Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

Student Log Analysis Functions for Web-based Programming Education Support Tool pgtracer

Tetsuro Kakeshita[1,a) Kosuke Ohta[1]

Received: May 5, 2018, Revised: November 29, 2018,
Accepted: March 11, 2019

Abstract: We are developing a programming education support tool pgtracer as a plug-in of well-known Web-based Learning Management System Moodle. Pgtracer provides fill-in-the-blank questions composed of a C++ program and a trace table to students. When a student answers a question by filling the blanks, pgtracer automatically collects student's answers, required time, evaluation result, etc. as student log. In this paper, we propose and evaluate seven analysis functions of the student log. The student log analysis functions are classified into the analysis functions of a student, those of a question, and those of an answering process. A teacher can analyze achievement level of the students and difficulty level of the problems utilizing the analysis functions. Some of the functions are also provided to the students as student feedback functions. We perform a preliminary evaluation of the analysis functions against two teachers teaching computer programming to demonstrate the usefulness of the proposed functions. We also perform an evaluation experiment at an actual class and demonstrate usability of the student feedback functions. The two teachers and more than 80% of the students positively appreciate the proposed functions through our survey evaluation.

Keywords: learning analytics, e-Learning, computer programming, fill-in-the-blank question, Moodle

1. Introduction

Computer programming is essential at universities particularly majored in engineering and science. However there are obstacles in programming education since student's ability is declining due to the increase of the students at higher education. The lack of support staff for programming education is frequently observed. It is important for a student to learn computer programming through exercise with an appropriate level according to the achievement level of the student.

We are developing a programming education support tool pgtracer [1], [2], [3]. Pgtracer is developed as a plug-in of a well-known learning management system Moodle so that a student can use pgtracer at any time and place as long as the internet connection and a personal computer are provided.

A teacher can create fill-in-the-blank questions using pgtracer. A fill-in-the-blank question is composed of a C++ program and a trace table representing the execution order of the statements and the values of the valuables at each step. A teacher can define various types of blanks within the question.

A student performs an exercise by filling out the blanks. Pgtracer automatically evaluates the submitted answers and provides the score to the student. Pgtracer also collects student log record each time a student fills a blank. The log record contains student's answer, required time, evaluation result etc.

In this paper, we propose seven analysis functions for the collected log records. The aims of the analysis functions are twofold. For the teachers, the analysis functions provide learning and achievement information of each student and the entire class as well as the analysis data of each blank and question. For the students, the analysis functions provide learning and achievement information about the student as well as the analysis data of each blank and question. We shall call the second type of the analysis functions as student feedback functions.

The analysis functions are composed of the analysis functions of each student, those of a question, and those of an answering process. A teacher can analyze the achievement of each student and the entire class by utilizing the analysis functions. The analysis result will be a useful input to improve programming education. We utilize the analysis functions utilizing the data collected at an actual class. We shall demonstrate the usefulness of the analysis functions through the analysis. Two teachers teaching computer programming at Kumamoto National College of Technology reviewed a preliminary version of the analysis functions and positively appreciate the current version of the analysis functions.

Some of the analysis functions are also provided to the students as student feedback functions. We perform an evaluation experiment of the student feedback functions at an actual class. We have obtained positive feedbacks from many of the students through the experimental evaluations.

This paper is organized as follows. We shall demonstrate the basic process of programming education utilizing pgtracer in Section 2. Basic functions and features of pgtracer are introduced in Section 3. We also demonstrate the difference with the similar tools in order to demonstrate originality of the proposed student log analysis functions. We next present the overview and the detail of the student log analysis functions in Section 4. Some of the functions are also provided to the students as student feedback functions. The outline of the evaluation experiment is explained

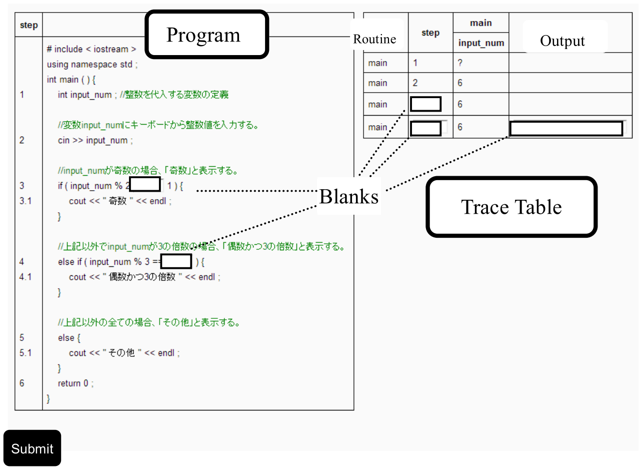
[1] Saga University, Saga 840–8502, Japan
[a)] kake@is.saga-u.ac.jp

in Section 5. The section also contains preliminary evaluation from the viewpoint of the two teachers teaching computer programming. Result of the experiment and the observation of the result for the entire class are presented in Section 6 in order to demonstrate usefulness of the analysis functions. In Section 7, we shall present the survey experiment to evaluate the student feedback functions from the viewpoint of the students.

2. Programming Education Utilizing Pgtracer

A fill-in-the-blank question of pgtracer is composed a pair of a C++ program and a trace table representing execution order of steps with the routine name, values of each variable and output of each step (**Fig. 1**). A student fills the blanks such that the program and the trace table become consistent. Trace table is important for program comprehension and can help students when they get stuck during programming. It is important to visualize execution process of a program for a novice programmer. Thus we expect a trace table as an effective means of programming education especially for beginners.

Pgtracer provides various types of blanks within a program and a trace table so that we can check both program comprehension through blanks within the trace table and program composition through blanks within the program. Therefore we can evaluate a wide range of students with various levels of achievement by providing the following types of fill-in-the-blank questions.



```
step
      # include < iostream >
      using namespace std ;
      int main ( ) {
1        int input_num ; //整数を代入する変数の定義

         //変数input_numにキーボードから整数値を入力する。
2        cin >> input_num ;

         //input_numが奇数の場合、「奇数」と表示する。
3        if ( input_num % 2 [    ] 1 ) {
3.1          cout << " 奇数 " << endl ;
         }

         //上記以外でinput_numが3の倍数の場合、「偶数かつ3の倍数」と表示する。
4        else if ( input_num % 3 ==[    ] ) {
4.1          cout << " 偶数かつ3の倍数 " << endl ;
         }

         //上記以外の全ての場合、「その他」と表示する。
5        else {
5.1          cout << " その他 " << endl ;
         }
6        return 0 ;
      }
```

Routine	step	main	Output
		input_num	
main	1	?	
main	2	6	
main	[ ]	6	
main	[ ]	6	[ ]

Fig. 1 Fill-in-the-Blank question of pgtracer.

- A blank at a value of a variable within the trace table can be used to check student's recognition of change of the value of the variable.
- A blank at a step number, a variable name or a routine name within the trace table can be used to check student's recognition of execution order of statements or corresponding variable.
- A blank at a single token within the program, such as variable name, operator, reserved word, etc. can be used to check student's ability of elementary programming.
- A blank at a sequence of tokens within the program such as expression, statement, compound statement, routine can be used to check student's ability of more advanced programming.

There is a case where more than one right answer exists for a fill-in-the-blank question. Pgtracer evaluates all of such answers as correct, as long as the execution result of the completed program and the correct trace table are consistent. Further detail can be found in Section 3.1.

Pgtracer is developed as a plug-in of well-known lecture management system Moodle. This approach enables to provide a one-stop service for programming education utilizing other educational contents. A student can learn online for 24 hours and 365 days. We utilize a (basically empty) template for Moodle modules to develop pgtracer so that pgtracer is completely our original implementation.

Figure 2 represents a process of programming education utilizing pgtracer. A fill-in-the-blank question is composed of a program, a trace table, a mask for the program and a mask for the trace table. They are described using XML. We separate a program and a mask for the program so that multiple masks with different difficulty levels can be defined for a program. Trace table and the corresponding masks are separated for the same reason. Question DB contains valid combination of the XML files.

When a student answers to a question, the system automatically evaluates the answer and feedbacks the score to the student. The student then can view a right answer after the evaluation. At the same time, pgtracer collects the log data of the answering process and the score whose detail are presented in Section 3.3.

The collected data is utilized to analyze the achievement level and the learning process of each student and the entire class. Pg-

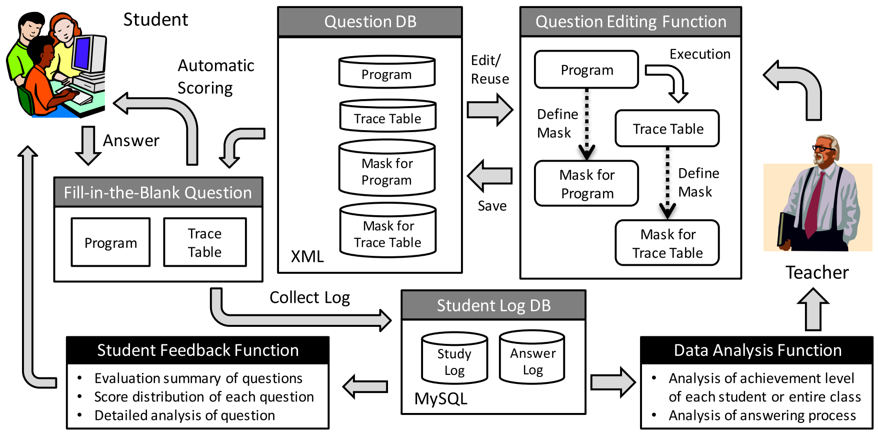


Fig. 2 Programming education process utilizing pgtracer.

tracer provides various analysis functions and the student feedback functions for the collected data proposed in this paper. The instructor can use the analysis functions to improve the educational contents including fill-in-the-blank questions and the instruction to each student. A student can utilize the student feedback functions to understand the achievement of the student.

Pgtracer also provides functions to create and edit XML files representing a program, a trace table, a mask for program and a mask for trace table. XML files representing a program and a trace table are generated by pgtracer. Pgtracer also provides functions to edit the XML files representing masks. The generated files and configurations are stored in the question DB. Further detail can be found in Section 3.2.

As the readers can observe from Fig. 2, various functions of pgtracer are developed to realize PDCA cycle (Plan, Do, Check and Action) of programming education. We thus can realize continuous improvement of the teaching activity through question design (plan), student exercise (do), data analysis (check) and improvement of the questions (action).

3. Basic Functions of Pgtracer

Pgtracer provides various functions to support the programming education process illustrated in Fig. 2. We shall introduce further detail of the basic functions in this section excluding the data analysis function and the student feedback function. We shall also demonstrate difference of pgtracer with the related programming education tools.

3.1 Automatic Scoring Functions

When a student fills a blank, pgtracer first compares the student's answer with the right answer as text. If the two answers are the same, then pgtracer evaluates the student's answer as correct. If the student's answer is an empty string, then pgtracer evaluates the answer as incorrect.

When the two answers are not the same, pgtracer generates a C++ program whose blanks are filled by the student's answer except that the blanks are filled by the right answer when the student answer is an empty string. Pgtracer also generates the filled trace table using the right answer. Then pgtracer executes the filled program to automatically generate the corresponding trace table. Pgtracer evaluates the correctness of the student's answer by comparing the generated trace table with the filled trace table. Therefore pgtracer can correctly evaluate student answers even if a fill-in-the-blank has multiple right answers.

The readers can refer to the detail of the automatic scoring function in our papers [1], [3].

3.2 Question Editing Function

A teacher first prepares a C++ program and the input data to execute the program [*1]. Then pgtracer automatically converts the program to the corresponding XML file representing the program after checking the program syntax through compilation.

Pgtracer provides a function to automatically generate XML file representing a trace table when the teacher specifies an XML file corresponding to a program and the input data.

Pgtracer also provides an editing function of XML files representing program mask and trace table mask. The function shows the selected program or trace table. A teacher can use the function to select an arbitrary token or sequence of tokens to define a mask or a hidden portion of the program or the trace table. Therefore, XML knowledge is not required to the teachers when they create and/or edit fill-in-the-blank questions.

The generated XML files are stored in the question DB. The detail of the question editing function can be found in our paper [2], [3].

3.3 Student Log Collection Function

Pgtracer provides a function to start and stop collection of the student log. A student log is composed of the study log and the answer log.

The study log stores records composed of studyId (id of the record), userId, questionId, score, startTime and endTime. Each record represents a summary of a student answer to a blank of a fill-in-the-blank question. The answer log represents the detailed answering process of the study log. A record of the answer log is composed of studyId, XPath expression of the corresponding blank, student's answer, right answer, elapsed time to fill the blank, and the score of the answer. Each record is collected when a student fills a blank. Since there is a case that the student rewrites a blank with different answer, the score value is calculated only for the final answer of the blank.

The above log collection becomes possible since pgtracer assumes fill-in-the-blank questions. The analysis functions proposed in the next section are designed using the above log records.

3.4 Comparison with Related Tools

Since computer programming is one of the core issues in science and engineering education, various tools are proposed to assist programming education. Pgtracer has various advantages over these tools.

Nishida et al. proposed a learning environment named PEN for novice programmer [4]. A student constructs a program from the scratch and PEN provides various functions for the student to understand the execution of the program. On the other hand, pgtracer utilizes fill-in-the-blank questions in order to focus on portion of the program. Then the learning time, which is the total time to learn a topic with programming exercise, will become shorter by using pgtracer. The difficulty level of the questions can also be adjusted more easily.

There is another programming education tool [5] which utilizes visual programming for novices. However the tool does not analyze learning process and achievement of the students.

Funabiki et al. also proposes a programming education system utilizing fill-in-the-blank question [6] so that there some similarity between the system and pgtracer. But the tool can only define blanks at each reserved word of a Java program so that the flexibility of pgtracer is significantly higher. The system also does not

[*1] Pgtracer recognizes C++ programs containing basic language features such as variable, type, statement, expression, flow control, compound data type, function and standard I/O library. But current version of pgtracer does not recognize advanced language features such as class, template and exception.

collect student log and thus cannot provide the log data analysis functions.

Deperlioglu et al. propose a blended learning model for computer programming [7]. There is a tool utilizing learning analytics [8]. However, the detail of the collected data is not presented.

In addition to the above features, the log analysis function in this paper is based on the log explained in Section 3.3. Although there are many log data analysis tools to support computer programming, pgtracer is the only system whose log contains information of each student answer at each blank. Thus the data analysis functions proposed in this paper can be considered original to the best of the author's knowledge.

4. Student Log Analysis Functions

We shall propose the student log analysis functions in this section. The functions are designed to analyze the student log introduced in Section 3.3 from various viewpoints and different levels of detail. Many of the analysis functions are provided also for the student after eliminating the score of other students so that privacy of other students can be protected.

4.1 Overview of the Analysis Functions

We design the student log analysis functions from three major viewpoints: analysis functions of each student, analysis functions of each question, and process analysis functions. Each viewpoint contains two or three analysis functions. Each function has a separate window. Transition diagram among the windows corresponding to the analysis functions is illustrated in **Fig. 3**.

The seven analysis functions are represented in rectangles in Fig. 3. The functions filled by the gray color summarize the analysis result from the corresponding viewpoint. The remaining functions provide detailed analysis result. Detail of each function will be explained in the succeeding subsections. Pgtracer allows transition between the windows connected by a directed edge.

4.2 Analysis Functions of Each Student

The analysis functions of each student provide analysis results of all questions for each student. A teacher can analyze the learning activities of each student by utilizing these functions.

4.2.1 Utilization Summary

The utilization summary function provides summary of each student in terms of number of attempted questions, total number of attempts, total study time and percentage of the correct answers

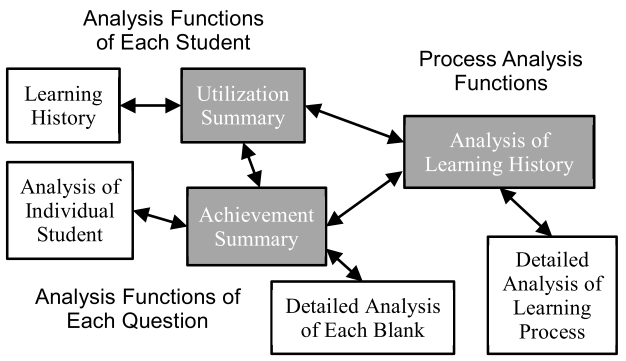

Analysis Functions of Each Student
Learning History
Utilization Summary
Process Analysis Functions
Analysis of Learning History
Analysis of Individual Student
Achievement Summary
Detailed Analysis of Learning Process
Analysis Functions of Each Question
Detailed Analysis of Each Blank

Fig. 3 Transition diagram among windows of the student log analysis functions.

(**Fig. 4**). The "show detail" link is used to view the learning history function of the student. Distributions of the right answer ratio and the study time are also presented. The relationship between total number of attempts and the total study time represents each student using a point.

This function allows a teacher to overview the learning activity and eagerness of each student. It is possible to sort the rows using an arbitrary field. It is also possible to sort the rows using more than one field [*2]. For example, the utilization summary sorted by the number of challenged questions can be used to find students who do not solve the instructed questions. The teacher can understand student workload by observing the distribution of the study time. Then the teacher can adjust difficulty level of the questions in order to control the workload.

4.2.2 Learning History

The learning history function provides the list of questions with the achievement summary of a selected student (**Fig. 5**). The achievement summary is composed of the score, start time and duration for the first trial and the trial with maximum score.

A teacher can check whether the student solved the assigned questions by using the function. It is also possible to retrieve questions which the student is not good at. This can be done by sorting the rows using the score or the duration. The achievement for the first trial represents the actual ability of the student. Thus the above sorting should be performed for the first trial.

On the other hand, the achievement of the trial with the maximum score is used to evaluate the effort of the student. This is because an eager student tends to solve the same question until desired score is obtained. A teacher can check the improvement of the student by observing the difference of achievements of the two trials.

The teacher can analyze more detail by using the process analysis functions.

4.3 Analysis Functions of Each Question

The analysis functions of each question provide analysis results of all students for each question. A teacher can analyze the difficulty level of each question by utilizing these functions.

4.3.1 Achievement Summary

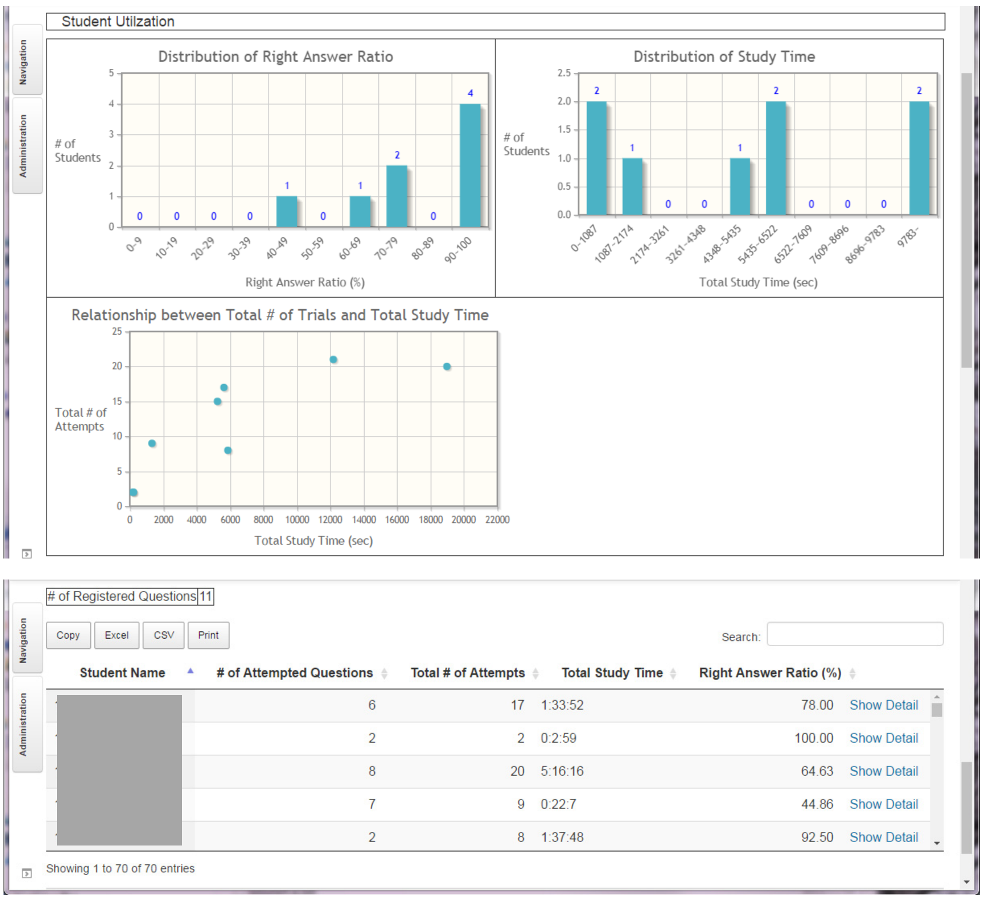
The achievement summary function provides average score, duration, number of students and the average number of trials of the students for each question (**Fig. 6**). The average score and duration are calculated for the first trial and the trial with maximum score of the students. The "show detail" link is used to access the analysis function of individual student.

A teacher can utilize this function to find difficult questions for the students by sorting the questions in the order of average score and duration for the first trial. It is also possible to observe the effect of repetition by comparing the average scores of the first trial and the trial with the maximum score. Such checking of the average score and duration is useful to validate the difficulty levels of the questions.

4.3.2 Analysis of Individual Student

The analysis of individual student provides distributions of the

[*2] The sorting function of pgtracer is implemented using Tablesorter plugin [9].

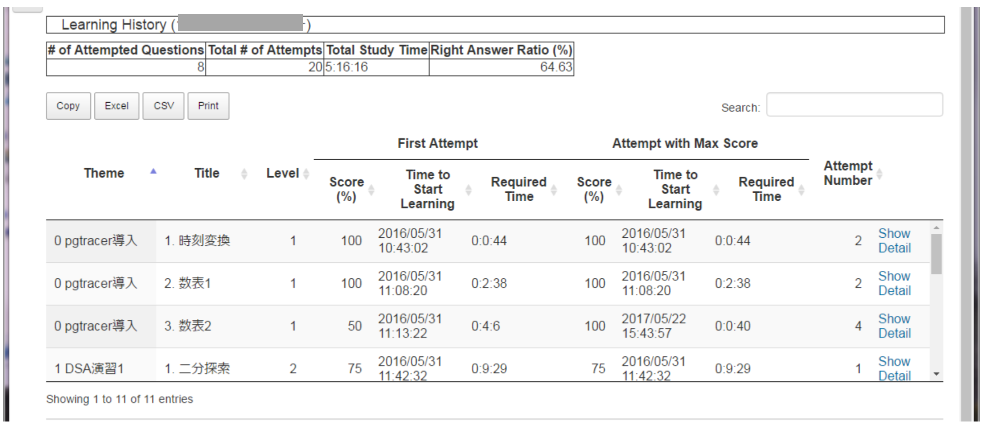


# of Registered Questions 11

Student Name	# of Attempted Questions	Total # of Attempts	Total Study Time	Right Answer Ratio (%)	
[redacted]	6	17	1:33:52	78.00	Show Detail
[redacted]	2	2	0:2:59	100.00	Show Detail
[redacted]	8	20	5:16:16	64.63	Show Detail
[redacted]	7	9	0:22:7	44.86	Show Detail
[redacted]	2	8	1:37:48	92.50	Show Detail

Showing 1 to 70 of 70 entries

Fig. 4 Utilization summary.



Learning History ([redacted])

# of Attempted Questions	Total # of Attempts	Total Study Time	Right Answer Ratio (%)
8	20	5:16:16	64.63

			First Attempt			Attempt with Max Score				
Theme	Title	Level	Score (%)	Time to Start Learning	Required Time	Score (%)	Time to Start Learning	Required Time	Attempt Number	
0 pgtracer導入	1. 時刻変換	1	100	2016/05/31 10:43:02	0:0:44	100	2016/05/31 10:43:02	0:0:44	2	Show Detail
0 pgtracer導入	2. 数表1	1	100	2016/05/31 11:08:20	0:2:38	100	2016/05/31 11:08:20	0:2:38	2	Show Detail
0 pgtracer導入	3. 数表2	1	50	2016/05/31 11:13:22	0:4:6	100	2017/05/22 15:43:57	0:0:40	4	Show Detail
1 DSA演習1	1. 二分探索	2	75	2016/05/31 11:42:32	0:9:29	75	2016/05/31 11:42:32	0:9:29	1	Show Detail

Showing 1 to 11 of 11 entries

Fig. 5 Learning history.

score and required time as well as the relationship between score and required time (**Fig. 7**). It also provides achievement summary of each student for the selected question. The achievement summary is composed of the score, start time and required time for the first trial and the trial with the maximum score.

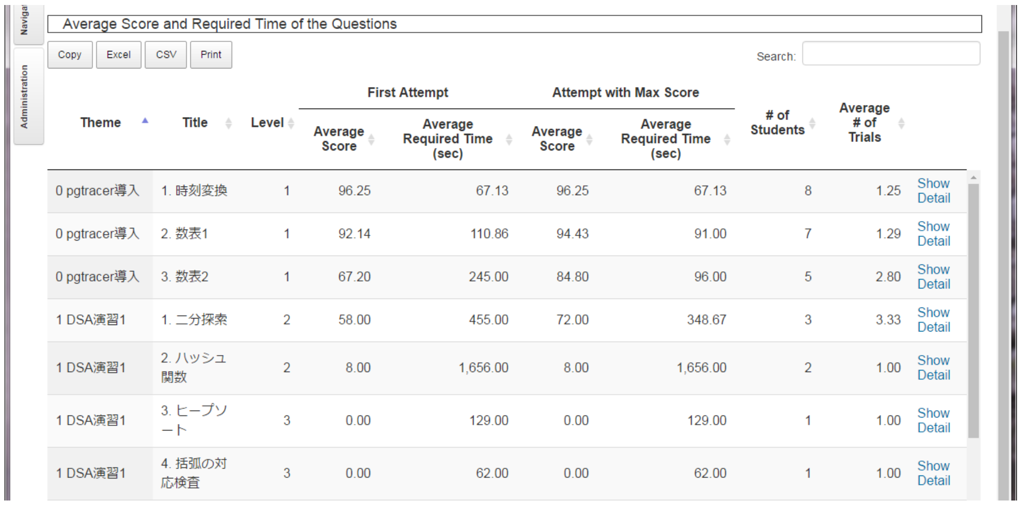
A teacher can easily find students who do not answer the question by sorting the students in the order of score and duration. Such sorting is useful to understand the achievement level of each student for the question. Then the teacher can further analyze the detail of the learning process of each student by using the process analysis functions.

This function provides a link "problem analysis" to the detailed analysis function of each blank which will be explained next.

4.3.3 Detailed Analysis of Each Blank

The analysis function of each question provides summary of the student answers for the selected blank of the specified question (**Fig. 8**). A fill-in-the-blank question is shown at the top of the window. Each blank is automatically assigned a unique number. When a teacher selects a "show detail" link at the blank list, the answers of the students are listed at the bottom of the window. The summary of the answers is also shown in the middle of the window.

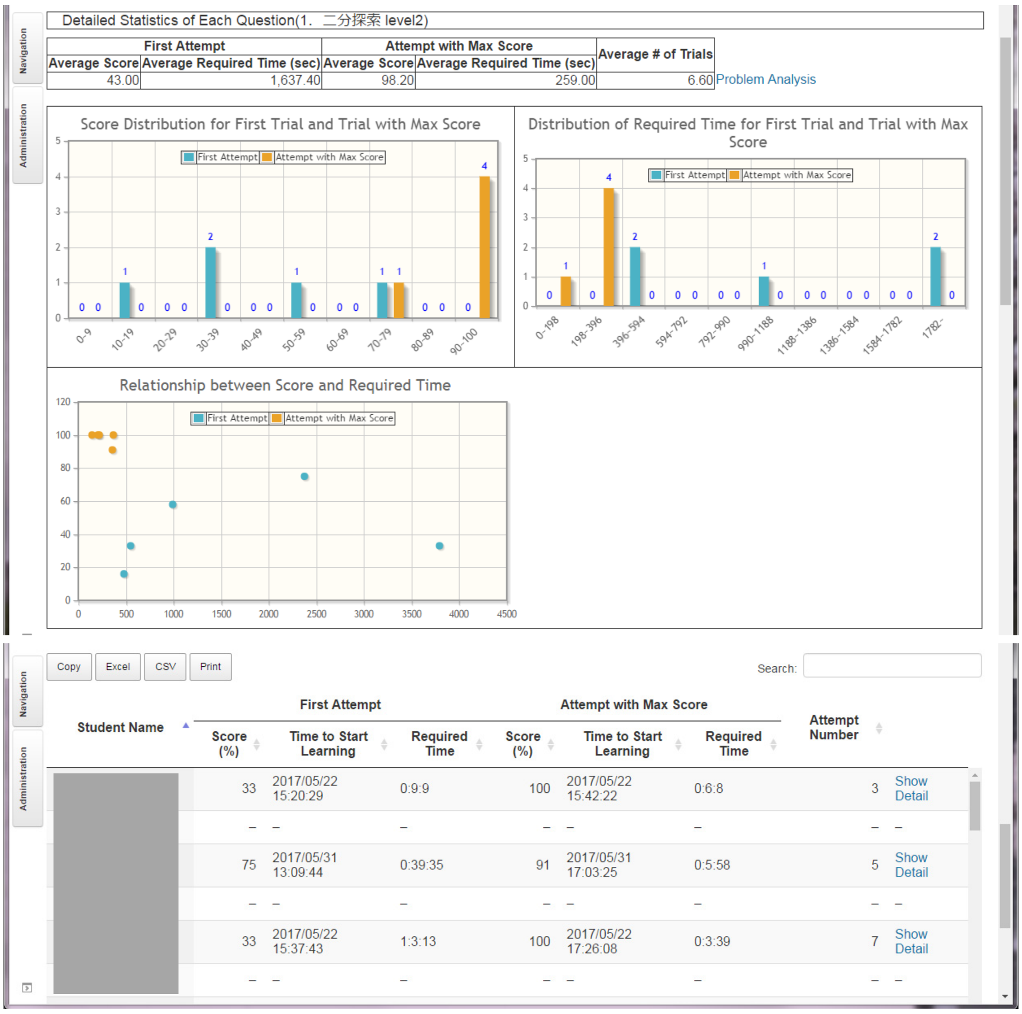
Summary of the student answers contains the number of students and the average duration of each answer for the first trial and the trial with the maximum score. A teacher can observe the summary to find typical mistakes of the students.



Average Score and Required Time of the Questions

Theme	Title	Level	First Attempt		Attempt with Max Score		# of Students	Average # of Trials	
			Average Score	Average Required Time (sec)	Average Score	Average Required Time (sec)			
0 pgtracer導入	1. 時刻変換	1	96.25	67.13	96.25	67.13	8	1.25	Show Detail
0 pgtracer導入	2. 数表1	1	92.14	110.86	94.43	91.00	7	1.29	Show Detail
0 pgtracer導入	3. 数表2	1	67.20	245.00	84.80	96.00	5	2.80	Show Detail
1 DSA演習1	1. 二分探索	2	58.00	455.00	72.00	348.67	3	3.33	Show Detail
1 DSA演習1	2. ハッシュ関数	2	8.00	1,656.00	8.00	1,656.00	2	1.00	Show Detail
1 DSA演習1	3. ヒープソート	3	0.00	129.00	0.00	129.00	1	1.00	Show Detail
1 DSA演習1	4. 括弧の対応検査	3	0.00	62.00	0.00	62.00	1	1.00	Show Detail

Fig. 6 Achievement summary.



Detailed Statistics of Each Question(1. 二分探索 level2)

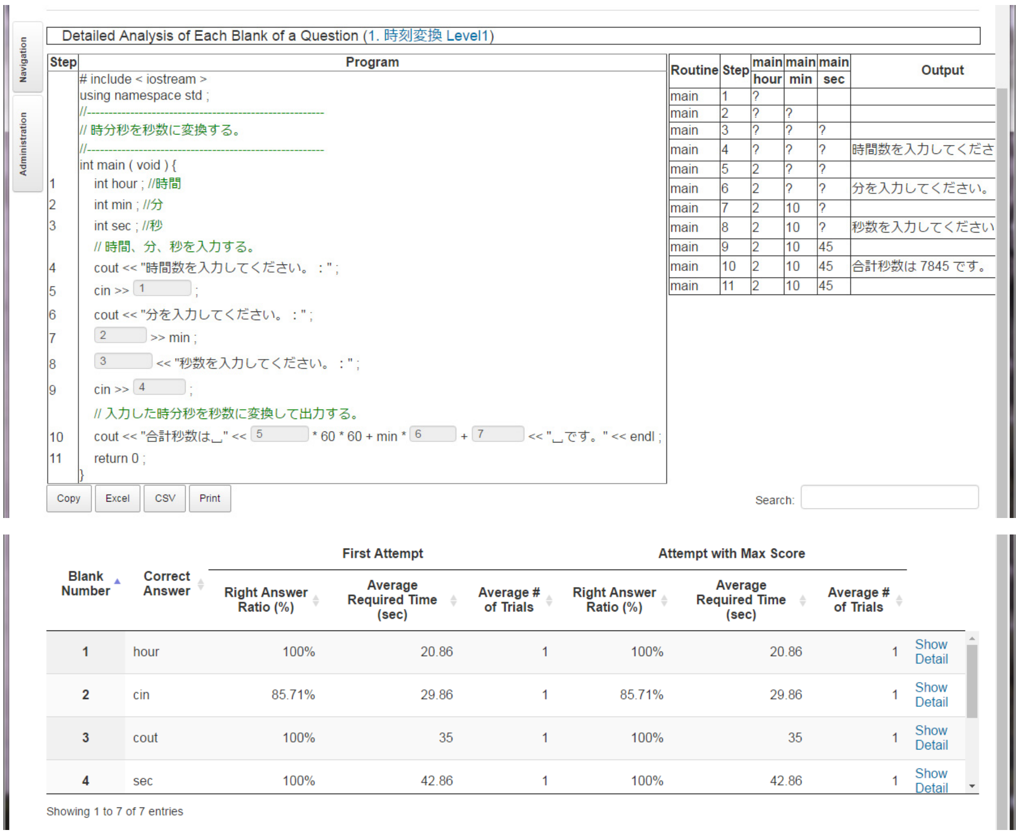
First Attempt		Attempt with Max Score		Average # of Trials	
Average Score	Average Required Time (sec)	Average Score	Average Required Time (sec)		
43.00	1,637.40	98.20	259.00	6.60	Problem Analysis

Student Name	First Attempt			Attempt with Max Score			Attempt Number	
	Score (%)	Time to Start Learning	Required Time	Score (%)	Time to Start Learning	Required Time		
[redacted]	33	2017/05/22 15:20:29	0:9:9	100	2017/05/22 15:42:22	0:6:8	3	Show Detail
	–	–	–	–	–	–	–	–
	75	2017/05/31 13:09:44	0:39:35	91	2017/05/31 17:03:25	0:5:58	5	Show Detail
	–	–	–	–	–	–	–	–
	33	2017/05/22 15:37:43	1:3:13	100	2017/05/22 17:26:08	0:3:39	7	Show Detail
	–	–	–	–	–	–	–	–

Fig. 7 Analysis function of individual student.

The detailed analysis function of each blank provides the list of student answers of the selected blank (**Fig. 9**). The list contains answer, evaluation result (correct or incorrect), required time and the total number of trials of the answer of each student for two types of trials. Since the list can be sorted by arbitrary fields, the teacher can find the students who made the same mistake.

Pgtracer also calculates the average score and the duration of the selected blank for the first trial and the trial with maximum score. The average values are useful to understand the difficulty level of the selected blank. There is a case when a student leaves his seat while answering the blank. We also find a case which a student gives up answering the blank without thinking. Pgtracer provides a function to compute the average values for the answers between the specified duration in order to exclude these cases.



Detailed Analysis of Each Blank of a Question (1. 時刻変換 Level1)

```
Step  Program
      # include < iostream >
      using namespace std ;
      //--------------------------------------------------------
      // 時分秒を秒数に変換する。
      //--------------------------------------------------------
      int main ( void ) {
1        int hour ; //時間
2        int min ; //分
3        int sec ; //秒
         // 時間、分、秒を入力する。
4        cout << "時間数を入力してください。 : " ;
5        cin >> [1] ;
6        cout << "分を入力してください。 : " ;
7        [2] >> min ;
8        [3] << "秒数を入力してください。 : " ;
9        cin >> [4] ;
         // 入力した時分秒を秒数に変換して出力する。
10       cout << "合計秒数は␣" << [5] * 60 * 60 + min * [6] + [7] << "␣です。" << endl ;
11       return 0 ;
      }
```

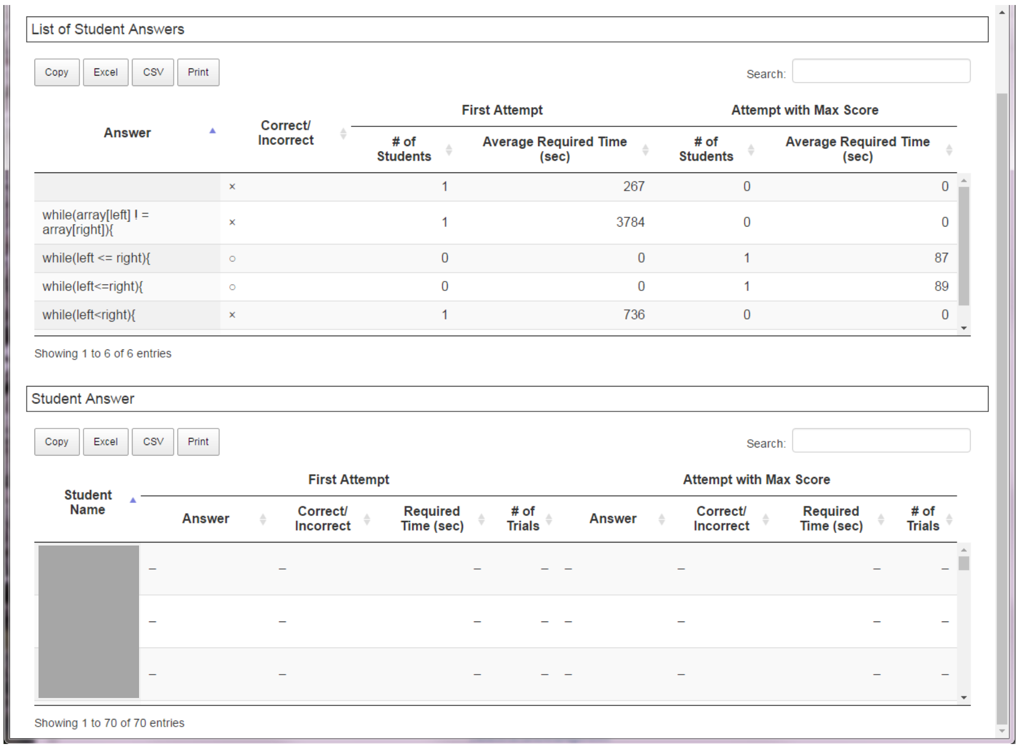
Routine	Step	main	main	main	Output
		hour	min	sec	
main	1	?			
main	2	?	?		
main	3	?	?	?	
main	4	?	?	?	時間数を入力してくださ
main	5	2	?	?	
main	6	2	?	?	分を入力してください。
main	7	2	10	?	
main	8	2	10	?	秒数を入力してください
main	9	2	10	45	
main	10	2	10	45	合計秒数は 7845 です。
main	11	2	10	45	

Copy Excel CSV Print Search:

Blank Number	Correct Answer	First Attempt			Attempt with Max Score			
		Right Answer Ratio (%)	Average Required Time (sec)	Average # of Trials	Right Answer Ratio (%)	Average Required Time (sec)	Average # of Trials	
1	hour	100%	20.86	1	100%	20.86	1	Show Detail
2	cin	85.71%	29.86	1	85.71%	29.86	1	Show Detail
3	cout	100%	35	1	100%	35	1	Show Detail
4	sec	100%	42.86	1	100%	42.86	1	Show Detail

Showing 1 to 7 of 7 entries

Fig. 8 Analysis function of a question.


List of Student Answers

Copy Excel CSV Print Search:

Answer	Correct/ Incorrect	First Attempt		Attempt with Max Score	
		# of Students	Average Required Time (sec)	# of Students	Average Required Time (sec)
	×	1	267	0	0
while(array[left] ! = array[right]){	×	1	3784	0	0
while(left <= right){	○	0	0	1	87
while(left<=right){	○	0	0	1	89
while(left<right){	×	1	736	0	0

Showing 1 to 6 of 6 entries

Student Answer

Copy Excel CSV Print Search:

Student Name	First Attempt				Attempt with Max Score			
	Answer	Correct/ Incorrect	Required Time (sec)	# of Trials	Answer	Correct/ Incorrect	Required Time (sec)	# of Trials
[redacted]	–	–	–	–	–	–	–	–
[redacted]	–	–	–	–	–	–	–	–
[redacted]	–	–	–	–	–	–	–	–

Showing 1 to 70 of 70 entries

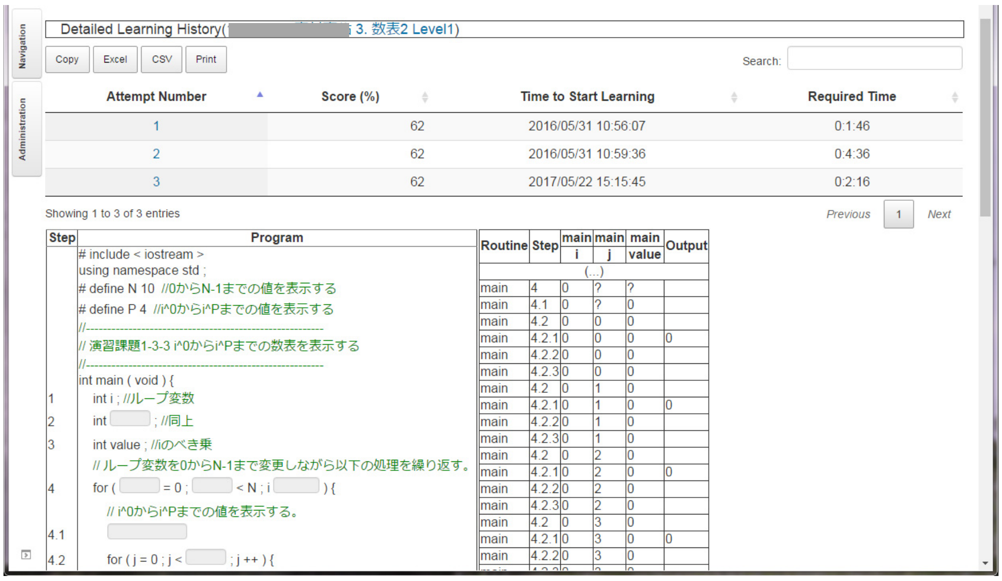
Fig. 9 Detailed analysis function of each blank.

4.4 Process Analysis Functions

The process analysis functions provide detailed analysis results of a pair of each student and each question.

4.4.1 Analysis of Learning History

The analysis function of learning history provides the score, start time and duration for each trial of the selected student and question (**Fig. 10**). A teacher can analyze the effect of multiple trials of the student for the question. The attempt number is also used as the link to the detailed analysis function of the learning process of the trial.


Detailed Learning History([redacted] 3. 数表2 Level1)

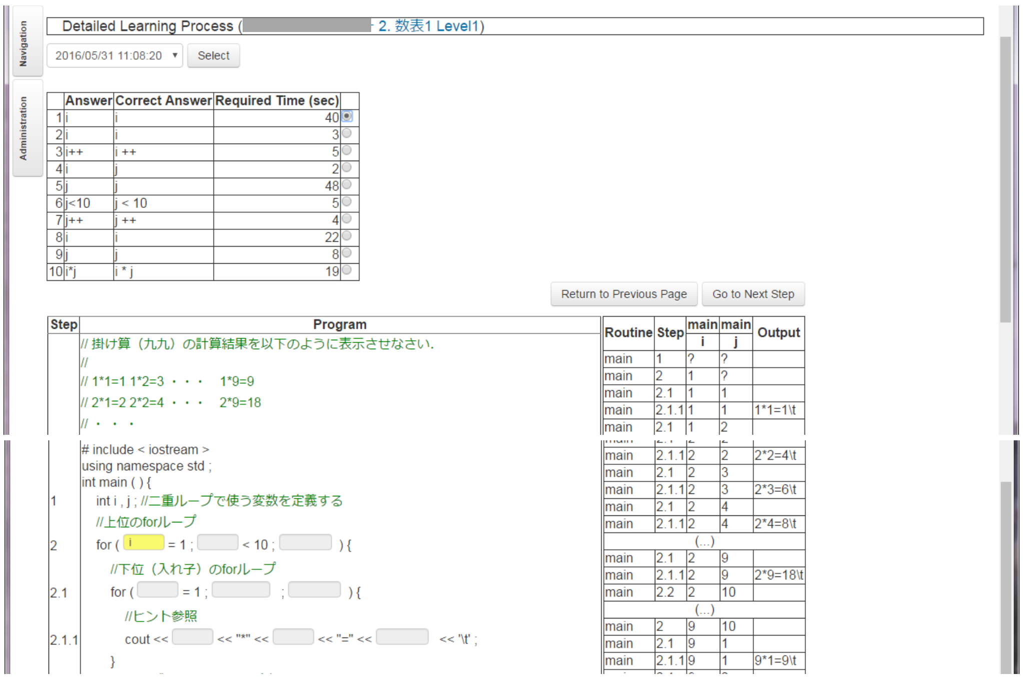
Attempt Number	Score (%)	Time to Start Learning	Required Time
1	62	2016/05/31 10:56:07	0:1:46
2	62	2016/05/31 10:59:36	0:4:36
3	62	2017/05/22 15:15:45	0:2:16

Showing 1 to 3 of 3 entries

```
Step  Program
      # include < iostream >
      using namespace std ;
      # define N 10  //0からN-1までの値を表示する
      # define P 4  //i^0からi^Pまでの値を表示する
      //-----------------------------------------------------
      // 演習課題1-3-3 i^0からi^Pまでの数表を表示する
      //-----------------------------------------------------
      int main ( void ) {
1        int i ; //ループ変数
2        int        ; //同上
3        int value ; //iのべき乗
         // ループ変数を0からN-1まで変更しながら以下の処理を繰り返す。
4        for (       = 0 ;       < N ; i       ) {
            // i^0からi^Pまでの値を表示する。
4.1
4.2         for ( j = 0 ; j <       ; j ++ ) {
```

Routine	Step	main i	main j	main value	Output
(...)					
main	4	0	?	?	
main	4.1	0	?	0	
main	4.2	0	0	0	
main	4.2.1	0	0	0	0
main	4.2.2	0	0	0	
main	4.2.3	0	0	0	
main	4.2	0	1	0	
main	4.2.1	0	1	0	0
main	4.2.2	0	1	0	
main	4.2.3	0	1	0	
main	4.2	0	2	0	
main	4.2.1	0	2	0	0
main	4.2.2	0	2	0	
main	4.2.3	0	2	0	
main	4.2	0	3	0	
main	4.2.1	0	3	0	0
main	4.2.2	0	3	0	

Fig. 10 Analysis function of learning history.


Detailed Learning Process ([redacted] 2. 数表1 Level1)

	Answer	Correct Answer	Required Time (sec)
1	i	i	40
2	i	i	3
3	i++	i ++	5
4	i	j	2
5	j	j	48
6	j<10	j < 10	5
7	j++	j ++	4
8	i	i	22
9	j	j	8
10	i*j	i * j	19

```
Step   Program
       // 掛け算（九九）の計算結果を以下のように表示させなさい.
       //
       // 1*1=1 1*2=3 ・・・   1*9=9
       // 2*1=2 2*2=4 ・・・   2*9=18
       // ・ ・ ・
       # include < iostream >
       using namespace std ;
       int main ( ) {
1         int i , j ; //二重ループで使う変数を定義する
          //上位のforループ
2         for ( i      = 1 ;       < 10 ;       ) {
             //下位（入れ子）のforループ
2.1          for (       = 1 ;        ;        ) {
                //ヒント参照
2.1.1           cout <<       << "*" <<       << "=" <<        << '\t' ;
             }
```

Routine	Step	main i	main j	Output
main	1	?	?	
main	2	1	?	
main	2.1	1	1	
main	2.1.1	1	1	1*1=1\t
main	2.1	1	2	
main	2.1.1	2	2	2*2=4\t
main	2.1	2	3	
main	2.1.1	2	3	2*3=6\t
main	2.1	2	4	
main	2.1.1	2	4	2*4=8\t
(...)				
main	2.1	2	9	
main	2.1.1	2	9	2*9=18\t
main	2.2	2	10	
(...)				
main	2	9	10	
main	2.1	9	1	
main	2.1.1	9	1	9*1=9\t

Fig. 11 Detailed analysis function of learning process.

4.4.2 Detailed Analysis of Learning Process

The detailed analysis function of learning process provides the state of the blanks of the selected question at an arbitrary point of the answering process of the selected student (**Fig. 11**).

The table shown at the top of the window contains the sequence of blank-filling activities of the student. Each activity represents the corresponding student answer, correct answer provided by the teacher and the duration to fill the blank. The teacher can analyze the answering process of the student. The duration can be utilized to evaluate the difficulty level of the blanks from the viewpoint of the student.

When the teacher selects an activity within the table, pgtracer illustrates the state of the blanks when the student performs the selected activity. The yellow blank represents the updated one so that the teacher can easily find the difference of the state of the blanks. The next and previous buttons are provided so that the teacher can easily select the next and previous activities to go forward and backward through the student's learning process.

The teacher can review the answering process of each student by using the detailed analysis function. For example, it can be observed that a student repeatedly fills the same blank. The answering processes are different between the students with high score and low score. Such information will be useful to improve programming education and to guide students effectively.

5. Experimental Evaluation from Teacher's Viewpoint

5.1 Evaluation Plan

We collect student log of 68 students through a programming exercise at an actual class in order to demonstrate effectiveness of the student log analysis functions from the viewpoint of the teachers. The students are taking "Data Structure and Algorithm" course and are learning C++ programming for 10 months at the department of information science, Saga University. The exercise is composed of four questions represented in **Table 1**. The table also contains statistics of each question, average number of trials of each student and the number of students taking the question.

Students are initially explained how to use pgtracer at a class and are given two weeks to complete the exercise. A student can try the same question multiple times. The evaluation score of each student is calculated based on the average score at the first trial of each question and the trial with the maximum score. The students have learned the algorithms used in the four questions at the lecture. The algorithms are described within the programs as high-level comments. Every blank is defined within a program and no blanks are defined within trace table. The program is described according to the guidelines presented in Ref. [12].

5.2 Review Comments and Improvement

We have obtained several review comments from the users of the preliminary version of the analysis functions. The reviewers are two teachers teaching computer programming at Kumamoto National College of Technology. One of the reviewers is a senior professor whose major is not IT, while the other reviewer is an associate professor with IT degree. The following is a list of comments they provided.

- It is better to show duration of the learning time instead of the end time.
- Sorting of the students should be based on student number by default.
- Difficult to compare fill-in-the-blank question and analysis result in the detailed analysis function of each blank.
- Need next and previous buttons to the detailed analysis function of learning process in order to check the answering process step by step.
- Difficult to find the updated blanks in the detailed analysis function of learning process.
- Need download function of the analysis result.
- Better to provide bar chart representing score distribution etc.

We improve the initial version considering the comments. The revised functions are listed below. The two teachers strongly support the revised functions.

- Show duration of the learning time instead of the end time.
- Show the students in the order of student number by default.
- Improve the detailed analysis function of each blank by adding vertical scroll bars to the program and trace table so that a user can easily compare a fill-in-the-blank question and the analysis result of the question.
- Add next and previous buttons to the detailed analysis function of learning process.
- Show the updated blank with yellow color in the detailed analysis function of a learning process.
- Implement the download function of the analysis result to an Excel file using PHPExcel [10])
- Provide a bar chart using JpGraph [11] representing score distribution etc.

6. Demonstration of the Analysis Function through Analysis of the Collected Data

We shall analyze the collected log using the analysis function in this section. We could not collect logs for the initial two hours of the exercise due to defect and correction in the log collection function. As a result, we have lost most of the answer log data for Hashing. The analysis is carried out only using the collected data. However, the analysis can be considered enough in order to demonstrate usefulness of the proposed analysis functions.

The analysis functions calculate all data values presented in **Tables** 1, **2**, **3**, **4**, **5**, **6** and **Figs. 12**, **13**. These tables and figures are prepared by the author for the conciseness and readability of the paper.

6.1 Analysis of Each Questions

62 among 68 students complete the four questions of the exercise. However, some of the answers took more than 5,000 seconds since the student left his seat during exercise. There are other answers whose score is less than 20% and whose duration is less than the required time of the teacher. This can be considered that the student quitted the exercise to see the right answer. Such answers are regarded as invalid and are excluded from the statistics. Table 1 shows statistics of the questions including the number of students providing valid answers. We shall analyze each question for these valid answers in this section.

Table 2 shows the average score, average duration and the number of students providing valid answers. These values are shown

Table 1 Statistics of the questions.

Question	# of Lines	# of Routines	# of Blanks	Average # of Trials	# of Valid Students
Binary Searching	33	1	12	2.54	58
Hashing	57	3	18	2.83	56
Heap Sort	60	3	22	2.61	41
Parenthesis Checking	94	7	26	2.47	45

Table 2 Analysis of trials of the questions.

Question	First Trial			2nd Trial			3rd Trial			Trial with Max Score		
	SC_{ave}	DUR_{ave}	Num	SC_{ave}	DUR_{ave}	Num	SC_{ave}	DUR_{ave}	Num	SC_{ave}	DUR_{ave}	Num
Binary Searching	64.16	983.34	58	86.56	326.94	52	91.67	400.38	24	96.02	398.39	58
Hashing	63.88	2337.25	56	81.22	647.14	51	89.72	461.64	36	91.71	798.14	56
Heap Sort	55.29	1871.34	41	84.37	674.04	39	89.15	496.73	26	91.98	594.85	41
Parenthesis Checking	60.42	1914.53	45	86.43	823.64	42	88.20	737.00	25	88.51	888.78	45

SC_{ave} : Average Score (%), **DUR_{ave}** : Average Duration (second), **Num**: Number of Students

Table 4 Typical mistakes of the students classified by causes.

Blank Type	Cause of Mistake	Example of Mistake	Correct Answer	# of Mistakes	Ratio of Mistakes
Subroutine Call	Incorrect parameter	push(i);	push(x);	40	27.03%
	Incorrect routine name	paren = push();	paren = pop();	7	4.73%
	Forget to call subroutine	paren = stack[i];	paren = pop();	35	23.65%
	Forget to save return value	pop();	paren = pop();	45	30.41%
	Incorrect function call	paren = pop;	paren = pop();	21	14.19%
If/else statement	Single error of comparison operand	if(a == c){	if(a == b){	74	66.67%
	Double errors of comparison operand	if(c == d){	if(a == b){	11	9.91%
	Incorrect comparison operator	if(a > b){	if(a == b){	3	2.70%
	Other causes of comparison expression	if(c > d){	if(a == b){	23	20.72%
Variable assignment	Incorrect variable to be assigned	left = mid – 1;	right = mid – 1;	98	46.89%
	Incorrect value assigned	right = left – 1;	right = mid – 1;	111	53.11%
Others	Misspelling	paran	paren	103	14.37% (※)
	Missing semicolon	paren = pop()	paren = pop();	25	3.49% (※)

※ Ratio against all types of mistakes.

Table 3 Statistics of the blank types.

Blank Type	Right Answer Ratio	Average Duration (second)	# of Blanks
Standard Input	93.10%	171.34	1
Standard Output	92.86%	46.57	1
Return statement	90.58%	40.02	5
Decrement	77.05%	65.69	2
Increment	74.24%	76.09	2
If/else statement	72.77%	64.39	13
Subroutine call	62.82%	167.62	2
Subroutine call & Variable assignment	60.95%	135.80	5
Variable assignment	57.90%	74.88	18
While statement	52.94%	95.32	5
While statement & Subroutine call	52.94%	111.65	1
If/else statement & Subroutine call	38.22%	118.62	3
Else statement & Variable assignment	31.03%	70.62	1
Variable assignment & Increment	29.73%	146.22	1

for various types of trials. The first trial represents the actual programming skill of the students. The readers can observe that the average score becomes higher for the second and third trials and that the average duration tends to become shorter. This can be considered as the improvement due to multiple trials. Another reason is that the students see the right answer at the end of each trial.

It should be noted that the average duration of Hashing for the first trial (2,337 seconds) is quite long compared with other questions and trials. This is because most of the students started the exercise from the Hashing question so that the students required time to become familiar with pgtracer.

Figures 12 and 13 represent the distribution of score and duration of the questions excluding Hashing for the above reason. The distribution is calculated based on the first trial of the question. It can be observed that Binary Searching is easier than the other two questions. The readers can also see Table 1 to find that Binary Searching program is significantly simpler. It can also be observed from Table 1 that Heap Sort is shorter than Parenthesis Checking. The reason of the similar distribution of the two questions will be discussed in Section 6.2.

Table 5 Student score summary.

Student	Heap Sort	Binary Searching	Parenthesis Checking
A	81%	75%	80%
B	90%	83%	23%

Table 6 Comparison of right answer ratio.

Blank Type	Right Answer Ratio		
	Entire Class	Student A	Student B
Return statement	90.6%	80.0%	80.0%
Decrement	77.1%	100.0%	50.0%
Increment	74.2%	100.0%	50.0%
If/else statement	72.8%	84.6%	53.9%
Subroutine call	62.8%	100.0%	50.0%
Subroutine call & Variable assignment	61.0%	100.0%	20.0%
Variable assignment	57.9%	72.2%	83.3%
While statement	52.9%	100.0%	60.0%
If/else statement & Subroutine call	38.2%	0.0%	0.0%

6.2 Analysis of Each Blank

6.2.1 Analysis Based-on Blank Type

Pgtracer allows defining various types of blanks within a program and a trace table. We shall analyze the difficulty level of each blank type in this section. 9 types of blanks are defined within the programs of the evaluation experiment. Some of the blanks are defined as a combination of multiple blank types. They are summarized in Table 3. The blank types are sorted according to the descending order of the ratio of right answers. A blank becomes more difficult if the right answer ratio becomes lower and the average duration becomes longer.

It can be observed that the blanks of a return statement are easy among the blank types having more than 2 blanks. This is because that the blanks ask the return value and the return values are described in the comment of the routines.

We can observe that the blanks having multiple blank types tend to become more difficult. For example, a blank whose correct answer is “stack[top++] = x;” is a combination of variable assignment and increment. Some students answer “stack[top] = x;” and forget to increment the variable “top”.

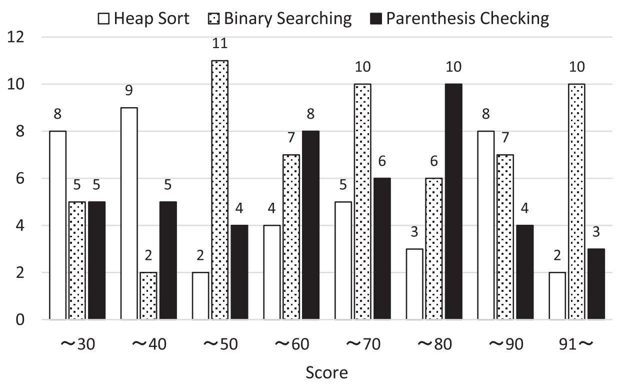


Fig. 12 Score distribution of three questions.

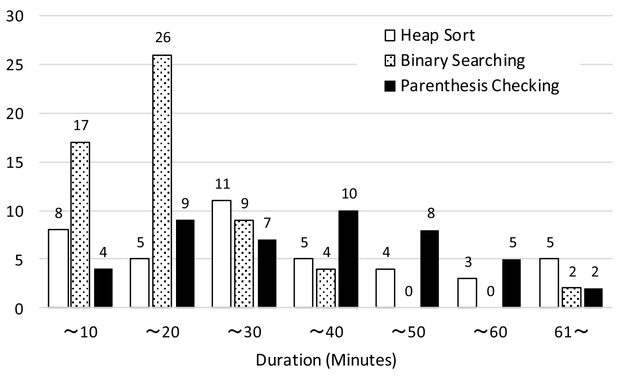


Fig. 13 Distribution of duration of three questions.

Many of the blanks of this type are contained in Heap Sort question. Heap Sort question also contains blanks of variable assignment and while statement whose right answer rates are low. These are the main reasons that the difficulty level of Heap Sort is similar to that of Parenthesis Checking.

6.2.2 Analysis of Student's Mistake

We list major mistakes of the students for the blank types having low right answer ratio in Table 4. We find that students made multiple mistakes within a blank. For such cases, we count each mistake independently. Major mistakes can be identified by observing the number of mistakes for each cause of the mistake.

We can observe that some students do not care about the return value of a function. More mistakes were found within assignment statements. Typical example is the assignment to an index of a heap. This is one of the reasons to increase difficulty level of the Heap Sort question. Pgtracer provides trace table so that a student can check the validity of his answer. But some students do not check consistency of their answer with the trace table. Possible reason for this is that the trace table becomes complex and that it is not easy for the student to understand which portion of the trace table is modified. Thus we have a plan to improve the appearance of the trace table such that the modified cells are clearly indicated by changing background color of the cell.

Another type of typical mistake is caused by misspelling and missing semicolon. Microsoft Visual Studio is used for C++ programming education in our department. Although such IDE is powerful, grammatical errors are automatically corrected so that students do not aware of slight mistakes.

6.3 Analysis of Student's Learning Process

We analyzed the answering processes of the students for two types of questions composed of multiple functions: Heap Sort and Parenthesis Checking. Most of the students fill the blanks in the order appeared in the program.

However we find that 5 students fill the blanks according to the execution order of statements containing the blanks for Parenthesis Checking. These students obtain high scores of 92%, 80%, 76% (two students) and 73% respectively. In case of Heap Sort, three students fill the blanks according to the execution order. These students also obtain 90%, 86% and 68%. We can conclude that a student who understands the execution order of a program can perform well at the experiment.

We also find students who start filling the blanks from the middle of the programs. These students typically cannot fill the blanks at the beginning of the program. The number of such students is 13 for Parenthesis Checking and 1 for Heap Sort. We observe that these students obtain low scores. The number of such students depends on the difficulty level of the question.

6.4 Achievement Analysis of Each Student

Achievement of each student can be analyzed by comparing the right answer rate of each blank type between the student and the entire class. We shall demonstrate an example to analyze the achievement of two students utilizing the analysis functions proposed in this paper. The achievement of the entire class is analyzed in Sections 6.1 to 6.3.

Table 5 represents scores of the two students.

Table 6 represents the ratio of right answer computed for each blank type. We omit the blank types in Table 3 such that the number of blanks is equal to 1. This is because that the blank types have small number of log records so that they are not suitable for statistical analysis.

The ratios which we focus on are represented using underline in Table 6.

Although student A generally marks high scores in all of the questions, he made three mistakes at the blanks containing a combination of if/else statement and subroutine call. Two of the three mistakes were caused by misspelling. He also made six mistakes caused by misspelling and missing semicolon among 13 mistakes he made. The student understands programming well but sometimes makes careless mistakes.

The right answer ratio of Student B is lower than the ratio of the entire class particularly for if/else statement and a combination of subroutine call and variable assignment. The mistakes are caused by the lack of understanding of comparison expression and by forgetting to save return values of a subroutine. He also left many unfilled blanks. This implies fundamental lack of understanding of the C++ language basics and the algorithm for parenthesis checking.

7. Evaluation of the Student Feedback Function

The analysis functions are also provided to the students after removing personal information of other students. We collectively call these functions as student feedback function. We utilize pg-

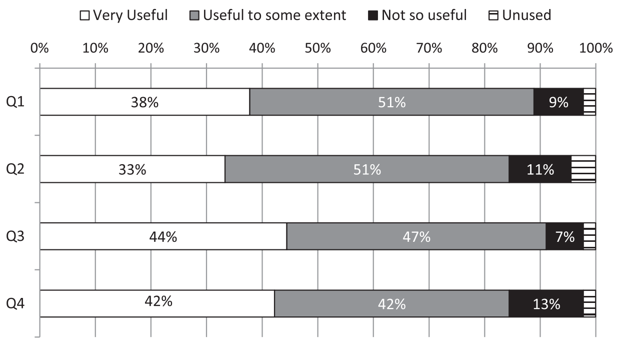


Fig. 14 Answer summary of each question.

tracer at a computer programming lecture at Kumamoto National College of Technology [13]. We performed the student survey for the 50 students taking the lecture. 45 students (90%) responded to the survey.

The survey contains four questions Q1 to Q4 defined below.

Q1: How useful is the score distribution function of each question?

Q2: How useful is the analysis function of blanks of a question?

Q3: How useful is the detailed analysis function of a blank of a question?

Q4: How do you feel is the overall usability of the student feedback function?

Each of the questions Q1 to Q3 corresponds to an analysis function illustrated in Fig. 7 to Fig. 9. We intend that usefulness of the student feedback function was evaluated through Q4.

Figure 14 represents the distribution of answers for the questions. From the viewpoint of avoiding bias, a strong negative option could be provided. But we did not provide such option since the students will choose the weak negative option “Not so useful” instead of the strong negative option under the Japanese culture.

The readers can observe from Fig. 14 that more than 80% of the students answer positively to all of the four questions. This indicates that the student feedback functions are well accepted among the students.

Particularly the positive answer ratio is around 90% for Q1 and Q3. The positive answer ratio for Q1 implies that most of the students are interested in the score distribution and their own achievement level analysis. We can also conclude from the answer to Q3 that most students are willing to understand their weak point by utilizing the detailed analysis function of each blank.

Compared with the above two functions, usefulness of the analysis function of blanks of a question is low from the answer to Q2. The reason of this can be observed that the amount of information provided by the analysis function is less than the two functions.

8. Conclusion

We proposed seven analysis functions to analyze student log collected by the programming education support tool pgtracer. The functions are provided both for teachers and students. We performed evaluation experiment of the proposed functions in order to demonstrate usefulness of the functions for both types of the users. The analysis result is useful for a teacher to understand the achievement level of the entire class and the individual student as demonstrated in Sections 5 and 6.

We also extend the analysis function to develop the student feedback function. The feedback function provides a student with the analysis result of himself and the entire class. Then the student can assess his own achievement by comparing with the achievement of the class. We received a positive feedback from most of the students as explained in Section 7.

From the viewpoint of systematic education of software development, education of requirement engineering and software design is also important especially for students willing to be IT professionals. We are also developing a series of education support tools for these phases. These tools and the evaluation result will be reported in our future publications.

Acknowledgments The authors are grateful to the students at Saga University and Kumamoto National Institute of Technology for their cooperation to the evaluation experiment. This research is supported by JSPS KAKENHI Grant Numbers 16K01022 and 17K01036.

References

[1] Kakeshita, T., Yanagita, R. and Ohta, K.: A programming education support tool pgtracer utilizing fill-in-the-blank questions: Overview and student functions, *Proc. 2nd Int. Conf. Education Reform and Modern Management* (*ERMM 2015*), Hong Kong, pp.164–167 (2015).

[2] Kakeshita, T., Ohta, K., Yanagita, R. and Ohtsuki, M.: A programming education support tool pgtracer utilizing fill-in-the-blank questions: Teacher functions, *Proc. 2nd Int. Conf. Education Reform and Modern Management* (*ERMM 2015*), Hong Kong, pp.168–171 (2015).

[3] Kakeshita, T., Yanagita, R. and Ohta, K.: Development and evaluation of programming education support tool pgtracer utilizing fill-in-the-blank question, *Journal of Information Processing: Computer and Education*, Vol.2, No.2, pp.20–36 (2016). (in Japanese)

[4] Nishida, T. et al.: Implementation and evaluation of PEN: The programming environment for novices, *Journal of Information Processing*, Vol.48, No.8, pp.2736–2747 (2007). (in Japanese)

[5] Hahul, R., Whitchurch, A. and Rao, M.: An open source graphical robot programming environment in introductory programming curriculum for undergraduates, *Proc. IEEE Int. Conf. MOOC, Innovation and Technology in Education* (*MITE*), pp.96–100 (2014).

[6] Funabiki, N., Korenaga, T., Nakanishi, T. and Watanabe, K.: An extension of fill-in-the-blank problem function in Java programming learning assistant system, *2013 IEEE Region 10 Humanitarian Technology Conference*, *R10-HTC 2013*, pp.85–90 (Aug. 2013).

[7] Deperlioglu, O. and Kose, U.: The effectiveness and experiences of blended learning approaches to computer programming education, *Computer Applications in Engineering Education*, Vol.21, No.2, pp.328–342 (June 2013).

[8] Malliarakis, C., Satratzemi, M. and Xinogalos, S.: Integrating learning analytics in an educational MMORPG for computer programming, *Proc. IEEE 14th Int. Conf. Advanced Learning Technologies*, pp.233–237 (2014).

[9] jQuery plugin: Tablesorter 2.0, available from ⟨http://tablesorter.com/docs/⟩ (accessed 2018-05-05).

[10] PHPExcel, available from ⟨https://github.com/PHPOffice/PHPExcel⟩ (accessed 2018-05-05).

[11] Asial Corporation: JpGraph, available from ⟨http://jpgraph.net/⟩ (accessed 2018-05-05).

[12] McConnell, S.: Code Complete: A Practical Handbook of Software Construction, 2nd Edition, Microsoft Press (2004).

[13] Murata, M. and Kakeshita, T.: Analysis method of student achievement level utilizing web-based programming education support tool pgtracer, *5th International Conference on Learning Technologies and Learning Environment* (*LTLE 2016*), Kumamoto, Japan, pp.316–321 (July 2016).

Tetsuro Kakeshita is an associate professor at Department of Information Science, Saga University, Japan. He received his Ph.D. degree in Computer Science from Kyushu University, Japan in 1989. His major research interests include quantitative analysis of ICT education and ICT certification, and complexity analysis of database and software systems. He received an excellent educator award from Information Processing Society of Japan (IPSJ) in 2013. He is a senior member of IPSJ.

Kosuke Ohta received his M.S. degree in computer science from Saga University, Japan in 2016. He joined the pg-tracer project and developed the analysis functions at Saga University. He also received an excellent student presentation award from IPSJ Computer and Education Workshop in March 2015. Currently he is working as an IT engineer for an IT company in Saga, Japan.