T. Hirashima et al. (Eds.) (2011). Proceedings of the 19th International Conference on Computers in Education. Chiang Mai, Thailand: Asia-Pacific Society for Computers in Education

# Practice and Effects of Algorithm Education through Manual Procedures

**Junko SHINKAI**[a*], **Isao MIYAJI**[b]
[a] *Toyama National College of Technology, Japan*
[b] *Faculty of Informatics, Okayama University of Science, Japan*
*shinkai@nc-toyama.ac.jp

**Abstract:** In algorithm education, creating a program is as important as understanding the principle of an algorithm. Even if students understand the principle of an algorithm, it is not easy to build an algorithm for programming. Therefore, the authors tried empirical algorithm education with manual activities by using sort algorithms as learning materials, so that the students could understand the principles of algorithms and improve their capability building algorithm for programming. The authors will report the details and learning effect of the educational practice.

**Keywords:** algorithm education, programming, experience learning, learning effect

## Introduction

In algorithm education, teachers tend to put emphasis on presenting algorithms and making their students understand the principles. Many students, however, cannot create an algorithm for programming even after they understand the principle. It is caused that students cannot find regularity how to express, using array and control structure of sequence, selection and repetition to create a program. Therefore, in this research, the authors propose a new approach in algorithm education to solve the problem, to motivate students create algorithm by themselves and express in C language.

Among previous studies, Abe [1] visualized the process of a sort algorithm by using cards that showed the steps in the algorithm so that the students could sort them. Abe also developed a tool to show the interaction between the sort program and data to help students understand the program, but did not mention writing programs. Sugiura et al. [3] reported introductory education to improve the algorithm building capability, where algorithms created manually could be described as executable programs. They used “Kotodama on Squeak” as the programming language because the difference between the expression of an algorithm and that of a programming language was small. They did not use common programming languages, such as the C language. Namatame [2] reported evaluation through group learning using peer review in the lesson of programming.

This paper is aimed to improve algorithm ability of students. In this research, the authors instructed the students to manually sort data and create the C programming in groups. The authors make students to describe data flow with index of array in order to lower a hurdle of processing flow and algorithm expression. Through experiencing these operations, students are expected to develop their sense to find regularity of sequence, selection and repetition in processing flow. Furthermore, the authors can offer an educational circumstance not to give up a trial and error process to create algorithm through group learning. This paper describes the educational method and the learning effect after the educational practice.

## 1. Algorithm education with manual activities

Some previous studies show that learning algorithms with manual activities is effective in understanding them [1],[3]. Therefore, the authors told the students to develop a sorting method and manually sort data. The authors also told them to describe the process clearly by using array to prevent manual sorting from being different from sorting on a computer. These activities enabled the students to check their ideas through manual activities and create algorithms that could be realized on a computer. The following shows the procedure that the authors suggest, from manual sorting to algorithm building.

Step 1: Describing a basic idea

The students wrote the numbers on the five cards and sorted these cards in ascending order. They described a basic idea about the process in which a computer can perform sorting that can be done easily by a human.

Step 2: Listing procedure steps and changes of data

The students listed the steps in the basic idea of their operation. They moved the cards according to the procedure to check whether they could sort them in ascending order. Then they wrote down each change in the arrangement of the cards as the procedure proceeded.

Step 3: Describing the steps in the list using array

The students described the procedure in step 2 using the array as shown in Table 1, which shows a sorting procedure created by one of the groups.

Step 4: Creating an algorithm for programming

The students found possible basic processes, such as sequential, selective, and repetitive processes in the flow expressed with array in step 3 by paying attention to regular changes in the subscript. Then they created an algorithm for programming.

Step 5: Creating and executing a program

The students created a program according to the algorithm and performed data sorting on a computer.

Table 1: The flows of card data and operation

| Numbers of Cades | | | | | Procedures |
|---|---|---|---|---|---|
| 8 | 5 | 3 | 6 | 2 | Compare a[1] to a[2] .   a[1] > a[2] .   Change a[1] for a[2] . |
| 5 | 8 | 3 | 6 | 2 | Compare a[1] to a[3] .   a[1] > a[3] .   Change a[1] for a[3] . |
| 3 | 8 | 5 | 6 | 2 | Compare a[1] to a[4] .   a[1] < a[4] . |
| 3 | 8 | 5 | 6 | 2 | Compare a[1] to a[5].   a[1] > a[5] .   Change a[1] for a[5] . |
| 2 | 8 | 5 | 6 | 3 | Compare a[2] to a[3] .   a[2] > a[3] .   Change a[2] for a[3] . |
| 2 | 5 | 8 | 6 | 3 | Compare a[2] to a[4] .   a[2] < a[4] . |
| 2 | 5 | 8 | 6 | 3 | Compare a[2] to a[5] .   a[2] > a[5] .   Change a[2] for a[5] . |
| 2 | 3 | 8 | 6 | 5 | Compare a[3] to a[4] .   a[3] > a[4] .   Change a[3] for a[4] . |
| 2 | 3 | 6 | 8 | 5 | Compare a[3] to a[5] .   a[3] > a[5] .   Change a[3] for a[5] . |
| 2 | 3 | 5 | 8 | 6 | Compare a[4] to a[5] .   a[4] > a[5] .   Change a[4] for a[5] . |
| 2 | 3 | 5 | 6 | 8 | |

## 2. Practice of algorithm education with manual activities

The algorithms and data structures lessons are given the third-year students in the department of information engineering in "A" technical college. A 100-minute lesson is given once a week for a total of 30 sessions. The students had taken introductory classes in C programming in the second year and had experienced creating easy programs. There were 44 students in the class.

The students were divided into 11 groups who consisted of four students before they learned the basic sort algorithms. In four exercises (4×100 minutes), they created an algorithm to sort five cards in ascending order, created a program, and made a presentation for the created algorithm.

In the first exercise, the students were instructed to perform Steps 1 and 2. In the second exercise, Step 3 and Step4 were explained. Then the students were told to write an algorithm with basic control structure. In the third exercise, the students were given instructions to create and execute a C program. In the fourth exercise, each group was required to make a presentation using PowerPoint and compare the algorithm with the ones by other groups.

Table 2 shows the algorithm created by each group and the feedback from the students. As shown in Table 2, in the algorithms created by the students in these exercises, five groups (1, 2, 5, 8, and 11) used selection sort. Two groups (3 and 9) used bubble sort. Group 6 used quick sort. Two groups (4 and 10) considered an algorithm using the list structure. Group 7 created a simple and fast algorithm by handling the array subscript as data, although it used much memory. Many of the groups used selection sort and bubble sort, even before they knew about basic algorithms. Some of the groups used the list data structure because they were instructed to use the knowledge they ever learned in creating a sort algorithm. The students had also learned the criteria for evaluating algorithms, so group 7 aimed at a program that performed sorting quickly. After the four exercises about creating sort algorithms with manual activities, basic sort algorithms were explained.

Table 2: List of algorithm created by each group

| Group No. | Idea | Comments |
|---|---|---|
| 1 | Find out the minimum data and put it in front of another array. Next, find the second minimum data and put it in the second from the top of another array. | If several same data were found, an array didn't work well. Therefore, students tried to find out the minimum data, counted data and added them to another array. |
| 2, 5, 8, 11 | Find out the minimum data and exchange them for the first data. Next, find the second minimum data and exchange them for the second minimum data. | Students discussed in a group and made a program by themselves in practice. They could have deeper comprehension. |
| 3<br>9 | Compare the data next to them and exchange one for another when the data is not arranged in the right direction according to their size. | Students thought that it was the clearest way to understand but realized that it was not easy to create algorithms. |
| 6 | Divide data into two groups: one less than the average and the other for more than the average. Next, divide each group into two by the average and divide the smaller groups again by the average. | Students found it difficult to program their ideas into algorithms and repeated the process. Finally, they realized that it was the same with the quick sort algorithm, and created algorithm referring to it. |
| 4 | Insert data in order to arrange data in descending order from the top, using a list structure. | Students could insert data into any place in the sequence so they could insert data like arranging data by their hands. |
| 10 | Arrange data in descending order, using an interactive a list structure. | Students wanted to create different algorithm from other groups. As a result, their algorithm structure became complicated to understand. Inefficient and useless program was found in the algorithm. |
| 7 | Store data into sequence b□ after displaying subscripts of unsorted sequence data a□. For example, a[2]=3 is displayed as b[3]=3. Lastly, find the same value with subscripts of sequence b□ and then store them into sequence c□ in order. | Larger size of array was needed as data became larger. If the same data were found, the process didn't work well. However, the program was simple and students could arrange it quickly. |

## 3. Analysis of the practice results

### *3.1 Change of abilities and awareness*

Table 3 shows the results of a questionnaire that was conducted about nine items of the abilities and awareness of algorithms before and after the exercises of creating algorithms with manual activities. The students responded on a scale of 1 to 5 (5. Strongly agree, 4. Somewhat agree, 3. No opinion, 2. Somewhat disagree, and 1. Strongly disagree). In Table 3, m, SD, and t value are the average, standard deviation, and test statistic respectively. In the results column, ** and * show that a significant difference was observed with the significance levels of 1% and 5% respectively. Items are sorted in descending order of the t value. The following was found from the items where a significant difference was observed with a significance level of 1%.

(1) Improvement in algorithm creation capability
Item 5 “Ability to create algorithm” improved significantly. From the result, it was found that students thought that they could improve an ability of creating algorithm.
(2) Improvement in using the algorithms the students learned
Item 9 “Ability to use the learned algorithms” improved most significantly among the evaluation items. In the introductory programming course in the second year, the students had learned algorithms for selecting maximum values, Euclidean algorithms, and the Sieve of Eratosthenes using array that is the basis of data structure. As shown in Table 2, groups 4 and 10 created an algorithm using the list structure by employing such knowledge. Group 7 created a simple algorithm using the fact that the subscript in the array was placed in ascending order. Group 1, 2, 5, 8, and 11 sorted by finding the minimum value. Thus, it can be said that the students sufficiently used the algorithms they learned.

The comments in Table 4 prove that group learning improved their willingness to learn. In group learning, as described in the comments, some students do not try to think. Therefore the students were instructed that they must participate in at least one of the activities of five steps. The students who could not create the algorithm or program were made create a PowerPoint in the presentation. The authors thought that they could understand the algorithm by summarizing what their groups considered.

On the other hand, the authors understood that students felt it took them time to finish, while sequential expression of a series of motions by manual algorithm creation easier.

Table 3: Growth of abilities and awareness

| No. | Item | Before | | After | | t-Test | |
|---|---|---|---|---|---|---|---|
| | | m | SD | m | SD | Value of t | Results |
| 9 | Ability to use the learned algorithms | 2.2 | 0.8 | 3.0 | 0.8 | 6.2 | ** |
| 3 | Ability to devise solution to the problem | 2.2 | 0.9 | 2.8 | 0.9 | 4.6 | ** |
| 8 | Ability to design test data | 2.1 | 0.9 | 3.0 | 1.0 | 4.2 | ** |
| 1 | Knowlegde about sort algorithm | 2.4 | 1.1 | 3.0 | 0.8 | 3.6 | ** |
| 5 | Ability to create algorithm (to find regularity) | 2.0 | 1.0 | 2.7 | 0.9 | 3.2 | ** |
| 7 | Ability to check for algorithm | 2.2 | 1.0 | 2.7 | 0.9 | 3.1 | ** |
| 4 | Ability to refine asolution to the problem in stages | 2.2 | 0.9 | 2.6 | 0.9 | 2.9 | ** |
| 6 | Ability to express in program language☐ | 2.9 | 1.0 | 3.2 | 0.9 | 2.3 | * |
| 2 | Ability to understand solution to the problem | 2.8 | 1.0 | 3.1 | 0.9 | 1.5 | |
| Average | | 2.3 | 1.0 | 2.9 | 0.9 | 10.2 | ** |

Table 4: Comments about practice of group exercises

| Comment |
|---|
| ・ I was able to hear other students’ opinions and think about various solutions. |
| ・ I was able to discuss what I did not understand with other students. |
| ・ Some students did not think. |
| ・ I could understand more deeply by thinking step by step. |
| ・ Manual processing facilitated making the algorithm. |
| ・ Manual operation took time. |

## *3.2 Comparison of scores in examination*

In 2009, seven classroom activities were practiced to teach students a principle of five sort algorithms and algorithm to create programming. Students did not work in groups.

In 2010, four exercises were practiced to teach how to create algorithm by manual in group learning as suggested in this paper. After the four exercises, three more classes were done to teach a principle of five sort algorithms. The three classes were not enough for the students, so students were assigned to create algorithm and program by themselves as homework after the classes. The classes were practiced seven times in 2010. It is just the same times as in 2009.

The authors included the same six questions among ones in the examination as shown in Table 5 to know quantum of knowledge in 2009 and 2010.

A test of significant difference shows that the average score of the six questions in 2010 was significantly higher as a whole (t (82) = 3.6, p < 0.01). It can be said that the students in 2010 have more knowledge of basic sort algorithms than those in 2009. It is found that the dispersion of score in 2010 is small because the standard deviation in 2010 is smaller than that in 2009.

For Question 3 and 4, significant differences were observed with a significance level of 1%. It can be said that the students in 2010, who wrote the flow in array, understood the idea of insertion sorting and data flow better than those in 2009. For question 6, a significant difference was also observed with a significance level of 1%. This shows that the students in 2010 understood better that a bubble sort program can be improved by using the fact that the data have been sorted if there is no change in data in sorting. The result of this test shows that activities to create manual algorithm are an effective method for students to concretely understand the principle and data flow in sort algorithms, and improve the retention of knowledge.

Table 5: Comparison between scores of examination in 2009 and 2010

| No. | Eexamination Questions | Score | 2009 | | 2010 | | t-Test | |
|---|---|---|---|---|---|---|---|---|
| | | | m | SD | m | SD | Value of t | Result |
| Q.1 | Fill in the blanks of the program for selection sort. | 18 | 15.8 | 3.8 | 16.5 | 2.8 | 1.0 | |
| Q.2 | Calculate the amount of calculation by the number of comparisons in selection sort. | 4 | 3.4 | 1.1 | 3.6 | 1.0 | 0.9 | |
| Q.3 | Fill in the blanks of the text about the amount of calculation in insertion sort. | 2 | 1.5 | 0.9 | 2.0 | 0.3 | 3.4 | ** |
| Q.4 | Write the data list after the loop outside of the insertion sort algorithm was performed once. | 9 | 5.1 | 3.6 | 8.6 | 1.5 | 5.6 | ** |
| Q.5 | Write the data list after the loop outside of the bubble sort algorithm was performed once. | 12 | 8.2 | 2.0 | 8.3 | 3.0 | 0.2 | |
| Q.6 | Efficiency can be improved in bubble sort. Write the reason and the program for the improvement. | 4 | 2.8 | 1.6 | 3.6 | 1.2 | 2.7 | ** |
| | Total | 49 | 36.6 | 8.1 | 42.5 | 6.7 | 3.6 | ** |

## 4. Conclusion

Sorting the cards manually, describing the flow with array, and creating an algorithm took time, but these activities made the flow easy to understand and the algorithm easy to create. In addition, from the comparison of the test scores, it turned out that the students could understand the principle of sort algorithms better and consider data flow more concretely with manual activities. From these results, it can be said that the proposed method is effective in fostering the ability to understand the principles of algorithms and create algorithms.

The authors appreciate the support of the Grant-in-Aid for Scientific Research, foundation study (C22500955) provided by the Ministry of Education, Culture, Sports, Science and Technology, Japan for this research.

## References

[1] Abe, T. (2003). Development of Tools for Computer Algorithm Instruction. *Japan Society for Educational Technology*, *27*(Suppl.), 45-48 (In Japanese).

[2] Namatame, Y. (2004). The Evaluation of the Group Study Method Using Peer Review in the Lesson of Programming. *Transactions of Information Processing Society of Japan*, *45*(9), 2226-2235 (In Japanese).

[3] Sugiura, M., Matsuzawa, Y., Okada, K., & Ohiwa, H. (2008). Introductory Education for Algorithm Construction: Understanding Concepts of Algorithm through Unplugged Work and Its Effects. *Transaction of Information Processing Society of Japan*, *49*(10), 3409-3427 (In Japanese).