④ Small O notation

$f(n) = o(g(n))$ ∴ $g(n)$ is upper bound of $f(n)$

if and only if

$f(n) < c\,g(n)$

$\forall\ n > n_0$ and for all constraints

$c > 0$

⑤ Small Omega Notation

$f(n) = \Omega(g(n))$ ∴ $g(n)$ is lower bound of $f(n)$

if and only if

$f(n) > c\,g(n)$

$\forall\ n > n_0$ and for all constraints

$c > 0$

Q2 What should be time complexity of

```
for (i = 1 to n) {i = i*2}
```

```
for ( int i = 1 ; i <= n ; )
{   ①          n/2 + 1
    i = i * 2 ;   n/2
}
```

$T(n) = n$

i: 1, 2, 4, 8, 16, . . . , n } K

$t_K = 1 \times 2^{K-1}$ (ar^{n-1} = g.p.)

$n = 2^{K-1}$

$$n = \frac{2^K}{2^1}$$

$$2n = 2^K$$

$$\log_2(2n) = K \log_2(2)$$

$$\log_2(n) + \log_2(2) = K \log_2(2)$$

$$\log_2(n) + 1 = K$$

$$O(\log_2(n))$$

$$T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ T(1) = 1 \end{cases}$$

Using forward substitution

$$T(2) = 3T(2-1)$$

$$= 3T(1) = 3$$

$$T(3) = 3T(3-1)$$

$$= 3T(2) = 3 * 3 = 9$$

$$T(4) = 3T(4-1) = 3 * T(3) = 3 * 9 = 27$$

$$T(n) = 1 + 3 + 9 + 27 + \ldots \} K$$

$$a r^{n-1}$$

$$T(n) = 1 \frac{r^K - 1}{3 - 1} = 1 \cdot \frac{3^n - 1}{2} = \frac{3^n}{3}$$

$$T(n) = O(3^n)$$

Q4 $T(n) = \{ 2T(n-1) - 1 \}$ if $n > 0$, otherwise 1

$T(1) = 1$

$T(2) = 2T(2-1) - 1$

$= 2T(1) - 1 = 2 - 1 = 1$

$T(3) = 2T(3-1) - 1$

$= 2T(2) - 1$

$= 2 - 1 = 1$

$T(4) = 2T(4-1) - 1$

$= 2T(3) - 1$

$= 2 - 1 = 1$

$T(n) =$ ~~[illegible]~~ n

$= O(n)$

Q5

```
int i=1 , s=1 ;          (1) (1)
While ( s<=n)            (n+1)
{
    i++;                 (n)
    s = s+i              (n)
    Printf ("#");        (n)
}
```

Total number of instruction

$= 1+1+n+1+n+n+n$

$= 3+4n$

$T(n) = O(n)$

Q6

```
void
{
}
```

i = 1

T

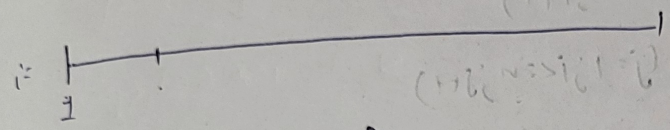
Q7:

```
void
{
```

Q6
```
void function(int n)
{
    int i, count = 0;
    for (i = 1 ; i*i <= n ; i++)
    {
        count++;
    }
}
```

$i^2 <= n$

$i <= \sqrt{n}$



$T(n) = O(n^{1/2})$

Q7.
```
void function(int n)
{
    int i, j, k, count = 0;
    for (i = n/2 ; i <= n ; i++)          - n/2
    {
        for (j = 1 ; j <= n ; j = j*2)    n/2
        {
            for (k = 1 ; k <= n ; k = k*2)  n/2
                count++;
        }
    }
}
```

$i = n/2$

$j = \log(n)$

$k = \log(n)$

Time Complexity - $O(n * \log^2(n))$

08 - Time Complexity of

```
function (int n)
{
    if (n==1)
        return;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf("*");
        }
    }
    function(n-3);
}
```

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

$= \frac{n(n+1)(2n+1)}{6}$ (sum of squares)

$= \frac{n^3}{6}\left(1+\frac{1}{n}\right)\left(2+\frac{1}{n}\right)$

$= \frac{n^3}{6}(2)$

$= \frac{n^3}{3}$

Time Complexity : $O(n^3)$

09 Time

void

010

Q9 Time Complexity

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

i=1 j => 1 to n

i=2 j => $\frac{n}{2}$ times

i=3 j = $\frac{n}{3}$ times

i=n j = 1 times

T(no. of instruction) = $n + \frac{n}{2} + \frac{n}{3} + \ldots + 1$

$= \log(n)$

$T(n) = \log_2(n)$

Q10 for the function n^K & c^n, what is the asymptotic relationship between these functions ? ($K >= 1$ & $c >= 1$)

n^K is $O(c^n)$