Ogata, H. et al. (Eds.) (2015). Proceedings of the 23rd International Conference on Computers in Education. China: Asia-Pacific Society for Computers in Education

Educational Practice of Algorithm using Learning Support System with Visualization of Program Behavior

Koichi YAMASHITA[a*], Ryota FUJIOKA[b], Satoru KOGURE[c], Yasuhiro NOGUCHI[c], Tatsuhiro KONISHI[c] & Yukihiro Itoh[e]
[a]*Faculty of Business Administration, Tokoha University, Japan*
[b]*Graduate School of Informatics, Shizuoka University, Japan*
[c]*Faculty of Informatics, Shizuoka University, Japan*
[e]*Shizuoka University, Japan*
*yamasita@hm.tokoha-u.ac.jp

Abstract: In this paper, we describe the educational practices of algorithm using learning support system with visualization of program behavior. The systems visualizing behavior of program have certain effect to understand algorithm behavior. Introducing them into a classroom of algorithm is expected to allow learners to cultivate better understanding. However, almost all of the systems cannot incorporate the teacher's intent of instruction that may be chose to suit the learners among several instructions. Moreover, the teacher needs to devise the contents of class when the teacher want to teach the property of algorithms with those systems, such as the number of comparisons or swaps. Based on these considerations, we conducted classroom practices of algorithm incorporating the system that we developed in our previous work. Our system visualizes the target domain world according to the visualization policy defined by the teacher. In addition, we included the contents based on discovery learning about the properties of algorithm in the practices. In this paper, we describe the overview of our educational practices and the reactions of learners. Furthermore, we show that the framework in our practices can be established in algorithm classes.

Keywords: Algorithm education, learning support system, domain world model, classroom practice, discovery learning

1. Introduction

As the computer education is recognized to be one of the fundamental science educations, the range of educational opportunities of algorithm available to students has been expanded. A growing number of university holds course of algorithm for non-computer science major, because algorithm education is expected to foster logical thinking and application skills of formulating general problem-solving.

Generally, a course of algorithm is held in the form of classroom lecture, and corresponding course of programming is also held in the form of exercise to confirm and establish a better understanding of the lecture. The learning by being told in the lecture and establishing understanding by the exercise construct a learning cycle. Learners can understand algorithms that is the essence of problem-solving and program-code that is the formal representation of the algorithm externalization in this cycle. However, according to our classroom experience, not a few learners cannot accommodate the learning style of this cycle and reach an impasse. We consider that this is because they would proceed to the programming exercise without sufficient understanding of the algorithm.

So far, several learning support systems are developed to support novice learners to understand several algorithms (Malmi, et al., 2004; Fossati, Eugenio, Brown, & Ohlsson, 2008; Rajala, Laakso, Kaila, & Salakoski, 2008; Ben-Ari, et al., 2011; Neve, Hunter, Livingstone, & Orwell, 2012; Yamashita, et al., 2014). These systems visualize processing objects of program-code and algorithm (i.e. target domain world) and reproduce the behavior of the code and algorithm. Introducing these systems into classes is expected to allow learners to cultivate a better understanding of algorithm (Robins, Rountree, & Rountree, 2003; Pears, et al., 2007).

However, with the range of educational opportunities being expanded, it has been required increasingly that teacher teaches various learners with various background knowledge. Depending on the learners, teacher must fix the content or intent of instructions such as point where learners should focus in the algorithm, abstraction or generalization degree of instruction, and so on. Almost all of the existing systems disallow these variation of teacher's intent, and visualize target domain world in the fixed visualization policy. Moreover, although existing systems tend to focus on reproducing entire flow of algorithm behavior, knowledge related to the algorithm is also important in algorithm education. For example, the property of algorithm such as the number of comparisons or swaps in the sorting tasks is one of the important learning target.

In this paper, we describe our classroom practices of algorithm, based on these consideration. We introduced the learning support system developed in our previous work (Kogure, et al., 2014) into the class. Our system visualizes the target domain world according to the visualization policy defined by the teacher. In addition, the contents of our practice includes not only learning about entire flow of algorithm behavior using our system, but also discovery learning about property of algorithm. The reaction of learners suggests that the framework in our practice can be established in algorithm classes. Furthermore, the deliverables of learners suggest that the understanding of algorithm property can contribute the understanding of entire algorithm.

2. Previous Work

In an understanding of algorithm and program, it is important to have an image about their behavior. However, novice learners often find difficulty obtaining it with correctness. Although several systems that visualize behaviors of programs have been developed thus far, teachers cannot define the visualization policy with those systems.

For example, a teacher may draw an array object in horizontal layout when the instruction target is sorting of the array (as in Figure 1), whereas the teacher may draw it in vertical layout for a stack. The change in visualization policy like this is derived by fitting instruction contents to the learner's background knowledge. For example, if the learners sufficiently understand a stack, either object in horizontal or vertical layout will be acceptable to them. Similarly, teacher will not need to draw the temporary variable in a task that swaps values of two variables for non-novice learners. Therefore, we developed the environment for teachers to define the policy of drawing the status of target domain world according to their own intent. The system we developed reproduces behavior of program based on the definition.

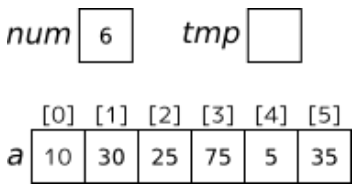

Figure 1. An example of a status of target world.

The drawing policy is defined by a set of rules consisting of object type, its attributes, drawing operation, and condition to operate. Object's attributes include its position, size, color, and so on. Drawing operation is chosen by one from create, delete, and update. Teacher describes a set of rules into the configuration file, and then our system reads it, interprets the drawing policy, and visualizes the target domain world according to it. The relationship among teacher, learner, and our system is shown by Figure 2. Teacher with some experiences of rule description can complete to define the entire drawing policy in almost same time as making slides using presentation software.

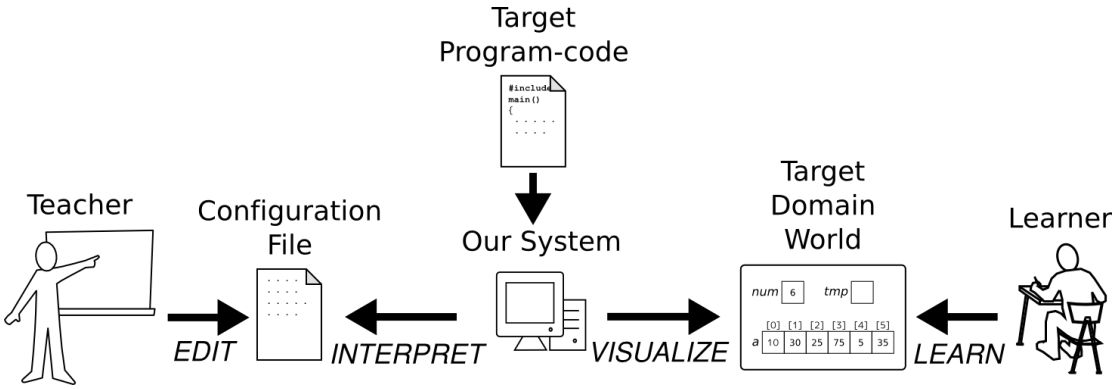

Figure 2. Relationship among teacher, learner, and our system.

Table 1 provides types of object, drawing operations, and attributes available for configuration of domain world status with our system. Circle, square, and rectangle object are mainly used to express directly value of a variable in program-code. Table object mainly expresses an array. Connector and line object express a relationship between two objects, and label and balloon object are used to describe a behavior of program or a role of object in natural language.

Table 1: Types of object, drawing operations, and attributes for configuration.

Objects	Operations	Attributes
circle	create	corresponded variable*
square	delete	main object ID**
rectangle	update	position
table		width
connector		height
line		color
label		line weight
balloon		line style

* Only for circle, square, rectangle and table object.
** Only for connector, line and balloon object.

Teacher can describe a condition to operate with referencing the statement number (statement ID) or variables in program-code, such as "when a certain statement is executed", "when the value of a certain variable satisfied the condition", and so on. A condition can be expressed with six types of comparison operator, ==, !=, >=, <=, <, and >, and three types of operand, immediate number, variable in program-code, and statement ID.

Figure 3 provides an overview of the environment generated by our system. Our system visualizes the target domain world in (C) according to teacher's configuration, and reproduces the behavior of program-code in (A). The world visualized in (C) corresponds to the status of target domain world after the execution of the statement highlighted in (A). When the learner clicks "Next" or "Prev" button, the highlight moves to the next or previous statement in the program-code and corresponding status of target domain world is also visualized in (C). The system simulates to execute the statements step by step, so that the learner understands the program behavior by observing the changes of target domain world. Moreover, if there are some connector objects or descriptions with label and balloon objects reflecting instructions in a class, they will assist the learner to understand the program behavior. The field (B) represents a status of memory image, which is implemented for the learning target that needs to reference to the main memory, such as pointer.

3. Classroom Practice

In this section, we describe two classroom practices with our system described in the previous section. First of them was for the fundamental sorting algorithms and was incorporated into the actual classes. Second was for the search algorithms and was conducted out of the actual classes. In both practices, we planned learning without referencing to memory image in (B). The participants referred the field (A) as the formal expression of the algorithm, and we omitted the details of the syntax. The participants focused on the observations on field (C) to understand behavior of algorithms.

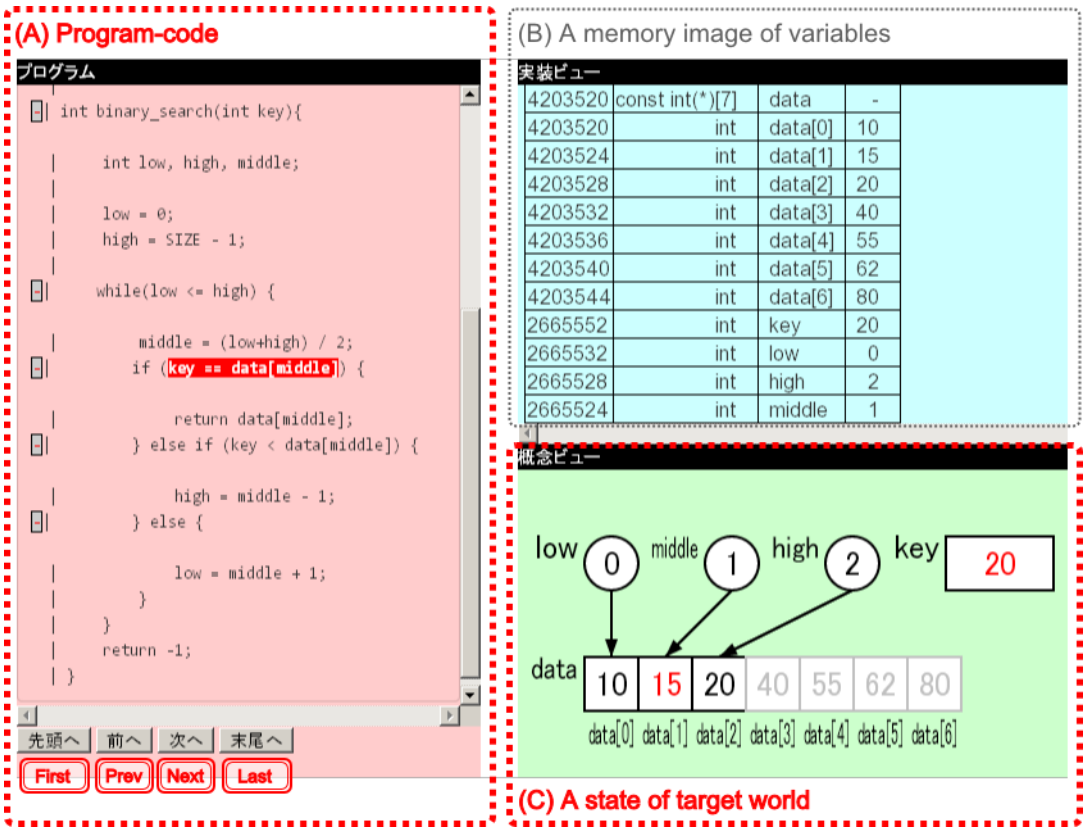

Figure 3. An example of a status of target world.

3.1 Practice about Sorting Algorithms

In the faculty that one of the authors belongs to, the lecture course "Algorithm" is held for third year student in business administration major. We incorporated two classes with our system mentioned in the previous section into the course. The contents of the practiced classes were three fundamental sorting algorithms that are selection sort, insertion sort, and bubble sort. The goal of the classes was to understand the behaviors of three sorting algorithms and to understand the differences among them based on the number of comparisons and swaps. We planned to achieve the former goal by observing the changes of target domain world using our system, and the latter by the tasks based on the discovery learning that find the maximum and minimum number of comparisons and swaps. There were 24 participants in this practice, all of whom were business administration major, 21 years old, and had less than a year's experience in programming.

In the first class of this practice, the teacher who regularly taught the course lectured on three sorting algorithm for 75-minute. At the end of the class, we conducted a 15-minute pre-test to evaluate the learners' understanding of three sorting algorithm before using our system. The pre-test asked the values of variables obtained from tracing the algorithms. In the second class conducted after a week, the teacher explained how to operate our learning environment and how to observe target domain world in the first ten minutes. After that, we allowed learners to learn each of sorting algorithms with our environment in the following step.

1. The learners observe the changes of target domain world all over the program-code and confirm the behavior of the algorithm.

2. The teacher makes the learners aware of the executions of comparison and swap operations, and suggests that the number of these operations could be varied according to the initial order of the array.
3. The learners input the initial order of the array to our environment, and confirm the change of the number of comparisons and swaps.
4. The teacher makes the learners find the initial order of the array that maximizes and minimizes each number of comparisons and swaps.
5. The teacher inquires of the learners their discovery of initial array order and explains the property of target algorithm based on learners' replies.

The learning order of three algorithms can be arbitrary, while we gave the environments in the order of bubble sort, insertion sort, and selection sort in this practice. We omitted step 2 and 3 above in the learning of insertion and selection sort. Step 4 is the task based on the discovery learning, hence we made the learners tackle it in all three algorithms. We incorporate the input process in the target program-code for realizing step 3 and 4, so that the learners can input the initial order of the array in tracing the program. After the 60-minute entire learning of three algorithms, we conducted a 15-minute post-test to evaluate the learners' understanding of three sorting algorithm after using our system. The contents of the post-test were the tasks tracing the same algorithms as pre-test except for the target array. After the entire practice, we conducted the brief questionnaire survey.

The intents of instructions included to our environment by the teacher were following:

- To suggest the role of the variables to learners by drawing arrowed connector from the variables indexing the array to corresponding array elements.
- To clarify the difference between sorted and unsorted elements by coloring sorted elements only.
- To clarify the difference of role by drawing circle objects for the index variables, where rectangle object for the temporary variable in swaps.
- To suggest the viewpoints in discovery learning by notifying the execution of the comparison and swap with label objects.

All of these intents of instructions were suggestive rather than codified, because we intended to make learners focus on the number of comparisons and swaps by themselves in step 4. Figure 4 provides an example of the status of target domain world visualized by our system.

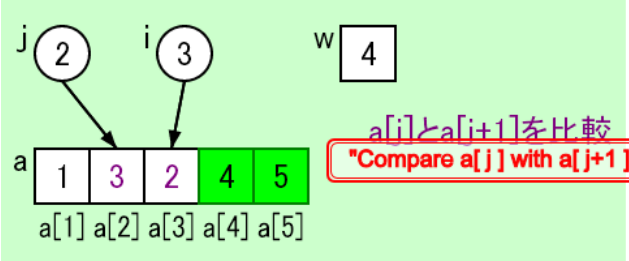

Figure 4. An example of the status of target domain world in sorting algorithm.

Table 2 presents the scores of pre- and post-test, grading both tests worth 100 points. Note that there are no participants with post-test score lower than pre-test score. The participants with no difference in scores are the one who got full scores and ones who got zero scores only. Those who had pre-test scores greater than zero grew the score of 22.1 points in average.

Table 2: Pre- and post-test scores of each participants in the practice about sorting algorithms.

learner	pre-test	post-test	difference	learner	pre-test	post-test	difference
1	10.9	42.9	31.9	13	0.0	5.4	5.4
2	35.9	69.6	33.7	14	0.0	0.0	0.0
3	42.2	58.9	16.7	15	0.0	0.0	0.0
4	0.0	0.0	0.0	16	50.0	71.4	21.4
5	37.5	41.1	3.6	17	39.1	100.0	60.9
6	64.1	80.4	16.3	18	34.4	50.0	15.6
7	100.0	100.0	0.0	19	78.1	98.2	20.1
8	0.0	0.0	0.0	20	0.0	0.0	0.0
9	15.6	35.7	20.1	21	21.9	75.0	53.1
10	0.0	0.0	0.0	22	0.0	0.0	0.0
11	39.7	42.9	13.2	23	31.3	35.7	4.5
12	0.0	0.0	0.0	24	40.6	55.4	14.7
				average	25.5	40.1	13.8

3.2 Practice about Search Algorithms

We conducted another practice for fundamental search algorithms out of the actual classes. Search algorithms had not been introduced in the actual classes of algorithm because of the shortage of time. The contents of the class were three search algorithms, sequential search, binary search, and hash search. The goal of the class was to understand the behaviors of three search algorithms and to understand the differences among them based on the search key and the search target. We planned to achieve the goal by observing the changes of target domain world using our system and by the tasks based on the discovery learning. The tasks assessed on learners were to find the maximum number of comparisons, the minimum number of them, and the synonym records. There were four participants in this practice, all of whom were business administration major, 21 years old, and had less than a year's experience in programming. All of them had participated in the practice described in the previous subsection and unlearned in search algorithms.

The practiced class consisted of 120 minutes. The teacher explained first how to operate our learning environment and how to observe target domain world, and then conducted the class in following steps:

1. The teacher gives the definition of search problem to the learners. The teacher explains that the class focuses on searching from array rather than file, and makes the learners think about search algorithm off the top of their head.
2. The teacher introduces the basic idea of sequential search algorithm as one of the simplest solution, and allows the learners to observe its behavior using our environment.
3. The teacher makes the learners input some search keys and observe the change of the number of comparisons, and then makes the learners find the search key that maximizes and minimizes the number of comparisons.
4. The teacher inquires of the learners their discovery of search key and explains consequent property of sequential search algorithm, naïve but inefficient, based on their replies.
5. The teacher mentions that there are more search algorithms, introduces the basic idea of binary search algorithm, and allows the learners to observe its behavior using our environment.
6. The teacher makes the learners input some search keys, observe the change of behaviors, and find the search key that maximizes and minimizes the number of comparisons.
7. The teacher inquires of the learners their discovery of search key and explains consequent property of binary search algorithm, fast but requires the search space to be sorted, based on their replies.
8. The teacher introduces the basic idea of hash search as yet another algorithm, and allows the learners to observe the behavior of hash search from the dataset with no synonyms using our environment.
9. The teacher makes the learners observe behavior of generating hash table using a certain dataset and observe occurrence of collisions. Here, the collision resolution of the algorithm used in the practice is based on open addressing.

10. The teacher inquires of the learners why collisions are occurred and how does the searching process change. Then, the teacher makes the learners find the dataset and search keys that trigger off collisions.
11. The teacher inquires of the learners their discovery and things they realized, and explains the property related to hash search such as hash function requirements and size of hash tables, based on learners' replies.
12. The teacher makes the learners observe behavior of hash search from the dataset with synonyms using our environment.

The intents of instructions included to our environment by the teacher were following:

- To suggest the role of the variables to learners by drawing arrowed connector from the variables indexing the array to corresponding array elements.
- To clarify the difference of role by drawing circle objects for the index variables, where rectangle object for the search key.
- To clarify the difference between search range and other array elements by coloring only out-of-range elements in the sequential and binary search algorithm.
- To suggest the viewpoints in discovery learning by coloring two objects that are the target of comparison process.

We intended to support learners' discovery by suggestive visualizations similarly as in the practice described in the previous subsection. The tasks based on discovery learning are included in step 3, 6, and 10. Figure 5 provides an example of the status of target domain world in binary search algorithm visualized by our system. The search key is stored in the variable *k*. The search range is presented by coloring the array elements out of search range gray. The comparison between two objects is presented by coloring target of comparison process, 93204 and 93309 with red. These visualization suggests the objects where the learners should focus on and support the learners' discovery learning.

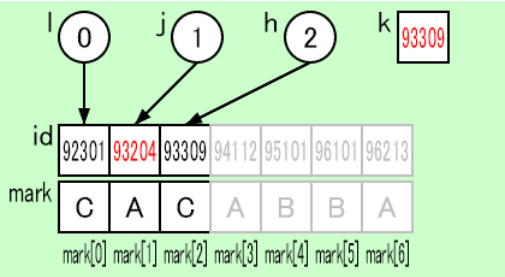

Figure 5. An example of the status of target domain world in binary search algorithm.

After the practice, we made the learners write a briefing paper on search algorithms with their behavior and property as post-test in 15-minute. We examined their contents from the viewpoints of whether the learners described the three algorithms and its property learned in this practice. Consequently, we found that while the learners had described all three algorithms and had explained roughly each behavior and property, they tended to assign their description with informal terms. We consider that the learners grasped the behavior and property of the algorithms but did not fix the term, because this practice did not include a lecture instruction.

4. Discussion

The learning support systems visualizing behavior of an algorithm or a program support learner to understand the structure of target domain world and principles to control behavior in the world by the system's visualization. In many existing systems, an action to the domain world is implemented by the target program-code or algorithm rather than users. User's GUI operations are used mainly to observe changes in the domain world. In the classroom practices with our system, we implemented an action to the domain world by setting initial state of the world. Responding the inquiries posed by the teacher such as what state does maximize or minimize the number of comparisons or swaps, and what state does

trigger off collisions, the learners generate hypothesis on initial state, input the state, operate our environment to test the hypothesis, and modify the hypothesis as necessary according to observed information. These tasks constitute the cycle of discovery learning without coding.

During the learning in coding exercise or the learning with learning support system, the learners develop the learning activity individually, and hence they would focus on irregular and inconsistent points for discovery. Our system provides not only objects consisted of target domain world, but also suggestive viewpoints to reduce the irregularity and inconsistency. We included the intents of instruction in visualized domain world not to explain the role of each object or the behavior of algorithm, rather to suggest points where the learners should focus on. In our classes practiced, the participants replied actively their discovery to the teacher. Therefore, we consider that our visualization worked in discovery learning with a certain effect.

We intended the discovery of algorithm property rather than algorithm itself because of the goal of the actual course of algorithm. The course is expected to provide case learning of formulating general problem-solving rather than the fundamental skills of program design. Hence the teacher has made much account of teaching the knowledge about the existing algorithms. The evaluation results in the first practice based on pre- and post-test suggests that the learners cultivated better understanding of sorting algorithms through the practice. The briefing papers written in the second practice suggest that the learners understood the behavior of search algorithm to some extent, while unsuccessful in terminology. We consider that the framework in our practices can be established in algorithm classes.

Our practices suggest that if the course of algorithm is hold as one of the fundamental science educations, the teacher could conduct the classes based on discovery learning by introducing appropriate learning support system. We summarize the requirements of learning support system appropriate for discovery learning as follows:

- System has function to design actions to target domain world and user-interface to apply the action other than coding.
- System has function to construct target domain world based on teacher's intent to suggest focusing points and support learner's discovery.

5. Conclusion

In this paper, we described the educational practices with learning support system developed in our previous work. The systems with visualization of program behavior support learners to understand algorithms, and hence introducing these systems is expected to allow learners to cultivate a better understanding. However, the existing systems tend to focus on reproducing entire flow of the algorithm behavior and devising the contents of classes is needed in teaching property of algorithm such as the number of comparisons or swaps. Furthermore, almost all of existing systems disallows the variation of teacher's intent of instruction that may be chose to suit the learners.

Based on these considerations, we introduced the system developed in our previous work. Our system visualizes the target domain world according to the visualization policy defined by teacher. We included the contents based on discovery learning about the properties of algorithm in our classroom practices. Teacher made learners externalize their discovery by inquiries and explained knowledge based on their externalized discovery. We could find the participants played an active role in learning activities in the practices. Moreover, we could find impression of the participants in the responses of survey conducted after the first practices, as "It is easy to understand the times when comparisons and swaps are occurred." That suggests our visualized intent of instruction guided appropriately the learners to discover in the learning cycle. Our evaluation results suggest that the learners cultivate better understanding of algorithm and that the framework in our practices can be established in algorithm classes.

One of the limitations of our framework is that it is difficult to encourage adoption of the terminology such as the name of algorithm. To teach knowledge as in lecture classes, the teacher need to balance the lecture and practice with the system. We plan to find ideal contents of classes by conducting classroom practices with our system continuously. Another future work is to evaluate more quantitatively the learning effects of our classroom practices. We will continue to investigate

quantitative evaluation of how much does understanding of algorithm property contribute to understand entire algorithm.

Acknowledgements

This study was supported by Japanese Grants-in-Aid for Scientific Research (B) 24300282 and for Young Scientists (B) 15K1610.

References

Ben-Ari, M., Bednarik, R., Ben-Bassat Levy, R., Ebel, G., Moreno, A., Myller, N., & Sutinen, E. (2011). A decade of research and development on program animation: The Jeliot experience. *Journal of Visual Languages & Computing, 22*(5), 375-384.

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.

Kogure, S., Fujioka, R., Noguchi, Y., Yamashita, K., Konishi, T., & Itoh, Y. (2014). Code reading environment according to visualizing both variable's memory image and target world's status. *Proceeding of the 22nd International Conference on Computers in Education*, 343-348.

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

Neve, P., Hunter, G., Livingstone, D., & Orwell, J. (2012). NoobLab: An intelligent learning environment for teaching programming. *Proceedings of the 2012 IEEE/WIC/ACM Joint Conferences on Web Intelligence and Intelligent Agent Technology, 3*, 357-361.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., . . . Paterson, J. (2007). A Survey of Literature on the Teaching of Introductory Programming. *ACM SIGCSE Bulletin, 39*(4), 204-223.

Rajala, T., Laakso, M.-J., Kaila, E., & Salakoski, T. (2008). Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. *Journal of Information Technology Education, 7*, 15-32.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.

Yamashita, K., Nagao, T., Kogure, S., Noguchi, Y., Konishi, T., & Itoh, Y. (2014). An educational practice using a code reading support environment for understanding nested loop. *Proceedings of 22nd International Conference on Computers in Education*, 848-857.