Tutorial-2

Q1 Time complexity of below code

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

$i=0 \rightarrow i=0+1$, $j=2$

$i=1 \rightarrow i=1+2$, $j=3$

$i=3 \rightarrow i=1+2+3$, $j=4$

$i=6 \rightarrow i=6+4$, $j=5$

$\vdots$

$i=k \quad i=\underbrace{k+j}_{\text{sum}}$ - $j=n$

OR

j	1	2	3	4	- - - -	n
i	1	3	6	10	- - - -	k

$$\frac{k(k+1)}{2} > n$$

$$\frac{k^2+k}{2} > n$$

$$k^2 = n$$

$$k = \sqrt{n}$$

Time complexity $= O(k)$ or $O(\sqrt{n})$

Q2 Write recurrence relation for recursive function that prints fibonacci series

```
int fib(int n)
{
    if(n<=1)
        return n;
    return fib(n-1)+fib(n-2);
}
```

$$T(n) = \begin{cases} 1 & , n <= 1 \\ T(n-1) + T(n-2) & , \text{otherwise} \end{cases}$$

$$T(n) = T(n-1) + T(n-2) + c$$
$$= 2T(n-1) + c$$

[$T(n-1) \sim T(n-2)$ from approximation]

$$T(n) = 2[2T(n-2) + c] + c$$
$$= 4T(n-2) + 3c$$
$$= 8T(n-3) + 4c$$
$$\vdots$$
$$2^k T(n-k) + (2^k - 1)c$$

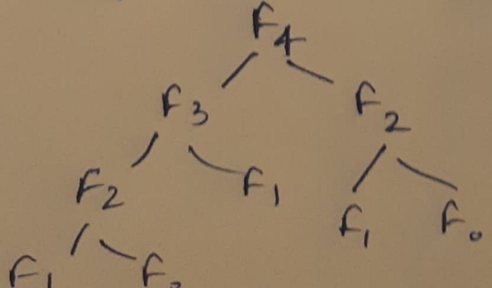
$$T(1) = 1$$
$$T(n-k) = T(1)$$
$$n = k + 1$$
$$\boxed{k = n-1}$$

$$T(n) = 2^{n-1} T(\cancel{n} - \cancel{n} + 1) + (2^{n-1} - 1)c$$

$$\boxed{\text{Time complexity} = O(2^n)}$$

<u>For space complexity</u>

Space required $\propto$ Max. depth of recursive tree



→ O(4)

→ For n elements

O(n)

Q3 Write programs which have complexity

1) n logn

```
A( )
{
  int i,j;

  for( i=1; i<=n; i++)          → O(n)
  {
    for( j=1; j<=n; j=j/2)      → O(logn)
    {
        printf("#");
    }
  }
}
```

2) n^3

```
A ( )
{
   int i,j,k;

   for( int i=0; i<=n; i++)        → O(n)
   {
      for( j=0; j<=n; j++)         → O(n)
      {
         for(k=0; k<=n; k++)       → O(n)
         {
            printf("*");
         }
      }
   }
}
```

Q4 Solve the following recurrence relation

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

$$T(n/4) \leq T(n/2)$$

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

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

Comparing with Master eqn we get

$a = 2$ $\quad$ $k = 2$ $\quad$ $a < b^k$

$b = 2$ $\quad$ $p = 0$

$e = \log_2 2 = 1$ $\quad$ $O(n^k \log^p n)$

$n < n^2$ $\quad$ $O(n^2 (\log n)^0)$

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

Q5 What is time complexity of following function

function fun()?

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

j incrementation depends on i,
we'll unroll all loops

$i = 1$ $i = 2$ $i = 3$ ------ $i = n$

$j = 1$ to n $j = 1$ to n $j = 1$ to n $j = 1$ to n

$\rightarrow n$ times $\rightarrow n/2$ times $\rightarrow n/3$ times $\rightarrow 1$ time

$$TC = n + \frac{n}{2} + \frac{n}{3} + \text{------} \frac{n}{n}$$

$$n\left(1 + \frac{1}{2} + \frac{1}{3} + \text{------} \frac{1}{n}\right)$$

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

Q6 Time complenity of :-

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

$i = 2$ $i = 2^k$ $i = (2^k)^k = 2^{k^2}$

'2 to n times'   '2^k to n times' 2^{k^2} to n times

$$i = 2^{k^{\log_k(\log n)}} = 2^{\log_2 n} = n$$

Total $\boxed{TC \rightarrow O(\log \log n)}$

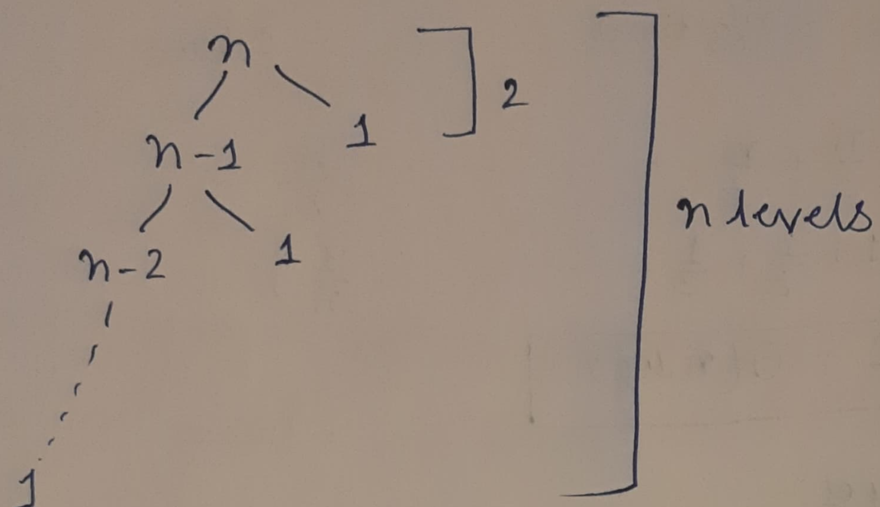
$$2^{k^l} = n$$

$$k^l = \log_2 n$$

$$l = \log_k \log_2 n$$

Q7

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



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

$= n \times n$

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

Lowest height = 2

Highest = n

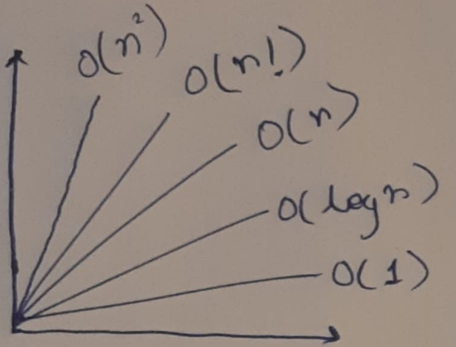
$\boxed{\text{Diff} = n-2}$, $n>1$

Given algo produces linear result.

Q8 Arrange the following in increasing order of rate of growth

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

$2^n, 2^{n^{2^n}}, 4^n, 100$



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

b)

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

c)

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