Tutorial → 2

Q1

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

i	j
0	1
0+1	2
0+1+2	3
0+1+2+3	4
0+1+2+3+4	5
⋮	⋮
0+1+2 --k	k

Assume $i \approx n$

$\therefore\ i = k(k+1)/2$

$k(k+1)/2 = n$

$k = \sqrt{n} \Rightarrow O(\sqrt{n})$

Q2 Recursive fibonacci:

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

$$T(n) = \underbrace{T(n-1)}_{x} + \underbrace{T(n-2)}_{y}$$

$x = T(n-1)$

Master theorem for decreasing func.

Here $a=1$, $b=1$, $f(n) = n^0$

$\therefore\ x = \theta\ n$

$y = T(n-2)$

$=$

Let the constants be c.

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

$= 2T(n-c)$

Here $a=2$, $b=c$ and $f(n) = n^0$

$\therefore\ T.C = O(2^n)$

(2)

As we have only one statement to execute, we can draw a recursion tree, then height of tree will be the n. So, space complexity is $O(n)$

Q3 i) $TC = n\log n$

```
for(i=0; i<n; i++)                 -> n
{ for(j=0; j<n; j*=2)              -> logn
      statement;
}
```

$T.C = n\log n$

ii) $TC = n^3$

$T(n) = 8T(n/2) + n \longrightarrow O(n^3)$

```
void func(int n)
{   if (n>1)
    {
      func(n/2); func(n/2); func(n/2); func(n/2);
      func(n/2); func(n/2); func(n/2); func(n/2);     8T(n/2)
    } for(i=0; i<n; i++)                               -> n
        stmt;
}
```

$T(n) = 8(n/2) + n$

$\rightarrow O(n^3)$

iii) $\log(\log n)$

Q4 $T(n) = T(n/4) + T(n/2) + cn^2$

$T(n/2) >= T(n/4)$ i.e $T(n/2)$ is more effective term than $T(n/4)$.

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

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

Master thm. for dividing func.

$a=2,\ b=2,\ f(n) = n^2 = n^k;\ k=2$

$\log_b a = \log_2 2 = 1 < k$

$\therefore\ \Theta(n^2)$.

Q5
```
int fun(int n)
{ for(i=1; i<=n; i++)
    for (j=1; j<n; j+=i)
{   // Some O(1) task
}}}
```

i	j
1	n
2	n/2
3	n/3
n	n/n

$T.C = n + n/2 + n/3 + \ldots + n/n$

$= n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + 1/n\right)$

$T.C = n \log n$

Q6
```
for(i=2; i<=n; i=pow(i,k))
{
  // O(1)
}
```

$i \to 2,\ 2^k,\ (2^k)^k,\ (2^{k^2})^k = 2^{k^3} \ldots,\ 2^{k^{\log_k(\log n)}}$

let $i = n$

$2^{k^{\log_k \log n}} = n$

$2^{\log n} = n \Rightarrow k = O(\log(\log n))$

=

8

a) ~~100~~ $n, n!, \log^3 n, \log\log n, \sqrt{n}, \log n!, n\log n, \log^n n$
$\log^2 n, 2^n, 2^{2n}, 4^n, n^2, 100$

Sol^n $100 < \log\log n < \sqrt{n} < n < n\log n \leq \log n! < \log^2 n$
$< n^2 < 2^n < 4^n < 2^{2n}$.

b) $2 \cdot 2^n, 4n, 2n, 1, \log n, \log(\log n), \sqrt{\log n}, \log 2n,$
$2\log n, n, \log n!, n!, n^2, n\log n.$

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

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

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