Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

Interactive System for Algorithm and Data Structure Visualization

Patrik Perháč, Slavomír Šimoňák

Abstract

This work is dedicated to the design, implementation and evaluation of a new algorithm visualization system. The currently available systems and libraries are briefly compared with each other based on the visualizations and functionalities they provide. Since the analyzed tools didn't meet all of the given requirements, we decided that the development of a new system for algorithm and data structure visualizations would be beneficial for use in teaching the subject Data Structures and Algorithms. The new system was designed to be easily usable, extensible, available and to cover the basic functionalities available in similar systems and other useful features. The proposed system provides three types of visualizations: predefined visualizations, to explain how each data structure and algorithm works; interactive visualizations, to let the user interact with the visualization directly; and interactive exercises, to let the users test their knowledge. These three types of visualizations cover the whole learning process, provide theoretical and practical knowledge, and also a way to test their knowledge. The system is implemented in the form of a web application and, for the visualizations, the JSAV library is used. The system was also evaluated by the users via a survey and several improvements were implemented in the system based on the feedback provided by the users.

Keywords: algorithms, data structures, visualization, web application, JavaScript, JSAV.

MSC 2010: 68P05, 68P10, 68Q65, 97U50.

1 Introduction

In this work, we focus on making the learning process of data structures and algorithms easier for the students. These topics are thaught in the subject Data Structures and Algorithms. This subject is taught in the second year of the bachelor's degree program Informatics of the Department of Computers and Informatics. In this subject, different abstract data types and algorithms are introduced to the students. Understanding how some of the more complex data structures and algorithms work can be harder to comprehend for some students at first. However, this process can be made easier by the use of supporting tools that help visualize the data structures, so students can understand them better. Different kinds of supporting tools were in use while teaching the subject Data Structures and Algorithms, each of which had its benefits and limitations. These tools are evaluated later in this paper.

The use of visualizations alongside with pseudocode, in which the currently executed line is highlighted can help significantly in understanding more complex data structures and algorithms [1]. This way the student has an overview of how each step of the algorithm affects the data structures used [2]. Another way to ensure the students get the most out of learning these topics is to provide them with engaging activities besides the visualizations of the data structures and algorithms [3]. Just looking at the visualizations may not be sufficient enough to fully understand the topic, but interacting with the data structures directly can help students gain a better, more in depth understanding. This is why, besides predefined visualizations, it is advantageous if the user has the ability to interact with the visualization in some way, either by defining the order of the operations executed, or defining the values used within the visualized data structures. This can help the user gain some practical knowledge too [4].

1.1 Requirements for the supporting tool

Before we select the supporting tool to be used in the teaching of the subject Data Structures and Algorithms, we need to define the require-

ments it needs to satisfy. These requirements need to be adjusted to the learning process and the material thought in this subject. Initially, there were two main requirements for the system:

- *Usability:* the system must be easy to understand and simple to use.
- *Accessibility:* the system must be widely accessible, and the initial configuration or setup of the system should not take more than a few minutes.

After defining these requirements, a more detailed specification was designed with the following parts [5]:

- *Online solution:* designing the system in the form of a web application and deploying it to a web server solves the accessibility requirement, since the application would be accessible to anyone.
- *Clear visualizations:* the visualizations included in the system should be easy to understand and interactions with the visualizations should be intuitive and straightforward.
- *Brief explanations:* the system should include a brief overview of the visualized algorithm or data structure.
- *Pseudocode:* for more complex visualizations, pseudocode should be provided to help better understand the visualized algorithm. The currently executed line should be highlighted in the pseudocode.
- *Controls:* a way to play the visualizations step by step in any direction should be provided [2].
- *Defining elements:* the user should be able to define the value of the data structure elements.
- *Extensibility:* the system must be easily extensible with new visualizations.
- *Predefined visualizations:* the system must provide predefined visualizations for the basic data structures.
- *Exercises:* the system should allow students to test their understanding of the visualized algorithms [6].

2 Related work

This section provides a brief overview of some of the existing systems that were analyzed in this work. The analysis included desktop applications previously used in teaching the subject Data Structures and Algorithms as well as some of the available web applications.

2.1 VizAlgo

VizAlgo is a desktop application for algorithm and data structures visualizaton, used in teaching of the subject Data Structures and Algorithms, written in the Java language [7]. Since it is written in Java, it doesn't require installation, but it has limited compatibility in Linux systems due to the libraries used, some of which need to be additionally installed on Linux systems. It provides visualizations for the basic data structures such as the Stack, Queue, Binary Search Tree. Sorting algorithms, tree traversal algorithms, MinMax and Chainmatrix algorithms are also visualized in the application [8]. Because of the need of obtaining a *.jar* file to run the program and the limited compatibility with Linux systems, VizAlgo didn't meet the requirements specified above.

2.2 Algomaster

Another desktop application analyzed was Algomaster which is also used in teaching the subject Data Structures and Algorithms. It's built on the .NET platform and provides functionalities like stepping the visualization, the ability for the user to define the data structures, pseudocode, and also call stack visualization, which is useful for visualizing recursive algorithms [9] [10]. The visualizations are implemented in the form of plugins [11]. Algomaster includes more visualizations of data structures and algorithms than VizAlgo, but since it is built on the .NET platform, it has limited compatibility.

2.3 Online learning tools

Besides the previously mentioned desktop applications, some of the online solutions available were also analyzed and evaluated based on the requirements.

2.3.1 Algorithm Visualizer

This online platform[1] provides visualizations directly from source code. The source code of the included visualizations can be edited and run after successful build. Since the visualization is generated from the source code directly, it is suitable for visualizing complex algorithms. However, this feature makes the code more complicated since the code doesn't only contain the code for the algorithm, but also the code responsible for visualizing the algorithm, which could lead to confusion while learning basic data structures and algorithms.

2.3.2 VisuAlgo

VisuAlgo[2] provides pseudocode for the visualizations as well as explanations for each crucial step of the visualization. The values in the data structures can be defined by the user, and stepping the visualization forward and backward is also supported. VisuAlgo is overall a great learning tool, but it's license doesn't allow it to be modified, which makes it not suitable for the purposes of this work.

2.3.3 Data structure visualizations

This system[3] uses its own visualization library, which uses the HTML canvas [12]. Visualizations for most basic data structures and algorithms are provided in this system, however it lacks features like the display of pseudocode or explanations for the individual steps of the algorithm. The more complex visualizations are also harder to follow

[1] Algorithm Visualizer: `https://algorithm-visualizer.org/`

[2] VisuAlgo: `https://visualgo.net/`

[3] Data structure visualizations: `https://www.cs.usfca.edu/~galles/visualization/Algorithms.html`

because of this, which makes it not suitable for the purposes of this work.

2.4 Summary

The analyzed desktop applications VizAlgo and Algomaster offer visualizations for a wide variety of data structures and algorithms, but they do not meet the requirements in terms of availability, since both applications require downloading an executable file, and have limited compatibility with operating systems other than Windows.

The analyzed online learning tools were either too complex for the teaching of basic data structures or algorithms (Algorithm Visualizer), or their license doesn't allow them to be customized for use in teaching the subject Data Structures and Algorithms (VisuAlgo). For these reasons, the development of a new system seems to be a preferable option.

3 Data structure visualization library

Since none of the analyzed learning tools met all the defined requirements, the creation of a new application, designed specifically for supporting the teaching process within the Data Structures and Algorithms subject was needed.

A framework or library was needed for implementing the visualizations in the proposed new application. For this purpose, three possibilities were compared:

- *Data structure visualizations* – the library used in the online learning tool with the same name. This library provides basic functionalities, but is not flexible enough to be used for the purposes of this work.
- *Custom library for data structure visualizations* – the creation of a custom visualization library built on the CreateJS library[4] was also considered.

[4]CreateJS: `https://createjs.com/`

- *JSAV* – provides visualizations for basic data structures, supports the creation of interactive exercises, and covers all functionalities needed for the application proposed in this work with little to no modifications.

The JSAV[5] library was selected for creating the visualizations in the new application, since it provides visualizations for the basic data structures like *array*, *list*, *graph*, *tree* or *matrix*. JSAV also supports stepping the visualization both forwards and backwards, displaying explanations for each step, displaying pseudocode with the possibility to highlight certain lines in each step. Another useful feature is the Exercise API, which helps creating interactive exercises for the students to test their knowledge of the visualized algorithm. These exercises are evaluated in real time based on a model solution which needs to be defined in advance.

4 Application design

The application that is the result of this work was designed with all the requirements in mind. It's implemented in the form of a web application, which fulfills the accessibility requirement. The usability requirement is covered by the design of the user interface of the application [13]. The application is implemented using *NuxtJS* [6], which is a framework for creating Vue applications. The layout of the application consists of three main parts, as seen on the screenshot of Figure 1.

4.0.1 Left sidebar

The left side contains the main navigation menu, where the included algorithms and data structures are listed and can be selected. The menu is implemented with the help of the *vue-sidebar-menu*[7] component.

[5] JSAV: `http://jsav.io/`

[6] NuxtJS: `https://nuxtjs.org/`

[7] vue-sidebar-menu component: `https://www.npmjs.com/package/vue-sidebar-menu`

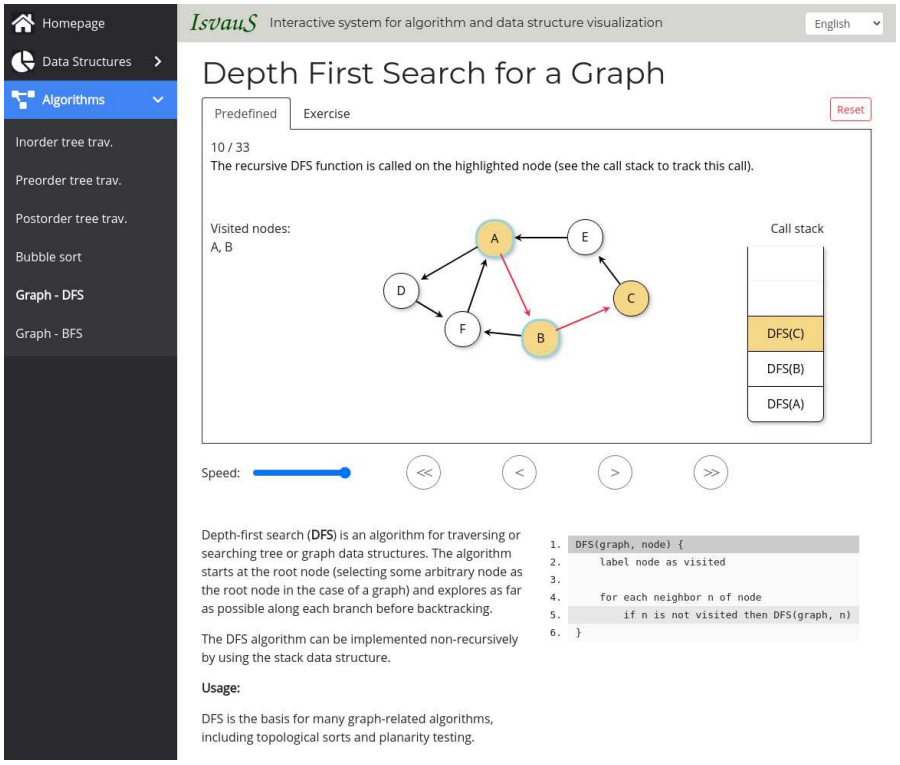

Figure 1. Screenshot of the application created as part of this work

The menu items are generated dynamically from the configuration files of the visualizations, which means that after a new visualization is added to the application the menu does not need to be changed, the new visualization is added automatically.

4.0.2 Visualizations

This is the main part of the webpage in which the visualization is displayed. The user can select which type of visualization should be displayed using the tabs beneath the title (see Figure 1). Each visualization of a data structure or algorithm can support any or all of the

three types of visualizations:

- *Predefined visualizations* consist of predefined steps containing detailed explanations for each step, a description of the selected data structure or algorithm, and pseudocode, in which the line that is currently being executed, is highlighted. Controls for the visualization are also provided, so the user can step through or jump to the beginning or the end of the visualization.

- *Interactive visualizations* allow the user to interact with the visualization directly using controls that are specific to each visualization. These interactions can include executing an operation over a data structure with user defined parameters or defining the values of a data structure for which the steps should be generated. The steps associated with each operation are recorded, which allows the user to step through the visualization if needed. Letting the user engage with the visualization increases their understanding of the visualized algorithm or data structure [4].

- *Exercises* allow users to test their knowledge about the visualized data structure or algorithm. Exercises consist of a description containing instructions and data structures that the user needs to modify in the correct way. The actions of the user are continuously compared to the model solution. The user can redo incorrect actions; however, points are not assigned for such actions.

4.0.3 Language selection

The application supports multiple languages, currently Slovak and English. The module i18n[8] is used for translating texts and automatic route generation for the different languages. Translations are loaded from *JSON* files, in which text in a given language is assigned to the translation key. Slovak language is configured as the default and fallback language of the application. The user can change language at any

[8]Module i18n: `https://i18n.nuxtjs.org/`

time using the language select component in the title of the application (as seen on the top right of Figure 1).

4.1 Configuration files

As mentioned previously, all information about the visualizations is included in the configuration files. There are two files, one for data structures and one for algorithms. These files contain *JSON* objects with all the information needed to display a visualization. The information is stored in the form of an associative array. The visualizations in the application are loaded based on the path defined under the *jsFile* key in this array. Information about each visualization includes:

- *title:* translation key for the main title of the visualization.
- *menuEntry:* translation key for the entry in the side menu.
- *jsFile:* path to the JavaScript file containing the definition of the visualization.
- *options:* a list of visualization types supported by the visualization (allowed values: static, interactive, exercise).
- *description:* object containing translations of the main description of the visualization that is displayed for all types. Translations for both languages are saved in the attributes *"sk"* and *"en"* of the object. These texts are not included in the files containing other texts because of their potential size[9].

5 Implementing visualizations

The visualizations in the application are created using the JSAV library, and each visualization is represented as a class extending a common abstract class in their own JavaScript files. This separation makes the

[9] Translations are loaded for both languages at all times, and since descriptions can be extensive, it is unnecessary for them to be loaded for all visualizations and both languages.

application easily extensible, since the structure of the application does not change when a new visualization is added. The file containing the visualization is specified by the attribute *jsFile* in the configuration. The class which defines the visualization is instantiated after the file containing it is loaded[10].

The different types of visualizations are defined in the same file by overriding one or multiple of the following methods: *initStatic*, *initInteractive*, *initExercise*. These methods are then called from the NuxtJS template when the user changes the visualization type. The data structures used in the visualization are defined and initialized in these methods. For the predefined visualizations, each step is also recorded. If interactive visualization is supported, the *setupCustomControls* method also needs to be overridden where the appropriate controls are defined for the visualization as needed. When adding controls to an interactive visualization, a callback function needs to be defined in which the structures in the JSAV visualization can be modified or new steps can also be created. Some controls require the user to input custom values. These values are encoded properly before being passed to the callback function to prevent Cross-Site Scripting (XSS) attacks [14]. To create an exercise, an initialization function and a model solution function must be created and passed to the JSAV exercise instance. More on exercise creation can be found in the documentation of the Exercise API[11].

6 Included visualizations

One of the requirements for the system was that it should include visualizations for the basic data structures and algorithms. The application currently provides visualizations for basic data structures such as the *linked list*, *stack*, and *queue*. Visualizations are also provided for *inorder*, *preorder*, and *postorder* tree traversal algorithms; *bubble sort*, *depth first search*, and *breadth first search* graph algorithms.

[10] JS files are loaded after information about the visualization is loaded form the configuration file using path parameters.

[11] Exercise API: `http://jsav.io/exercises/exercise/`

6.1 Linked list visualization

For this visualization both predefined and interactive types are supported. The predefined visualization shows how the *insert*, *remove*, *append* and *prepend* operations work on a linked list [15]. The *append* and *prepend* operations are not standard operations on a linked list, but they are suitable for visualizing how pointers behave when adding new elements to the beginning or end of the list. The interactive visualization is initialized with a linked list containing five elements. The user can interact with this list by inserting a new element to any index, removing the element on any index, appending or prepending a new element.

6.2 Stack and queue visualizations

The predefined visualization of the stack data structure uses an indexed array, a stack and a pointer structure from the JSAV library [16]. The two main operations *push* and *pop* are explained in the predefined visualization [17] [18]. The interactive visualization uses the same structures and the user can push a new value to the stack, pop the value on the top of the stack or retrieve the value on the top of the stack with the *top* operation.

The visualization of the queue data structure is similar to the visualization of the stack data structure. It also uses an indexed array, a stack, and two pointer structures from the JSAV library [16]. One pointer is for the top of the stack, the other is for the tail. The *enqueue* and *dequeue* operations are explained in detail in the predefined visualization [17] [18]. The interactive visualization lets the user enqueue custom values in the queue, dequeue the value from the front of the queue or retrieve the value on the front of the queue with the additional *front* operation.

6.3 Tree traversal algorithms

There are visualizations in the application for the *preorder*, *postorder* and *inorder* tree traversal algorithms [18]. Each visualization provides

a predefined type and an exercise. The algorithms are explained step by step in the predefined visualizations. The trees on which the algorithms are illustrated share the same structure, so the relation between the three algorithms can be understood more easily. Since these algorithms are recursive, a visualization of the call stack is also provided [19]. A screenshot of the Preorder tree traversal visualization can be seen in Figure 2.

In the interactive exercises, the user's task is to highlight the nodes of the displayed tree in the correct order according to the chosen tree traversal algorithm. The trees in the exercises are generated pseudo randomly with a maximum height of five.

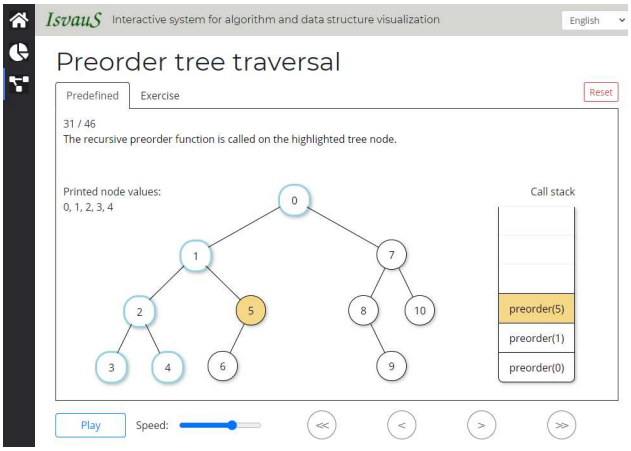

Figure 2. Preorder tree traversal visualization

6.4 Graph algorithms

The *depth first search* (DFS) and *breadth first search* (BFS) algorithms also consist of a predefined part and an exercise. The recursive DFS algorithm visualization makes use of the call stack visualization to help better explain the algorithm. In the BFS algorithm, a queue is used to help keep track of the order of the nodes, which is also included in the visualization. Similarly to the tree traversal algorithms, the

predefined visualizations of both graph algorithms also use a graph with the same structure to help understand the difference between the two algorithms [20].

While solving the exercise, the user must highlight the nodes of the displayed graph in the correct order according to the selected algorithm. The graphs generated in the exercises have eight nodes and the edges between the nodes are directed and generated randomly, while ensuring that there are no isolated nodes in the graph. However, the graph can contain cycles.

6.5 Bubble sort algorithm

For the bubble sort algorithm all three types of visualizations are supported. The predefined visualization shows the algorithm in detail on a randomly generated array of seven elements [21], as seen in Figure 3. The steps of the visualization are generated dynamically for the randomly generated array. The interactive visualization allows the user to enter custom values for the array separated by a space character. The steps of the visualization are then generated for the array defined by the user. In the exercise provided for this algorithm, the user is required to highlight the pairs of elements that are to be swapped according to the bubble sort algorithm in the correct order.

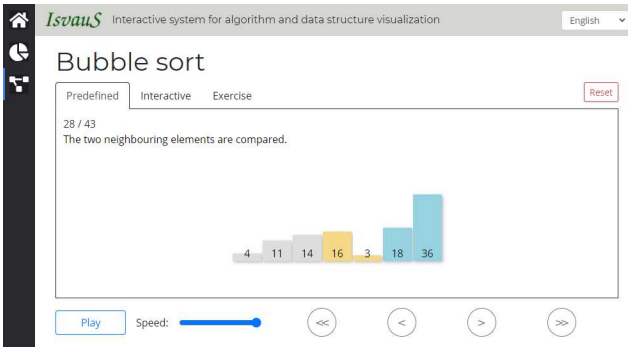

Figure 3. Bubble sort algorithm visualization

7 Application evaluation

In comparison with the other applications analyzed in this work, the designed system covers most of the functionalities which are provided by the other applications. These functionalities include stepping the visualization forward and backward, pausing and playing the visualization, changing the speed at which the visualization is played, and interacting with the data structures being visualized. The new application also provides better availability than the systems previously used in teaching the subject Data Structures and Algorithms. The shortcoming of the newly developed application is the number of provided visualizations, which is less than the number of visualizations included in the previously used systems. Although the new application provides less visualizations, new visualizations can be implemented and added to the application quickly, due to the way the application is implemented.

Evaluation of the application was conducted also by users via a survey. This survey was filled out by 23 students aged between 18 and 25, most of whom took part in the subject Data Structures and Algorithms. The respondents gave the application an overall rating of 4.6 out of 5. The participants were evaluating the application based on the initial requirements. Visualizations of the linked list, stack, and queue data structures and the three tree traversal algorithms were included in the application at the time when the user tests were conducted. The survey was focused on the overall design and the three types of visualizations included in the application. All three types of visualizations were evaluated as easy to understand and helpful in the learning process. The main part of the survey consisted of ten statements, where the respondents needed to mark the extent they agreed with the statement (a score of 1 means they strongly disagreed, a score of 5 means they strongly agreed). The average results of this part of the survey are displayed in Table 1 alongside with the statements.

After filling out the main part of the survey the users were asked to give feedback on which of the provided visualizations were the most helpful for them. The results of this question are displayed in Figure 4. The most helpful were the tree traversal algorithm visualizations

according to the respondents. In the last part of the survey, the respondents had an opportunity to give feedback on the application and suggest new visualizations or functionalities that would be beneficial to implement. The respondents requested the addition of new visualizations, such as the heap data structure, AVL tree, graph algorithms, sorting algorithms, and more. Visualizations of the graph algorithms DFS and BSF and also the bubble sort algorithm were added to the application based on these requests. The most requested functionality was the autoplay feature, which was also implemented in the application after the review of the survey, and responsive design for mobile devices.

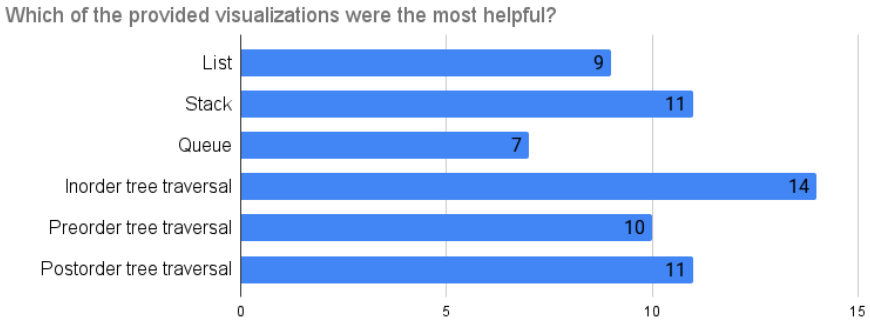

Figure 4. The most useful visualizations according to the survey

8 Conclusion

The result of this work is a web application implemented using the NuxtJS framework and the JSAV library for visualizing data structures and algorithms. The application is designed to be easily extensible with new visualizations without the need to modify its structure. The three supported visualization types allow users to learn from predefined visualizations, interact with them and then test their knowledge via the provided interactive exercises.

The resulting application meets the initial requirements defined in

Table 1. Survey results

Statement	Avg. response
The application is intuitive and easy to use.	4.54
The application is easy to navigate.	4.58
The layout of the application is appropriate for displaying visualizations alongside with pseudocode and descriptions.	4.25
The application helped you to understand how the visualized algorithms and data structures work.	4.58
The predefined visualizations are easy to follow.	4.5
The displayed notes for each step of the predefined visualizations are helpful.	4.38
Having pseudocode alongside the visualizations makes it easier to understand the algorithm.	4.54
Having interactive visualizations in the application is helpful.	4.5
The controls for the interactive visualizations are intuitive and easy to use.	4.42
The prepared exercises are easy to understand and complete.	4.5

this work. By implementing the system in the form of a web application and deploying it to a web server the availability requirement is satisfied. This also ensures that when a new version of the application is released, all users have access to it without the need to download a new executable file or install the new version of the application on

their device.

The oveverall design and layout of the application ensures that the usability requirement is also met as it was confirmed by users during tests.

Descriptions for the included data structures and algorithms are provided in the application. The predefined visualizations contain explanations for each step of the visualized algorithm or data structure, to help better understand them. The visualization of pseudocode in which the currently executed line is always highlighted also helps in the learning process.

Interactive visualizations allow the user to modify the visualized data structures directly, or define custom values for the data structures in the visualization. Each visualization has its own set of controls by which the user can interact with it.

The prepared exercises ensure that the users have a way to test their knowledge. The data structures in the exercises are generated pseudorandomly to ensure the students can't memorize the solutions of the exercises.

One of the main features of the resulting application is that it's easily extensible. Satisfying this requirement affected the design of the application greatly. Adding a new visualization to the application can be done in three steps: extending the configuration file with information about the new visualization; extending the common abstract class and implementing some or all types of visualizations using the JSAV library; finally, providing translations for the newly added translation keys. Everything else (like menu entry and link to the new visualization) will be generated automatically. An example of adding a new visualization to the application is provided in the System manual included in the related work [5].

The application provides visualizations of the basic data structures such as the linked list, stack, queue. These visualizations are made up of predefined and interactive parts. Visualizations for tree traversal algorithms, graph algorithms, and a sorting algorithm are also provided. Exercises are also provided for all the mentioned algorithms.

As of now, the application includes visualizations for only the basic

data structures and algorithms. In the future, we would like to extend the application with new visualizations selected from the topics covered by the Data Structures and Algorithms subject. Implementation of the visualizations suggested by the users should also be considered. New features requested by the users can also be considered for implementation in the future. From the teachers point of view, implementation of group testing of the students via interactive exercises, with an overview of the student's scores might also be beneficial.

References

[1] V. Karavirta and C. A. Shaffer, "Creating engaging online learning material with the jsav javascript algorithm visualization library," *IEEE Transactions on Learning Technologies*, vol. 9, no. 2, pp. 171–183, 2016.

[2] G. Rößling, "A first set of design patterns for algorithm animation," *Electronic Notes in Theoretical Computer Science*, vol. 224, pp. 67–76, 2009, proceedings of the Fifth Program Visualization Workshop (PVW 2008). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066108005136

[3] D. Dicheva and A. Hodge, "Active learning through game play in a data structures course," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 834–839. [Online]. Available: https://doi.org/10.1145/3159450.3159605

[4] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," *SIGCSE Bull.*, vol. 35, no. 2, p. 131–152, Jun. 2002. [Online]. Available: https://doi.org/10.1145/782941.782998

[5] P. Perháč, "Interactive system for algorithm and data structure visualization (in Slovak)," Master's thesis, Department of Com-

puters and Informatics Faculty of Electrical Engineering and Informatics Technical University of Košice, 4 2021.

[6] G. Rößling, M. Mihaylov, and J. Saltmarsh, "Animalsense: Combining automated exercise evaluations with algorithm animations," in *ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science*, 01 2011, pp. 298–302.

[7] S. Šimoňák, "Algorithm visualizations as a way of increasing the quality in computer science education," in *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2016, pp. 153–157.

[8] S. Šimoňák, "Using algorithm visualizations in computer science education," *Open Computer Science*, vol. 4, 10 2014.

[9] S. Šimoňák, "Increasing the engagement level in algorithms and data structures course by driving algorithm visualizations," *Informatica*, vol. 44, 09 2020.

[10] M. Benej and S. Šimoňák, "Algomaster platform extension for improved usability," *Journal of Electrical and Electronics Engineering*, vol. 10, no. 1, pp. 27–30, 2017.

[11] S. Šimoňák and M. Benej, "Visualizing algorithms and data structures using the algomaster platform," *Journal of Information, Control and Management Systems*, vol. 12, no. 2, pp. 189–201, 2014.

[12] D. Galles, "Data structure visualizations," Library homepage, 2011. [Online]. Available: https://www.cs.usfca.edu/~galles/visualization/about.html

[13] S. Khuri, "A user-centered approach for designing algorithm visualizations," *Informatik/Informatique, Special Issue on Visualization of Software*, 2001.

[14] V. Papaspirou, L. Maglaras, and M. A. Ferrag, "A tutorial on cross site scripting attack - defense," 10.20944/preprints202012.0063.v1, 12 2020.

[15] M. Azmoodeh, *Abstract Data Types and Algorithms*, ser. Macmillan Computer Science Series. Macmillan Education UK, 1990.

[16] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures: The*

Basic Toolbox. Springer, 2008.

[17] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data structures and algorithms*. Addison-Wesley, 1985.

[18] D. P. Mehta and e. S. Sahni, *Handbook of data structures and applications*, 1st ed., ser. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC, 2004.

[19] A. Cisneros, "Visualizing recursion," 6 2020. [Online]. Available: https://medium.com/swlh/visualizing-recursion-6a81d50d6c41

[20] K. Mocinecova and W. Steingartner, "Software support for visualizing of the graph algorithms in a novel approach in educating of young it experts," *IPSI BGD TRANSACTIONS ON INTERNET RESEARCH*, vol. 16, pp. 14–23, 7 2020.

[21] P. Kavdikar, "Comparative study of sorting algorithms," 04 2021. [Online]. Available: https://www.researchgate.net/publication/350959496_Comparative_study_of_sorting_algorithms

Patrik Perháč, Slavomír Šimoňák,

Received September 17, 2021
Accepted October 15, 2021

Patrik Perháč
Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic
E–mail: perhac.patrik97@gmail.com

Slavomír Šimoňák
Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic
E–mail: slavomir.simonak@tuke.sk