

Benefits

Starting in
mint

Competitive Programming

DSA (VS) CP
Dev (VS) CP

- Edge in Resume shortlisting & Referrals (TIER 3)

Practice

- Problem solving ability $\uparrow\uparrow$
 - Logical thinking $\uparrow\uparrow$
- } lot of help in DSA

- Online Coding Round \rightarrow Cakewalk (TIER 1/2)
 \searrow Pressure of time bound contest

- Interviews \rightarrow Expectation $\uparrow\uparrow$

\swarrow
sp Salesforce

Pre-requisite

#language

C++

STL

→ my notes

Java

+ Collection frameworks

→ OOPS

Java

→ Backend

Why to prefer C++ over Java?

→ Faster Execution

→ Editorials → C++

→ Standard Template library

→ Less verbose

Mathematics

DSA - Graph/DP (C&D)

- ① logarithms
- ② permutations & ~~Combinations~~ ^{★★}
- ③ Sequence & Series (AP & GP)
- ④ Basics of Geometry \longleftrightarrow
- ~~⑤~~ Number Theory & Combinatorics
 - \swarrow GCD & LCM (Euclid)
 - \swarrow Prime No & Factorization
 - \searrow Inclusion Exclusion Principle
- ⑥ Bit Manipulation
- ~~⑦~~ Modular Arithmetic \rightarrow $10^9 + 7$
- ⑧ Game Theory

Setting up

Code Editor \rightarrow Sublime Text (for C++)

• \rightarrow Starter Code

• \rightarrow Fast Input

• \rightarrow Codes snippets

• \rightarrow Keyboard shortcuts

\rightarrow Typing speed $\uparrow\uparrow$

typedefs

module (IO + 7)

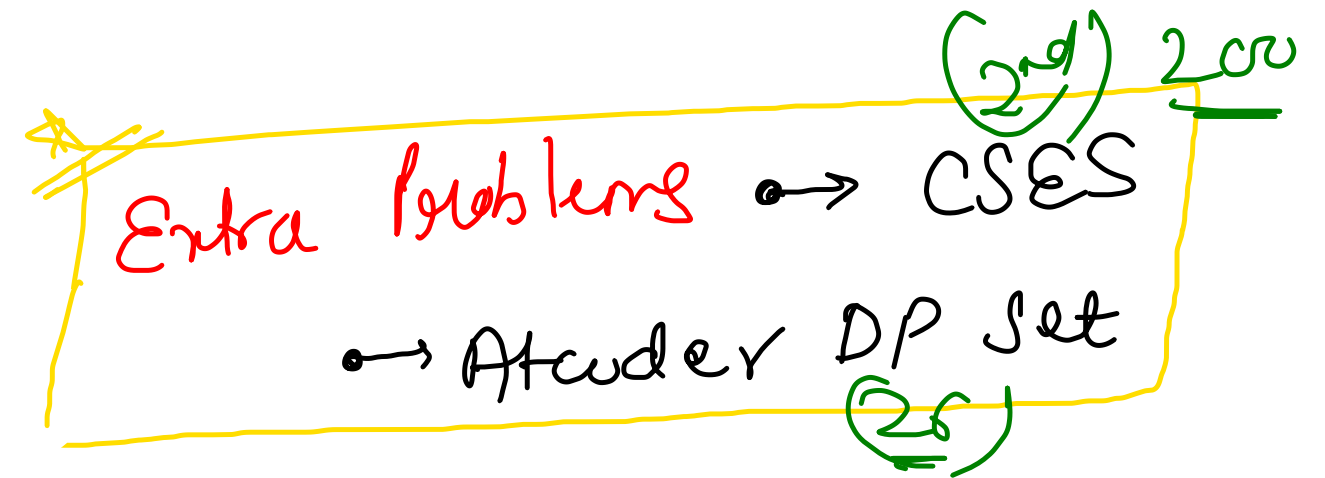
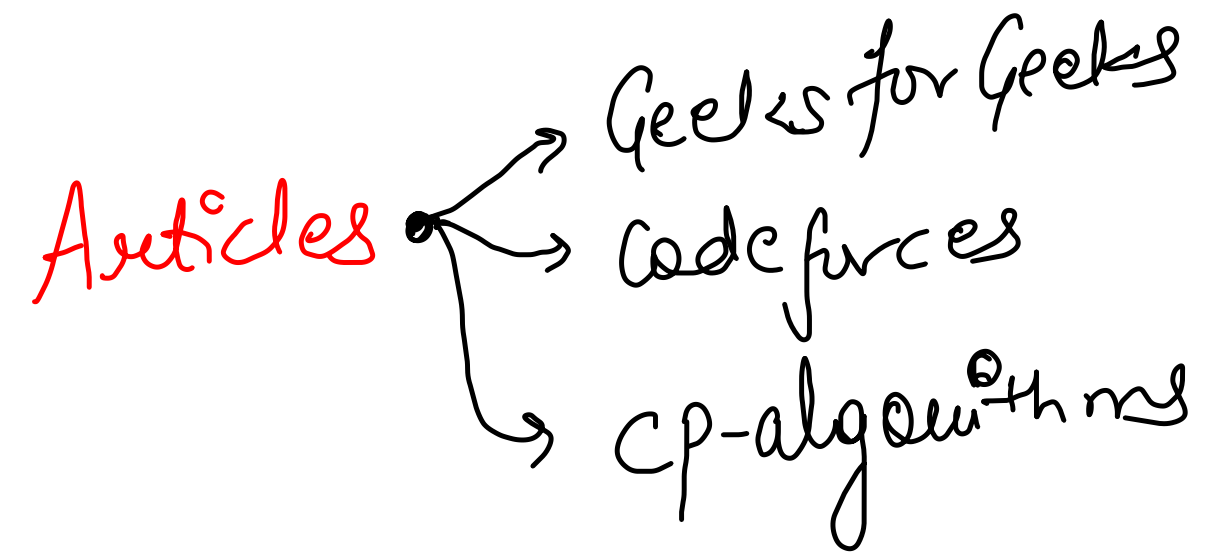
Bitwise operations

segment Tree, Number theory, etc.

<https://www.youtube.com/watch?v=8SPJHYDGIu0>

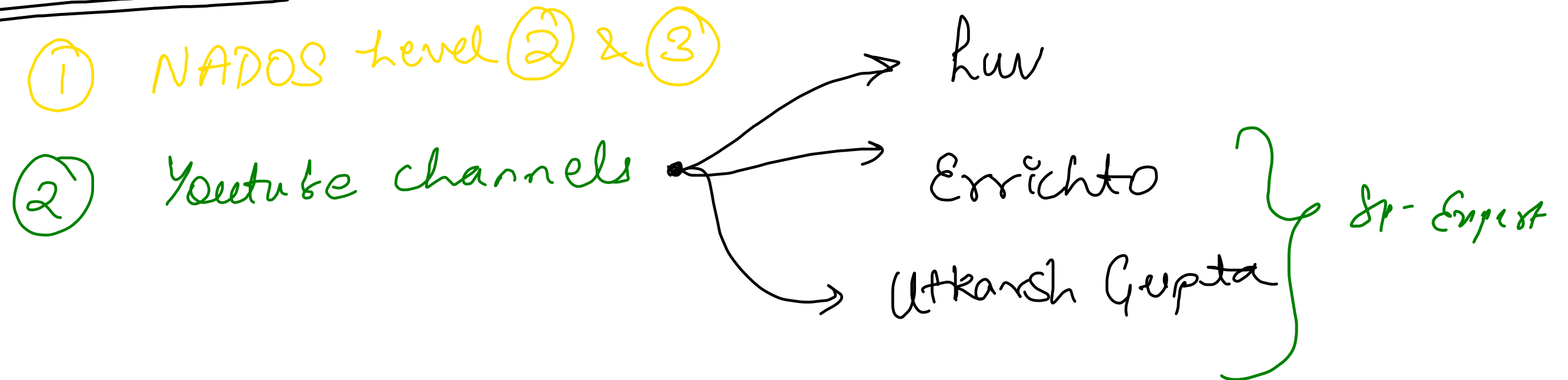
<https://www.youtube.com/watch?v=Mt6Jb8u9XBk>

} Chabon (*)



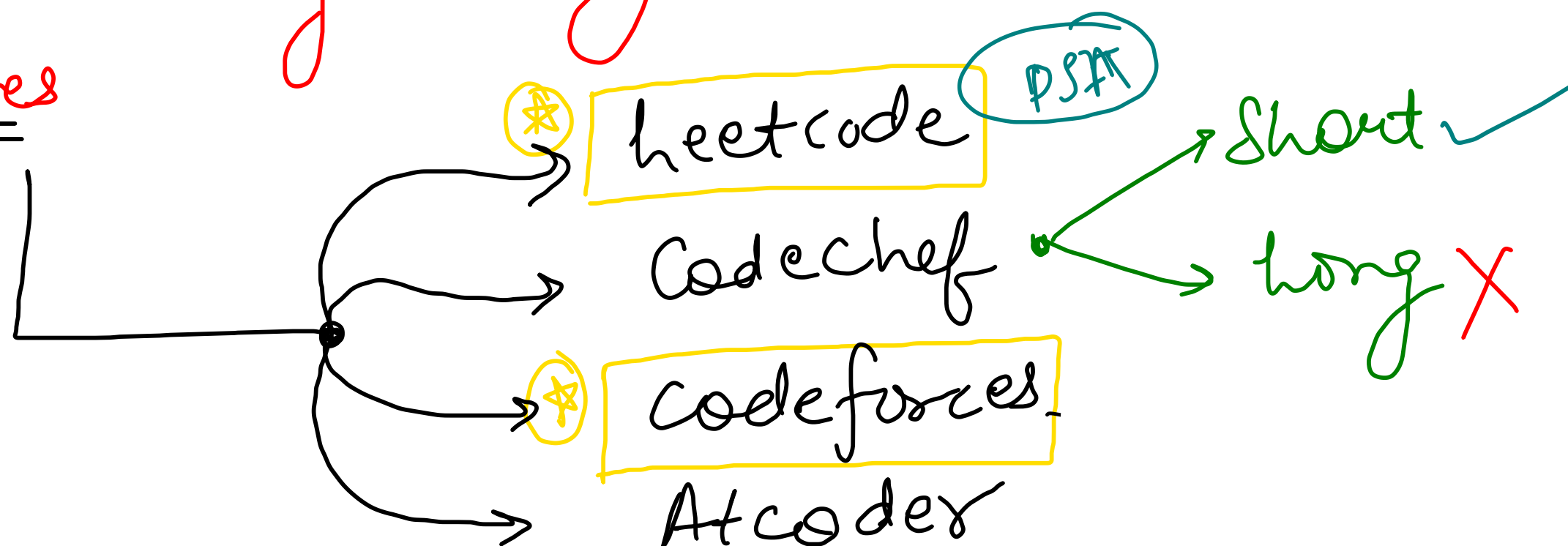
Video Resources

① NADOS level ② & ③



Competitive Programming Websites

* Pick one platform
& solve every
live contest



Big competitions

GOOGLE

- Hash code { in group of 2-4 } → Hackathon (21 Feb)
- Code Jam { 3-4 rounds } → Codeforces Div 1
- ~~★~~ Kickstart ^{★★} { monthly 2-3 hr contests }
↳ LeetCode & Codeforces Div 2

FACEBOOK

→ HackerCup → Codeforces Div 1

ACM ICPC

→ for Candidate Master & above!
(1st)

Codeforces

how to approach a problem?

→ Extract WHAT from a story.
↳ Examples (★)

→ Try to figure out which DS or Algorithm to apply

(★) → Check time constraint & code only if Time Complexity will satisfy.

trivial cases / boundary cases

→ Figure out corner cases
↳ Integer overflow
↓ TLE
↳ Stack overflow

→ Submit only if 90% sure!
{ Dry run on 2-3 ~~test cases~~ }

Suggestion ! → Check submission & acceptance count before trying

Div 1 vs Div 2 vs Div's
↓
Expert
onwards
↓
below
candidate
master
↓
below expert

Educational contests

↳ Problems weightage equal

Sample Test cases vs System Test cases

★ # UPSOLVING → Reading editorial part by part

★★ # Pair programming

★★ # Virtual contests
(Unrated)

2 ques

3 ques

100
20M
30 4