Today's agenda

- No. of iterations
- Big O Notations

- The magic you are looking for is in the work you are avoiding.

Quiz

↳ How many numbers are in range [3,10] (Corners included)

{ 3 4 5 6 7 8 9 10 } → 8

[a,b] → b-a+1	[a,b) → b-a	(a,b) → b-a-1
inc inc	inc exc	exc exc

//log basics

$x * 2 = 10 \Rightarrow x = 10/2$

$x + 2 = 10 \Rightarrow x = 10 - 2$

$x^3 = 27 \Rightarrow x = \sqrt[3]{27}$

$x^2 = 16 \Rightarrow x = \sqrt{16}$

$x^2 = 16 \rightarrow 2 = \log_x 16$

$ans = \log_2 10$

→ $\log_a b$ — b: number; a: base of log; log: representation

→ $\log_2 10 = ans$

$10 = 2^{ans} \rightarrow 2^{3.33}$

∴ ans = 3.33...

* $\log_b a = ans$ (a & b are given) $\Rightarrow a = b^{ans}$ → then hit & trial for ans.

(i) $\log_2 64 = ans$

$64 = 2^{ans} \rightarrow 2^6 = 64 \Rightarrow ans = 6$

(ii) $\log_3 343 = ans$

$343 = 3^{ans} \Rightarrow ans = 5.2...$

(iii) $\log_2 33 = ans$

$33 = 2^{ans} \Rightarrow ans = 5.something$

Properties:

↳ (I) $\log_a a^n = n$

↳ (II) $\log_c (a*b) = \log_c a + \log_c b$

ex: $\log_3 10 = \log_3 5 + \log_3 2$
$\downarrow$
$(*2)$

Quiz

↳ No. of steps for $N \to \frac{N}{2} \to \frac{N}{4} \to \dots\dots \to 1$

$$\left(\left(N * \frac{1}{2}\right) * \frac{1}{2}\right) * \frac{1}{2} \dots\dots = 1$$

No. of steps == How many times you can multiply by $\frac{1}{2}$ before reaching 1.

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

↳ $N = 2^K \Rightarrow 2^K = N$

$K = \log_2 N$

A.P → Arithmetic Progression

3 7 11 15 19 23 . . . .

+4 +4 +4 +4 +4

first term = a : Starting number, a = 3

Common diff = d : difference bet^n cons. no.s, d = 4

1 2 3 4 nth term

(a+0*d) (a+d) (a+2d) (a+3d) . . . . a+(n-1)*d

↳ Sum of first N terms of A.P = $\frac{N}{2}[2*a+(N-1)*d]$

G.P → geometric Progression

3 6 12 24 48 . . . .

*2 *2 *2 *2

50 25 12.5 6.25

*1/2 *1/2 *1/2

first term = a : Starting number a = 3

Common ratio = r : multiple to get next no.

1 2 3 4 nth term

ar^0 $a*r$ $a*r^2$ $a*r^3$ . . . . ar^{N-1}

↳ Sum of first N terms of G.P = $a * \frac{r^N-1}{r-1}$

Quiz

```
int Sum = 0;
for (int i=1; i<=N; i++){
    Sum = Sum + 1;
}
```

[1, N] → N-1+1

N iterations

O(N)

Quiz

```
void func (int N, int m){
    for (int i=1; i<=N; i++){
        Point(i);
    }
    for (int i=1; i<=M; i++){
        Point(i);
    }
}
```

→ [1, N] = N-1+1

N iterations

→ [1, M] = M-1+1

M iterations

Total = N+M iterations

O(N+M)

N > M → O(N)

N < M → O(M)

Qui2

```
int fun (int N){
    int S=0;
    for (int i=0; i<=100; i++){
        S= S+ i2;
    }
    return S;
}
```

→ [0, 100]

↳ 100 − 0 + 1

↳ 101 iterations

↓

O(1)

Qui2

```
void fun (int N){
    int S=0;
    for (int i=1; i*i<=N; i++){
        S= S+ i2;
    }
    return S;
}
```

→ [1, √N] → √N − 1 + 1

= √N iterations

↓↓

O(√N)

Quiz

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

n → n/2 → n/4 → n/8 → ... → 1

→ log n iterations → O(log n)

Quiz

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

→ [0, 0, 0, 0 - ____] infinite

Quiz

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

→ [1 → 2 → 4 → 8 ... ——— n]

↳ logn iterations

↓

O(logn)

n → n/2 → n/4 → n/8 → ... → 1

1 → 2 → 4 → 8 → ... → n

Break till 9:25 pm

P1 →

P2 →

P3 →

P4 →

P5 →

→ 25

Nested loops

Quiz

```
void fun (int N){
    int s=0;
    for(int i=1; i<=10; i++){
        for(int j=1; j<=N; j++){
            s=s+10;
        }
    }
}
```

i	j	count
1	[1,N]	N
2	[1,N]	N
3	[1,N]	N
⋮		
10	[1,N]	N
		10*N iteration

$\downarrow$ O(N)

Quiz

```
void fun (int N){
    int s=0;
    for(int i=1; i<=N; i++){
        for(int j=1; j<=N; j++){
            s=s+10;
        }
    }
}
```

i	j	count
1	[1,N]	N
2	[1,N]	N
3	[1,N]	N
⋮	⋮	⋮
N	[1,N]	N
		(N*N) iterations

$\downarrow$

$O(N^2)$

Qui2

```
void fun (int N){
    int s=0;
    for(int i=1; i<=N; i++){
        for (int j=1; j<=i; j++){
            s=s+10;
        }
    }
}
```

i	j	count
1	[1,1]	1
2	[1,2]	2
3	[1,3]	3
⋮	⋮	⋮
N	[1,N]	N

$\frac{N*(N+1)}{2}$ iterations

↳ $O(N^2)$

Qui2

```
void fun (int N){
    for(int i=1; i<=2^N; i++){ → [1, 2^N] → 2^N iterations
        Print(i);
    }
}
```

↳ $O(2^N)$

Qui2

```
void fun (int N){
    int s=0;
    for(int i=1; i<=N; i++){
        for(int j=1; j<=2^i; j++){
            s=s+10;
        }
    }
}
```

i	j	count
1	$[1, 2^1]$	2^1
2	$[1, 2^2]$	2^2
3	$[1, 2^3]$	2^3
⋮	⋮	⋮
N	$[1, 2^N]$	2^N

Sum of first N terms of G.P $= a * \frac{r^N - 1}{r - 1}$

$2^1 + 2^2 + 2^3 + 2^4 - - - - - - - + 2^N$

(*2, *2, *2)

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

$\Downarrow$

$O(2^N)$

Comparison of iteration → $N = 10^5$

$1 < \log_2 N < \sqrt{N} < N < N\log N < N\sqrt{N} < N^2 < 2^N$

Time Complexity

↳ Big O Notation → Approx iteration count

(i) Calculate iteration.
(ii) around '+' sign, neglect lower order term.
(iii) Neglect Constants.

iteration count

eg: ~~$20N^2$~~ N^2 + ~~$20N$~~ + ~~$30N^0$~~

⇓

$O(N^2)$

eg: ~~$2N^2$~~ + $15N\log N$ + ~~$20N$~~

⇓

$O(N^2)$

ex: ~~10~~ $N\log N + 1.5N + 6$

$\downarrow$

$O(N\log N)$

↳ No. of iterations for same Code is different

$N = 20$, $K = 100$

```
for (int i=1; i<=N; i++){
    if (i==K){
        break;
    }
}
```

↳ 2 iterations

↳ 10 iterations

↳ N iterations

Big O notation will Consider the worst case Scenario.

↳ $O(N)$