Hello Everyone ☺

→ Welcome to intermediate module of D.S.A

→ Jitender Punia (Jeetu)

→ B.Tech from VSIT, co-founder of pepcoding

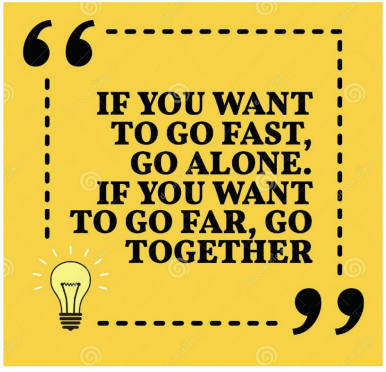
→ ~3 years of teaching experience.

FAQ's.

- → Notes will be uploaded after the class.
- → Assigments will be unlocked after the class ends.
- → There is no deadline for assigments.
- → Separate classes for HW / assigment problems.

[question → public. [to everyone]
answer → private [to me].]

Today's Quote →



→ {peer to peer} learning

① Count of factors

⇓

any no. which divides N completely.

$N \% i == 0$

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

N=10 { 1, 2, 5, 10 }

$N \% i == 0 \Rightarrow i$ is a factor of N.

pseudo-code.

```
int countFactors ( int N ){
    int factors = 0
    for( i = 1 ; i <= N ; i++){
        //is i a factor of N
        if ( N%. i == 0) { factors = factors +1 }
    }
    return factors ;
}
```

→ N ?
→ system?

i : [1, N]
⇓
N iterations.

{ Assumption → 10^8 iterations per sec }

N	iterations	Execution Time?
10^8	10^8 iterations	1 sec.
10^9	10^9 "	10 sec.
10^{18}	10^{18} "	10^{10} sec. ≈ 317 years.

10^{18} iterations → $\frac{1}{10^8} \times 10^{18}$ sec.

$= 10^{10}$ sec.

10^8 iterations → 1 sec.

1 iterations → $\frac{1}{10^8}$ sec.

10^9 iterations → $\frac{1}{10^8} \times 10^9$

→ 10 sec.

You → children → Grand-children → 4^{th} → 5^{th} / 6^{th}.

Optimise.

$i * j = N \Rightarrow \{ i$ and j are factors of $N \}$

$\Rightarrow \boxed{j = N/i} \Rightarrow \{ i$ and N/i are factors 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

observation → After a certain no, the factors are repeating.

→ All the factors are present in part-1.

In part 1:

$$i <= N/i$$

$$i * i <= N$$

$$i^2 <= N \implies \boxed{i <= \sqrt{N}}$$

optimised code.

```
int countFactors(int N){
    int factors = 0
    for(int i = 1; i*i <= N; i++){
        if(N%i == 0){
            if(i == N/i){ factors += 1 }
            else { factors += 2; }
        }
    }
    return factors;
}
```

[i : [1, √n]
∴ √n iterations]

N = 100 → factors = 9

i = 1 2 3 4 5 6 7 8 9 10.

↳ i and N/i are factors of N.

{Assumption → 10^8 iterations per sec}

N	Iterations	Executions time
10^{18}	10^9	10 sec.

⇒ {Most important skill for problem solving → observation}

$$i <= \sqrt{N}$$
$$i^2 <= N$$
$$i * i <= N$$

Q: Given N. You need to check if it is prime or not.

Nos ⇒ { 10✗, 11✓, 23✓, 2✓, 25✗, 27✗, 31✓ } ans → 4.

prime → number which is having only 2 factors.
⇕
[1 and N itself]

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

→ 1 is neither prime nor composite.
↳ nor having more than 2 factors.

Gauss's

4th class. // Gauss

$$S = 1 + 2 + 3 + \_\_\_\_\_ 99 + 100$$

$$S = 100 + 99 + 98 + \_\_\_\_\_ 2 + 1$$

$$2S = 101 + 101 + 101 + \_\_\_\_\_ 101 + 101$$

$$2S = (101) * (100)$$

$$S = \frac{(101) * (100)}{2}$$

Sum of 1st - N natural no's.

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

$$S = N + (N-1) + (N-2) + \_\_\_ 2 + 1$$

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

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

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

Q) Given a no. N → perfect square. Find sqrt(N).

N = 25 → 5

N = 36 → 6

N = 100 → 10

N = 30 { We will never get invalid inputs }

```
int sqrt (N) {
    for ( i = 1 ; i <= N ; i++){
        if( i*i == N) return i;
    }
}
```

Amazon MCQ.

(a) $\log_2 N$

(b) N

(c) $\sqrt{N}$

(d) None of these.

N = 25 i = 1, 2, 3, 4, 5

Q) Find sqrt(N).

If N is not a perfect square. return floor (sqrt(N))

N = 49 ⇒ 7

N = 60 ⇒ 7

N = 31 ⇒ 5

N = 29 ⇒ 5

floor (x) → Greater integer <= x value

floor (7.238) → 7.

N = 50.

i	i * i <= N
1	ans = 1
2	ans = 2
3	ans = 3
4	ans = 4
5	ans = 5
6	ans = 6
7	ans = 7
8	8 * 8 <= 50 ✗

```
int sqrt(N) {
    int i = 1 , ans = 0
    while ( i * i <= N ) {
        ans = i ;
        i++ ;
    }
    return ans;
}
```

i : [1, √n]

{ # No. of iterations : √n. }

N = 29.

i = 1 2 3 4 5 6

ans = 0 1 2 3 4 5

sqrt(N) → √n iterations.

sqrt(N) → ideal → $\log_2 N$ iterations [Advance module]

Log- Basics

$\log_b a = c$ [to what value we need to raise b so that value = a]

$b^c = a$

$\log_2 64 = 6$ $\quad 2^? = 64$ $\quad 2^? = 2^6$

$\log_3 27 = 3$ $\quad 3^? = 27$

$\log_5 25 = 2$ $\quad 5^? = 25$

$\log_2 32 = 5$ $\quad 2^? = 32$

$\log_2 10 = 3$ $\quad 2^? = 10$

$\log_2 40 = 5$ $\quad 2^? = 40$

***

$2^K = N$

$\log_2 N = K$

$\log_2 2^{10} = 10$ $\quad 2^? = 2^{10}$

$\log_3 3^5 = 5$ $\quad 3^? = 3^5$

***

$\log_a a^N = N$

Q. Given a +ve integer N. How many times do we need to divide it by 2 until it reaches 1.

N = 100

↓ /2

50

↓ /2

25

↓ /2

12

↓ /2

6

↓ /2

3

↓ /2

1

ans = 6

N = 324

Homework

↓ /2

162

↓ /2

81

↓ /2

40

↓ /2

20

↓ /2

10

↓ /2

5

↓ /2

2

↓ /2

1

ans = 8.

Intermediate content

- Introduction to Problem Solving
- Time Complexity - 1, 2
- Arrays - 6 {prefix Sum, subarrays, carry-forward, sliding window, 2-D matrix}
- Interview Problems 2
- [Bit Manipulations - 2
  → Modular Arithmetic - 1]
- Sorting - 1 ✓
- Strings - 1 ✓
- Hashing - 2 ✓
- Recursion - 2 ✓
- classes & objects - 1 ✓
- LinkedList Basics - 1 ✓
- Trees Basics - 1 ✓

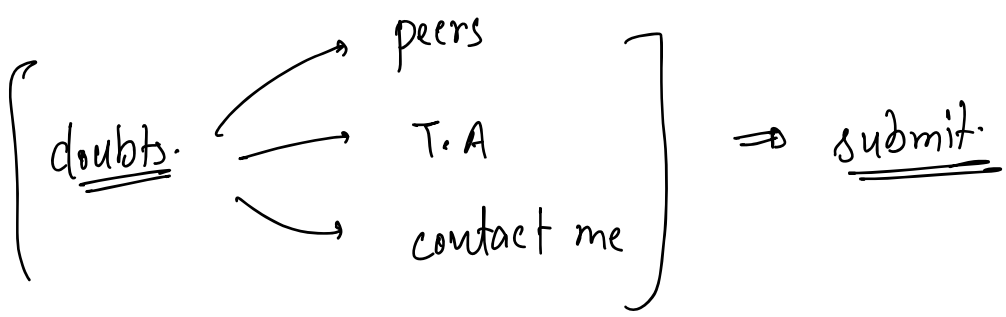
[2 months.]

Advance module.
↓
≈ 4 months..

→ Ping me on slack [Jitender Punia]

→

→ {fun while problem solving}



floor (5.--)

⇓

5.

{Time Complexity & Space}