Today's Content

1) Time Complexity & Space Complexity
2) Assymptotic Analysis (Big O)
3) Big (O) Meaning
4) Time Limit Exceed (TLE)

} Time Complexity - II

Today's Content

1) How to Calculate Iterations

Quiz:1 : $N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \rightarrow \cdots \rightarrow 1$

$= \log_2 N$ steps

Quiz:2 : [3 10] : 3 4 5 6 7 8 9 10

8 Numbers

$10 - 3 + 1 = 8$

Number of Elements from [a b] both Included = $b - a + 1$

[→ Included (→ Excluded

[3 10) = 3 4 5 6 7 8 9 ~~10~~

[a b) = b - a (a, b] = b - a (a b) = b - a - 1

A.P : Arithmetic Progression

A.P : 4 7 10 13 16 ---

+3 +3 +3 +3

a, a+d, a+2d, a+3d, ----

d d d d

a - first Term
d - common difference
N - Number of Terms

Sum of N Terms in A.P : $\frac{N}{2}[2a + (N-1)d]$

G.P

$\frac{20}{10} = 2$ $\frac{80}{40} = 2$

// Common ratio is same

5 10 20 40 80 160 ---

$\frac{10}{5} = 2$ $\frac{40}{20} = 2$

$a \quad ar \quad ar^2 \quad ar^3 \quad ar^4 \ldots$

r r r r

a - first Term
r - common ratio
N - Number of Terms

Sum of N Terms in G.P $= \begin{cases} \frac{a \times (r^N - 1)}{r - 1} & r \neq 1 \\ & r > 1 \end{cases}$

log Basic :

$\log_a a^n = n$ | $\log_2 2^{10} = 10$ | $\log_{10} 10^{10} = 10$

1)

```
void fun(int N){
    s = 0
    for(i = 1; i <= N; i++){
        s = s + i
    }
    return s
}
```

$i = [1 \quad N]$ # iterations?

$= N - 1 + 1 = N$ iterations

$\Rightarrow O(N)$

2)

```
void func(int N, int M)
    for(i = 1; i <= N; i++){
        if(i % 2 == 0){
            print(i)
        }
    }
    for(i = 1; i <= M; i++){
        if(i % 2 == 0){
            print(i)
        }
    }
}
```

$i = [1, N] \Rightarrow N$ iterations

$i = [1, M] \Rightarrow M$ iterations

Total $= (N + M)$ iterations

$\Rightarrow O(N + M)$

(Q3)
```
int fun(int N){
    s = 0
    for(i = 1; i <= N; i = i+2){
        s = s + i
    }
    return s
}
```

i
1
3
5
7
9
11
:

$i = i + 2$, $i = i + 2$, $i = i + 2$ → Odd Numbers

// Iterations = Number of Odd Numbers from [1 N]

$N = 7 : \{1, 3, 5, 7\} = 4$ ✓ $\quad N = 6 : \{1, 3, 5\} = 3$ ✓

$N = 7 \Rightarrow [7+1]/2 = 4$ ✓ $\quad N = 6 \Rightarrow [6+1]/2 = 7/2 = 3$ ✓

// Odd Numbers [1 N] = $\frac{(N+1)}{2} \Rightarrow \frac{N+1}{2} \Rightarrow \frac{N}{2} + \frac{1}{2} : O(N)$

(Q4)
```
int fun(int N){
    s = 0
    for(i = 0; i <= 100; i++){
        s = s + i + i^2
    }
    return s
}
```

$i = [0 \rightarrow 100]$

$= 100 - 0 + 1 = 101$ iterations

Irrespective of N it is always 101 iterations

⇒ Constant Iterations : $O(1)$

Q5)

```
void fun(N){
    s=0{i=1; i*i<=N; i++}
    for(i=1; i*i<=N; i++){
        s=s+i^2
    }
    return s
}
```

$i * i <= N$

$i^2 <= N$

$i <= \sqrt{N}$

$i = [1, \sqrt{N}]$ (a, b) $\Rightarrow \sqrt{N} - 1 + 1$

$\Rightarrow \sqrt{N}$ iterations

$\Rightarrow O(\sqrt{N})$

Q6)

```
void fun(N){
    int i = N
    while(i>1){
        i = i/2
    }
}
```

Before i	Iteration	After $i = i/2$
N	1	$N/2 : N/2^1$
N/2	2	$N/4 : N/2^2$
N/4	3	$N/8 : N/2^3$
N/8	4	$N/16 : N/2^4$

$i = N \to N/2 \to N/4 \to N/8 \to \cdots 1$

// After (k) iterations, say it stops

$N/2^k = 1$

$i = N/2^k = 1$

$N/2^k = 1$

$N = 2^k$

$2^k = N$

Apply $\log_2$ on both sides

$\log_2 2^k = \log_2 N$

$k = \log_2 N$

// Total iterations $\Rightarrow \log_2 N$

$\Rightarrow O(\log N)$

$N \geq 1$:

$10/2 \rightarrow 5/2 \rightarrow (2)/2 \rightarrow (1)$

$12/2 \rightarrow 6/2 \rightarrow (3)/2 \rightarrow (1)$

} Any +ve Integer keep dividing by 2 you will reach 1

2 → 1

3 → 1

```
void fun(N){
    Note: N is a +ve Integer
    S = 0
    for(i=0; i<N; i=i*2){
        S = S+i
    }
}
```

Before	Iteration	After
i = 0	1	i = 0
i = 0	2	i = 0
i = 0	3	i = 0
i = 0	4	i = 0

i = i*2

∞ loop

```
void fun(N){
    Note: N is a +ve Integer
    S = 0
    for(i = 1; i < N; i = i*2){
        S = S + i
    }
}
```

Before	Iterations	After i (i = i*2)
1	1	2 : 2^1
2	2	4 : 2^2
4	3	8 : 2^3
8	4	16 : 2^4

$i = 1, 2, 4, 8, \ldots\ldots, N$

$\log N$

1) After k iterations : 2^k

2) Assume after k iterations loop stops

$2^k = N$ ⟶

Apply $\log_2$ on both sides

$k = \log_2 N$

After $\log_2 N$ iterations loop will breaks.

10:30PM : + break

Nested loops 1

```
void func(N){
    for(i=1; i<=10; i++){
        for(j=1; j<=N; j++){
            print()
        }
    }
}
```

Table → Only loop variable

i	j	
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
:		:
10	[1 N]	N
11	X	10 N

→ $O(N)$

```
void fun2(N){
    for(i=1; i<=N; i++){
        for(j=1; j<=N; j++){
            print(i,j)
        }
    }
}
```

Table, i, j

i	j	
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
:		:
N	[1 N]	N
N+1	X	N^2 N

→ $O(N^2)$

```
void fun3(N) {
    for(i = 0; i <= N; i++) {
        for(j = 0; j <= i; j++) {
            print(i * j)
        }
    }
}
```

i=0:
  j = 0 to 0 } : 1

i=1;
  j = 0 to 1 } : 2

i=2;
  j = 0 to 2 } : 3

Table → i & j $[a\ b] = b - a + 1$

i	j : [0 i]	Iterations
0	[0 0]	0 - 0 + 1 = 1
1	[0, 1]	1 - 0 + 1 = 2
2	[0 2]	2 - 0 + 1 = 3
3	[0 3]	3 - 0 + 1 = 4
⋮	⋮	⋮
N-1	[0 N-1]	N-1-0+1 = N
N	x	$\frac{(N)(N+1)}{2}$

→ $\frac{N^2 + N}{2}$

→ $\frac{N^2}{2} + \frac{N}{2}$

→ $O(N^2)$

```
void funy(N) {
    for(i = 1; i <= N; i++) {
        for(j = 1; j <= N; j = j * 2) {
            print(i * j)
        }
    }
}
```

i	j	
1	[1 N]	$\log_2^N$
2	[1 N]	$\log_2^N$
3	[1 N]	$\log_2^N$
i		
N	[1 N]	$\log_2^N$
N+1	×	N log N

$\Rightarrow O(N \log N)$

```
void funs(N) {
    for(i = 1; i <= 2^N; i++) {
        print(i)
    }
}
```

$i : [1 \quad 2^N] : 2^N$ times $\Rightarrow 2^N$

$\Rightarrow$ Iterations $= 2^N$

$\Rightarrow O(2^N)$

```
void func(N) {
    for(i = 1; i <= N; i++) {
        for(j = 0; j <= 2^i; j++) {
            {print(i, j)}
        }
    }
}
```

i	j : [1, 2^i]	
1	j : [1, 2]	$2 = 2^1$
2	j : [1, 2^2]	$4 = 2^2$
3	j : [1, 2^3]	$8 = 2^3$
4	j : [1, 2^4]	$16 = 2^4$
⋮	⋮	⋮
N	j : [1, 2^N]	$2^N = 2^N$

i : 1

j : 1 → 2 : 2

i : 2

j : 1 → 4 : 4

Total Iterations =

$$2^1 + 2^2 + 2^3 + 2^4 + \cdots + 2^N$$

Sum of N Terms in GP

$$= a \times \left(\frac{r^N - 1}{r - 1}\right)$$

$a = 2$

$r = 2$

N Terms

$$= \frac{2 * (2^N - 1)}{2 - 1}$$

$$= \frac{2 * (2^N - 1)}{}$$

$$= 2 * 2^N - 2 \} = O(2^N)$$

How to Calculate Big O Notation

1) Calculate No: of Iterations

2) Neglect lower order terms

3) Neglect constant coefficient terms

Except highest order
If Term, neglect others

Eg1: Iterations = $4N^2 + 3N + 1$

↳ Neglect

↳ $4N^2 \rightarrow O(N^2)$

Eg2: $100N^2 + 32N^3 + 51N + 1007$

↳ $32N^3 \rightarrow O(N^3)$

Eg3: $3N\sqrt{N} + 4\log N + 31N\log N$

↳ $3N\sqrt{N} \rightarrow O(N\sqrt{N})$

Comparison Table

$$O(1) < \log_2 N < \sqrt{N} < N < N\log N < N\sqrt{N} < N^2 < 2^N < N!$$

$N = 32$

$\log_2^{32}$	$\sqrt{32}$	32	$32 * \log_2^{32}$	$32 * \sqrt{32}$	32^2	2^{32}	$32!$
5	5.65	32	160	181	1024	$4 * 10^9$	

$> 4 * 10^9$

- $\log N$ – logarithmetic
- N – linear
- $N \log N$ – linear logarithmetic
- N^2 – Quadratic
- 2^N – Exponential

TODO

```
void funs(N) {
    for(i=1; i<=N; i=i+2){
        for(j=1; j<=i; j++){
            print()
        }
    }
}
```

Doubts:

`for(i=0; i<100; i++)` $i : [0, 100] \rightarrow \{100 - 0 + 1\} \rightarrow 101$

`for(i=1; i<=N; i++)` $i : [1, N] \rightarrow \{N - 1 + 1\} \rightarrow N$

- `i++` , `i = i+2`
- `i--` ; `i = i-2`
- `i *= 2` ; `i = i*2`
- `i /= 2` ; `i = i/2`

```
void fun6(N) {
    for(i=0; i<N; i++){
        for(j=N; j>=i; j--){
            print()
        }
    }
}
```

i	j : [N, i+1]	Total

// Sum of ∞ terms in GP, $(r < 1)$

⇒ $\left\{ \frac{a}{1-r} \right\}$; a is first term, r = common ratio

```
for(i=N; ; i=i/2)
    for(j=1; j<i; j++){
        print()
    }
{j=1; j<=0}
```

i	j [1 N]	
N	: [1 N)	N
N/2	: [1 N/2]	N/2
N/4	[1 N/4]	N/4
⋮		⋮
1	[1 1]	1

$N + N/2 + N/4 + \cdots$

$N\left[1 + 1/2 + 1/4 + 1/8 + \cdots\right]$, $r = 1/2$

0 (1) times

0 1 times

0 1 times