Time Complexity Method #1

Ex :- (1)

doesn't Req

① for (i=0; i<n; i++) → n+1
{
statement; → n
}

O(n)

---

if decreamenting.

② for (i=n; i>0; i--) → n+1 (desn't Req)
{
Stmt; → n
}

Simple ex:- if you go to
1 to 10 (or) 10 to 1
it'll take 10 steps only.
So no matter of code if its
increamenting or decreamenting

1 to 10 (or) 10 to 1

* here (n+1) of for loop desn't effect the T/C

③

```
for (i=1 ; i<n ; i=i*2)
{
    stmt ;
}
```

stmt ; → $\frac{n}{2}$

$O(n)$

$f(n) = n/2 \rightarrow O(n)$

Even if the $f(n) = n/20$ still → $O(n)$

Ex:- ②

```
for (i=0; i<n; i++) → n+1
{
  for (j=0; j<n; j++) n*(n+1)
  {
    stmt : → (n × n)
  }
}
```

$f(n) = O(n^2)$

Ex:- (3)

```
for (i=0; i<n; i++)
{
    for (j=0; j<i; j++)
    {
        Stmt;
    }
}
```

i=0 i++	j=0 j<i	(no of times Exicute)
0	0✗	0
1	0✓ 1✗	1
2	0✓ 1✓ 2✗	2
3	0✓ 1✓ 2✓ 3✗	3
.	.	.
.	.	.
.	.	.
n	n	n

$f(n) = \frac{n(n+1)}{2}$

$f(n) = \frac{n^2+1}{2}$

degree n^2

$f(n) = O(n^2)$

How many no. of times it's exicuted

$1+1+2+3+4+\ldots+n = \frac{n(n+1)}{2}$

Ex 4

```
p=0
for (i=1; p<=n; i++)
{
    p=p+i;
}
```

Assume $p > n$

then only it will stop

i	p=p+i
1	0 ← initially
1	0+1
2	0+1+2
3	1+2+3+0
4	1+2+3+4
:	
K	1+2+3+4...+K

$K = \sqrt{n}$

$\Rightarrow$ Assume $p > n$ its stop

$\therefore p = \frac{K(K+1)}{2}$

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

$K = K^2$

$K^2 > n$

$K > \sqrt{n}$

$K = \sqrt{n}$

Time Complexity #2

Ex:- (5)

```
for (i=1 ; i<n ; i=i*2)
{
    Stmt ;
}
```

i = i*2
1
1×2 = 2
2×2 = 2^2
2^2×2 = 2^3
2^3×2 = 24
⋮
2^k

Assume $i >= n$

$i = 2^k$

$\therefore 2^k >= n$

$2^k = n$

$k = \log_2 n$

Time Complexity is $O(\log_2 n)$

$T/C = O(\log_2 n)$

From this we can observe that where the iterable down terminable i = multiplied by 2 then T/C would be ⇒ $O(\log_2 n)$

Ex:- (5) a,b

a)

```
for (i=1; i<n; i=i*2)
{
    Stmt;
}
```

i
1
1×2×2×2×2×2 .... = n

$i = 1 \times 2 \times 2 \times 2 \times 2 \times 2 \ldots = n$

$2^k = n$

T/C $k = \log_2 n$

b)

```
for (i=1; i<n; i++)
{
    Stmt;
}
```

i
1
1+1+1+1 ... +n

$k = n$

O(n) T/C

(5)

```
for (i=1 ; i<n ; i=i*2)
{
    Stmt ;
}
```

→ $(\log n)$ T/C.

* ⇒ log n may give decimal values, float value

n = 8

i
1

$n=8$ (i=1, i<n; i=i*2)

i=1	i<n	T/F	i=i*2	Output i
1	1<8	T	1*2	2
2	2<8	T	2*2	4
4	4<8	T	4*2	8
8	8<8	F	X Exit	Exit

if $n=10$

i=initially	i<n	T/F	i=i*2	Output i
i=1	1<10	T	1*2	2
i=2	2<10	T	2*2	4
i=4	4<10	T	4*2	8
i=8	8<10	T	8*2	16
i=16	16<10	F	X	X

$\log 8 = 3$

$\log_2 2^3 = 3 \log_2 2 = 3$

$\log_2 10 = 3.3$

It'll Give decimal value for 10

$\lfloor \log n \rfloor$ Real Value

ex:- ⑥

```
for (i=n; i>=1;
        i=i/2)
  {
    stmt;
  }
```

i starts from

n

$n/2$

$n/2^2$

$n/2^3$

⋮

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

assum $i<1$

$\therefore \frac{n}{2^k} < 1$

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

$n = 2^k$

$k = \log_2 n$

T/C $O(\log_2 n)$

ex:- ⑦

```
for(i=0; i*i<n; i++)
  {
    stmt;
  }
```

So Basically the loop Runs till the $i*i<n$

Remember Here Other words, don't make Required just the middle one which is Complex that is $i*i<n$

if this is Becomes $i*i>n$ then it will stop Terminate.

$i*i>n$

$i^2 = n$

$i = \sqrt{n} \rightarrow O(\sqrt{n})$

T/C

All T/C formula's in short :-

Time/Complexity

for (i=0; i<n; i++) = O(n)

for (i=0; i<n; i=i+2) = O(n/2) = O(n)

for (i=n; i>=1; i--) = O(n)

for (i=1, i<n; i=i*2) = O($log_2 n$)

for (i=1, i<n; i=i*3) = O($log_3 n$)

for (i=n; i>1; i=i/2) = O($log_2 n$)

nested loop

n ← for (i=0; i<n; i++) ⇒ incrementing

}

n × ($log_2 n$) for (j=0, j<n; j=j*2) = O(n log n)

{

n — log n

*

log(n)

stmt

} multiply by 2

}

2 log n + n

O(n log n)

ex :- (8)

```
for (i=0; i<n; i++)
  {
    stmt;  ⟶ O(n)
  }

for (j=0; j<n; j++)
  {
    stat;  ⟶ O(n)
  }
```

f(n) = 2n

f(n) = O(n)

ex :- (10)

```
for (i=0; i<n; i++)        ⟶ n
{
  for (j=1; j<n; j=j*2)    ⟶ n * log n
  {
    stmt ⟶ n * log n
  }
}
```

n

n × log n

n × log n

2 log n + n

T/C = O(n log n)

ex :- (9)

```
p=0
for (i=1; i<n; i=i*2)
{
  p++;  ⟶ log n
}
for (j=1; j<p; j=j*2)
  {
    stmt ⟶ log(p)
  }
```

log p = log n

log p

O (log . log n)