Tutorial 2.

Solution 1:

i	j
0	1
1	2
3	3
6	4
10	5

No. of times loop is running be k.

$S_k = 1+3+6+10+\dots+T_k$

$S_{k-1} = 1+3+6+\dots+T_{k-1}$

subtracting both

$S_k - S_{k-1} = 1+2+3+4+\dots+(k-1)$

$T_k = (k-1)k/2.$

Given that k^{th} term is n.

$T_k = n$

$k(k-1)/2 = n \Rightarrow k^2/2 - k/2 = n$ ignoring lower order terms and constants.

$\Rightarrow k^2 = n$

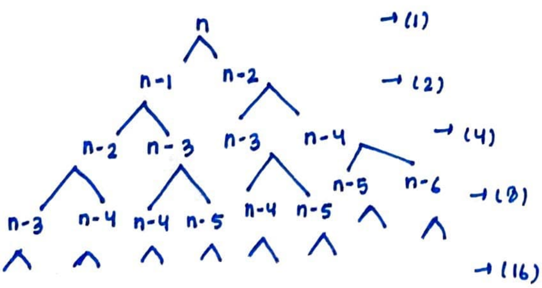
$k = \sqrt{n}$

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

Solution 2:

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

For recursive fibonacci solution



No. of times function is running will be sum of the series.

$S = 1+2+4+\dots+2^n$

$= 2^{n+1}-1/2-1 = 2^{n+1}-1$

Time complexity

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

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

Ignorings lower order terms :

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

usings Master Theorem:

$a = 1$, $b = 2$, $f(n) = n^2$

$c = \log_b a = \log_2 1 = 0$

$\boxed{0 < n^2}$ true.

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

Solution 3: Code having time complexity

$O(n \log n)$:

```
for (int i=1; i<=n; i++)
{
    for (int j=1; j<n; j=j*2)
        printf ("Hello");
}
```

$O(n^3)$:

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

$O(\log(\log n))$:

```
for (int i=2; i<=n; i=pow(i,3))
{
    printf ("Hello");
}
```

Solution 5 :

i	j
1	n
2	n/2
3	n/3
4	n/4
⋮	⋮

Time complexity will be sum of series

$$S = n/1 + n/2 + n/3 + \dots$$

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

$$\text{complexity} = n \times \sum_{i=1}^{n} (n/i)$$

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