Example: find Time Complexity

```
int a=0, b=0, n, m;
cin>>n>>m;
for int(i=0; i<n; i++){ → Time Complexity = O(n)
    cout<<"Hi\n";
}
for (int i=0; i<m; i++){ → Time Complexity = O(m)
    cout<<"Hi2\n"; }
```

Time complexity = max(O(m), O(n)) i.e = O(N) + O(M)
= O(n) (as n > m)

Understanding the time Complexity

→ Estimation of total CPU computation required to execute an algorithm.

Time Complexity of an algorithm = Total No. of CPU Computations = Sum of frequency count of each instruction of an algorithm

↓

Number of times an instruction is executed

① Time Complexity of Single loop:

```
for(i=1; i<=n; i++)
    printf("Hello"); ← n times
```

= O(n)

② Single loop increment by a constant

```
for (i=1; i<=n; i=i+2)
    (out<< ("Hi");
```

$\frac{n-1}{2} + 1$ times $= O(n)$

$i = 1, 1+2, 1+2\times2 + 1+3\times2 \ldots, \underbrace{1+K\times2}$

$1+K\times2 = n$

$\Rightarrow K\times2 = n-1$

$\Rightarrow K = \frac{n-1}{2}$

$= O(n)$

③ Single loop decrement by constant

```
for (i=n; i>=1; i=i-2)
```

$\frac{n-1}{2} + 1$ times $= O(n)$

$i = n, n-2, n-2\times3, n-3\times2 \ldots, n-K\times2$

$n-K\times2 = 1$

$\Rightarrow K = \frac{n-1}{2}$

$= O(n)$

④ Single loop - Multiplying

```
for(i=1; i<=n; i=i*5)
```

$\log_5 n + 1$ times $= O(\log n)$

$i = 1, 5, 5^2, 5^3 \ldots 5^K$

$5^K = n$

taking log both side

$\log_5 5^K = \log_5 n$

$\Rightarrow K = \log_5 n$

$\Rightarrow O(\log n)$

⑤ * Single loop - Dividing the expression

for (i=n; i>=1; i=i/5) ← $\log_5 n$ + 1 times

$i = n, n/5, n/5^2, n/5^3 \ldots n/5^k$

$n/5^k = 1 \Rightarrow 5^k = n$

taking log both side,

$\log_5 5^k = \log_5 n$

$\Rightarrow k = \log_5 n$

$= O(\log n)$

⑥ * Single loop - Update expression with Power

for (i=3; i<=n; i=i^2) ← ($\log_2 \log_3 n$)+1 times

$i = 3, 3^2, 3^{2^2}, 3^{2^3}, \ldots, 3^{2^k}$

$3^{2^k} = n$

taking log both side,

$\log_3 3^{2^k} = \log_3 n$

$\Rightarrow 2^k = \log_3 n$

$\Rightarrow \log_2 2^k = \log_2 \log_3 n$

$\Rightarrow k = \log_2 \log_3 n \quad = O(\log \log n)$

2.) for (i=n; i>=5; i=√i)

$i = n, n^{1/2}, n^{1/2^2}, n^{1/2^3}, \ldots, n^{1/2^k}$

$n^{1/2^k} = 5$

taking log both side,

$\log_5 n^{1/2^k} = \log_5 5$

$\Rightarrow 1/2^k \cdot \log_5 n = 1$

$\Rightarrow 2^k = \log_5 n$

taking log both side

$\log_2 2^k = \log_2 \log_5 n \Rightarrow k = \log_2 \log_5 n$

$\Rightarrow O(\log \log n)$

(7) Single loop - Conditional statement with a function

1.) for (i=1; i<=2^n; i++) ← 2^n times

$i = 1, 2, 3, \ldots k$

$k = 2^n$

$O(2^n)$

2.) for (i=2; i<=2^n; i=i^2)

$i = 2, 2^2, 2^{2^2}, 2^{2^3}, \ldots 2^{2^k}$

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

$n = 2^k$

taking log both side

$\log_2 2^k = \log_2 n \Rightarrow k = \log_2 n$

$O(\log n)$

3) for (i=1; i<=√n ; i++)

i = 1, 2, 3, ... k

k = √n

= O(√n)

(8) Single loop - initialization Statement with a function

» for (i=n²; i>=1 ; i = i/2) ← 2log₂n + 1 times

$i = n^2, \frac{n^2}{2}, \frac{n^2}{2^2}, \frac{n^2}{2^3}, \ldots \frac{n^2}{2^k}$

$\frac{n^2}{2^k} = 1 \Rightarrow n^2 = 2^k$

$\log_2 n^2 = \log_2 2^k$

$\Rightarrow 2.\log_2 n = k \log_2 2$

$\Rightarrow 2\log_2 n = k$

$\Rightarrow O(\log n)$

Nested loop

Types of Nested loop

1.) Independent Nested loop 2.) Dependent nested loop

Independent Nested loop → Inner loop variable is independent of the outer variable.

- Time complexity of Independent Nested loop = Multiplication of frequency count of each loop

Example:

①
```
for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
        x = x+1;
```
} n times } n×n times

i=1	i=2	i=3	....	i=n
j = 1 to n	j = 1 to n	j = 1 to n	....	j = 1 to n

⇒ Time Complexity = $O(n^2)$

(ii)
```
for (i=1; i<=n; i++)        ---> n times
    for (j=1; j<=n; j++)    ---> n times
        for (k=1; k<=n; k++)---> n times
```

⇒ Time Complexity = $O(n^3)$

(iii)
```
for (i=3; i<=n; i=i^3)      ---> log_3 log_3 n times
    for (j=1; j<=n; j*2)    ---> log_2 n times
        for (k=1; k<=n; k*4)---> log_4 n times
```

⇒ Time Complexity = $O(\log^2 n \cdot \log \log n)$