Solve Me First

by shashank21j

Editorial by shashank21j

You can look at summing two numbers as a very basic operation available in all programming languages. The solution is prewritten in the template.

General Note on Competitive Programming

Competitive programming should be taken up like any other sport. It's highly intimidating, and it's very easy to be overwhelmed. One should understand when to look at the editorial, when to ask for hints and, lastly, when to look at other's code.

The general sequence of steps is shown below:

1. Examine the challenge carefully. Try to guess which approach should be used.
2. Try solving the challenge using that approach. If you pass a few test cases, that means you are headed in the right direction.
3. If some test cases are giving the wrong answer, download them and try them in your local or HR platform to see what the error is. Be aware that a test case being wrong is highly unlikely since many hackers have already solved the problems. Blaming the challenge is the wrong attitude to have and doesn't help you to learn.
4. If you are completely clueless on how things are working, unlock the editorial and look at the solution. (DO NOT LOOK AT THE CODE)
5. Now, build your own solution and, when you get accepted, look at the code given to see how much better/worse you were compared to the setter/tester.
6. Lastly, a sport should be respected. Any attempt at cheating is harming only you. It's like any other sport: football, swimming, etc. To get better at it, you have to work hard by yourself, for yourself, and win.

How to start with competitive programming

Competitive programming is one of the most difficult sports and can be extremely overwhelming. It'll compel you to give up or look for an easy way out by checking for solutions on the internet that solve the challenge without programming the solution for yourself to see how the algorithm is working.

There are two aspects to your learning.

1. Participating in active contests (Online).
2. Learning newer and newer approaches (Offline).

Participating in active contests

By participating, you are putting your skills to the test. Your goals should be limited to coding your solutions as fast and as accurately as possible. Do not expect to discover a mind blowing approach to a difficult problem because, chances are, you might have never learned the algorithm. Great algorithms were not designed overnight, nor did anyone solve a path breaking research problem in a contest.

Once you have participated, read the editorials to the problems that you thought you could have done but missed due to some error. Try to follow a select few great programmers and look at

Statistics

Difficulty: Easy
Time Complexity: O(1)
Required Knowledge: Sum of two numbers
Publish Date: Sep 19 2014

This is a Practice Challenge

their code for problems you have already solved. See if their code is more elegant than yours. Many times, good programmers can solve with a better/faster solution if such a solution exists.

Offline Training

Find yourself a good team or a partner to practice with and, if possible, a mentor to watch your back when you get extremely frustrated and stuck. (Write a message to me if you need help in this.)

Learn a topic either by solving a random problem that you have no idea about or picking up a standard topic from any programming guide.

A general sequence of learning concepts is:

1. Solving implementation and ad-hoc problems that just require a good grasp on your programming language and simple elegant code.
2. Learn Data Structures. Get yourself familiar with Stacks, Queue, Arrays, Dictionaries, etc., and start using these structures to solve more implementation problems.
3. Learn: Programming paradigms, Search algorithms, Greedy algorithms, Dynamic programming, and Divide and Conquer. By this time, you should be comfortable with terms like: BFS, DFS, backtracking, recursion, Top Down DP, Bottom up DP, Memorization, Sorting, Segment Trees, Red-Black trees, Binary Search Tree, AVL Tree, etc.
4. After completing the previous steps, you are good to go for all general programming and can call yourself a beginner. Now, you can look at specific algorithms in Strings, Graph Theory, Mathematics, Game Theory, other advanced algorithms and data structures.

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature