Date / /

Tutorial - 2

Name - Deepak kohli
Section - F
Roll no. - 53
Uni. Roll No. - 2016718

Q-1 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++;
    }
}
```

$j=1$ $\quad i=1$
$j=2$ $\quad i=1+2$
$j=3$ $\quad i=1+2+3$
} m - level

for(i)

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

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

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

$m \approx \sqrt{n}$

By Summation method

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

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

Date / /

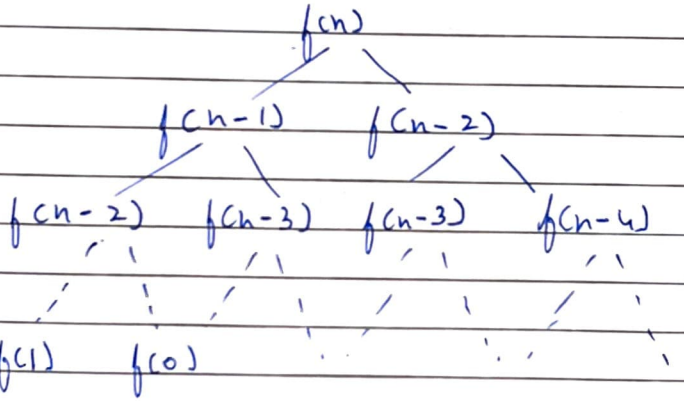
Q-2 Write reccurence relation for function that prints Fibonacci series. Solve it to get the time complexity. What will be the space complexity and why?

For Fibonacci Series

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

$$f(0) = 0$$
$$f(1) = 1$$

By forming a tree



∵ At every function call we get 2 function calls

∴ for n levels.

We have : $2 \times 2 \ldots\ldots$ n times

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

Date / /

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)

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

Q-3 Write program which have complexity :

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

1) $n \log n$ → Quick start

```
void quickstart (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);
    }
}
```

Date / /

```
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];
        }
```

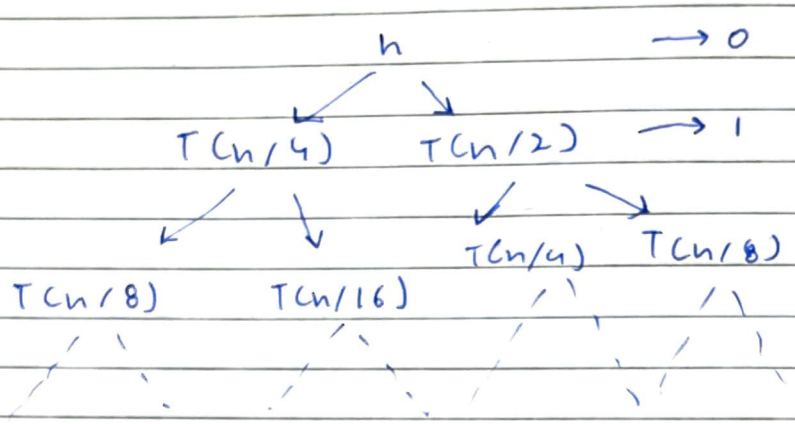
3) log (log n)

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

Date / /

Q-4 Solve the following recurrence relation

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



At level

$0 \rightarrow cn^2$

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

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

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

$= k = \log_2 n$

$T(n) = c(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \ldots$

$\ldots + (5/16)^{\log n} n^2)$

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

Date / /

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

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

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

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

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

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

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

$T(n) = n[1 + 1/2 + 1/3 - \cdots + 1/n] - [1 \times 1/2 + 1/3 + \cdots + 1/n] \approx n\log n - \log n$

$T(n) = O(n \log n)$ — Ans

Date / /

Q-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	where
2^1	$2^{k^m} <= n$
2^k	$k^m = \log_2 n$
2^{k^2}	
2^{k^3}	$m = \log_k \log_2 n$
.	
.	
.	
2^{k^m}	

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

$1 + 1 + 1 - - - - m$ times

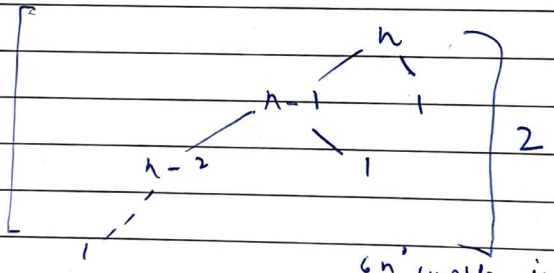
$T(n) = O(\log_k \log_2 n)$ — Ans.

Date / /

Q-7 Write a recurrence relation which quick sort repeatedly divide array into 2 parts of 99% and 1%. Derive time complexity in this case. Show the recurrence tree while deriving time complexity and find difference in heights of both extreme parts. What do you understand by this analysis?

Given algorithm divides array in 99% and 1% part

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



'n' work is done at each level.

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

$= n \times n$

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

Lowest height = 2

highest height = n

$\therefore$ difference $= n-2$ $\quad n \geq 1$

The given algorithm produces linear result.

Date / /

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

a) $n, n!, \log_{10}\log n, \text{Root}(n), \log(n!), n\log n, \log^{2}(n), 2^n, 2^{2^n}, 4^n, n^2, 100$

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

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) $8^{2n}, \log_2(n), n\log_6(n), n\log_2(n), \log(n!), n!, \log_{10}(n), 96, 8n_2, 7n_3, 5n$

$96 < \log_2 n < \log 2n < 5n < n\log(n) < n\log_2 n < \log(n!) < 8n^2 < 7n^3 < n! < 8^{16n}$.