Developing Algorithmic Thinking by Inventing and Playing Algorithms

Gerald Futschek, *futschek@ifs.tuwien.ac.at*
Institute of Software Technology and Interactive Systems, Vienna University of Technology

Julia Moschitz, *moschitz@ifs.tuwien.ac.at*
Institute of Software Technology and Interactive Systems, Vienna University of Technology

Abstract

In many cases at school and at universities most of the learners consider the topic of algorithms as hard and not very attractive. Very often the focus of traditional courses is on learning specific algorithms that are considered as important in education or in practice. Often these algorithms are sequential algorithms. We show in contrast to these courses a way of learning principles and concepts of algorithms that is much easier to comprehend by the learners and makes them more fun. The idea is that we do involve as many students as possible in playing algorithms that are moreover usually proposed by them.

The task of the teacher is to give proper problem statements and to ask proper questions to keep the students thinking to create working algorithms that solve these problems. The teacher also motivates the students to improve their algorithms to find more efficient solutions.

Compared to a theatre play the students have the roles of the actors and the idea deliverers and the teacher has the role of the stage manager.

We give two examples: The first example is the calculation of a maximal value of a set of values, where each student represents a value. Parallel activities may improve the efficiency of the algorithm. Usually the students find good solutions and learn a lot about concepts of sequential and parallel algorithms that are usually learned in advanced algorithm courses.

It is a form of explorative learning, where the students can experience algorithms by playing them and they can determine the progress and invent algorithms that they play.

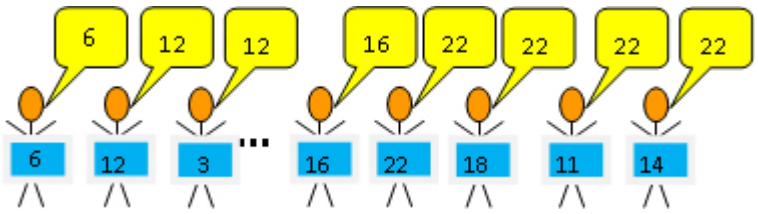

Figure 1. Sequential algorithm to calculate the maximum value. Each student passes the maximum_value_so_far to the next student in row.

The second example is named ‘let’s play robots’, where students assume the roles of a robot and a navigator. By this way students learn more about basic algorithmic thinking while they are playing algorithms. A model of learning by inventing and playing algorithms is presented that proposes a cycle of 5 processes which the learners may perform for inventing algorithms.

Keywords

Algorithmic thinking, explorative learning, group learning, creativity

Motivation and Introduction

Algorithmic thinking is considered to be one of the main abilities that pupils may achieve in informatics education at school and university level. Algorithmic thinking as fundamental idea of informatics education is very complex and consists of a wide variety of abilities that can be understood at different intellectual levels, see Futschek (2006).

Furthermore finding and inventing appropriate algorithms is a necessary prerequisite of computer programming. Understanding algorithms is known as one of the difficulties students are confronted with when starting to learn programming, see Jenkins (2002).

To attract especially the younger and the newcomer to be more interested in computer science, it is very important to convey that algorithms are powerful tools that open a wide field of interesting activities where it is possible to achieve significant progress by providing new ideas.

There are several ways to help learners to understand principles of algorithms. Often animations of algorithms are used that are played by computer programs. Other approaches provide tasks and scenarios for learning principles of algorithms without using computers, see Bischof and Mittermeir (2008) and the Computer Science Unplugged activities from Fellows et al (2005). In these approaches the students play algorithms and get in this way a better understanding of given algorithms. In Futschek (2007) a learning scenario for playing counting algorithms is presented where a larger group of students can be involved in the play.

In this paper we present an approach where students invent algorithms to solve given problems. Because the students invent the algorithms they usually want to play their algorithms. While playing the algorithms the students find out the advantages and disadvantages of their algorithms and are highly motivated to discover necessary algorithmic concepts to improve their algorithms. A model for learning by inventing algorithms is presented.

Important Aspects of Algorithmic Thinking for Beginners

Algorithmic thinking is a special problem solving competence, which consists of several abilities, see Futschek (2006):

- analyze given problems
- specify problems precisely
- find the basic actions that are adequate to given problems
- construct correct algorithms to given problems using the basic actions
- think about all possible special and normal cases of a problem
- evaluate algorithms (correctness, efficiency, termination)
- improve the efficiency of algorithms

Algorithmic thinking consists amongst others of this wide range of abilities and is also influenced by many other human cognitive factors: abstract and logical thinking, thinking in structures, creativity and problem solving competence. This complexity makes algorithmic thinking not easy to learn and explains the need for a good didactic approach especially for beginners. However, this is not the only reason why beginners consider learning algorithmic thinking as hard.

Another reason is that algorithmic thinking at a certain point is an unnatural thinking type which has to be especially trained by a learner. In everyday life we often have to solve natural problems by algorithms where we have to find good solutions for having a better life. However, if we have to write a software program, we have to find a solution for a machine. This solution has to be made comprehensible for machines which usually like sequential instructions and have their own very basic instruction set. Humans in contrary like to cooperate and prefer parallel actions and have also a high level instruction set.

We think that especially for beginners the complexity should be reduced to that level where the concepts of algorithmic thinking can be learned in a natural way.

Therefore we need

- tasks that the learners know from daily life
- a natural description language for algorithms
- basic actions that the learners know from daily life
- a system that runs the algorithm
- a systems that allows the learners to experiment with the algorithm
- a system that gives immediate learning experiences
- a system that is flexible to run a variety of algorithms
- somebody who provides feedback

As system to run the algorithms we engage the learners themselves, they are playing the algorithms. The learners are intelligent processors that also run concurrent algorithms and can execute also natural high level commands. Therefore the tasks can be taken from daily life. As we will show in this article the learners can make fast learning progress in all abilities that constitute algorithmic thinking and they experience also advanced algorithmic concepts in a very natural and comprehensible way.

Important is that the problems to be solved are adequate to the pre-knowledge of the beginners. In the best way the given problems are from the students experience or from everyday life, because familiar examples can be comprehend immediately by students. These problems should be so general that they give way to a variety of different algorithmic solutions.

For students and their solution it is not necessary to know an exact language to describe algorithms (like programming languages). At the beginning the native language is adequate.

For beginners the knowledge of specific algorithms is not so important, but the ability to understand principles of algorithms and to find or create own algorithms for new problems. One main educational objective for beginners is to know that an algorithm prescribes exactly what to do in all possible situations. So, an algorithm prescribes not only the activities in the main situations but governs all possible situations. This can be experienced in a play that follows accurately the script of a self-invented algorithm.

The teacher has an important role, although he or she should take a back seat. The aim of the teacher is not to present solutions but to support students in their learning process, he should motivate students to make progress in finding solutions.

Learning by Playing Algorithms

Written algorithms are often too abstract for beginners to be understood. The students have to understand the syntax of the language that describes algorithms and also the idea how the algorithm solves the problem. Algorithms involve too many concepts, so a good learning approach needs a systematic stepwise strategy in a way the students understand why these concepts are necessary. We have the concepts of algorithm description languages, sequential and parallel algorithms, efficiency of algorithms, software agents, synchronization of parallel processes, broadcasting, shared variables, atomic actions, state transitions, correctness, etc. Playing algorithms is a good way to learn the principles more slowly and in a more effective way. We distinguish the following possibilities of playing algorithms:

- teacher is playing
- software is playing
- some students are playing
- all students are playing

We ordered these possibilities by its learning efficiency in learning. If the teacher demonstrates algorithms and shows what is going on, the students see that the teacher knows how the algorithms work but often it does not help them to get better understanding. The disadvantage is that students only watch the solutions and so they are in a passive role.

If a piece of software plays the algorithm the students can actively experiment with different inputs. Active learning is more efficient than passive learning. Students who play algorithms themselves get more insight in the algorithm details. The more students are involved in playing algorithms the better. Therefore our goal is to find problems where all students are involved in playing algorithms. If students are playing algorithms, they are in an active role. This method is time-consuming thereby effective learning is possible because “feelings, thinking, memories and physical sensations” are activated, see Siebert (2005).

The advantage of playing given algorithms is that the students see good solutions for problems and get inspiration for inventing other algorithms.

Model for Learning by Inventing Algorithms

Much more motivation and identification with the topic arises when the students get the possibility to invent own algorithms to solve a problem. Then playing the algorithms is more attractive for the students. A good choice of the problems to be solved is a prerequisite.

In this chapter we define a process of learning by inventing algorithms by dividing the process into five main steps based on the model of problem-solving thinking by Tümmers, see Seel (2005). The role of the teacher changes in this process from a traditional teacher to a coach taking a back seat. He is a supporter of the students. He motivates the students and states the original problem statement. By this way the students come to the front seats and are forced to be active. All five processes of figure 2 are done by the students. If necessary the teacher initiates the processes but he does not actively participate in finding, testing and improving solutions.

Important for this process are good problems according to age, previous knowledge and experience of the students.

Additionally, reflecting the learning process is a major part of this process. Often reflecting the learning process for students is something new and strange. However, on this way students learn to understand their programming problems by and by.

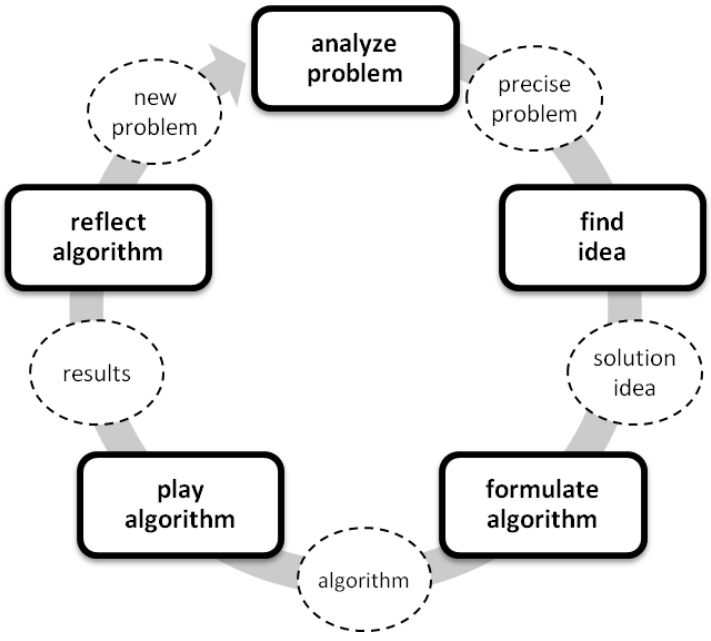

Figure 2. The process of learning by inventing algorithms

Analyze problem

The first step is to find out more details about the problem. In this process students try to find the main problem and to split it up into smaller problems.

Find solution idea

In this process the students have to be creative and should propose ideas, how they can solve the problem. If they have a problem, which is divided into smaller problem tasks, students can start to look for ideas for the smaller problems. They can write their ideas down or in a better way they discuss in groups their ideas. After the discussion they evaluate their ideas under the following aspects: Which idea can most properly solve the problem? Which idea can easily put into practice? Which of the ideas are efficient?

Formulate algorithm

In this step the goal is to write down or say a precise formulation of the ideas to solve the problem. For beginners the solutions can also be described orally. Advanced learners write down a precise formulation. At this time students have often problems to formulate specifically their solutions. Exchange of problems and questions with other students or the teacher can help solving this problem. Especially the basic actions should be clear.

Playing the algorithm

Main goal of playing the algorithm is to find out if it is working at all and how good it is working. Educational objective of this phase is to learn that not all ideas are working and to get a feeling for possible sources of failures.

Reflect algorithm

The purpose of reflecting the algorithm is to improve the solution. The results of the reflection are new problems to be solved. So the process starts again from the beginning. One possible reflection task is to find out the efficiency of the algorithm. Is it fast or slow, does it do unnecessary actions? In trying to give answers to these questions beginners can develop a feeling of the time complexity of an algorithm.

While learning in groups the students learn a lot from other students and their ideas. Furthermore motivation and assistance from the teacher help students getting ahead with their learning progress.

Examples

The following examples are taken from the authors teaching courses for students that are starting to learn about algorithms in tertiary education. The given examples show typical tasks and possible ways of teaching that were successful and may be taken to be used in a similar way in another learning context. The problem statement of the task to be solved should be very easily understandable and should give room for a wide variety of solutions. In Futschek (2007) the task of counting the number of a group of people was given and different algorithmic solutions were discussed. In our opinion this is the best first task for learning algorithms in a larger group (from about 10 up to some hundred students). All group members can be involved in counting activities and there is enough room for inventing efficient solutions. Here we want to give another example of this kind that is also very suitable for beginners.

Example: Maximum value

The task is to find the eldest or youngest student or to find the student with the highest student number or with the latest birthday within a year. We also tried this task where we distributed

sheets with numbers on it. Very often some students know a sequential solution, where the students form a row of values:

> *"Beginning with the first value in the row one after the other compares its value with the maximum_value_so_far and if it is larger the maximum_value_so_far becomes this value. At the beginning the maximum_value_so_far becomes the first value in the row. After the last value in row has done his job the maximum_value_so_far holds the maximum of all values."*

Usually this solution is proposed if students know already the sequential solution and also if they are sitting in rows it is very natural to find a sequential solution.

Usually there emerge some minor problems while playing this algorithm. It is not always clear in what form the maximum_value_so_far is represented, so that the students that have to compare it with their own value have access to it. There are different possible solutions to this. First the maximum_value_so_far is always passed by the previous student to the next one. And the last one delivers the maximum value as result of the problem. Another solution is to involve a board, where the actual value of maximum_value_so_far is written and can be seen by all students. If these different solutions arise the teacher should discuss with the students the differences of these approaches. A board that is accessible and visible by all is not easy to handle if there are hundreds or even thousands of students. Who has writing access to the board?

Although this algorithm seems to be easy and clear, it is important to play the algorithm. The students can see that the activity moves from the beginning to the end and that there is a need to remember the maximum_value_so_far and that all players must have on their turn access to the actual value of the maximum_value_so_far.

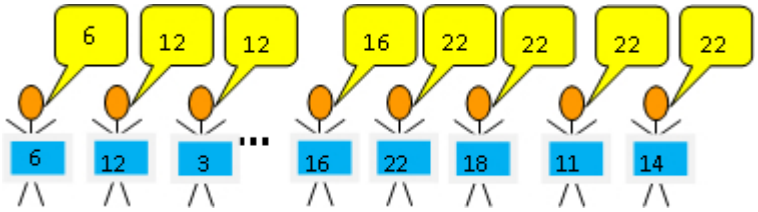

Figure 3. Sequential algorithm to calculate the maximum value. Each student passes the maximum_value_so_far to the next student in row.

The teacher makes the students to play their algorithms, and motivates them to write down the algorithms and poses questions to keep the learning process in progress. Possible questions of the teacher to induce an analysis of this algorithm by the students:

- Does each student run exactly the same algorithm?
- How do you know who is the next in a row?
- Does this algorithm always find the maximum value?
- How many steps does it take to find the maximum value?
- How would you solve this if you are standing in a group and there are no rows?

The last question leads to a nice solution where the students find ad hoc a next student who has not yet compared his value with the maximum_value_so_far. For this case the students have to indicate if they are still candidates for a next value or not, for example by standing instead of sitting. The following questions should motivate to find a more efficient solution:

- How many minutes does the algorithm take if there are a thousand students?
- Can the problem be solved faster?
- Are all players very active in this algorithm?
- Can the algorithm be improved by making the players more active?

Parallel Algorithms

A simple speed improvement often proposed by students is:

> *"Calculate in parallel the maximum of front half and the maximum of back half and deliver the maximum of these two."*

A question to clarify the improvement may be:

- How much is this parallel variant better than the sequential one?
- Are the students more active than before?

The teacher encourages the students to find a much faster solution. A possible idea to calculate the maximum much faster is as follows:

> *"Parallel calculation of maxima of rows. Then follows the calculation of the maximum of all row maxima."*

Usually this algorithm is suggested when the students are sitting in rows. It is nice to find out how much faster this algorithm is than the previous ones. If appropriate the teacher can address the problems of describing the efficiency of an algorithm. In this particular case the order of the algorithm is $\sqrt{N}$ if we have N rows and N columns.

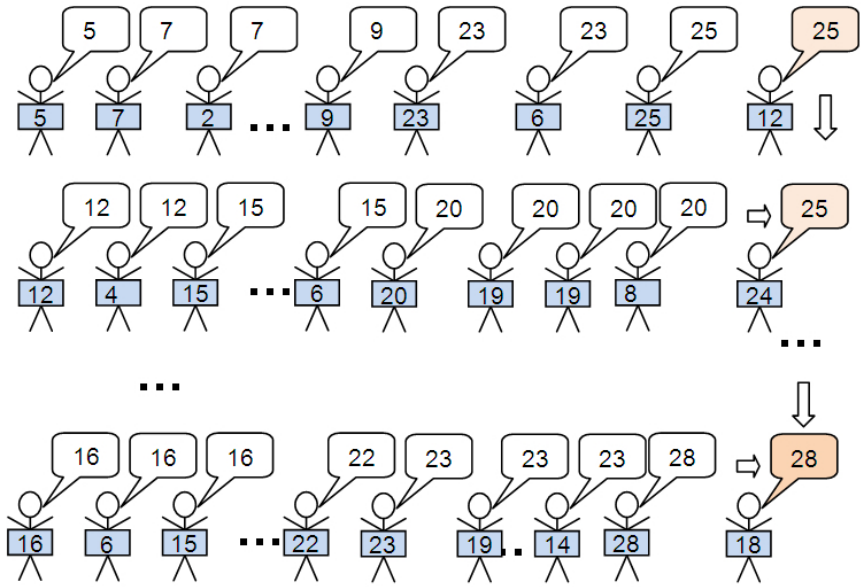

Figure 4. Parallel algorithm to calculate the maximum value. Each student passes the maximum_value_so_far to the next student in row. The students at end of each row calculate the maximum of three values: from side, from behind and its own value.

While playing this algorithm the students may find out that especially for the last in each row it is hard to coordinate getting values from the side and from behind that arrive at the same time. There arises the necessity to synchronize the two players that pass a value. So there is an agreement between the two players at what time both are ready to pass the value. A good synchronization technique is to look in each other eyes.

Sometimes students propose algorithms that need more clarification. For example the following parallel algorithm:

> "*All students compare in parallel their value with maximum_value_so_far and if their value is higher they replace the maximum_value_so_far by their own value.*"

Here it is necessary to clarify that comparing and replacing has to be a single atomic action to achieve a correct solution. While playing this algorithm the students can discover that replacing the maximum value cannot be done in parallel.

Another proposal of the students may be to exclude values that are smaller than other values:

> **repeat**
> choice of a value that is still in game
> all values less than this value are removing themselves from the game
> **until** only one value is remaining

Here a sort of broadcasting is necessary, since all students have to know the value that is selected.

Possible questions of the teacher to clarify additional problems:

- Does this algorithm always find a solution? Under what condition?
- How is it indicated that a value is still in the game (or is already excluded from the game)?

Another algorithm that may be proposed: All students have sheets with values that are readable also by other students.

> *All students show their sheet to the other students and all look in an arbitrary order at the sheets of the other students. When a student sees a sheet with a larger number than his own number, he removes himself from the game.*

Questions:

- How many steps are needed to find the maximum at least, at most, in average?
- What are the basic actions?
- Do we need synchronization? If not, why?
- When does this algorithm terminate?

It seems to be important to involve all students in playing algorithms. Although the students have to follow accurately the given algorithm, they know better what is going on, how long it takes and where the problems arise. After each play they can analyze the play and can think about better or other algorithms.

The role of the teacher is not to write the script of the play. He proposes the problem to be solved and he is the play manager. Additionally he has teaching goals to fulfil, he can tell the students specific technical terms of informatics concepts of specific findings of the students.

Example: Let's Play Robot

The topic 'Robots' is fascinating for students, especially for young pupils. The topic "robots" is appropriate for projects at school, because it allows being creative in many ways. This fascination can be experienced by the students in the next example. This simple game is suitable for learners in small groups and is for learners from age 8 who have no or only a little experience in algorithmic thinking. By this game students learn describing algorithms exactly and in a specific order. Furthermore students learn to invent easy algorithms in small groups, they find out that not every solution works and they improve their already found solutions. The duration of this teaching method depends on the given exercises and takes between 15 and 60 minutes.

In practice this game is really simple and all students like assuming roles, even students at university level. Students assuming robots often think too active and show intelligent behaviour that is not part of the algorithm. Preventing this we give robots clothes to blindfold them. As a

result of such games we observed also positive social skills effects on groups. The groups seem to be more communicative, have more confidence within the group and participate more active in the course.

The teacher asks the students to arrange in pairs. One of the pair assumes the role of the robot and the other one is the navigator of the robot. The teacher hands out papers to navigators with short instructions for the robot (forward, backward, put, …) and different exercises. This game can be played by all small groups at the same time. To begin this game the teacher gives a signal like “let’s play robot”. During the game students should be invited to change the roles. After the game the students present their solution to the others. Often beginners consider improving an algorithm as hard. To activate students thinking about improving their solution the teacher can ask questions like: Are you satisfied with your solution? Such questions help students reflecting their solution without anticipating a better solution (by the teacher).

Examples for tasks

- Task 1
  The first task is to navigate the robot through the room without touching something or somebody. Through this task the students learn the meaning of commands and to follow basic actions in a specific order.
- Task 2
  In the second task the navigator determines a point in the room and tries to navigate the robot to this point. Enhancing the level of the difficulty the navigator writes the algorithm before the robot starts. In the next level the algorithm should work for every point in the room. By this way students learn analyzing the problem, designing an algorithm and finding a solution for all possible special and normal cases of this problem.
- Task 3
  In this task the goal is finding the best solution for painting a star on a paper on the floor. The navigator tries to find an algorithm and gives the instruction to the robot. The robot paints a drawing according the instruction on the paper.
  In this task the students learn expanding their basic actions, because they have to find new instructions for the robot. Further they gain knowledge in analyzing a problem, finding solutions and get first experience in improving algorithms.

These examples of tasks show that the tasks can be adapted in many ways to the knowledge level of the learner. Ideas for other exercises can be found in books and publications about Logo or robots, because these exercises can be allocated also to the game “let’s play robot”.

Summary

Inventing algorithms is an effective learning method that can be done also with novices in algorithms. The students can play the algorithms to well chosen tasks of daily life and find out in a natural way even advanced algorithmic concepts like concurrency, synchronization, broadcasting, shared variables, etc.

In the proposed learning scenarios the students

- learn actively
- learn in groups
- govern the progress of learning
- are actors
- are script writers

and the teachers

- have to deal with unexpected proposals
- should have explored possible solutions before the lecture
- must be very firm in algorithms (analyse, create, not just replicate)
- are the play managers
- propose the original problem
- ask questions (that provoke often new sub-problems).

References

Bischof E. and Mittermeir, R. (2008) Informatik erleben (in German). Institut fur Informatik-Systeme, Alpen-Adria University Klagenfurt, see also http://informatik-erleben.uni-klu.ac.at/.

Fellows M., Bell T. and Witten I. (2005) Computer Science Unplugged, see also http://csunplugged.org/.

Futschek, G. (2006) *Algorithmic Thinking: The Key for Understanding Computer Science*. In Lecture Notes in Computer Science 4226, Springer, pp. 159 - 168.

Futschek, G. (2007) *Logo-Like Learning of Basic Concepts of Algorithms – Having Fun with Algorithms*. In Proceedings of Eurologo 2007. Edited by I. Kalas, Bratislava.

Jenkins T. (2002) *On the Difficulty of Learning to Program*. In Proceedings of 3rd annual Conference on LTSN-ICS, Loughbourgh, pp. 53-58.

Seel N. (2003) Psychologie des Lernens (in German), Ernst Reinhard Verlag München Basel.

Siebert H. (2005) Pädagogischer Konstruktivismus (in German), Beltz Verlag.