Competitive Programming

From Problem 2 Solution in O(1)

Training

Juniors Road Map

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Programming Competitions

Programming Prerequisites

- Basic Programming Knowledge
  - **Data types** … Be **very clever**
  - Variables, Conditions, Arrays, Loops, Functions, Recursion, STL, **Basic** Using Structure or Class
  - Read / Write to file
  - **Nothing** more (e.g. No OOP, Exceptional Handling, …)
  - Solve **exercises ..** do some programming **projects**
- Using **Debugger** cuts much of your time
- C++ **STL** is a great plus
- My C++ Series Covers the needed
  - Finish it. Do the projects. Read suggested book.

Getting started

- Watch newcomers series
- First 2 videos to know what and why
- Short plan
  - Watch Online Judge Video
  - Then watch Codeforces Video
  - Later, listen to others
- Long plan
  - Watch all of them
  - ICPC video should tell you about our formal competition

Main Levels

- New Comers
- **Juniors**
- Semi-Seniors
- Seniors

Training Plan

- Junior **target**: Being clever up to Div2-C
- You should always train in incremental way
  - E.g. focus on Div2-A till be good, then B, then C
- A sheet of ~450 is created for you: it answers
  - Which problems to solve? In which order?
  - What algorithmic knowledge to learn during that?
  - We initially learn the algorithm, solve few problems
  - End this phase with solving more on these algorithms
  - Algorithms selected to cover what usually appears in Div2-C

Algorithmic knowledge?

- For D2-CF-A and D2-TC-250
  - **Generally Nothing**! Sometimes little high school math
- For Division in CF-B and D2-TC-500
  - 90% Again **nothing**.
  - Little problems needs graph basics: DF / BFS
  - Basic Math: GCD, LCM, Modular Arithmetic, Geometry
- D2-CF-C
  - Half of problems may not need knowledge
  - Algorithms: DP, Greedy, Graph, Binary Search, Math
- Note, TC-500 is in range of CF-B and CF-C

Take notes during practice

- Nice trick you learned
- New approach to a problem..Thinking style
- Math note
- A bug you created
  - Doing same bug again and again is bad sign
- Revise your notes regularly
  - Or you will get little benefit from your practice
  - Remember...train **smart** not just hard

Thresholding the solving time

- For a junior, we **target** being **exposed** to many problems & solutions. It is **NOT** good at all to keep solving for **hours** in a problem
- Let the maximum time per problem ~**2 hours**
- Think in the problem solution.
  - As long as you have ideas => Keep Thinking
    - If have solution, verify + estimate its time and memory
  - When stuck => Think maximum for extra 30 minutes
  - If still stuck, **read** the editorial (or solutions if needed)
    - If can't understand within 30 minutes, skip it (put in TODO list)

Thresholding the solving time

- Coding it
  - Have vision for the code => Go with it
  - When stuck => Think maximum for extra 30 minutes
  - If still stuck, **read** others solutions.
  - No way => Skip it. Otherwise, Recode it
- Failed in a test case?
  - Revise the code. Generate some small cases and trace.
  - Revise problem statement.
  - Try for 15 minutes, then see the wrong test case
- Once done: See editorial / other's code.
  - Rewrite code..e.g. to cleaner...shorter..faster code

Todo list for hard problems

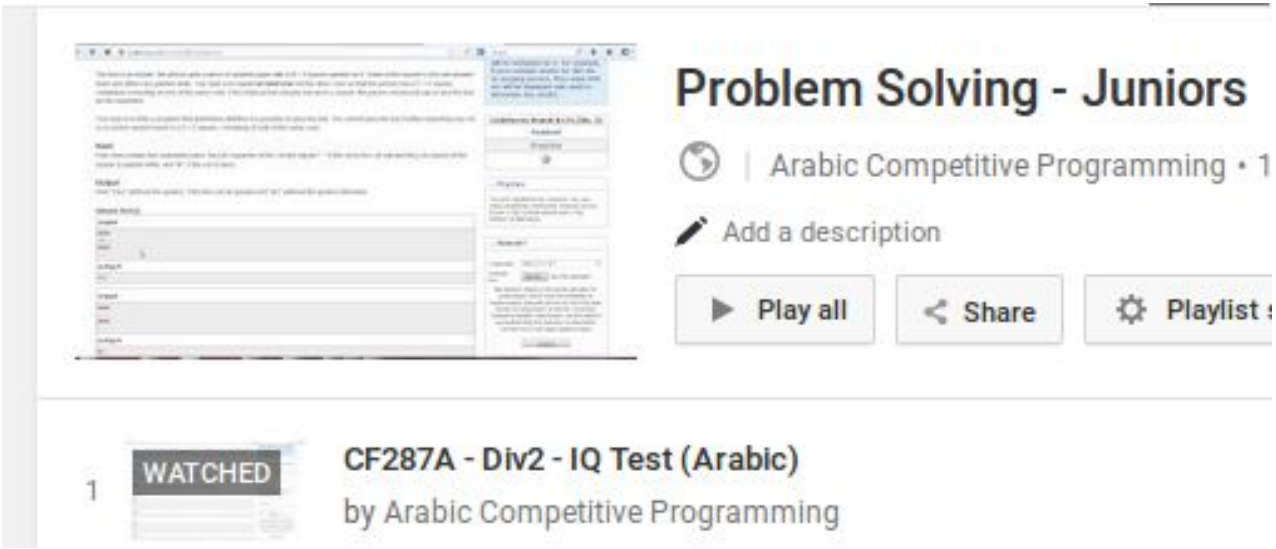
- Hard Problem Properties:
  - You couldn't get it idea or code by yourself
  - Or you kept getting wrong answers
  - Couldn't solve and don't understand editorial/solutions
- Put it in **todos** and retry later
  - Try ot later again
  - Did very well => cool..remove from **retry todos** list
  - Otherwise (can't solve it, or made many mistakes)=> Put again in TODOs
- If you couldn't do it for missing algorithm, when learn knowledge => learn it

Have Coach, Supervisor or Nothing?!

- **Coach** is someone doing close training to a trainee (e.g. hints/coding/debugging)
- **Supervisor** communicates regularly (e.g. each 2 weeks): gets status / gives feedback.
- Both helps to tell you what to do/solve next
- In the junior phase, you **DON'T** need both
  - you just need to practice with yourself!
- Even when you grow up, nowadays you can depend on yourself and perform great.

Thinking Process

- Videos for juniors are created to show you how one can think. Try them, then watch video



Moving faster in sheet?

- We are not equal! Some can take much time to be good in Div2-A...but others can jump
- Try harder problems in same level...or try next level..if you can go, jump to this level
- If you are in some level, try next level:
  - doing bad? Back to the last level
  - doing slow? Work on this level
  - doing well? Jump to next level
- Remember to threshold in solving and see editorial. Max 2 hours per a problem

Terminologies

- **Ad Hoc problem**
  - The problem doesn't belong to any well studied problem. So problem and its algorithm are **unique**.
    - Shortest path is a well known problem with several algorithms.
  - So you **don't need** to study algorithms to solve it
- **Brute Force Technique/Search**
  - **General** problem solving technique..**ZERO** thinking
    - It can be written iterative or recursive...based on problem nature
  - Try **every** solution and pick the right now (that stupid)
  - E.g. given sorted list of numbers, find 2 numbers with sum 1000. **BF**: Try every 2 numbers in O(n^2)
    - A smarter solution will make use of the **sorting** property.

Terminologies

- **Simulation** Problem: A problem that tells you exactly what to simulate in steps
- **Implementation** problem: Any problem with much coding. It enhances coding skills
- **String manipulation** problem: Problem with focus on playing with strings
- **Constructive algorithms**: Hmm

Terminologies

- **Greedy - Two Pointers**: you can solve them without knowing what are these :)
  - Greedy is a technique to be studied later (vs DP)
  - Two pointers is just ad hoc problem!
- **Others**: Dynamic programming (DP), Binary Search, Shortest Path, DFS, BFS, Flow, 2-SAT, Fenwrik Tree, Segment Tree, ….
  - You will be exposed to some of them during this phase

Sheet Demo

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً