Few Terms that you shall see/hear throughout the course:

PSP (Problem Solving Percentage) - Solved Assignment Problems / Total Open Assignment Problems

- There are two types of section - Assignment and Additional. Assignment section consists of implementation of the problems done in class. PSP is calculated based on only Assignment Problems.
- Additional Problems are slight modifications of assignment problem, they are not part of PSP but once you're done with assignment, we highly recommend to complete additional problems as well.
- Try to keep PSP least 85% no matter what. It shall really help you to stay focused and we have seen in the past that people with >= 85%, do well in Interviews.

Attendance

- Try to maintain at-least 75% attendance either through live classes or by watching recording.
- Though I will recommend you to come to classes regularly because otherwise it may create backlogs.
- So, I expect all of you to attend live classes and if for any reason you are unable to, then please send me a message stating the reason.

Intermediate Module Description

- Introduction to Problem Solving
- Time Complexity
- Introduction to Arrays
- Prefix Sum
- Carry Forward
- Subarrays
- 2D Matrices
- Sorting Basics
- Hashing Basics
- Strings Basics
- Bit Manipulation Basics
- Interview Problems
- Contest [covers Full Intermediate DSA]

Note:

In Intermediate, we shall be learning the concepts around different topics and how to work with certain data structures.

- This module is dedicated to make you comfortable with Programming.

Contest will be organised after Intermediate Module.

- It'll will be for 1.5 hours and will be conducted within class duration followed by Contest Discussion (Instructor shall be discussing contest problems).
- It'll consist of 3 questions and we expect you to solve >=2 problems. If for any reason you are unable to solve, then we shall also be having re-attempts as well.(We'll provide more info on re-attempts moving forward)
- Contests are critical to retaining what you have learnt and measuring where you need improvement. Please take contests seriously.

Be consistent in solving problems. If stuck, please post the issue in your WA/Slack group and let's make it a habit of helping each other as it will eventually help you to be better.

FAQs :

- Notes will be uploaded after the class.
- Assignments will be unlocked after the class ends.
- There is no deadline for assignments.
- If asking a question, ask in public chat.
- If answering a question, answer in private chat.

Content

- Count the factors
- Optimisation for count the factors
- Check if a number is prime
- Sum of N natural numbers
- Definition of AP and GP
- How to find the no. of times a code runs
- How to compare two algorithms.

What is a factor ?

N is number

N is an integer

$\frac{N}{x}$

x is a factor of N.

24

is 2 a factor ?

Remainder when 24 is divided by 2 = 0

How to check for a factor programatically

N % x == 0 { x is a factor of N }

Count factors of a given number N N>0

N = 24 1 2 3 4 6 8 12 24 = 8

N = 10 1 2 5 10 = 4

N = 4 1 2 4 = 3

What is the minimum value of a factor ?

1

What is the maximum value of a factor ?

N

Pseudocode

```
count = 0
for i ⟶ 1 to N {
    // check if i is a factor of N
    if ( N % i == 0 ) {
        count += 1
    }
}
print (count)
```

iterations = N

what is iteration ?

No. of times a for loop runs

Any machine takes 1 sec to proces 10^8 iterations

N	iteration	time
10^8	10^8	1 sec
10^9	10^9	10 secs

10^8 iterations ⟶ 1 sec

1 iteration ⟶ $1/10^8$ sec.

10^9 iteration ⟶ ?

$$\frac{10^9 \times 1}{10^8} = 10$$

10^9 iteration = $10 * 10^8$ iterations

$10 * 1$ sec

10 sec

N	iteration	time
10^8	10^8	1 sec
10^9	10^9	10 secs
10^{18}	10^{18}	10^{10} secs. $\approx$ 317 yrs

10^8 iterations $\longrightarrow$ 1 sec

1 iteration $\longrightarrow$ $1/10^8$ sec.

10^{18} iteration $\longrightarrow$?

$$\frac{10^{18} \times 1}{10^8} = 10^{10}$$

$$a^b * a^c = a^{b+c} \qquad \frac{a^b}{a^c} = a^{b-c}$$

Optimization for counting factors.

$a * b = N$ a is a factor of N

b is also a factor of N

N = 24

a	b	a < b
1	24	True
2	12	True
3	8	True
4	6	True
6	4	False
8	3	
12	2	
24	1	

Repition of factors

$a * b = N \longrightarrow N/a$

$\longrightarrow$ if a is a factor of N

$b = \frac{N}{a}$ is also a factor of N

Pseudocode

```
count = 0

for (a = 1 ; a < b ; a++) {
    // if a is a factor
    if (N % a == 0) {
        // => b is also a factor ie N/a
        count += 2
    }
}
print (count)
```

$a < \frac{N}{a} \longrightarrow a*a < N$

N = 4

a		b	a <= b
1	*	4	True
2	*	2	True
4	*	1	False

If a == b then increment the count by 1

a <= sqrt(N)

```
count = 0
for (a = 1 ; a*a <= N; a++) {
    // if a is a factor
      if (N % a == 0) {
          // => b is also a factor ie N/a
          b = N/a
          if (a == b)  count += 1
          else count += 2
      }
}
print (count)
```

iterations = sqrt(N)

N = 12	a	b	count
	1	12	2
	2	6	4
	3	4	6
	4		

since a*a > 12

N = 12 1 2 3 4 6 12

N = 36	a	b	count
	1	36	2
	2	18	4
	3	12	6
	4	9	8

5
6 6 9
7

$7*7 > 36$

Assumption ⟶ It takes 1 sec to process 10^8 iterations

N	iterations	time
10^{18}	$\sqrt{10^{18}} = 10^9$	10 secs.

317 yrs ⟶ 10 secs.

Break ⟶ 22:38

Given N, check if N is prime or not ?

A prime no. may exactly two factors

10 11 23 2 25 27 31

```
bool isPrime(int N) {
    if (countFactors(N) == 2) {
        return true
    }
    else {
        return false
    }
}
```

Sum of all no. from 1 to 100

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

$$S = 1 + 2 + \ldots\ldots + 99 + 100$$
$$S = 100 + 99 + \ldots\ldots + 2 + 1$$
$$2*S = \underbrace{101 + 101 + \ldots\ldots + 101 + 101}_{100 \text{ times}}$$

$$2S = 101 * 100$$
$$S = \frac{101 * 100}{2} = 5050$$

$$S = 1 + 2 + 3 + \ldots\ldots + n-1 + n$$
$$S = n + n-1 + \ldots\ldots + 2 + 1$$
$$2S = \underbrace{(n+1) + (n+1) \ldots\ldots (n+1) + (n+1)}_{n \text{ times}}$$

$$2S = (n+1) * (n)$$

$$S = \frac{(n+1) * (n)}{2}$$

Basic Math Concepts

$[1, 5]$ ⟶ All the values from 1 to 5 inclusive of 1 and 5

$[1, 5)$ ⟶ All the values from 1 to 4 include 1 and exclude 5

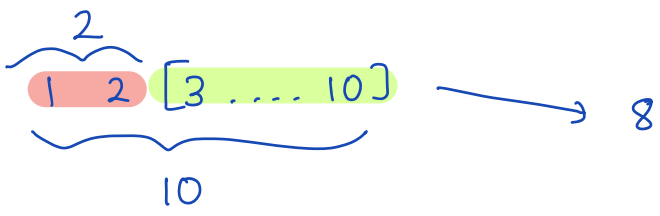
$[a, b]$ ⟶ All values from a to b, including a and b

(a, b) ⟶ All values from a to b, excluding a and b

$[3, 10]$ ⟶ 3 4 5 6 7 8 9 10

$[1, 10]$ ⟶ 1 ..... 10 {10 values}.

$[1, 2]$ ⟶ 1 2 {2 values}



$[a, b]$ ⟶ $b - (a-1) = b - a + 1$

$[1, b] - [1, a-1]$

$[2, 3]$ ⟶ ~~$b - a$~~ (1) | ~~$b - a - 1$~~ (0) | $b - a + 1$ (2)

Calculate the iterations for below code

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

iterations = N

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

[0, 100]

100 − 0 + 1 = 101

iterations 101

```
for ( i = 1 ; i <= N ; i++ ) {
    if ( i % 2 == 0 ) print(i)
}
```

N

```
for ( i = 1 ; i <= M ; i++ ) {
    if ( i % 2 == 0 ) print(i)
}
```

M

total N + M

Geometric Progression

	a				
	5	10	20	40	80
		$\frac{10}{5}$	$\frac{20}{10}$	$\frac{40}{20}$	$\frac{80}{40}$
r		2	2	2	2

$$r = \frac{A_{i+1}}{A_i}$$

a ⟶ first term of GP

r ⟶ common ratio

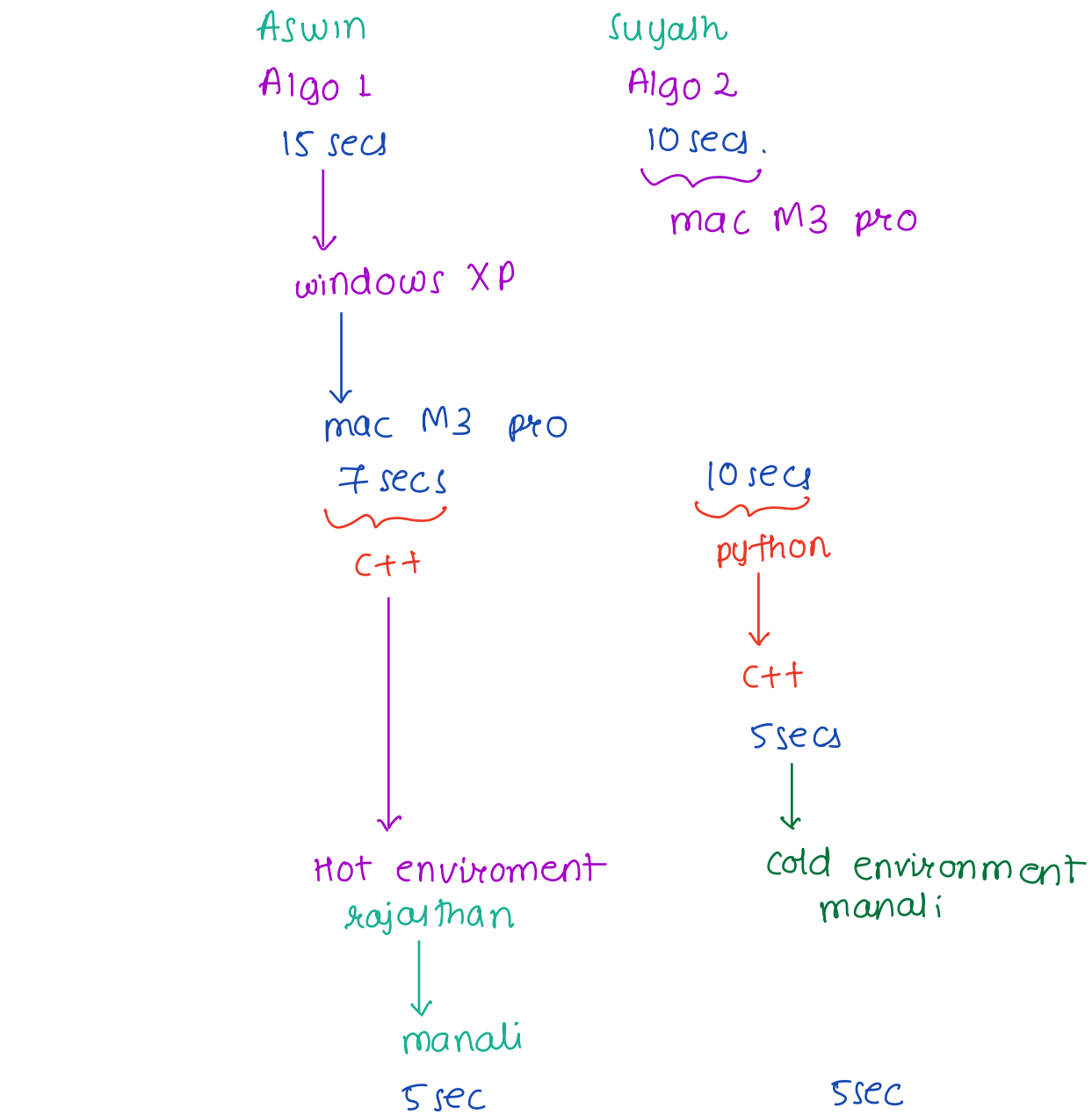
Sum of a GP

$$a, \quad a*r, \quad a*r^2, \quad \ldots\ldots \quad a*r^{n-1}$$

$$\text{sum of nterms of GP} = \frac{a*(r^n - 1)}{r - 1}$$

$r != 1$

if $r == 1$ then sum $= a*n$



Conclusion ⟶ Running time of algos depend on lot of factors

⟹ Its always better to compare the no. of iterations

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

iterations = N