Hello Everyone

- Welcome to Scaler
- Satya Sai Siva Rama Krishna 2019 CSE graduate
  - VNR VJIET
- 2½ years teaching experience

Today's Content:

- Count no: of factors
- Gauss Sum
- Divide num/2 till it reaches 1
- Sqrt()
- Entire Content of Intermediate Content
- How to make best of Course

FAQ's : a) Notes will be uploaded right after session

b) Assignments will be unlocked once session ends

c) There is no deadline for assignments

d) During doubt session, attendance not counted
   - recorded: Yes
   - Notes: Yes

e) Language independent { Pseudo Code }

Count No: of factors:

Q1: Is 4 a factor of 24 → Yes, 24 % 4 == 0

N=10 : {1, 2, 5, 10} → 4

N=12 : {1, 2, 3, 4, 6, 12} → 6

1 iteration → $1/10^8$ sec

Ass: 10^8 iterations per sec

countFactors(N) {

i : [1 ... N] : # N iterations

```
int c = 0 → {no: of factors} → Input N ⟶ # N iterations
for ( i = 1 ; i <= N ; i = i + 1 ) {
    // i is factor of N
    if ( N % i == 0 ) { c = c + 1 }
}
return c;
}
```

N	→ #iterations	→ Execution time
10^9	10^9	10 sec

1 iteration → $1/10^8$ sec

10^9 iterations - $10^9/10^8$ sec

- 10 sec

10^{18} 10^{18} ⟶ 10^{10} sec

1 iteration → $1/10^8$ sec

10^{18} iteration → $10^{18}/10^8$ sec

→ 10^{10} sec

Converting

Sec → days → Years

1 day - 86400 sec

1 Year - 365 days = [365 * 86400] sec

$10^{10} / [365 * 86400]$ → 317 years

You → child → grandchild → 3rd → 4th → 5th → 6th/7th

optimize:

a) $i * j = N$

(i & j are factors of N)

$j = N/i$

(i & N/i are factors of N)

Claim: { If i is a factor of N

N/i is also factor of N }

b) Relation operator:

$<, >, <=, >=, ==, !=$

c) a = 50,

max a value = 50

N = 24

c = 0

i	Part I	N/i	c
1	a =	24	+2
2	a =	12	+2
3	a =	8	+2
4	a =	6	+2
			8
6	Part-2	4	
8		3	
12		2	
24		1	

Claim: If i iterates only in part-I we can get all factors

i a = N/i

$i * i$ a = N

i^2 a = N

$(i^2)^{1/2}$ a = $(N)^{1/2}$

i a = $\sqrt{N}$

In part-1

i : [1, $\sqrt{N}$]

N = 100

c = 0

i	Part I	N/i	c
1	a =	100	+2
2	a =	50	+2
4	a =	25	+2
5	a =	20	+2
10	a =	10	+1
			9
20	Part-2	5	
25		4	
50		2	
100		1	

If i only iterates from [1, $\sqrt{N}$] we can get all factors of N?

$\sqrt{N}$ → may not be an integer always?

countFactors(N) {

int c = 0

for(i = 1 ; i * i <= N ; i = i+1) {

  if (N % i == 0) {

    i is a factor of N, N/i is also factor

    if(i == N/i) { c = c+1 }

    else { c = c+2 }

  }

}

return c;

}

i <= N/i or i * i <= N or i <= √N ← use this condition

for a given N : i = [1, √N]

↳ given N → √N iterations

Ass: 10^8 iterations per sec

N	→ #iterations (√N)	→ time
10^{18}	$\sqrt{10^{18}} = 10^9$	10 sec

countFactors ($N = 10^{18}$)

↳ Idea 1 → 317 years

↳ Idea 2 → 10 sec

huge option

N = 40, c = 0

i, i * i <= 40, N % i == 0 → i, N/i

1	1 * 1 <= 40	1	40, c = c+2
2	2 * 2 <= 40	2	20, c = c+2
3	3 * 3 <= 40		
4	4 * 4 <= 40	4	10, c = c+2
5	5 * 5 <= 40	5	8, c = c+2
6	6 * 6 <= 40		
7	7 * 7 <= 40	{ come out of loop, return c } → 8	

Intermediate:

↳ Increasing observation & skills

↳ learn interesting techniques

↳ Start {observations} / {skills} → { when to use a data structure }

Tricks:

Shilpa: → Gauss
4 4th class

$$S = 1 + 2 + 3 + 4 + \dots + 98 + 99 + 100$$
$$S = 100 + 99 + 98 + 97 + \dots + 3 + 2 + 1$$

$$2S = 101 + 101 + 101 + 101 + \dots + 101 + 101 + 101$$

$$2S = 100 * 101$$

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

// Sum of 1st N Natural Numbers? $= \frac{N(N+1)}{2}$

$$S = 1 + 2 + 3 + 4 + \dots + N-2 + N-1 + N$$
$$S = N + N-1 + N-2 + N-3 + \dots + 3 + 2 + 1$$

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

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

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

log Basics:

$\log_b^a = c$, $b^c = a$, { To what power we need to raise b to get val a }

$\log_b^a$ ($a \to 64$, $b \to 2$) $= \log_2^{64} = 6$, $2^6 = 64$

$\log_b^a$ ($a \to 27$, $b \to 3$) $= 3 \Rightarrow 3^3 = 27$

$\log_5^{25} = 2$, $5^2 = 25$

$\log_2^{32} = 5$, $2^5 = 32$

$\log_2^8 = 3$, $2^3 = 8$

$\log_b^a$ ($a \to 10$, $b \to 2$) $= c = 3$ { $2^3 = 8$, $2^4 = 16$, $2^c = [10]$, $c = 3.24..$ }

$\log_b^a$ ($a \to 33$, $b \to 2$) $= c = 5$ { $2^5 = 32$, $2^6 = 64$, $2^c = 32$, $c = 5.3..$ }

$\log_b^a$ ($a \to 40$, $b \to 2$) $= c = 5$ { $2^5 = 32$, $2^6 = 64$, $2^c = 40$, $c = 5.4..$ }

→ log properties

$\log_b^a$ ($a \to [2^{10}]$, $b \to 2$) $\Rightarrow 10$

$\log_b^a$ ($a \to [3^5]$, $b \to 3$) $\Rightarrow 5$

$\log_3^{3^7} \to 7$

$\log_a^{a^n} \to n$ // Very Important

$N = 2^k$, $k \Rightarrow \log_2^N$ // Very Important

$\log_c^{a \times b} \Rightarrow \log_c^a + \log_c^b$

// Given +ve N how many times we need divide it by 2 until it reaches = 1

↳ {Homework}

↳ {Next Session}

N:

1

2

4

8

9

12

24

16

32

Perfect Square

Given N a perfect Square find Sqrt(N)

N = 25 → 5

N = 36 → 6

N = 49 → 7

N = 30 → { We will never get invalid inputs }

int Sqrt(N) ↦ perfect square → amazon: Calculate iterations ?

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

a) N b) N/2

c) $\log_2^N$ d) $\sqrt{N}$ } ↳ correct answer

// Note: If we execute return we will come out of the function

i

N=36 : 1 2 3 4 5 **6** **6*6=36** {return 6} = # 6 iterations

N=64 : 1 2 3 4 5 6 7 **8** **8*8=64** {return 8} # 8 iterations

N=25 : 1 2 3 4 **5** **5*5=25** {return 5} # 5 iterations

// Sqrt(N)

Note: If N is not perfect square, return floor(sqrt(N))

	sqrt(49)
N = 49 →	7
N = 60 →	7
N = 31 →	5
N = 29 →	5
N = 16 →	4

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

→ TC

Iterations

→ i : [1, $\sqrt{N}$ + 1]

→ Code : $\sqrt{N}$ Iterations

// N = 50 :

i :	i*i <= 50
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 * {return ans} → {7}
9	
10	

N = 16

i :	i*i <= 16
1	ans = 1
2	ans = 2
3	ans = 3
4	ans = 4
5	5*5 > 16 *

return ans = 4

floor(5) = 5

floor(6.3) = 6

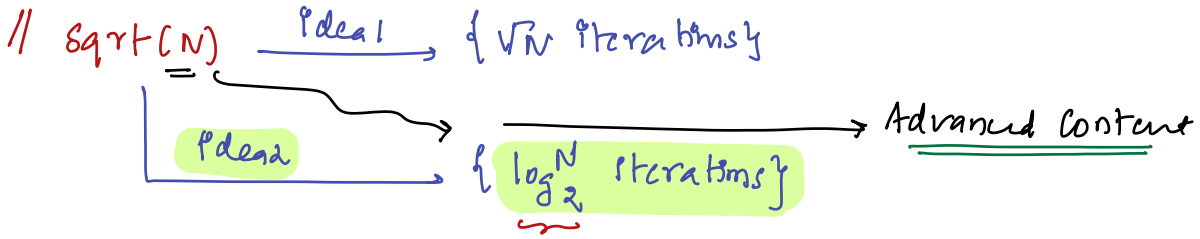
floor(6.7) = 6

} Concept

floor(2.4) = 2

floor(x) = {greatest Integer i <= x}

ceil(x) = {smallest Integer z >= x}


// sqrt(N)
Ideal
{√N iterations}
Ideal
{log2N iterations}
Advanced content

Content: → 2 months { Intermediate }

1) Introduction to Problem Solving ✓

2) TC → big O / TLE / SC / TC
   → 2 sessions

3) Arrays → 6 sessions
   ↳ { Introduction }
   - → prefix sum
   - → carry forward
   - → Subarrays
   - → Sliding window
   - → Interview Problem
   - → 2D matrix

4) Bit Manipulation → 3

4) Arrays & Maths → 2

5) Sorting & String → 2

6) Hashmap → 2 (depth ?)

7) Recursion → 2/3

Basic idea about OS

8) Class & Object → 1

9) Linked list → 1

10) Stacks → 1

11) Trees → 2

12) Subsets / Subseq → 1

{ Basic idea about data structure }

In Advanced Content, more detailed discussion

Advanced - 4 1/2 Months

Arrays - 3

Bit Manipulation - 2

Maths - 4

Recursion + Sorting - 5

Searching + 2 Pointer - 5 → {not c/c++ Pointers, plean no worries}

Hashing + Strings - 5

Linked List - 3

Stacks/Queues/De-Que - 5

Trees - 6/7

Tries / Heaps / Greedy - 5

Back Tracking - 2/3

Dynamic Programming - 7

Graphs - 5

Content Structure

Intermediate : 2
↓
Advanced : 4 1/2

DSA : 6 1/2 Months

Expectations:

1) Attend Session

2) Revise Notes

3) Solve Assignments
   - Classwork → Mostly discussed in class
   - Homework Assignn → Indirect questions / topics discussed in class

4) Doubts
   - Content/Copy
     a) Please ask, in live class
     b) Stay back & get it clarified in doubts session
   - Assignments
     a) Debug on your own
     b) Rain TA request {Teaching assistant}
     c) Doubt Session: {End of each class}
     d) Once in 2/3 weeks

Problem Solving Sessions → recorded

{Non Session days} ⇔ Problems with most doubts → optional
↳ Attendance not counted

5) Join on Time

6) If you miss
   a) Watch recording
   b) Atleast revise notes