Today's Content

T.C → 2 { → Big O - O( )
→ TLE - (Time limit Exceeded error)
→ why TLE ?

TC → 1 { How to calculate iterations } → 15/17 Question

← Quote →

Don't tell your god how big your storm is.
Tell your storm how big your god is.

Basic maths Revision

Quiz 1: Sum of Natural numbers: → Quiz

$S_N = 1 + 2 + 3 + \dots N = \frac{N(N+1)}{2}$

Quiz 2: $[3 \quad 10] \rightarrow \{3,4,5,6,7,8,9,10\} \rightarrow$ 8 elements

$[4 \quad 8] = \{4,5,6,7,8\} \rightarrow$ 5 elements

No. of elements from [a b] included

Quiz 3: $[a \quad b] = b - a + 1$ # imp

Gp basics: → geometric Progressions. → [when we divide any 2 cons. elements, ratio have to be same]

$S_1 = 3 \quad 6 \quad 12 \quad 24 \quad 48 \quad 96 \dots$

ratio $= \frac{6}{3} = \frac{12}{6} = \frac{24}{12} = \frac{48}{24} = \frac{96}{48} \dots 2$

$S_2 = 2 \quad 6 \quad 18 \quad 54 \quad 162$

ratio $= \frac{6}{2} = \frac{18}{6} = \frac{54}{18} = \frac{162}{54} \dots 3$

// given G.P.

first term = a

common ratio r

1st	2nd	3rd	....	N-2nd	N-1th	Nth
a	ar	ar^2	ar^3 ..... ar^{N-3}		ar^{N-2}	ar^{N-1}

// Given S_N = sum of N terms of G.P.

$$S_N = a + ar + ar^2 + ar^3 + \dots ar^{n-3} + ar^{n-2} + ar^{n-1}$$

Multiply both sides with r.

$$rS_N = \cancel{ar} + \cancel{ar^2} + \cancel{ar^3} + \cancel{ar^4} + \dots \cancel{ar^{n-2}} + \cancel{ar^{n-1}} + ar^n$$

$$- \quad S_N = a + \cancel{ar} + \cancel{ar^2} + \cancel{ar^3} + \dots \cancel{ar^{n-3}} + \cancel{ar^{n-2}} + \cancel{ar^{n-1}}$$

$$rS_n - S_n = ar^n - a$$

$$S_n(r-1) = a(r^n - 1)$$

$$\boxed{S_n = \frac{a(r^n-1)}{r-1}}$$

Ex

$a = 5,\ r = 2,\ N = 5$

$\rightarrow$ 5, 10, 20, 40, 80 = 155

$$S_5 = \frac{5(2^5-1)}{2-1} \Rightarrow 5 \times 31$$

$\Rightarrow$ 155

// log basics :

$\log_b^a = c \iff b^c = a \rightarrow$ To what power we need to raise b to get a.

$\log_2(8) = 3$

$\log_3(81) = 4$

$\log_2(2^5) = 5$

$\log_a(a^x) = x$

$\boxed{n = 2^k, \quad k = \log_2(n)}$

Q1

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

i: [0 100] → 101

101 iterations.

Q2)

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

i : [35 87] → 87-35+1

= 53

\# 53 iterations.

Q3:

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

i: [1 N] ⇒ N-~~1~~+~~1~~ → N

\# N iterations.

Q4:

```
void func(int N) {
    int S = 0;
    (i=1; i<=N ; i++) {                 i -> [1 N] => N-1+1 => N
        if (i%2==0) {                   # N iterations,
            S = S+i
        }
    }
    (i=1; i<=M ; i++) {                 i -> [1 M] => M-1+1 => M
        if (i%2==1) {                   # M iterations,
            S = S+i
        }
    }
}
```

Total iterations $\underline{N+M}$

Q 6)

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

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

$i : [\overset{a}{1} \quad \overset{b}{\sqrt{N}}] \to \sqrt{N} - \cancel{1} + \cancel{1} \to \sqrt{N}.$

$\to$ # $\sqrt{N}$ iterations.

Q7) $i = N$

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

iterations	i value after iteration
1	$i = i/2$ $N/2$: $N/2^1$
2	$i = i/2$, $N/4$: $N/2^2$
3	$i = i/2$, $N/8$: $N/2^3$
4	$i = i/2$, $N/16$: $N/2^4$
5	$i = i/2$, $N/32$: $N/2^5$

After k iterations.

$$\left[i = \frac{N}{2^k} \right]$$

N	i
10	10 → 5 → 2 → 1 {Break}
19	19 → 9 → 4 → 2 → 1 {Break}

Obs1 :- when ever i reaches 1 it breaks.

Ans. It breaks after y iterations.

$i = 1$ && $i = \frac{N}{2^y}$ ⇒ $\frac{N}{2^y} = 1$

$\Rightarrow N = 2^y$

$\Rightarrow y = \log_2(N)$

After $\log_2 N$ iterations code breaks.

Ques)

```
void func (int N) {
    int i = N
    while (i>0) {
        i = i/2
    }
}
```

→ Todo & Try it out

→ No. of iterations

Q8)

```
void func(int N) {
        int S = 0;
    (i = 0; i <= N; i = i*2) {
            S = S + i
    }
}
```

$i = 0$

iterations	i value after every iteration	
1	i = i*2	0
2	i = i*2	0
3	i = i*2	0
	⋮	

∞ loops

10-10 pm → 10:17 pm

90) → { Near approximations }

i = 1

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

iterations	i value after each iteration
1	$i = i*2$, $i = 2 \rightarrow 2^1$
2	$i = i*2$, $i = 4 \rightarrow 2^2$
3	$i = i*2$, $i = 8 \rightarrow 2^3$
4	$i = i*2$, $i = 16 \rightarrow 2^4$

After k iterations,

$i = 2^k$

$i > N$, it'll break

Ass., Say after k iterations, loop breaks.

$i = 2^k$, $i > N$

$2^k > N$.

$k > \log_2(N) \Rightarrow k = \log_2 N + 1$

↓

For Breaking

10Q)

```
void func (int N) {
    (i = 1 ; i <= 4; i = i+1) {
        J = 1; J <= 3; J++) {
            Print ("Hello World");
        }
    }
}
```

Table :-

i	J	Total iterations
1	[1 : 3]	3
2	"	3
3	"	3
4	"	3
5	Break	12 iterations

11Q)

```
void func (int N) {
    (i = 1 ; i <= 10; i = i+1) {
        (J = 1; J <= N; J++) {
            Print ("Hello World");
        }
    }
}
```

i	J	Total iterations
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
	⋮	
10	[1 N]	N
11	breaks	

No. of iterations → 10 N

12)

```
void func (int N) {
    (i = 1 ; i <= N ; i = i+1) {
        (J = 1 ; J <= N ; J++) {
            Print ("Hello World");
        }
    }
}
```

i	J	Total iterations
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
⋮		
N	[1 N]	N
N+1	break	

Total iterations = N^2

13)

```
void func (int N) {
    (i = 1 ; i <= N ; i = i+1) {
        (J = 1 ; J <= i ; J++) {
            Print ("Hello World");
        }
    }
}
```

i	J	Total iterations
1	[1 1]	1
2	[1 2]	2
3	[1 3]	3
⋮		⋮
N	[1 N]	N
N+1	→ break	

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

Iterations

14)

```
void func (int N) {
    (i=1 ; i<=N; i=i+1) {
        (J=1; J<=N ;J=J*2) {
            Print ("Hello World");
        }
    }
}
```

i	J	Total iterations
1	[1 N]	$\log_2 N+1$
2	[1 N]	$\log_2 N+1$
3	[1 N]	⋮
⋮		
N	[1 N]	$\log_2 N+1$

$(n(\log_2 N+1))$

Q15)

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

i $[1\ \ 2^n] \to 2^n-1+1$ X

$\Rightarrow 2^n$ iterations.

Q 16)

```
void func (int N) {
    (i=1 ; i<=N; i=i+1) {
        (J=1; J<=2^i; J=J+1) {
            Print ("Hello World");
        }
    }
}
```

i	J	Total iterations
1	[1 2]	2^1
2	[1 2^2]	2^2
3	[1 2^3]	2^3
⋮		⋮
N	[1 2^N]	2^N

Total Iterations $= 2(2^n - 1)$

← Geometric Progression →

$a, \quad ar, \quad ar^2, \quad ar^3 \; \ldots \; ar^{n-1}$

$a = 2, \quad r = 2$

$$S_n = \frac{a(r^n - 1)}{r - 1} \Rightarrow \frac{2(2^n - 1)}{1}$$

$$\Rightarrow 2(2^n - 1)$$

Q 17

```
void func (int N) {          → Todo
    (i=N ; i>0 ; i= i/2) {
        (J = 1 ; J<=i ; J++ ) {
            Print(" Hello World");
        }
    }
}
```

Comparison functions : { large N values }

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

N^2 N^3

$n\sqrt{n}$ n^2 } → Always pick higher order term, others can be ignored.

Big O → what ?

→ why ?

→ where do we use it ?

→ How do we calculate Big O for any code

→ calculate iterations

→ Only take higher order term

→ Neglect constant coefficients.

Ex1:- iterations $\rightarrow$ $3N^2 + 5N + 10^4$ $\rightarrow$ $O(N^2)$

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

Ex3: iterations $\rightarrow$ $4N^2 + 3N + 10^6$ $\rightarrow O(N^2)$

Ex4: iterations $\rightarrow$ $4N + 3N\log N + 10^6$ $\rightarrow O(N\log N)$

Ex5:- iterations $\rightarrow$ 10^3 $\rightarrow$ Constant iterations $\rightarrow$ $O(1)$

```
— func(N) {
    (i=0; i<=9; i++) {
        print()
    }
}
```

i: [0 9] $\rightarrow$ $9-0+1 = 10$

10 iterations
$\downarrow$
Constant

Todo:- For all the questions of class, calculate their O(__)