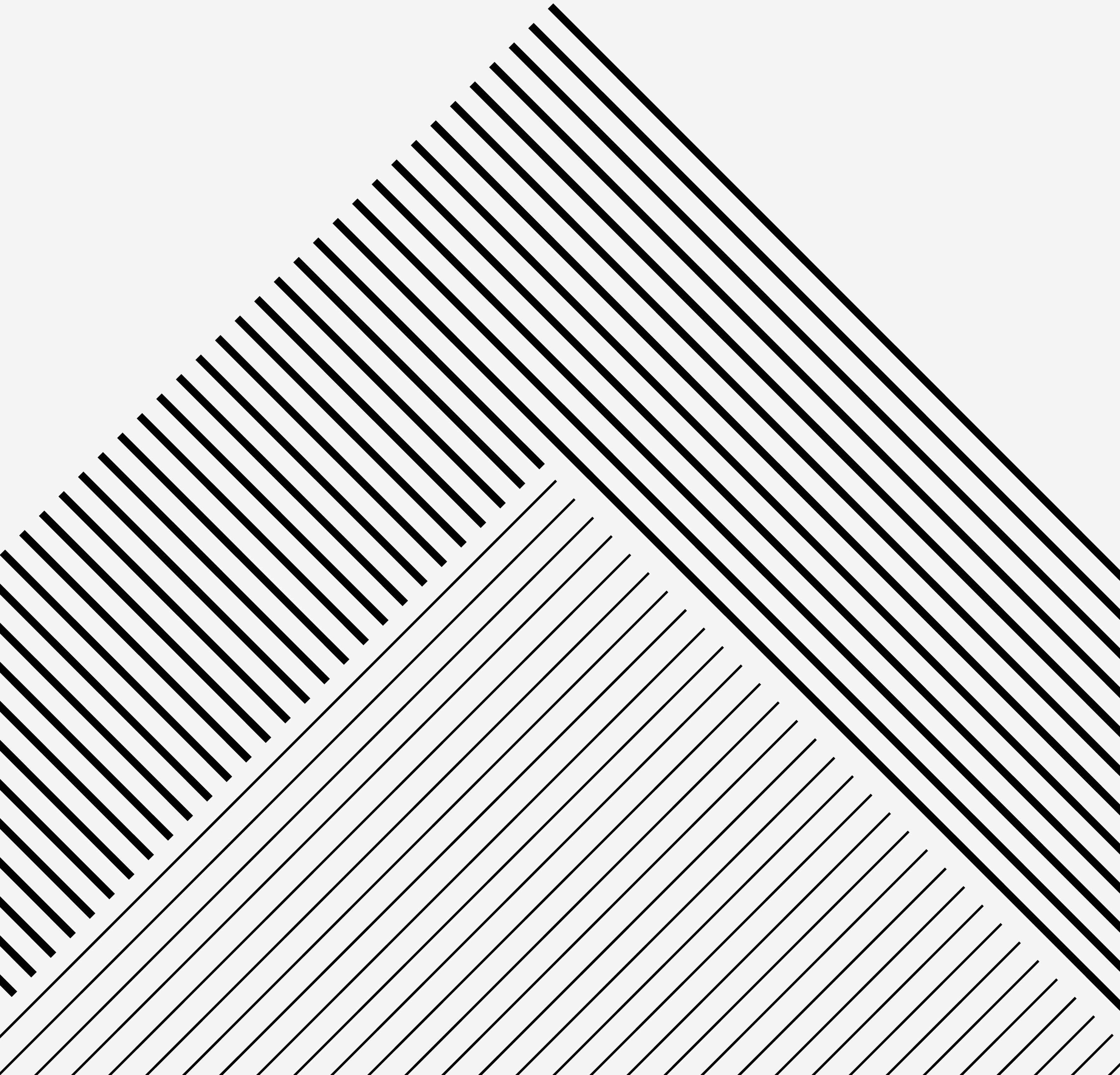

Quiz

1. How many numbers are in the range [3, 10] (corners included)?

(a) included

$[a, b]$

$\{3, 4, 5, 6, 7, 8, 9, 10\}$

$\boxed{\text{Formula} = b - a + 1}$

(b) inc exc

$[a, b)$

$\boxed{\text{Formula} = b - a}$

(c) exclusive

(a, b)

$\boxed{\text{Formula} = b - a - 1}$

Log basics

Log - inverse of an exponent

Some known facts.

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

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

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

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

If in $x^2 = 16$, we wanted to send the base of an exponent to RHS, then we use log.

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

generalizing — $\log_a b$

$\log_a b$ → number

a → base of log

$\log_2 10$ - to calculate a log we convert to powers.

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

$\log_2 10 = ans$

$10 = 2^{ans}$

$10 = 2^{3.33}$

ans = 3.33

$\log_b a = ans$

$a = b^{ans}$ → a and b would be given, so hit and trial for answer.

$\log_2 64 = ans$

$64 = 2^{ans}$

$64 = 2^6$

$\therefore ans = 6$

$\log_3 243 = ans$

$243 = 3^{ans}$

$243 = 3^5$

$\therefore ans = 5$

$\log_2 33 = ans$

$33 = 2^{ans}$

$33 = 2^{5.5}$

$\therefore ans = 5.5$

Properties of log

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

$a^n = a^{ans}$

$ans = n$

$\therefore \log_a a = n$

(b) $\log_c (a \times b) = \log_c a + \log_c b$

ex:- $\log_3 10 = \log_3 5 + \log_3 2$

we have seggregated 10 into one of its pair of factors (5, 2) in this case.

Quiz

2. How many number of steps for $N = N/2 + N/4 + N/8 + \ldots + 1$

Here, N is multiplied by ½ every time.

∴ no. of steps = number of times you multiply by ½ before you reach 1

$$(((N * \frac{1}{2}) * \frac{1}{2}) * \frac{1}{2}) * \ldots\ldots 1 = 1$$

if 2 multiplied K times it is 2^K

for the above equation it's multiplied by $\frac{1}{2^K}$ times

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

$$N = 2^K$$

$$2^K = N$$

$$K = \log_2 N$$

Arithmetic Progression (A.P)

Basically, you start from a specific number and keep adding the same number to each of the next elements in the series.

Ex: $\underbrace{3 \quad 7}_{+4} \underbrace{\quad 11}_{+4} \underbrace{\quad 15}_{+4} \underbrace{\quad 19}_{+4}, \underbrace{\quad 23}_{+4}$

let first term = a = 3

common difference (diff b/w two consecutive no^{rs}) - d = 4

generalizing,

$(a + 0*d) \quad (a+d) \quad (a+2d) \quad (a+3d) \quad \ldots\ldots \quad \overset{n^{th}\text{ term}}{(a+(n-1)d)}$

$\therefore$ sum of first N terms of A.P = $\frac{N}{2}(2a + (N-1)*d)$

Geometric Progression (G.P)

Basically, you start from a specific number and keep multiplying by the same number with each of the next elements in the series.

Ex: $\underbrace{3 \quad 6}_{*2} \underbrace{\quad 12}_{*2} \underbrace{\quad 24}_{*2}, \underbrace{\quad 48}_{*2} \underbrace{\quad 96}_{*2}$

In G.P, we can divide as well.

Ex: 50 25 12.5 6.25

$*\frac{1}{2}$ $*\frac{1}{2}$ $*\frac{1}{2}$

first term = a = 3

common ratio = r = multiple to get next number

generalising,

ar^0 ar^1 ar^2 ar^3 ar^4 . . . . . ar^{n-1} (n^{th} term)

$\therefore$ sum of first N terms of G.P = $\boxed{a * \frac{(r^n - 1)}{r-1}}$

Quiz

1.
```
for (int i=1 ; i<=N ; i++) {
        sum += i;
}
```
number of iterations → $N - \cancel{1} + \cancel{1}$ → N → O(N)

2.
```
void func(int N, int M) {
    for (i=1 ; i<=N; i++){
        print(i);
    }
    for (i=1 ; i<=M; i++){
        print(i);
    }
```
$[1, N] = N - \cancel{1} + \cancel{1} = N$ iterations

$[1, M] = M - \cancel{1} + \cancel{1} = M$ iterations

Total no. of iterations = N+M iterations → O(N+M)

$O(N+M) \xrightarrow{N>M} O(N)$

$\xrightarrow{M>N} O(M)$

3.

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

$[0, 100]$

$100 - 0 + 1$

101 iterations

$O(1)$ for greater values of N (10^5) and the no of iterations are completely independent of N and small as well

4.

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

$\rightarrow [1, \sqrt{N}] = i^2 <= N$

$i <= \sqrt{N}$

$\sqrt{N} \cancel{-1} \cancel{+1} = \sqrt{N}$ iterations

$O(\sqrt{N})$

5.

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

Here, we divide N by 1/2 every time.

So, it would be log N iterations

$O(\log N)$

6.
```
void fun(int N){
    int S=0;
    for(int i=1 ; i<=N; i*=2){
        S=S+i;
    }
}
```

values of i → [1 → 2 → 4 → 8 . . . . . N]

in reverse order it would look like

$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \rightarrow \frac{N}{8} \rightarrow \frac{N}{16} \rightarrow \ldots\ldots \rightarrow 1$

same

so it would be $\log N$ iterations. → $O(\log N)$

7.
```
void fun(int N){
    int S=0;
    for(int i=0 ; i<=N; i*=2){
        S=S+i;
    }
}
```

Here, the value of i would be zero always so this runs infinitely.

Nested loops

Quiz

1)
```
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
4	[1, N]	N
5	[1, N]	N
⋮		⋮
10	[1, N]	N

N added 10 times

∴ Total iterations - (10 * N) - O(N)

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

same as above, for each iteration of i, we perform N iterations of j.

∴ N is added N times - (N*N) iterations

- O(N*N)

- $O(N^2)$

3. void fun(int N) {

```
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 +
4	(1,4)	4 +
5	(1,5)	5
.		. +
.		.
.		.
.		.
N	(1,N)	N

sum of first N natu-ral numbers

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

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

4.

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

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

$O(2^N)$

5.

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

i	j	count
1	(1,2)	2^1
2	(1,4)	2^2
3	(1,8)	2^3
4	(1,16)	2^4
5	(1,32)	2^5
.		.
.		.
.		.
N	$(1,2^i)$	2^N

use G.P to find number of iterations

$$2^1 + 2^2 + 2^3 + 2^4 + 2^5 + \ldots\ldots + 2^N$$

$a = 2 \quad r = 2$

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

$O(2^N)$

Comparison of iterations

$$1 < \log N < \sqrt{N} < N < N\log N < N\sqrt{N} < N^2 < N^3 < 2^N$$

Time complexity is nothing but counting number of iterations

Big-O notation: - Approximate iteration count

To write Big-O notation of any code:

a. Calculate iteration

b. around "+" sign, neglect lower order terms.

c. neglect constants

Ex:- a) $\cancel{20}N^2 + \cancel{20N} + \cancel{300} \Rightarrow O(N^2)$

b) $\cancel{6}N^2 + \cancel{15 N\log N} + \cancel{20N} \Rightarrow O(N^2)$

c) $\cancel{10}N\log N + \cancel{15\sqrt{N}} + \cancel{60} \Rightarrow O(N \cdot \log N)$

Cases where break and return statements are used:

There are possibilities the number of iterations for same code might be different.

Ex:-

```
int N = 20;   int K = 2;
for(i = 1 ; i <= N ; i++){
    if (i == K)
        break;
}
```

depending on the input size, same loop executes differently. In such cases, we consider the worst case scenarios.

In worst case, the above code is O(N).