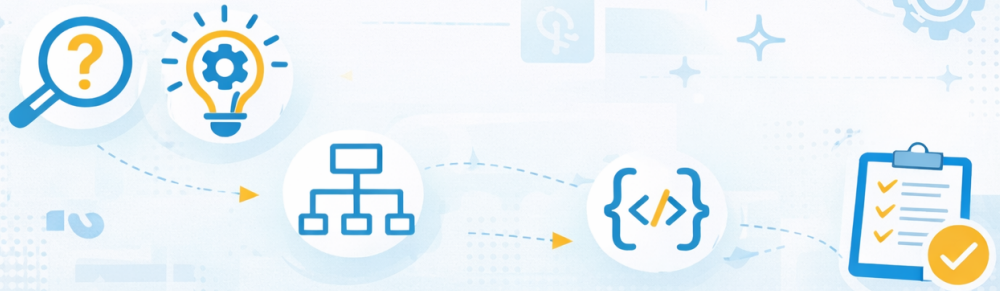
Code Greedy

How to Approach a DSA Problem & Develop Logic



What will we discuss

How Top Problem Solvers Think

How To Develop Logic

Code Quality

How to Approach a Problem

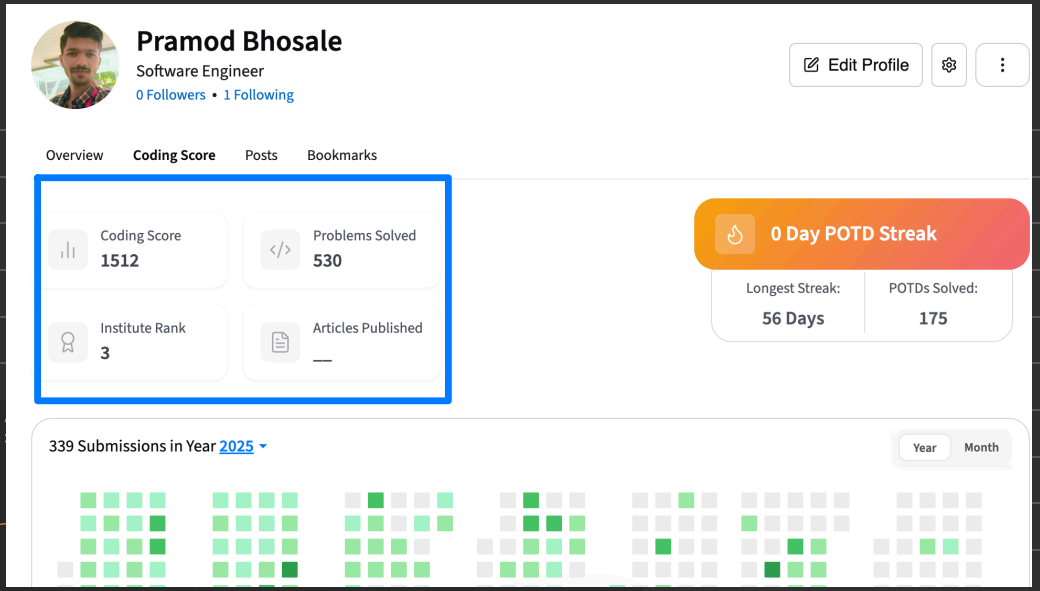
Perfect Order to Learn Data Structure

Which Language to choose.

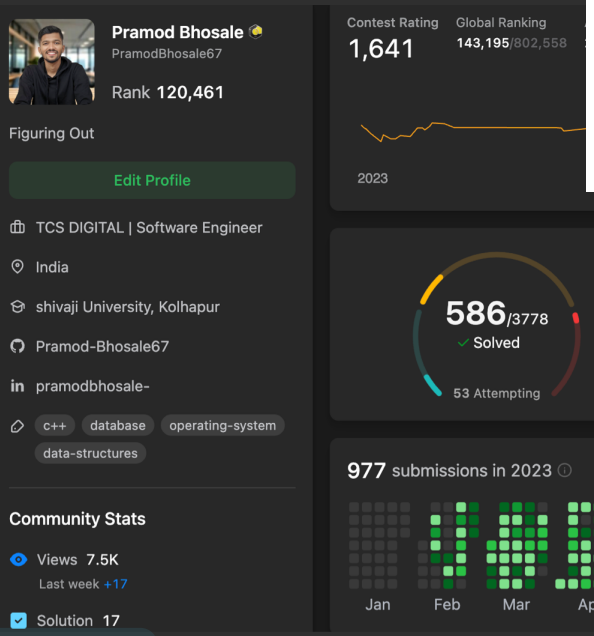
Pre-requisite for DSA

Platform to practice

Your Doubts....



Leetcode



GFG

Why DSA

Problem solving skills.

Highest Package -----> DSA vs Dev

Scalability ------> Wide Applications

Startup → less DSA / more development

MNC (TCS, infosys, wipro, capg) → fundamentals (2)

DSA / SD (

Sept 2022 → 7.5 l.

(3)

Dec → 11.5

(Barclay → 27

Product Based

Barclay / JPM / MS / G / A / M

Start Career

2nd → DSA → 3rd 2nd sem

3rd → M-F DSA ←

S-S → we

How Weak Problem Solvers Think

How many of you read a problem and directly jump to code?”

30 min

How many of you feel: ‘I understand the solution… but I can’t think of it myself’?

Which Language to choose.



Pre-requisite for Language

CPP: Standard Template Library

JAVA: Collections Framework

Python: Libraries

Perfect Order to Learn Data Structure

- ~~Array~~
- ~~String~~
- ~~2D Array~~
- ~~Searching Sorting~~
- ~~Recursion~~
- ~~Linked List~~
- ~~Stack Queue~~
- Bit Manipulation
- Greedy Algorithms
- ~~Tree~~
- ~~Graph~~
- Dynamic Programming
- Tries
- String Hard level
- Recursion Hard Level

Typing Speed Matters

Typing Master

keybr.com

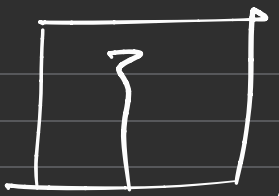
Typing master

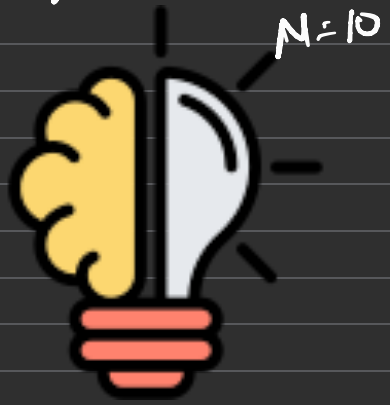
HOW TO DEVELOP

LOGIC

Start with Easier problems

- Problem to learn, Not to solve
- *Pen & Paper are you best friend* 3D
- When to see the solution
- Prove your logic, before submitting
- Consistency





WAY TO LEARN ANY TOPIC

01 Chose a topic which you're learning

02 Learn all it's theory & applications

03 Implement the DS & it's applications

04 Pick 20 Questions 30

05 10 Easy 10 Medium 15 5 Hard

From where to learn Topics or find problems

CodeGreedy's Pro DSA Sheet

Welcome to CodeGreedy's Pro DSA Sheet – Loved by Students, Trusted by Toppers
This isn't just another collection of problems—it's a battle-tested roadmap designed to turn you into a real problem solver, not just a copy-paste coder.

What Makes It Different?

📈 Zero to Hero Journey: Whether you're a beginner or brushing up for MAANG, this sheet takes you step-by-step from fundamentals to advanced topics like DP, Graphs

🎯 Handpicked Problems, CodeGreedy Style: Each topic is backed by carefully chosen LeetCode problems, simplified and explained with our signature easy-to-understand approach.

🧠 Structured for Placements: Patterns, strategies, and mindsets—all tailored for cracking real interviews, not just clearing topics.

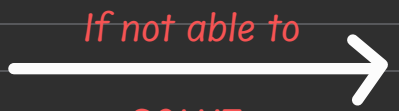
💙 Built for the Community: Thousands of learners have already trusted this sheet—not because of the name, but because it works.

How much time to give for a single Problem

5 min - Read the Problem

5 min - Understand the Input Output

30 min - Think Different solutions



Watch HINTs

Again think for 10 min

If not, watch Solution

Code Quality

HOW TO APPROACH THE PROBLEM

01 Read the Problem Statement.

02 Test cases & get the obeservation.

03 Think! How the results are obtained.

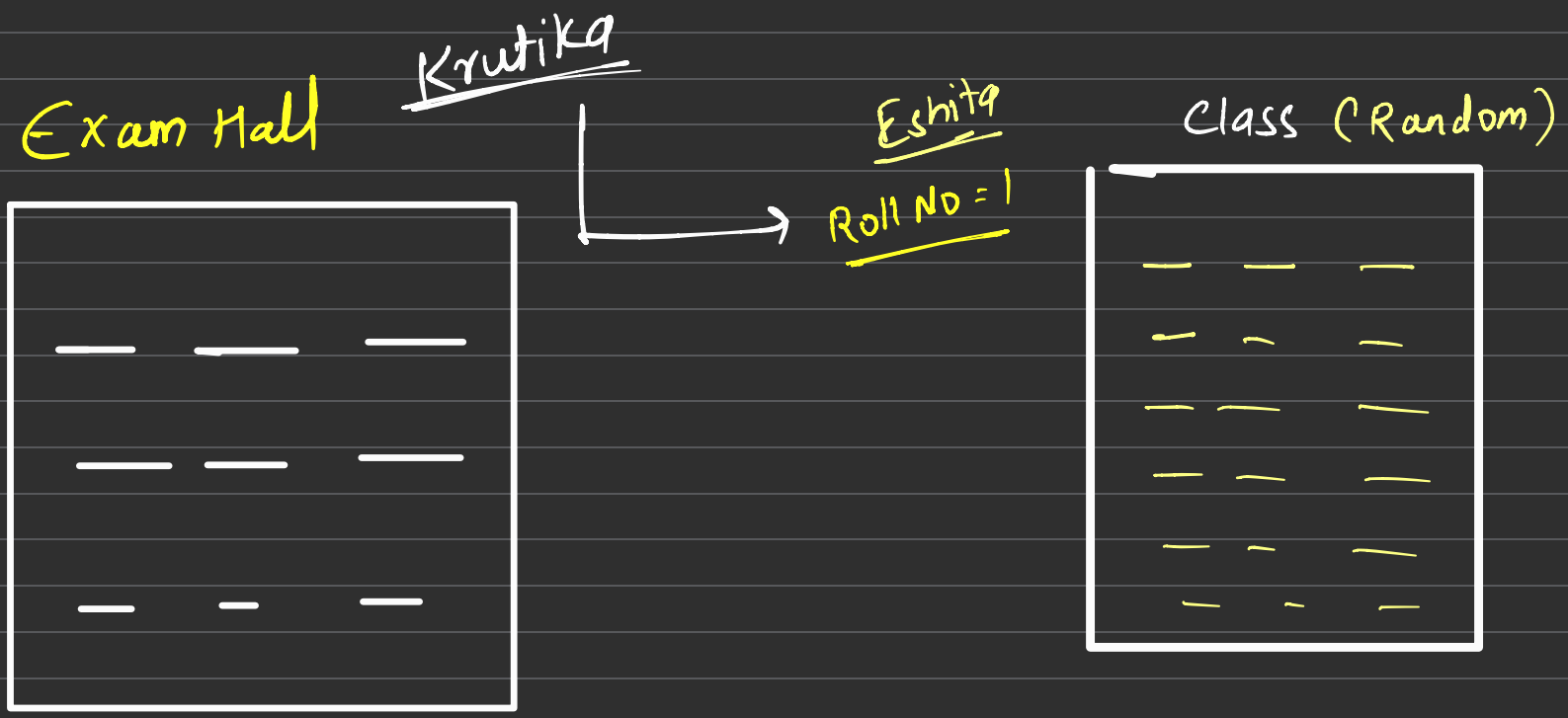
04 Now, think, which DSA can be used.

05 Brute -> Better -> Optimal

Platform to practice

Your Doubts

What is Data Structure And why do we need it.



Type of Data structure

Linear

Array, Linked List, Stack, Queue

Non - Linear

TREE, Graph

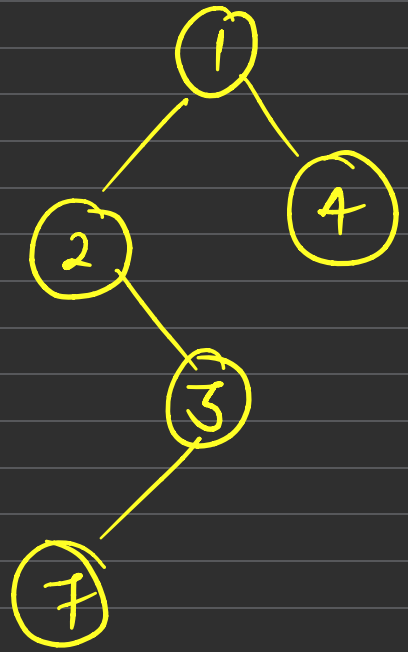
Hashing

Linear:

- sequentially order / linear order

Non-linear

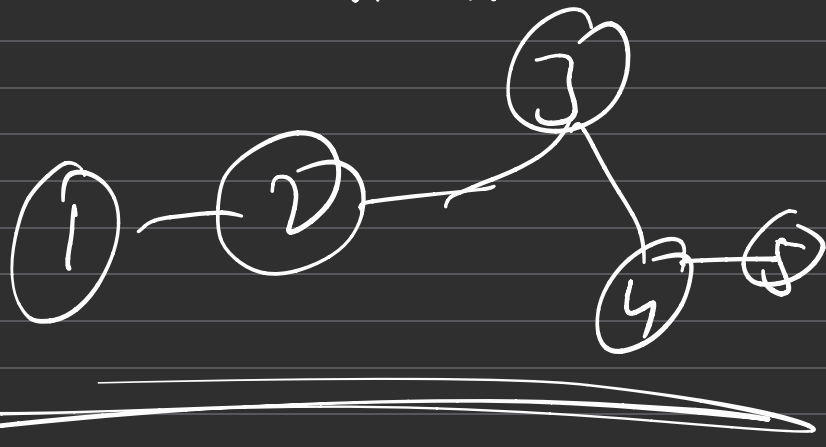
- Randomly
- level by level
- Dynamic



NL

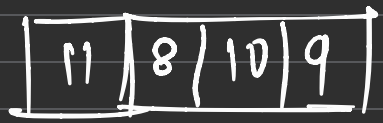
1	4	6	3

linear



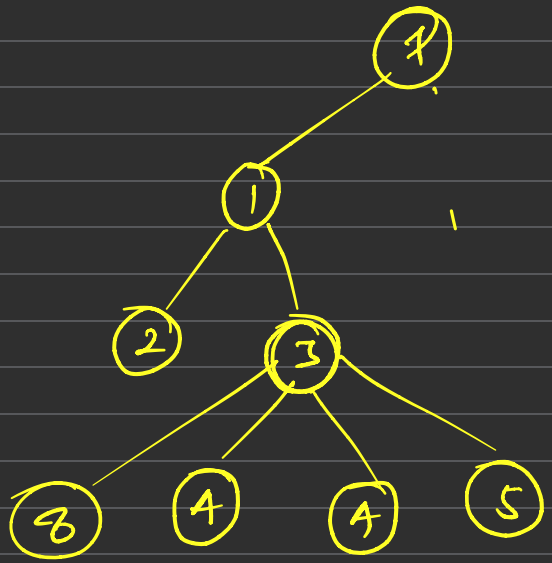
linear → Every element will have atmost 2 neigh

{0, 1, 2}

arr = 

→ 0, 1, 2

Non-linear: Every element will have atleast 2 neigh.



→ 0, 1, 2, 3, 4

Time Complexity

Amount of time req. for Execution of Algo.

	2008 Eshita ₹ 96,000	2020 Vrishali ₹ 40,000	2024 Sanchita Mac 1000
Sec			
hr	4	3	1
min	15	10	1

Time Complexity

- Best Case (Big Omega) Ω
- Avg Case (Big Theta) Θ
- Worst Case (Big oh) O

* Remember

  * Ignore the constants

  * Always consider higher values

  * Test Using infinite value
    (
    greater

N	$f(n) = 3n + 3$	$<$	$g(n) = n^2 + 2$
1	6		3
2	9		6
3	12		11
4	15		18
5	18		27
6	21		38
7	24		51
8	27		66

$2n^2 + 4n + 6$

$O(n^2)$

$5n^2 + 8n^2 = O(n^2)$

$\frac{n}{100} = O(N)$

$\frac{n^3}{100} + 8n^4 = O(n^4)$

$100n$ $O(N)$

$100 = O(1)$

$\frac{n}{100} + 2n$ $O(N)$

$n + n^2 + n^3 + n^4$

$O(n^4)$

N , $N \cdot \log N$, $10\,N \cdot \log N$, $\frac{\log N}{100}$, $\frac{N \cdot \log N}{100}$

Increasing order

$\rightarrow$ $\underset{10}{\log_2 N} < \underset{1024}{N} < \underset{1024 \times 10}{N \log_2 N} < \underset{1024 \times 10}{N \log_2 N} < \underset{1024 \times 10}{N \log N}$

$N = 1024$

for (i=1 ; i ≤ 100 ; i++) → O(1)

for (i=1 ; i ≤ N ; i++) → O(N)

for (i=n ; i ≥ 0 ; i--) → O(N)

for (i=1 ; i ≤ n ; i=i*2) → O(N

for (i=1 ; i ≤ n ; i = i*2)

$i = 1 = 2^0$

$i = 2 = 2^1$

$i = 4 = 2^2$

$i = 8 = 2^3$

$i = 16 = 2^4$

$i = 32 = 2^5$

$\vdots$

$2^K \leq N$

$\log 2^K \leq \log N$

$K \times \log 2 \leq \log N$

$K \leq \frac{\log N}{\log 2}$

$$K \leq \log_2 N$$

```
for (i=1 ; i≤N ; i++)        O(N)        O(N²)
{
    for (j=1 ; j≤N ; j++)    O(N)
    {
    }
}
```

$i=1$ $\quad O(N) \times O(N)$

$N=3$

$i=1$	$i=2$	$i=3$
$j=1,2,3$	$j=1,2,3$	$j=1,2,3$

```
for (i=1 ; i≤N ; i++)
{
    for (j=M ; j≥0 ; j--)
    {
    }
}
```

$O(N \times M)$

```
for (i=1 : i≤N : i++)
{
    for (j=1 : j≤N : j=j*3)
    {

    }
}
```

$O(N \times \log_3 N)$

7

```
for (i=1 : i≤N : i++)
{
}
```

O(N)

```
for (j=1 : j≤N : j++)
{
}
```

O(N)

4 O(N)+O(N) = O(2n)

input size	required time complexity
$n \leq 10$	$O(n!)$
$n \leq 20$	$O(2^n)$
$n \leq 500$	$O(n^3)$
$n \leq 5000$	$O(n^2)$
$n \leq 10^6$	$O(n \log n)$ or $O(n)$
n is large	$O(1)$ or $O(\log n)$

S.NO	Big O Notation	Name
1.	O(1)	Constant Time Complexity
2.	O(log n)	Logarithmic Time Complexity
3.	O(n)	Linear Time complexity
4.	O(n log n)	Linearithmic Time Complexity
5.	O(n^2)	Quadratic Time Complexity
6.	O(n^3)	Cubic Time Complexity
7.	O(n^y)	Polynomial Time Complexity
8.	O(2^n)	Exponential Time Complexity
9.	O(n!)	Factorial Time Complexity

Algorithm (applied to Array)	Time Complexity		
	Best Cases	Average Cases	Worst Cases
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Shell sort	$O(n \log(n))$	$O(n \log^2(n))$	$O(n \log^2(n))$
Merge sort	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$
Quick sort	$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$
Heap sort	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$
Counting sort	$O(n+k)$	$O(n+k)$	$O(n+k)$
Bucket sort	$O(n+k)$	$O(n+k)$	$O(n^2)$
Radix sort	$O(nk)$	$O(nk)$	$O(nk)$

5 X 3

15

```
for(i=1 : i <= N^2 : i++)
3
```

```
for(j=1 : j <= N : j++)
```

$O(N^2) \times O(N)$

$\Rightarrow$

$(N^2 \times N) = O(N^3)$