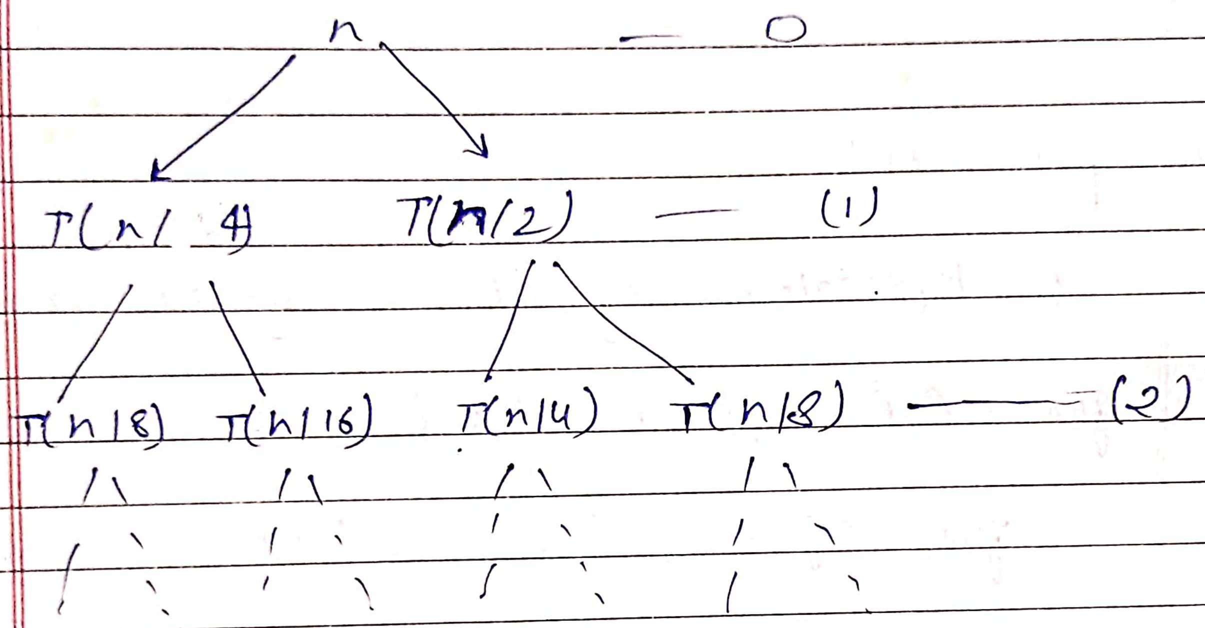
```
for (i = 2; i < n; i = i*i)
{
    count ++;
}
```

Sakshi Sahu

Ques 4 solve the following recurrence relation

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

Sol^n 4



At level :-

$0 \rightarrow Cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{C5n^2}{16}$

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

⋮

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

Sakshi Sahu

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

$$= \left.\left(\left(\frac{5}{16}\right)^{\log n} n^2\right)\right)$$

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

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

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

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

$$\downarrow$$

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

Sakshi Sahu

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) task
        }
    }
}
```

Sol 5

i	j	$j = (n-1)/i$ times
1	1	
2	1+3+5	
3	1+4+7	
1	1+5+9	
|	|	
n	|	

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

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

Sakshi sahu

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

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

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

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

Qus 6 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)
   }
```

Sayoshi Sahu

for

where, $2^{k^m} = 2 = n$

2^1

2^k

2^{k^2}

2^{k^3}

|

|

2^{k^m}

$k^m = \log_2 n$

$m = \log_k \log_2 n$

$\therefore \sum_{i=1}^{m} 1$

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

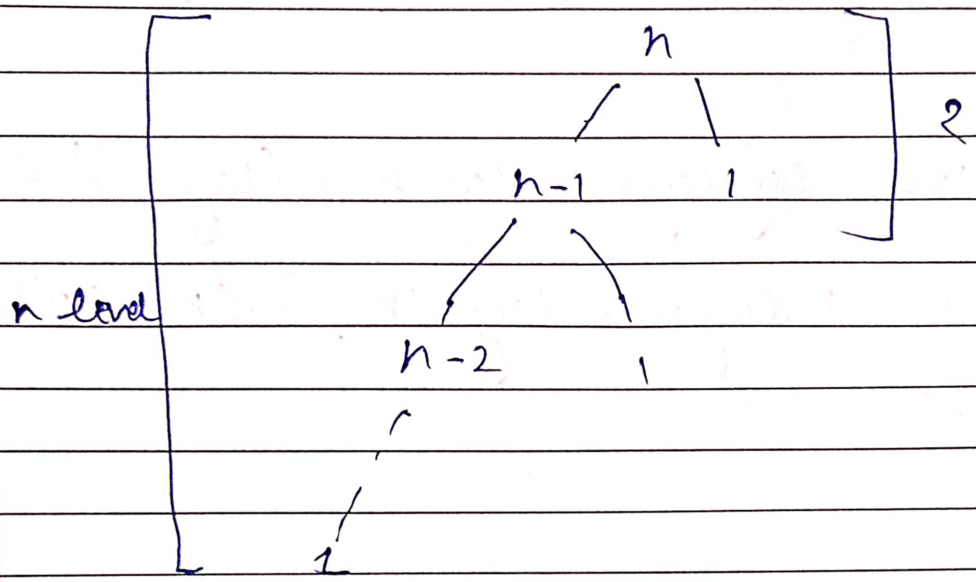
$$\boxed{T(n) = O(\log k \log n)}$$

Ans 7 Given algo divides array in

Sol 7 99% & 1% part

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

Sakshi Saw



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

$$= n \times n$$

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

lowest height = 2
height height = n

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

The given algo provides linear result.

Sakshi Sahu

Ques 8 Arrang 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) $1 < \log\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!) < n^2 < n! < 2^{2^n}$

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

Sakshi Sahu