A Programmer Self-training System with Programming Skill Evaluation and Personalized Task Recommendation

Shu Lin, Qinjian Zhang, and Wenxin Li
Department of Computer Science, Peking University, Beijing, P.R.China

Abstract— *Thanks to the widespread use of computers, the basic programming ability is becoming an essential skill for almost all the college students. Although many current online judge systems provide stable and efficient services in automated programs testing [1], it seems that there is limited help to the learners since few study suggestions is given to them. In this case, learners may feel disoriented during their studies. This paper aims to solve this problem by implementing a system based on one existing platform – Bailian, which will evaluate the learner's programming skills by combining three scores from different perspectives separately, and then recommend some suitable tasks to them according to the grades of both learners and problems. The system also includes an automatic contest generator to generate contests for self-testing. The quality of these contests is guaranteed since the grades as well as the categories of problems are considered during generating.*

Keywords: Online judge system, Bailian, programming skill evaluation, personalized task recommendation, automatic contest generation

1. Introduction

With the advent of information era and the popularization of computers, programming is no longer a peculiar skill of programmers. Almost all subjects requires programming skills – for example, we need to process large data sets or solve complex formulas in mathematics and physics, tabulate statistics data or predict the developing trade of some object in social science, design and produce fantastic animations in arts, and so on. Without programming, these jobs would become tall orders.

Nowadays, there are many kinds of platforms designed for helping learning programming, and the most efficient one is called Online Judge System, or "OJ" for short. An online judge system may contain hundreds of thousands of problems which demand diverse ideas and techniques to solve. A learner can practice his programming skills by registering a new account, opening one problem and reading its description, coding a program which does exactly what the problems demand, and submitting the solution to the online judge system. Because the online judge system uses robot judges instead of humans, it will return an objective verdict in mere fractions of seconds just after the submission. Thus the learner is able to know whether his solution is correct or not, and may correct the errors according to the result [2].

However, it is far from enough for learners. Since most of online judge systems are initially designed for programming contests, beginners will be completely flummoxed by the sea of problems – in another word, the green hands have no idea where to begin. Even old hands may lose their positions if they are not aware of their actual grades. It happens occasionally when a learner

- Tries to solve hard problems which beyond his ability, or
- Has solved many similar problems redundantly, which he should not have to.

Both of them are wasting learner's precious time.

Another flaw in most of the current online judge systems is the utilization rate of problems is quite low. Learners prefer to solve "famous" problems, because these problems have been solved many times, and proved to be good ones. Also it is easy to find lots of reference solutions for these problems, which is really helpful once a learner is getting into trouble. If things continue in this way, the "famous" will become more "famous", fewer and fewer learners show their interest in other "infamous" problems, even some of those "infamous" ones may be good.

In this paper, we are going to design and implement a system that

- Evaluates programming skill of learners;
- Gives some personalized study advices (by task recommendations) to learners;
- Generates contests for testing learner's programming ability.

The system is based on OpenJudge, an existing online judge system maintained by Peking University.

In order to provide a better programming learning platform, at first we have graded all problems according to the levels of difficulty. We have also classified them by what algorithm is used during solving. Then for each learner, we determine the grade of his programming skill by the grades as well as the classifications of solved problems. After that, we can recommend some proper problems to the learner. Finally, we provide a service that generates contests individually for each learner, which will test learner's programming skill in a more effective way.

The next section describes the current online judge systems in detail. From Section 3 to Section 7 shows the current

implementation of our system. Section 8 introduces how to use our system. Finally, we draw a conclusion in Section 9.

2. Brief Introduction to Online Judge Systems

2.1 General View

In a general online judge system, users need to solve problems in the problem set. Two kinds of problem sets are available:

- Practices: Users can solve the problems whenever they want;
- Contests: Users need to solve the problems within the limited time.

While solving a problem, the user reads the problem description, picks one of his favorite programming languages and writes a program which does exactly what the problem demands, then submits it to the online judge system. Once the robot judge on the system receives a program from the user, it will test the program by some test cases to see if it runs normally, and produces the right answer under the problem requirements[1]. It takes only a few seconds. Then the user will receive a verdict indicating the result of the judgment.

The rules of scoring the solution vary in different systems. Many of them use the ACM International Collegiate Programming Contest (ACM/ICPC) rules. The user gets full score if and only if the solution produces right answers in all test cases, otherwise, no score points received. The submitted time and the penalty time (twenty minutes for every previously wrong solution for that problem) is also considered in the contests [3]. Other systems use rules that the solution gets score points of each right answers. Some systems even use more complicated method to calculate the score, for example, TopCoder [4].

2.2 POJ

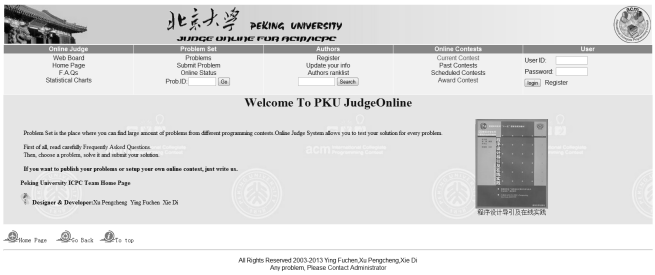

Fig. 1: POJ Home Page

Figure 1 shows the home page of Peking University Online Judge System (POJ [5]). It is developed by Artificial Intelligence Lab of Peking University in 2003. POJ used to be an ACM/ICPC training platform for Peking University students only. Today, POJ is a platform not only for contest training but also for daily practice, and it is providing services to learners around the world.

POJ is a typical "ACM/ICPC-style" online judge system. Most of the problems are selected from ACM/ICPC and the ACM/ICPC rules are applied in scoring users' solutions. As a result, the problems might be too hard for the beginners to start with.

2.3 OpenJudge

Fig. 2: OpenJudge Home Page

Several years ago, the POJ development team had released a free version POJ, so that everyone can build their own judge systems for programming related courses. However, due to the difficulties of system configuration and lack powerful servers to support, it does not work out fine. Therefore, a new online judge system – OpenJudge [6] is built for this application. Teachers or coaches can create their own private groups on OpenJudge, and use their own problem sets for teaching or training [7].

At present, there are over 180 active groups on OpenJudge, including middle school hobby groups, collegiate teaching groups, as well as many on-job programmer groups.

The problems in OpenJudge are from many different sources: ACM/ICPC, Olympiad in Informatics for high school students, homework of academic courses, etc. So that learners in any grade can find suitable problems for themselves in OpenJudge.

3. Preparation

We choose the largest public group – Bailian [8] (Means "Hundreds of Practices" in English) group as the basic platform to implement our system. That is because:

- This group contains the most learners among all groups;
- Everyone can join this group without any limitation;
- All problems in the problem set are public to everybody;

[1]Each problem has a time limit as well as a memory limit for the programs.

- There are many easy as well as hard problems in the problem set, meeting the needs of both beginners and expert programmers.

So far, there are up to 2267 problems in the problem set of Bailian. It takes nearly half a year grading and classifying these problems manually.

3.1 Grading Problems

We have graded all problems in our system from Grade 1 to 5 according to the levels of difficulty, Grade 1 is the easiest and Grade 5 is the hardest.

Table 1: The result of problem grading.

Grade	Count	Level of Difficulty
Grade 1	288	Basic programming training
Grade 2	368	Simple but fallible process
Grade 3	429	Data structures and algorithms
Grade 4	418	Algorithm analysis
Grade 5	641	Challenges for contestants
Other	123	Do not recommend

Table 1 shows the number of problems and the levels of difficulty on each grade.

We also suggest that:

- On Grade 1, learners are going to get familiar with programming languages as well as our system.
- On Grade 2, learners are facing problems which may be simple but required a lot of patience. These problems will make the learners realize that how important a clean coding style is – nobody enjoys finding bugs in a mess.
- On Grade 3, learners start to learn some basic algorithms as well as data structures, which are important in increasing performance of programs.
- On Grade 4, learners need to know how to analyze advanced algorithms, for example, discuss the time complexity and space complexity of them, or do some mathematical deductions before solving problems.
- On Grade 5, these problems are so hard that only recommended to the expert programmers.
- For other ungraded programs, we will not recommend them to anyone because they are not worth solving[2].

3.2 Classifying Problems

Another preparation before evaluating learner's programming skill is classifying problems by the algorithms used in the standard solutions.

We roughly classify the problems into ten categories [9], which are listed in Table 2.

[2]Sometimes the description of a problem is ambiguous, or there is something wrong in the test data, or the problem is almost the same as another one.

Table 2: Categories of problems.

Categories	Count
Basic Practice	382
Divide-and-conquer	49
Greedy Algorithms	86
Dynamic Programming	335
Search Algorithms	306
Simulation	237
Data Structures	130
Graph Algorithms	236
Computational Geometry	131
Number-Theoretic Algorithms	252

4. Programming Skill Evaluation through Different Perspectives

Almost every online judge system is using the same method for calculating user's rank: just sort the users in a decrease order of their scores.

Though this method is quite simple, it has some disadvantages:

- The weight gap between easy problems and hard problems is too small, especially in ACM/ICPC scoring rules (no difference at all). In this case, many users will prefer to solve easier problems in order to raise their rank quickly. There is no benefit to improving their programming skill.
- Some users practice hard and have solved a lot of problems, but always get bad scores in contests; other users seldom practice programming on the system, but often perform well during contests. In this method, the previous ones are better, but apparently the truth is to the contrary.
- Users only need to concentrate on solving their problems, though helping others is also a good way to improve their programming skill. There is no reward mechanism for contribution.

In our evaluation system, we evaluate learner's programming skill in three independent perspectives: accumulation, challenge, and contribution. The result of evaluation is much more accurate and it encourages learners to do things which are more useful.

4.1 Accumulation Score

We still calculate accumulation score by adding the weights of solved problems. However, these weights are quite different from previous ones – they are related to how many times the problems have been solved. The fewer times a problem has been solved, the higher weight it will have, and vice versa. In addition, we want to enlarge the weight gap between easy problems and hard problems. Then the learners may be led to challenge harder ones.

The problem weights are calculated by Function 1.

$$W(x) = \max\left(100 - 10\log_2(x), 1\right) \tag{1}$$

Where x is the times the problem has been solved. The image of it is shown in Figure 3.

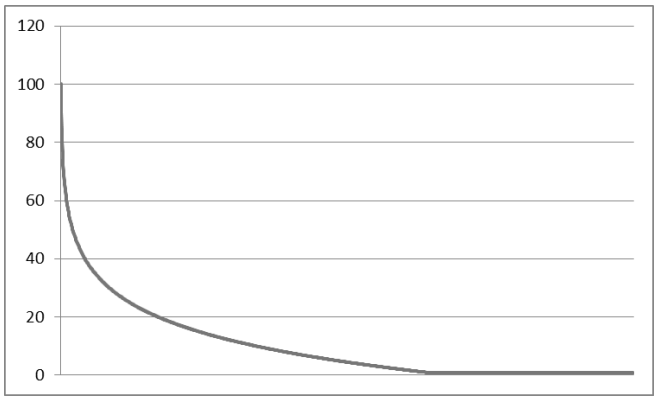

Fig. 3: The Image of W(x)

4.2 Challenge Score

Challenge score is designed for evaluating learner's performance during the contests. It can also make the contests more competitive, and contestants will gain more experience from it.

In our system, learner's challenge score gained in a contest can be calculated as follow:

CALCULATE-SCORE(CONTEST)

```
1  user ← {users joined in contest ordered by rank}
2  N ← length(user)
3  for i ← 1 to N
4        do score[i] ← previous score of user[i]
5  ▷ Discard contests which few users join in
6  if N < 10
7        do exit
8  for i ← 1 to N
9        do point[i] ← (−99i+100N−1)/(N−1)
10 for i ← 1 to N
11       do for j ← 1 to i − 1
12             do if score[i] > score[j]
13                   do point[i] ← point[i] − 1
14 sum ← 0
15 for i ← 1 to N
16       do sum ← sum + point[i]
17 for i ← 1 to N
18       do score[i] ← score[i] + (10·N·point[i])/sum
```

The function for calculating $point$ on Line 9 is linearly related to the rank of contestants, and the champion will get 100 points, the last one will get only one point.

For each contestant, his $point$ will decrease by the number of players who perform better than him in this contest but have lower challenge score before this contest. The code from Line 10 to Line 13 is dealing with the process.

The final challenge score a contestant received is calculated on Line 18. It ensures that the total challenge scores received in a contest is ten times of the number of contestants. Notice that someone may "gain" negative challenge score due to his extremely awful performance during the contest.

To avoid losing challenge score, contestants will pay more attentions in contest. Everyone will try his best to compete, making the contest becomes a rat race, which is good for every contestants.

4.3 Contribution Score

We also encourage programming learners to help others or even make contributions to the development of our system by giving them contribution score. It gives the learners a sense of accomplishment, which will push them to study harder. We always believe that, only when a learner can be able to teach or test others, does he really mastery the knowledge.

We provide three different ways of contribution, from easy to hard:

1) Write short but critical hints for the solved problems;
2) Share the studying experience with others by giving a public lecture;
3) Set some new problems for coming contests.

5. Grading Programming Ability

We make use of the result of problems grading and classifying to grade the learners' programming ability. The accumulation, challenge, and contribution scores are also take into account.

We divide the learning process into three phases:

1) The main task at Phase 1 is practicing more, and learning new skills.
2) Once the learner holds lots of knowledge in hand, he may start to join contests frequently, in order to raise the ability of programming under pressure.
3) After programming for several years, the learner may become an expert. Then he is encouraged to show his talent in helping others.

6. Personalized Task Recommendation

The other important service provided by our system is personalized task recommendation, which can be important study advices for learners.

Before recommend problems to a learner, we first examine into the problems solved by him to determine which grade he is on. On each grade, a learner is required to solve enough number of problems of the same grade for each category. Once the requirement is met, the learner will upgrade.

After determined the learner's grade, we find some suitable tasks for recommendation using the algorithm:

TASK-RECOMMENDATION(USER)

```
1  solved ← {problems solved by user}
2  grade ← previous grade of user
3  recommended ← {previous recommendation}
4  while solved meets the requirement on grade
5        do grade ← grade + 1
6           recommended ← {}
7  recommended ← recommended − solved
8  while length(recommended) < 5
9        do problem ← select(grade, recommended)
10          Insert problem into recommended
```

The function "select(grade, recommended)" on Line 9 will select a problem randomly from problem set which:

- It has not been solved by the learner yet;
- Its grade is equal to the learner's grade;
- Its category is not appeared in previous recommended problems.

These limitations ensure that, the recommended problems are suitable for the learner according to their difficultness, and we will not give two or more similar problems to the learner at the same time.

In addition, we put the recommended problems on a striking position – on the right side of Bailian home page (see Figure 4). Thus, it will remind the learners to solve these problems all the time.

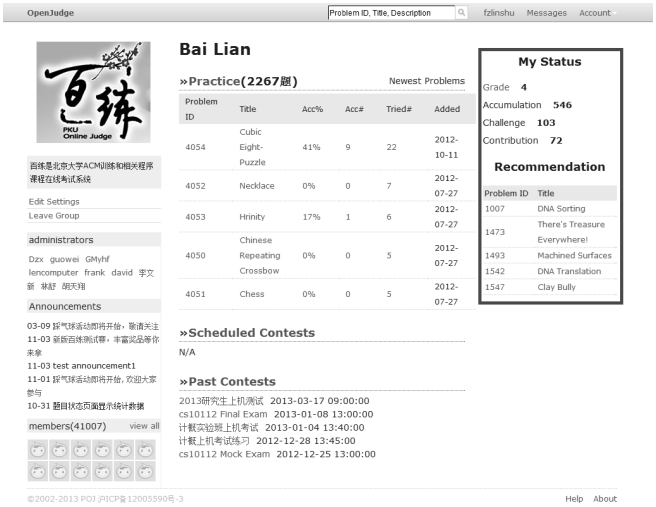

Fig. 4: Bailian Home Page

7. Automatic Contest Generator

A regular contest provides a platform for contestants to enjoy competing with each other. However, it requires new problems, which is hard to get. And the result of the contest contains no useful guideline for testing what learners have already learned, because at most time, the problems are either too easy or too hard for them.

We also implement an automatic contest generator for generating private contests for learners. Unlike a regular contest, a private contest is only available for a particular learner. The only opponent in a private contest is the learner himself. And we do not need to worry about having no new problems, since the problem set of Bailian is so large that we can always find some suitable unsolved problems for only one learner.

The process of problems selection is quite similar to the one in problem recommendation. However, in private contest, eight problems of different categories will be selected instead of five, and two of them come from the next grade – to see if the learner is ready for upgrading. The result of the private contests will be recorded.

The automatic contest generator is used for not only testing oneself but also generating examinations for course teaching. Teachers may simply input the expected number of problems, the average difficultness of problems, and then press "OK" – a suitable exam is born. It will save teacher's time from problem selection, and make the resource of problems be fully used.

8. Directions for Using Our System

At present, our system provides 2267 problems for users. Thousands of active users and hundreds of active groups are using our system for programming learning or teaching or something related.

We are pleased to give some brief instructions to both learners and teachers who want to experience the services on our system.

8.1 Directions for Learners

As a learner who is going to start your way of programming learning on our system, you need to

1) Register a new account on OpenJudge [10].
2) Join in Bailian group by clicking "Join in" on the home page of Bailian [8].

After that, you are able to see the "My Status" column on the right side of the page, including evaluation scores of your programming skill as well as recommendations. Now you can practice yourself by solving recommended problems, or generate a contest for self-testing.

8.2 Directions for Teachers

As a teacher who decides to use our system on your course, you are required to

1) Register a new account on OpenJudge.
2) Send a Request for creating a new group to the webmaster of OpenJudge [11]. You will received a reply from the webmaster soon.
3) If your request is accepted by OpenJudge, you will see a new private group under your full control. You can arrange the problem set, contests, and users in this

group freely. And you can use the automatic contest generator as many times as you want.

4) You need to ask your students to join in this group first, or they will have no access permissions on the resource in it.

9. Conclusion

The evaluation and recommendation system and the automatic contest generator we have implemented make a great improvement on the original Bailian group. The purpose is to provide a more suitable programming learning platform for learners at all levels. Learners can be able to studying programming on our system by themselves. As well, they are encouraged to exchange their experience and ideas during learning. And the system can be applied in any programming related courses.

There is still room for improvement in our system. For example, grade problems in an automatic way. Although the result of classifying a problem's category is quite unanimous among most experienced programmers, the grade of the problem may be controversial. That is because the difficulty of the problem depends on subjective judgments from different people. In this case, we need to do more research on automatic problem grading to make the grades more objective and forcible.

We also plan to evaluate the actual effectiveness of our system by tracking learner's behavior. We will collect the feedbacks from learners as well as the records of learner's actions, in order to figure out how much has the system changed in learner's programming learning.

We are welcoming programming learners from all over the world into our system for studying programming. And we are also looking forward to seeing more and more teachers are using our system in their courses.

References

[1] Wikipedia. (2013) Online judge - Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Online_judge

[2] S. S. Skiena, M. A. Revilla, *Programming Challenges*, 2003 ed. New York, USA: Springer, 2003.

[3] ACM/ICPC. (2013) World Finals Rules. [Online]. Available: http://icpc.baylor.edu/worldfinals/rules

[4] Topcoder. (2012) Algorithm Competition Rating System. [Online]. Available: http://apps.topcoder.com/wiki/display/tc/Algorithm+Competition+Rating+System

[5] (2013) The POJ website. [Online]. Available: http://poj.org/

[6] (2013) The OpenJudge website. [Online]. Available: http://openjudge.cn/

[7] OpenJudge. (2013) About. [Online]. Available: http://openjudge.cn/about.html

[8] (2013) The Bailian website. [Online]. Available: http://poj.grids.cn/

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, USA: MIT Press, 2009.

[10] OpenJudge. (2013) Register. [Online]. Available: http://openjudge.cn/register/

[11] OpenJudge. (2013) Create New Group. [Online]. Available: http://openjudge.cn/groups/new