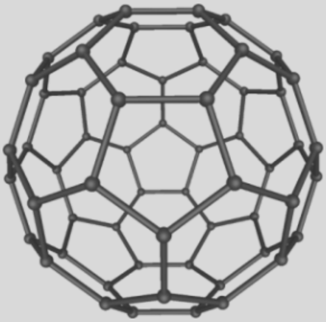
Proceedings of XXXIV. DidMatTech 2021 Conference

New Methods and Technologies in Education, Research and Practice

Eötvös Loránd University (ELTE)
Faculty of Informatics



Editors
dr. Andor Abonyi-Tóth, PhD., Eötvös Loránd University, Budapest, HU
prof. Ing. Veronika Stoffová, CSc., Trnava University in Trnava, SK
doc. dr. hab. László Zsakó, PhD., Eötvös Loránd University, Budapest, HU

ISBN 978-963-489-413-1

SPREADSHEET AS AN ALGORITHM VISUALIZATION TOOL

Gábor TÖRLEY, Péter BERNÁT, HU

Abstract: This paper first presents the definition of algorithm visualization (AV) and its potential in increasing students' engagement in the learning process according to Bloom's taxonomy. Then it demonstrates in detail that not only can spreadsheet develop and support computational and algorithmic thinking, but it can also be used as an AV tool. Authors give examples as well on how learners can reach increasing levels of engagement using spreadsheet as an AV tool.

Keywords: spreadsheet, algorithm visualization, computational thinking, algorithmic thinking, programming

1 Introduction

Among the aims of the Hungarian primary and secondary education, the development of students' cognitive skills and the improvement of their thinking have received more and more attention. Real life creates real problems that is why conscious thinking is needed in order to find a quick and efficient answer for everyday problems. Acquiring the ability of computational and algorithmic thinking also provides help in reaching this goal.

Educational programming can take a decisive role in improving the students' cognitive skills by teaching basic algorithms; however, based on past experiences – abroad included [6, 7] –, it is rather complicated to learn and teach algorithms.

One of the strengths of AV can be involving more sensory organs in learning. Kátai et al [8, 10] confirmed that if a teaching method impacts on different sensory organs, it can effectively support the teaching and learning of algorithms. Furthermore, AV tools can be used to develop algorithmic thinking not only in computer science students. [9]

2 Literature review

Algorithm visualization (AV) is a subclass of software visualization, and it handles the illustration of high level mechanisms of computer algorithms, usually in order to help students understand better how algorithms work. [3]

Naps et al expanded [1] the conclusions of Hundhausen et al [3] i.e. a student who is actively engaged with the visualization technology has consistently outperformed learners who passively view visualizations. A taxonomy was defined which determines the level and type of the students' activity. Six levels were stated:

1. No viewing: in this case AV is not used at all.
2. Viewing: students only look at the execution and the steps of AV.
3. Responding: students are presented with questions during the visualization.
4. Changing: students can alter data or make other changes during the visualization.
5. Constructing: students construct the algorithm's visualization themselves.
6. Presenting: students explain the algorithm using visualization and ask for feedback and discussion.

This taxonomy relates to Bloom's hierarchy [11], which is possibly one of the best known and most widely used models of human cognitive processes. A revised version of the taxonomy was published in 2001 (Figure 1.) [2].

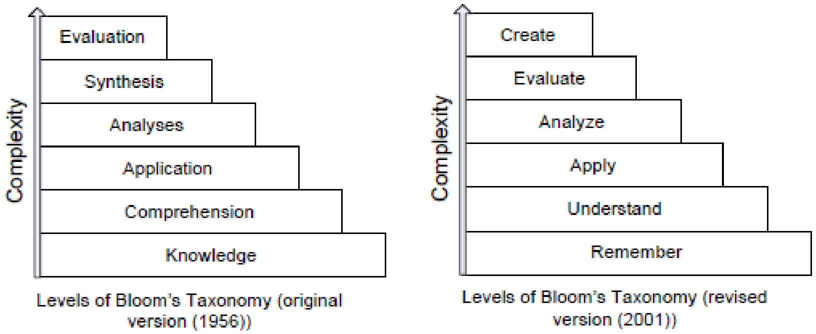


Figure 1: Bloom's Taxonomy, Original and Revised [2, 11]

Learning taxonomies can help description and categorization in cognitive, affective, and other dimensions, in which an individual operates as part of the learning process. In other words, learning taxonomies help us to "understand about understanding". [12]

The new Hungarian Curriculum Framework 2020 [13] adopts an approach according to which spreadsheet is a new topic in the field of "Developing problem-solving skills". Unfortunately, we must note that the new coursebook for spreadsheeting (Grade 9) does not follow this concept. The old tool-centered method is presented instead.

The spreadsheet teaching field includes basic programming concepts [14] like data types, operations, variables, and functions. Tort also suggests adding procedures, scopes of variables, data tables, sorting, etc. because a spreadsheet can be considered as a program and building a spreadsheet is partly programming. If we use a model of a spreadsheet explicitly, then we can help learners in the process of abstraction, which is a very important part of programming.

According to Szalayné [4], a table or a spreadsheet can be considered as a program with data and pre-defined algorithms. Although students can see a table or a spreadsheet on their screens, they need to understand the "program", which consists of their solutions implemented by functions.

Csernoch and Bíró claim [5] that a spreadsheet software can be used as a problem-solving tool. Their method, called Sprego, "is a deep approach metacognitive problem-solving environment, which has borrowed and combined proven methods from high level programming languages. The three milestones of Sprego are

- using as few and as simple general-purpose functions as possible,
- building multilevel formulas,
- building array formulas." ([5] p. 27)

This method can develop students' computational thinking and algorithmic skills. Teaching spreadsheet has an important role in ICT education because students learn several aspects of computer science and develop skills connected to this field, for example, handling data structures, database management, programming principles, logical and computational thinking, and algorithmic skills. Sprego also promotes schema construction through authentic problem-solving and algorithm construction [15].

3 AV in Spreadsheet

Our study connects these two fields above: AV supports demonstration, spreadsheet develops algorithmic thinking. Spreadsheet can visualize the input, the output and the state of the output variable at each step of the algorithm. This means that spreadsheet can show the whole state space (i.e., input, output, and local variables). A spreadsheet software is available for every student, and since this topic is included in the curriculum, students can have the required knowledge. We will show that great programming knowledge is not required in order to create an interactive environment in spreadsheet and by creating the visualization, the student will use algorithmic thinking.

Basic algorithms can be demonstrated in three different levels in spreadsheet:

1. with the appropriate built-in functions, students can become familiar with the concept of programming theorems;
2. using spreadsheet as an algorithm visualization tool, students can understand how programming theorems work;
3. most programming theorems can be implemented using array formulas based on the postconditions of their specifications.

In our paper, we will focus on level 2.

4 Programming theorems and the Counting algorithm

During programming, using algorithm patterns can give general and basic solutions to recurring problems, such as, how many elements have a given attribute in a data sequence, or which is the greatest (or smallest) element of that. In Hungarian programming education, algorithm patterns which relate to one or more data sequences are called "Programming theorems". [16] We will use this term in our paper. Most of modern programming languages include solutions for these problems as built-in functions but we feel strongly that students must understand and create the appropriate basic algorithms by themselves.

According to the new coursebooks based on the new curriculum [13], students learn the concept of programming theorems at Grade 10. Our introductory programming courses at Eötvös Loránd University are also based on the programming theorems. That is why we suggest using our AV examples, presented below, at secondary school (from Grade 10) and at university for students who are familiar with the basic concepts of programming (such as variables and control structures) and can also read pseudo code.

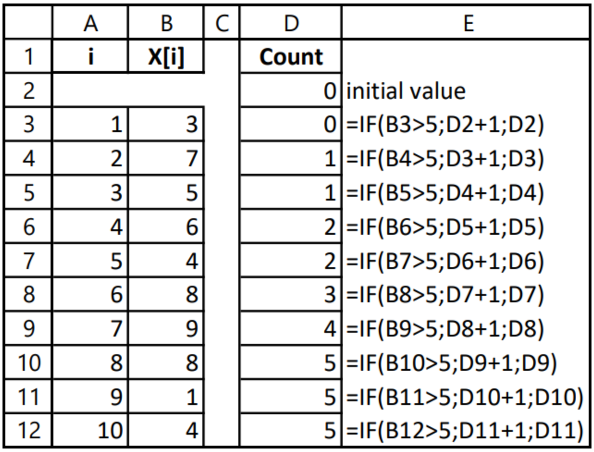
We have chosen the Counting programming theorem as an example. We will demonstrate two kinds of AV of this algorithm: a basic one and an advanced one. Our sample task is the following. At a ski resort, we measured the snow depth in centimeters on some consecutive days. Let us determine the number of days on which the measured value was greater than 5. We will store data (numbers) in a sequence. The algorithm follows the everyday method with which we count objects with a specific attribute. We ask a yes/no question (Does the current object have the specific attribute?) for each object, and if the answer is yes, then we write a mark on a paper, and finally, we count the marks. (Or we can also keep the current count in mind after each yes/no question). Accordingly, the algorithm of the programming theorem uses one auxiliary variable whose initial value is 0 and the algorithm checks all the elements of the data sequence from the beginning to the end and if the current element has the given attribute, then the auxiliary variable is incremented by one. After traversing the whole sequence, the final result can be found in the auxiliary variable.

```
Counting(N, Array, Count)
Begin
  Count := 0
  For i:=1 to N do
    If Array[i]>5 then Count := Count + 1
  End For
End.
```

It is very simple to create the Basic AV in spreadsheet. It will demonstrate the main steps of the programming theorem as a still picture. In order to create the Advanced AV, advanced spreadsheet skills (complex (nested) formulas, conditional formatting, lookup functions) will be necessary. It will demonstrate all the basic steps of the algorithm as an animation, that is why it can support creating the code and novice students can have a more understandable and more spectacular tool.

5 Basic AV

A sequence with 10 integer elements is given on the worksheet (B3:B12), indices are added to the elements (A3:A12) (Figure 2). Our task is to count the numbers that are greater than 5. By copying a formula, let us demonstrate the current value of the above-mentioned auxiliary variable after examining each element of the sequence. (D3:D12)!



	A	B	C	D	E
1	**i**	**X[i]**		**Count**	
2				0	initial value
3	1	3		0	=IF(B3>5;D2+1;D2)
4	2	7		1	=IF(B4>5;D3+1;D3)
5	3	5		1	=IF(B5>5;D4+1;D4)
6	4	6		2	=IF(B6>5;D5+1;D5)
7	5	4		2	=IF(B7>5;D6+1;D6)
8	6	8		3	=IF(B8>5;D7+1;D7)
9	7	9		4	=IF(B9>5;D8+1;D8)
10	8	8		5	=IF(B10>5;D9+1;D9)
11	9	1		5	=IF(B11>5;D10+1;D10)
12	10	4		5	=IF(B12>5;D11+1;D11)

Figure 2: Basic algorithm visualization
for the counting programming theorem

A copyable formula needs to be created, which should be based on the previous element of the sequence. Before the first element, we use the initial value of 0. Then, this value (Count) should be incremented every time when the current value (X[i]) has the given attribute. The appropriate formula can be seen in column E.
This solution corresponds to the algorithm of the programming theorem in details: value 0 in cell D2 can be comparable to the initialization of the auxiliary variable, copying the relative reference of the formula refers to the traverse of the sequence and IF function refers to the branch (if conditional) in the loop.
With this Basic AV created in spreadsheet, students can reach all the levels of the earlier mentioned Engagement Taxonomy [1].

5.1 Viewing

The values of the auxiliary variable can be seen for every step of the algorithm (at the same time). If we copy the formula line by line we can see the change of this variable as an animation and we can control its speed easily.

5.2 Responding

Teacher can ask questions in written form about the value of the auxiliary variable. For example, what will the value of variable Count be at specified step of the algorithm? At which step of the algorithm change the variable Count to a specified value? etc.

5.3 Changing

Thanks to the spreadsheet as a developing environment, the elements of the sequence can be modified easily. In order to change the given attribute, it is enough to modify the logical condition in the formula. We can motivate the modification by asking questions. For example, let us modify the sequence in order to get a specified value in variable Count at the end of the algorithm. Of course, questions can refer to modifying the given attribute as well.

5.4 Constructing

We have chosen spreadsheet as a developing environment because students know this tool consequently they can create their own AV in that, with or without support of the teacher. It is a great advantage that for creating the "Basic AV" needs similar (and not more complicated) concepts to those that we would use at programming.

5.5 Presenting

The well-known environment and the few and easy tools which are needed for the visualization support the presentation of the algorithm. Students can copy the above-mentioned formula (column E on Figure 1) as a demonstration. Moreover, they can create the formula during the presentation and can explain the required steps. Of course, they can change the sequence, its size or the given attribute during presentation and can ask questions in order to engage their audience.

6 Advanced AV

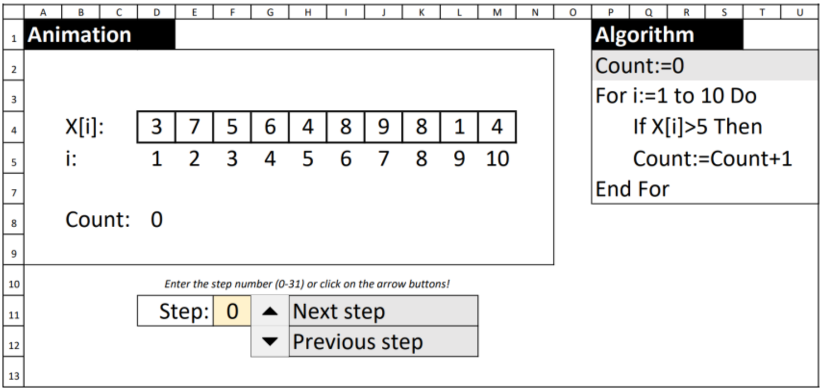
Although advanced spreadsheet skills (complex (nested) formulas, conditional formatting, and lookup functions) are required to create the algorithm visualization described below, it provides additional features compared to the basic version (Basic AV).

A ten-element sequence of positive integers is given on the worksheet (D4:M4), each element of which is numbered (D5:M5) (Figure 3). The task is the same as earlier, let us find the number of elements greater than 5. Let us create an animation that demonstrates the operation of the required algorithm and can be played step by step clicking on a spin button (G11:G12)!

During the animation,

1. a black arrow should point to the current element of the number sequence (D3: M3);
2. the background color of the current element should be gray when selected, yellow when tested, and then green or red depending on whether it had the attribute in question (and it should turn back to white when selecting the next element) (D4: M4);
3. the current value of the auxiliary variable should continuously be visible!

Then, let us display the pseudo code or the program code of the programming theorem next to the animation area, and highlight the currently executed instruction with the background color that the current element has in the animation!



Animation

X[i]:	3	7	5	6	4	8	9	8	1	4
i:	1	2	3	4	5	6	7	8	9	10

Count: 0

Enter the step number (0-31) or click on the arrow buttons!

Step: 0

Algorithm

```
Count:=0
For i:=1 to 10 Do
    If X[i]>5 Then
    Count:=Count+1
End For
```

Figure 3: Advanced algorithm visualization for the counting programming theorem

In the implementation, the current step number is stored in cell F11, which value goes from 0 to 31 with the spin button. The step numbers were chosen arbitrarily as follows.

In step 0 the auxiliary variable is set to 0. In step 1, the first element of the number sequence is selected (and gets a black arrow and a gray background color).

In step 2 the element is tested (and gets a yellow background color), and in step 3 the decision is made (green or red background color), and if the answer is yes to the yes/no question, the auxiliary variable is increased by one. In step 4, we move on to the second element, and so on (Figure 4).

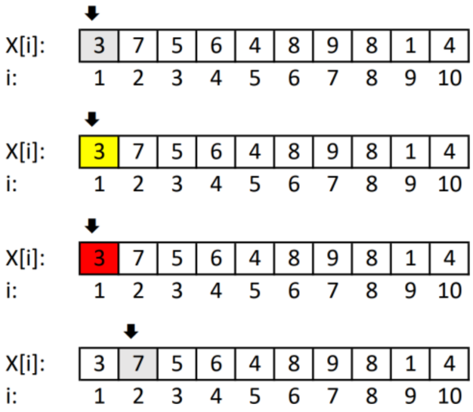


Figure 4: Animation in step 1, 2, 3, and 4

Thus, in step 30, the last element is the current one, and the solution of the original problem can be read from the auxiliary variable (D8). In the very last step (step 31), traversing of the number sequence terminates (the black arrow and the green or red background color disappear).

Conditional formatting was used in the animation, as well as in the pseudo code to highlight the currently performed instruction.

In each cell of range D3:M3, there is an arrow symbol with white font color, from which the only one that points to the current element of the number sequence is displayed in black by the help of conditional formatting. The index number of the current element of the number sequence can be easily calculated from the current step number of the animation.

The background color of range D4:M4 (in which the number sequence is located) is conditionally formatted depending on the current step number of the animation, and the value and the index number of the current element of the number sequence.

Each line of the pseudo code in range P2:U7 is formatted by a separate conditional formatting rule, which depends on the current step number of the animation and whether the tested element has the specified attribute (greater than 5).

Finally, the current value of the auxiliary variable is determined in D8. To this, in a hidden row (row 6) under each element we calculate its temporary value based on the elements so far and the specified attribute just like we did with the basic AV earlier. Then, in D8, we select the proper value from the hidden row according to the current step number of the animation.

Regarding how each level of the Engagement Taxonomy can be accomplished using the algorithm visualization in question, we can say the following.

6.1 Viewing

At the beginning, the pseudo code or the actual program code can be hidden to direct the attention to the animation. We believe that this animation demonstrates the operation of the algorithm in a clearer and more detailed way than the basic visualization. The animation can be viewed at any pace, which can be dictated by the teacher during the presentation and then by the students as they solve tasks. If necessary, they can also jump back to the previous step. During coding, the pseudo code or the program code can be displayed, allowing students to see the algorithm visualization and the algorithm itself at the same time. Because the currently executed step is highlighted in both the animation and the program code, it is easy to determine the relationship between each step of the animation and each instruction in the code.

6.2 Responding

Similar to the first visualization, teacher can ask questions verbally or in writing about how the algorithm works. The question can be about the value of the auxiliary variable, or the animation itself: for example, where the black arrow will be (move) in the next step, and which element will get a background color (and what color).

6.3 Changing

The elements of the input sequence can be easily changed in this visualization as well, but the examined attribute must be modified in several formulas, making it more cumbersome. Otherwise, changes can be motivated with the same questions that we have already outlined in the section on our basic algorithm visualization.

6.4 Constructing

The creation of this algorithm visualization requires advanced spreadsheet knowledge and a longer time (much longer than comprehending the programming theorem in question and writing the proper program code), that is why we recommend this visualization primarily to use and not to create in the class.

6.5 Presenting

At the same time, not only the teacher but also the students can use this visualization very well to present the programming theorem, taking advantage of the already mentioned possibilities of the visualization.

7 Conclusion and future work

In our paper, we have showed the benefits of spreadsheet and AV in order to develop algorithmic thinking, and we linked these two fields together.

We have introduced two types of AV in spreadsheet: an easier one e.g., Basic AV and an advanced one e.g., Advanced AV. With both AV, students can reach all the levels of the Engagement Taxonomy, but there are some differences between them.

Basic AV is very simple. Basic spreadsheet and programming knowledge is enough to use and understand it. Using this kind of visualization, it is easy to understand the current state of the algorithm (at a specific step). It is easy to change and create new visualization as well. That is why we recommend this type of AV for students who have basic knowledge of spreadsheet and programming.

Advanced AV is a better tool for viewing and presenting an algorithm. Its creation and modification (except changing the values of the sequence) needs advanced spreadsheet skills. We recommend this type of AV if the animation and the "output" of the animation is important for the students and the teacher.

We plan to do an empirical study and try out Basic and Advanced AV in our course Programming fundamentals at Eötvös Loránd University. Our students have course on spreadsheet parallel, so we may see how these two fields (programming and spreadsheet) can have impact on each other and how this teaching method can develop students' algorithmic skills.

References

1. NAPS, T. L. – ROSSLING, G. – ALMSTRUM, V. – DANN, W. – FLEISCHER, R. – HUNDHAUSEN, C. D. – VELAZQUEZ–ITURBIDE, J. EXPLORING THE ROLE OF VISUALIZATION AND ENGAGEMENT IN COMPUTER SCIENCE EDUCATION. *SIGCSE BULLETIN*, 2003, 35(2), 131– 152
2. ANDERSON, L. – KRATHWOHL, D. – AIRASIAN, P. – CRUIKSHANK, K. – MAYER, R., PINTRICH, P. – RATHS, J. – WITTROCK, M. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Complete Edition. Longman 2001

3. HUNDHAUSEN, C. – DOUGLAS, S. A – STASKO, J. T. *A meta-study of algorithm visualization effectiveness*. Journal of Visual Languages and Computing, 2002
4. SZALAYNÉ TAHY, ZS. How To Teach Programming Indirectly – Using Spreadsheet Application. Acta Didactica Napocensia 9 (1), 2016, 15-22, ISSN 2065-1430
5. CSERNOCH, M. – BIRÓ, P. *Sprego programming*. Spreadsheets in Education (eJSiE): Vol. 8: Iss. 1, Article 4., 2015 https://sie.scholasticahq.com/article/4638-sprego-programming (Retrieved: 19.07.2021.)
6. LATTU, M. - MEISALO, V. - TARHIO, J. A visualisation tool as a demonstration aid, *Computers & Education, v.41 n.2, p.133-148*, September 2003.
7. URQUIZA-FUENTES, J. – VELAZQUEZ-ITURBIDE, J. Toward the effective use of educational program animations: The roles of student's engagement and topic complexity, *Computers & Education, vol. 67, pp. 178 – 192*, 2013
8. KATAI, Z. – JUHASZ, K. – ADORJANI, A. – K. On the role of senses in education, Computers & Education (2008), Vol. 51, No 4, 1707-1717, ISSN: 0360-1315
9. KATAI, Z. – KOVACS, L. I. – KASA, Z. – MARTON, GY. – FOGARASI, K. – FOGARASI, F. Cultivating algorithmic thinking: an important issue for both technical and HUMAN sciences, *Teaching Mathematics and Computer Science, 9 (2011) 1, 1-10.*
10. KATAI, Z. – TOTH L. Technologically and artistically enhanced multi-sensory computer programming education, *Teaching and teacher education 26 (2010), 244-251.*
11. BLOOM, B. S. – KRATHWOHL, D. R.. *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. Harlow, England: Longmans, 1956
12. SOUSA, D.A. *How the brain learns*. 3rd edn. Corwin Press 2006
13. NAT (2020) National Core Curriculum Framework in Hungary 2020 https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat (retrieved: 22.01.2021.) (in Hungarian)
14. TORT, F. Teaching Spreadsheets: Curriculum Design Principles, *In Proceedings of EuSpRIG 2010 Conference*. ArXiv, abs/1009.2787. https://arxiv.org/ftp/arxiv/papers/1009/1009.2787.pdf (Retrieved: 16.08.2021)
15. CSAPÓ, G. – CSERNOCH, M. – ABARI, K. Sprego: case study on the effectiveness of teaching spreadsheet management with schema construction. *Educ Inf Technol 25, 1585–1605*, 2020.. https://doi.org/10.1007/s10639-019-10024-2
16. SZLÁVI, P. – TÖRLEY, G. –ZSAKÓ, L. Programming Theorems Have the Same Origin. Central-European Journal of New Technologies in Research, Education and Practice, 1(1), 1-12. https://doi.org/10.36427/CEJNTREP.1.1.380

Reviewed by: Dr. Csaba Holló, PhD., Dr. Mária Csernoch, PhD.

Contact address

Gábor Törley, PhD., ORCID: 0000-0002-0496-936
ELTE Eötvös Loránd University, Faculty of Informatics
Address: H-1117 Budapest, Pázmány P. sétány 1/C, Hungary
e-mail: gabor.torley@inf.elte.hu
Péter Bernát, PhD., ORCID: 0000-0002-3759-264X
ELTE Eötvös Loránd University, Faculty of Informatics
Address: H-1117 Budapest, Pázmány P. sétány 1/C, Hungary
e-mail: bernatp@inf.elte.hu