Time complexity - (ignore small powers and constants)

$f(n) = 3n^2 + 5n \longrightarrow O(n^2)$

$f(n) = n + 100 \log n \longrightarrow O(n)$

$f(n) = 3n^2 + 4n^5 \longrightarrow O(n^5)$

$f(n) = 10000 \longrightarrow O(1)$

Example

$O(n)$: $n/4$, $n+100/4$, $5n + 3\log n$

$O(n^2)$: $5n^2+7n+2$, $3n^2+\log n$, $n^2/1000$

constant time → $O(1)$ → for(i=0 to i=10){...}

linear time → $O(n)$ → for(i=0 to i=n){...}

logarithmic time → $O(\log n)$ → binary search

Quadratic time → $O(n^2)$ → for(1−n)① { for(1−n)② { } }

Cubic time → $O(n^3)$ →

```
for(1-n)①
{
    for(1-n)②  for(1-n)②
    {
    }
}
```

$O(1)$ $O(\log n)$ $O(n)$ $O(n \log n)$ $O(n^2)$ $O(n^3)$ $O(2^n)$ $O(n!)$ →

least ← Complexity. → highest

Ques.

```
int a=0, b=0;
for(i=0; i<N; i++) { ... }  → O(N)
for(j=0; j<M; j++) { ... }  → O(M)
```

Time complexity $= O(N+M)$

Ques.

```
for(i=0; i<N; i++) {  → N
    for(j=0; j<N; j++) { a = a+j; }  → N   } → O(N^2)
}
for(k=0; k<N; k++) { b = b+k; }  → O(N)
```

$\therefore TC = O(N^2) + O(N) = O(N^2)$

Que.

```
for(i=0; i<N; i++){
    for(j=N; j>i; j--){ a=a+i+j;}
}
```

(outer loop → N, inner loop → N; j: N → 0)

$\therefore TC = O(N^2)$

Que `for(i=1; i<n; i=i+2) { statement; }`

→ $n/2 = O(n)$

Que.

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

i	p
1	0+1=1
2	1+2=3
3	3+3=6
4	6+4=10
⋮	⋮
k	1+2+3+....+k

Assume $p > n$

and $p = k(k+1)/2$

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

$k^2 > n$

$k > \sqrt{n}$

$\therefore O(\sqrt{n})$

Que. `for(i=1; i<n; i=i*2){ print; }`

i
1
1×2=2
2×2=2^2
2^2×2=2^3
⋮
2^k

assume $i >= n$

$\therefore 2^k >= n$

$2^k = n$

$k = \log_2(n)$

$\therefore TC = O(\log_2 n)$

Que. `for(i=n; i>=1; i=i/2){ stat; }`

i
n
$n/2$
$n/2^2$
⋮
$n/2^k$

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

$n = 2^k$

$k = \log_2 n$

$\therefore TC = O(\log_2 n)$

Date : / /
Page :

Que. `for( i=0; i*i<n; i++){ statu; }`

$i * i >= n$

$i^2 = n$

$i = \sqrt{n}$

$\therefore TC = O(\sqrt{n})$

Que `p=0;`

`for( i=1; i<n; i=i*2) { p++; }` — $\log_2 n = p$

`for( j=1; j<p; j=j*2) { statu; }` — $\log p$

$\log(\log_2 n)$

$\therefore TC = O(\log \log n)$

Que.

```
for(i=0; i<n; i++) {                    // n
    for(j=1; j<n; j=j*2) { statu; }     // log2n
}
```

$\therefore TC = O(n \log_2 n)$

* **Summary :**

1. `for( i=0; i<n; i++)` — $O(n)$
2. `for( i=0; i<n; i=i+2)` — $n/2$ — $O(n)$.
3. `for( i=n; i>1; i--)` — $O(n)$
4. `for( i=1; i<n; i=i*2)` — $O(\log_2 n)$
5. `for( i=1; i<n; i=i*3)` — $O(\log_3 n)$
6. `for( i=n; i>1; i=i/2)` — $O(\log_2 n)$

Page :

Stuck in TLE ?

10^8 operation rule → most of the modern machine can perform 10^8 operation/second.

constraints (n)	Time complexity (worst case)
$< [10..11]$	$O(n!)$, $O(n^6)$
$< [15..18]$	$O(2^n * n^2)$
< 100	$O(n^4)$
< 400	$O(n^3)$
< 2000	$O(n^2 * \log_2 n)$
$< 10^4$	$O(n^2)$
$< 10^6$	$O(n \log_2 n)$
$< 10^8$	$O(n)$, $O(\log n)$, $O(1)$