Boosting contestants performance

The Discipline-Monitor Approach

Vision written by Eng **Mostafa Saad**

Performance problems?

- Juniors
    - Much Training – Weak Performance
- Seniors
    - Plateau Status

HOW ... according to top world finalists?

Hard Practice

1. Practice
2. Go to 1

Problems Decomposition

- No magic - matter of discovering sub-problems you experienced
- Just Practice as much as you could

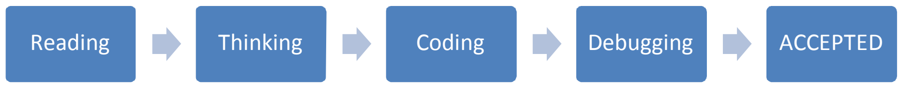
Effective Training

- Just practice and practice is not enough
- Tough problems requires thinking out of the box
- Work on hard problems, think NOT memorize sub-problems

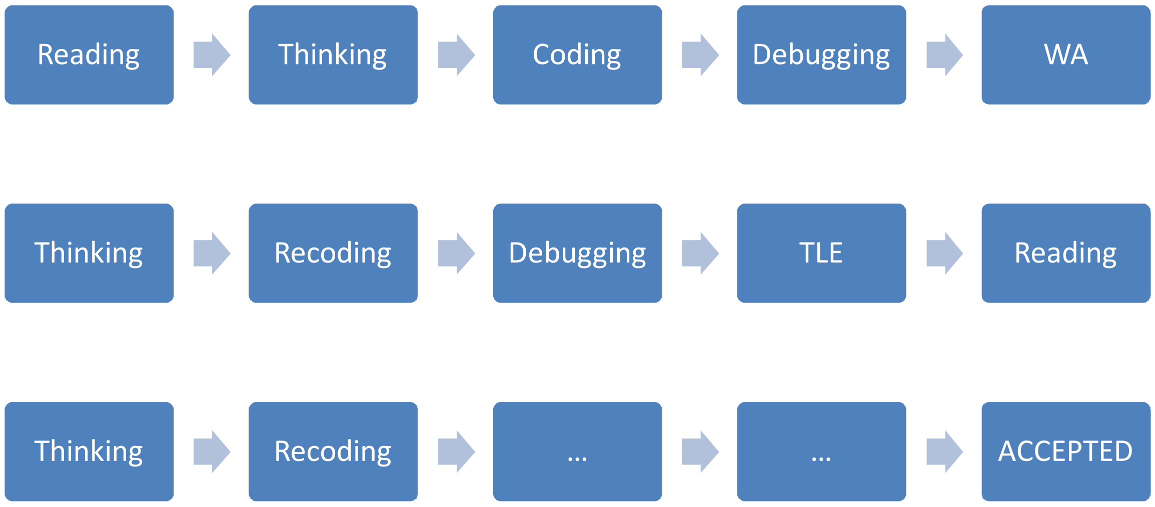
Following is proposed approach to push automatically toward **effective overall training**.

Back to Juniors

Problem Solving Phases?



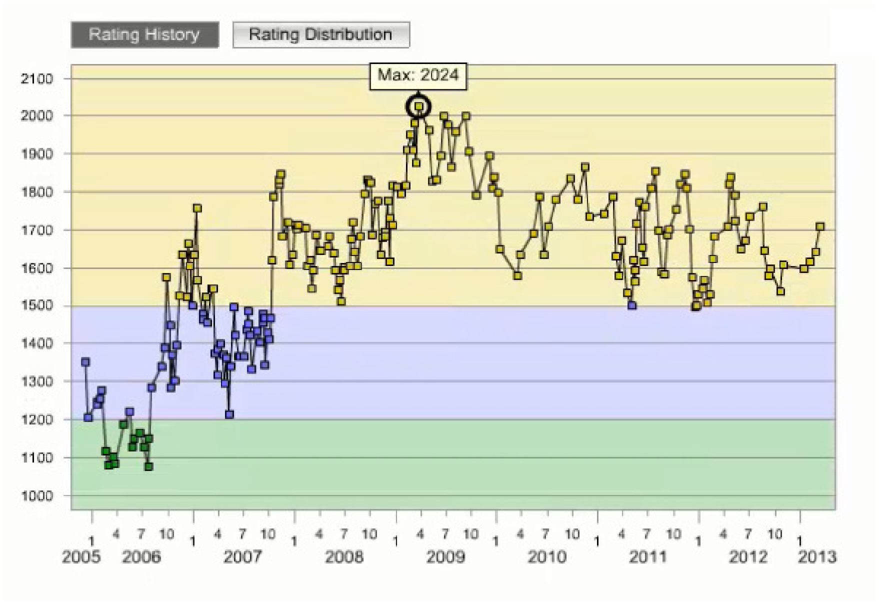
What actually happens?



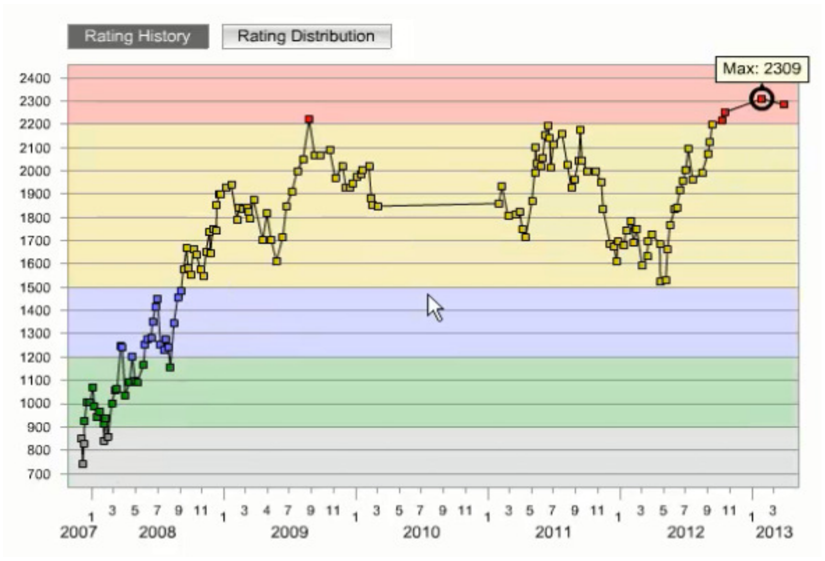
After solving 3000+ Problem?

We become **seniors ...** How is our **performance**?

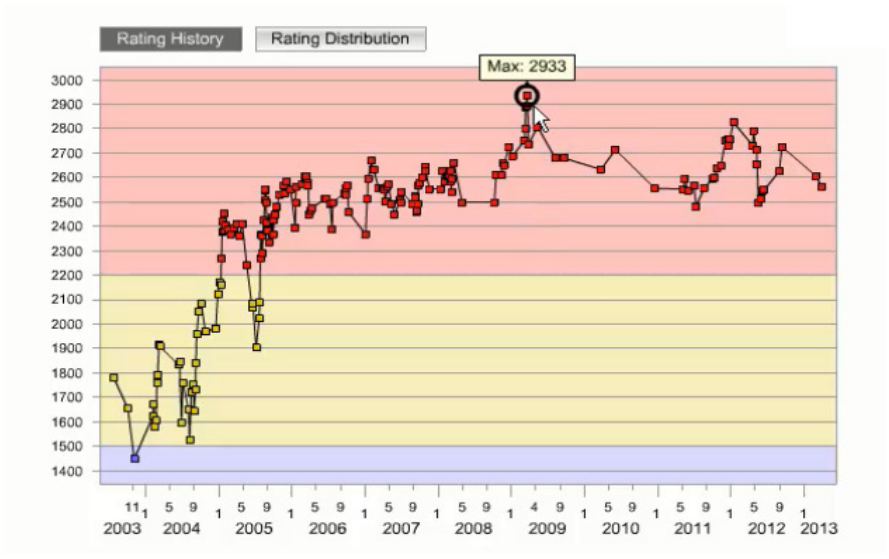
For example:



Or



BUT NOT



Time for **Simple** but yet **Powerful** approach

The Discipline-Monitor Approach

- It is an optimized trial-and -error approach.
- About how to **behave** correctly to end at the ACCEPTED status.
- We might call the behavior of most contestants with chaos .. solving/training randomly
- Little discipline cases (E.g. Contest Strategy), and rare monitoring
- What do we mean by discipline? by monitoring?

Discipline

- Establishing guidelines, rules, or checklist to follow during performing a task.
- It is matter of behaving systematically
- Develop discipline for everything!
    - Problem phases (Reading, thinking, solving...)
    - Training phases
    - Contest Strategy
    - etc
- It is tough to do mentality shift. You need to enforce yourself till it be your main behavior

Reading Discipline Example

- Read carefully, make sure no statements conflicts
- Extract constraints info, don't make assumptions
- Record weird constraints
- Trace Samples as long as they are traceable
- Think in Missed, Boundary & Especial cases
- Revise carefully output cases and formats

Passing the Plateau?

- It is mainly about thinking skills.
- It is tough to think out of the box
- Thinking Discipline helps too much

Thinking Discipline Example

- Think: Problem Simplification
- Think: Concretely, Symbolically & Pictorially
- Think: Problem Abstraction & Simplification
- Think: Divide & Conquer problem
- Think: Forward & Backward
- Rank & Attack Ideas

More disciplines already explained in **How to Practice** Series.

Monitor

- What if we have discipline but we don't follow?
- What if we have discipline but it doesn't achieve its targets?

- We need to setup goals from the discipline
- We need to review the result of the discipline
- We need to criticize the discipline and update again
- We need to monitor time for tasks to detect the bottlenecks.

Log as much as possible

- 7:00 - Started Reading Problem
- 7:05 – Prepares the constraints section
- 7:07 – Tracing sample 1, 2 is long
- 7:10 – Started Thinking
- 7:12 – Brute force will TLE, could be optimized?
- 7:20 – What about greedy? Not comfortable
- …
- 7:40 – Let's code this DP approach
- 8:10 – Need to debug!
- ……..
- YES, accepted!

Log analysis

- Followed discipline for each phase?
    - Yes
        - Which phases are Long?
        - Performed well?
            - No, How to update discipline to avoid that?
        - …. Whatever possible analysis to be better
    - No, Need more enforcing!

Passing the Plateau?

- It is very important to know how your mind works, How does it order its solving steps?
- Say you thought about 2 possible solutions, which one your mind started with?
- Why? Randomly? Or by intuitive?
- If you picked a thinking path of one of them? You keep in it till exhausted or back to 2nd one earlier as it is now more promising?
- Each time you fail to find the solution, you should revise the log and ask
    - Was the solution part of my skills?
        - If yes, Why I couldn't do it?
    - If you discarded a correct thinking path, how to avoid that next time?
    - If you will choose among 2 possible thinking paths, how to select the promising one correctly?

Performance Boosting Factors

- Major Factors
    - Problem solving skills: Which Skills? How to enhance?

    - Training Principles
    - Flat Training hierarchy
- Supporting Factors
    - Community inside an institution
    - Community among the institutions

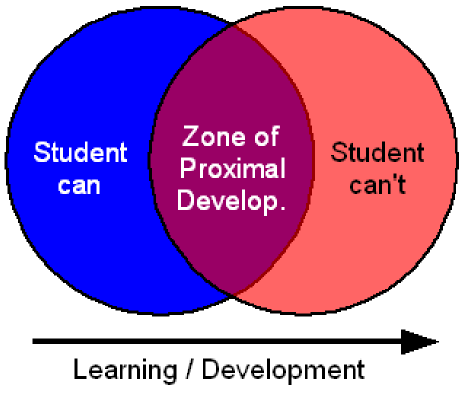
Problem solving skills

- Problem Reading Skills
- Thinking skills, the critical part
- Verification skills
- Implementation Skills
- Debugging Skills
- Testing Skills

- Optimizing the pipeline phases = boosting performance
- Optimizing = efficiency and effectiveness -> requires Discipline and Monitor
- Check my video series for their suggested discipline in depth
    - **Google** Algorithms Mostafa Saad YouTube

Training Principles

- Pump up the challenging spirit
- Work Smart not Hard
- Never to Memorize
- Never to use library during training
- Don't let time stress corrupt your training
- Practice as if you are in a contest
- Depend on variety of training source (TC, UVA, IOI, CodeForces, ...)
- Balance the training among different solving skills
- Recognize your weakness points, and attack them
- Selective problems to match your Zone of proximal development



Flat Training Hierarchy

- Training might be layered to 3 levels: Juniors, Semi-Seniors and Seniors
- Coaches manage seniors. Each senior manage 2-3 semi-seniors. Also Seniors & Semi-Seniors manage the juniors training. We might remove the semi-senior layer.
- When some juniors are good enough, they become semi-seniors. When some semi-seniors are good enough they become seniors. "Good Enough" is a criteria defined by coaches and the top seniors.
- Coaches should define what are the training plans for all layers. What are the expected short/long term achievements of the plan?
- Coaches should build software tools to measure different statistics for contestants. E.g. who fast is the coder? How many submissions needed for code to be accepted?...
- Coaches should define & announce the criteria to select the teams sharing in the formal contests. They should strictly follow what is announced. When it is about criteria for technical issues (e.g. teams selection).
- **Monitoring** meeting is done once or twice yearly to review the teams status, different criteria, training strategies and update the plans if required.

Junior Training

- Introduce the ICPC competitions and the future benefits from sharing
- Guide them to the judges and how to use them
- Training is 2 levels.
  - In **level 1**, run an online qualification to select the ones who solve some trivial problems
  - In **level 2**, qualification is based on a stable performance in online contests (e.g. green in TC)
- Define for each level its targets and divide it to set of phases to achieve targets
- **Level 1** Targets could be: basic think skills, coding skills. Don't explain any algorithms.
- **Level 2** Targets could be: basic think skills, coding skills, basic techniques (E.g. DP, BS..)
- Problems should be **well** selected, to achieve the targets.
- Define **Expectations**: E.g. Level 1 Expectation is a stable green in top coder. Level 2 Expectation is a stable blue in top coder. Or similar measures in code forces

Semi-Senior Training

- Survivals from level 2 should be ordered based on performance. Seniors should pick 2-3 contestants to work hard on them
- Training style could be: Pairing
  - Pairing styles could be: Enemies, Twins, Semi-Twins or Guide-Review style.
  - "**Enemies**" Style: 2 members work against each other
  - "**Twins**" Style: Both work together in all problem phases, Typically, applied on hard problems

        - "**Semi-Twins**" Style: Both work together in finding the idea only, then both work individually on coding the problem
        - "**Guide-Review**" Style:
            - Contestant1 solves the problem from A-Z. Prepare good hints for the problem
            - Contestant2 starts to solve the problem
            - Phase by phase, Contestant1 reviews the application of the Discipline-Monitor approach over Contestant2
            - Whenever Contestant2 is stuck, Contestant1 gives him a hint
- Training targets might be
    - Apply Monitor-Disciple approach extensively
    - Solving process important than the acceptance of a problem
    - 75% is hard ad-hocks and techniques according to their levels. 25% for knowledge: Techniques, Data Structures then algorithms
- Intended to be long term training

Seniors Training

- In this training, remaining important knowledge and experience transferred to seniors
- Hard problems on algorithms
- Working on seniors weak points
- Individual contests to measure seniors levels
- Teams constructions - Establish discipline for contest strategy
- Start to solve 2 contests weekly in the start, 3 when near to formal competition
- After contest
    - Revising the performance of every team. Monitor applying the discipline
    - Discuss possible solutions for unsolved problems
    - Solving form the unsolved problems as possible

Community inside an institution

- Each Institution should push as many as possible students to join the competitive programming community
- Community should attract young students (High schools) to share. Very important to start early as possible.
- A group of 10 is not like a group of 40. A group of 50 is not like a group of 100
- More competing students = much challenge spirit, the fuel to improving performance
- Institution, its coaches and contestants should work all the time on finding new students, and describe for them the future benefits of joining competitive programming
- The most promising status might be when it is shame to not belong to this community, and if belong to it, it is shame to not be part of its top seniors
- Each Institution should provide enough resources (Labs, Money for registration, Running camps, prizes for local contests, ...) for coaches and contestants
- Each Institution should ease on students academic related issues (E.g. Midterm is on contest day)

- Each Institution should maintain the hierarchy of the coaches, seniors, semi-seniors and juniors. Some professors/coaches role should be maintaining and revising the hierarchy.
- Institution should push all toward the volunteer work, but It should do whatever required to keep the hierarchy up and running. E.g. Coaches might be hired as full/part time employees to minimize the continuous leaving of coaches & seniors from community.
- Each Institution should try to run a camp from time to time by world finals leaders. The main purpose is to discuss the training phases and the applied criteria for teams selection.

Community among institutions

- Institutions should collaborate together: knowledge and experience share is a must.
- Yearly meeting among coaches and their top seniors is hold. Share the training phases, problems and suggestions.
- If possible, coaches swap might be good for in-depth sharing.
- Many local warm-up contests should be carried. Competitions among teams will push the spirit and alarm teams to work harder and better. Online Contests where teams can share should be target to all communities. Scoreboard for results should be collected. This will need good arranging between Institutions
- Similar well managed advices should be applied among countries sharing in ACPC.