Journal of Educational Studies in
Mathematics and Computer Science

Interactive Textbook for Algorithms and Data Structures

Vesna Marinković [1, a, *], Filip Marić [1, b]
[1]University of Belgrade, Faculty of Mathematics, Belgrade, Serbia
[a]vesna.marinkovic@matf.bg.ac.rs, [b]filip.maric@matf.bg.ac.rs

Received: 28 September 2024; **Accepted:** 20 November 2024

Abstract

In this paper, an electronic and interactive version of the textbook for the course "Construction and Analysis of Algorithms", which is held at the Faculty of Mathematics, University of Belgrade, is presented. The material covered in the textbook includes advanced data structures, graph algorithms, algebraic algorithms, string processing algorithms, and geometric algorithms. What makes this textbook different from traditional textbooks is its interactive component, which is implemented in JavaScript language using modern web technologies. This feature enables students to understand better how described algorithms and data structures work. Namely, besides being able to follow the execution of the algorithm step-by-step, the textbook often asks readers to independently carry out certain procedures, controlling the correctness of the results. In this way, students can gain information on how well they understand the algorithm. This is consistent with a constructivist approach to teaching in which the student does not acquire knowledge only passively but must have an active role during the entire teaching process.

Since algorithms and data structures play an essential role in computer science studies at all faculties, we believe this textbook can be useful for computer science students of different study programs studying in the Serbian language. The textbook, together with its interactive web version, is freely available online. Libraries developed for the implementation of this textbook are also freely available and can be used to develop similar textbooks for other fields of computer science or mathematics.

Keywords: interactive digital publications, algorithms, data structures.

MSC2020: 97M99

1. Introduction

Today's computer science studies cannot be imagined without at least one course focused on standard algorithms and data structures. Algorithms are used to solve a wide range of practical problems, from ranking students based on the number of points received at the exam to finding the shortest road between two cities. Data structures affect the way data is organized in the memory of computers, as well as the efficiency of accessing and manipulating data. Some common examples of data structures are arrays, linked lists, stacks, queues, maps, and sets.

In programming, some common problems are often encountered, like sorting an array of elements, traversing a graph, or finding the convex hull of a set of points. Typically, there exist different ways in which these problems can be solved, but not all of them are equally efficient. To

* Corresponding author
E-mail address: vesna.marinkovic@matf.bg.ac.rs

be able to write high-quality software, one must be familiar with the most efficient algorithms to solve such problems.

Learning about data structures teaches students how to efficiently store and manipulate data, thereby making the program code more efficient: by choosing the proper data structure, both the time and space complexity of the algorithm can be improved (Dwaraka Srihith, 2023). A good understanding of data structures provides a deeper understanding of the nature of the problem that is being solved. Choosing the proper data structure also makes the code easier to understand and maintain.

For all the reasons listed, algorithms and data structures play a crucial role in computer science and, therefore, traditionally represent the cornerstone of computer science education. In today's competitive labor market, good knowledge of algorithms and data structures represents a competitive advantage. Many companies are looking for candidates who have a strong background in this field, as it is one of the critical skills in the technology industry nowadays.

However, mastering data structures and algorithms is not an easy task, especially for novice students (Zingaro, 2018, p. 169). Many different textbooks and online resources exist for learning algorithms. However, they are often focused on the main ideas and implementation of algorithms and not sufficiently on the understandability of the ideas behind them. To fully understand how an algorithm works, students often need to simulate the step-by-step execution of the algorithm on different instances of input on a piece of paper. This process is tedious and error-prone, and the user does not get any feedback about its correctness.

Nowadays, teachers are constantly searching for ways to improve the learning process and hold students' attention (Knight, 2015). One way to achieve this is to use the computer to aid visualization and better understand the problems being solved. Computer technology enables animations to be shown, which allows algorithms to be actively explored and makes learning more dynamic (Vegh, 2017). Animations are performed step-by-step, and users can navigate forward and backward through the steps. This helps the student remember which steps were carried out by the algorithm but also to understand the meaning of the algorithm and its performance (Kann, 1997). The set of objects to which an algorithm is applied should be large enough to illustrate all the key properties of the algorithm but also not too large to "not see the tree for the forest".

A modern web interface supporting mouse and keyboard interactions enables students to simulate algorithm execution step by step on their own in a very natural and comfortable manner. By doing this, students can immediately see if they understand how the algorithm (or a data structure) functions and can immediately correct their mistakes. The system can also provide hints and next-step guidance, helping students to correct their mistakes as soon as possible. Without such a system, students must explicitly write intermediate states (for example, draw the binary tree after each element insertion) on a paper that a teacher later checks. Intermediate states are necessary because the process is not clear only from the final result (for example, the final state of a tree does not reveal its transformations through time). This is quite unnatural and much more tedious than transforming the tree with the mouse in an interactive setting. Students can also demonstrate the algorithm execution in front of a blackboard or on a piece of paper (for example, they can manipulate the drawing of a binary tree), but this requires that the teacher watches the whole process (either live or recorded). This is very expensive and cannot scale well to a large group of students.

In this paper, we give an overview of the textbook "Construction and Analysis of Algorithms" for the obligatory course of the same name for the 2nd year computer science students, held at the Faculty of Mathematics, University of Belgrade. The textbook in PDF format and HTML format with its interactive component is freely available from the course web page: http://algoritmi.matf.bg.ac.rs/kiaa/index.html, and we hope that it can be a valuable resource for both students and teachers. The motivation beside this textbook is to present covered topics in a comprehensible and illustrative way while keeping a high level of mathematical rigor. The textbook contains many examples, illustrations, and explanations with the intention of making even complex

concepts accessible to students. Interactive animations that illustrate the step-by-step execution of the algorithm, along with examples where students can independently conduct the algorithm and check its validity, are what give this textbook special importance (Brown, 1988). The applets provided are aligned in terminology and variable names with the surrounding text of the textbook.

Since we advocate that all kinds of knowledge are more easily acquired when they are visualized, we follow this idea both in this paper and provide many illustrations from the textbook.

2. Related Work

Algorithm visualization is a well-studied topic. There are many algorithm visualization websites that provide visualizations of algorithms from different areas and of various difficulties (e.g., https://algorithm-visualizer.net/, https://visualgo.net, https://algorithm-visualizer.org/, https://csvistool.com/). Computer technology enables even more: to check students' understanding of how the algorithm works and provide immediate feedback to students' answers (Lust, 2020). Some systems available on the web, in addition to animations, offer interactive visualizations and interactive exercises (Perhác, 2022). Studies show that students prefer using interactive tools to traditional ones and that they help them achieve better results (Färnqvist, 2016). All these interactive components can be gathered into an online learning platform, such as https://opendsa-server.cs.vt.edu/. However, we are not aware of any online platform or website focused on the visualization of algorithms in the Serbian language.

There exist several meta-studies that compare and evaluate various visualization systems. Shaffer et al. made a catalogue of over 350 algorithm visualizations (Shaffer, 2010). They came to the conclusion that the focus of these systems is mostly on easier algorithmic topics and that most visualizations are of low quality. Hundhausen et al. conducted a systematic meta-study of 24 experimental studies and concluded that how students use visualization is more important than what the visualization shows them (Hundhausen, 2002). Grissom et al. compared levels of student engagement and showed that learning increases as the level of student engagement in the learning process increases (Grissom, 2003).

Electronic textbooks have gained popularity in recent years, especially after the COVID-19 pandemic. Official frameworks for the design of electronic textbooks are developed. Kreuh et al. recognized four different levels of electronic textbooks: plain PDF format, electronic textbook enriched with multimedia (hyperlinks, video, audio, animations), electronic textbook with active user involvement (quizzes, simulations), and virtual collaborative learning environment (Kreuh, 2015). Each subsequent level brings more benefits to students. When evaluating printed and electronic textbooks, different quality standards exist. Evaluation of digital textbooks may focus on the presence of multimedia, interactivity, branched structure, and navigation features (Pešikan, 2023).

3. Course Curriculum

In the Informatics Studies curriculum at the Faculty of Mathematics, there are two obligatory algorithmic courses. The first of them, "Algorithms and Data Structures", introduces students to basic data structures (like arrays, trees, maps, and sets), as well as to different strategies to construct algorithms (such as backtracking, divide and conquer and greedy algorithms). It also teaches them how to prove the correctness of the algorithm and to perform an analysis of time and space complexity. The second algorithmic course, "Construction and Analysis of Algorithms" covers advanced topics, including advanced data structures (like prefix and segment trees and union-find structures), graph algorithms (including traversal, topological sort, shortest paths, Euler/Hamiltonian paths, and spanning trees), algebraic algorithms (spanning from simpler topics dealing with primes, divisibility, modular arithmetic, etc. to more demanding ones like Fast Fourier Transformation), string processing algorithms (including hashing, pattern matching, and

palindromes), as well as geometric algorithms (such as the construction of simple polygons and convex hulls). This course covers different algorithmic problems that one can encounter in practice, classified by the domain they belong to. For instance, convex hulls have a wide range of different applications like planning robot movements, modeling smooth curves, and tracking the spread of epidemics; shortest path algorithms play an important role in navigation and data routing in networks; algebraic algorithms are important for cryptography and quantum computing.

Since this study program is conducted at the Faculty of Mathematics, students enrolled in the course "Construction and Analysis of Algorithms" are already familiar with some of the topics taught: they study basic geometric algorithms on the course "Geometry 1", learn math derivations needed for developing algebraic algorithms on the course "Algebra 1", as well as how graph algorithms work on the course "Discrete Structures 2". However, in algorithm courses, we look at the problems from a different perspective: we discuss which data structures are suitable for the problem being solved, analyze the correctness of the developed algorithm and its time complexity, implement it, etc.

Based on many years of experience in teaching algorithm courses, we got the impression that learning the course "Construction and Analysis of Algorithms" can be challenging for many students. The course syllabus is extensive, covering advanced algorithmic concepts, thus requiring continuous and dedicated student work. Therefore, students would benefit from any help that would enable them to understand the ideas behind the algorithms learned more easily and to have an opportunity to check their understanding independently.

4. Results Visual Component of the Textbook

4.1. Illustrations

Algorithms in classical textbooks are usually described using pseudocode or implementation in program code, followed by an explanation of how an algorithm works. We stand behind the maxim that "a picture is worth a thousand words" and advocate that visual aids facilitate understanding (Fleischer, 2002). Therefore, both mathematical and computer science textbooks should be rich in examples and illustrations. Visualization makes understanding abstract ideas tangible: students can grasp core principles much easier by using illustration than by solely having abstract information. Visual information holds the student's attention, and some information can be "picked up" along the way.

Having this in mind, in the textbook "Construction and Analysis of Algorithms", solutions are, whenever possible, supported by using different types of sketches, illustrations, and diagrams. It can help in learning the new terms that are being introduced (for instance, the definition of the different types of edges in a graph, illustrated in Fig. 1), understanding the idea behind the algorithm (for instance, the way how in Dijkstra's shortest path algorithm, the set of k nearest nodes to the given node is expanded, illustrated in Fig. 2), or how the algorithm works (like an illustration of partially done execution of z-algorithm and analysis of relevant values for the next execution step, illustrated in Fig. 3).

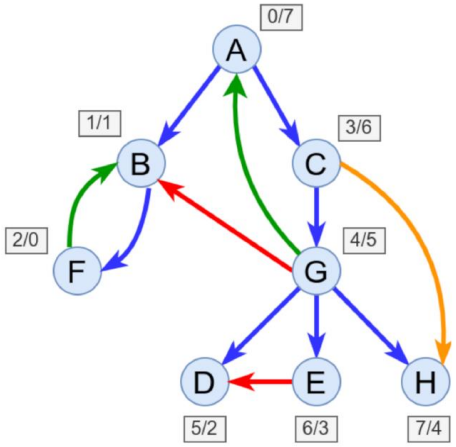


Fig. 1. Classification of edges of a directed graph, together with preorder and postorder numeration of its vertices.

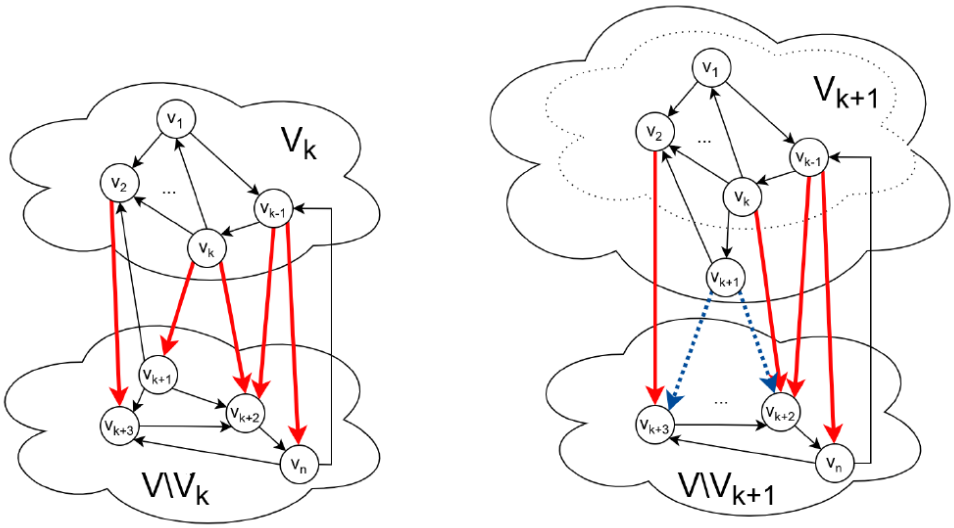


Fig. 2. Identifying the next closest vertex to the vertex v_1 in Dijkstra's algorithm – expanding the set of closest vertices to v_1 by v_k and illustration, which vertices are further considered.

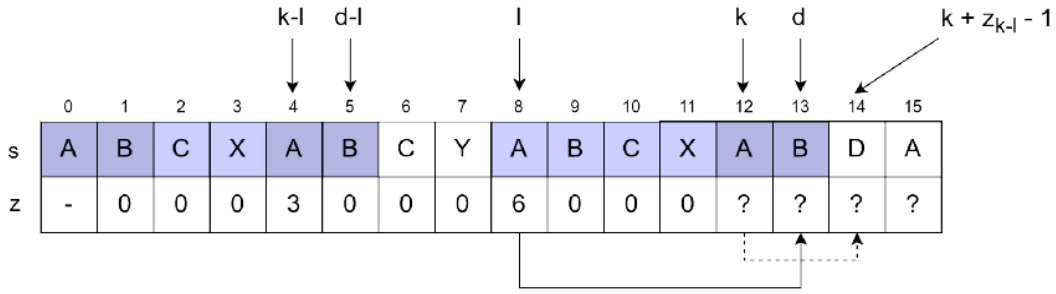


Fig. 3. Illustration of z-box considered while calculating the values of z-array of string s.

Sometimes, illustrations can help in comprehending the algorithm's correctness and its complexity. Visualization can make it easier to see invariants, as well as infer the order of magnitude of the number of steps that are executed in the algorithm. In Fig. 4 the relationship between lowlink values of parent and child vertices is illustrated, which is used to prove the correctness of Tarjan's algorithm for identifying bridges in the graph. The illustration of the worst-case time complexity of the Knuth-Moris-Prath (KMP) algorithm for pattern matching is illustrated in Fig. 5.

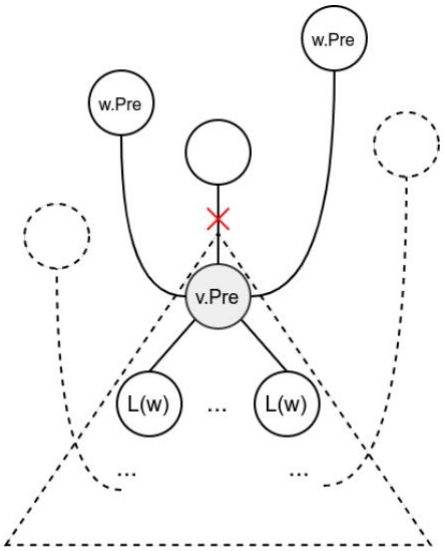


Fig. 4. Illustration of the relationship between the lowlink value of the vertex v with lowlink values of its children and preorder values of its ancestors to which a back edge lead.

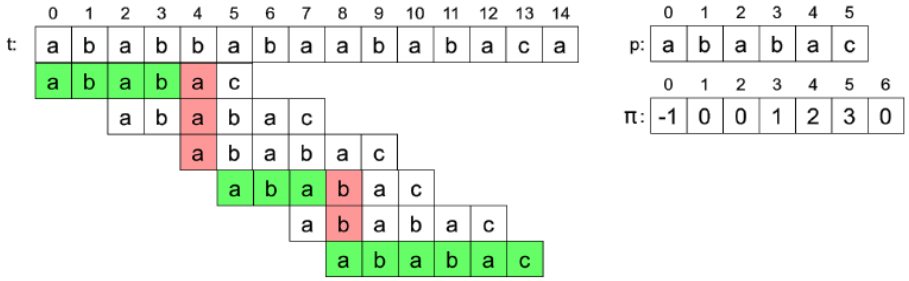


Fig. 5. Illustration of time complexity of KMP algorithm: successful and unsuccessful comparisons of characters form "stairs" whose number is mostly two times greater than the length of string t.

4.2. Step-by-step Execution

During algorithm execution, the values of relevant variables change. In class, this can be simulated by the teacher erasing one value on the blackboard and replacing it with another. However, this makes taking notes by students almost impossible and causes them to often lose track of the main ideas of the algorithm. Therefore, it would be helpful for the textbook to contain a step-by-step illustration of the execution of the algorithm.

Stepwise execution can be illustrated by showing the values of intermediate values of relevant variables (see illustration of breadth-first search (BFS) algorithm given in Fig. 6), or algorithm state after each step of the algorithm (see illustration of execution of Prim's algorithm for calculating minimum cost spanning tree given in Fig. 7 and gift-wrapping algorithm shown in Fig. 8).

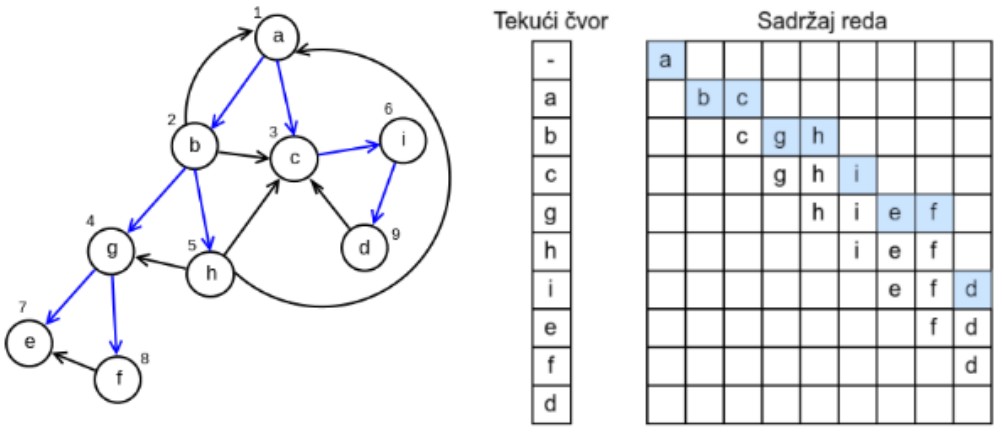
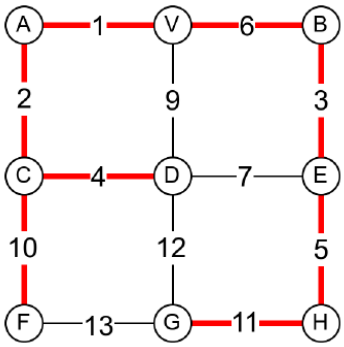


Fig. 6. Illustration of step-by-step execution of BFS algorithm: each row of the table lists the vertex that is currently traversed and shows the current content of the queue.



	V	A	B	C	D	E	F	G	H
V	-	$1(V)$	$6(V)$	∞	$9(V)$	∞	∞	∞	∞
A	-	-	$6(V)$	$2(A)$	$9(V)$	∞	∞	∞	∞
C	-	-	-	-	$4(C)$	∞	$10(C)$	∞	∞
D	-	-	$6(V)$	-	-	$7(D)$	$10(C)$	$12(D)$	∞
B	-	-	-	-	-	$3(B)$	$10(C)$	$12(D)$	∞
E	-	-	-	-	-	-	$10(C)$	$12(D)$	$5(E)$
H	-	-	-	-	-	-	$10(C)$	$11(H)$	-
F	-	-	-	-	-	-	-	$11(H)$	-
G	-	-	-	-	-	-	-	-	-

Fig. 7. Illustration of step-by-step execution of Prim's algorithm for calculating minimum cost spanning tree of the graph illustrated above.

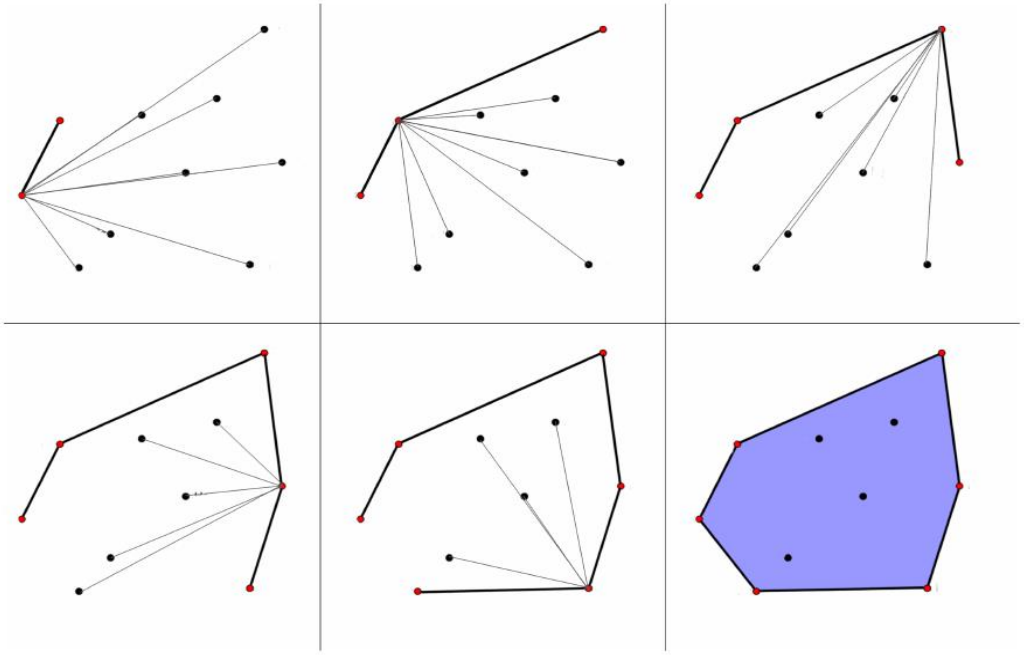


Fig. 8. Illustration of step-by-step execution of the gift-wrapping algorithm for calculating the convex hull of a set of points.

4.3. Animations

Animations can be static, meaning they are provided only for a fixed set of examples, or dynamic, meaning the user can supply his/her own example. The first type of animation can be implemented as a series of static images (such as the construction of a simple polygon over a given set of points illustrated in Fig. 9), while the second type of animation requires the implementation of an algorithm and its visualization (like the execution of the first phase of the KMP algorithm for string provided by the user, illustrated in Fig. 10).

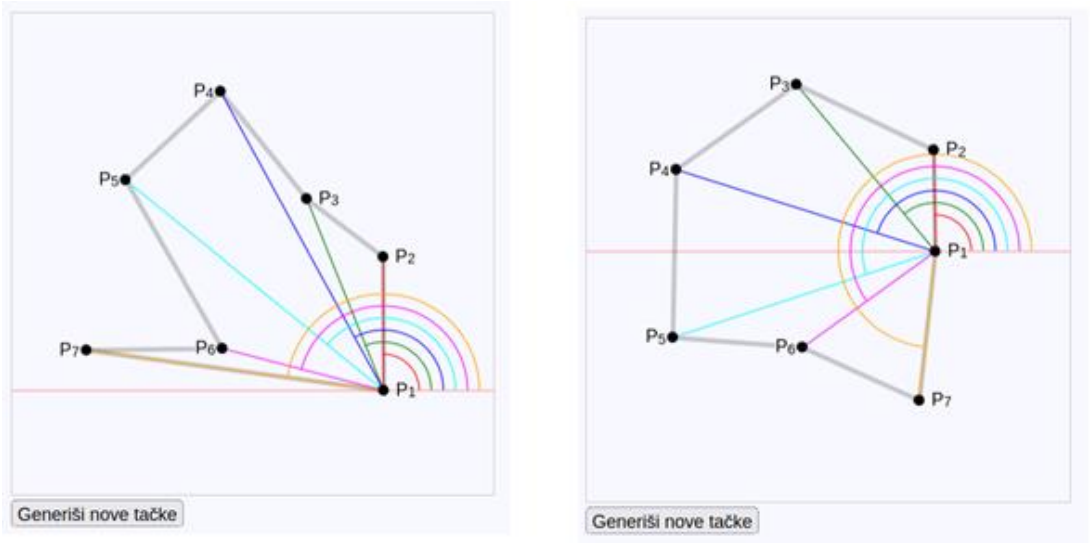


Fig. 9 Illustration of a simple polygon construction over two different input sets of points.

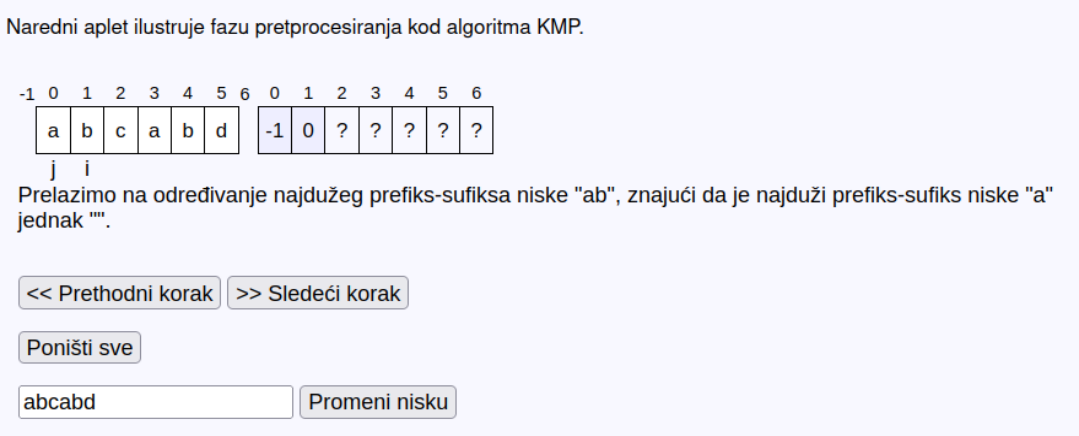


Fig. 10 Execution of first phase of KMP algorithm for user-provided string

Notice that for some problems, it is satisfiable to pick random input and illustrate execution on it; however, sometimes it is more illustrative to carefully choose the representative input data.

5. Interactive Component of the Textbook

We are aware that modern times bring reduced attention among students and that it is necessary to keep them an active participant in the learning process. Namely, a student who just watches how animation executes is still quite passive. There is no external check that he understands how the algorithm works. Students should be tempted to check their understanding as often as possible.

One well-known example that illustrates our approach is sorting algorithms. Traditionally, there are many videos and applets that show their step-by-step execution. Testing understanding of those algorithms often requires students to write the array content after each sorting step (swapping of two elements). The same can be achieved in a much more comfortable way using modern web technologies. By using mouse actions such as drag and drop, students can simulate algorithm execution by swapping array elements that are swapped by the algorithm. The system can provide immediate feedback by reacting to each step and complaining after a wrong step is made. In another mode, the system can record all the steps and give feedback only after the exercise is submitted (which is a convenient feature for automated grading).

Our textbook has its interactive counterpart, freely available from the address https://algoritmi.matf.bg.ac.rs/kiaa/index.html, where exercises for students are incorporated into illustrations and animations. Students are asked to enter their solution into the appropriate place, and the message box stating whether the solution is correct appears (see the assignment to students to fill in missing values in the segment tree, illustrated in Fig. 11).

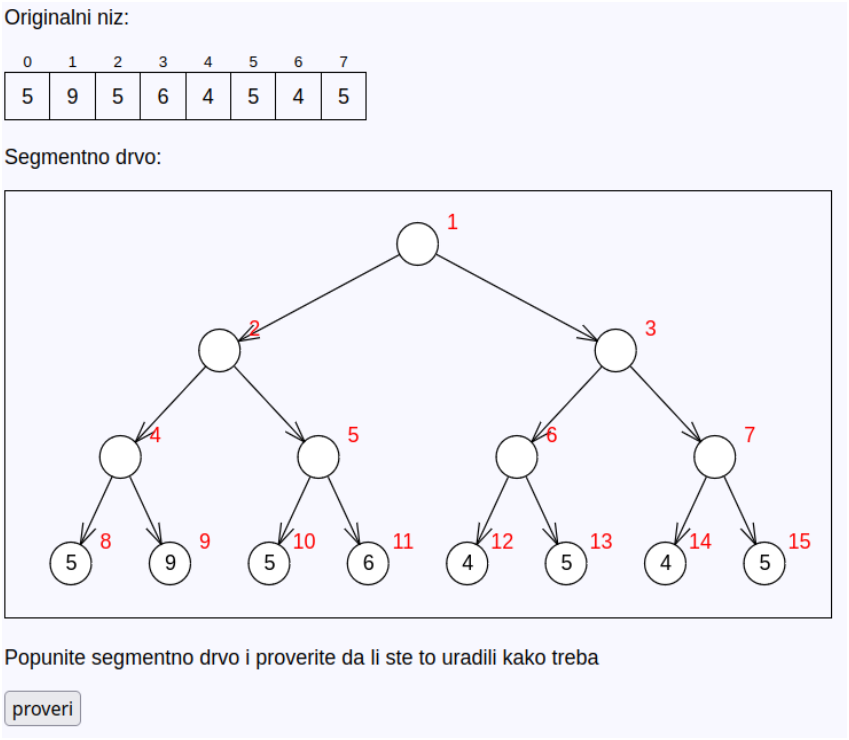


Fig. 11. Illustration of exercise to students to fill in empty cells in segment tree

Another possibility is to ask students to simulate algorithm execution and check if their simulation is correct: it can be achieved by clicking on the objects shown in the animation in proper order or by entering correct values (for instance, to simulate the execution of Dijsktra's algorithm, exemplified in Fig. 12). Once they finish, they will get immediate feedback on whether the simulation was done correctly.

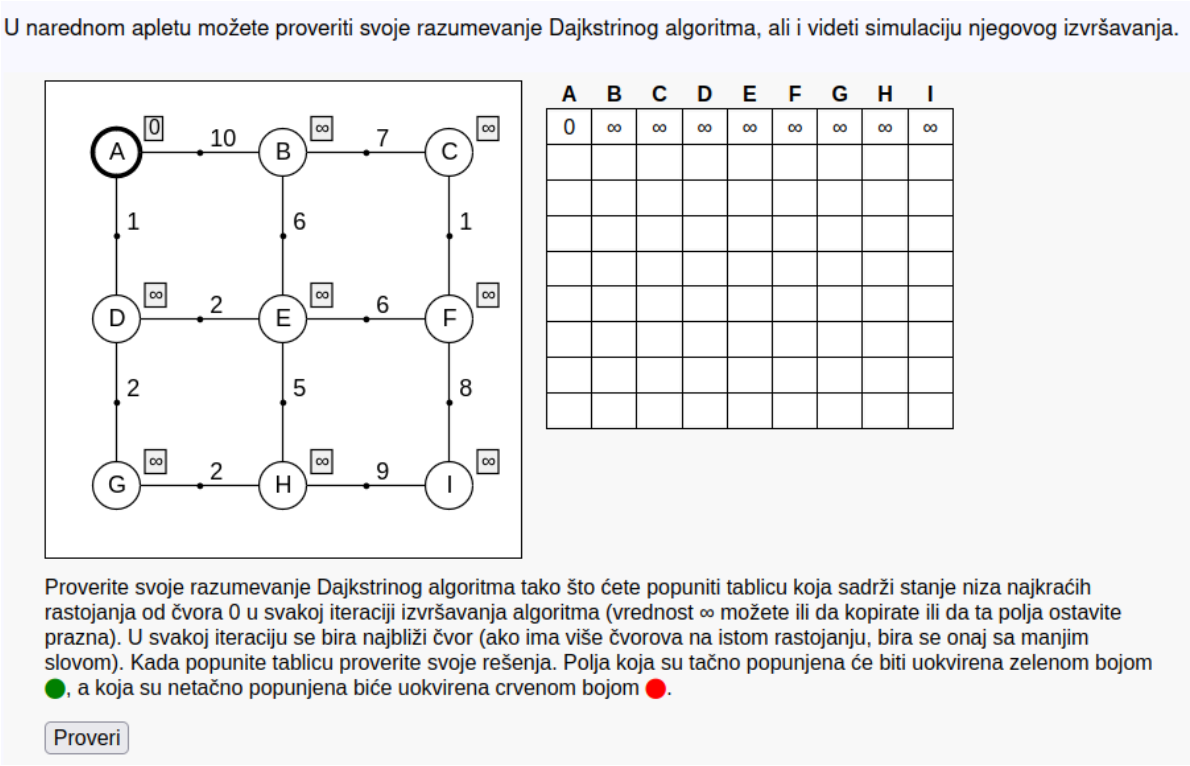


Fig. 12 Illustration of exercise to students to simulate the execution of Dijkstra's algorithm.

Our interactive textbook is implemented in JavaScript using several custom-developed libraries. We have developed a library for showing arrays and matrices, supporting showing pointers, and marking contents of current cells (array or matrix elements). This library is useful for a wide range of algorithms that manipulate arrays and matrices (e.g., sorting algorithms, two-pointer algorithms, dynamic programming algorithms, etc). For example, it is used in Fig. 10 to show the content of two arrays, along with two pointers i. and j, and in Fig. 12 to show the content of the matrix (that contains successive states of the shortest path array, as it changes during the algorithm execution). We have also developed a library for working with graphs, supporting automated graph drawing and node layout, marking nodes and edges, showing node and edge labels, dragging and dropping nodes, and many other operations. For example, it is used in Fig. 11 to show the content of a segment tree and in Fig. 12 to show the weighted graph and current values of node distances from the starting node. We have developed a library, ArgoDG, for dynamic geometry, supporting drawing and animating geometric objects. We have used this library for other purposes (for example, for visualizing constructions in hyperbolic geometry (Marinković, 2023)), but in the context of the algorithm's textbook, it was very useful for demonstrating geometric algorithms. For example, it is used in Fig. 8 and in Fig. 10 for visualizing convex hull construction algorithms. We have also developed a library for showing intermediate values of variables, arrays, and matrices in a step-by-step execution of a given pseudo-code. All these libraries are open-source and available upon request.

6. Conclusion

In this paper, an electronic and interactive version of the textbook for the course "Construction and Analysis of Algorithms" is presented. When creating the textbook, special care was taken to make the material as illustrative and comprehensible as possible for the students. Therefore, many examples and illustrations showing algorithm execution, but also illustrating terms and concepts introduced, are added. The special value of this textbook is its interactive electronic version, which not only contains a number of animations but also offers students the chance to test their understanding by simulating algorithm execution. Compared to similar available interactive resources for learning algorithms that usually deal with standard elementary algorithmic problems such as sorting, this textbook focuses on advanced algorithmic problems.

We believe that this textbook can be useful not only for students attending this university course but also to anyone interested in learning more about algorithms and data structures. As far as we know, this is the first interactive textbook for learning algorithms in Serbian language. Libraries developed for the implementation of this textbook are freely available, and we plan to use them for the development of another algorithm course. Moreover, these libraries can be used to develop similar textbooks for different fields of computer science or mathematics, fostering in that way collaboration in the field of interactive learning resources.

References

BROWN, M. (1988). Algorithm animation. *The MIT Press*.

SRIHITH, I. D., DONALD, A. D., SRINIVAS, T. A. S., ANJALI, D., & VARAPRASAD, R. (2023). The backbone of computing: an exploration of data structures. *International Journal of Advanced Research in Science, Communication and Technology*, *3*, 155-163. https://doi.org/10.48175/IJARSCT-9105

FÄRNQVIST, T., HEINTZ, F., LAMBRIX, P., MANNILA, L., & WANG, C. (2016). Supporting active learning by introducing an interactive teaching tool in a data structures and algorithms course. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 663-668). https://doi.org/10.1145/2839509.2844653

FLEISCHER, R., & KUČERA, L. (2002). Algorithm animation for teaching. In *Software Visualization* (pp. 113–128).

GRISSOM, S., MCNALLY, M., & NAPS, T. (2003). Algorithm visualization in CS education: Comparing levels of student engagement. *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization* (pp. 87–94). https://doi.org/10.1145/774833.774846

HUNDHAUSEN, C., DOUGLAS, S., & STASKO, J. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing, 13*(3), 259–290. https://doi.org/10.1006/jvlc.2002.0237

KANN, C., LINDEMAN, R., & HELLER, R. (1997). Integrating algorithm animation into a learning environment. *Computers & Education, 28*(4). https://doi.org/10.1016/S0360-1315(97)00015-8

KNIGHT, B., & WANG, S. (2015). Teachers' use of textbooks in the digital age. *Cogent Education, 2*(1). https://doi.org/10.1080/2331186X.2015.1015812

KREUH, N., KAČ, LJ., & MOHORIČ, G. (2015). Framework for the design of electronic textbooks. *The National Education Institute Slovenia*.

LUST, M., TAMMETS, P., LAANPERE, J., & LAANPERE, M. (2020). Interactive Textbooks in School Informatics: A Case Study in Estonia. In *ISSEP (CEURWS Volume)* (pp. 107-119).

MARINKOVIĆ, V., ŠUKILOVIĆ, T., & MARIĆ, F. (2023). Automated triangle constructions in hyperbolic geometry. *Annals of Mathematics and Artificial Intelligence, 91*(6), 821–849. https://doi.org/10.1007/s10472-023-09850-5

PEŠIKAN, A., & LALOVIĆ, Z. (2023). Quality standards for digital textbooks and auxiliary digital educational materials. *Institute for Textbooks and Teaching Aids, Podgorica*.

PERHÁC, P., & ŠIMOŇÁK, S. (2022). Interactive system for algorithm and data structure visualization. https://doi.org/10.56415/csjm.v30.02

SHAFFER, C. A., COOPER, M. L., ALON, A. J. D., AKBAR, M., STEWART, M., PONCE, S., & EDWARDS, S. H. (2010). Algorithm visualization: The state of the field. *ACM Transactions on Computing Education (TOCE), 10*(3), 1-22. https://doi.org/10.1145/1821996.1821997

VEGH, L., & STOFFOVA, V. (2017). Algorithm animations for teaching and learning the main ideas of basic sortings. *Informatics in Education, 16*(1), 121–140. https://doi.org/10.15388/infedu.2017.07

ZINGARO, D., TAYLOR, C., PORTER, L., CLANCY, M., LEE, C., NAM LIAO, S., & WEBB, K. C. (2018). Identifying student difficulties with basic data structures. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (pp. 169-177).