Novice Difficulties in Graph Layering for Algorithm Design

Hongxuan Chen
University of Illinois Urbana-Champaign
Urbana, IL, USA
hc10@illinois.edu

Katherine Braught
University of Illinois Urbana-Champaign
Urbana, IL, USA
braught2@illinois.edu

Geoffrey L. Herman
University of Illinois Urbana-Champaign
Urbana, IL, USA
glherman@illinois.edu

Jeff Erickson
University of Illinois Urbana-Champaign
Urbana, IL, USA
jeffe@illinois.edu

Abstract

Graph data structures and algorithms play an essential role in computer science, and one of the ultimate goals of learning graphs is to solve more complicated algorithm design problems with them. A common way to solve a novel, complex problem is to reduce the problem to a standard graph problem, which often requires modeling a graph, and one essential way to model a graph is a technique called *graph layering*. Graph layering is often considered difficult by students and rarely studied by computer science education researchers despite its significance in algorithm design. To understand students' struggles with graph layering and improve teaching of algorithm designs, we conducted this qualitative study using think-aloud interviews with current students from an algorithm course. Participants were asked to solve algorithm design problems meant to be solved with graph layering. We used thematic analysis to extract difficulties observed in these interviews. We share our preliminary findings in this poster, and propose next steps for this study and future research.

ACM Reference Format:
Hongxuan Chen, Katherine Braught, Geoffrey L. Herman, and Jeff Erickson. 2025. Novice Difficulties in Graph Layering for Algorithm Design. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3641555.3705221

1 Introduction and Background

We teach students data structures and standard algorithms, but eventually we want them to design algorithms on their own to solve real-life problems [5]. One common approach of algorithm design is to reduce the problem to a well-known one, and then apply a standard algorithm as a black box [9], and a large subset of these canonical problems are formalized in terms of graphs, which are a natural and powerful abstraction for many computational problems. While there have been a variety of studies about teaching standard graph algorithms [2, 4, 6, 7, 10], using graphs in algorithm design is often overlooked in teaching and educational research.

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-0532-8/25/02
https://doi.org/10.1145/3641555.3705221

In real-life problems, graphs are not naturally provided. Instead, one needs to model a graph to represent the inputs and conditions before calling a black box algorithm. Sometimes the modeling process is easy and intuitive, such as defining locations on a map as vertices and roads as edges. However, many problems cannot be solved by applying a standard algorithm to the most obvious graph; instead we must first construct a more complex graph. One form of advanced graph modeling is an approach informally named *graph layering*, which is to create multiple copies (layers) of the vertices of the "base graph" and define the intra- and inter-layer edges properly. This approach is often used when there are extra requirements on top of a standard problem, such as "find the shortest walk between two points whose length is divisible by 3" (see Figure 1).

While there have been many studies about how students learn the implementation of standard graph algorithms [2, 6, 10], little is known about how to teach advanced graph modeling for problem reduction in algorithm design. To investigate how students understand and apply graph layering, we conducted this qualitative research aiming to investigate students' thinking processes when applying the graph layering technique. This study aims to answer the following research question (RQ): **What are the difficulties novices encounter when they solve algorithm design problems that require graph layering?**

We used think-aloud interviews with 15 students from an algorithm course covering graph layering (and other modeling techniques) in Spring 2024 at a large public research university in the United States. We consider our participants novices in graph layering since the interviews happened shortly after they learned this topic in class and practiced it in labs and homework. Our study was heavily inspired by Zehra et al. [11] and Shindler et al. [8] for the similarity of research procedures and relevance of topics, since they investigated students' misconceptions in dynamic programming. We used thematic analysis [1] to identify challenges students encountered while solving graph layering problems, and here we share our preliminary findings.

2 Think-aloud Interviews

We conducted "think-aloud" interviews with the participants [3], where they were asked to solve algorithm design problems and vocalize their thought process. Each interview was around one hour and had three problems, but only a few participants reached the last problem within the interview period. Therefore, we only include the first two problems in this poster.

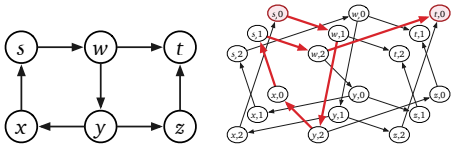

Figure 1: An example of base graph and derived layered graph for the problem "find the shortest walk from s to t whose number of edges is divisible by 3".

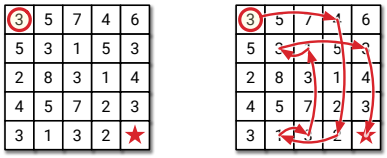

Figure 2: The number maze used in Problem 1. Numbers on each square specify the exact distance that a token can travel, in any cardinal direction that keeps the token on the board.

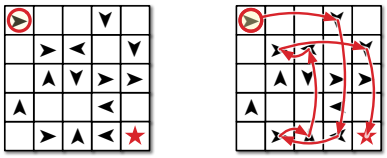

Figure 3: The arrow maze used in Problem 2. Arrows on each square specify the direction that a token can travel in, for any distance that keeps the token on the board.

Problem 1: Find a shortest sequence of legal moves that move a token from the top-left corner to the bottom-right corner in a number maze (Figure 2).
Problem 2: Find a shortest sequence of legal moves that move a token from the top-left corner to the bottom-right corner in an arrow maze (Figure 3).

For Problem 1, half participants were asked to find a shortest move sequence *whose length is divisible by 5*, and the other half were asked to find a shortest move sequence *containing each of the cardinal directions at least once*. For Problem 2, they were given the other additional constraint they did not receive for Problem 1.

These problems have been deliberately designed so that they can be solved by constructing an appropriate graph and invoking a standard shortest-path algorithm. These problems could not be easily solved using dynamic programming or naive greedy algorithms, which were also recently covered in the participant's algorithms course. Therefore, a significant part of the answering the questions included choosing which algorithm design approach to take.

All interviews were conducted over Zoom by the first author, and audio recordings and video recordings of participants' screens with their typed solutions were used for data analysis.

3 Thematic Analysis

We analyzed the interviews using thematic analysis [1], with the first and the second authors conducting the coding while the third and the fourth authors providing guidance and advice. First, we (first and second authors) watched the recordings and created initial codebooks independently. Then, we grouped interesting codes into themes and created a new, combined codebook after reaching a consensus. Using the proposed themes and updated codebook, we watched all videos again using a deductive approach, where we either updated existing codes or added new ones. After coding all videos,we discussed and resolved disagreements so that all codes converged. Lastly, codes were grouped into higher-level themes.

Based on the thematic coding results, the common themes in mistakes that our participants made were:

- Use of dynamic programming when not applicable.
- Greedy solution based on misconceptions of BFS/Dijkstra's.
- Incorrect graph layering construction.

Due to page limits, relevant quotes are not included in this extended abstract, but they are included in the actual poster.

4 Future Work

This poster presents the preliminary results of our observations, and our next step is to explore possible reasons behind these difficulties We will also study ways to improve the teaching of graph modeling and other algorithm design topics.

Acknowledgments

This work was partially supported by the Strategic Instructional Innovations Program at the University of Illinois Urbana-Champaign.

References

[1] Virginia Braun and Victoria Clarke. 2012. *Thematic analysis.* American Psychological Association.
[2] Bekir Cevizci. 2018. Calculating the Shortest Path Using Dijkstra's Algorithm. *Journal of Inquiry Based Activities* 8, 2 (2018), 70–85.
[3] Elizabeth Charters. 2003. The use of think-aloud methods in qualitative research an introduction to think-aloud methods. *Brock Education Journal* 12, 2 (2003).
[4] J. Paul Gibson. 2012. Teaching graph algorithms to children of all ages. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (Haifa, Israel) *(ITiCSE '12).* Association for Computing Machinery, New York, NY, USA, 34–39. https://doi.org/10.1145/2325296.2325308
[5] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. 2024. *Computer Science Curricula 2023.* Association for Computing Machinery, New York, NY, USA.
[6] Janka Medová, Kitti Páleníková, L'ubomír Rybanskỳ, and Zuzana Naštická. 2019. Undergraduate students' solutions of modeling problems in algorithmic graph theory. *Mathematics* 7, 7 (2019), 572.
[7] M. Gloria Sánchez-Torrubia, Carmen Torres-Blanc, and Sonia Escribano-Blanco. 2010. GRAPHs: a learning environment for graph algorithm simulation primed for automatic fuzzy assessment. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '10).* Association for Computing Machinery, New York, NY, USA, 62–67.
[8] Michael Shindler, Natalia Pinpin, Mia Markovic, Frederick Reiber, Jee Hoon Kim, Giles Pierre Nunez Carlos, Mine Dogucu, Mark Hong, Michael Luu, Brian Anderson, et al. 2022. Student misconceptions of dynamic programming: a replication study. *Computer Science Education* 32, 3 (2022), 288–312.
[9] Steven S Skiena. 1998. *The algorithm design manual.* Vol. 2. Springer.
[10] Artturi Tilanterä, Juha Sorva, Otto Seppälä, and Ari Korhonen. 2024. Students Struggle with Concepts in Dijkstra's Algorithm. In *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1* (Melbourne, VIC, Australia) *(ICER '24).* Association for Computing Machinery, New York, NY, USA, 154–165.
[11] Shamama Zehra, Aishwarya Ramanathan, Larry Yueli Zhang, and Daniel Zingaro. 2018. Student misconceptions of dynamic programming. In *Proceedings of the 49th ACM technical symposium on Computer Science Education.* 556–561.