Today's Content

- → Big O
- → TLE - (Time Limit Exceed Error)
- → Why TLE ?

→ 2nd Session

Today's Session : Calculating no: of iterations

Basic Maths Revision

Quiz 1 : Sum of Natural Numbers : $\frac{N(N+1)}{2}$

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

corners included

$$\boxed{\text{Quiz 3 : } [a \ b] : b - a + 1}$$

Series : 3 6 12 24 48 96 ....

$2 = 6/3 = 12/6 \quad 24/12 = 48/24 \ldots 96/48 = 2$

common ratio

geometric progress

common ratio : r

1	2	3	4	5	.....	N^{th}
a	ar	ar^2	ar^3	ar^4		ar^{N-1}

last term

$$\boxed{\text{Sum of N Terms in GoP} = a * \left\{ \frac{r^N - 1}{r - 1} \right\}}$$

$r \neq 1$

Q1

```
void func(int N){
    int s = 0;
    i = 0 ; i <= 100; i++){
        s = s + i
    }
}
```

$b - a + 1 \Rightarrow$

$i : [0, 100] \Rightarrow 100 - 0 + 1 = 101$

$\rightarrow O(1)$

Q2)

```
void func(int N){
    int s = 0;
    i = 35; i < = 87; i = i + 1)
        s = s + i
}
```

$i : [35 \quad 87] = 87 - 35 + 1$

$= 52 + 1$

$= 53$

$\rightarrow O(1)$

Q3)

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

$i : [1 \quad N] : N - 1 + 1 \Rightarrow N$

N iteration

$\rightarrow O(N)$

Q4)

```
void func(int N, int M){
    int s = 0;
    i = 1; i <= N; i++){
        if( i % 2 == 0){ ---> N times
            s = s + i
        }
    }
    i = 1; i <= M; i++){
        if( i % 2 == 1){ ---> M times
            s = s + i
        }
    }
}
```

→ $(N+M)$

→ $O(N+M)$ ✓

Q5)

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

Q): How many odd numbers from [1 N] / → (Integer)

$N = 9 \rightarrow (9/2) \rightarrow (4)$ *

$N = 9 \rightarrow \left(\frac{9+1}{2}\right) \rightarrow (5)$

$N = 12 \rightarrow (12)/2 \rightarrow (6)$

$\rightarrow \left(\frac{12+1}{2}\right) \rightarrow (6)$

$\left[\frac{(N+1)}{2}\right]$

$O(N)$

N = i

9 [1 3 5 7 9] odd numbers from [1 9]

12 i [1 3 5 7 9 11] odd number from [1 12]

N i [1 . ___] odd numbers from [1 N]

Q6)

```
void func(N){
    int s=0
    i=1; i*i<=N; i++){
        s=s+i
    }
}
```

$i^2 <= N \Rightarrow i <= \sqrt{N}$

$i: [1, \sqrt{N}] \Rightarrow [\sqrt{N} - \cancel{1} + \cancel{1}] \Rightarrow \sqrt{N}$

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

Q7)

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

iteration	i Value after (i = i/2)
1	$N/2 \rightarrow i = (N/2^1)$
2	$N/4 \rightarrow i = (N/2^2)$
3	$N/8 \rightarrow i = (N/2^3)$
4	$N/16 \rightarrow i = (N/2^4)$
5	$N/32 \rightarrow i = (N/2^5)$

After k iterations $i = (N/2^k)$

$N = 10$

$i = 10 \rightarrow 5 \rightarrow 2 \rightarrow 1 > 1$: (break)

$10 > 1$ ✓, $5 > 1$ ✓, $2 > 1$ ✓

N

$i = N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \rightarrow N/16 \cdots$ — $i == 1$ Break

After k iterations it will break

$i = 1$

$N/2^k = 1$

$\log_a a^x = x$

$2^k = N \longrightarrow \log_2(2^k) = \log_2(N)$

// $\log_2()$ function on both sides

$k = \log_2 N$

// After $\log_2 N$ iterations it will break $\rightarrow O(\log_2 N)$

8Q) void fun(int N){

```
    int s = 0
    i = 0; i <= N; i = i*2){
        s = s + i
    }
}
```

iteration	i value after ($i = i*2$)
1	0
2	0
3	0
4	0

i not getting updated & $0 <= N$: T

∞ loop

$O(\infty \text{ loop})$

9Q)

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

Iterations	i value after ($i = i*2$)
1	2 → ($i = 2^1$)
2	4 → ($i = 2^2$)
3	8 → ($i = 2^3$)
4	16 → ($i = 2^4$)
5	32 → ($i = 2^5$)

// After k iterations

$i = 2^k$

$N = 8$ $\quad i = 1 \rightarrow 2 \rightarrow 4 \rightarrow 8 == 8$

$i < 8 \quad i < 8 \quad i < 8 \quad$ Break

$i == N$ (Break)

After k iterations it Breaks

$i = N$

$\boxed{2^k = N} \Rightarrow \boxed{k = (\log_2 N)}$

$\rightarrow O(\log N)$

10Q) void func( ){

```
    i = 1; i <= 4; i = i+1){
        j = 1; j <= 3; j = j+1){
            print("Hello World")
        }
    }
}
```

i	j	
1	[1 3]	(3 times) +
2	[1 3]	(3 times) +
3	[1 3]	3 times +
4	[1 3]	3 times
5		12 times

$O(1)$

11Q)

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

i	j	iterations
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
⋮		
10	[1 N]	N
11		10N iterations

$O(N)$

12)

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

i	j	iterations
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
⋮		⋮
N	[1 N]	N
		N^2

$O(N^2)$

13)

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

input

i	j : [1 i]	iterations
1	[1 1]	1
2	[1 2]	2
3	[1 3]	3
4	[1 4]	4
⋮		
N	[1 N]	N

N+1

Sum of N Natural Numbers $= \frac{(N)(N+1)}{2}$

$= \frac{N^2+N}{2}$

$= \frac{N^2}{2} + \frac{N}{2}$

$= O(N^2)$

14)

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

i	j	iterations
1	[1 N] : j=j*2	$(\log_2 N)$
2	[1 N] : j=j*2	$(\log_2 N)$
3	[1 N] : j=j*2	$(\log_2 N)$
⋮		
N	[1 N] : j=j*2	$(\log_2 N)$
N+1		

$N \log_2 N$

Break

$\rightarrow O(N \log N)$

Q15)

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

$i \in [1, 2^N] \rightarrow 2^N$ iterations

$\rightarrow O(2^N)$

Q16) void func(N){

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

i	j : [1 2^i]	iterations
1	j : [1 2]	2^1
2	j : [1 2^2]	2^2
3	j : [1 2^3]	2^3
⋮		
N	j : [1 2^N]	2^N

Total iterations ⇒

$$S = 2^1 + \cancel{2^2} + \cancel{2^3} + \cancel{2^4} + \cancel{\ldots} \cancel{2^N}$$

$$2S = \cancel{2^2} + \cancel{2^3} + \cancel{2^4} \cancel{2^5} + \cancel{\ldots} \; 2^{N+1}$$

$$S = 2^{N+1} - 2 \Rightarrow 2 * 2^N - 2 \Rightarrow 2(2^N - 1) \Rightarrow O(2^N)$$

TODO:

Apply formula & check ?

9:45 break

Q17)

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

i	j : [1 i]	iterations
N	j : [1 N]	N
N/2	j : [1 N/2]	N/2
N/4	j : [1 N/4]	N/4
N/8	j : [1 N/8]	N/8
1	j : [1 1]	1
0		

$$2S = 2N + \cancel{N} + \cancel{N/2} + \cancel{N/4} + \cancel{N/8} \cdots \cancel{1} \cancel{2}$$

$$S = \cancel{N} + \cancel{N/2} + \cancel{N/4} + \cancel{N/8} + \cancel{N/16} \cdots \cancel{1}$$

$$\boxed{S = 2N - 1} \rightarrow O(N)$$

Comparism functions :

// functions : { for large N value }

$$\boxed{\log_2(N) < \text{sqrt}(N) < N < N\log N < N\sqrt{N} < N^2 < N^3 < 2^N}$$

// N^2 N^3 → lower order

// $N\log N$ $\log N$ → lower order

// $N\sqrt{N}$ $N\log N$ → lower order

Big O Notation Basics → What? / Why/Where? } → Next Session

→ How to calculate ?

How to Calculate Big (O) for a Code

1) → Calculate iterations

2) → Neglect all lower order terms

3) → Neglect Constant Coefficient

Ex1: Iterations → $3N^2 + 5N + 10^4$ → $O(N^2)$

↳ lower order terms

Ex2: Iteration → $5N^2 + 10N^3 + 6N\log N + 100$ → $O(N^3)$

lower

Ex3: Iterations → $4N^2 + 3N + 10^6$ → $O(N^2)$

Ex4: Iterations → $4N + 3N\log N + 10^6$ → $O(N\log N)$

Ex5 Iteration → 10^3 → Constant iterations → $O(1)$

{ Indicates that it is constant }