

Using algorithm visualizations in computer science education

Research Article

Slavomír Šimoňák*

*Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic*

Received 12 February 2014; accepted 23 May 2014

Abstract: Algorithm visualization illustrates how algorithms work in a graphical way. It mainly aims to simplify and deepen the understanding of algorithms operation. Within the paper we discuss the possibility of enriching the standard methods of teaching algorithms, with the algorithm visualizations. As a step in this direction, we introduce the VizAlgo algorithm visualization platform, present our practical experiences and describe possible future directions, based on our experiences and exploration performed by means of a simple questionnaire.

Keywords: algorithm visualization • plugin-based visualization platform • computer science education
© Versita sp. z o.o.

1. Introduction and motivation

Algorithms and data structures as an essential part of knowledge in a framework of computer science¹ have their stable position in computer science curricula², since every computer scientist and every professional programmer should have the basic knowledge from the area [1]. With the increasing number of students in Central European's higher education systems in last decades (more concrete numbers and impacts for the case of Slovak one can be found in [2]), introduction of appropriate methods into the process of their education is also required. Our scope here is the higher education in the field of computer science. So within the paper, we discuss the extension of standard methods of teaching algorithms, using the whiteboard or slides, with the algorithm visualizations. According to [3] they can be used to attract students' attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and explore the ideas with respect to their individual needs. Extensive studies on algorithm visualization effectiveness are

* E-mail: slavomir.simonak@tuke.sk

¹ Computer Science Curriculum 2008, Association for Computing Machinery (ACM). Available: <http://www.acm.org/education/curricula>

² Curriculum Guidelines for Undergraduate Degree Programs in Information Technology, IT 2008 Curriculum, Association for Computing Machinery (ACM). Available: <http://www.acm.org/education/curricula>

available nowadays, and results are quite encouraging. A systematic meta-study of 24 experimental studies can be found in [4]. Results of empirical study aimed at the determination of factors influencing the effectiveness of algorithm visualization are published in [5]. Another example is the study with the objective to determine learning advantage of the interactive prediction facility provided by the courseware containing algorithm animations and data structure visualizations [6]. Based on above mentioned reasons, results of studies carried, as well as our own experiences and explorations, we consider algorithm visualization important and perspective area of further research and application of its results in nowadays computer science education.

Except the algorithm visualization, the term software visualization is also often used within the papers published in last years. It usually covers both visualization of algorithms and visualization of data structures, but sometimes also another aspects of software (like its development process) are considered, too [7]. Algorithm visualization, as part of software visualization, could be described as "graphical representation of an algorithm or program that dynamically changes as the algorithm runs" [8]. An overview of visualization taxonomies [9], together with an analysis of factors increasing the effectiveness of software visualization, is summarized in [10].

Even if the beginnings of algorithm visualization date back into 1940's [11], the greatest development in the area we could observe within the last 20–30 years. Modern approaches to software visualization were brought in the 1980's by the introduction of system Balsa (Brown & Sedgewick, Brown University, USA) [12]. Some of contemporary solutions include systems like TRAKLA2³, ANIMAL⁴ [13], JAWAA⁵ or Algorithms In Action⁶. Concise overview of development in the area of software visualization we provided in [14], so it is not our intention to analyse this topic within the paper.

2 The Vivaldi Algorithm Visualization Platform

Based on analysis of existing solutions, we decided to start developing our own algorithm visualization platform named Vivaldi. The motivation behind the decision is described in [15] and it includes the fact, that the platform is intended to be used as a support tool within the subject Data Structures and Algorithms taught in a bachelor study program at the Faculty of Science, Brno University of Technology. The collection of topics within the scope of the subject is quite wide and it could probably be changed over the time. To cover the scope of the subject probably more tools would be used, as quite big collection of collected tool would be required. Taking also possible changes in the subject's structure into account, we believed it was better to start developing and developing our own solution.

There are some specific issues of analysis and design of algorithm visualization systems, as it was described in [16]. Within the section 2 of our paper [15] we tried to give answers at least to most important questions from the said analysis, such as analysis, tool analysis, information analysis and domain analysis part of it.

As a development platform for the project was selected Java, ensuring high portability and very good support by available tools, libraries, etc. Another important decision to make was the selection of software framework to support extensibility. Also the analysis of available solutions for JSP was chosen, designed to reduce the time of development of page-based applications.

2.1 Architecture of the Platform

Basically, we can think of the Vivaldi application as consisting of two cooperating parts. The main module and a set of independent page modules. The main module consists of several classes providing support for controlling the algorithm execution and rendering the algorithm visualization (Figure 1). Other important of them include Vivaldi class providing execution logs, algorithm settings, execution control and driving a cooperation of the main module with page modules. VivaldiPage class providing visualization related services for different components and selection of suitable language, and Rendering class responsible for algorithm control and visualization. The rest of classes

³ TRAKLA2 Software Project Website: <http://rakl2.cba.hawaii.edu/~rakl2/tra2/tra2.html>

⁴ ANIMAL Java Page Website: <http://rakl2.cba.hawaii.edu/~rakl2/animal/>

⁵ The JAWAA Homepage Website: <http://rakl2.cba.hawaii.edu/~rakl2/jawaa/>

⁶ Algorithms in Action Website: <http://rakl2.cba.hawaii.edu/~rakl2/aii/>

⁷ JSP Java Script Page Framework Website: <http://rakl2.cba.hawaii.edu/~rakl2/jsp/>

provide supporting evidence for the application functionality, and their description can be found in [\[10\]](#).



A *plugin module*, on the other hand, contains the code of algorithms to be visualized and can utilize the services provided by the main module. To the plugin module, offering its services has only to fill up predefined data structures (strings, graphical objects, etc.) with the data to be displayed in a process of algorithm visualization and it is the role of main module to display them properly. This helps to keep the code of the module code smaller, as well as the time in deriving a plugin module performing visualization of particular algorithm. Rendering capabilities of the main module are provided by offering the data being library of the module, but it can be changed in the future, if needed. Communication between the main module and a plugin module is performed by calling dedicated methods, externally.

2.2. Developing Plugin Modules for the Platform

Figure 10 illustrates development starts by creating the interface with the name of algorithm to be executed (e.g., `Simplex`). The interface you created will contain two public abstract methods `generateData()` and `execute()` as it is shown in Figure 10.

These methods will be accessible in the plugin module implementing this class (`GraphHogPlugin`) in our code. The class controls the `FeatureType` class mentioned above and implements methods `GraphHog` and `Baseline`. In Figure 2, an essential part of `construct()` method used in this test construction is given. First `BuildHog()`, `Hog()` and `Baseline()` represent methods of the `Hog` test algorithm. First methods called within the code were construction process. For example method `constructBaseline()` highlights the currently working test or score procedure. `constructHog()` method provides following of elements in array to be compared. `GraphHog()` method displays the array data in a form of binary tree (Figure 3). The next essential part of code is `test` method that was added to



Figure 5. HeapSort process and related heapSort step with heap sort visualization.

also of better illustration of algorithm operation.

2.5. Current state

The list of currently available design modules includes sorting algorithms like Bubble sort, Heap sort, Bubble sort, Insertion sort and Selection sort. Visualizations of some basic data structures such as queue and also available and visualizations of sort algorithms are under development at the time.

Upon the start of the application and selection of algorithm to visualize, there is a possibility to enter input data for the algorithm of Figure 5. Otherwise input data are generated randomly.

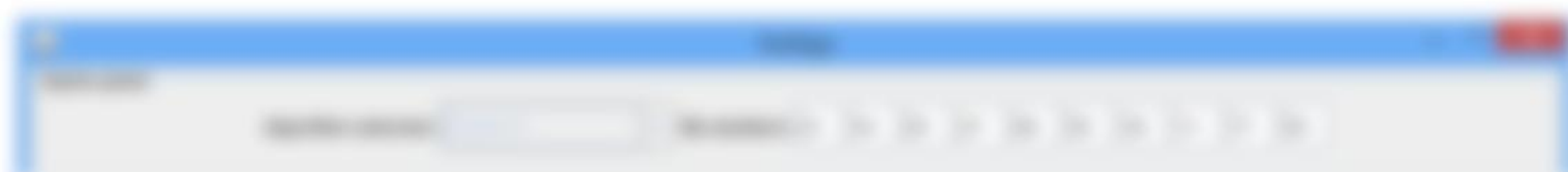


Figure 6. Entering input data to the algorithm window

To illustrate the heap sort process, the above mentioned heap sort visualization is used (Figure 5). The visualization provides multiple views on algorithm operation parallelly with currently executing tree highlighted, array of sorted data, binary tree form of data as it was mentioned yet. Within the binary tree representation, elements sorted yet are displayed with different colors. The rest of elements still to sort is from the heap by calling the `Heapify()` method and picking up the largest one (`Heap()` method) to exchange the sorted part of the array.

Currently, the usage of the tool within the teaching process is mostly connected with sorting algorithms. Issues not covered by the tool yet, are taught by the help of different tools or available web-based visualizations. These complex



Figure 5: Screenshot of the Trifly application with working flow and algorithm visualization.

data structures, the built tables and tree structures, can serve as an example here. A clear visualization of this tool is available in [34], whereas basic principles of building are clearly presented in [35].

3. Research and experiences

The evaluation behind the effect of gathering information from students by means of simple questionnaires was limited. Firstly, we wanted to know how well the tool is accepted by a group of its potential users. Whether our design decisions were right and whether the visualization of algorithms is considered helpful at all.

Secondly, we were curious what new visualizations and new features in general are requested by the users of the system. These opinions could serve as as a motivation and inspiration for decisions within the next development.

Regardless to our case were 10 students from two different groups having completed the Data structures and algorithms subject taught in the second year of the Information Technology study program.

Questionnaires had consisted of five questions. Four two of them were used to get the feedback on the tool in its actual state and its usefulness in a process of teaching algorithms and data structures. Questions from three to five were oriented to the future development of the tool. Written questions three and four, students by their answers were able to express their opinions on the list of algorithms that could be implemented, as well as more generally on the areas of topics from the subject that should be covered by the tool in the future. While these two questions were connected to the development of new or enhancing existing single modules, last question concerns the user interface and its potentially new features.

The language of the questionnaire was Slovak (same as the language in which the subject is taught in participating study groups) and the questions had the following meaning:

1. How using of the Trifly tool help in understanding operation of algorithms?
a) Yes b) No c) Don't know
2. Which of currently available visualizations was most helpful for you?

a) Reflection by thought c) description of Reflection by Reflection

3. Which new algorithms should be included?

Please specify:

4. Which areas covered by the subject require modification more than others?

a) Describing data structures by Algorithm design techniques c) Testing of Abstract data structures by ADT for implementation of Data structures and algorithms for external storage

5. What new features of the test are welcomed?

a) Test bank in testing modification b) Testing mode c) On-line availability of Other features

Summary of responses to subject form and their final analysis follow within the rest of the section.

Table 1. Response to question 1

	Yes	No	Other	Total
Response	18	2	1	21

The opinion of almost all of students (88% from 21) Table 1, about helping in understanding operation of algorithms was positive. Even if we expected it could be useful, we didn't know it will be so close. The result suggests we believe that it is preferable to continue with development of the test and its use within the teaching process.

Table 2. Response to question 2

	Reflection	Thought	description	Reflection	description
Response	2	15	2	18	1

The Reflection modification seems to be the most useful with its 70 votes from students. The second place was modification of thought algorithms. As it could be seen from the vote of students in Table 2, vote of respondents showed more than one algorithm form.

Table 3. Response to question 3

	Algorithm	Test	Length	Testing
Response	12	2	2	2

Most wanted modification is dealing in the use of Algorithm algorithms (Table 3), followed by modification of operations in different levels of tests. It should be noted here, that quite a big part of students did not explain the possibility to improve their opinion and found this question without an answer. Some of answers probably would not be satisfied as they including as much as it is possible, or even all algorithms (probably mentioned in the subject).

Table 4. Response to question 4

	Describing	Algorithm	Testing	ADT	External storage
Response	2	15	8	22	1

Area requiring modification more than others, according the answers in Table 4 are testing (22) and abstract data structures (22). Some of testing algorithms were implemented all over, so this result confirms our decision to start with development of plugin modules for the platform from that area. Abstract data structures in case of the subject include different levels of test structures as built tables.

According to results summarized in Table 5, test bank in testing modification was selected as the most useful feature to implement in the future, followed by the on-line availability of the test. Quite small number of suggestions was provided here within the 'Other features' option (including graphically better modification, optional changing algorithm properties, and better specification of input sets).

Table 2. Responses to question 2

	Very good	Very good, with	Very good, other features
Response	2	2	2

4. Conclusions

According to our findings, algorithm visualisation can be seen as a valuable supporting tool, used in addition to standard ways of education in the field of computer science. Within the paper we provided an overview of the Truffle algorithm visualisation platform as well as our practical experiences with the system. We believe that the results of questionnaire support our belief in a hope to improve the quality of education in the field and contribute to the solution for some of the problems in higher education mentioned at the beginning of the paper.

There are still open issues with using algorithm visualisations. Algorithm visualisations can help understanding the principles, but do not replace the need to implement algorithms by students in a chosen programming language. Further drawback of using algorithm visualisations within our subject is the lack of the tool offering required visualisations in a single package with the related exercises. The Truffle platform can also be considered as a step in this direction. Generally, more systematic evaluation of algorithm visualisation tools is required, as there is rather limited evidence available that applications of algorithm visualisations are useful [3].

We summarised results of the questionnaire filled in by students in order to support our decisions on further development of the platform, too. Our intentions here include development of new plugin modules from the area of sorting algorithms and more complex data structures. Some of proposed user-related features are on the list too like graphically better visualisations, optional changing of algorithm properties, but some of them will probably not be implemented in a near future like extending tool to covering visualisations, as they would require more fundamental changes. Though the intentions mentioned within the questionnaire, we also consider some other interesting features. Dynamic changes in algorithm procedures reflected in visualisation, different visual views on sorting algorithm or simultaneous comparison of different algorithm visualisations.

Acknowledgments

This work was supported by VEGA Grant No. 1/0161/17, Principles and methods of automated abstraction of computer languages and software development based on the semantic verification issued by communication VEGA. This work is also the result of the project implementation ITR7 – Information technology for knowledge transfer project number 20.00.00.00 supported by the Research & Development Operational Program funded by the ERDF (2014).

References

- [1] R. Sedgwick, P. Fowler, *Algorithms and Data Structures* (Springer Verlag, Berlin Heidelberg, 2008).
- [2] J. Guck, *Foundations in Data Science of the Czech Higher Education Platform Using Information Technologies*, in proceedings of 18th IEEE International Conference on Emerging Learning Technologies and Applications, ICETEA 2017, IEEE Press, The High School, Slovakia, November 8-9, 2017.
- [3] V. Klenz, *Designing Effective Algorithm Visualisations*, in proceedings of 17th International Program Visualisation Workshop, ICIPV, Paphos, Cyprus July 7 - 8, 2006. Available: <http://www.cs.cyp.ac.cy/Workshop/ICIPV2006/>
- [4] J. G. Hoffmann, T. A. Dwyer, J. J. Heule, *A Formal Study of Algorithm Visualisation Effectiveness*, in Visualizing Computer Science, pp. 259-266, 2002.
- [5] V. Lascaris, V. Tassara, A. Vlastakis, *An empirical study on factors influencing the effectiveness of algorithm visualisation*, Comput. Appl. Eng. Cybern., 21, 440-448, 2011.
- [6] D. Lee, M. P. Fellenz, R. J. Miller, *Assessing the benefits of interactive problem solving Web-based algorithm animation environments*, Proceedings of ICETEA 2008 (IEEE Press, New York, 2008).

- [2] S. Edelkamp, *Algorithm visualization: Visualizing the Structure, Behaviour, and Evaluation of Algorithms*. Teubner, 2007, 387.
- [3] S. Edelkamp, S. Edelkamp, & M. A. Heuleman-Engels, C. Krumm, *Empirical Evidence that Algorithm Animation Promotes Understanding of Distributed Algorithms*. In proceedings of 40th Symposium on Foundations of Computing, Languages, and Complexity, FOCLAS, Arlington, Virginia, September 2007.
- [4] S. Edelkamp, S. Edelkamp, & M. A. Heuleman-Engels, *A Pragmatic Taxonomy of Algorithm Visualization*. In Visualizing Computing, VIZ, 271-286, 2007.
- [5] J. Edelkamp, *Visualization of Algorithms and Data Structures*. Birkhäuser Series, 2011, 400, 11. ed. Berlin, Birkhäuser Series, 2010 (in German).
- [6] S. Edelkamp (Ed.), *Algorithm Visualization: Lecture Notes in Computer Science* (2006-2007).
- [7] S. Edelkamp, S. Edelkamp, & J. Edelkamp, *A system for algorithm animation*. Proceedings of the 11th annual conference on Computer graphics and interactive techniques, SIGGRAPH 2000, New York, NY, 1-10, 2000.
- [8] S. Edelkamp, & J. Edelkamp, *Edelkamp*. A system for supporting multiple views in algorithm animation. In Visualizing Computing, VIZ, 241-256, 2007.
- [9] S. Edelkamp, *Algorithm Visualization Using the Teaching Platform*. *Acta Cybernetica et Informatica* 1995, 29-40, 2011.
- [10] S. Edelkamp, *Algorithm Visualization*. Birkhäuser Series, 2011, 400, 11. ed. Berlin, 2011 (in German).
- [11] S. Edelkamp, *Acta Cybernetica et Informatica*, 2007, available: <http://www.gmd.fhnw.de/lehre/acta/acta.htm/2007/2007-01-02.html>.
- [12] S. Edelkamp, *Acta Cybernetica et Informatica*, 1998, available: <http://people.cs.gpi.fhnw.de/~edelkamp/acta/acta.htm/1998/1998-01-02.html>.