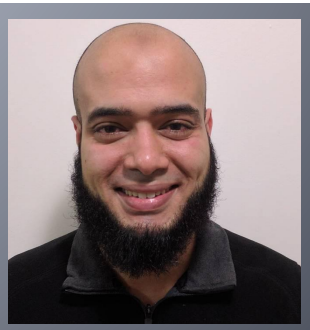
Competitive Programming

From Problem 2 Solution in O(1)

ACM ICPC Training

The Contest - 2 - Strategies

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University



No-teamwork Strategy

- No collaboration
  - This strategy fits strongly with the very strong teams
  - Each individual can get many of the contest by himself
  - Split the problems somehow, each one solve what he can among them
  - When it comes to really hard problems, switch to team!
  - Terminal can be blocked! Write on papers & sync!
  - This strategy fits too with teams that did not practice team working. So in contests, be individual as long as applicable. Then switch to 2-3 guys collaboration

3-in-1 Strategy

- Triple work
  - The 3 members do the same thing together.
  - Read together. Think together. Solve together
  - Maybe at coding time, 2 members code together on the terminal, while the 3rd member explores more problems.
  - This strategy might be used in the last 1 hour of the contest, where problems became so hard!
  - Also juniors who lack most of the skills may use it

Imbalanced team strategy

- 1 strong, 1 medium, 1 weak team members
  - What if a team is highly imbalanced
  - Let's say, contest has 10-13 problems
    - The easies 2-3 of them within the weak level guy
    - The medium 2-3 of them within the medium level guy
    - 2-3 of the hard problems are within the strongest guy skills
  - Let each one solves the highest problems within his level
    - E.g. The weak guy solves the easiest problem in 14 min / 2 submissions
    - Note that, the strongest can get it in 6 minutes only / 1 submission
    - But if he did that, he consumed his valuable time for something someone else can do, but they can't do what he can solve!
  - Overall: Proper utilization of team skills

Terminal man Strategy

- 2 strong + 1 weak but fast team members
  - Terminal man strategy is to use the **weak** member as the **main person using the computer**!
  - While the other 2 members think, decide, **code on paper** the next problems.
  - The 2 members do the **minimal** things such that the terminal guy code/finish remaining subtasks
  - E.g. Terminal guy tasks: code the IO parts / Sample input / Extra cases / Functions within his levels + type from the paper what the other 2 prepared / some debugging
  - Cons: *Terminal man may be confused about some functions and keep interpret the other guys*

Think-Tank Strategy

- Think-Tank
  - Purpose: **In the first hour**, the 2 strongest members read **all problems** and **decide** what will be solved in the contest in which order. For each problem, they wrote over it the possible **notes**: Its level, algorithms
  - Within the first hour, the **3rd member** prepares the code templates and solve the easiest 1-3 problems.
  - Then, the 3 members finalize their decisions
  - Each hour, they revise the scoreboard and make sure the decisions are still valid

ACPC team strategy Example

- Customize a strategy
  - Discuss today concerns with your team
  - From each concern/tip, decide how to make use of
  - Develop your own strategy that fits with your team skills
  - Let's develop 1 team strategy
- Assume Team skills
  - The 3 members are seniors: Very knowledgeable / skilled
  - Assume their order based on CF/TC profiles
  - *Ziad = Fast coder, low rate of bugs, individual behaviour*
  - *Saad = Fan of analyzing, like pair thinking, Not fast*
  - *Hani = Not biased to a specific concern*

ACPC team strategy Example

- The opening (1 problem, the ace)
  - Goal: Find the easiest problem
  - Ziad: Jump to the machine - prepare the code templates
  - Saad: order problems based on length
  - Saad / Hani: Inspect the problems very fast to find the easiest and push to Ziad to code
- The early game (1-2 problems)
  - Goal: Catch the other 1-2 easy problems
  - Keep your eyes on the scoreboard for new ACs
  - Avoid/Minimize idea/code reviews -Ziad on machine

ACPC team strategy Example

- The mid game (2-3)
  - Team members read at least 75% of the problems
  - Identify mid-level problems - **reviewer role - code on paper -** Owner **types** code to the terminal
  - Short codes directly to the machine NOT on papers
  - Check point after 1.5 & 3 hours
  - Read all papers by 1.5-2 hours
- The end game
  - Don’t open new problems if multiple non ACs
  - Start with *1-code-2-think* with **rotations**
  - Switch to *3-in-1* when problems are very hard

ICPC finals team strategy notes

- The final battle of the brains
  - The contest is tough and causes much pressure
  - Without proper individual and team training = many mistakes occur even if members are strong
  - Typically other teams lead the scoreboard. You will recognize the easy 1-3 problems reasonably
  - For problems 4-6 the scoreboard also will highlight them but in a little rate. It might be misleading scoreboard
  - Avoid starting in a problem that not one tries so far!
  - Avoid problems solved only by the top 10 teams (it can be 8/10 success rate, and feels promising, but it is not)
  - Very early switch to 2-3 members per a problem

Other tips during the contest

- A tracking sheet
  - For each problem in this sheet
  - Who tried it, whatever mini-comments
- Got problem AC! Congrats!
  - Get rid of all its related paper (save time)
- Submit and print
  - If you should leave the PC after submission to a college
  - After submission, print directly in case needed debugging
- Opening new problems
  - If team has many non AC problems after 3.5 hours, don't open new problems. Focus to get these problems first

Other tips during the contest

- Stuck and can't get an idea
  - Discuss with other team members
  - Try other problems
  - Go to toilet or go for prayings [e.g. muslim]= Relax
- Stuck with several WAs!
  - Rewriting the whole solution in a bit different way?
  - Or, relax / Try other problem
  - Check your WA list
  - Generate cases and compare your solution vs BF one
  - Don't let it eats your time. After 3 non ACs, leave it and try later. After another 2 failures, ignore it to last hour

Other tips

- Individual discipline
  - Individually, develop your guidelines for the different phases: Reading, thinking, coding, testing & debugging
  - E.g. in **reading**: Trace all samples
  - E.g. in **thinking**: Keep it sample, what is the possible orders, brute force is enough, check algorithms list...
  - E.g. in **coding**: think from different perspectives, code less, use meaningful variable names
  - E.g. in **testing**: Develop extra 2-3 cases. Check boundaries, stress test if needed
  - E.g. in **debugging**: Decide which is faster: Print statements vs using a debugger

Useful Resources

- Contest Strategy resources
  - Link 1, Link 2, Link 3 Link4, Link5
- My problem Cheat sheet
- Strategy samples

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً