21 March 2022

TUTORIAL - 2

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

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

2. Write a reccurance relation for the recursive function that prints Fibonacci Series. Solve the reccurance relation to get time complexity of the program. What will be the space complexity and why?

3. Write programs which have complexity:

(i.) $n(\log n)$

(ii.) n^3

(iii.) $\log(\log n)$

Aaksha.

4. Solve the following recurrence relation
$T(n) = T(n/4) + T(n/2) + cn^2$.

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

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

6. What should be the time complexity of

```
for (int i = 2; i <= n; i = pow(i, k))
{       // O(1) }
```

where k = constant.

7. Write a RR when quick sort repeatedly divides array into 2-parts of 99% & 1%. Derive T.C. Show R. Tree and find diff. in heights of both extreme parts.

8. Arrange in increasing order of rate of growth.

Kshitija.

Solutions

1. for $j = 1$ $i = 1$

$= 2$ $= 1 + 2$

$= 3$ $= 1 + 2 + 3$

$\vdots$

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

for while condition

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

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

$$n^2 + n < 2n$$

$$n^2 < n$$

$$n < \sqrt{n} \qquad n \approx \sqrt{n}$$

Using Summation Method:

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

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

2. For fibonacci series:

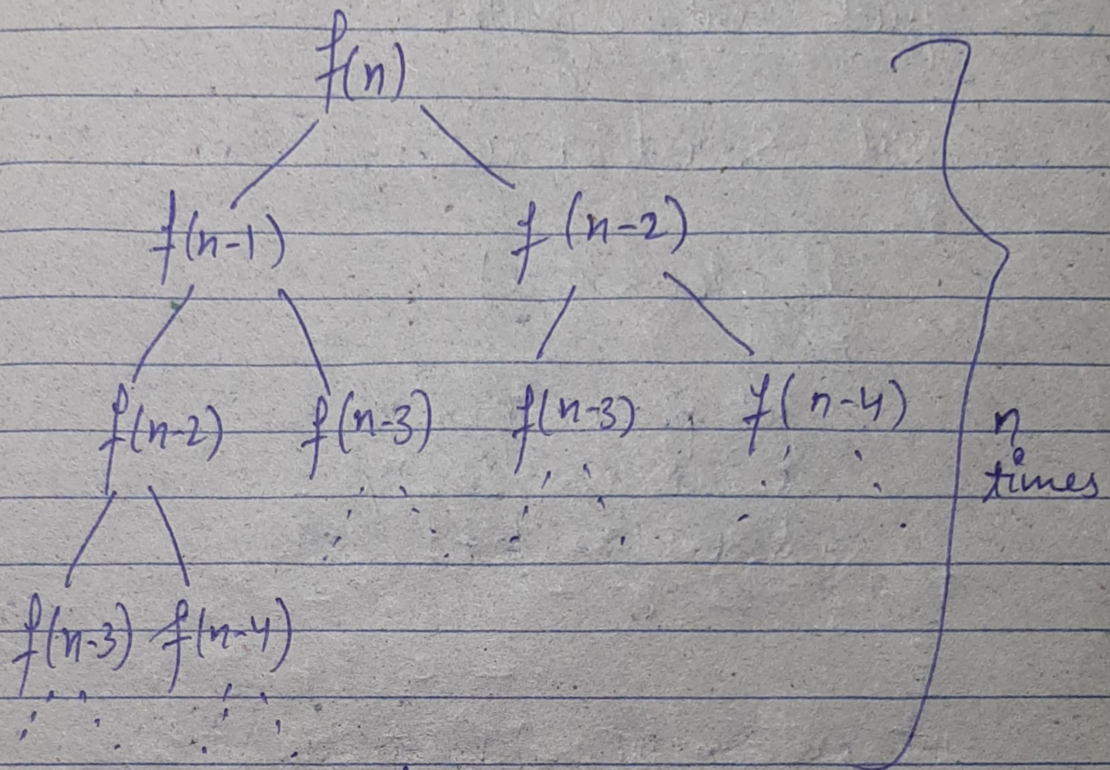
$$f(n) = f(n-1) + f(n-2)$$

$$f(0) = 0$$

$$f(1) = 1$$

Aleksha

Tree :



∵ At every function call we get 2 function calls.

∴ for 'n' levels :

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

∴ $T(n) = O(2^n)$

→ Maximum Space ⇒ $O(n)$ for recursive call (max. no. of calls)

→ without recursive stack :

$T(n) = O(1)$

A. Saksha.

3. 1. n(log n)

→ quick sort algorithm :

```
void Q_sort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        Q_sort (arr, low, pi-1);
        Q_sort (arr, pi+1, high);
    }
}
int partition(int arr[], int low, int high)
{
    int piv = arr [high];
    int i = (low - 1);
    for (int j = low ; j<=high-1 ; j++)
    {
        if (arr [j] < piv)
        {
            i++;
            swap ( &arr[i] , &arr[j]);
        }
    }
    swap (& arr [i+1] , & arr [high]);
    return (i+1);
}
```

Saksha.

2. n^3

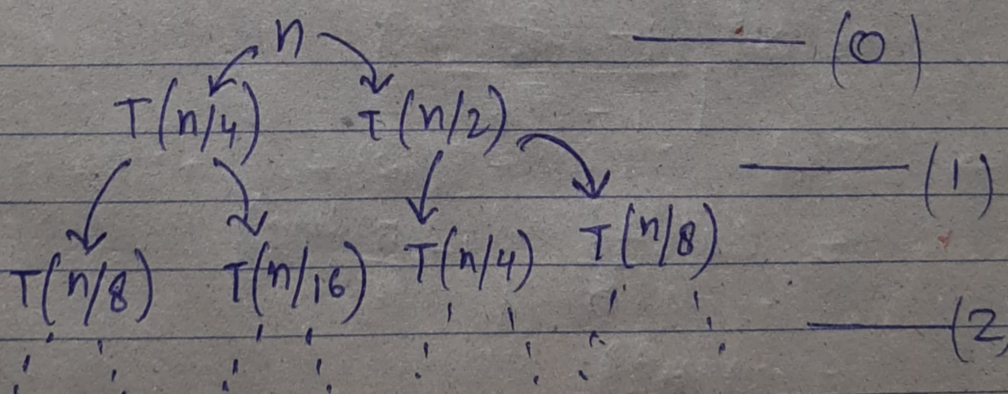
Multiplication of two sq. matrix.

```
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++;
}
```

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



$0 \longrightarrow cn^2$

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

$$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2}$$

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

$\vdots$

max level $\Rightarrow \dfrac{n}{\cancel{2^k}\,2^k} = 1$

$$\Rightarrow k = \log_2 n$$

$$\therefore T(n) = c\left(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \dots\right)$$

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

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

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

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

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

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

$$= O(cn^2)$$

Sakshi

5.
```
int fun(int n)
{
  for (i=1; i<=n; i++)
    for (j=1; j<n; j+=i)
    { // O(1) }
}
```

for

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

$\Rightarrow \sum_{i=1}^{n} \frac{(n-1)}{i} \quad \Rightarrow T(n) = \frac{(n-1)}{1} + \frac{(n-1)}{2} + \dots + \frac{(n-1)}{n}$

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

$= n\log n - \cancel{\log n}$

$= O(n\log n)$.

6.
```
for (i=2; i<=n; i=pow(i,k))
{
    // O(n)
}
```

pow (i, k)

$i = 2^1$

$= 2^k$

$= 2^{k^2}$

$= 2^{k^3}$

$\dots$

$\therefore 2^{k^m} <= n$

$k^m = \log_2 n$

$m = \log_k \log_2 n$

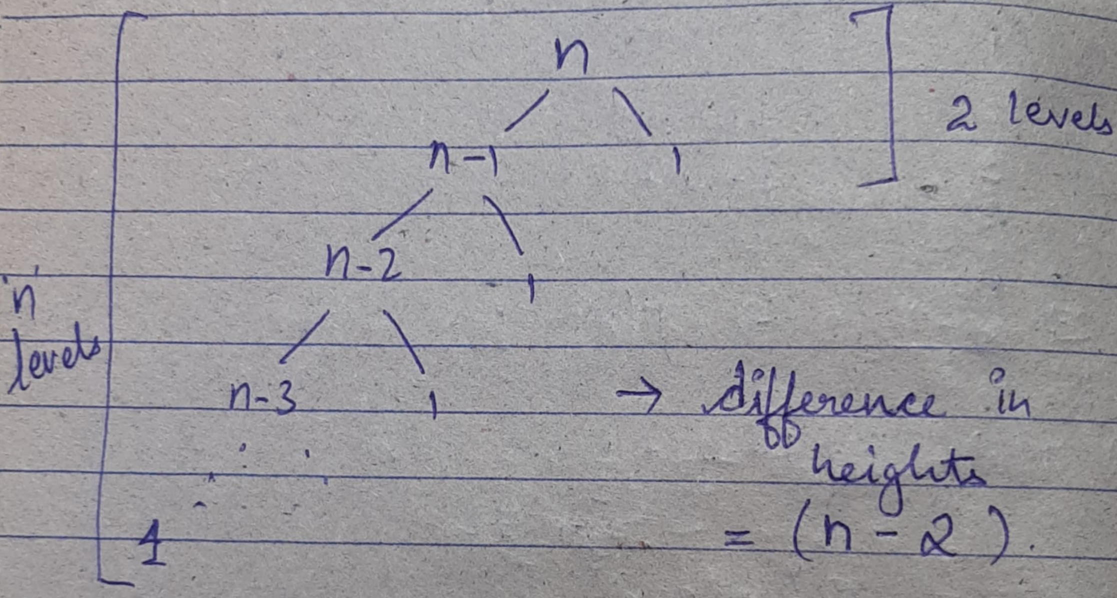
Diksha

and $O(1) \Rightarrow$ m times

$$\therefore \quad T(n) = O(\log_k \log_2 n) + \cancel{O(1)}$$

$$= O(\log_k \log_2 n).$$

7. $T(n) = T(n-1) + O(1)$



→ difference in heights $= (n-2)$.

→ lowest height $= 2$
highest height $= n$

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

$$= n \times n$$

$$= \underline{\underline{O(n^2)}}.$$

Sucksha.

8. Considering large value of 'n',

(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) $96 < \log_8 n < \log 2n < 5n < n(\log_6 n)$

$< n(\log_2 n) < \log(n!) < 8n^2 < 7n^3 < n!$

$< 8^{2n}$.

Saksha