Cent. Eur. J. Comp. Sci. • 4(3) • 2014 • 183-190
DOI: 10.2478/s13537-014-0215-4

VERSITA

Central European Journal of **Computer Science**

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

1. Introduction and motivation

Algorithms and data structures as an essential part of knowledge in a framework of computer science[1] have their stable position in computer science curricula[2], 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 TRAKLA2[3], ANIMAL[4] [13], JAWAA[5] or Algorithms In Action[6]. 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 VizAlgo Algorithm Visualization Platform

2.1. Architecture of the Platform

provide supporting methods for the application functionality and their description can be found in [15].

Figure 1. The structure of the main module.

A plugin module, on the other hand, contains the code of algorithm to be visualized and can utilize the services provided by the main module. So the plugin module, utilizing its services has only to fill up predefined data structures (strings, graphical shapes, 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 size of the module code smaller, as well as the time to develop a plugin module performing visualization of particular algorithm. Rendering capabilities of the main module are provided by utilizing the Java Swing library at the moment, 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 mutually.

2.2. Developing Plugin Modules for the Platform

Plugin module development starts by creating the interface with the name of algorithm to be visualized (e.g. HeapSort). The interface just created will contain two public abstract methods pseudoCode() and actionLoad() as it is shown in Figure 2.

```
public abstract interface HeapSort extends Plugin {
    public abstract void pseudoCode();
    public abstract void actionLoad();
}
```

Figure 2. An essential part of HeapSort interface.

These methods will be overwritten in the plugin module implementing class (also (HeapSortImpl) in our case). The class extends the ModulPlus classes mentioned above and implements interfaces HeapSort and Runnable.
In Figure 3 an essential part of actionLoad() method used in Heap sort visualization is given. Here BuildHeap(), Heap() and Heapify() represent methods of the Heap sort algorithm. Their methods called within the code serve visualization purposes. For example method colourCodeMarker() highlights the currently executing line in source pseudocode. colourMarker() method provides colouring of elements in array to be swapped. PrintTree() method displays the array data in a form of binary tree (Figure 5). The tree-based point of view on sorted data was added for

```
[illegible]
```

Figure 3. Simplified listing of [illegible] method of [illegible] class used in Heap sort visualisation

sake of better illustration of algorithm operation.

2.3. Current state

The list of currently available plugin modules includes sorting algorithms like Radix sort, Heap sort, Bubble sort, Insertion sort and Selection sort. Visualizations of some basic data structures (stack, queue) are also available and visualizations of next algorithms are under development at this time.
Upon the start of the application and selection of algorithm to visualize, there is a possibility to enter input data for the algorithm (Figure 4). Otherwise input data are generated randomly.

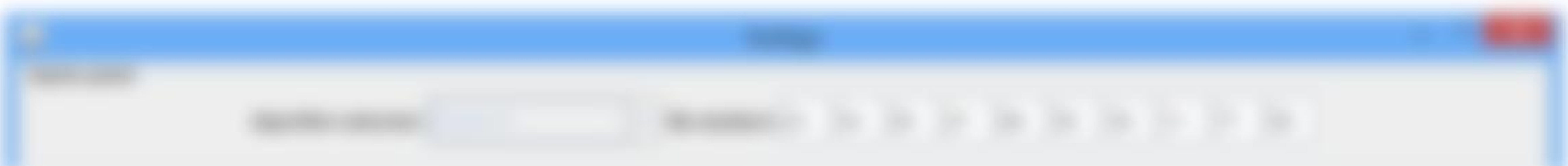

Figure 4. Entering input data for the algorithm selected

To illustrate the tool's interface, the above mentioned Heap sort visualization is used (Figure 5). The visualization provides multiple views on algorithm (algorithm pseudocode with currently executing line 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 colour. The rest of elements still is used to form the heap (by calling the Heapify() method) and picking up the largest one (Heap() method) to enlarge the sorted part of the array.
Currently, the usage of the tool within the teaching process is mainly connected with sorting algorithms. Areas not covered by the tool yet, are taught by the help of different tools or available web-based visualizations. More complex

Figure 5. Screenshot of the VizAlgo application with running Heap sort algorithm visualization.

data structures, like hash tables and tree structures, can serve as an examples here. A clever visualization of RB tree is available in [16], whereas basic principles of hashing are nicely presented in [17].

3. Research and experiences

Our motivation behind the effort of gathering information from students by means of simple questionnaire was twofold. Firstly, we wanted to know how well the tool is accepted by a group of it's 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 expected by the users of the system. Their opinions could serve us as a motivation and inspiration for decisions within the next development.
Respondents in our case were 53 students from four different groups having enrolled the Data structures and algorithms subject, taught in the second year of the Informatics bachelor study program.
Questionnaire itself consisted of five questions. First 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. Within 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) plugin modules, last question concerns the main module 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. Does using of the VizAlgo tool help to understand operation of algorithms?
   a) Yes b) No c) Don't know

2. Which of currently available visualizations was most helpful for you?

a) BubbleSort b) HeapSort c) InsertSort d) RadixSort e) SelectSort

3. Which next algorithms should be visualized?
   Please specify.

4. Which areas, covered by the subject, require visualization more than others?
   a) Elementary data structures b) Algorithm design techniques c) Sorting d) Advanced data structures for ADT Set implementation e) Data structures and algorithms for external storage

5. What new features of the tool are welcomed?
   a) Step back in running visualization b) Testing mode c) On-line availability d) Other features

Summary of responses in tabular form and their brief analysis follow within the rest of the section.

Table 1. Responses to question 1

	Yes	No	Don't know
responses	38	0	2

The opinion of almost all of students (38 from 41, Table 1) about helping to understand operation of algorithms was positive. Even if we expected it could be useful, we didn't know it will be so clear. The result supports our belief that it is purposeful to continue with development of the tool and its use within the teaching process.

Table 2. Responses to question 2

	BubbleSort	HeapSort	InsertSort	RadixSort	SelectSort
responses	6	15	5	35	4

The RadixSort visualization seems to be the most useful with its 35 votes from students. The second place took visualization of HeapSort algorithm. As it could be seen from the sum of numbers in Table 2, some of respondents selected more than one algorithm here.

Table 3. Responses to question 3

	QuickSort	Trees	Graphs	ShellSort
responses	11	5	2	2

Most wanted visualization to develop is the one of QuickSort algorithm (Table 3), followed by visualization of operations on different kinds of trees. It should be noted here, that quite a big part of students did not exploit the possibility to express their opinion and leaved this question without an answer. Some of answers probably would not be satisfied so easy, including 'as much as it is possible', or even 'all algorithms' (probably mentioned in the subject).

Table 4. Responses to question 4

	ElemDS	Alg.techniques	Sorting	AdvDS	External storage
responses	5	11	36	23	4

Areas requiring visualization more than others, according the answers in Table 4 are sorting (36) and advanced data structures (23). Some of sorting algorithms were implemented till now, so this result confirms our decision to start with development of plugin modules for the platform from that area. Advanced data structures in case of the subject include different kinds of tree structures or hash tables.
According to results summarized in Table 5, step back in running visualization was selected as the most useful feature to implement in the future, followed by the on-line availability of the tool. Quite small number of suggestions was provided here within the 'Other features' option (including graphically better visualizations, optional changing algorithm properties, and better specification of input set).

Table 5. [illegible]

	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

4. Conclusions

[illegible]

Acknowledgments

[illegible]

References

[illegible]

[7] S. Diehl, Software visualization: Visualizing the Structure, Behaviour, and Evolution of Software (Springer, New York, 2007) 187

[8] M.D. Tudoreanu, R. Wu, A. Hamilton-Taylor, E. Kraemer, Empirical Evidence that Algorithm Animation Promotes Understanding of Distributed Algorithms, In proceedings of IEEE Symposium on Human Centric Computing Languages and Environments, HCC02, Arlington, Virginia, September 2002

[9] B.A. Price, R.M. Baecker, I.S. Small, A Principled Taxonomy of Software Visualization, J. Visual Lang. Comput. 4(3), 211–266, 1993

[10] Ž. Tuchová, Visualization of Algorithms and Data Structures, Bachelor thesis, DCI FEEI TU of Košice, Bachelor thesis, 2010 (in Slovak)

[11] S. Diehl (Ed.), Software Visualization, Lecture Notes in Computer Science 2269, 2002

[12] M.H. Brown, R. Sedgewick, A system for algorithm animation, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, SIGGRAPH'84 (ACM New York, NY, USA, 1984)

[13] G. Rößling, B. Freisleben, ANIMAL: A system for supporting multiple roles in algorithm animation, J. Visual Lang. Comput. 13(3), 341–354, 2002

[14] S. Šimoňák, Algorithm Visualization Using the VizAlgo Platform, Acta Electrotechnica et Informatica 13(2), 54–64, 2013

[15] A. Tagliy, Algorithm Visualization, Bachelor thesis, DCI FEEI TU of Košice, 2012 (in Slovak)

[16] A. Gangwisch, AVL Tree visualization, 2007, available: http://www.qmatica.com/DataStructures/Trees/AVL/AVLTree.html

[17] J. Morris, Hash Table Animation, 1998, available: http://people.cs.pitt.edu/~kirk/cs1501/animations/Hashing.html