Today's agenda

- No. of iterations
- Big O Notations

Quiz

↳ How many numbers are in range [3,10] {Corners included}

{ 3 4 5 6 7 8 9 10 }

[a,b] = (inc, inc)	[a,b) = (inc, exc.)	(a,b) = (exc, exc.)

//log basics

$n * 2 = 10 \rightarrow n = 10/2$

$n + 2 = 10 \rightarrow n = 10 - 2$

$n^3 = 27 \rightarrow n = \sqrt[3]{27}$

$n^2 = 16 \rightarrow n = \sqrt{16}$

$n^2 = 16 \rightarrow 2 = \log_n 16$

→ $\log_a b$ ← number

↑ base of log

en: $2^{ans} = 10$ ⟶ $ans = \log_2 10$

↓

$ans = 3.33...$

(a & b are nos.)

* $\log_b a = ans$ ⇒ $a = b^{ans}$

(hit and trial for ans)

(i) $\log_2 64 = ans$

$64 = 2^{ans}$

↳ $ans = 6$

(ii) $\log_3 343 = ans$

$343 = 3^{ans}$

↳ $ans = 5.\ldots$

Properties:

(i) $\log_a a^n = n$

(ii) $\log_c (a*b) = \log_c a + \log_c b$

ex: $\log_2 10 = \log_2 (2*5) = \log_2 2 + \log_2 5$

Quiz

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

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

$$N * \frac{1}{2^{\text{no. of steps}}} = 1$$

$$N = 2^{\text{no. of steps}}$$

$$\log_2 N = \text{no. of steps}$$

A.P → Arithmetic Progression

3 7 11 15 19 23 .....

+4 +4 +4 +4 +4

first term = a = Starting no. → a = 3

Common difference = d = diff. betⁿ cons. nos → d = 4

1 2 3 4 Nth term

(a+0d) (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 - - - - -

*½ *½ *½

first term = a = Starting no. → a = 3

Common ratio = r = multiplier to get next no. → r = 2

1 2 3 4 nth

a ar ar^2 ar^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++){ → [1, N] → N-1+1
    Sum = Sum + 1;
}
```

N iterations

↓

O(N)

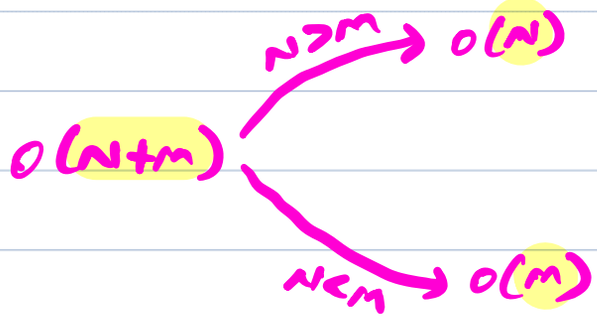
Quiz

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

N + iterations

M iteration

Total: N+M iterations



Quiz

```
int fun (int N){
    int S=0;
    for (int i=0; i<=100; i++){ → [0, 100] → 100-0+1
        S= S+ i2;                  ↓
    }                           101 iterations
    return S;                      ↓
}                               O(1)
```


Quiz

```
void fun (int N){
    int S=0;
    for (int i=1; i*i<=N; i++){ → [1, √N] → √N-1+1
        S= S+ i2;                  ↓
    }                           √N iterations
    return S;                      ↓
}                               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 ... ____]

infinite

Quiz

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

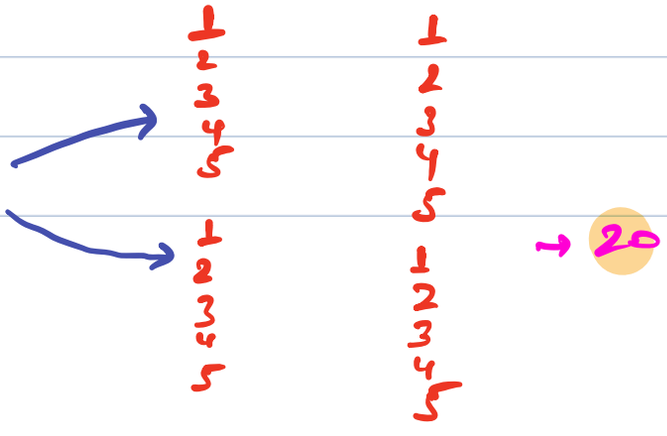
1 → 2 → 4 → 8 ... $\frac{n}{4}$ → $\frac{n}{2}$ → n

$\log_2 n$ iterations

$O(\log n)$

P1 P2

Break till 10:38 PM



Nested loops

Qui2

```
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 iterations

O(n) ←

Qui2

```
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
⋮	⋮	⋮
n	[1,n]	+ n
		n*n iterations

O(n²) ←

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

$\frac{N^2+N}{2} = \frac{N^2}{2} + \frac{N}{2} = O(N^2)$

Qui2

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

→ $[1, 2^N]$ → 2^N iterations

$O(2^N)$

Quiz

```
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

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

$a = 2$

$r = 2$

no. of terms → N

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

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

$= 2 * (2^N - 1)$ iterations

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

Comparison of iteration → $N = 10^5$

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

Time Complexity → Approx iteration Count
↳ Big O Notation

(i) Calculate iteration Count
(ii) around + Count, neglect lower order term. → keep highest order term
(iii) neglect constants

ex: iteration count: ~~1~~N^2 + ~~2~~N + ~~3~~

↓

$O(N^2)$

ex: ~~1~~N^2 + 15N~~log$N$~~ + ~~2~~N

↓

$O(N^2)$

ex: ~~1~~nlogn + ~~3~~n + ~~5~~

↓

O(nlogn)

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