9/1 (2011), 107–116
tmcs@math.klte.hu
http://tmcs.math.klte.hu
Teaching Mathematics and Computer Science

Cultivating algorithmic thinking: an important issue for both technical and HUMAN sciences

Zoltán Kátai, Lehel István Kovács, Zoltán Kása, Gyöngyvér Márton, Kinga Fogarasi and Ferenc Fogarasi

Abstract. Algorithmic thinking is a valuable skill that all people should master. In this paper we propose a one-semester, algorithm-oriented computer science course for human science students. According to our experience such an initiative could succeed only if the next recipe is followed: interesting and practical content + exciting didactical methods + minimal programming. More explicitly, we suggest: (1) A special, simple, minimal, pseudo-code like imperative programming language that integrates a graphic library. (2) Interesting, practical and problem-oriented content with philosophical implications. (3) Exciting, human science related didactical methods including art-based, inter-cultural elements.

Key words and phrases: Algorithmic thinking, teaching learning programming, multi-sensory education, intercultural education, curriculum.

ZDM Subject Classification: B50, B70, U60,U80.

1. The importance of cultivating algorithmic thinking for all students

An algorithm is a precise, step-by-step set of instructions for solving a task. An algorithm does not solve a task, it rather gives us a series of steps that, if executed correctly, will result in a solution to a task. All people use algorithms

Acknowledgment: The elaboration of the didactical tool that integrates the "sorting-dance performances" (Algo-rythmics project) was supported by "Szülőföldalap".

every day but they often do not explicitly think about the individual steps of the algorithm. Algorithmic thinking is the ability to understand, execute, evaluate, and create algorithms. According to Futschek [3] algorithmic thinking is a valuable skill that all people should master.

In all fields of science efficiency presupposes developed modeling skills. While engineers are modeling physical processes, human sciences are concerned with the modeling of social phenomena. In the course of modeling we create entities out of real objects by abstraction. The human mind observes, abstracts, differentiates and systematizes the objects or real entities it is surrounded by. The supreme goal is to arrive at an understanding of how the complex system works. The means to achieve this is modeling. Modeling is but the use of a basic and elementary train of thought (an algorithm) which human beings are capable of abstracting, differentiating, categorizing, generalizing or specifying, analyzing as well as establishing relations. Cultivating algorithmic thinking students also develop their modeling skills.

Any device can be used more efficiently if it is clear how it works. Understanding the principles behind computer sciences (and related fields of science) results in more qualified computer users. For example, since the two thousand years old Euclid algorithm is a basic pillar of today's data security, teachers can use it as a starting point to explain practical aspects of cryptography and computational complexity: why it is hard to decipher somebody's password, how we can send confidential letters, what lies behind banking transactions, etc.

On the other hand, the huge amount of computation that can be done with computers has resulted in new ways of thinking about proving something, being certain of something etc. The philosophical aspects of these issues represent further reasons why human science students should be initiated into computer sciences. This train of thought reminds us of the polyhistors of the past who had a wide range of knowledge, and it raises the question if nowadays scientists should not be such as they were (at least in matter of principles).

Accordingly, in this paper we propose a one-semester, algorithm-oriented computer science course for human science students.

2. Promoting algorithmic thinking by teaching-learning programming

Creating algorithms without implementing them could be a frustrating task. Experiencing that the algorithm we have just created works is an encouraging feeling. Computers are undisputedly the most effective educational tools for implementing algorithms. However, this means computer programming. Although, obviously, most of the human science students will not become programmers, basic algorithm-related elements of the computer sciences can improve their algorithmic thinking. (Human scientists regularly use word-processors and spreadsheet applications that support macros, and creating macros implies programming)

According to our experience such an initiative could succeed only if the next recipe is followed: interesting and practical content + exciting didactical methods + minimal programming. More explicitly, we suggest: (1) A special, simple, minimal, pseudo-code like imperative programming language that integrates a graphic library. (2) Interesting, practical and problem-oriented content with philosophical implications. (3) Exciting, human science related didactical methods including art-based, inter-cultural elements.

2.1. The proposed syllabus

According to the USA National Research Council's Committee on Information Technology Literacy the general concept of algorithmic thinking includes functional decomposition, repetition (iteration and/or recursion), basic data organizations, generalization and parameterization, algorithm vs. program, top-down design, and refinement etc. [9] In line with this we propose the following minimal syllabus:

- Introduction (1^{st} week)
    - The concept of variable (we prefer the notion of storage as denomination instead of variable). To avoid the concept of type, all variables are integers. As data structures, we introduced only the one- and two-dimensional arrays.
- Elementary algorithms
    - Interchanging the content of two storages (with or without the use of a third storage). (2^{rd} week)
        - Assignment statement; simplified input/output operations; expressions (basic operators)

- ◦ Decision-structures. (3rd week)
    - ■ if [else] statement. (Parallel scenarios)
- ◦ Iterative algorithms (Loop-instructions: while, for)
    - ■ Searching algorithms (linear/binary) (4th week)
    - ■ The Euclidian algorithm. (Data security: basic concepts) (5th week)
    - ■ Sorting algorithms (insert-, bubble-, shell- and select-sort) (6th week)
        - Imbricated loop-structures
    - ■ The concept of algorithm-complexity

- Subprograms (Top-down algorithm design strategy) (7th week)
    - ◦ Global/local storages
    - ◦ Parameter passing by value/reference
- Recursive algorithms (8–9th week)
    - ◦ Recursion in nature, arts and literature
- Algorithm design strategies (classic examples)
    - ◦ Greedy strategy ("Job sequencing with deadlines" problem) (10th week)
        - ■ Greedy heuristic
    - ◦ Backtracking strategy ("Queens" problem) (11th week)
    - ◦ Divide and conquer strategy ("Towers of Hanoi" problem) (12th week)
    - ◦ Dynamic programming (13th week)
    - ◦ Comparing analyze of the strategies ("Mouse in the maze" problem) (14th week)
        - ■ Top-down vs. bottom-up strategies
        - ■ Drawing a parallel between algorithm design strategies and different life philosophies.

2.2. The suggested didactical methods

Algorithmic thinking (and programming) goes hand in hand with abstract reasoning. According to Navrat [10] a possible factor contributing to students' difficulties in learning to program is the abstractness of the programming process. In view of this Kátai and his colleagues [6], [7], [8] analyzed how multi-sensory or technologically enhanced learning can be implemented in computer-programming education and in what ways this process can be catalyzed by arts (dance, music, rhythm, theatrical role-playing). According to Stevens and Goldberg [14], our brains desire multi-sensory input and learning engages the whole body. Staley [13] emphasizes that senses do not reach only our feelings, emotions and aesthetic sense, but our intellect as well. Technology (like computer simulation) has a key-role in creating multi-sensory environment in classrooms.

Science and art are a winning combination in educational contexts because: (1) It creates a multi-sensory learning environment that involves almost all senses: visual, auditory, kinaesthetic, and tactile; (2) It contributes to a balanced involvement of both sides (academic/artistic) of the brain in the classes that could significantly improve teaching-learning process [2]; (3) It promotes various ways of learning that also enhance the educational process [4], [5].

Illustrations and examples are powerful teaching devices, too. They often command and hold attention with remarkable effectiveness, and stimulate the thinking faculties. They are also an effective memory aid. In the followings we propose a variety of multi-sensory didactical methods for almost all subjects included in the above suggested syllabus.

2.2.1. Teaching-learning elementary algorithms

Any algorithm has a loop skeleton, its structure of loops. Teachers should help students comprehend clearly the loop skeleton of the algorithm already in the analyzing phase of the problem- solving process. We have developed a multi-sensory software-tool to enhance students' skills to identify the loop skeleton of a certain algorithm.

The application makes it possible to create program-skeletons with different loop structures in an automatic way: (1) giving the parameters of the loop skeleton; (2) drumming the loop skeleton in (using the keyboard students drum the rhythm-pattern of certain loop skeletons). On the other hand, the software integrates piano sound and delay procedures in the nuclei of each loop instruction. This module functions as a loud speaker of the loop skeletons. When the algorithm has loops in both branches of a selection, in these parallel loops the software implements the same sounds but with different musical instruments. The application makes possible the following multi-sensory learning experience: While students are listening to the loop skeleton of the algorithm (represented by its piano-sound sequence), they keep their eyes on the running program (the instruction which is being executed is highlighted). [6]

2.2.1.1 Teaching-learning searching algorithms

To help students imagine how linear/binary searching algorithms work, the teacher should invite them to play these strategies. The teacher needs $n + 2$ actors (n is the length of the sequence): n students for the number-sequence, one for the loop-variable and one for the searched value. The scenario should closely follow the corresponding searching strategy. Whereas spectator-students get an overview of the whole process, actor-students memorize in their muscles (kinaesthetic memory) specific moves associated with elementary operations related to the algorithms. [7]

2.2.1.2 Teaching-learning sorting algorithms

We have designed a special didactical method that integrates multicultural folk-dance performances (Romanian, Hungarian, German and Gipsy) in the teaching-learning process of sorting algorithms (Insert-sort, Select-sort, Bubble-sort, Shell-sort). The number-sequence is personified by the dancer-sequence (dancers wear the corresponding number on their dress) (see Figure 1) [8]. We suggest the following syllabus to implement this method:

- Students view the dance-performance that illustrates the analyzed sorting algorithm.
- Using the segmented version of the video-recording students delimit and identify the key-operations of the algorithm.
- Students are invited to reconstruct the operation-sequence. At each step they have to choose the next operation to be executed and identify the elements the selected operation has to be applied on. For example, if the selected operation is comparing, the software directs the user to click on the corresponding dancers.
- Students are invited to repeat the reconstruction process of the algorithm on a black-box sequence. The software informs them about the results of the "comparing" operations.
- Teachers briefly discuss cultural aspects of dance-performances with students. (Dance-choreographies were designed to promote both Computer Science education and appreciation for cultural diversity in Transylvania, Romania)

2.2.2. Top-down algorithm design strategy (sub-programs)

There are a lot of practical examples in all fields of science which can be used as illustrations for the top-down problem-solving strategy. Additionally, if

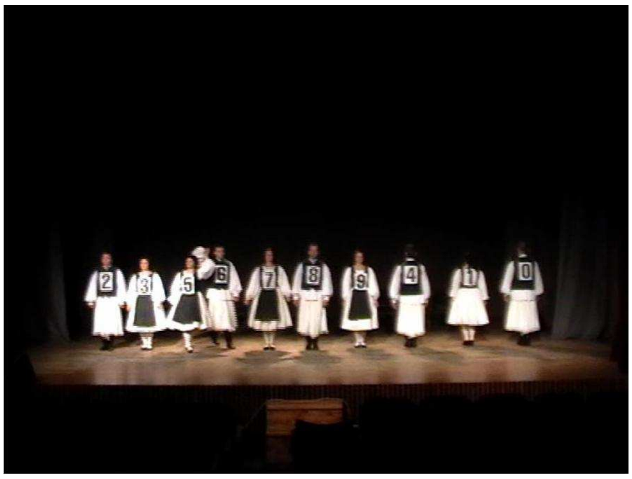

Figure 1. Romanian folk-dance (sub-region Bihor, Transylvania). Interchanging two elements

teachers present the sub-program as a scenario, then the effective parameters can be considered as actors. In the case of the parameters passed by value the roles are played by doublers.

2.2.3. Teaching-learning recursive algorithms

Recursion in computer science is a way of thinking about and solving problems. Solving a problem using recursion means the solution depends on solutions to smaller instances of the same problem. When a recursive procedure/function is called the computer keeps track of the various instances of the function. The teacher can didactically compare these instances to a clone-population. Creating a recursive sub-program essentially requires defining a base-case and then defining rules to break down more complex cases into the base-case. The base-case condition is usually implemented by an if-instruction. This key-if delimits two scenarios for all clone-instances: the base-case-scenario and the recursive-scenario. The current clone chooses between scenarios according to the task it received through its parameter list.

We have developed a multimedia instrument to help students to understand how recursion works. We have implemented successive segments from the classical piano masterpiece, Für Elise by Ludwig von Beethoven, as accompanying music for the execution of particular zones of the recursive scenario. Between successive processing levels, a constant shift is applied (on the score of pitch). For the base-case zone we chose a different music-fragment. (see Figure 2)

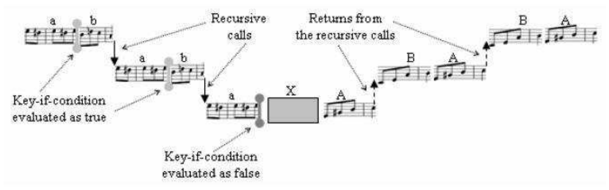
a
b
Recursive calls
Returns from the recursive calls
B
A
a
b
Key-if-condition evaluated as true
a
X
A
B
A
Key-if-condition evaluated as false

Figure 2. The staircase illustration of a three level recursive call

Role-playing could be an efficient didactical tool in case of teaching learning recursion too. (see Figure 3) [7]

Figure 3. Staging a recursive scene

2.2.4. Teaching-learning algorithm design strategies

We suggest problem-oriented methods and algorithm-animation. The problems should be chosen very carefully. According to Futschek [3] the problems to be solved should not be too simple, and the problem statement should be easily understandable. More complex problems give more space to creativity. With respect to visualization, Shu [12] also considers programming to require both parts of the brain, and focuses on the need to involve the artistic half -expressing the need to involve pictures in the process. van Dam [15] points out that 60% of our neurons are located in the visual cortex.

3. Conclusions

The presented teaching-learning project is built on three central concepts of the modern education: multisensory-learning, multiculturalism and interdisciplinarity. Some of the multi-sensory software-tools we described make it possible for students to feel the pulse of the algorithms. They promote learning with the whole body. More senses involved does not only mean more information, better perception, more efficient memorizing and deeper understanding, but ensures, also, the same chance for students with different dominant senses.

A recent report of the Council of Europe Parliamentary Assembly (Committee on Culture, Science and Education) stated that society’s need for precisely those competences and qualities that are developed through artistic and cultural education is greater than ever. The project illustrates how the unity-in-diversity ideal (the European Union’s motto) can be implemented in educational context: multicultural artistic performances promote the cause of universal science. The method we suggest for teaching-learning sorting algorithms illustrates how arts can be integrated in science education at all levels (scientific content in artistic frame).

According to the suggestions of Schaffer et al. [11], the science-art combination is strongly recommended (among others) for the infusion of energy and excitement that can make students more receptive to learning. They observed that students for whom to concentrate can be difficult were very engaged in lessons that integrated dance, and they also enjoyed it.

Integrating the presented CS course in the human sciences curricula will expectedly result in more efficient education and, consequently, in more skillful professionals who are more prepared to face the challenges of the twenty-first century.

References

[1] Council of Europe, Committee on Culture, Science and Education, Cultural education: the promotion of cultural knowledge, creativity and intercultural understanding through education, Retrieved May 3, 2010, from The Parliamentary Assembly Web Site, 2009, `http://assembly.coe.int`.

[2] Eisenhower SCIMAST, How can research on the brain inform education, *Classroom Compass* **3(2)**, no. 1–2, 10, Retrieved May 3, 2010, 1997, `http://www.sedl.org/pubs/classroom-compass/cc_v3n2.pdf`.

[3] G. Futschek, *Algorithmic Thinking: The Key for Understanding Computer Science*, Lecture Notes in Computer Science, 4226, Springer-Verlag, 2006, 159–168.

[4] H. Gardner, *Frames of mind (The tenth anniversary ed.)*, Basic Books, New York, 1993.

[5] H. Gardner, *Intelligence reframed*, Multiple intelligences for the 21st century, Basic Books, New York, 2000.

[6] Z. Katai, K. Juhasz and A. K. Adorjani, On the role of senses in education, *Computers & Education* **51**, no. 4, 2008, 1707–1717.

[7] Z. Katai, Multi-sensory method for teaching-learning recursion, *Computer Applications in Engineering Education*, 2009, 1061–3773, doi:10.1002/cae.20305.

[8] Z. Kátai and L. Toth, Technologically and artistically enhanced multi-sensory computer programming education, *Teaching and teacher education* **26**, 2010, 244–251.

[9] National Research Council, *Fluent With Information Technology*, Retrieved May 3, 2010, National Academy Press, 1999, `http://www.nap.edu/html/beingfluent/`.

[10] P. Navrat, Hierarchies of programming concepts: abstraction, generality, and beyond, *ACM SIGCSE Bulletin* **26**, no. 3, 1994, 17–21.

[11] K. Schaffer, E. Stern and S. Kim, *Math dance with Dr. Schaffer and Mr. Stern (Prelim. ed.)*, Santa Cruz, CA: MovespeakSpin. Retrieved May 3, 2010, 2001, `http://www.mathdance.org`.

[12] N. C. Shu, *Visual Programming*, Van Nostrand Reinhold Co., New York, 1988.

[13] J. D. Staley, Imagining the multisensory classroom, *Campus Technology*, 2006, Retrieved May 3, 2010, `http://campustechnology.com/articles/40941`.

[14] J. Stevens and D. Goldberg, *For the learners' sake: A practical guide to transform your classroom and school*, AZ: Zephyr Press, Tucson, 2001.

[15] A. van Dam, *Exploratories: From Algorithm Animations and Interactive Illustrations to Explorable Microworlds*, SIGCSE Technical Symposium, Austin, Texas, Number: UMI Order No. GAX95-09398, University of Washington, 2000.

ZOLTÁN KÁTAI
SAPIENTIA UNIVERSITY
MATHEMATICS AND INFORMATICS DEPARTMENT
TÎRGU MUREŞ, ROMANIA

E-mail: `katai_zoltan@ms.sapientia.ro`

(Received November, 2010)