

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

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

↳

j=1	i=1	] m-level
j=2	i=1+2	
j=3	i=1+2+3	

for (i)

$$\therefore 1 + 2 + 3 + \dots + < n$$

$$\therefore 1 + 2 + 3 + m < n$$

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

$$m \approx \sqrt{n}$$

By summation method

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

$$\boxed{T(n) = \sqrt{n}} \rightarrow \text{Ans}$$

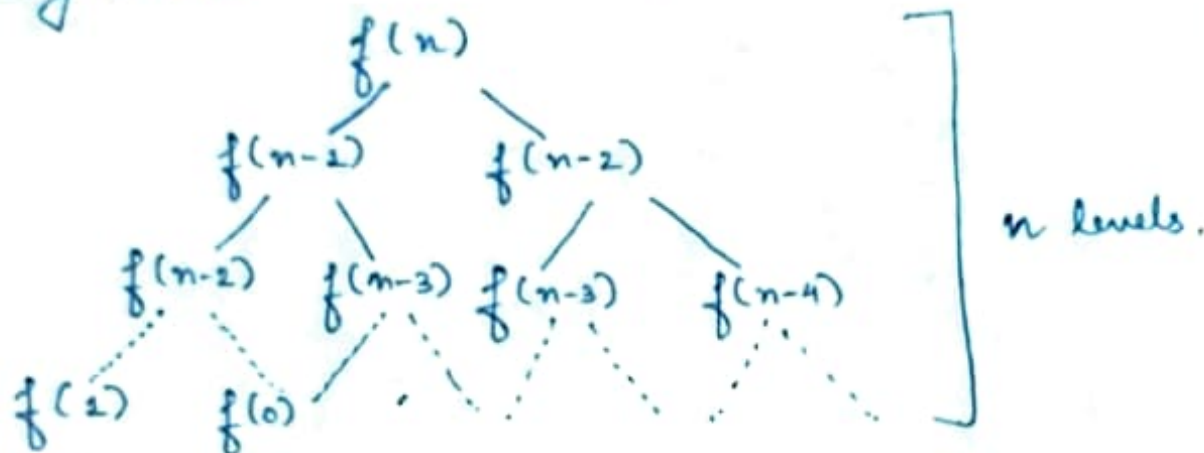
Q2. Write recurrence relation for function that prints Fibonacci series. Solve it to get the time complexity. What will be the space complexity and why?

→ For Fibonacci Series

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

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \end{aligned}$$

By forming a tree



- ∴ At every function call we get 2 function calls
- ∴ for  $n$  levels

We have  $\times 2 \times 2 \dots n$  times

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

MAXIMUM SPACE

### Considering Recursive

Stack :

No. of calls maximum =  $n$

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

$$\therefore T(n) = O(n)$$

Without considering Recursive stack:

each call we have time complexity  $O(1)$

$$\therefore \tau(n) = O(1)$$

Q3.

Write programs which have complexity:

$n(\log n)$ ,  $n^2$ ,  $\log(\log n)$

1)  $n \log n \rightarrow$  Quick sort

```
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 pivot = arr[high];
```

```
    int i = (low - 1);
```

```
    for (int j = low; j <= high - 1; j++)
```

```
    {  
        if (arr[j] < pivot)
```

```
        {  
            i++;
```

```
            swap (&arr[i], &arr[j]);  
        }  
    }
```

}

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

```
    return (i + 1);  
}
```

}

2)  $n^3 \rightarrow$  Multiplication of 2 square matrix

```
for (i = 0; i < n1; i++) {
```

```
    for (j = 0; j < c2; j++)
```

```
        for (k = 0; k < c1; k++)
```

```
        {
```

```
            res[i][j] += a[i][k] * b[k][j];
```

```
        }  
    }
```

log(log n)

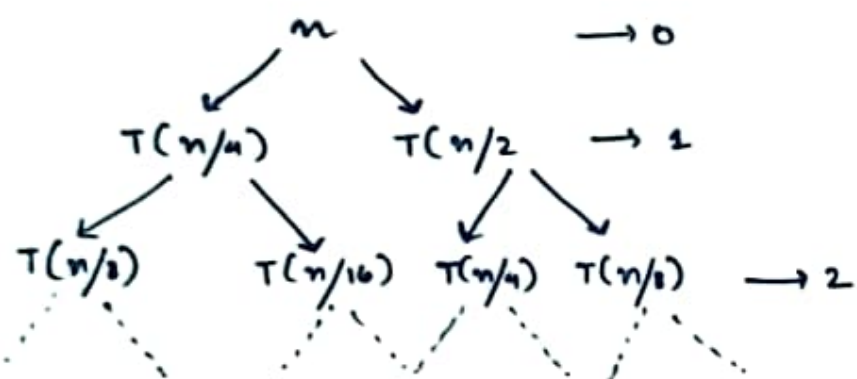
(4)

```
for (i=2; i<n; i=i+1)
{
    count++;
}
```

Q4. Solve the following recurrence relation

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

→



At level

$$0 \rightarrow cn^2$$

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

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

⋮

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

$$= k = \log_2 n$$

$$T(n) = c \left( n^2 + \left(\frac{5}{16}\right)n^2 + \left(\frac{5}{16}\right)^2 n^2 + \dots + \left(\frac{5}{16}\right)^{\log_2 n} n^2 \right)$$

$$T(n) = cn^2 \left[ 1 + \left(\frac{5}{16}\right) + \left(\frac{5}{16}\right)^2 + \dots + \left(\frac{5}{16}\right)^{\log_2 n} \right]$$

$$T(n) = cn^2 \times 1 \times \left( \frac{1 - \left(\frac{5}{16}\right)^{\log_2 n}}{1 - \left(\frac{5}{16}\right)} \right)$$

$$T(n) = cn^2 \times \frac{11}{5} \times \left( 1 - \left(\frac{5}{16}\right)^{\log_2 n} \right)$$

$$T(n) = O(n^2 c)$$

$$\underline{O(cn^2)} \quad \text{Ans.}$$

Q5. What is the time complexity of following fun()? (5)

```
int fun (int n) {
    for (int i=2; i<=n; i++) {
        for (int j=1; j<=n; j++) {
            // Some O(1) task
        }
    }
}
```

→ for

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

$j = (n-1)/i$  times

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

~~✓~~

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

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

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

$$T(n) = O(n \log n) \rightarrow \text{Ans.}$$

Q6. What should be time complexity of  
for (int i=2; i<=n; i=pow(i,k))

// Some O(1)

where k is a constant

→ for

i  
2<sup>1</sup>  
2<sup>2</sup>  
2<sup>4</sup>  
2<sup>16</sup>  
⋮  
2<sup>2<sup>m</sup>}</sup>

where

$$2^{2^m} < n$$

$$k^m = \log_2 n$$

$$m = \log_k \log_2 n$$

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

1 + 1 + 1 + ... m times

$$T(n) = O(\log_k \log_2 n) \rightarrow \text{Ans.}$$



Q7. Write a recurrence relation when quick sort repeatedly divides array into 2 parts of 99% and 1%. Derive time complexity in this case. Show the recurrence tree while deriving time complexity. If find difference in heights of both extreme parts. What do you understand by this analysis?

→ Given algorithm divides array in 99% and 1% part

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



'n' work is done at each level

$$T(n) = (T(n-1) + T(n-2) + \dots + T(1) + O(1)) \times n$$

$$= n \times n$$

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

lowest height = 2

highest height = n

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

The given algorithm produces linear result

Q8) Arrange following in increasing order of rate of growth:

- a)  $n, n!, \log n, \log \log n, \text{root}(n), \log(n!), n \log n, \log^2(n), 2^n, 2^{2^n}, 4^n, n^2, 100$
- $100 < \log \log n < \log n < (\log n)^2 < \sqrt{n} < n < n \log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$

- b)  $2(2^n), 4n, 2n, 1, \log(n), \log(\log(n)), \sqrt{\log(n)}, \log 2n, 2 \log(n), n, \log(n!), n!, n^2, n \log(n)$
- $1 < \log \log n < \sqrt{\log n} < \log n < \log 2n < 2 \log n < n < n \log n < 2n < 4n < \log(n!) < n^2 < n! < 2^{2^n}$

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