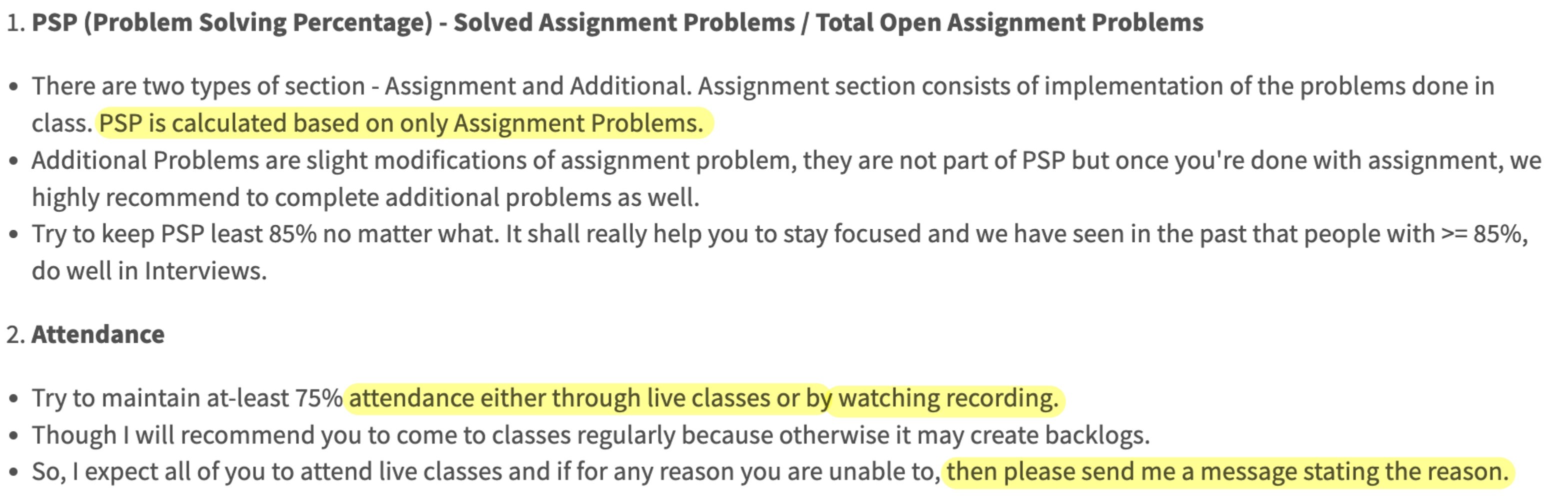
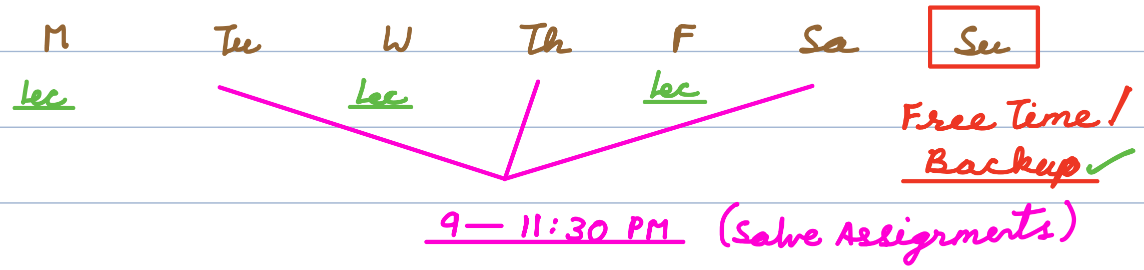


1. **PSP (Problem Solving Percentage) - Solved Assignment Problems / Total Open Assignment Problems**

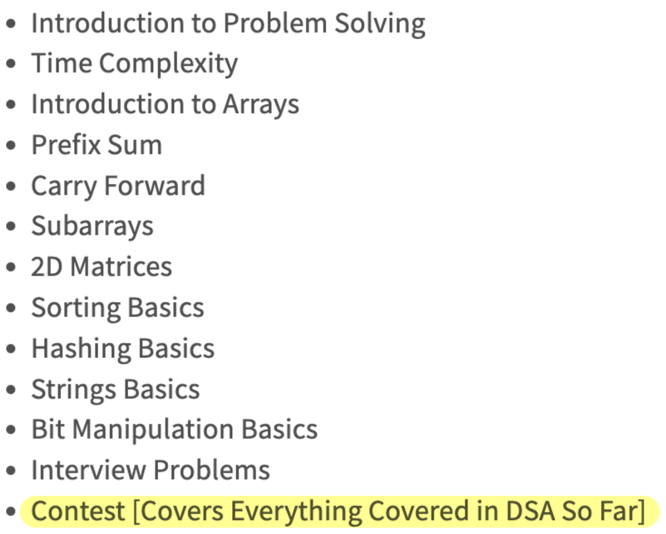
- There are two types of section - Assignment and Additional. Assignment section consists of implementation of the problems done in class. PSP is calculated based on only Assignment Problems.
- Additional Problems are slight modifications of assignment problem, they are not part of PSP but once you're 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.

2. **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



- 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 Everything Covered in DSA So Far]

1) Join by 9:05 PM
2) Questions → Question Tab
3) Answer → Privately
4) Feedback to be filled

Q → What is the count of factors of 10?

$\frac{10}{x}$ = integer $\qquad x \rightarrow 1, 2, 5, 10$

x → factor

$N = 24$ factors → {1, 2, 3, 4,
6, 8, 12, 24}

Q → What is the count of factors of N ?

$N = \underline{10^9}$

check if 'x' is a factor of 'N' → $\underline{N \% x = 0}$

smallest factor → $\underline{1}$

Largest factor → $\underline{N}$

```
cnt = 0
for x → 1 to N {        // Bruteforce
    if (N % x == 0)
        cnt ++
}
return cnt
```

Let say server have the capacity of running 10^8 iterations in 1 sec.

$N = 10^7 \longrightarrow \underline{0.1 \text{ sec}}$

$N = 10^9 \longrightarrow \underline{10 \text{ sec}}$

$$\boxed{\begin{array}{l} 10^8 \longrightarrow 1 \text{ sec} \\ 10^7 \longrightarrow \frac{1}{10^8} * 10^7 \end{array}}$$

$N = \underline{10^{18}} \longrightarrow \frac{10^{18}}{10^8} = 10^{10} \text{ sec} \rightarrow \frac{10^{10}}{60*60*24} \text{ days} \rightarrow \frac{10^{10}}{3600*24*365} \text{ years}$

≈ $\underline{317 \text{ years}}$ 😮

correct → may not be the best

[Need to optimize]

Better Solution

N = 24 → {1, 2, 3, 4, 6, 8, 12, 24}

N = 10 → {1, 2, 5, 10}

a	*	b
1	*	10
2	*	5

a	*	b
1	*	24
2	*	12
3	*	8
4	*	6

$\boxed{N = a * b \,,\quad a <= b}$

$b = \frac{N}{a}$

$a <= N/a$

$\Rightarrow a^2 <= N$

$\Rightarrow a <= \underline{\sqrt{N}}$

$a_{min} = \underline{1}$

$a_{max} = \underline{\sqrt{N}}$

```
cnt = 0
for a → 1 to $\sqrt{N}$ {        // for (a=1; a*a <=N; a++)
    if (N % a == 0)
        cnt += 2
}
return cnt
```

N = 10 o/p = 4

a = ~~1~~ ~~2~~ ~~3~~ 4 (stop)

cnt = 0 + 2 + 2 = 4

N = 36

o/p → ~~10~~ → 9

a	*	b
1	*	36
2	*	18
3	*	12
4	*	9
6	*	6

```
cnt = 0
for a → 1 to $\sqrt{N}$ {
    if (N % a == 0) {
        if (a*a == N) cnt++
        else cnt += 2
} return cnt
```

#iterations = $\sqrt{N}$

$N = 10^{18} \rightarrow \#iterations = \sqrt{10^{18}} = \underline{10^9} \rightarrow \underline{10 \text{ sec}}$

317 year ⟶ 10 sec

Prime Numbers

Positive number with exactly 2 factors 1 & itself.

10   11 ✓   23 ✓   2 ✓   25   27   31 ✓

$-1 \rightarrow factors = \{-1, 1\}$

→ not a prime

check if a number is prime →

```
if ( N > 0 && countofFactors(N) == 2)
        return true
else  return false
```

Sum of Natural Numbers

$1 + 2 + 3 + \ldots + 100 = \underline{5050}$

$$S = 1 + 2 + 3 + \ldots + (N-1) + N$$

$$+ \quad \underline{S = N + (N-1) + \ldots + 2 + 1}$$

$$2S = (N+1) + (N+1) + \ldots + (N+1) + (N+1)$$

$$= (N+1) * N$$

$$\Rightarrow \boxed{S = \frac{N*(N+1)}{2}} \qquad \frac{\overset{50}{\cancel{100}} * 101}{\cancel{2}} = \underline{5050}$$

$[l \quad r] \rightarrow \{l, l+1, l+2, \dots r-1, r\}$

close → including l, r

$(l \quad r) \rightarrow \{l+1, l+2, \dots r-2, r-1\}$

open

excluding l, r

#elements

$$\begin{array}{r} 1 \text{———} r \\ -\; 1 \text{——} (l-1) \\ \hline r-(l-1) = \underline{r-l+1} \end{array}$$

#elements

$r-l+1-2$

$= \underline{r-l-1}$

$[4 \quad 10]$

↓

1 2 3 4 5 6 7 8 9 10

$10-(4-1)$

$r-(l-1) = \boxed{r-l+1}$

<u>Iterations</u>

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

$\#\text{iterations} = \underline{N}$

2)
```
for (i = 0 ; i <= 100 ; i++) {
        s = s + i + i²
}
```

$i \rightarrow 0$ to 100 $\qquad [0 \quad 100]$

$\#\text{iterations} = 100 - 0 + 1 = \underline{101}$

3)
```
for (int i = 1 ; i <= N; i++) {
    if (i%2 == 0)
        print (i)
}
```

$i \rightarrow 1$ to N

#iterations = N

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

$j \rightarrow 1$ to M

#iterations = M

#iterations = $\underline{\underline{N + M}}$ → one after the other

---

Geometric Progression (G.P.)

5 10 20 40 80 . . .

$T(N) = T(N-1) * r$

common ratio

first term

$$- \quad S = a + \cancel{a*r} + \cancel{a*r^2} + \cancel{a*r^3} + \ldots + \cancel{a*r^{N-1}}$$

$$+ \quad \underline{S*r = \cancel{a*r} + \cancel{a*r^2} + \cancel{a*r^3} + \ldots + \cancel{a*r^{N-1}} + a*r^N}$$

$$S*r - S = a*r^N - a$$

$$\Rightarrow \quad S(r-1) = a(r^N - 1)$$

$$\Rightarrow \quad \boxed{S = \frac{a*(r^N-1)}{(r-1)}}$$

Context

	Himangi (Algo 1)	Faisal (Algo 2)
Execution Time →	15 sec	10 sec
	(Windows XP)	(Mac M2)
	↓	↓
	(Mac M2)	
	8 sec	10 sec
	(c++)	(Python)
	↓	↓
		(c++)
	8 sec	5 sec
	(very hot)	(super cold)
	↓	↓
	same loc.	
	5 sec	5 sec

Moral → Execution time depends on multiple factors & hence not best way to compare algorithms.

#iterations → do not depend on any factor.

↘ Time Complexity (Next Class)

$a = 3$

$r = 5$

3

$15 = 3 * 5$

$75 = 15 * 5$

$\vdots$

Practice