Shruti Kanchi
Roll No - 2017970
Section - AI & DS

Tutorial - 2

Ques-1 What is the time complexity of below code and how ?

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

→ $j=1$ $\quad i=1$
$j=2$ $\quad i=1+2$
$j=3$ $\quad i=1+2+3$

m-level

for (i)

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

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

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

$m \approx \sqrt{n}$

By summation method

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

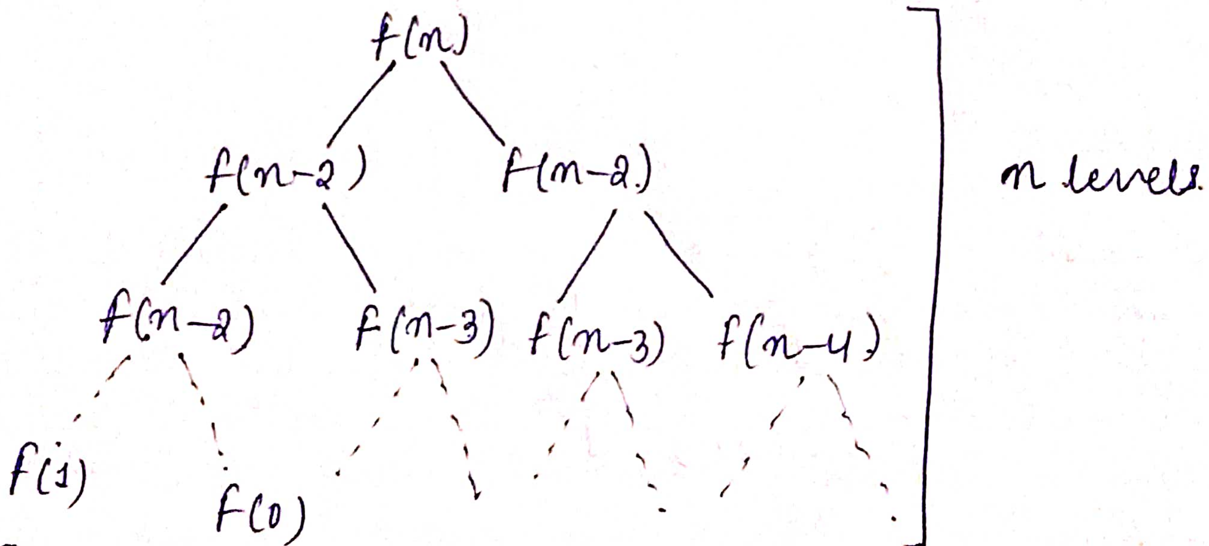
$$\boxed{T(n) = \sqrt{n}} \text{ — Ans}$$

<u>Ques2</u> Write recurrence relation for function that prints fibonacci series. Solve it to get the time complexity. What will be the space complexity and why?

<u>For Fibonacci series</u>

$$f(n) = f(n-2) + f(n-2) \qquad f(0)=0, \; f(1)=1$$

By forming a tree



∵ At every function call we get 2 func calls

∴ for n levels

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

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

<u>Maximum Space</u> →

considering Recursive

stack:

no. of calls maximum $= n$

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

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

without considering recursive stack;

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

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

Ques 3 write programs which have complexity:

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

1) $n(\log n)$ — 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 → multiplication of 2 square matrix

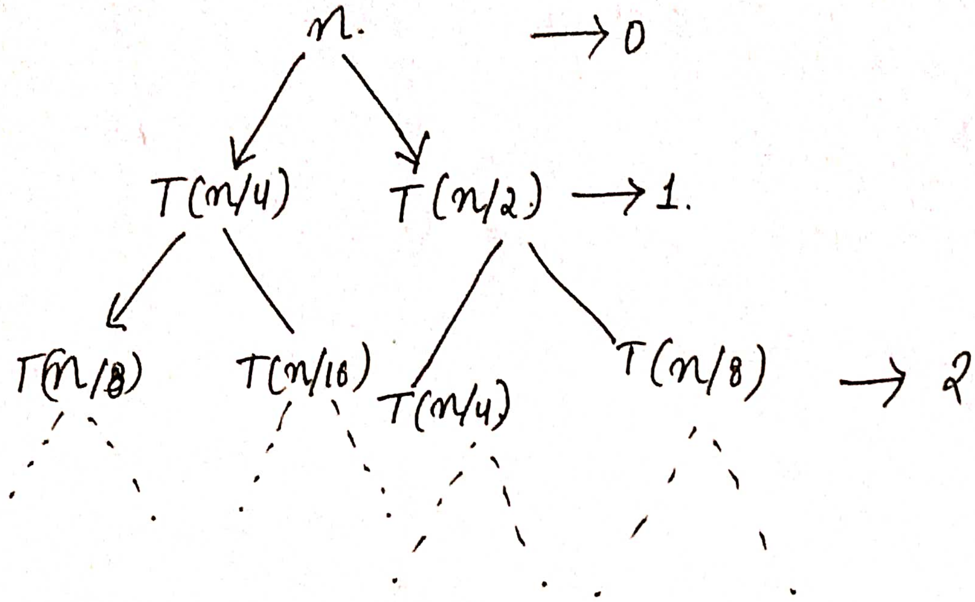
```
for (i=0 ; i<r1; i++)
  {
    for (j=0 ; j< c2 ; j++)
    for (k=0; k<c1; k++)
      {
        res[i][j] += a[i][k] * b[k][j];
      }
```

3) log (log n) →

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

Ques 4 Solve the following recurrence relation

$$T(n) = T(n/4) + T(n/2) + c\,n\text{^}2$$



At level

$$0 \rightarrow cn^2$$

$$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} \Rightarrow \frac{c5n^2}{16}$$

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

$$\vdots$$

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

$$= K = \log_2 n.$$

$$T(n) = C\left(n^2 + (5/16) + (5/16)^2 n^2 + \dots + (5/16)\log n\; n^2\right)$$

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

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

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

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

$O(cn^2)$

Ques-3 what is the time complexity of following fun()?

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

for

i	j
1	1
2	1 + 3 + 5
3	1 + 4 + 7
⋮	
n	

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

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

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

$$T(n) = n\left(1 + 1/2 + 1/3 + \dots + 1/n\right) - 1\times[1 + 1/2 + 1/3 + \dots + 1/n]$$

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

$$T(n) = O(n \log n) \text{ — Ans}$$

Ques 6 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
$\downarrow$

2^1

2^K

2^{K^2}

2^{K^3}

|

2^{K^m}

where $\rightarrow$

$$2^{K^m} <= n.$$

$$K^m = \log_2 n$$

$$m = \log K \log_2 n$$

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

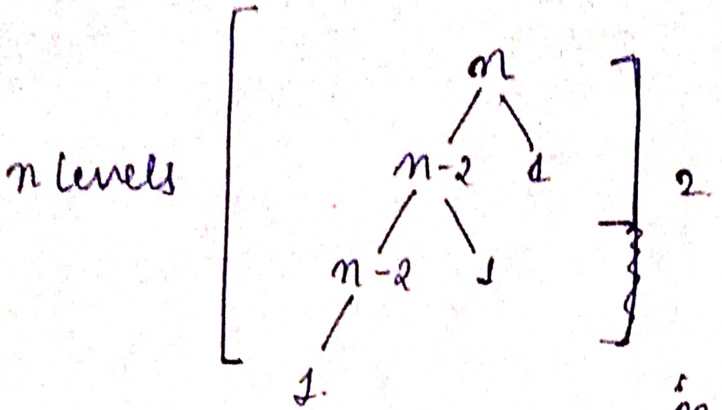
$1+1+1+ \ldots m$ times

$T(n) = O(\log_K \log n)$ — Ans

Ques-7. 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 & find difference in heights of both extreme parts. What do you understand by this analysis?

Given algo divides arr in 99% and 1% part

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



'n' work is done at each level

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

$= n \times n$

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

lowest height = 2

highest height = n

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

The given algo produces linear result.

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

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

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

b) $2(2^n), 4n, 2n, 1, \log(n), \log(\log(n)), \sqrt{\log(n)}, \log 2n, 2\log(n), n, \log(n!), n!, n2, n\log(n)$

$\rightarrow 1 < \log\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!) < n^2 < n! < 2^{2n}$

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

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