monday

Notes

Topic Name : Introduction to Problem Solving.

TODAY:

AGENDA:

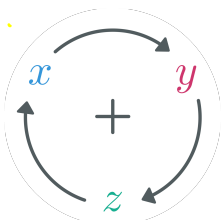
$$\vec{a} \times \vec{b} = \begin{vmatrix} x & y & z & x & y \\ a_1 & a_2 & a_3 & a_1 & a_2 \\ b_1 & b_2 & b_3 & b_1 & b_2 \end{vmatrix}$$

$$\vec{a} \times \vec{b} = (a_2b_3 - a_3b_2)x + (a_3b_1 - a_1b_3)y + (a_1b_2 - a_2b_1)z$$

A. => Need For Problem Solving

1. Some Simple & Fun Problems.
   Quizzes
   - Prime No. ✓
   - Square Root (sqrt) ✓
2. Real World Examples

9.15 - 11 pm



B. => Next 2 Months Topics & Schedule

C. => Doubts

11 - 11.30 pm

About Me : Sahil Bansal

- Graduated from B.Tech CSE, IIT Jammu
- 2.5 Years of Teaching Experience
  - ✓ → DSA (Recursion, Hashing, Sorting, Maths)
  - ✓ → OOP:
  - ✓ → DBMS
  - ✓ → Dev : Django
  - · Workshops at College
  - · C++
  - · Python
  - · Competitive Programming (ICPC, CodeChef)
- TA @ InterviewBit Academy (Now Scaler)
  Apr 2019 - Oct 2019
- Software Engineer & Instructor @ Scaler
  June 2020 - Present

Prime Number

15

21

23

35

23 ✓

1- Private Chat
Answeres

2- Public Chat
Questions
(Questions tab)

3- Acknowledge
with yes/No
reactions.

1

Prime Number

1 ⇒ Not a prime

N : a) 1 and itself as factors ✗

b) cnt factors >= 2

c) cnt factors = 2

⇒ (1) : 1 ✗ exception. NOT prime

N >= 1

Smallest prime number = 2.

Count of factors exactly = 2

(6) : 1, 2, 3, 6 ✗

(2) : 1, 2 ✓

(3) : 1, 3 ✓

(4) : 1, 2, 4 ✗

(5) : 1, 5 ✓

(1) Check if input N is prime.

Count of factors = 2

[1 - N]

Algorithm

- Brute Force
- Optimize

Pseudo-code

```
def isPrime(N):
    cnt = 0
    # i : 1 to N | for(i=1; i<=N; i++)
    for i in range(1, N+1):
        if N % i == 0:
            cnt += 1
    if cnt == 2:
        return True
    return False
```

range(1,3) = [1,2)

(1,4) = [1,2,3)

↑ excluded

n times

15

1, 2, 3, 4, ----, 15

15 % 1 = 0 ⇒ cnt = 1

15 % 2 = 1

15 % 3 = 0 ⇒ cnt = 2

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

1 2 3 4 5 6 7 8 9

[2,10] : 10 - 2 + 1 = 8 + 1 = 9

(10 - 2) + 1

[a, b] : b - a + 1

[a, b] ⇒ (b - a) + 1

excluded [a, b) : b - a

[1, N+1) : N+1 - 1 = N

⇒ Assumption : 10^8 iterations in 1 sec

Input	Iterations	Execution Time
$N = 10^9$	10^9 → 10s	$10^8 \to 1$, $1 \to \frac{1}{10^8}$, $10^9 \to \frac{1}{10^8} * 10^9 = 10$

* $\frac{a^m}{a^n} = a^{m-n}$

TLE $N = 10^{18}$ $10^{18} \to \frac{10^{18}}{10^8} = 10^{18-8} = 10^{10}$ sec.

10^{10} sec → hours → days → years ≃ 317 years

Optimize Solution

Observations

Factors N

Given

$a * b = N$

Both a, b are factors ✓

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

N = 16

1
2
4
8
16

$1 * 16 = 16$

$2 * 8 = 16$

$4 * 4 = 16$

N = 24

i			
1	1	* 24 (24/1)	= 24
2	2	* 12 (24/2)	= 24
3	3	* 8 (24/3)	= 24
4	4	* 6 (24/4)	= 24
5	6	* 4	= 24
6	8	* 3	= 24
7	12	* 2	= 24
8	24	* 1	= 24

N = 100

a		b
1	→	100/1 = 100
2	→	100/2 = 50
4	→	100/4 = 25
5	→	100/5 = 20
10	→	100/10 = 10
20	→	100/20 = 5
25	→	100/25 = 5
50	→	100/50 = 2
100	→	100/100 = 1

Part I: a <= b, a <= √100

Factors come in pairs. (a, b)

$a <= b$

$a * b = N$

$a <= b$.

Max such value

$a <= 57$

max a = 57

$a = b$

$a * b = a^2 = N$

$\Rightarrow a = \sqrt{N}$

a, b

a * b = N

If we divide into 2 parts → a <= b P1

→ a > b P2

Do not need to iterate over P2

Max value for Part-1 a

a <= b

What can be max a?

a = b

a * b = N

=> a * a = N

Take sqrt both sides

=> $a = \sqrt{N}$

a <= 57

max a = 57

N = 24

$\sqrt{24} = 4.9\ldots$

$= 4$

Integer value

N = 100

$\sqrt{100} = 10$

Observation 1 : If we iterate on Part-1, we can get all the factors of N.

Observation 2 : In Part-I, min $i = 1$

Observation 3 : In Part-I, max $i = \sqrt{N}$.

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

If N is prime

Needed $[1, N]$ => How many factors = 2

$[1, \sqrt{N}]$ => How many factors = 1

$[2, \sqrt{N}]$ => How many factors = 0

N is always a factor.

Reduced the problem.

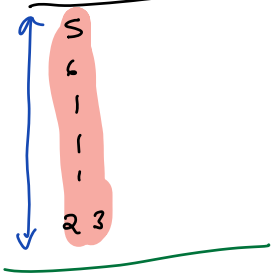
$N = 23$ $\sqrt{N} = 4.$

Integer part = 4

i	N/i
1	23
2	x x
3	x x
4	x x

=> 2 factors

=> 0 factors from $[2, \sqrt{N}]$ $[2, 4]$



$2 \to \sqrt{N}$

$2 \to 4$

$[2, 4] = 4 - 2 + 1$

$= 3$

range(1, 5)

[1, 2, 3, 4]

ANSWER

```
def isPrime(N):
    cnt = 0
    // i from 2 to sqrt(N)
    for i in range(2, sqrt(N) + 1):   # No factor in this range.
        if N % i == 0:
            cnt += 1                  # return False
    if cnt == 0:
        return True
    return False
```

convert to an int.

N = 1 → fail ✗

⇓

Edge case

Boundary

range(2, 2) = [ ]

N = 1 // 1

N = 23

$\sqrt{N} = 4.6\ldots$

4

range(2, 5) = [2, 3, 4]

int.

```
def isPrime(N):
    if N == 1:
        return False
    prime = True
    // i : 2 → √N
    for i in range(2, sqrt(N) + 1):
        if N % i == 0:
            prime = False ✓
            break ✓
    return prime
```

[2, √N + 1)

√N + 1 − 2

= √N − 1

≃ √N.

range(2, 2) = [ ]

range(2, 3) = [2]

N = 1

Won't go into the loop.

N = 2 $\sqrt{2} = 1.\ldots$

2.−

True

range(2, 2) = [ ]

N = 3 $\sqrt{3} = 1.\ldots$

N = 4 $\sqrt{4} = 2$

range(2, 3) = [2]

$\sqrt{a} = a^{1/2}$

Input	Iterations	Execution Time
10^{10}	$\sqrt{10^{10}} = 10^{10/2} = 10^5$	$\frac{10^5}{10^8} = \frac{1}{10^3} = 1$ ms.
10^{18}	$\sqrt{10^{18}} = 10^9$	$\frac{10^9}{10^8} = 10$ sec.

$10^8 \rightarrow 1$ sec.

Excellent

KEY to solving problems is

OBSERVATIONS.

Observations ✓✓

Key

Quiz / Q2

2 Sum of first 100 natural numbers.

Gauss

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

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

$$+$$

$$2S = \underbrace{101 + 101 + 101 + 101 + \dots + 101 + 101}_{100}$$

$$2S = 101 * 100$$

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

Sum of 1st N natural numbers

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

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

$$+$$

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

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

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

QUIZ 3

Given N, how many times we need to divide by 2 to make it 1. $\simeq \boxed{\log_2(n)}$

integer part

$N // 2$

↑ integer division

$15 // 2 = 7$

$\log_2 2^n = n$

$\log_a a^n = n$

$x = a^n \Rightarrow \log_a x = n$

2^1 ← 2 → $1 = 2^0$

3 → 1

2^2 ← 4 → 2 → 1

2^3 ← 8 → 4 → 2 → 1

15 → 7 → 3 → 1

2^4 ← 16 → 8 → 4 → 2 → 1

23 → 11 → 5 → 3 → 1

2^6 ← 64 → 32 → 16 → 8 → 4 → 2 → 1

(1 2 3 4 5 6)

steps	
1	$= \log_2(2^1) = 1$
1	
2	$= \log_2(2^2) = 2$
3	$= \log_2(2^3) = 3$
3	$\log_2(15) = 3.96$ … int → 3
4	$= \log_2(2^4) = 4$
4	
6	$= \log_2(2^6) = 6$

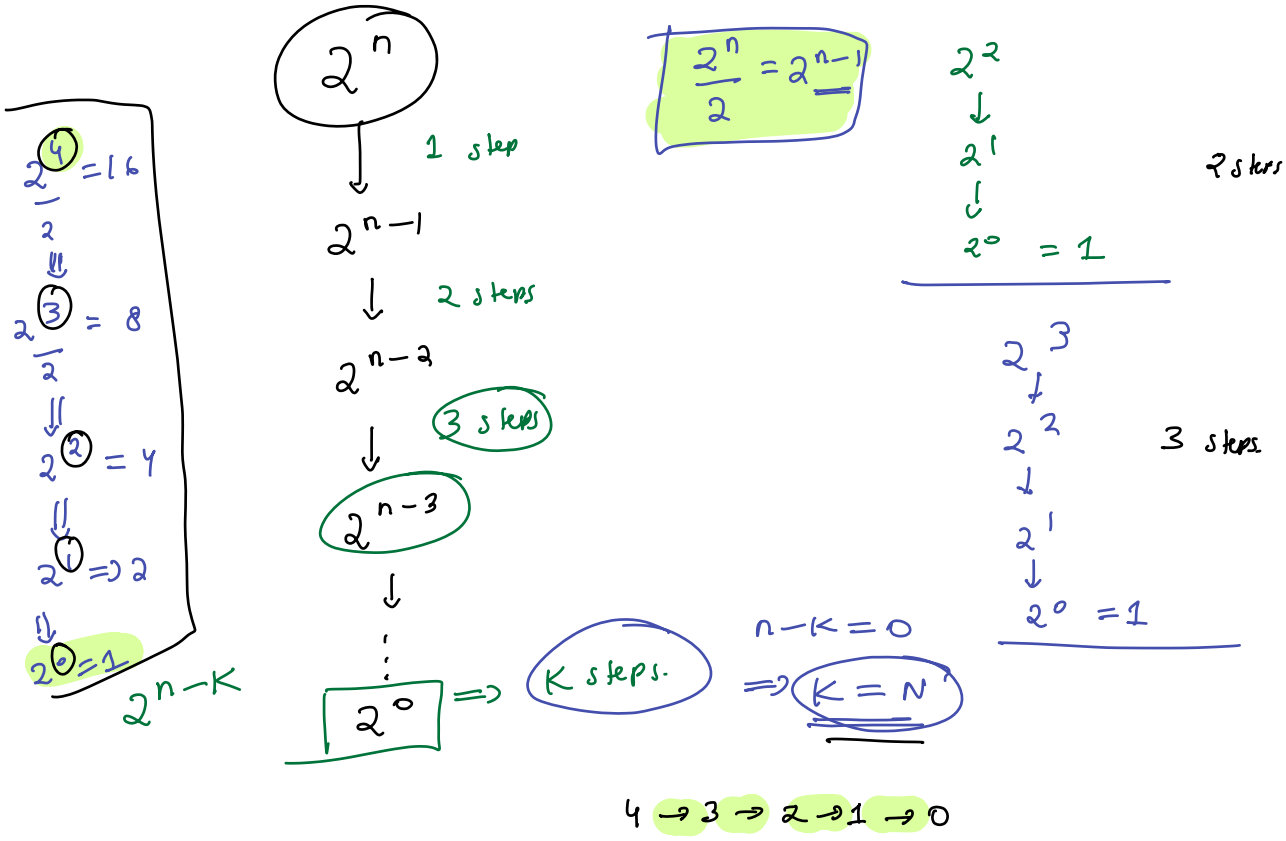
HOMEWORK: Read about log

Perfect Square N

NOTE:

- Attendance (Passive) watching recording
- get updated.

(Doubt 20-30)


2^n
1 step
2^{n-1}
2 steps
2^{n-2}
3 steps
2^{n-3}
2^0
K steps.
$n-K=0$
$K=N$
$2^4=16$
$2^3=8$
$2^2=4$
$2^1=2$
$2^0=1$
2^{n-K}
$\frac{2^n}{2}=2^{n-1}$
2^2
2^1
$2^0=1$
2 steps
2^3
2^2
2^1
$2^0=1$
3 steps
4 → 3 → 2 → 1 → 0

Perfect Square (Amazon)

Given N (Perfect Square) → $\sqrt{N}$ will be an integer

$N > 0$

Find its square root.

$\sqrt{49} = 7$

$7 * 7 = 49$

$N = 100$ → $\sqrt{100} = 10$ → $10 * 10 = 100$

$N = 25$ → $\sqrt{25} = 5$ → $5 * 5 = 25$

$N = 30$ ✗ → $\sqrt{30}$ not an integer

$N = 24$ ✗

perfect square

$\sqrt{N} \times N$

Brute Force

```
def my_sqrt(N):
    for i in range(1, N+1):
        if (i*i == N):
            return i
```

→ Given N is a perfect square

★ No of iterations

N

N/2

log N

sqrt N iterations

N = 100

i: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ... 100

i^2: 1, 4, 9, 16, 25, 36, 49, 64, 81 ... 10000

✗ ✗ ✗ ✗ ✗ ✗ ✗ ✗ ✗

$\sqrt{N}$

1 → N

10 → 100

$100 ** (1/2) = 10$ Can't use

$\sqrt{N}$ → Can't use inbuilt method.

Optimize

$x \cdot x = N$

$N = 100$

Hint

[1 ,] [ ] 100]

$N = 25$

$N = 4$

Is is a sqrt of $N = 100$?

$25 \cdot 25 = 625$

$N = 100 < 625$

1	1
2	4
3	9
4	16
10	100
25	625
100	10000

increasing

26, 27, 28, 29, - - - 100 } neglect [25, 100]

$26 * 26 > 100$

$27 * 27 > 100$

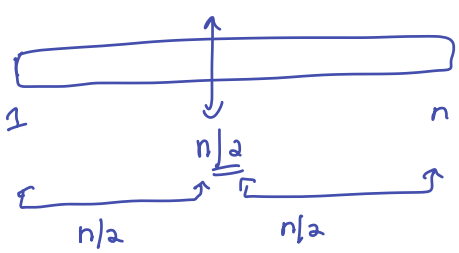
Range of sqrt will now be [1, 24]

$4 * 4 = 16 < 100$ } neglect [1, 3]

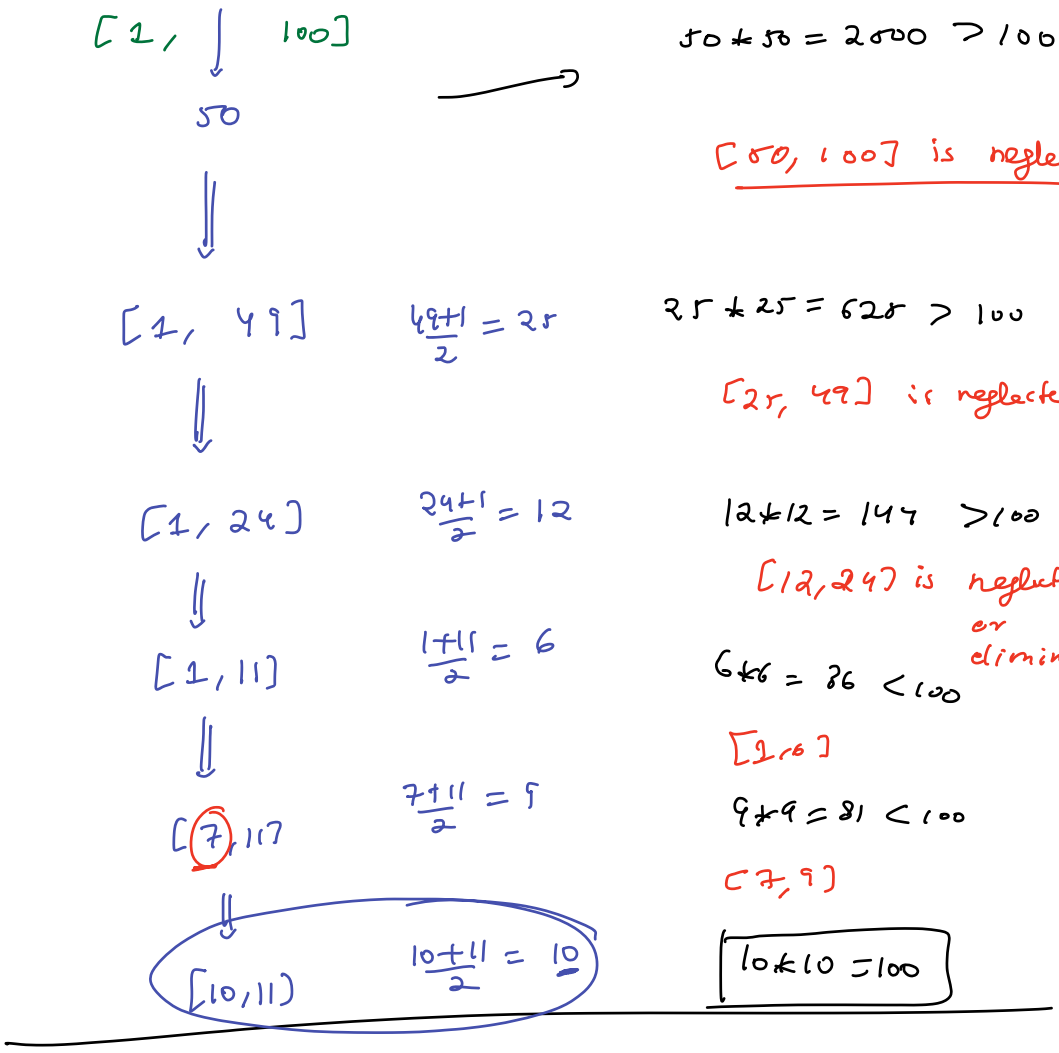
1, 2, 3

$1 * 1 < 100$

Q → How to neglect the max portion at each step?



Binary Search


[1, 100]
50
50 * 50 = 2500 > 100
[50, 100] is neglected.
[1, 49]
49+1 / 2 = 25
25 * 25 = 625 > 100
[25, 49] is neglected.
[1, 24]
24+1 / 2 = 12
12 * 12 = 144 > 100
[12, 24] is neglected or eliminated.
[1, 11]
1+11 / 2 = 6
6 * 6 = 36 < 100
[1, 6]
[7, 11]
7+11 / 2 = 9
9 * 9 = 81 < 100
[7, 9]
[10, 11)
10+11 / 2 = 10
10 * 10 = 100

pseudo-code

[1, N]

N/2 | N/2

① Find mid of range

low = 1
high = n

mid = (low+high)/2

② Check if mid is sqrt & update ranges

while low <= high

i) if mid * mid < N

— // reject left half

ii) if mid * mid > N

— // reject right half

iii) if mid * mid == N

return mid

No. of iterations

[a, b] = b - a + 1

	Range	
[1, 100]	100 - 1 + 1 = 100	= N
[1, 49]	49 - 1 + 1 = 49	≃ N/2
[1, 24]	24 - 1 + 1 = 24	≃ N/4
[1, 11]	11 - 1 + 1 = 11	≃ N/8
[7, 11]	11 - 7 + 1 = 5	≃ N/16
[10, 11]	11 - 10 + 1 = 2	≃ N/32
[10, 10]	10 - 10 + 1 = 1	≃ N/64

1

Observation

Every step we discard half of the range.

How many times divide by 2 to make 1 = $\log_2 N$

$N = 10^{10}$

$\sqrt{10^{10}} = 10^5$

$\log_2(10^{10}) = 33.22$

$\log_2 100 = 6.$

int(6.__) = 6

$N = 2^{60}$

$\sqrt{N} = 2^{30} \simeq 10^9$

$\log_2 N = \log_2 2^{60} = 60$

$[1, 1000]$ $\quad$ $[1, N]$

500

$[1, 499]$ $\quad$ $[501, 1000]$ $\quad$ $[1, N/2]$

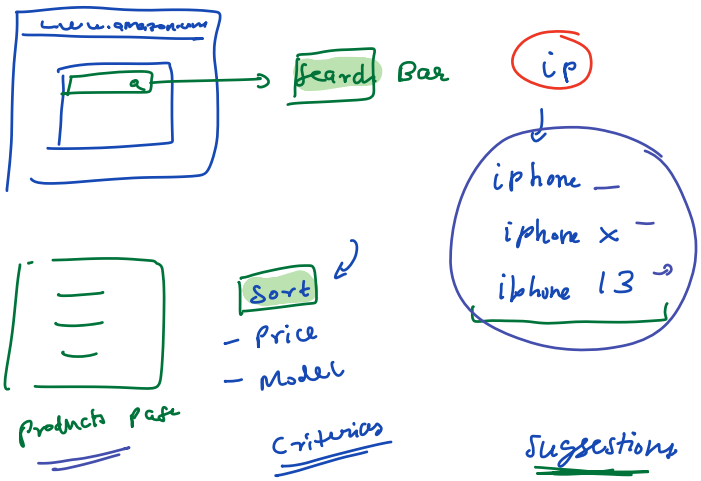
$N = 49$

$[1, 49]$	$\frac{49+1}{2} = 25$	$25 * 25 > 49$
$[1, 24]$	$\frac{1+24}{2} = 12$	$12 * 12 > 49$
$[1, 11]$	$\frac{1+11}{2} = 5$	$5 * 5 < 49$
$[6, 7]$	$\frac{6+7}{2} = 6$	$6 * 6 < 49$
$[7, 7]$	$\frac{7+7}{2} = 7$	$7 * 7 = 49$

⇒ Conclusions

⇒ Observation is the KEY to be better at Problem Solving.

Amazon



✓ → Data Structures

✓ → Algorithms

. steps to solve a problem

Undo / Redo ⇒ Stack Data Structure

Back / Forward

Mo ney bile nday

✓ Facebook: Mutual Friends Graph algorithms

BFS

✓ Linkedin: Degree of Connection.

Google Maps: Dijkstra's Algorithm.

Eng Dictionary: Searching for word "Book"

d

ba

Hash Map
Python Dictionary

(Logarithm) → [1, 2, 5, 10]
Base No

Appendix
Concepts of KC Verma.
Appendix

Intermediate Content

→ Mon/Wed/Fri
→ 9 PM

* 9 - 11 Content
* 11 - 11.30 Doubts

2 months Content

Time Complexity - 2

Arrays / Lists - 6/7 sessions
- Intro
- prefix Sum
- Carry Forward
- Subarrays
- 2D Matrices
- Interview Problems

Bit Manipulation - 2

Maths - 2

Sorting - 1

Strings - 1

Hashing - 2

Recursion - 2/3

Linked List Basics - 1

Stacks & Queues Basics - 1

Trees Basic - 2

✓ Problem Solving - 2/3

DSA Adv module (4-5 months after DSML)
- Adv Sorting
- Searching
- 2 pointers
- Stacks
- Queue
- LL
- Trees
- BSTs / Heaps
- Tries
- Greedy
- DP
- Graphs

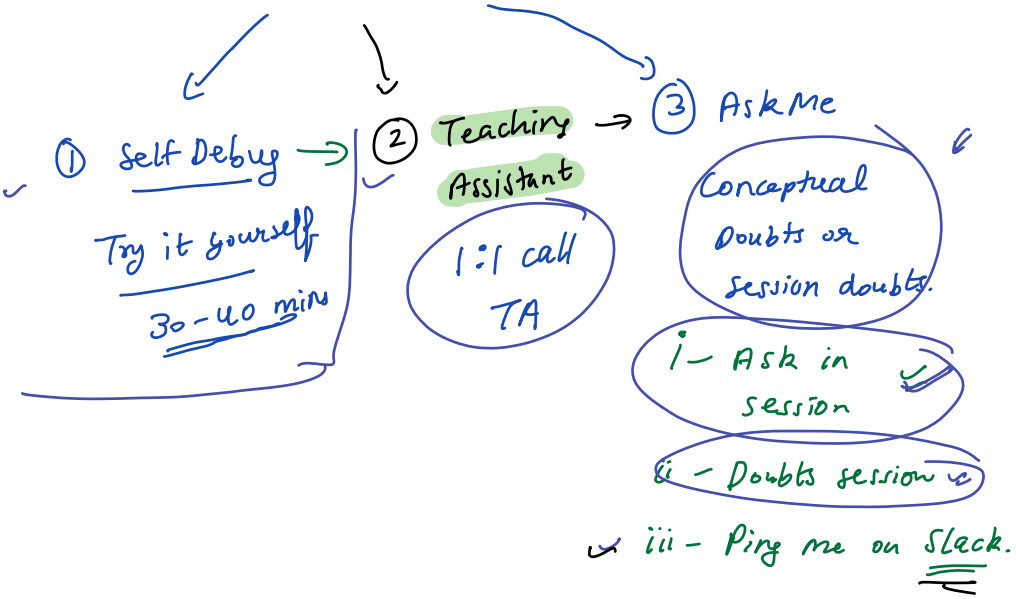
✓ Notes - uploaded (Lecture Notes in recording right side bar)

≃ 6-7 assignment + HW (Python 3.x)

≃ 20 problems Weekly

8/9 ≃ 160 - 170 problems

Assignment → Doubts



Doubts

Python Primer ⇒ Available in Dashboard

Optional

Refresher

All Classes → Optional Classes

×

Doubt Session

- Competitive Programming.
- Codeforces

Division 2 → A, B, C