Chen, W. et al. (Eds.) (2017). Proceedings of the 25th International Conference on Computers in Education.
New Zealand: Asia-Pacific Society for Computers in Education

Algorithm Learning by Comparing Visualized Behavior of Programs

Daiki IHARA[a*], Satoru KOGURE[b] , Yasuhiro NOGUCHI[b],
Koichi YAMASHITA[c], Tatsuhiro KONISHI[b] & Yukihiro ITOH[d]
[a]*Graduate School of Informatics,Shizuoka University, Japan*
[b]*Faculty of Informatics,Shizuoka University, Japan*
[c]*Faculty of Business Administration, Tokoha University, Japan*
[d]*Shizuoka University, Japan*
*ihara.daiki.17@shizuoka.ac.jp

Abstract: It is said that visualizing the behavior of programs is an effective way for novice programmers to understand algorithms. Based on discussions with teachers of programming subjects, it was suggested that learners could deepen their understanding of algorithms by comparing visualizations of programs. This paper proposes learning processes to understand algorithms by comparing two figures. In addition, we develop a system that provides supporting functions for comparisons. Our experimental results show that the learning process using our systems is an effective way for novice programmers to understand algorithms.

Keywords: Program visualization, behavior comparison, algorithm understanding.

1. Introduction

Developing skills for understanding the behavior of programs is important in programming education. It is said that visualizing the behavior of programs is an effective way for novice programmers to understand algorithms. It is better to demonstrate the mechanism of the given algorithm in words and pictures than only in words (Gabor, 2009). Therefore, many program visualization tools have been developed (Rößling et al., 2002, Yamashita et al., 2016, Moreno et al., 2004, Fossati et al., 2008, Malmi et al., 2004). Rößling developed ANIMAL, which shows the behavior of programs based on the ANIMAL SCRIPT written by teachers. Teachers can achieve visualization at low cost by using preset sample scripts. Yamashita developed TEDViT (Teacher Explaining Design Visualization Tool), which has more distinctive features than existing program visualization tools. By using TEDViT, teachers can control the visualization of the unique concepts of algorithms (e.g., tree node and stack) and the behavior of programs. Teachers can present program visualization to their students based on their teaching design. If needed, teachers can use the figures from their classroom textbooks with TEDViT. The figures are called the Status of Target Worlds (STWs). To use TEDViT, teachers must write rule sets. The rule sets are then referred to as *drawing rules*. By using the tool, learners can trace the steps of programs with program visualizations.

Through discussion with teachers of programming, it was suggested that learners could deepen their understanding of algorithms by comparing the STWs of two scenes (scene means view of specific point in time). Through the discussion, we thought that learning by comparing scenes could be classified into three viewpoints: (A) Compare different time points, (B) Compare different types of data to be processed, and (C) Compare different programs. These details are described in section 2. Learning by comparing scenes is referred to as *Scenes Comparison Based Code Reading (SCBCR)*.

Above mentioned program visualization tools do not support such comparison of multiple scenes. Therefore, the aim of this work is to construct a system that supports SCBCR. We extend the current TEDViT to support SCBCR. The target audience of the system includes learners who do not attain an adequate understanding after a lecture class. It is difficult for such learners to find scenes that should be compared. Therefore, a teacher should be able to designate scenes suitable for SCBCR and to make messages that advise learners what they should focus on in SCBCR. We extend the existing drawing rules to enable teachers to realize SCBCR.

2. Fundamental Consideration

2.1 Classification of Comparison

Based on discussions with teachers of programming subjects, we classified SCBCR into three viewpoints based on comparison targets. In this section, we describe SCBCR in detail and discuss the three viewpoints based on the comparison programs defined by teachers.

(A) Comparing Different Time Points (Viewpoint (A)): Learners compare scenes that correspond to two different points in time during the execution process. The learners understand the behavior of programs from the two scenes at different time points. As an example, in the process of a sort program, the learners can understand the step of sorting by comparing the scene of the array up to the nth element with the scene of the array up to the (n+1) th element.

(B) Comparing Different Types of Data to be Processed (Viewpoint (B)): Learners compare two scenes in the same program with different data. Through this comparison, learners understand the program operation independent of data, and the difference in execution efficiency due to the difference in data. As an example, in a sort program, the learners compare the execution process of sorted data with data in reverse order. This allows them to observe the difference in the sorting process with different data.

(C) Comparing Different Programs (Viewpoint (C)): Learners compare scenes from the execution of two different programs. This type can be further classified into the following three categories:

- **Programs with the same purpose but different algorithms**: Learners can understand the difference in execution efficiency due to different algorithms.
- **Programs with the same algorithm but different implementations**: Learners can understand the difference in execution efficiency due to different implementations.
- **Correct program, and program that have bugs**: Learners focus on the behavior of the buggy programs compared with the behavior of the correct programs. This comparison leads learners to a better understanding of the behavior of the correct programs.

2.2 Required Functions for Our Proposed System

The existing TEDViT supports the visualization of only one program. To realize SCBCR, TEDViT requires a new interface design and configurable rules for comparing two different scenes. In this paper, these configurable rules are referred to as *rules for supporting SCBCR*.

In SCBCR, the learners compare two scenes, so the system needs to show two scenes. In SCBCR with viewpoint A, there is one execution process. In SCBCR with viewpoints (B) and (C), there are two execution processes. We develop an SCBCR system for viewpoint (A) and an SCBCR system for viewpoints (B) and (C). In SCBCR, learners sometimes compare two scenes in single execution process. The interface supporting SCBCR should show the two scenes in sync to the step of the program. The learners can easily step forward and backward through the scenes. In this paper, the function is referred to as *the step observation function*.

Teachers should be able to configure rules for supporting SCBCR for the comparison of two scenes. To realize an effective comparison, two types of rules are needed. First, teachers set the scenes to be compared by learners. In this paper, the first rule type is referred to as *saving scenes rules*. Second, teachers set up comparison viewpoints with two scenes and add an advisory message. In this paper, the second rule type is referred to as *design comparison rules*. To design these rules, we analyzed some algorithms that were used in programming lectures.

3. Design of Systems and Rules for Supporting SCBCR

3.1 Proposed System for SCBCR with Viewpoint (A)

Figure 1 is the system overview for SCBCR with viewpoint (A). It consists of five areas.

Program code area (Area 1): The system shows the program code in this area. The system emphasizes a statement whose scenes are shown in the system. The system supports two scenes identified by red

and blue. Two scenes corresponding with a statement are highlighted by color gradation from red to blue.

Buttons area (Area 2): The system creates buttons for operating the system. Learners proceed with SCBCR by using these buttons. The system normally shows two scenes defined by rules for supporting SCBCR. If learners would like to check the steps between the two scenes, they use the change mode button. The system then shows the first scene and the scene in the next step.

Scene area (emphasized by red or blue) (Area 3 and 4): Area 3 and 4 shows the scene corresponding with the statement highlighted in red or blue in area 1.

Message area (Area 5): The system shows messages defined by design comparison rules.

The scenes in Figure 1 are ones comparing the nth with the (n+1) th laps of a for-loop that searches for a character string in a longer character string by using the KMP algorithm. Learners compare two scenes in the execution process to grasp how the state changes in one lap of the for-loop.

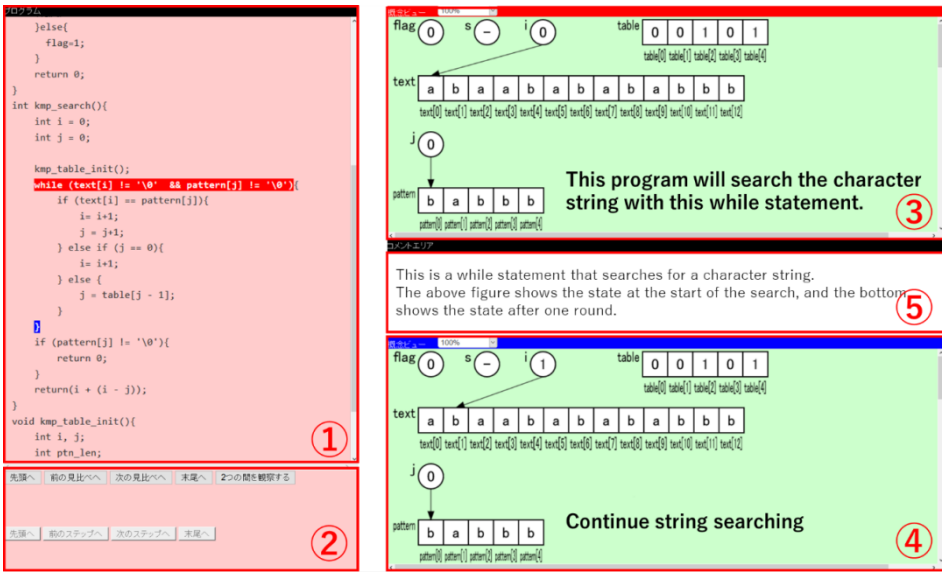

Figure 1. An overview of the system for viewpoint (A).

3.2 Proposed System for SCBCR with Viewpoints (B) and (C)

Figure 2 is the system overview for SCBCR with viewpoints (B) and (C). It consists of eight areas.

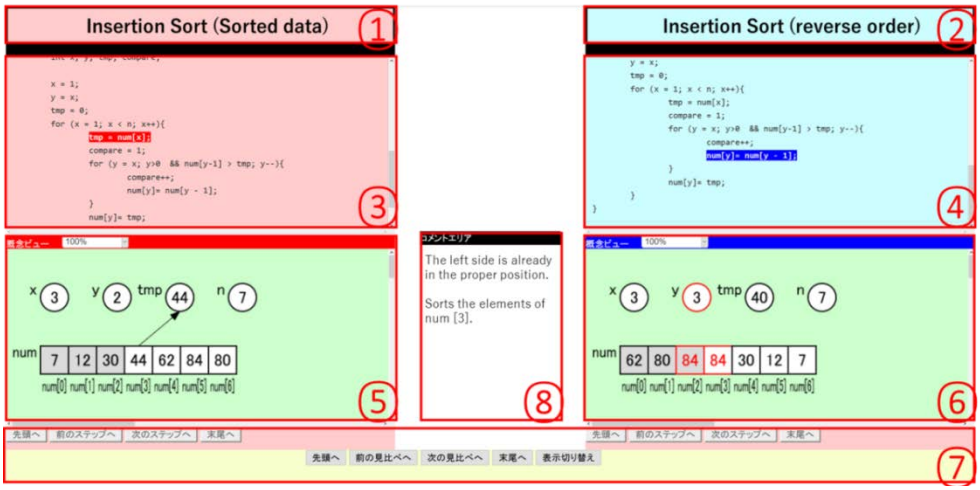

Figure 2. An overview of the system for viewpoints (B) and (C).

In the SCBCR with viewpoint (B) and (C), there are two execution processes. In this system, one process is shown in area 1, area 3, and area 5 on the left side. Another process is shown in area 2, area 4, and area 6 on the right side.

Program title area (areas 1, 2): The system shows the titles explaining the execution processes.
Program code area (areas 3, 4): Areas 3 and 4 show program code to be compared. Areas 5 and 6 show the scenes after execution of the highlighted statements in areas 3 and 4 respectively.
Scene area (highlighted by red or blue) (area 5 and 6): Areas 5 and 6 show the scene corresponding with a statement in areas 3 and 4, respectively.
Buttons area (area 7): The system prepares buttons for operating the system.
Message area (area 8): The system shows a message defined by design comparison rules.

The scenes in Figure 2 are visualizing the comparison of two execution processes of an insertion sort program that has been given different data sets. The process on the left side is given sorted data. The process on the right side is given data in reverse order. Suppose a learner compares the two processes at beginning of the second lap of outer loop: In the left process, the sorting from the first element to the third has been completed. On the right side, the sorting of these three elements has not finished. If the learner proceeds with the next comparison, areas 3 and 5 are fixed, and areas 4 and 6 reproduce the sorting from the first element to the third until the sorting of the three elements is complete. By this observation, the learners understand that it takes more time to sort data in reverse order than it does to sort ordered data.

3.3 Proposed Rules for Supporting SCBCR

To save scenes which should be compared by learners, teachers write saving scenes rules. In the rules, teachers can tag the scene with a unique label name. To designate scenes for comparison and messages to advise what should be focused on in SCBCR, teachers write design comparison rules. In these rules, teachers specify tags for the scenes to be compared, and compose the messages. We have to omit these description methods for want of space.

4. Experimental Evaluation

We conducted an experiment to confirm the effectiveness of our two support systems. The hypotheses of the experiment are that our system for SCBCR is useful for learners as following:

- Learners can cultivate better understanding by SCBCR supported by our system.
- Learner's understanding of processes is related to comparison scenes.
- Learners perform a long and thoughtful observation of comparison scenes.
- There is no significant problem in using SCBCR with our system's interface and the messages.

4.1 Methods of the Experiment

We recruited experiment participants who had C language knowledge and had attended programming lectures. Seven university students participated in the experiment. Six of them were in the faculty of informatics and one was in the faculty of engineering. They had a half-year or more of experience attending C language programming lectures. We employed four algorithms for learning: the KMP algorithm, the insertion sort, the bubble sort, and the selection sort. The informatics participants had knowledge of all four algorithms, and the participant of engineering had knowledge only of bubble sort. In this experiment, the participants proceeded with three types of SCBCR.
SCBCR 1 (the KMP algorithm with viewpoint (A).): This includes two points we want learners to understand. The first is that to process the contents of a for-loop is to proceed with string search by changing the position of interest in the search target strings and pattern character strings. The second is that the focusing position of the pattern character string may change according to the table.
SCBCR 2 (the insertion sort algorithm with viewpoint (B).): This includes the content that we want learners to understand; that is, in the case of reverse order data, the calculation time required for sorting is increased compared with sorted data.
SCBCR 3 (the bubble sort and the selection sort algorithms with viewpoint (C).): This includes the content we want learners to understand is that although the number of comparisons between selection sort and bubble sort is the same, the number of exchanges in selection sort is less than in bubble sort. Therefore, selection sort is more efficient.

We conducted the experiment using the following procedure.

(1) Answer pre-test questions for SCBCR1, 2 and 3.
(2) Receive an explanation on how to use the system for SCBCR with viewpoint (A).
(3) Receive an explanation on SCBCR 1 with viewpoint (A) and proceed with that
(4) Answer post-test questions for SCBCR 1 with viewpoint (A) and operation test for SCBCR 1.
(5) Receive an explanation for how to use the system for SCBCR with viewpoints (B) and (C)
(6) Receive an explanation for SCBCR 2 with viewpoint (B) and proceed with that.
(7) Answer post-test questions for SCBCR 2.
(8) Receive an explanation for SCBCR 3 with viewpoint (C) and proceed with that.
(9) Answer post-test questions for SCBCR 3.
(10) Answer the systems questionnaire for SCBCR and messages for comparison.

The pre-test in (1) consists of three questions; the first one is about the KMP algorithms, the second one is about the insertion sort, and the third one is about the bubble sort and the selection sort. The post-test consists of the same three questions as the pre-test. The explanation for SCBCR 1, 2, and 3 does not contain the answers for the pre-test and post-test. In (4), we give participants an operation test. In the operation test, the participant answers to supervisor with pairs of scenes that perform specific processes indicated by the question. The questionnaire in (10) consists of seven questions about the evaluation of the systems interface and comparison messages. The questionnaire items are evaluated on a five-point grading scale, with a higher number indicating a better rating.

After experiment, we compare answers for the pre-test and the post-test. We analyze whether there is an improvement in the answers about SCBCR, including what we want learners to understand. In the case that there is such an improvement, we conclude that SCBCR has shown learning effects. If the participant answers with the correct pairs of scenes in the operation test, we can evaluate the learner's understanding related to the comparison scenes shown by the system.

4.2 Results of the Experiment

4.2.1 Analysis of Test Results

We compare the answers of the pre-test and post-test. Table 1 shows scores of the three questions. The values in the column are the number of informatics students, engineering student.

For the KMP algorithm, we give participants an operation test. The operation test has three correct answers. All of informatics participants gave two or more correct answers (Three participants obtained full marks). In addition, the engineering participant also obtained full marks. The results suggest that the learner's understanding of process is related to the comparison scenes shown by the system. The facts on Table 1 and operation test suggest that SCBCR with viewpoint (A) has some learning effect. Also the facts on Table 1 suggest that SCBCR with viewpoint (B) and (C) have some learning effects.

From the above results, it can be concluded that SCBCR with our systems has some learning effects. In addition, since SCBCR is also effective for a participant of engineering, it suggests that SCBCR with our systems is effective for various knowledge levels of learners.

Table 1: Scores of the three questions (Informatics students, Engineering student).

	KMP algorithm	Insertion sort	Selection sort/bubble sort
correctly answered in the pre-test	2, 0	4, 0	0, 0
improved their answer in the post-test	4, 1	2, 1	6, 1
did not improve their answer	0, 0	0, 0	0, 0

4.2.2 Analysis of Operation Logs

The system for SCBCR collects operation logs of learners. We define irregular comparison behavior in SCBCR as a learner proceeding with successive comparisons in a short time in comparison scenes that the learner has not observed yet. We searched for irregular comparison behavior in the operation log of the learners. In SCBCR with viewpoint (A) and (C), no one showed irregular behavior. In SCBCR with

viewpoint (B), three informatics students forwarded comparisons continuously, with less than one second between comparisons in certain circumstances. We guess they do not need to observe for a long time because they understand the target algorithm well. These facts suggest that the system almost be used as we intended.

4.2.3 Analysis of Questionnaire Answers

Table 2 shows the results of the questionnaire to evaluate educational effectiveness of the system. As for (A), the results suggest that the participants evaluated the system highly. As for (B) and (C), the first question suggests that the participants evaluate highly, but the evaluation with viewpoints (B) and (C) is lower than (A) in the second and third questions. For messages from the system, the participants' evaluation was high with an average of 4.00. One participant commented that it was easy to proceed with SCBCR because messages of comparison were brief and easy to understand. These results suggest that there is no significant problem in our system's interface and messages.

From the results in 4.2, we conclude that SCBCR is effective for learners to understand algorithms and that our systems can support learners for SCBCR.

Table 2: Results of the questionnaire.

	(A)	(B), (C)
How easy did you feel to understand the content of the component elements on the interface?	4.14	4.14
How intuitive did you feel arrangement of component elements on the interface was?	4.14	3.43
How easy did you feel the system's operation?	4.43	3.71

5. Conclusion

In this paper, we proposed comparison learning processes called SCBCR. We developed systems for SCBCR by extending TEDViT. The results of our experiments suggest that our systems for SCBCR are effective in helping learners understand algorithms. Going forward, we will improve the system based on the results of a questionnaire and the opinions of teachers.

Acknowledgements

This research was supported by the Japanese Grant-in-Aid for Scientific Research (B) Grant Number JP24300282, (C) Grant Number JP16K01084 and for Young Scientists (B) Grant Number JP15K16104.

References

Yamashita, K., Fujioka, R., Kogure, S., Noguchi, Y., Konishi & T., Itoh, Y. (2016). Learning Support System for Visualizing Memory Image and Target Domain World, and Classroom Practice for Understanding Pointers. *Proceedings of the 24rd International Conference on Computers in Education*, 521-530.

Rößling, G., Freisleben, B. (2002). ANIMAL: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages & Computing, 13(3),* 341-354.

Gabor, T., (2009). Algorithm visualization in programming education. *Journal of Applied Multimedia. 4(3),* 68-80.

Moreno, N., Sutinen, M. E. (2004). Visualizing programs with Jeliot 3. *AVI 04: Proceedings of the Working Conference on Advanced Visual Interfaces*, 373-376.

Fossati, D., Eugenio, B. D., Brown, C., & Ohlsson, S. (2008). Learning linked lists: experiments with the iList system. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* 80-89.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppala, O. & Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education, 3(2),* 267-288.