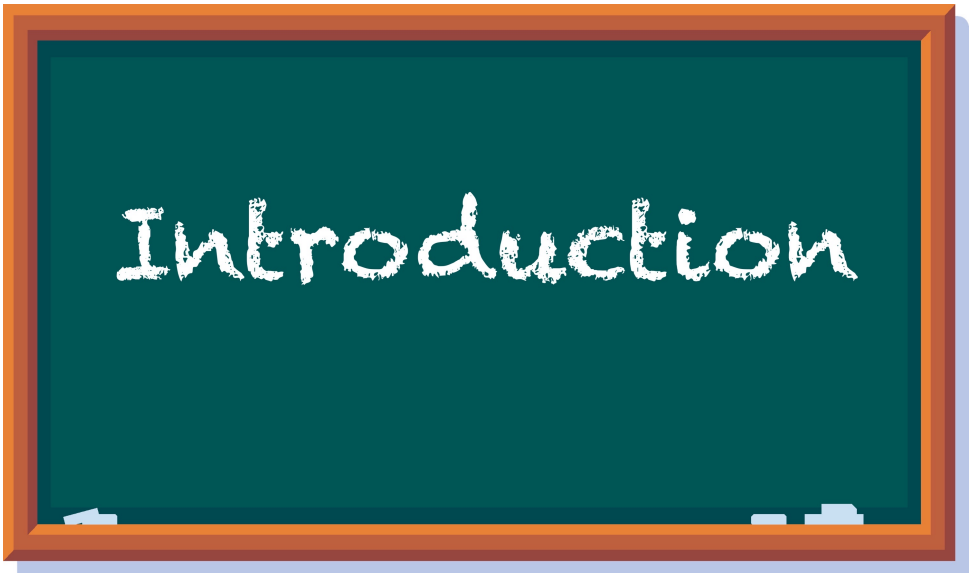
SCALER

Introduction to Problem Solving

TABLE OF CONTENTS



→ Jitender Punia (Jeetu)

→ Full time Instructor + Program Management Work

→ 4.5 years of teaching experience.

Few terms that you shall see/hear throughout the course!

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 problems, 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 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 classes or by watching recordings.

- 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 reasons you are unable to, then please send me a message stating the reason.

Next Month

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

Contest will be organised after Intermediate Module

- It will be for 1.5 hours and will be conducted within class duration followed by Contest Discussion (Instructor shall be discussing contest problems).
- It will 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 will 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.

FAQs

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

Question To : Everyone

Answering To : Jitender. (private)

Factor → i is a factor of N if i divides N completely.
i.e. the remainder is 0.

< *Question* > : Given N. Find the count of factors of N. (N > 0)

1. **N = 24**

1, 2, 3, 4, 6, 8, 12, 24

count of factors = 8.

2. **N = 10** [1, 10]

1, 2, 5, 10

count of factors = 4.

% → 10%2 → 0
10%3 → 1
10%4 → 2
10%5 → 0

idea → iterate on all the numbers from 1 to N and check if that no. divides N completely.

$N \geq 1$.

```
int countFactors(int N){
    count = 0;
    for(i = 1; i <= N; i++){   // i → 1 to N
        if(N % i == 0){
            count++;
        }
    }
    return count;
}
```

→ (Next return)

Assumption: → 10^8 iterations are executed in 1 sec.

N	no. of iterations	execution time
10^8	10^8	1 sec.
10^9	10^9	10 sec.
10^{18}	10^{18}	10^{10} sec. ≈ 317 years.

10^8 itr → 1 sec

Yle → 1st Gen - 2nd Gen - 3rd Gen → 4th / 5th.

Optimisation

$i * j = N$, Both i and j are factors of N.

$j = N/i$, Both i and N/i are factors of N.

⇓

If i is a factor, then N/i is also a factor of N.

$N = 24$

i		N/i
1	≤	24
2	≤	12
3	≤	8
4	≤	6

6	4
8	3
12	2
24	1

$N = 100$

i		N/i
1	≤	100
2	≤	50
4	≤	25
5	≤	20
10	≤	10

20	5
25	4
50	2
100	1

- After a certain value, factors are repeating.

$i \leq N/i \Rightarrow i * i \leq N$

$\Rightarrow i \leq \sqrt{N}$

</> Code

```
int countFactors2 (int N){
    count = 0;
    for( i = 1; i*i <= N; i++){
        if( N%i == 0){ // i, N/i are factors of N
            if( i == N/i ) { count++ }
            else { count += 2 }
        }
    }
    return count;
}
```

$i \le \sqrt{N}$

N → 100

i	N/i
1	100
2	50
4	25
5	20
10	10

count++

count → ~~0~~ 2 4 6 8

Assumption :- 10^8 iterations are executed in 1 sec.

N	no. of iterations	execution time
10^{18}	10^9	10 sec.

∴ Observation is the key ingridient to improve Problem Solving.

Prime Numbers ⇒ no's having only 2 factors.

[1 and no. itself]

no → 10, 11, 23, 2, 25, 27, 31 → ans = 4

< *Question* > : Given a number N. Check if it is prime or not.

< / > *Pseudo Code*

```
boolean checkPrime (int N) {
    int count = countFactors2(N);
    if (count == 2) {
        return true;
    }
    else {
        return false;
    }
}
```

1 is neither prime nor composite no.

prime → exactly 2 factors

composite no. → >2 factors.

// Gauss. (4th class)

$$S = 1 + 2 + 3 + 4 + \cdots\cdots 98 + 99 + 100$$

$$S = 100 + 99 + 98 + 97 + \cdots\cdots 3 + 2 + 1$$

$$2.S = \underbrace{101 + 101 + 101 + 101 + \cdots\cdots 101 + 101 + 101}_{100 \text{ times}}$$

$$2.S = 101 \times 100$$

$$S = \frac{101 \times \cancel{100}^{50}}{\cancel{2}} = 5050$$

Sum of first N natural no's →

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

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

$$2.S = \underbrace{(N+1) + (N+1) + \cdots\cdots (N+1)}_{N \text{ times}}$$

$$\boxed{S = \frac{N(N+1)}{2}}$$

Range

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

[1 , 11] → 11 - 1 + 1 ⇒ 11

[3 , 10] → 10 - 3 + 1 = 8

[a , b] → b - a + 1

[a , b) → b - a

(a , b] → b - a

(a, b) → b - a - 1

What is an iteration?

⇒ No. of times a loop runs.

Quiz- 1

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

i → [1, N]

⇒ N iterations

Quiz- 2

```
s=0;
for(int i=0; i≤100; i++){
    s=s+i+i²
}
```

i : [0, 100]

iterations → 100 - 0 + 1 = 101

Quiz- 3

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

$i : [1, N]$ → N iterations

+

$j : [1, m]$ → m iterations

⇒ $N+m$ iterations.

Geometric Progression

- Series where the ratio of two consecutive terms remains same.

5 , 10, 20 , 40 , 80 , 160 , 320 , 640

2 2 2 2 2 2 2

first term → a , common ratio → r

$a, ar, ar^2, ar^3, ar^4, ar^5, ar^6 \ldots \underbrace{ar^{N-1}}_{N^{th}\ term}$

Sum of first N terms of G.P

$$\frac{a \cdot (r^N - 1)}{(r-1)}$$

$$rS = ar + ar^2 + ar^3 + \cdots + ar^{N-1} + ar^N \quad \text{—①}$$

$$S = a + ar + ar^2 + \cdots + ar^{N-2} + ar^{N-1} \quad \text{—②}$$

$$rS - S = a \cdot r^N - a$$

$$S(r-1) = a(r^N - 1)$$

$$\left\{ S = \frac{a(r^N - 1)}{(r-1)} \right\}$$

$\left\{ \begin{array}{l} a \rightarrow 1^{st} \text{ term} \\ r \rightarrow \text{common ratio} \\ N \rightarrow \text{no. of terms} \end{array} \right\}$

How to compare two algorithms?

N = 10^8. Given N elements, sort the elements in increasing order.

Amit. {Algo-1}

↓

15 sec

(Windows XP)

↓ (Macbook)

7 sec.

(C++)

↓

7 sec. (near volcono)

↓ (Mt. Everest)

5 sec.

Tushar {Algo-2}

↓

10 sec.

(Macbook)

↓

10 sec

(Python)

↓ C++

5 sec. (Mt. Everest)

↓

5 sec.

Execution time

{Conclusion - We can't use execution time to compare 2 algorithms.}

How should we compare them?

No. of iterations.

---

3 sessions. → Advanced DSA-1

→ Permutations & combination

→ Prime

→ GCD.