Automatic, Suggestive Feedback in Algorithm Visualisation Exercises

Artturi Tilanterä
Department of Computer Science
Aalto University
artturi.tilantera@aalto.fi

Abstract

This short paper discusses an ongoing doctoral research on the field of computing education research. The subject is undergraduate computing education at universities: teaching basic data structures and algorithms, meaning the principles of designing computer programs that solve tasks efficiently, without wasting computing time or memory. The research aims to deepen knowledge on what are students' difficulties in learning the subject, and how educational tools, such as visualisations and automatically assessed exercises, could help the student. Published and preliminary results display the technical and theoretical complexity of the subject.

1. Introduction

Computing has become ubiquitous: consider electric vehicles, mobile devices, balancing production and consumption of renewable energy, robotics, or Internet of Things. We need scientists and engineers capable of understanding computing regardless of whether their major is in computer science. University studies of computing typically begin with a basics of programming course on the idea of "How to instruct the computer to do new tasks?"

One typical course after the very basics introduces *data structures and algorithms* (DSA). The key aspect of this course is efficiency, as in "How to make the computer solve new tasks within reasonable processing time and memory?" Classic problems introduced on an DSA course are sorting a collection of data items according to some property, keeping the collection organized and rapidly searchable while adding and deleting items, and exploring alternative routes in a map or more abstract problem.

Teaching and learning programming is not a trivial psychological task. Since 1960s *computing education research* has revealed that for humans, it is hard to grasp how a computer runs a program, and how to formulate precise instructions for a computer. (Guzdial & du Boulay, 2019) Correspondingly, understanding data structures and algorithms is not trivial either, as it requires learning even more abstract concepts.

2. Background

One approach to improve the teaching of any STEM subject (Science, Technology, Engineering, and Mathematics) is to list the important concepts that students should learn. Moreover, the list can be refined into a set of multiple-choice questions which test students' knowledge across different courses in different universities. This carefully selected and validated set of questions is called a *concept inventory*. A concept inventory for data structures and algorithms has been built recently. (Porter et al., 2019)

Regardless of quality of instruction, students may struggle by learning an incorrect mental model while thinking they understood the subject; this is called a *misconception*. For example, a physics misconception is that summer is warmer than winter, because the Earth is closer to the sun on its orbit. Misconceptions can be seen to relate concept inventories in a way that they are popular way of students diverging from the correct mental model, even encountering confusion later in their studies. DSA misconceptions are known to exist for various subtopics, for example recursion (Götschi, Sanders, & Galpin, 2003) and heaps (Seppälä, Malmi, & Korhonen, 2006).

Data structures and algorithms are abstract, time-dependent concepts which are dynamic by nature: an algorithm processes data over time following certain instructions, and the data is converted into a

result of computation. Therefore *algorithm visualisation* has been studied as a pedagogical tool: data structures can be represented as figures of one- or two-dimensional tables (arrays), chains (lists), trees, and networks (graphs). For example, Figure 1 shows an example of a visualisation of a graph: *nodes* labeled by letters A–P may represent road intersections, while the lines between the nodes, *edges*, can represent road segments, with numbers representing the length of the roads. The figure is a part of an image sequence depicting how the computer finds shortest paths from the starting node A (Dijkstra's algorithm). Essentially, the algorithm adds edges to the tree of shortest paths one by one. The brown color represents all possible road segments, while the yellow color represents choices already made by the algorithm.

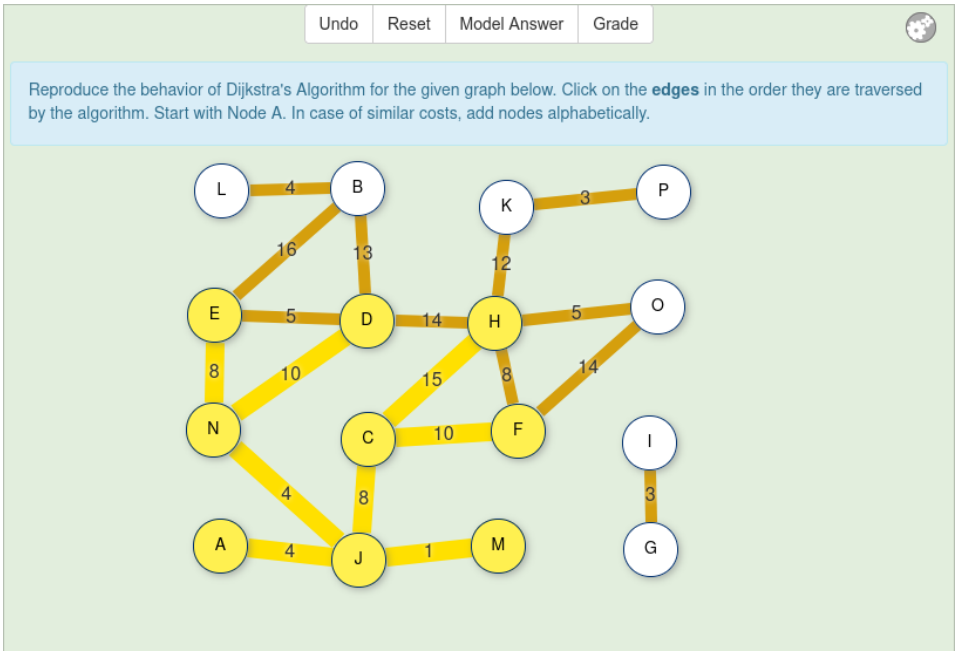

Figure 1 – A visual algorithm simulation exercise on Dijkstra's algorithm for single-source shortest paths.

Algorithm visualisation as a learning aid has been studied for decades. Many computing educators believe in the power of visualisation as a learning aid. However, the educational effectiveness of visualisation depends on what the student does with it, not the careful design of a visual presentation. A student can *engage* many ways with the visualisation. It seems that interaction (e.g. constructing, responding to questions) has significant learning effect compared to passively viewing e.g. an animation. (Naps et al., 2002) A meta-research of the algorithm visualisation (Shaffer et al., 2010) revealed that the visualisation tools are often low-quality, emphasizing easy topics, and hard to find for instructors. Therefore, to support instructors with a collection of algorithm visualisations, an open-source, community-based interactive textbook named OpenDSA was established (Shaffer, Karavirta, Korhonen, & Naps, 2011; *OpenDSA*, n.d.).

One particular type of engaging algorithm visualisation is *visual algorithm simulation* (VAS) exercises, where the student simulates the steps of an algorithm with certain input. For example, the whole Figure 1 is a screenshot of a VAS exercise about Dijkstra's algorithm. The input (graph) is given, and the student must click on the edges of the graph one by one in the same order as the algorithm adds them to the tree of shortest paths. The exercise is assessed automatically: when the student clicks the Grade button, they receive a grade depending on the number of correct steps in the simulation. These kind of interactive, visual exercises with automatic assessment are currently implemented with the JSAV software in OpenDSA (Karavirta & Shaffer, 2013, 2016), while Matrix / TRAKLA2 (Malmi et al., 2004) is an older software with the same principles. The exercise in Figure 1 has been implemented with JSAV.

3. Research problems and methods

The overarching research problem in this doctoral research is to give automated feedback for learners in VAS exercises at a Data Structures and Algorithms course in tertiary education. The feedback should be *suggestive*, meaning that it contains information on how to proceed. The following general research questions characterise the publications.

- RQ1. How data structures and algorithms should be taught?
- RQ2. What are students' key difficulties in DSA?
- RQ3. How can we give automatic, corrective feedback in DSA exercises, especially VAS?

An answer to RQ1 and RQ2 is sought by a systematic literature review on computing education research related to data structures and algorithms. It is expected that we will have an organised overview of this subtopic listing the key methods, problems, and misconceptions. For example, how has the field of algorithm visualisation evolved since of the report (Shaffer et al., 2010) ?

Another perspective to RQ2 begins with an existing study (Nelson et al., 2020), which includes a problem set to tests students' knowledge related to both programming and DSA. Next, a qualitative research will test the problem set empirically. What information does the problem set reveal in practice when students are asked to solve it?

VAS exercises can be used to study student's misconceptions of DSA (see, e.g. (Seppälä et al., 2006)), thus providing answers to RQ2. Quantitative analysis of students' solutions to VAS exercises combined with think-aloud interviews of students solving the same exercises is expected to reveal common misconceptions among students. Similarly, it would be interesting to see whether the same misconceptions appear in students' solutions to DSA-related programming exercises.

RQ3 seeks to improve the quality of automatically assessed DSA exercises. The JSAV-based VAS exercises only report the correctness of the answer to the student. The student might like to see where they diverged from the correct solution, or even an automatic hint on how to proceed. Correspondingly, the automatic assessment of programming exercises is relatively easy to implement by designing unit tests for the student's program and reporting to the student which tests did not pass. Answers to RQ1 and RQ2 might help constructing elaborate, suggestive automatic feedback. The technical challenges of producing such feedback is a research problem itself.

4. Publications

Publication I. Algorithm visualisation and the Elusive Modality Effect

I assisted a research related to the cognitive effects of multimedia instruction, a so-called *modality effect*. The research involved an experiment where students were exposed to different versions of learning videos of the Dijkstra's algorithm, while tested with multiple-choice questions and a VAS exercise Figure 1. The experiment failed to show the existence of the modality effect, which seems to imply that the phenomenon is more complex than expected. However, simultaneously, the piloted a software to record students' solutions to the VAS exercise on Dijkstra's algorithm in a real learning context: an electronic material of a DSA course provided by the A+ Learning Management System (Karavirta, Ihantola, & Koskinen, 2013).

Publication II. Towards a JSON-based Algorithm Animation Language

The software development experience related to the modality effect research (Zavgorodniaia et al., 2021) was discussed in another paper, (Tilanterä, Mariani, Korhonen, & Seppälä, 2021). The conclusion was that software and storage format which the authors developed for recording solutions to JSAV-based VAS exercises required further development. The paper presents new requirements for the software.

5. Emerging results

This section describes emerging results which are not yet published.

(Submitted) Empirical Evaluation of Differentiated Assessment of Data Structures: The Role of Prerequisite Skills. Marjahan Begum, Pontus Haglund, Ari Korhonen, Filip Strömbäck, Artturi Tilanterä.

The authors evaluated the DSA-related question set designed in (Nelson et al., 2020) with students on a DSA course in two institutions. We interviewed total 18 students on their written answers to the question set. Qualitative content analysis of the results revealed that some of the questions do indeed reveal students' fragile knowledge about basics of programming or DSA. However, the extent to which we can pinpoint student's lack of knowledge varies by question. Moreover, there seems to be skills which are not taught explicitly on an introductory programming course, but which help students on the following DSA course, such as program tracing and the reasoning about constraints of program execution; these are called *middle-ground skills*.

(Work in progress) Students' Misconceptions of Dijkstra's Algorithm in Visual Algorithm Simulation. Artturi Tilanterä, Ari Korhonen, Otto Seppälä, Juha Sorva.

The software described in Publication II has been developed further. There is now a formal specification of the storage format as JSON Schema. The recorder software has been updated according to the redesign of the storage format (Mariani, Sänger, & Tilanterä, n.d.). We have collected hundreds of students' solutions to the exercise in Figure 1 with the updated software. Moreover, we have conducted think-aloud interviews where some students solve the same exercise. This mixed-methods research aims to find hypotheses for misconceptions in the interviews and then strengthen the evidence by quantitative analysis of the automatically collected solution data.

(Work in progress) Computing Education Research on Data Structures and Algorithms. Artturi Tilanterä.

Related to RQ1, I have conducted a preliminary literature review as a part of my doctoral studies. It seeks answers to the following questions: (i) What literature exists on computing education research related to data structures and algorithms? (ii) How should the topics in the existing literature be categorized? The material is:

1. Jan Vahrenhold's publications (University of Münster, 2022).
2. Proceedings of SIGCSE from year 2000 onward (ACM, 2022b)
3. Proceedings of ITiCSE from year 2000 onward (ACM, 2022a)
4. The bibliography in Christopher Hundhausen's dissertation (Hundhausen, 1999)

I have first screened the publications by title and then by abstract. Rudimentary topic-based categorization reveals the following. Basic DSA (linear structures, trees, graphs, hashing, sorting, complexity) has been discussed in 204 publications. Teaching of advanced topics, such as algorithmic techniques (e.g. dynamic programming, approximation, heuristics), and applications of algorithms (e.g. data compression, computational geometry, networking, optimization) have been studied to a lesser extent (22 and 23 publications, respectively). Based on the preliminary results, there is a plan to conduct a more comprehensive literature review on the topic which could lead to a journal publication or a concept inventory.

6. Conclusion

This short paper summarized two years of doctoral research related to teaching data structures and algorithms. Besides empirical experiments, the research has involved developing suitable research soft-

ware and conducting a literature review. The research problem is manyfold, involving misconceptions, middle-ground skills, visualisation, and automatic assessment and feedback.

7. References

ACM. (2022a). *Iticse conference - proceedings.* Retrieved 2022-08-11, from https://dl.acm.org/conference/iticse/proceedings

ACM. (2022b). *Sigcse conference - proceedings.* Retrieved 2022-08-11, from https://dl.acm.org/conference/sigcse/proceedings

Götschi, T., Sanders, I., & Galpin, V. (2003). Mental models of recursion. In *Proceedings of the 34th sigcse technical symposium on computer science education* (pp. 346–350). New York, NY, USA: ACM. doi: 10.1145/611892.612004

Guzdial, M., & du Boulay, B. (2019). The history of computing education research. In S. A. Fincher & A. V. Robins (Eds.), *The cambridge handbook of computing education research* (p. 11–39). Cambridge University Press. doi: 10.1017/9781108654555.002

Hundhausen, C. D. (1999). *Toward effective algorithm visualization artifacts: Designing for participation and communication in an undergraduate algorithms course* (Doctoral dissertation, Graduate School of the University of Oregon). Retrieved 2023-01-11, from https://eecs.wsu.edu/~veupl/proj/dis/Tech%20Report.pdf

Karavirta, V., Ihantola, P., & Koskinen, T. (2013, 7). Service-oriented approach to improve interoperability of e-learning systems. In *2013 ieee 13th international conference on advanced learning technologies* (p. 341-345). IEEE. doi: 10.1109/ICALT.2013.105

Karavirta, V., & Shaffer, C. A. (2013). Jsav: The javascript algorithm visualization library. In *Proceedings of the 18th acm conference on innovation and technology in computer science education* (pp. 159–164). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2462476.2462487 doi: 10.1145/2462476.2462487

Karavirta, V., & Shaffer, C. A. (2016, 4). Creating engaging online learning material with the jsav javascript algorithm visualization library. *IEEE Transactions on Learning Technologies*, *9*(2), 171-183. doi: 10.1109/TLT.2015.2490673

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., & Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: Trakla2. *Informatics in Education*, *3*(2), 267–288. doi: 10.15388/infedu.2004.19

Mariani, G., Sänger, J., & Tilanterä, A. (n.d.). *Jsav exercise recorder.* Retrieved 2023-04-13, from https://github.com/Aalto-LeTech/jsav-exercise-recorder/tree/jaal2.0 (Software version 2.0)

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., ... Velázquez-Iturbide, J. A. (2002). Exploring the role of visualization and engagement in computer science education. In *Working group reports from iticse on innovation and technology in computer science education* (p. 131–152). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/960568.782998 doi: 10.1145/960568.782998

Nelson, G. L., Strömbäck, F., Korhonen, A., Albluwi, I., Begum, M., Blamey, B., ... Monga, M. (2020). Assessing how pre-requisite skills affect learning of advanced concepts. In *Proceedings of the 2020 acm conference on innovation and technology in computer science education* (p. 506–507). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3341525.3394990 doi: 10.1145/3341525.3394990

Opendsa. (n.d.). Retrieved 2023-04-13, from https://opendsa-server.cs.vt.edu/

Porter, L., Zingaro, D., Liao, S. N., Taylor, C., Webb, K. C., Lee, C., & Clancy, M. (2019). Bdsi: A validated concept inventory for basic data structures. In *Proceedings of the 2019 acm conference on international computing education research* (pp. 111–119). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3291279.3339404 doi: 10.1145/3291279.3339404

Seppälä, O., Malmi, L., & Korhonen, A. (2006). Observations on student misconceptions—a case study

of the build – heap algorithm. *Computer Science Education*, *16*(3), 241 - 255. doi: 10.1080/08993400600913523

Shaffer, C. A., Cooper, M. L., Alon, A. J. D., Akbar, M., Stewart, M., Ponce, S., & Edwards, S. H. (2010, 8). Algorithm visualization: The state of the field. *ACM Trans. Comput. Educ.*, *10*(3). Retrieved from https://doi.org/10.1145/1821996.1821997 doi: 10.1145/1821996.1821997

Shaffer, C. A., Karavirta, V., Korhonen, A., & Naps, T. L. (2011). Opendsa: Beginning a community active-ebook project. In *Proceedings of the 11th koli calling international conference on computing education research* (p. 112–117). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2094131.2094154 doi: 10.1145/2094131.2094154

Tilanterä, A., Mariani, G., Korhonen, A., & Seppälä, O. (2021). Towards a json-based algorithm animation language. In *2021 working conference on software visualization (vissoft)* (p. 135-139). doi: 10.1109/VISSOFT52517.2021.00026

University of Münster. (2022). *Publications - jan vahrenhold.* Retrieved 2022-08-11, from https://www.uni-muenster.de/Informatik.AGVahrenhold/en/personen/prof.dr.janvahrenhold/publikationen.shtml

Zavgorodniaia, A., Tilanterä, A., Korhonen, A., Seppälä, O., Hellas, A., & Sorva, J. (2021). Algorithm visualization and the elusive modality effect. In *Proceedings of the 17th acm conference on international computing education research* (p. 368–378). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3446871.3469747 doi: 10.1145/3446871.3469747