# ­­­­­Tackling a problem

Before tackling a problem you should be

* Ready/Relaxed
* Active as if you are in a contest
* Challenge Sprit
* Don’t rush / Don’t panic
* Check your watch – Monitor your progress

We could think in Solving a problem as stages

1. Reading and correctly “understanding” problem statement
2. “Thinking” of a solution -> verifying it
3. Coding
4. Debugging if necessary
5. Testing

Hence your overall performance depends on performance in every stage. The more better in one of the stages, the better overall performance. Then, your practice plans must care with all the stages. Always measure time you do for every phase, this will help you know your problems. The training must be varied and you should focus on your weaknesses. Another dimension for performace is your speed.

# Reading problem statement

Although we read it fast, it is one of popular reason for failure!

The best way to avoid that is to be organized – to have a DISIPLINE during reading: a systematic way of reading

1

* 1. Read the problem slowly and think in each statement.
  2. Make sure every statement does not conflict what you overall understood
  3. Re-Think in a statement, If it seems a tricky statement
  4. Number All important details

2

* 1. In some Problems, constraints are clear, but somethimes they are not.
  2. Each time you read a constraint in mid of description , write it down
  3. Never to avoid any constraints especially unusual one. Try to know why such constraints
  4. Sometimes constraints make problem a special case iof a general one. While general may not be solvable, a specific one could be.
  5. Ignoring constrains may push you to approach problem trivially while it needs careful work (n<=10^18)
  6. Ignoring constrains may push you to approach problem complicatedly while it could be solved trivially (n<10)
  7. Sometimes more input space analysis is needed: Given a string of (a,b,c) chars & length <= n -> we hacve 3^n possible string

3

3.1. Trace samples as long as they are traceable

\* Many times students write solutions and find samples doesn’t work. They have to debug

\* Sometimes they have code mistakes and original idea is correct / orginal idea has flaws / original idea is incorrect.

3.2. Sometimes samples are trivial and mislead you.

1. If Test is not small, Re-read the problem statement once. Make sure you have the full picture.
2. Think in missed cases. Most of times authors don’t put all basic cases. Think in them.
3. Think in boundary