Design and Analysis of Algorithms
Tutorial 2

(1) What is the time complexity of below code & how?

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

1, 3, 6, 10, ...

$j=1 \quad i=1$
$j=2 \quad i=1+2$
$j=3 \quad i=(1+2)+3$ } M-levels

for(i)

$\because$ $1+2+3+ \cdots +$ ~~[illegible]~~ $m < n$

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

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

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

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

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

Anushka

__/__/___

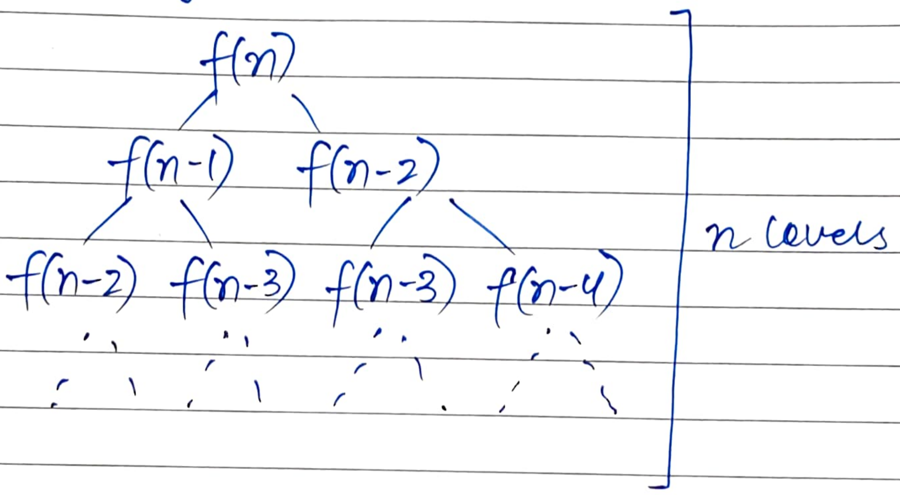
Q2. Write a recurrance relation for the recursive func that prints fibonacci series. Solve the recurrance relation to get the time complexity of this program and why?

For fibonacci series:

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

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

By forming a tree



Since we got two function calls at each level
∴ for n levels
we have : 2×2×...n times

$T(n) = 2^n$

Maximum space:
considering recursive stack:
no. of calls maximum = n

for each call we have space complexity of O(1)
∴ $T(n) = O(n)$
without considering recursive stack
each call will have a time complexity of O(1)

$T(n) = O(1)$

Anushka//

Q3. write programs which have complexity of $n(\log n)$, n^3, $\log(\log n)$

$(n\log n) \rightarrow$ 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++)
    {
        int (arr[i] < pivot)
        {
            i++;
            swap(&arr[i], & arr[j])
        }}
    swap(&arr [i+1], & arr[high]);
    return (i+1);
}
```

Anushka

_/_/___

$n^3 \rightarrow$ multiplication of a ~~[illegible]~~ 2 square matrix

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

log(log n)

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

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

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

Anushka

__/__/__

for	i	j_1
	1	
	2	$1+3+5$
	3	$1+4+7$
	⋮	⋮
	n	$1+5+7$

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

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

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

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

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

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

⑥ what should be the 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
	2^{k^2}
	2^{k^3}
	⋮
	2^{k^m}

where

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

$$k^m = \log_2 n$$

$$m = \log_k \log_2 n$$

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

$1+1+1+ \cdots m$ times

$$T(n) = O(\log_k \log n)$$ Ans

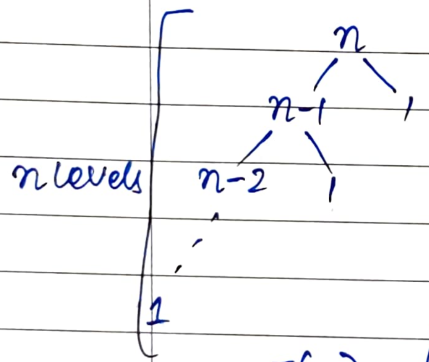
Anushka

__/__/____

7. Write a recurrance relation when quick sort repeatedly divides the array into two parts of 99% and 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity in this case, and find the difference in height of both the extreme parts. What do you understand by this analysis?

Given algo divides array in 99% and 1% part

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



'n' work is done at each level

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

$= n \times n$

$= n^2$

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

lowest height $= 2$

highestheight $= n$

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

[illegible]

The given algo produces linear result.

8. Arrange the following in inc order of rate of growth

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

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

(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, $8n^2$, $7n^3$, $5n$

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

Anushka