Optimization and Improvements of a Moodle-Based Online Learning System for C Programming

Xiaohong Su, Jing Qiu, Tiantian Wang, and Lingling Zhao
School of Computer Science and Technology
Harbin Institute of Technology, Harbin 150001, China
Email: sxh@hit.edu.cn, topmint@hit.edu.cn, sweetwtt@126.com, zhaolinglinghit@126.com

***Abstract*—It is important for students to solve problems with specific requirements in the programming teaching. Our teaching system is a Moodle-based interactive teaching platform for C programming. Its online judging system can grade students code automatically. It plays an extremely important role in programming language teaching. This paper is devoted to optimizing and improving the system. We firstly analyze the five problems in the system according to the feedback from the teachers and students: 1) logical errors in students programs cannot be located; 2) a cheating that directly outputs answers cannot be detected; 3) evaluation results lack statistics and visualization; 4) the code submitting procedure is complicated; 5) the feedback of incorrect answers is not detailed. In order to solve these problems, we employ a fault localization algorithm, revise the evaluation logic, introduce third-party visualization plug-ins and refactor the system, respectively. Detailed and exact solutions are given also. After optimizing and improving the system, the user experience is significantly improved. It is convenient for the student to find and correct errors in their programs. Also, it is easier for teachers to acquire valuable feedback and master students' learning situation.**

Keywords*—Online judge system, test cases, software fault localization, visualization.

I. Introduction

In the face of the information technology revolution, it is necessary to change the traditional education teaching mode. Information technology brings new vigor to education teaching mode and more demand to the education teaching level. Programming language is a basic course in most universities. At the same time, programming is the important index to evaluate the student IT application ability for many institutions and organizations. As a consequence, it is really important for students to practice and master some programming ability.

At present, the main way of improving learners' programming ability is acquired by the act of constant practice. In the practice, accurate and instant feedback is very important for teachers to help students to correct mistakes. In the traditional education teaching mode, manual marking is the main method to score a students program. A teacher gives corrections through reading code on a paper written by the student. This way of evaluation program requires a lot of manpower and material resources, and the efficiency is very low. At the same time, it is difficult for students to obtain accurate and objective feedback timely. Thus, the following questions often haunt education workers for a long time [1]: How to reduce the workload of the program judge? How to improve the existing program judge mode? How to solve the large demand for the program judge for students?

Fortunately, with the rapid development of information technology, computer application technology has gradually penetrated into all areas of our life [2]. In recent years, along with in-depth study of the examination system, the program online judge (OJ) system has been gradually implemented to judge various programming languages automatically. The program OJ system originated in a world-class programming ability contest – ACM/ICPC. This competition is organized by the world's largest computer society ACM once a year. Now each year, there are thousands of universities and students around the world participating in this competition. ACM/ICPC uses OJ-PC^2 as its OJ, but this system is designed specifically for the game, not suitable for everyday practice. Therefore, in order to train players, many programs OJ systems have been developed, and provide services to the world.

The basic form of the program OJ system is to provide a huge problem set. The participants search for interesting topics, and try to solve a problem. Once participants submit source code to the system, the results will be given immediately by the system. For students, introducing OJ system in programming courses as secondary education can provide a platform for self-learning, create more opportunities to practice programming, and enhance the programming ability of students.

Program OJ system fully demonstrates the power of computer aided teaching. It can greatly promote the changing of education model. In practice, in programming languages teaching courses, by using OJ systems, teachers can provide remote guidance, arrange quiz exercises, schedule after-school assignments, and arrange online examinations, etc. Compared with the traditional programming language teaching, program OJ systems mainly have the following advantages:

1) It provides all day online judge system. Students can finish an assignment anywhere, and arrange learning progress independently. It provides a good environment for students to improve self-programming ability.
2) It trains students the rigor of programming by using a rigorous test cases for judging the submitted programs.
3) A teacher can view the program submitted by students, easily know students' progress, as well as mastery of knowledge.
4) The teacher does not need to spend a lot of time and effort to complete the evaluation of the follow-up program for students, which greatly saves manpower and material costs, such that teachers have more energy to focus on teaching itself.

5) Its automatic evaluation logic allows students to quickly and accurately obtain feedback, and ensures the objectivity and accuracy of the feedback.

Program OJ system provides a new model to improve students' programming ability, and a perfect interactive mode of teaching and learning for education reform. The main feature of the OJ system is that the efficiency is very high. To a certain extent, it helps to stimulate students' enthusiasm for programming. OJ system allows students and teachers to better adapt to the rapid development of the information age, and plays a role in promoting education in the field of today's computer and other industries.

The rest of this paper is organized as follows. Section II describes related works. Section III gives a brief introduction of Moodle and our moodle based system. Section IV describes our work. Section V gives the evaluation result. Finally Section VI states the summary.

II. Related Work

ACM/ICPC is the world's largest, highest level of Collegiate Programming events, with high authority in the international computer industry, which aims to show students' innovative ability, analytical skills and problem solving skills of teamwork [3]. During the contest, all teams need to solve problems within a limited time and memory space, and submit programs to the OJ system. The judge will determine whether a program is passed according to the running result of the OJ system.

OJ system is a contest system that can automatically compile, run, and score the submitted code. It is mainly used as a contest platform early. Along with the teaching model reform brought by information technology, many OJ systems are developed for curriculum development and design.

In addition to the basic functions of a common OJ system, such systems also added many assistant teaching modules, which fully mobilize the interaction between teachers and students, and promote the development of education and teaching model. As in OJ system of Peking University, assignment correction and online real-time tests of "Data Structure", "Introduction to Computing","Problem Solving and Program Design", "Programming Practice" and many other courses can be done on this system [4].

The first online learning system appeared in the 1996. Now, after many research and practice, the technology of OJ system becomes more and more mature. Throughout the various OJ systems, they all have the basic functions such as users registering, question set management, online testing, and automatic scoring, etc. Although these systems are still inadequate in some respects, they basically meet the demand of programming language curriculum for students after the actual use. Teachers will be freed from the heavy marking work, and they can take more time on teaching, which greatly promoted the reform of education and teaching mode.

The PC^2 (PC Two) of University of California is the world's most typical and influential OJ system. It is designed to support ACM/ICPC Programming Contest. In addition to automatic compiling and executing the submitted programs and returning the result, PC^2 can archive the submitted programs intelligently, display of the process of the contest in real-time. It also allows contestants to query request to judgers. The judgers at the other end of the system can give an appropriate answer to contestants through the system. Contest information can be transferred between peers, such that scores between peers are updated automatically.

The OJ system (acm.sgu.ru) of Russia's Saratov State University is a world-renowned OJ site. Although it has only over five hundred exam questions, each question is very refined. There are two million registered users. The site breeds a very famous web site in 2010: Code-forces. Its running mode is different with ACM contests. If a user submitted a program, he can lock his code, and view and test other people's code. In this way, the final test data can be collected. Thus, the system can give a more rigorous test to a problem. It makes the scoring more accurate and gives the problem writers more convenience.

The OJ system Spoj (spoj.com) of Poland Gdansk University of Technology was established in 2004. It is characterized by large scale: it supports up to 56 kinds of programming languages; it has as many as 300,000 total number of registered users; it is used in more than 4,000 worldwide organizations; and it has more than 5,000 open problems. Registered users can write problems. More than 500 problems were contributed by registered users.

III. Background

A. Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) is an open source course management system, and it is published by an Australian teacher Dr. Martin Dougiamas based on constructivist theory in 2002 [5]. The so-called constructivist theory can be summed up as follows. Educators/teachers and learners/students are equal subjects in the teaching activities, and they work together to build knowledge based on their existing experience [6].

Moodle uses B/S architecture and is written by PHP programming language. As a result, it is very portable. Moodle is a highly modular software, and using common technologies, such as shared libraries, database abstraction layers and cascading styles to define the structure. The application of this series technology makes Moodle greatly facilitate users to customize the system. Moodle has the functions that virtually all network learning platforms have, such as assignments, quizzes, forums, discussion boards, news releasing, and content management, etc. As a prestigious free open source course management system, Moodle has been widely used in schools, educational institutions and even business training [7].

B. Our Moodle based Online Judge System

Our online judge system uses a results-oriented judge method. The system compares the output of the program to be scored with the expected content. The program is correct if its output matches the correct one, otherwise it is incorrect.

When teachers arrange online assignments, in addition to set the assignments requirements, only test cases are set, can students answer the assignments. A test case consists of

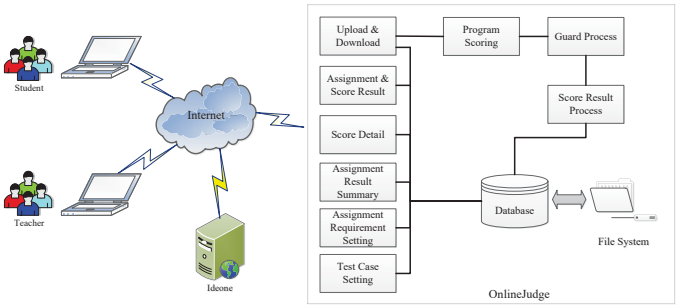

Fig. 1. Our Moodle based online judge system

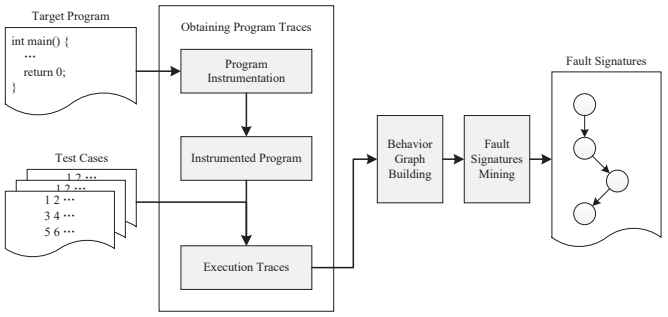

Fig. 2. Basic procedure of the mining weighted software behavior graph based fault localization algorithm

the input (which will input to students' program) and the corresponding correct output.

When students submit the program, the system will automatically load test cases, and then execute the program once for each set test case and get the output. Then the system compares the output with the correct one and computes the score with the comparison result.

For a programming problem, the ultimate goal is to achieve a certain function. Due to the diversity of the grammatical structures, students can use a variety of ways to achieve this function, resulting in that the correct answer of the problem is not unique. The system that using a results-oriented judge methods meets the needs of the diversity of answers, coupled with restrictions on the program's performance, ensuring the correctness, fairness and objectivity of the judge.

The architecture of our system is shown in Figure 1. Students and teachers interact over a network, completing the appropriate actions at any time and any place.

IV. Our Work

A. Fault Localization

Fault localization can find out the bugs of a program. It is especially useful for beginners.

For submitted programs, the system only feeds back general and brief information, such as runtime overtime, compilation error, incorrect answer, etc. Only when compilation errors occur, will the system give their locations and types. However, the system cannot provide the location of the logical errors but feed back "incorrect answer". With the help of the information, a student knows nothing about the problem in his program. He often inspects the program line by line, audits the algorithm, and considers boundary values, etc. In most cases, this solution only works for a simple program. For a problem with a certain degree of difficulty, the fault is hard to be located. That frustrates students, brings fears of the programming to students, and makes them stop answering the question.

Existing fault localization algorithms only result the sorted sequences of equivocation. For understanding the logical errors, the result needs the context of the related statements of the error. Hsu presented a signature based fault localization algorithm perfectly solves the issue [8]. His algorithm outputs not only the locations of fault statements, but also the related contextual information of these statements. Based on Hsu's algorithm, Cheng introduces a graph mining based algorithm, which mining the information stored in the signatures and greatly improves the efficiency and precision.

Shaojun Yang presented an approach based on mining weighted software behavior graph, further extending Cheng et al's approach and obtaining favorable results [9]. This algorithm generates the weighted behavior graphs of the successful and unsuccessful executions by comparing the output of an instrumented program with that of the correct program. Then it mines the differences between these two kinds of graph to determine the fault statements.

Fig. 2 shows the basic procedure of the algorithm. First, the target program is instrumented (i.e., a special statement is inserted after each statement which indicates the location of the statement, such that the executed trace of the program can be obtained). Second, the instrumented program is compiled and executes test cases, resulting corresponding sets of output and execution traces. Execution traces can be classified into successfully executed and unsuccessfully executed traces according to the output and the expected output of the program. Third, weighted behavior graphs are constructed, resulting successfully executed and unsuccessfully executed graphs. Finally, the fault signatures are obtained by identifying the maximum different subgraph in the successfully and unsuccessfully graphs.

This algorithm applies to large scale programs and requires many test cases. But in the OJ system, the effective amount of a student's program is about 20 lines and the actual demand requires the fault localization algorithm to have higher efficiency.

In the testing process, we found that the time of the localization algorithm is mainly spent on executing a large number of test cases, and for small-scale programs with five or six test cases, the system can also give a good result. Thus, the fault localization algorithm can directly use the test cases in our OJ system.

In the actual demand, the result of fault localization is only a reference for a student to fix bugs, and its precision is allowed to be low. Meanwhile, in order to save the overhead of the system, the fault localization is performed when a student's code is successfuly compiled. The fault location algorithm is test-case-driven and is not feasible when there are few test cases (e.g., only one or two). Thus, the algorithm is not performed in this situation.

B. Directly Outputting Detection

Existing OJ system uses a rough method to judge a student's code: compiles code first, runs it with test cases, then compares its running output with the output of test cases, calculate the score finally. Thus, as long as the output matches with the output of test cases, the code is regarded as passed the test, no matter how the logic of the code is implemented. Such testing can deal with various solutions of one problem. However, a student can directly forge the output when he knows the test cases that the online judge system uses.

For example, a programming problem is to solve an equation that has only one solution. The motivation of the problem is to examine the mastery of the loop structure of students. But as the equation has only one answer, a student can calculate the answer manually, and outputs the answer with the library function "printf()". According to the scoring principle of the OJ system, the student's code is correct because its output matches with the correct one. But it is not the expectations of teachers.

We have developed a new method to deal with this case. This could be done in the following steps.

First, the code to be checked is compiled to assembly code with compiler optimization. This step tries to use an existing compiler to perform optimization such as constant folding, loop unrolling, and pointer analysis, etc. After compiler optimization, most tricks for bypassing detection could be removed. A simple example is shown below. A student knows the answer, and he uses some computations on the answer instead of directly outputting the answer. Without compiler optimization, the judge system will be cheated.

```
a = 12;
b = a + 1;
printf("%d", b);
```

a) Original program

```
printf("%d", 13);
```

b) After optimization

Second, all library calls for outputting result are scanned in the assembly file. Parameters of each call are checked whether they are all constant values. Because the analysis is performed on assembly code, instructions that translated from invoking "printf" may not be continues. As a result, it is essential to perform control and data flow analysis on assembly code. Fortunately, the analysis is quit simpler than that on the C source code.

The control flow analysis consists of two steps. First, basic blocks [10] are partitioned. Then, the control dependence among basic blocks is analyzed. Data flow analysis is simple here. Only the usage of the stack is needed to be tracked. That is because the library function "printf()" uses the stack to pass its parameters. In the control flow graph G, for each instruction I that calls "printf", a backward slice for the stack pointer register is performed and a subgraph G_I is obtained.

In G_I, the first predecessor of I (mark it as I') is checked first. If the operand of I' is a format string, the information of the parameters of the call is extracted. If the operand of I' is a constant value, or a constant string without any format specifiers, the call is marked as a suspect cheat call. Once the information of the parameters is successfully extracted, they are checked whether they are all constant values. If so, the call is marked as a suspect cheat call. A program is marked as suspect cheat program if and only if all library calls for outputting result are suspect cheat calls.

Finally, the similarities among the code and the correct answers are computed. It is nearly impossible to avoid false positives. If a student's code has a loop with less than 100 iterations, or has simple computations, it could be a false positive. To filter out these false positives, the suspect code is compared with the correct answers in the OJ system. If the suspect code has low similarities to these correct answers, it is marked as a cheating.

C. Evaluation Visualization

For judge results, our system only lists simple information. Teachers are hard to know the overall status of answering questions, weaknesses and other information of students. If teachers can have statistical and visualized feedback, and from which digging out potential information that teachers need, then teachers can pay more attention to analyzing the typical problems that most students fail to pass, and intensively training students for the weak knowledge points. It is undeniable that these feedbacks will be better service to the teaching. Thus, it is essential to perform a reasonable statistical analysis on evaluation results (e.g. pass rate statistics on each test case of a problem, overall pass rate statistics on each problem).

In order to visualize evaluation results, statistical charts are added to the system to represent the statistics. If we manually implement this function in the system, no doubt it will take a lot of energy. The performance, user experience, and scalability are also very hard to guarantee. Therefore, the third-party component - Baidu Echarts [11] is used, which has mature technology and is extensively used by many applications.

In the existing system, only a simple set of students' assignments is shown and its related statistical information cannot be shown clearly and directly. Therefore, we add statistical visualization to the existing system. The user interface is shown in Fig. 3.

The features of the design are as follows.

1) The system can show the statistical graph for the pass rate of a problem. It helps teachers to know the pass rate of each test case and the common mistakes of students.
2) The system can show the statistical graph for the pass rate of all problems of a course. In order to enhance the comparability, the numbers of students that answer and pass a problem are listed in a graph. The difficult of a problem can be known from the graph. It helps teachers arrange the content of a course.
3) The system can display the pass rate of a problem to students.

D. Submitting Procedure Simplification

In the existing system, during solving a problem, a student needs to save the answer in a file, and then submits the file to the server for judging via the upload module. The detailed steps are shown in Fig. 4. There are six click actions

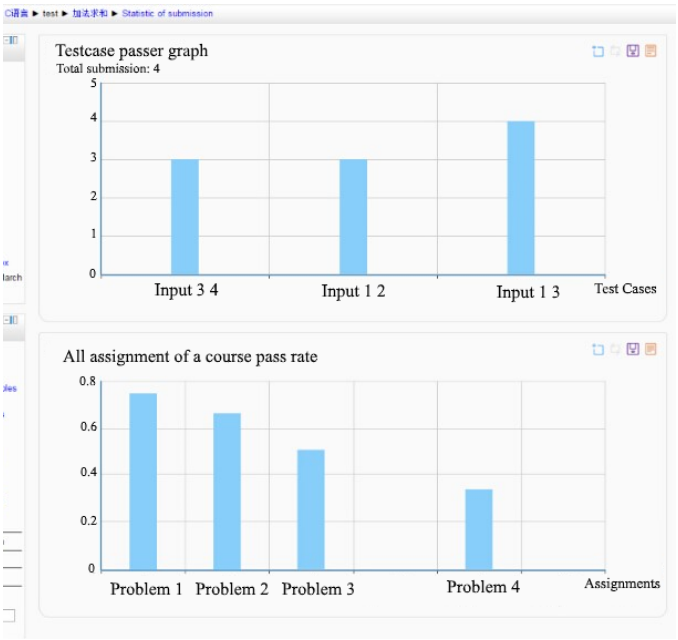

Fig. 3. The page for showing statistical result

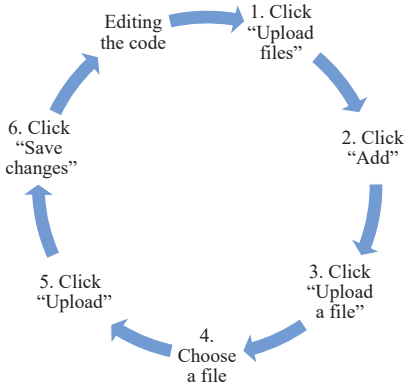

Fig. 4. Steps for submitting code in current system

for a student submitting a program if the steps for saving and choosing files are not considered. When a student trying to solve a hard or error-prone problem, he needs to submit and judge one more times. The tedious steps will make him impatient undoubtedly and even that makes him give up. This will undoubtedly reduce the student's enthusiasm for learning.

Another problem is also often encountered in practice. In the submitting page, students can only view the name of the last submitted file but cannot view its specific content. When they upload a new file with the same name as the last submitted one, or upload a file whose changes are not saved after modifying the file, even the correct answer cannot pass the test of the judge system because of such careless, and any bugs in the algorithm or logic cannot be found. Such a completely avoidable problem also extremely disruptive the learning efficiency of students. Therefore, simplifying the submission operation and displaying contents of the submitted file can greatly improve the user experience.

In order to increase the convenience and transparency of the code uploading, the feature that uploading code by using

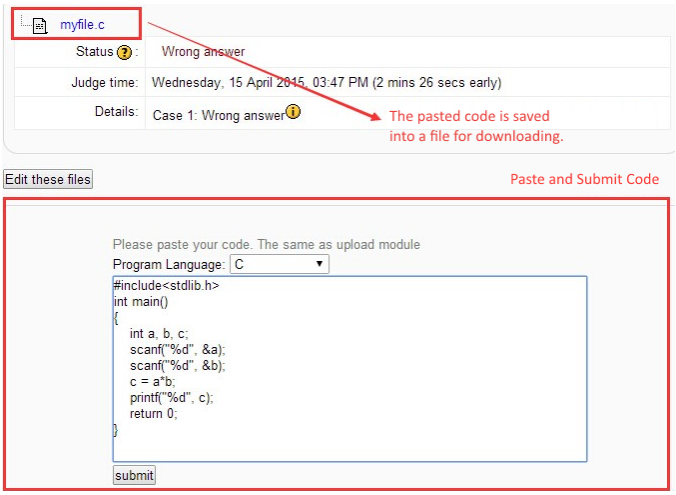

Fig. 5. Paste code

a text box is added into the system. In the submission page, a text box that used to paste code is added. For a submission, a student only needs to paste the code into the text box, then click the submit button. The implementation is shown in Fig. 5. When a user returns to the page for viewing the results, or answer the question again, the text box displays the contents of the last submitted code. That enables the students to check the last submission.

The entire process is equivalent to the process of uploading a file, downloading files and reading the contents. For practical needs, the system's original upload module (such as uploading multiple files or text files) should not be removed. Thus, the original and new upload module coexist.

E. Detail Compiler Feedback

In the real world, the following case is often occurred. The overall idea of a student's code is correct, the computation result of his code is correct also. But because the output format does not match the required one (e.g there are more or less white spaces or carriage returns, spelling mistakes), the output does not match the correct one. As a result, his code cannot pass the test cases of the system and "incorrect answer" is simply feeded back by the system. When such a case is encountered, a student will be confused and his first response is considering the reason of the algorithm, boundary values, or any other logical aspects. If repeated attempts still cannot solve the problem, a student wastes time while suffering a tremendous blow. Therefore, the actual demand requires the system to indicate what specific information is inconsistent with the desired output, helping students quickly rule out errors in the program, and locating their mistakes.

First, we improved the system by providing detailed feedbacks for different situations (Table I).

Second, we improved the system by computing and marking the differences between the expected output and the output of a student's program. An example is shown in Fig. 6. That helps a student find and fix bugs in his program, guiding him to pass the test.

TABLE I. FEEDBACK OF OUR ONLINE JUDGE SYSTEM

Judge Result	Message
Scoring	Your code is under scoring.
Compile Failed	Your code cannot be compiled. The detail is list below.
Correct	Congratulations! Your code passes all the test cases.
Partially Correct	Your code cannot pass some test cases.
Output Format Unmatched	The output format of your code is incorrect.
Run Timeout	The run time exceeds the limitation.
Memory Overrun	Your code ran out all the limited memory.
Exception	Your code exits with exceptions.

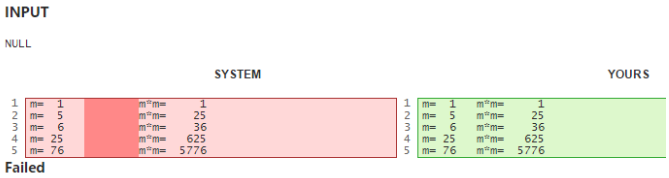

Fig. 6. An example for the diff result between the correct answer and a student's answer

V. EVALUATION

We have evaluated the improvement of fault localization and directly outputting detection on real world data. For other functions, because their effect already stated in the paper, we did not perform any additional evaluation for them.

A. Setup

The evaluation is performed on our OJ system hosted on a machine with 32GB internal memory, Win 2008 operation system, and Intel E7 CPU. The system has run several years. We pick up the data of the last semester for the evaluation. In the data, there are 423 students and 35,117 records.

B. Fault Localization

We only discuss three typical results here.

1) Example 1: The problem is to find the index of the element with the maximum value in an array (Fig. 7). It outputs the minimum index if two or more elements have the same maximum value. The test cases and the program are given in Table II.

The fault is located at the line 14: there should be a greater-than sign not a greater-than-equal sign. When the input array contains two or more maximum values, the program does not output the one with minimum index. The fault localization result is shown in Table III.

As shown in Table III, the result provides two suspect statement sequences. The three columns represent the rank, the possibility, and the context of a fault statement, respectively. The $\rightarrow$ denotes the jump of statements for understanding the information of a fault. For example, the first row in Table III represents that the fault occurs at "else if (v >= nums[maxId])" executed and "maxId=i" follows.

The feedback indicates the scope of the fault statements. A student only needs to follow the rankings of statements to check his code. This method greatly facilitates a student to correct errors in his program. For this example, students only need to check the first statement in the localization result to find bugs.

TABLE II. TEST CASES OF EXAMPLE 1

ID	Input	Output
1	20 8 13 19 17	0
2	16 12 0 19 20	4
3	14 5 13 6 7	0
4	8 13 14 18 18	3
5	13 11 3 18 14	3

```
1  #include <stdio.h>
2
3  void findmaxIdx(int n, int* nums)
4  {
5      int i = 0;
6      int maxId = -1;
7      while (i < n)
8      {
9          int v = nums[i];
10         if (maxId == -1) {
11             maxId = i;
12         }
13         /* Operator Changed */
14         else if (v >= nums[maxId])
15         {
16             maxId = i;
17         }
18         i++;
19     }
20     printf("%d", maxId);
21 }
23 int main()
24 {
25     int num[5];
26     scanf("%d", &num[0]);
27     scanf("%d", &num[1]);
28     scanf("%d", &num[2]);
29     scanf("%d", &num[3]);
30     scanf("%d", &num[4]);
31
32     findmaxIdx(5, num);
33     free(num);
34
35     return 0;
36 }
37
38
39
40
41
```

Fig. 7. Program of Example 1

2) Example 2: The code is shown in Fig. 8. The program exchanges the maximum and minimum values of an array. The main issue of the program is that the variable "maxPos" and "minPos" are not initialized before using. If in the input, the maximum one is the first element and the minimum one is the last, or the maximum one is the last one and the minimum is first one, the program will fail. During testing the program, we add a group of test cases of this kind.

In the evaluation, our system cannot locate the fault. After checking the output files of the system, we found that when executes the above kind of test cases, the program will crash. As a result, the trace cannot be obtained and our method will fail.

3) Example 3: This code (shown in Figure 9) is to determine whether a string is a palindrome. The bug in this program

TABLE III. FAULT LOCALIZATION RESULT OF EXAMPLE 1

ID	Rank	Statements
1	0.171	else if (v >= nums[maxId]) $\rightarrow$ maxId = i;
2	0.171	maxId = i; $\rightarrow$ i++;)

```
#include  <stdio.h>
#define ARR_SIZE 10
void  MaxMinExchang(int a[], int n)
{
    int  maxValue = a[0], minValue = a[0];
    int  maxPos, minPos;
    int  i, temp;
    for (i=0; i<n; i++)
    {
        if (a[i] > maxValue)
        {
        maxValue = a[i];
              maxPos = i;
        }
         if (a[i] < minValue)
        {
                minValue = a[i];
            minPos = i;
        }
    }
    temp = a[maxPos];
    a[maxPos] = a[minPos];
    a[minPos] = temp;
}

int main(void)
{
    int a[ARR_SIZE], i, n;
    printf("Input n(n<=10):");
    scanf("%d", &n) ;
    printf("Input %d Numbers:\n", n);
    for (i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    MaxMinExchang(a, n);
    printf("After MaxMinExchange:\n");
    for (i=0; i<n; i++)
    {
            printf("%d ", a[i]);
    }
    printf("\n");
    return 0;
}
```

Fig. 8. Program of Example 2

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#define N 10
int main()
{
    int len;
    char *pstr,*pend;
    char str[N];
    printf("Input string\n");
    scanf("%s",str);
    len=strlen(str);
    for(pstr=str,pend=str+len-1;
        (pstr<=pend)&&(strcmp(pstr,pend)==0);
        pstr++,pend--)
    {
       ;
    }
    if(strcmp(pstr,pend)==0)
    printf("Yes");
    else
    printf("No");
    return 0;
}
```

Fig. 9. Program of Example 3

is in the statement "strcmp(pstr,pend)==0". It should compare two chars not strings. Thus, the final result is always "No" no matter what test cases are executed. In other words, the program will pass the test cases that expect the result is "No" while will not pass the test cases that expect the result is "Yes". As a result, our system cannot mine the fault information because there is no difference between the successful and unsuccessful test cases.

4) Summary: The fault localization algorithm of our system is based on mining weighted software behavior graph, i.e. mining the difference between correct execution and incorrect execution. This algorithm has good effect, but its principle has some inherent problems which should not be ignored. On the one hand, if the target program crashes, the trace cannot be obtained such that no more information can be mined. On the other hand, the premise of mining differences is that there exist differences between the two kinds of paths. Thus, for the program with less code, fewer branches, the system cannot give the localization results.

C. Directly Outputting Detection

Only one record is detected and confirmed. The problem is to compute the π using the formula $\pi/2 = (2/1) * (2/3) * (4/3) * (4/5) * (6/5) * (6/7) * \ldots$. The number of the multipliers is 100. A student directly outputs the answer. The code is shown below.

```
#include <stdio.h>
int main()
{
    printf("pi␣=␣\%f\n",3.126079);
    return 0;
}
```

The problem has only one test case whose input is empty. Then the judge system could be fooled by directly outputting the only answer of the problem. In this case, the student tried to cheat the system by this way.

VI. Conclusion

Existing program online judge systems have been able to automatically score programs, and play an important role in college teaching. But with the change of demand and the actual use of experience, there is certainly room for improvement. This paper aims at the online judge system of Harbin Institute of Technology for C language, according to the feedback from teachers and students, presents five problems that need to be solved, and gives the specific solutions.

The contributions of this paper are as follows.

1) We introduce fault localization based on traditional online judge systems while traditional online judge systems can only locate the compilation errors.
2) We present an approach that can detect the cheat that forging answers. Most online judge systems use a test-case-driven way for scoring. The problem that they cannot detect the cheats by forging answers is their inherent problem. Our approach addresses this issue.
3) We add a convenient method for submitting code that enables a student to submit code by pasting instead of uploading a file. A student can view the last submitted code when opens the submitting page again. It assures the correctness of a submission and improves the usability of the system.

References

[1] S. H. Edwards and M. A. Perez-Quinones, "Web-cat: Automatically grading programming assignments," in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '08. New York, NY, USA: ACM, 2008, pp. 328–328. [Online]. Available: http://doi.acm.org/10.1145/1384271.1384371

[2] G. Rößling and A. Kothe, "Extending moodle to better support computing education," in *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '09. New York, NY, USA: ACM, 2009, pp. 146–150. [Online]. Available: http://doi.acm.org/10.1145/1562877.1562926

[3] J. Hollingsworth, "Automatic graders for programming classes," *Commun. ACM*, vol. 3, no. 10, pp. 528–529, Oct. 1960. [Online]. Available: http://doi.acm.org/10.1145/367415.367422

[4] S. Kumar, A. Gankotiya, and K. Dutta, "A comparative study of moodle with other e-learning systems," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 5, April 2011, pp. 414–418.

[5] M. Dougiamas and P. Taylor, "Moodle: Using learning communities to create an open source course management system," 2003.

[6] K. Brandl, “Are you ready to moodle,” *Language Learning & Technology*, vol. 9, no. 2, pp. 16–23, 2005.

[7] Y.-f. XIE, M.-x. DU, T.-t. SU, and J. TIE, “Extension and development of moodle teaching system,” *Modern Computer*, vol. 23, p. 024, 2013.

[8] H.-Y. Hsu, J. A. Jones, and A. Orso, “Rapid: Identifying bug signatures to support debugging activities,” in *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 439–442. [Online]. Available: http://dx.doi.org/10.1109/ASE.2008.68

[9] S. Yang, “Reseach on fault localization and fault comprehension method based on weighted software behavior graph,” Master’s thesis, Harbin Institute of Technology, June 2014.

[10] A. Aho, M. Lam, R. Sethi, and J. Ullman, *Compilers: principles, techniques, and tools*. Pearson/Addison Wesley, 2007, vol. 1009.

[11] B. Inc, “Baidu echarts,” https://ecomfe.github.io/echarts/index-en.html, 3 2016.