Some methods for improving data structure teaching efficiency

Zarema S. Seidametova[0000−0001−7643−6386]

Crimean Engineering and Pedagogical University,
8 Uchebnyi per., Simferopol, 95015, Ukraine
z.seydametova@gmail.com
http://cepulib.ru/index.php/ru/resursy/personalii/47-s-personalii/169-sejdametova-zarema-sejdalievna

Abstract. Computer simulation and modeling are now widely used in computer science teaching. There are various approaches to make the educational process more efficient. Visualization is one of them, and the flipped classroom approach is another. The goal of this research is to show how to improve the efficiency of teaching data structures (hashing, trees) in the Algorithms and Data Structures course. We conducted two tests with four study groups of second-year bachelor students in two Algorithms and Data Structures subjects – (1) Hashing, (2) Trees (BST, RBT, AVL). In the first experiment, study groups on subjects (1) and (2) were formed using the same type of teaching technique, either with or without visualization tools. In contrast to the first experiment, we employed flipped classroom approaches for one study group and a traditional teaching method for the other.

Keywords: computer science education · algorithms · data structure · visualization · flipped classroom

1 Introduction

Computer modeling and simulation has become a widely used tool for solving real-world problems, for training and research on human-computer interaction with complex work and different environments. A simulation is defined as a working representation of reality [9] and can be an abstracted, simplified, or accelerated model of a process or system. Computer modeling and simulation allow exploration cases even if reality is too expensive, complex, or dangerous [9]. For purpose of appropriately using computer modeling and simulation, we need to prepare students to develop and operate with such applications.

Many concepts of computer science and software engineering are difficult for students to understand [24]. Algorithms and Data structures are one of the most important subjects in Computer Science, which is required by all programmers. Programmers need to know how to handle the data in huge quantities, how the data should be stored in the memory and how the memory should be used efficiently. We search the effective ways for implementing pedagogical tools and

approaches for proper demonstration of important concepts of algorithms and data structures.

Lee and Lee [12], Knutas et al. [11], Ling et al. [13] discuss the flipped classroom methods for programming courses and promoting student engagement using this method for ordinary or large-scale courses.

The topics of team-based learning and other trends in higher education are considered by Vlachopoulos et al. [27], Burbules et al. [4]. Chang et al. [5] examine the difference in student motivation during data structure courses conducted with the visual programming language and Java.

In [2, 3, 10, 17, 23] described the taxonomy of algorithm animation languages, multimedia applications for data structure visualization, and tools for visualizing contents media of data structure courses for mobile learning. Wang [28] proposes a public computer lab management system with four functional parts that provide a convenient and efficient way for the teachers and students to communicate and improves the management level of a public computer lab.

The flipped classroom approaches presented in [1, 7, 8, 16, 26]. These studies show the effectiveness of flipped classrooms in a wide range of subjects like Engineering, Education Technology, Mathematics, and Statistics, etc. The studies also show the effectiveness of flipped classrooms in programming courses. These studies also found that the flipped classroom method facilitated learning, helped students to develop soft and hard skills, improved students' innovative consciousness and attitudes.

There are other pedagogical approaches that can be used in teaching. In [14, 15, 18–22, 25] described different algorithm visualization platforms that can be divided as web, mobile and desktop applications.

To study this problem, we use such *methods* as a literature review, own experience in teaching algorithms and data structures, tasks, tests, statistical analysis, etc.

2 Algorithms and Data Structures learning roadmap

Algorithms and Data structures are really important courses for every concept used in computer science and software engineering. There are many concepts involved in the subjects of Algorithms and Data structures. There will be many questions for educators like how to learn Algorithms and Data structures, as there are many concepts involved without getting confused during attending the course. For teaching the one-semester course "Algorithms and Data structures" we used the third edition of the fundamental classic CLRS book [6].

The roadmap of teaching and learning and Data structures we define as two main parts presented in schemes on figure 1 (part 1 – Algorithms) and figure 2 (part 2 – Data structures). Part 1 includes such topics as:

– Algorithms analysis technics – asymptotic analysis and notations, Big-Oh, complexities of algorithms, worst, average, best cases of algorithm performing, recurrences, master theorem for solving some types of recurrence equations;

– Sorting algorithms with comparisons – bubble sort, selection sort, insertion sort, merge sort, heap sort, quicksort;
– Sorting algorithms without comparisons (in linear-time) – bucket sort, counting sort, radix sort;
– Searcher algorithms – linear search, binary search, jump search.

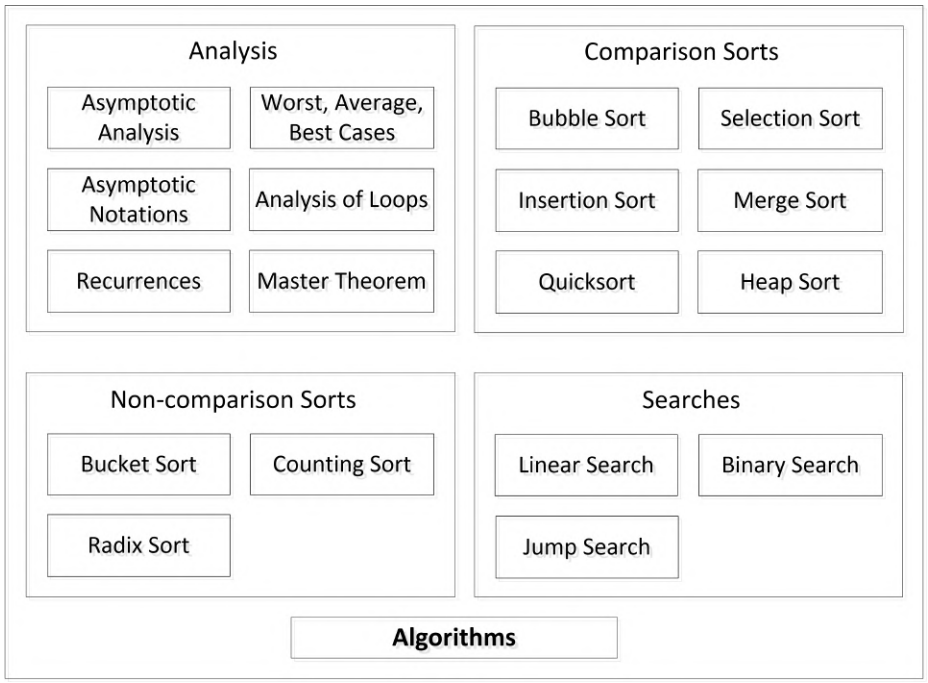

Fig. 1. Algorithms learning roadmap.

Part 2 includes data structures topics:

– Linear data structures – array, stack, queue, linked list;
– Non-linear data structures – hashing (direct hash tables, collisions, open addressing, double hashing, perfect hashing), trees (BST, RBT, AVL), graphs, heaps;
– Augmented data structures.

Algorithms and Data structures are used to design an efficient algorithm, organize data access, and economize the execution time of programs. Data structures are widely used in operating systems, database management systems, statistical soft-ware, compiler design, numerical analysis, and artificial intelligence simulations. Algorithms and Data structures play a significant role in the computer science and soft-ware engineering domain. The course of algorithms and

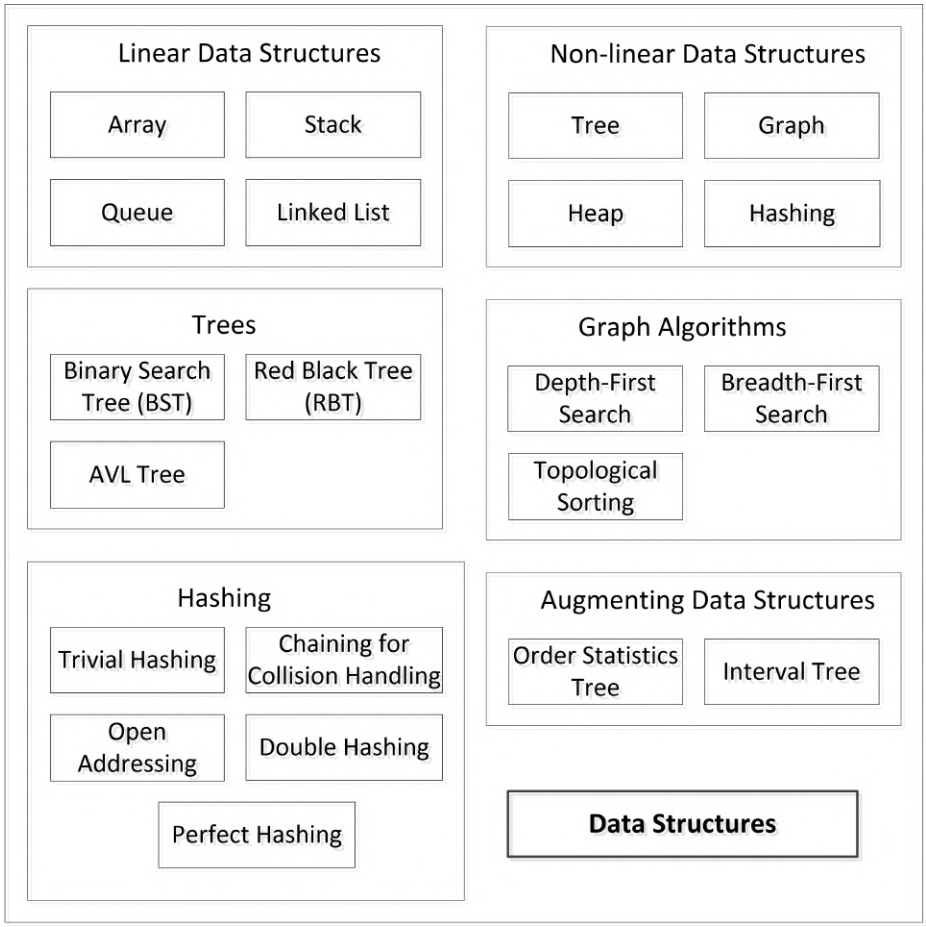

Fig. 2. Data Structures learning roadmap.

data structure introduces the basic data type, such as byte and character string, and presents collective data types, linked lists, stacks, arrays, queues, trees, heaps, and priority queues and complex data structures and, and focuses on programming methodologies, problem-solving strategies, and algorithm development.

The algorithms and data structures courses are typically conducted with lectures with homework, labs, and projects. Students are asked to learn algorithms implementation, data structures representations. However, students often have misconceptions. Without proper knowledge of the many classical algorithms and data structures considered in the course, it is difficult for students to understand when best to use which data structures. Such approach with the in-class lecture was mostly teacher-centered and student-teacher interaction in the class was not enough.

To find more effective ways of teaching algorithms and data structures we considered different approaches – visualization and flipped classroom concept. We decided to compare the effect on students learning results of using both approaches. The results of our research are presented in the last section. We chose Hashing and Trees topics for our research study.

The main problem of hashing, studied in the classical course Algorithms and Data Structures includes the following topics: hash tables (a generalization of an array representation), direct addressing (using an array, the size of which is comparable to the number of actual keys stored in it), resolving possible collisions (avoiding cases of different keys entering the same slot), constructing hash functions (there are a large number of implementation options), open addressing (collision resolution method), universal and ideal hashing (used to maintain $O(1)$ time in the worst case for a statistical set of keys when performing dictionary operations).

Binary Search Trees (BST) support operations: queries – tree traversal, finding the minimum by value, finding the maximum by value, finding the next by value, finding the previous by value, finding a node with a given key value, and operations that update BST – Insertion and Removal of a Node. When studying these operations, it is necessary to draw the attention of students to the peculiarities of performing these operations, as well as the peculiarities of inserting and removing nodes. When deleting nodes, students need to consider all three possible deletion scenarios.

Red and Black Trees (RBT) are challenging enough for students to learn. RBT is a binary search tree that has an additional field `x.color` (red or black) and obeys the red-black properties (RB properties):

- each node is either red or black;
- tree root and leaves (`nil[T]`) are black;
- if the node is red, then both child nodes are black;
- for each node, all paths from it to the leaves, descendants of this node, contain the same number of black nodes.

The color field and RB properties allow the tree to be balanced. Query operations in the RB-tree are performed in the same way as in the BST. Structure-altering operations, INSERTs, and DELETEs can violate RB properties and require balancing. In the case of RB trees, the ROTATE operation is used to restore the RB property. In codes, the rotation operation is implemented by redesignating node pointers.

Another example of a balanced binary search tree that we recommend to study with students is AVL trees, which for balancing are supplemented with a node height parameter (or store the value of the difference in heights of the left and right subtrees, i.e., one of the values -1, 0, +1), and also obey the property of the AVL tree: the value of the left and right subtrees of any node should not differ by more than 1. The operations supported by AVL trees are the same as in the BST (binary search tree), RBT (red-black trees). Searching for the smallest (MIN), largest (MAX), next, and preceding element by value, searching for a

node with a given key – are performed according to the same algorithms as in BST. The operations of inserting and deleting a node are first performed, as in the binary search tree, but then it is necessary to check whether the AVL property is violated. After that, it is necessary to carry outbalancing.

Flipped classroom approach has become popular in education in recent decades. The flipped classroom is a student-centered pedagogical approach in which the typical course lecture and homework are reversed. Successful flipped classroom implementation involves not just having lecture videos available before class.

The algorithms and data structures course lasts 18 weeks over a semester. The class meets twice a week. We chose two topics of data structures – Hashing and Trees – for using flipped classroom concept. Each week, we recorded the lecture in several 15-to-30-minute videos. The lecture videos are made available on Moodle before the in-class lecture. Students were instructed to watch the lecture videos before coming to class each week. In a class, face-to-face meeting, students are given worksheets, which ask students to practice the topics of data structures presented in the lecture videos each week. At the end of class, the worksheets are checked for completeness and returned to the students. Students are encouraged to help answer any questions by a teacher, to discuss, and to work out the tasks on the worksheets together with other students.

The same two topics – Hashing and Trees – we chose for implementing the visualization approach. There are a lot of algorithms and data structures visualization systems available. And it is not easy to find tools that fully meet the needed requirements. As a tool of visualization, we chose the visualization modules suggested by `Mathematica 12.0` that include visualization features of hashing and trees.

Algorithms and data structures visualization can be seen as a valuable supporting pedagogical tool, used as an addition to standard classical ways of education in the field of computer science and software engineering.

Šimoňák [21] concludes, that studies in which students only viewed visualizations, usually did not indicate significant learning advantages over students using conventional learning materials. It can mean that using visualizations does not guarantee students' better understanding of algorithms.

3 Experiments and results

We conducted two experiments with four study groups of the two subjects of the Algorithms and Data Structures – (1) Hashing, (2) Trees (BST, RBT, AVL), – taught second-year bachelor students. In the first experiment, we had two study groups (A1, B1). These study groups were prepared on the subjects (1) and (2) by the same level of teaching technique (same level of explanation by the same teacher using the traditional approach). The difference was that that group A1 used visualization tools while group B1 did not apply such tools. We divided both groups as A1_h, B1_h when we collected results of experiment 1 on the "Hashing" topics, A1_t, B1_t – in case then students were learning "Trees" topics.

The second experiment involved two groups – A2, B2. They also studied Hashing (A2_h, B2_h) and Trees (A2_t, B2_t) topics. Unlike the first experiment, in the second experiment, we used flipped classroom concept for group A2 as a pedagogical approach. Study group B2 was prepared by the same teacher in a standard way using the whiteboard.

In the first experiment, the number of students in group A1 was 28, group B1 had 26 students. In the second experiment, the number of students in group A2 was 31, the number of students in group B2 was 25.

The tests at the end of studying topics of subjects (1) Hashing, (2) Trees (BST, RBT, AVL) consisted of four questions (0.5 points each) and were provided using a free and open-source learning management system Moodle and Google Forms. The results of the tests by groups and topics are shown in figures 3 (for experiment 1) and 4 (for experiment 2).

In both experiments questions on Hashing were selected from the set of questions from trivial hashing, chaining for collision handling, open addressing, double hashing, perfect hashing. For a test on Trees subjects, we selected questions from BST, RBT, and AVL trees.

The total scores achieved by students from group A1 on Hashing (A1_h) and Trees (A1_t) were 30 and 22 respectively. Average scores within the group A1 on Hashing (A1_h) and Trees (A1_t) were approximately 1.071 points and 0.789 points respectively.

The total scores for group B1 on Hashing (B1_h) and Trees (B1_t) were 20 and 20 respectively. Average scores within the group B1 on Hashing (B1_h) and Trees (B1_t) were approximately 0.769 points for B1_h and B1_t.

The total scores achieved by students from group A2 on Hashing (A2_h) and Trees (A2_t) were 32 and 31.5 respectively. Average scores within the group A2 on Hashing (A2_h) and Trees (A2_t) were approximately 1.032 points and 1.016 points respectively. The total scores for group B2 on Hashing (B2_h) and Trees (B2_t) were 19 and 19.5 respectively. Average scores in group B2 were approximately 0.760 points for B2_h and 0.780 for B2_t.

Table 1. Statistical results of experiments

	Experiment 1: Visualization				Experiment 2: Flipped classroom			
	Hashing		Trees		Hashing		Trees	
	A1_h	B1_h	A1_t	B1_t	A2_h	B2_h	A2_t	B2_t
Number of students	28	26	28	26	31	25	31	25
Total score	30	20	22	20	32	19	31.5	19.5
Average	1.071	0.769	0.786	0.769	1.032	0.760	1.016	0.780
Variance	0.587	0.505	0.397	0.445	0.566	0.523	0.541	0.481
Standard deviations	0.766	0.710	0.630	0.667	0.752	0.723	0.736	0.693

Results of the first and second experiments are presented in table 1. From the results we can conclude the following:

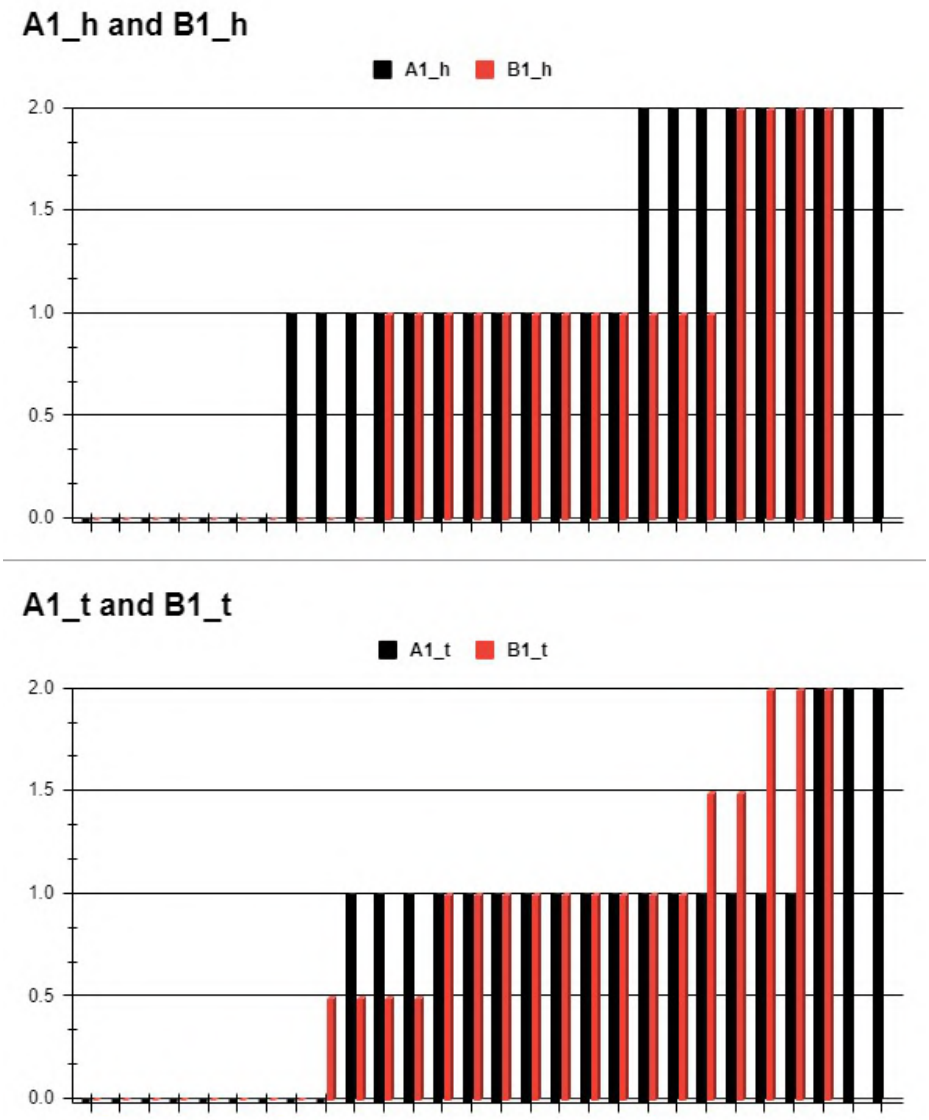

Fig. 3. Scores were achieved by students in the first experiment.

– In the first experiment, group A1 that used visualizations achieved better results. In the case of topic (1) Hashing the difference in average score was 1.071 vs. 0.769, with a standard deviation of 0.766 and 0.710 respectively. In the case of topic (2) Trees, the difference in average score was 0.786 vs. 0.769, with a standard deviation of 0.630 and 0.667 respectively. Thus, the

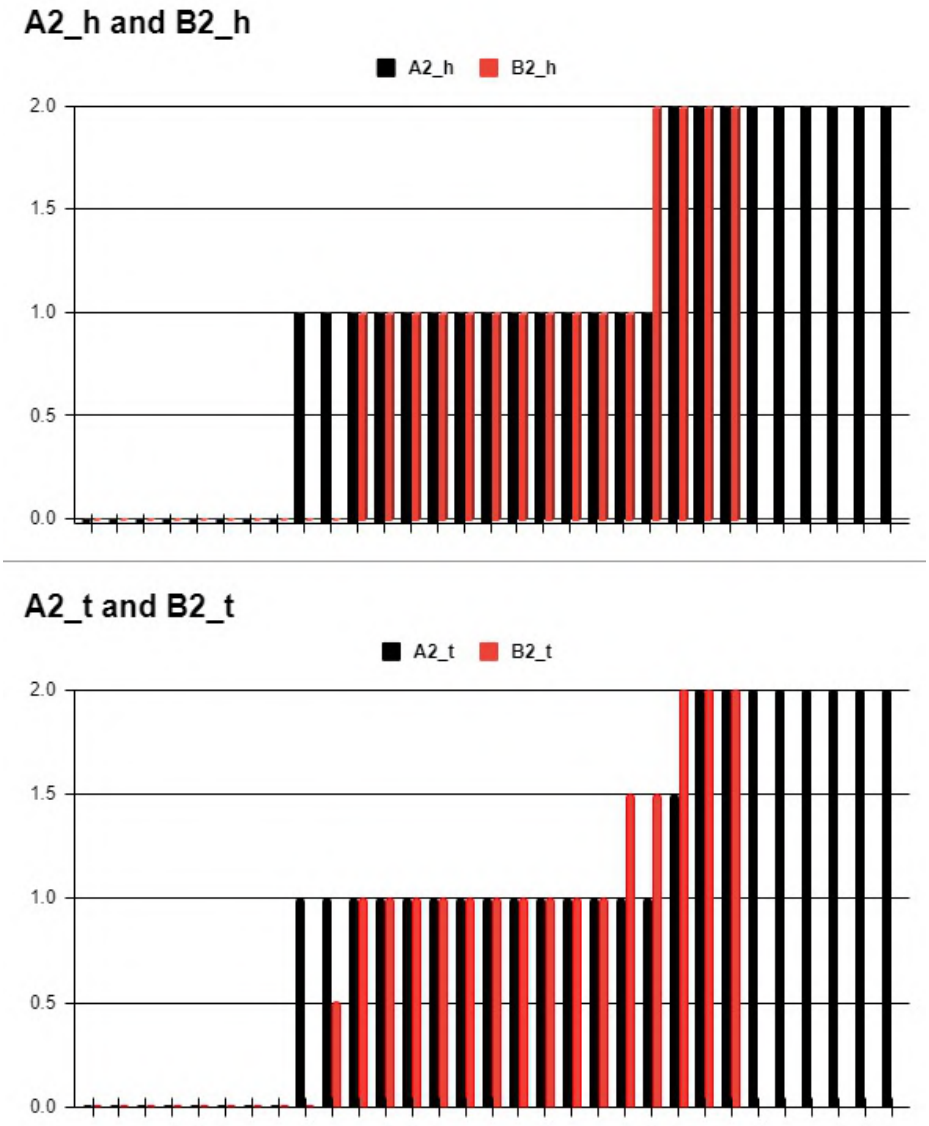

Fig. 4. Scores were achieved by students in the second experiment.

result for the topic (1) was much better for group A1 than for group B1. The result for the topic (2) was almost the same for both groups A1 and B1.

– In the second experiment, group A2 that used flipped classroom concept achieved better results. In the case of topic (1) Hashing the difference in average score was 1.032 vs. 0.760, with a standard deviation of 0.752 and 0.723 respectively. In the case of topic (2) Trees, the difference in average

score was 1.016 vs. 0.780, with a standard deviation of 0.723 and 0.693 respectively. Thus, the result for topics (1) and (2) was much better for group A1 than for group B1.

4 Conclusions

There are a lot of approaches to the implementation of some ways of increasing the efficiency of teaching data structures (hashing, trees). We conducted two experiments with four study groups of two subjects of the Algorithms and Data Structures – (1) Hashing, (2) Trees (BST, RBT, AVL), – taught second-year bachelor students. In the first experiment, study groups were prepared on the subjects (1) and (2) by the same level of teaching technique with or without visualization tools. Unlike the first experiment, in the second experiment, we used flipped classroom concept for one study group and a standard way of teaching for another group.

Flipped classroom and visualization as pedagogical tools can be used to teach students the necessary skills and competencies in algorithms and data structures.

References

1. Bikanga Ada, M.: Teaching algorithms and data structures: A tale of two approaches (2020). https://doi.org/10.25416/NTR.13302383.v1, https://eprints.gla.ac.uk/237310/
2. Budiman, E., Haeruddin, Hairah, U., Alameka, F.: Mobile learning: Visualizing contents media of data structures course in mobile networks. Journal of Telecommunication, Electronic and Computer Engineering **10**(1-9), 81–86 (2018)
3. Budiman, E., Pusnitasari, N., Wati, M., Haeruddin, Widians, J.A., Tejawati, A.: Mobile learning media for computer science course. In: 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC). pp. 262–267 (2018). https://doi.org/10.1109/KCIC.2018.8628559
4. Burbules, N.C., Fan, G., Repp, P.: Five trends of education and technology in a sustainable future. Geography and Sustainability **1**(2), 93–97 (2020). https://doi.org/10.1016/j.geosus.2020.05.001
5. Chang, C., Yang, Y., Tsai, Y.: Exploring the engagement effects of visual programming language for data structure courses. Education for Information **33**(3), 187–200 (2017). https://doi.org/10.3233/EFI-170108
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. The MIT Press, third edn. (2009), https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf
7. Hendrik, H., Hamzah, A.: Flipped classroom in programming course: A systematic literature review. International Journal of Emerging Technologies in Learning **16**(02), 220–236 (2021), https://online-journals.org/index.php/i-jet/article/view/15229
8. Hossain, M.M.: Application of Flipped Learning Approach in Computing Education. Master's thesis, Itä-Suomen yliopisto (2020), https://erepo.uef.fi/bitstream/handle/123456789/23068/urn_nbn_fi_uef-20200916.pdf

9. Ifenthaler, D.: Computer simulation model. In: Encyclopedia of the Sciences of Learning, pp. 710–713. Springer US, Boston, MA (2012). https://doi.org/10.1007/978-1-4419-1428-6_500
10. Karavirta, V., Korhonen, A., Malmi, L., Naps, T.: A comprehensive taxonomy of algorithm animation languages. Journal of Visual Languages & Computing **21**(1), 1–22 (Feb 2010). https://doi.org/10.1016/j.jvlc.2009.09.001
11. Knutas, A., Herala, A., Vanhala, E., Ikonen, J.: The flipped classroom method: Lessons learned from flipping two programming courses. In: Proceedings of the 17th International Conference on Computer Systems and Technologies 2016. p. 423–430. CompSysTech '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2983468.2983524
12. Lee, G.C., Lee, P.L.: Data structures in flipped classroom: Students' effort and preference. In: 2015 International Conference on Learning and Teaching in Computing and Engineering. pp. 152–155 (2015). https://doi.org/10.1109/LaTiCE.2015.28
13. Ling, E.W.M., Li, C.Y.Y., Deni, A.R.M.: Promoting student engagement using flipped classroom in large introductory financial accounting class. In: Proceedings of the 2019 3rd International Conference on Education and E-Learning. p. 61–66. ICEEL 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3371647.3371658
14. Mocinecová, K., Steingartner, W.: Software Support for Visualizing of the Graph Algorithms in a Novel Approach in Educating of Young IT Experts. IPSI Transactions on Internet Research **16**(2), 14–23 (2020), http://ipsitransactions.org/journals/papers/tir/2020jul/p3.pdf
15. Mutua, S., Wabwoba, F., Ogao, P., Anselmo, P., Abenga, E.: Classifying program visualization tools to facilitate informed choices: teaching and learning computer programming. International Journal of Computer Science and Telecommunications **3**(2), 42–48 (2012), http://www.ijcst.org/Volume3/Issue2/p8_3_2.pdf
16. Peethambaran, M.K.P., Renumol, V.G., Murthy, S.: Flipped Classroom Strategy to Help Underachievers in Java Programming. In: 2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE). pp. 44–49. IEEE (2018)
17. Ševčíková, A., Milková, E.: Multimedia applications: Graph algorithms visualization. In: 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI). pp. 000231–000236 (2016). https://doi.org/10.1109/CINTI.2016.7846409
18. Šimoňák, S.: Algorithm visualization using the vizalgo platform. Acta Electrotechnica et Informatica **13**(2), 54 (2013), https://www.deepdyve.com/lp/de-gruyter/algorithm-visualization-using-the-vizalgo-platform-zoCNHQpqtx
19. Šimoňák, S.: Using algorithm visualizations in computer science education. Central European Journal of Computer Science **4**(3), 183–190 (Sep 2014). https://doi.org/10.2478/s13537-014-0215-4
20. Šimoňák, S.: 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). pp. 153–157. IEEE (2016)
21. Šimoňák, S.: Increasing the engagement level in algorithms and data structures course by driving algorithm visualizations. Informatica **44**(3) (2020), http://www.informatica.si/index.php/informatica/article/view/2864
22. Šimoňák, S., Benej, M.: Visualizing algorithms and data structures using the algomaster platform. Journal of Information, Control and Management Systems **12**(2), 189–201 (2014), https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1005.9137&rep=rep1&type=pdf

23. Stasko, J.: Data structure visualization. In: Mehta, D.P., Sahni, S. (eds.) Handbook of Data Structures and Applications, pp. 697–705. Chapman and Hall/CRC, Boca Raton, 2nd edn. (2018). https://doi.org/10.1201/9781315119335
24. Striuk, A.M.: Problematic questions of software requirements engineering training. Educational Dimension **56**(4), 90–101 (Mar 2021). https://doi.org/10.31812/educdim.v56i4.4441, https://journal.kdpu.edu.ua/ped/article/view/4441
25. Supli, A.A., Shiratuddin, N., Zaibon, S.B.: Critical analysis on algorithm visualization study. International Journal of Computer Applications **150**(11) (2016). https://doi.org/10.5120/ijca2016911633
26. Tyler, B., Abdrakhmanova, M.: Flipping the CS1 and CS2 classrooms in Central Asia. In: 2016 IEEE Frontiers in Education Conference (FIE). pp. 1–5. IEEE (2016). https://doi.org/10.1109/FIE.2016.7757739
27. Vlachopoulos, P., Jan, S.K., Buckton, R.: A case for team-based learning as an effective collaborative learning methodology in higher education. College Teaching **69**(2), 69–77 (march 2020). https://doi.org/10.1080/87567555.2020.1816889
28. Wang, F.: The design of public computer lab management system based on network environment. In: Proceedings of the 2016 International Conference on Education, Management and Computer Science. pp. 263–267. Atlantis Press (2016/05). https://doi.org/10.2991/icemc-16.2016.55