1) Time complexity & Space complexity
2) Asymptotic Analysis (Big O)
3) Big O → meaning
4) TLE

} TC - 2

Finding T.C needs analysis of no. of iterations

Agenda: count the no. of iterations.

Maths Concepts

QUIZ 1

1) $\log_2 N$ → No. of times we need to divide N by 2 to reduce it to 1.

[a, b] → a, b are included

(a, b) → a, b are not included

QUIZ 2 Nos. in range [3, 10]

3, 4, 5, 6, 7, 8, 9, 10 → 8

Nos. in [a, b] ⇒ b - a + 1

[1, 7] ⇒ 7 - 1 + 1 = 7

[12, 23] ⇒ 23 - 12 + 1 = 12

Arithmetic Progression [AP]

$4, 7, 10, 13, 16, \ldots$

$3 \quad 3 \quad 3 \quad 3$

$a, a+d, a+2d, a+3d, a+4d, \ldots$

$d \quad d \quad d \quad d$

$a \Rightarrow$ first term

$d \Rightarrow$ common difference

Terms $\rightarrow N$.

Sum of first N terms of AP $= \frac{N}{2}[2a + (N-1)d]$

Geometric Progression [GP]

Series of terms, where every adjacent term has common ratio

$5, 10, 20, 40, 80, 160, \ldots$

$2 \quad 2 \quad 2 \quad 2 \quad 2$

Generalise: $a, ar, ar^2, ar^3, ar^4, \ldots$

$r \quad r \quad r \quad r$

$a \rightarrow$ first term

$r \rightarrow$ common ratio

$N \rightarrow$ no. of terms

Sum of GP : $\frac{a(r^N - 1)}{r - 1}$ $\quad r > 1$

$\frac{a(1 - r^N)}{1 - r}$ $\quad r < 1$

log Basics

$\log_a x$ ⇒ No. of times we need to divide x by a till it reaches 1.

$$\log_a a^x = x$$

$\log_2 2^{10} = 10$ | $\log_{10} 10^{12} = 12$

Que.

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

$i \rightarrow [1 \quad N]$

Iterations : $N - X + 1$
$= N$

* NO QUIZ present for this

Que.

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

$i \rightarrow [1 \quad N]$

Iterations → N

$j \rightarrow [1 \quad M]$

Iterations → M

Total no. of iterations : $N+M$

Que:

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

i	
1	i = i+2
3	i = i+2
5	i = i+2
7	
.	
.	

odd

No. of iterations
= No. of odd nos. from [1 N]

How many are there from [1 N] ?

N=7 {1 2 3 4 5 6 7} ⇒ 4

N=6 {1 2 3 4 5 6} ⇒ 3

No. of odd nos. b/w [1 N] ⇒ $\frac{(N+1)}{2}$

[1 3] : $\frac{3+1}{2} = 2$

[1 8] : $\frac{9}{2} = 4.5 = 4$

Que:

```
int func(N) {
    S=0
    for(i=0 ; i<=100 ; i++) {
        S = S+ i + i^2
    }
    return S
}
```

i : [0 100]

100-0+1 = 101 ans

Que.

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

$i * i <= N$

$i^2 <= N$

$i <= \sqrt{N}$

$i : [1 \ \sqrt{N}]$

Iterations : $\sqrt{N} - 1 + 1 = \sqrt{N}$

Que.

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

i			
Before	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	$N/2^4$
		⋮	
		1	$N/2^K$

2 → 1, 3 → 1

$N \geq 1$

$\frac{10}{2} \to \frac{5}{2} \to \frac{2}{2} \to 1$

$\frac{12}{2} \to \frac{6}{2} \to \frac{3}{2} \to 1$

$i : N \to \frac{N}{2} \to \frac{N}{4} \to \frac{N}{8} \to \dots 1$

K iterations

$\frac{N}{2^K} = 1$

$N = 2^K$

$\log_2 N = \log_2 2^K$

$K = \log_2 N$

Say now instead to go from $N \to 1$

we want to go $1 \to N$ [multiplying by 2]

$1 \to 2 \to 4 \to 8 \to 16 \dots N$ (K steps)

$1 \to 2^1 \to 2^2 \to 2^3 \to 2^4 \dots 2^K$

$2^K = N$

$K = \log_2 N$

Eg $N = 32$

$1 \to 2 \to 4 \to 8 \to 16 \to 32$

$2^1 \ 2^2 \ 2^3 \ 2^4 \ 2^5$

$2^5 \to 2^K$

K steps → 5 steps

Que.

```
void func(N) {
    for(i=0; i<=N; i=i*2) {
        print(i)
    }
}
```

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

No. of Iterations → ∞

Que.

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

i		
Before	Iteration	After
1	1	2
2	2	4
4	3	8
8	4	16

i = 1 → 2 → 4 → 8 → 16 → ... N

$\log_2 N$

Break !

Nested Loops

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

x ← for(i=1; i<=10; i++){
y ← for(j=1; j<=N; j++){
print(i*j)
}
}
}

1 ⟶ y

x ⟶ x*y

i	j	Inner loop Iteration
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
.		
.		
.		
10	[1 N]	N
		10 x N

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

i	j	Inner loop Iterations
1	[1 N]	N
2	[1 N]	N
3	[1 N]	N
.	.	.
.	.	.
.	.	.
N	[1 N]	N
		N^2

Que.

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

i	j	iterations
1	1 → N	$\log_2 N$
2	1 → N	$\log_2 N$
.		
.		
.		
N	1 → N	$\log_2 N$

$N * \log_2 N$

Que.

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

$i \rightarrow [1, 2^n]$

#iterations $= 2^n$

Que.

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

i	j	iterations
	iterations	
1	[1 2]	2^1
2	[1 2^2]	2^2
3	[1 2^3]	2^3
4	[1 2^4]	2^3
5	[1 2^5]	2^4
.	.	.
N	[1 2^N]	2^N

sum of GP

$2^1 + 2^2 + 2^3 + \ldots . 2^N$

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

How to calculate Big O Notation

1) Calc no. of iterations
2) Neglect all lower order terms
3) Neglect constants from higher order term

eg① #iterations ⇒ $4N^2 + 3N + 1$ → $O(N^2)$

Quiz

eg② #iterations → $4N^2 + 3N + 10^6$ → $O(N^2)$

eg③ #iterations:

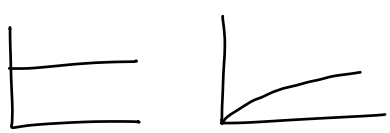
$100N^2 + 32N^3 + 51N + 1007$

eg④ #iterations:

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

#Calc BigOH for above iterations

Complexities precedence



N=32

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

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