A survey on online competitive coding and skill enhancement platforms

N. SreeRam

Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, India, sriramnimmagadda@gmail.com

Abstract.

Competitive programming, or competitive coding, is a kind of mental sport in which participants must solve certain algorithmic and computational challenges in a predetermined amount of time. Students studying computer science, software engineers, and coding aficionados all like doing this. The purpose of competitive coding contests and platforms is to evaluate competitors' ability to solve problems, understand algorithms, and write code. Software developers, programmers, and anybody else working in the IT sector should focus on improving their coding skills since technology is changing quickly and remaining current is critical. Numerous platforms are available to do competitive coding and skill enhancement such as hackerank, codechef, Leet code, code forces. These websites frequently hold coding contests where users may compete with one another and show their coding prowess. These platforms have its own communities, challenges, and competitions. We may choose the platform that best fits our interests and ability level, whether we are a novice trying to get better at coding or an accomplished competitive programmer A survey of the literature on many online competitive coding platforms is presented in this work.

Keywords: Competitive coding, skill enhancement, computational challenges, coding aficionados.

1. Introduction

Competitive programming, or competitive coding [1], is a mental sport or leisure activity that entails completing well-defined algorithmic and computational challenges in a predetermined amount of time. Students studying computer science, software engineers, and coding aficionados all adore it. The main objectives of competitive coding are to develop algorithmic knowledge, problem-solving abilities, and effective coding techniques. Competitive coding is beneficial to people for a variety of reasons, but it's especially crucial for those in the computer science and software development fields. These are some of the main justifications for the significance of competitive coding such as enhances problem solving skills, improves algorithmic knowledge, efficiency and optimization, and coding proficiency. There are a few requirements and abilities that you should acquire or hone in order to succeed in competitive coding like knowledge on programming languages, Data structures, algorithms, Mathematics, problem solving skills, and time complexity analysis. Skill enhancement in coding is crucial if you want to stay up to date with technology, progress professionally, produce excellent work, and support your own sense of accomplishment. It is a journey that lasts a lifetime and has several rewards. Enhancing your coding skills is a continuous process that requires dedication, practice, and a structured approach. Keep in mind that improving coding abilities is a process that requires patience and time. Challenges and errors shouldn't deter us; they are important chances for development and learning. To become a more skilled and adaptable developer, have a growth mentality and keep pushing the envelope. Test cases are a crucial component of competitive coding since they allow you to check for accuracy and efficiency, debug code, optimize performance, and validate correctness. Developing good test case creation and usage abilities is crucial for competitive programming success. The main challenge in competitive coding is to satisfy given constraints by solving various test cases.

Competitive coding problems must be solved methodically in order to effectively address algorithmic and coding difficulties. Here are some criteria and steps to consider when solving problems in competitive coding: The problem statement must be well understood as the first stage. Make sure you understand the specifications, limitations, and expected results by thoroughly reading it. Ask the platform community or the competition organizers for clarification if there are any questions. Sort the issue into groups according to its attributes. Is it a dynamic programming problem, a graph problem, a sorting problem, or another kind? Determining the category will help you choose the right algorithm and strategy. Think over your answer before coding. Choose the algorithm or data structures that you want to employ. Think about input restrictions, edge cases, and the general organization of your code. Put your answer in high-level pseudocode. This helps you understand the logic of the problem before you implement it and acts as a template for your real code. Begin coding your solution using the pseudocode as a guide. Observe the algorithms and data structures that have been selected. Write code that is clear, organized, and simple to read. Use a range of test cases, such as the sample inputs given in the issue statement, to test your code. Make that your solution works within the time and memory limits and generates the right results. Once the problem has been resolved, search for ways to improve your approach. Think of methods to simplify space and time. Efficient code is typically rewarded in competitive programming. Make sure your solution handles boundary conditions and edge circumstances. To ensure its resilience, test it with both lowest and maximum input values. If your code doesn't pass all test cases, use debugging techniques to identify and fix errors. Trace your code's execution, print debug statements, and examine intermediate results to pinpoint issues. Examine the intricacy of your solution in terms of space and time. Make sure it satisfies the limitations of the issue. Large-scale input handling and efficient solutions are frequently needed for competitive programming. Competitive coding is a skill that improves with consistent practice. Regularly participate in coding contests, solve problems on online platforms, and tackle a variety of challenges to sharpen your skills. Recall that competitive coding requires not just fast issue solution but also accurate and effective problem solving. A rigorous and systematic approach to problem-solving is essential for competitive coding success.

Developing strong test cases is crucial in competitive coding to guarantee the accuracy and effectiveness of your solutions. A variety of scenarios, such as edge cases, corner cases, and standard instances, should be covered by test cases. These test case types are frequently utilized in competitive coding such as sample testcases, boundary testcases, edge testcases, random testcases, negative testcases, worst testcases, adversarial testcases and stress testcases. Aim for a varied collection of test cases when constructing them so that they may fully assess all facets of your solution. This helps guarantee that your code is reliable, efficient, and accurate in a range of situations. Examining diverse test cases from various competitive coding platforms will help us become more proficient in code and provide us with a broader understanding of problem-solving methods. The difficulties that come with each platform could be different, and becoming knowledgeable about the nuances of many issues will help you become a more adaptable coder. This paper provides an overview on variety of test cases in different competitive coding platforms

2. Leet code

LeetCode is a popular online platform for practicing coding problems, preparing for technical interviews, and enhancing programming skills. The platform provides a large selection of programming contest [2] covering a wide range of subjects, including databases, algorithms, and data structures. Anyone can learn both basic and complex data structures and algorithms on LeetCode through tutorials and study plans, participate in competitions, utilize guidelines to be ready for top organizations, practice problem-solving by answering actual interview questions, and much more. The following tips can help us to work with LeetCode platform

i. Explore problems from different categories, including Arrays, Strings, Linked Lists, Trees, Dynamic Programming, Sorting, Searching, and more

ii. Start with problems of lower difficulty and gradually progress to more challenging ones. LeetCode categorizes problems as Easy, Medium, and Hard

iii. After solving a problem or if you're stuck, check out the discussions for that problem. You can find various solutions and insights shared by the community

iv. Practice time management, especially if you're preparing for coding interviews. LeetCode often provides time and space complexity information for solutions

v. After attempting a problem, read the editorial or official solution. Understanding different approaches is crucial for improving your problem-solving skills

vi. Participate in LeetCode Weekly Contests to simulate real interview conditions and challenge yourself with time-constrained problem-solving

vii. LeetCode allows you to run custom test cases to further test and debug your solutions. Take advantage of this feature to validate your code

viii. LeetCode allows you to track your progress and provides a history of your submissions. Use this feature to monitor your improvement over time

Remember, the goal is not just to solve problems but to understand the underlying concepts and patterns. LeetCode is a fantastic resource for honing your coding skills and preparing for technical interviews.

In LeetCode, test cases are sets of inputs and corresponding expected outputs that are used to evaluate the correctness and efficiency of your code. When you submit your solution to a problem, LeetCode runs it against multiple test cases to ensure it produces the correct results. Here's a general overview of how test cases work on LeetCode

i. Example Test Case: LeetCode typically provides one or more example test cases in the problem description. These examples help you understand the expected input and output format.

ii. Hidden Test Cases: Apart from the example test cases, LeetCode includes additional hidden test cases. These are not visible to you, and your code is evaluated against them when you submit.

iii. Custom Test Cases: LeetCode allows you to run custom test cases on your code before submission. This feature enables you to test your implementation with specific inputs and verify that it produces the expected outputs.

iv. Edge Cases: LeetCode test cases often include edge cases to ensure that your solution handles extreme or boundary scenarios correctly. Consider cases with the minimum or maximum input values specified in the problem constraints.

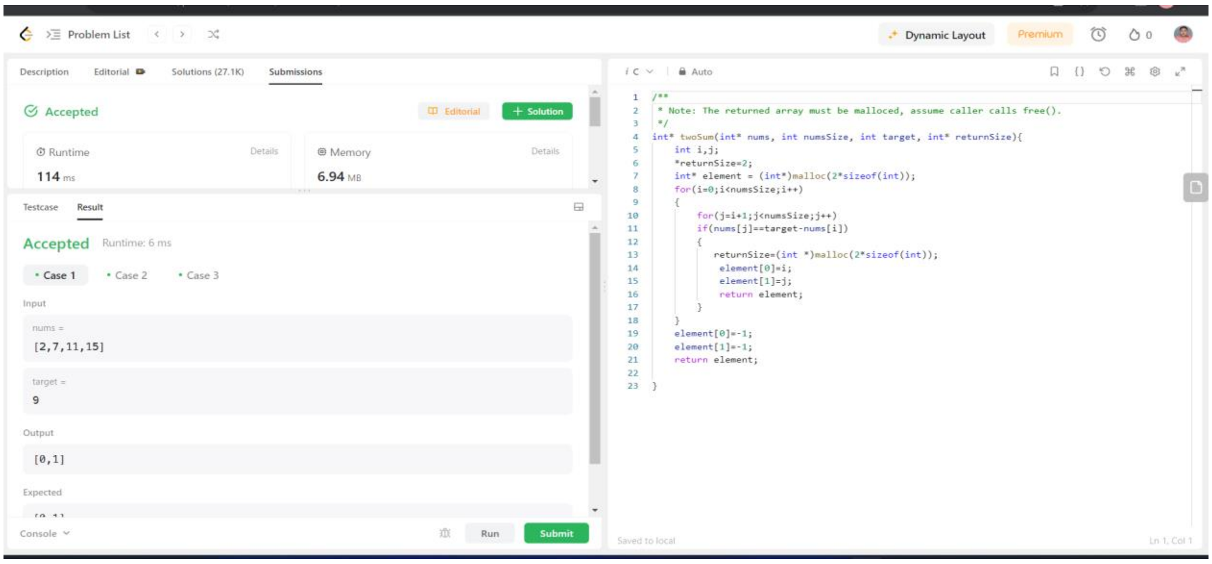
v. Performance Testing: In addition to correctness, LeetCode evaluates the efficiency of your code. Some problems may have large input sizes, and your solution needs to run within the specified time and space constraints.

vi. Randomized Testing: For certain problems, LeetCode might include randomized test cases to ensure that your solution is not just hard-coded for specific inputs.

vii. Multiple Submissions: LeetCode allows multiple submissions, and you can iterate on your code based on the feedback from test cases. If your initial submission fails some test cases, you can make improvements and submit again.

viii. Discussion Test Cases: In the discussion section of a problem, you may find additional test cases shared by other users. Exploring these can be beneficial to further test and refine your solution.

When solving problems on LeetCode, it's crucial to thoroughly test your code with a variety of cases, including those provided in the problem description, custom cases, and any additional edge cases you can think of. Understanding the expected output for different inputs is key to successfully solving problems and passing the test cases on LeetCode. The following figure demonstrates that how submitted code evaluated against testcases.



```
nums =
[2,7,11,15]
target =
9
```

```
[0,1]
```

```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* twoSum(int* nums, int numsSize, int target, int* returnSize){
    int i,j;
    *returnSize=2;
    int* element = (int*)malloc(2*sizeof(int));
    for(i=0;i<numsSize;i++)
    {
        for(j=i+1;j<numsSize;j++)
        if(nums[j]==target-nums[i])
        {
           returnSize=(int *)malloc(2*sizeof(int));
            element[0]=i;
            element[1]=j;
            return element;
        }
    }
    element[0]=-1;
    element[1]=-1;
    return element;

}
```

Figure 1 Test cases evaluation for submitted solution in LeetCode

3. Codeforces

Codeforces is another popular online competitive programming platform that hosts regular contests and provides a platform for users to practice and improve their algorithmic and coding skills. The contests on Codeforces involve solving a set of challenging problems within a time limit. The user can perform the following activities in codeforces platform.

i. Codeforces hosts a plethora of problems ranging in difficulty. It's a great place to sharpen your problem-solving skills by tackling algorithmic challenges

ii. Regular contests are held, providing a competitive environment similar to real-world coding competitions. Participating in these contests can help you improve your speed and accuracy under time constraints

iii. Codeforces has a rating system that reflects your skill level. As you solve more problems and participate in contests, your rating changes, giving you a sense of progression and helping you gauge your improvement

iv. The platform has an active community where users can discuss problems, share solutions, and learn from each other. Engaging in discussions can broaden your understanding of different approaches to problem-solving

v. Codeforces occasionally hosts educational rounds, where the emphasis is on learning and understanding the concepts rather than competition. These rounds often have editorial discussions that can be valuable for learning

In Codeforces, test cases are usually grouped into several categories, and each category represents a different aspect of the problem that your solution needs to handle correctly.

i. Sample Test Cases: These are the cases that are provided in the problem statement. They are meant to help you understand the input/output format and may cover some basic scenarios. Make sure your solution passes these cases before moving on.

ii. Boundary Test Cases: These test cases are designed to check if your code handles extreme or edge conditions correctly. For example, the minimum or maximum allowed values for input variables.

iii. Normal Test Cases: These are typical cases that test the main logic of your solution. They cover a range of scenarios and are crucial for ensuring your code works correctly in various situations.

iv. Special Test Cases: Some problems have special conditions mentioned in the problem statement. Test cases related to these conditions are included to verify that your solution adheres to the specified constraints.

v. Random Test Cases: These are additional test cases generated randomly to further verify the correctness and efficiency of your solution. They may include scenarios that are not explicitly mentioned in the problem statement.

4. Codechef

One of the most well-liked platforms for programmers to practice and solve a wide range of issues from many fields, such computer programming and algorithms, is Codechef [12]. Programmers may also use it to enter contests. The programmer needs to write to and read from standard output. Following solution submission, the Codechef IDE will display the following outcomes

I. Accepted: If code ran successfully and gave expected output. If there is a score for the problem, this will be displayed in parenthesis next to the checkmark.

II. Time Limit Exceeded: If code was compiled success- fully, but it didn't stop before time limit. Try optimizing solution approach.

III. Wrong answer: If code compiled and ran successfully but the output did not match with expected output. Then there is need to change solution approach.

IV. Runtime Error: If code compiled and ran but encountered an error. The most common reasons are using too much memory or dividing by zero. Then there is need to change conditional statements in the code.

V. Compilation Error: If there are syntactical mistakes in code, it will return compilation error. Then there is need to check and modify the syntax of the statements.

5. Conclusions

While competitive coding is a fantastic learning tool, it's not the only way to become a proficient programmer. Real-world projects, collaborative coding, and understanding how to write maintainable code are also crucial aspects of a well-rounded developer. In conclusion, competitive coding is a valuable addition to your skill set, providing a structured and competitive environment for learning and honing your coding skills. Because these kinds of competitive coding platforms are available, users may utilise them as software as services instead of expressly purchasing and installing various programming language compilers.

References

1. Parth Dave, Rushikesh Deshmukh, Dipak Kumar Gautam "Competitive Coding Website" International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 10 Issue 04, April-2019
2. S. S. Skiena and M. A. Revilla. Programming challenges: The programming contest training manual. Springer Science Business Media, 2003
3. https://www.sas.com/content/dam/SAS/sv_se/doc/Presentation/SAS-Forum-SAS-in-Cloud.pdf
4. Hyunmin Seo, Caitlin sadowski, Sebastain Elbaum, Robert Bowdidge "Programmers' Build Errors: A Case Study (at Google) ICSE '14, May 31 - June 7, 2014, Hyderabad, India ACM 978-1- 4503-2756-5/14/06

5. Judy C.R.Tseng, Chih-Hsiang WU “An expert system approach to improving stability and reliability of web service Expert Systems with Applications “ Volume 33, Issue 2, August 2007, Pages 379- 388
6. https://en.wikipedia.org/wiki/Test_case
7. https://technologyconversations.com/2013/12/20/test-driven- development-tdd-example-walkthrough/
8. https://www.codechef.com/problems/school
9. Reema Thareja “Introduction to C programming” Second Edition university of Delhi, OXFORD