Tutorial-2

Name - Kulmeet Singh
Section - F
Roll No. - 13
University Roll No. - 2016327

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++;
    }
}
```

Sol:-

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

for (i)

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

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

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

$m \simeq \sqrt{n}$

By summation method

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

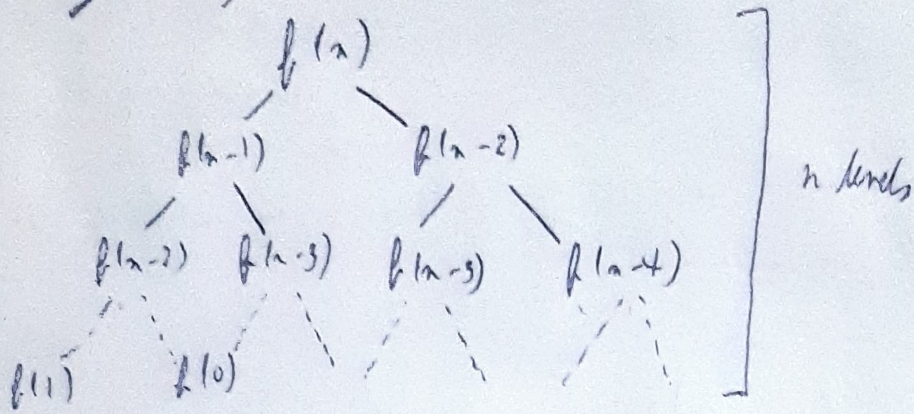
$\boxed{T(n)=\sqrt{n}}$

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?

Sol: For Fibonacci series

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

By forming a tree,



∴ At every function calls we get 2 function calls

∴ for n levels

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

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

Maximum Space

Considering Recursive

Stack: No. of calls maximum = n

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

∴ $\boxed{T(n) = O(n)}$

Without considering Recursive Stack:

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

∴ $\boxed{T(n) = O(1)}$

Q3. Write programs which have complexity:

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

Sol:- 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 [i] < 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 < r; i++)
  for (j = 0; j < c; j++)
    for (k = 0; k < c1; k++)
      { res [i][j] = a [i][k] * b [k][j];
      }
```

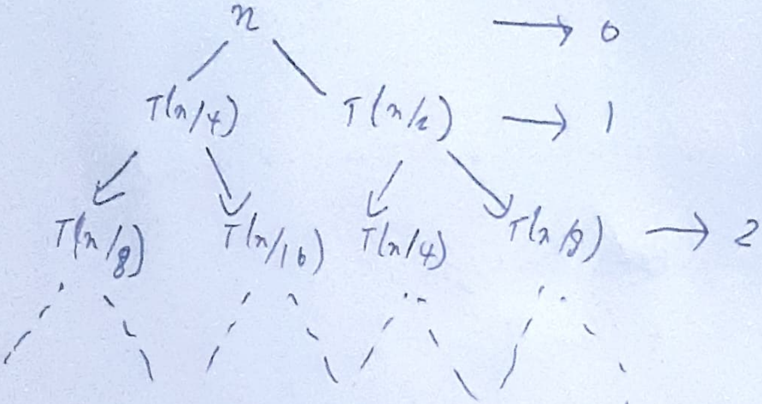
3) $\log(\log n) \rightarrow$

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

Q4. Solve the following recurrence relation

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

Sol:-



at level

$0 \rightarrow cn^2$

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

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

⋮

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

$\Rightarrow \boxed{k = \log_2 n}$

$$T(n) = c\left(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \ldots\ldots + (5/16)^{\log n} n^2\right)$$

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

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

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

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

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

Q5. 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	1+5+9

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

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

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

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

$= n \log n - \log n$

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

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

2^1

2^k

$2k^2$

$2k^3$

⋮

2^{km}

where

$2k^m \le n$

$k^m = \log_2 n$

$m = \log_k \log_2 n$

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

$1 + 1 + 1 + \dots$ m times

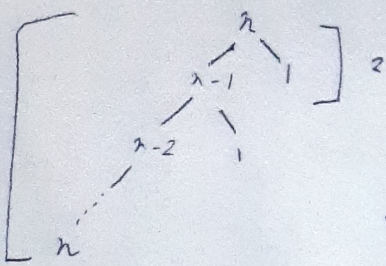
$\boxed{T(n) = O(\log_k \log n)}$

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

Sol:- Given algorithm divides array in 99% and 1% part.

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

n levels



'n' work is done at each level.

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

$= n \times n$

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

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

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

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, $n\log(n)$

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

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

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