Algorithmic Intuition

Evan M. Gertis, Charles Gary

April 26, 2021

Abstract

The purpose of this study was to learn how algorithm visualizations affect a student's ability to conceptualize algorithmic processes. Our hypothesis is that students should be able to answer questions on tree traversals more effectively and more quickly after having been exposed to an algorithm visualizations. We used depth first search and breadth first search algorithm visualizations to test our hypothesis. We developed a fully functioning algorithm visualization system. We used our system to test the affects of using visualizations as a teaching aid. Since learning how algorithms work is an essential part of a Computer Science program we believe that this will be a valuable contribution to the field. Our research will help shed light on how students develop algorithmic intuition.

1 Introduction

Most computer science students experience difficulties visualizing algorithms. Although algorithm visualizations are well received by computer science students they do not always lead to overall improvement of algorithmic intuition. The fundamental question is do they help students develop intuition for algorithmic processes? This has been a topic of debate for quite some time with studies yielding mixed results. Some studies have shown that there is no significant difference, while other studies have shown that algorithm visualizations can improve the understanding of algorithmic processes.

This lack of evidence supporting the advantages of AV (algorithm visualizations) can be attributed to a multitude of factors such as usability, algorithm difficulty, animation quality, and experimental design. The questions that can be asked regarding AV then becomes:

- Are algorithm animations effective at educating computer science students?
- What makes for an educationally effective algorithm visualization?
- Should students construct their own, or view an expert's?
- Are interactive multimedia presentations better than static presentations?

In this study, we want to determine what features should be included and what should be avoided. We seek the answers to these questions and offer an AV implementation to test the effectiveness of the suggested best practices.

2 Literature review

The use of algorithm visualizations, particularly animation, is generally well-received by students and educators alike, as they typically agree that a dynamic visual representation of an algorithms' execution is better than a static textual explanation. For this reason, development of effective algorithm visualizations is central to the learning success of computer science students in their algorithm and data structures courses. Studies yielding contradictory or sub par results, however, have opened the door to speculation as to whether or not this is true.

For insight into the state of the field of AV and methods for designing educationally effective AV we considered the following two scholarly papers. [Shaffer et al., 2010] [Hansen et al., 2002]

2.1 Algorithm Visualization: The State of the Field

Hundreds of AVs exist, but not all AVs are created equal. Many of the available visualizations are of poor quality or focus on simpler algorithms making it difficult for educators to use them effectively. Unfortunately, there is no repository of the effective AVs available in one location. It can be said that, due to this fact, educators are at a disadvantage when it comes to sorting through the existing AVs to find the few gems. The authors set out in this paper to provide instructors with a list of effective AVs based on studies and user reviews, developers with a list of best practices for designing AVs, and a means for building community among developers, educators, researchers, and end-users.

In 2010, when this paper was published, the authors had dedicated four years to the creation of the AlgoViz Wiki. The AlgoViz Wiki is a catalog of more than 500 AVs (not including over 100 older systems that do not run in modern computers) organized by topic and link sites. AVs in the catalog are evaluated and assigned an editor rating of "Recommended", "Has Potential", and "Not Recommended" to help instructors easily find effective visualizations when needed. The authors note that majority of AVs are skewed toward freshman and sophomore level subjects, such as stacks, queues, and sorting algorithms, whereas subjects such as NP-complete problems and dynamic programming are underrepresented as show in fig 1.

Table I. Counts by Major Category (with Significant Subcategories) for Visualizations of Data Structures in the AlgoViz Catalog

Linear Structures	46	
Lists		13
Stacks & Queues		32
Search Structures	76	
Binary Search Trees		16
AVL Trees		8
Splay Trees		9
Red-Black Trees		13
B-Trees and variants		17
Skiplist		6
Spatial Search Structures	31	
Point representations		13
Rectangle representations		10
Other Data Structures	25	
Heap/Priority Queue		11

Table II. Counts by Major Category (with Significant Subcategories) for Visualizations of Algorithms in the AlgoViz Catalog

Search Algorithms	18	
Linear/Binary Search		5
Hashing		11
Sort Algorithms	130	
Sorting Overviews		8
Quadratic Sorts		25
Shell Sort		13
Quicksort		24
Mergesort		25
Heapsort		16
Radix and Bin Sort		14
Graph Algorithms	68	
Traversals and Searches		15
Shortest Paths		20
Spanning Trees		17
Network Flow		4
Compression Algorithms	18	
Huffman Coding		13
Networking & OS	10	
Dynamic Programming	9	
Computational Geometry	20	
String Matching	6	
$\mathcal{NP}$-complete Problems		9
Other Algorithms	47	
Recursion & Backtracking		10
Mathematical Algorithms		8

Figure 1: Visualization count by category

The hope is that awareness of this imbalance and building of a digital community where developers can interact and discuss best practices will spark development in these areas. Lack of communication leads to repeating the same mistakes, or reinventing the wheel, and can slow or halt the

development process.

2.2 Designing Educationally Effective Algorithm Visualizations

In this paper, written in 2002, the authors apply aspects of the learner-centered approach to designing effective algorithm visualizations. This approach focuses on engagement (be it with the instructor, the content, or other students) and developing the student's analysis, problem-solving, and decision-making skills. From this approach the authors present the Hypermedia Algorithm Visualization (HalVis) system which they developed to show the "instructional superiority of algorithm animations."

Hypermedia is described as the use of multiple media, semantic links and other cognitive devices to help the student form accurate mental models of algorithms. In hypermedia environments, visualizations are meant to involve more than just the visual, but should be an immersive experience that is engaging and includes animations, text, static diagrams, etc., allowing the student to make the appropriate temporal and spatial connections. Each algorithm visualization created using the HalVis system has five key features:

- An animated analogy
- Questions that stimulate thinking
- Three distinct kinds of animations
- Segmented animations (which users can play, pause, and slow down)
- Embedding animations with textual descriptions (pseudocode) and hyperlinks.

By designing AV that are carefully arranged with these features, the authors conclude that novice and advanced students perform better on tests than students studying typical textbook explanations and attending traditional lectures. Four experiments were conducted for comparison and analysis that led to this conclusion:

- HalVis vs textbook (using sophomore students and the MergeSort algorithm)
- HalVis vs textbook (using junior students and MergeSort and QuickSort algorithms)
- HalVis vs lecture (using sophomore students and MergeSort and SelectionSort algorithms)
- HalVis vs textbook/animation (using junior students and Shortest Path algorithm).

As shown in fig 2 Students in the HalVis group showed on average a 30% higher test score than students in the textbook/lecture group in the first three experiments, and a 20% higher test score compared to students in the textbook/animation group.

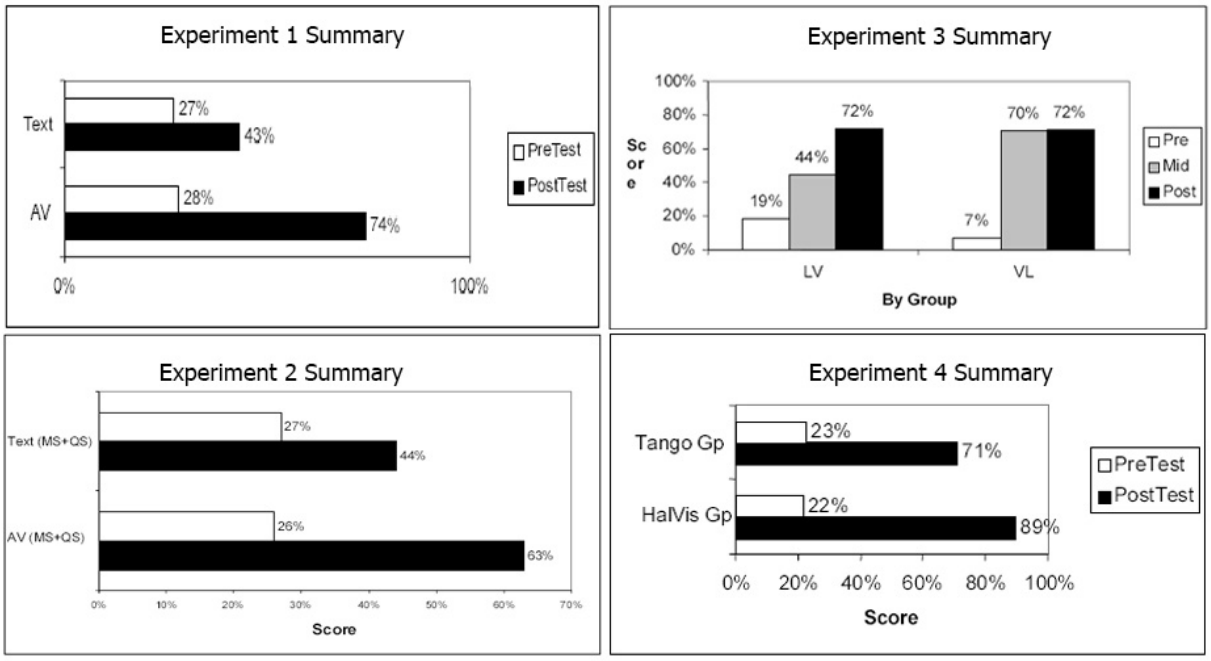

Figure 2: Results of HalVis system experiments

It's worth noting that students test scores were noticeably higher in the textbook/animation group than the textbook alone and lecture groups, suggesting that even lesser quality animations would still be better than no animations. Tests using others systems seemed to confirm their findings. Lawrence (segmented animations. 1994), Hansen (analogies. 2000), Saraiya (segmented animations. 2002), and Hundhausen (interactivity. 2002) all determined that how students use an AV has more effect than on the outcome than what they see.

3 Theory

Simple AVs help students understand the underlying concepts of algorithms at a high level. We determined that the following criteria is necessary for developing a successful algorithm system.

1. Effective algorithm visualizations must be dynamic and interactive. Static algorithm visualizations do not convey the same level of information as dynamic algorithm visualizations.

2. User feedback is a necessary component for helping students develop intuition from an algorithm visualization.

We expect that our algorithm visualization system will help users to develop a natural intuition for the following fundamental concepts:

1. DFS expands nodes recursively.

2. BFS expands nodes in order of their depth from the root.

4 Implementation

We set out to develop a system that provided interactivity, dynamic animations, and user feedback. Our approach was influenced by the five best practices proposed by Hansen Narayanan and Hegarty [Hansen et al., 2002]. These practices include testing of fundamental concepts, conceptual views, detailed views, populated views, and questions. Our system consists of a web service and front-end client. The client provides a rich user interface that contains visualizations, interactive feedback, and animations. The system was built around the AV entity. Each AV has a set of chapters. Each chapter has a set of exercises. An exercise is conditionally linked to a visualization. Whether the exercise contains a visualization or not is denoted by a BIT field. The ER diagram for the system is shown in fig. 3.

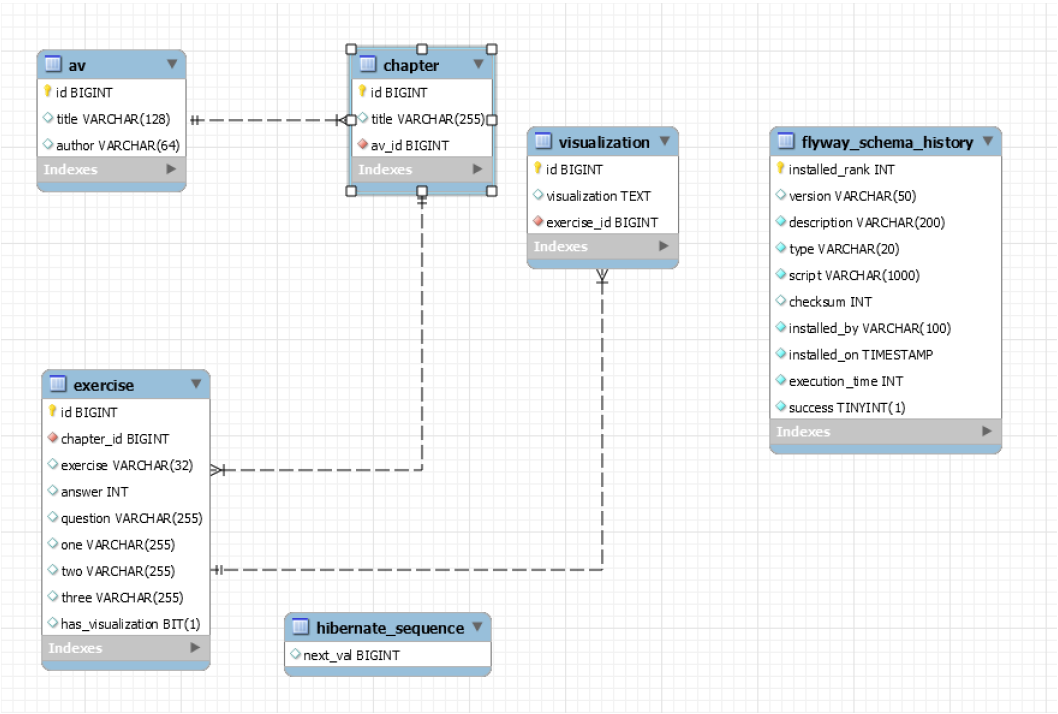

Figure 3: ER diagram

The database was designed in a hierarchical manner with the focus of the AV entity as the fundamental entity.

The web service was implemented with a popular java based web framework, Spring Boot. The front-end client utilizes Java Server Pages, HTML, JavaScript, and CSS. An example of how an AV object is served is shown in fig. 4

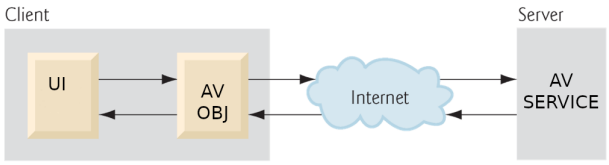

Figure 4: algorithm visualization System

The primary goal of our implementation was to create an interactive and user-friendly interface. Complicated visualization tools can quickly become too confusing for users to understand [Shaffer et al., 2010]. We mitigated this drawback by restricting user navigation to a hierarchical navigation path. The navigation path follows the child parent constraint relationship that was built into the underlying database.

In order for our application to facilitate rich algorithm visualizations we used d3, a popular JavaScript visualization library. An example of our DFS tree traversal animation can be seen in fig. 5.

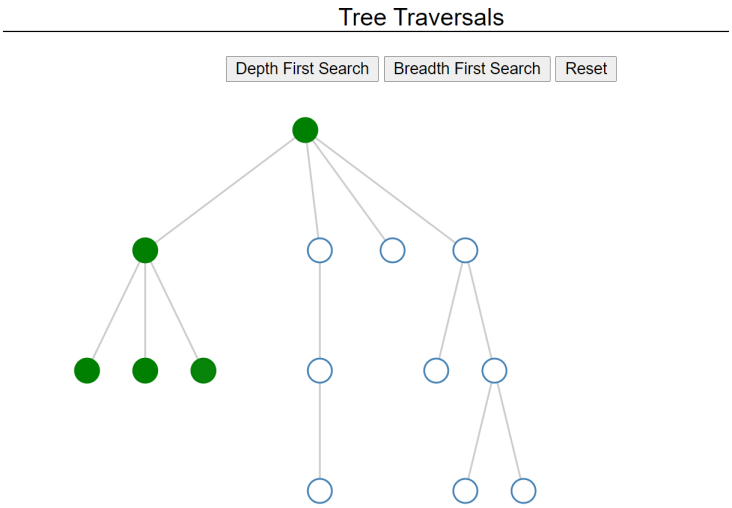

Figure 5: Tree Traversal Animation

User feedback is an important component for successful algorithm visualization systems [Shaffer et al., 2010]. Each question module consists of three multiple choice questions with a corresponding user prompt. The prompt contains either an image or pseudo code. A user has the ability to click on one of the answer choices. After they have clicked on a choice they receive a prompt that indicates whether their selection was correct or not. We felt that this feature was necessary for users to gauge their performance. Feedback is an essential part of learning. We expect that this feature helps

students self-correct or at least think more deeply about their answer choices. A static description cannot provide this sort of interaction.

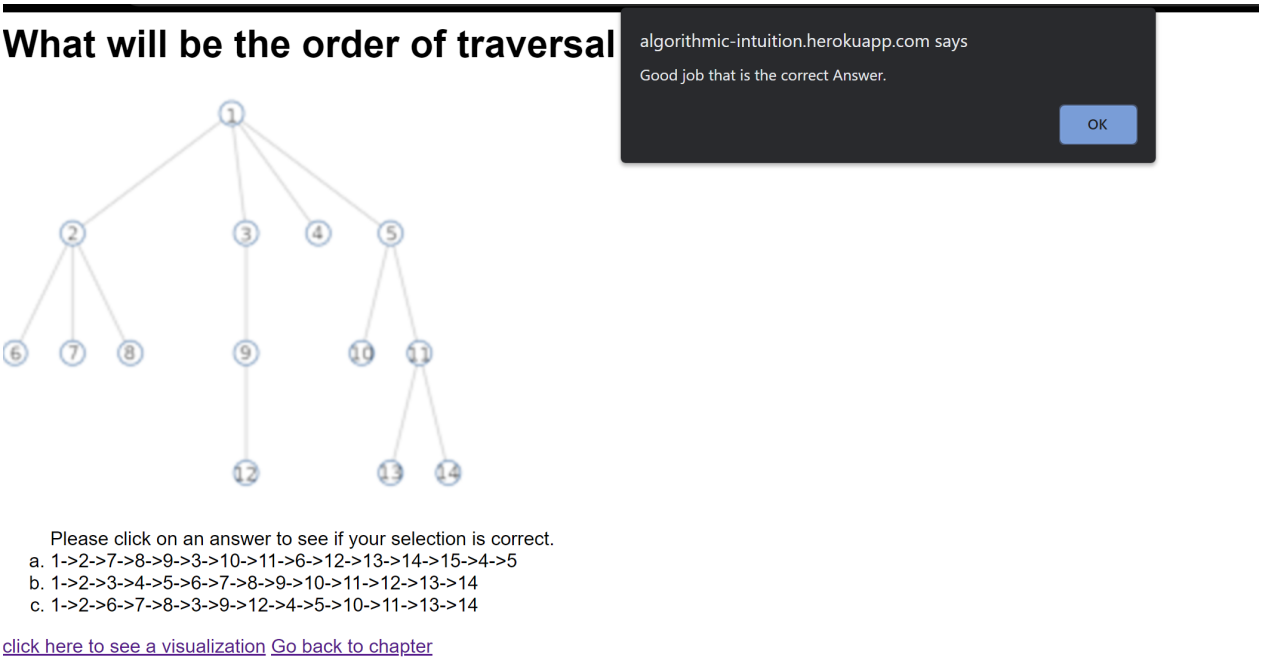

Figure 6: Question View

An additional module was introduced to help build upon the users understanding of fundamental concepts. This module, Graphs and Game Trees, shows the construction of a search tree based on a simple tic tac toe graph. We expected that the module would help solidify the users ability to identify the fundamental concepts of BFS and DFS.

The Graphs and Game Trees module leverages a third party javascript library for visualizing strategy games, mauler.js. The questions follow the same line of questioning as the tree traversal questions. The aim of this question was to help the user understand the fundamental differences between BFS and DFS. An example question is show in fig. 7.

5 Research Design

Our algorithm visualization system provided us with two separate modules that could be used to test our hypothesis. Each module's set of questions were designed to test the two fundamental concepts described in our theory. The tree traversal questions focus on the order in which the nodes are explored through DFS and BFS. The Graphs and Game Trees module is centered around how DFS and BFS construct the search tree. Graphs and Game Trees show the construction of search tree from from a simple tic tac toe graph. We expected that the user would develop an intuitive understanding

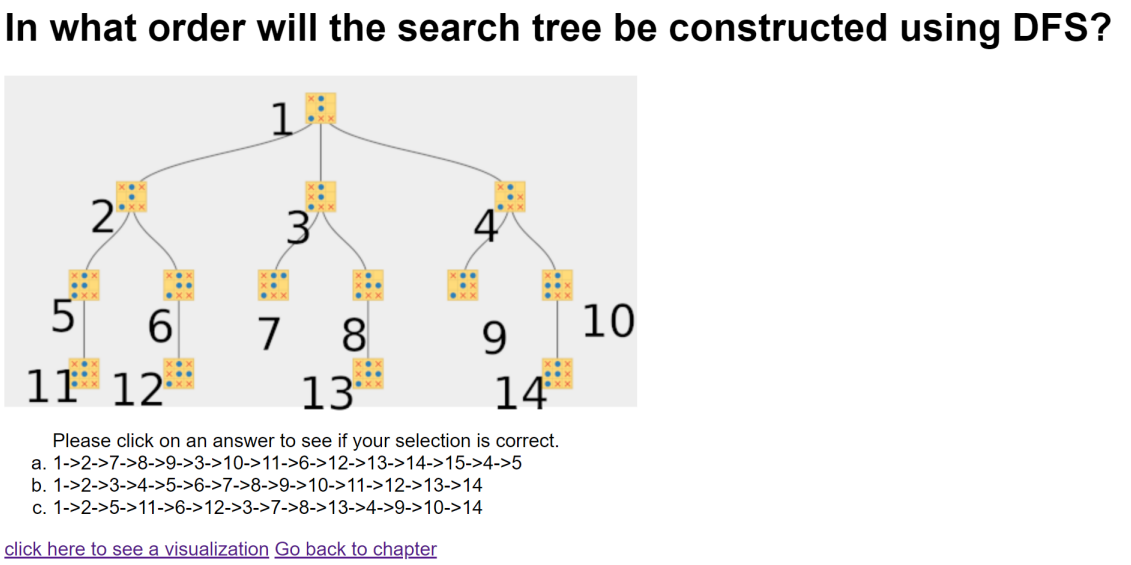

Figure 7: DFS Search Tree question

that DFS expands nodes in a recursive manner and BFS expands nodes in order from the root node.

The tree traversal module consisted of the following questions:

1. "What will be the order of traversal for DFS for the given tree?" with visualization.
2. "What will be the order of traversal for BFS for the given tree?" with visualization.
3. "Which of the following answers describes DFS traversal?" without visualization.
4. "Which of the following answers describes BFS traversal? " without visualization.

The search tree (Graphs and Game Trees) module consisted of the following questions:

1. "In what order will the search tree be constructed using DFS?"
2. "In what order will the search tree be constructed using BFS?"

6 Analysis

We tested two different types of users. One user who had very little experience with BFS/DFS algorithms and another user who had significant experience with BFS/DFS algorithms. Both sessions were recorded and uploaded to youtube.

We will refer to the users as inexperienced user and experienced user. The inexperienced struggled to answer the simple BFS/DFS tree traversal questions. After presenting the users with an algorithm visualization they were able to answer the questions more successfully. We noted increased confidence in their answers even if their explanations were incorrect. We made the following observations:

1. They remembered the traversal pattern from the visualization.

2. They did not review of the code in the pseudo code based questions.

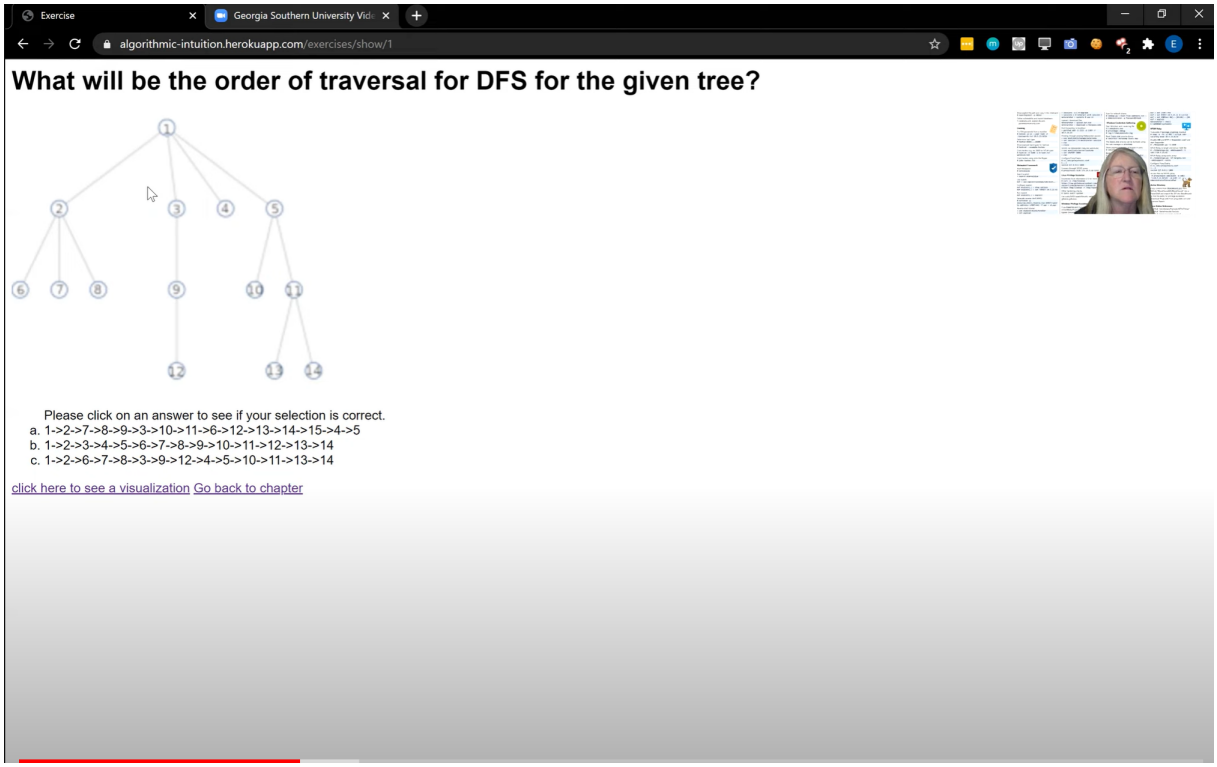

Figure 8: Experiment 1 Inexperienced User

The experienced user was able to answer the BFS/DFS questions successfully without the need of algorithm visualizations. However, we noted that:

1. The experienced user identified the first three nodes of the multiple choice questions and selected their answer because they remembered the pattern from the visualization.

2. The user did not need to scroll through the pseudo code to select the correct answer for the pseudo code based questions. They said they did not need to review the code because they remembered the visualization.

We observed that the inexperienced user was able to successfully answer questions based on BFS/DFS tree traversal. Although the user was not capable of providing an accurate explanation they were still able to intuit the algorithmic process. This indicates that the user developed a degree of intuition for how the algorithm worked after seeing it in visualization and receiving feedback on their answer choices.

We performed a third experiment with an experienced user. In this experiment the user was presented with a module that shows how a search tree is constructed with either DFS or BFS. The

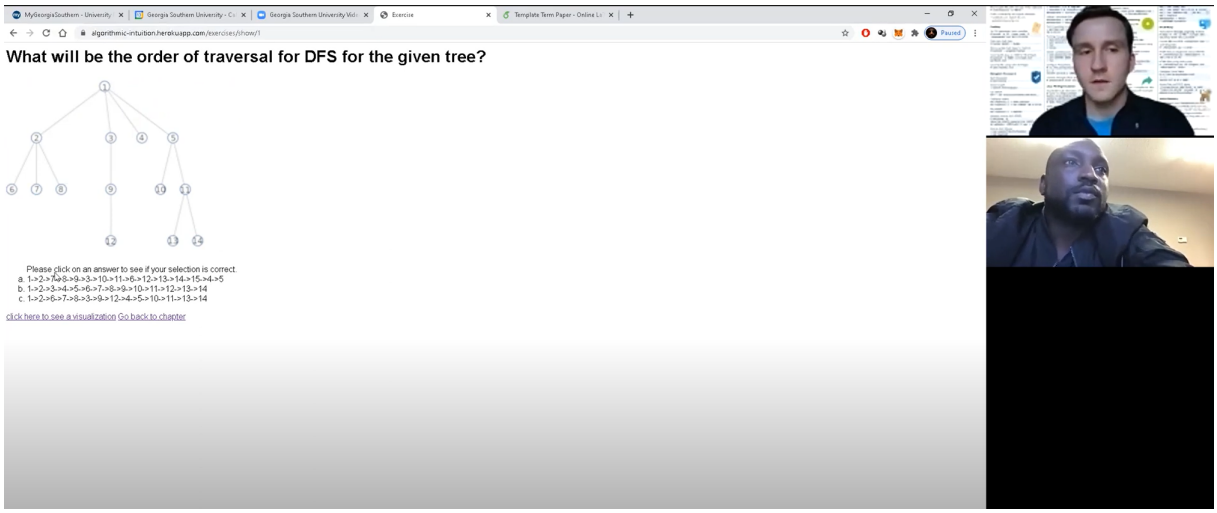

Figure 9: Experiment 2 Experienced User

purpose of this module was to build on the foundations presented in the Tree Traversals Module.

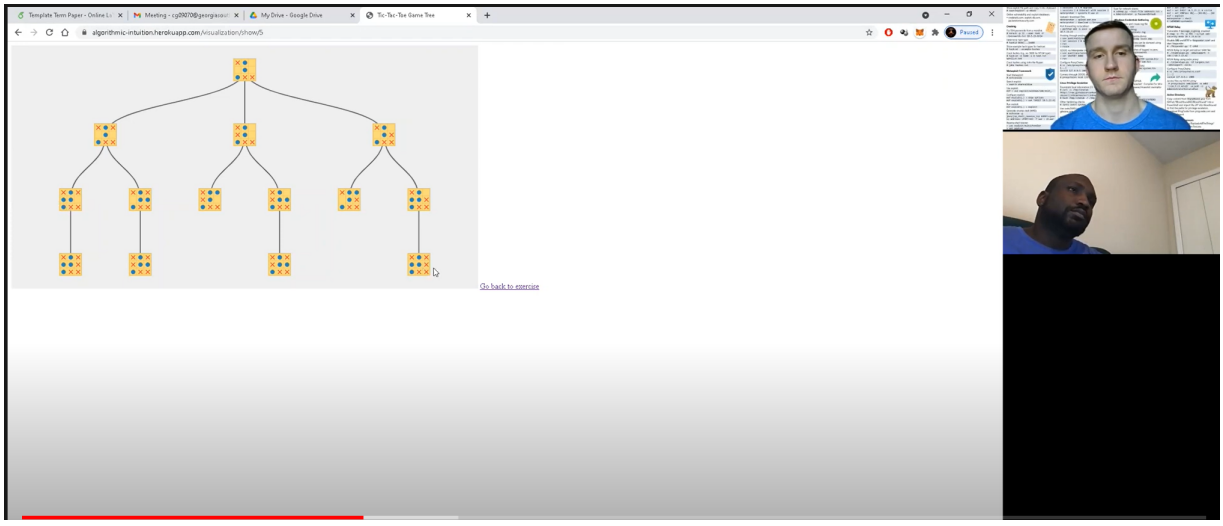

Figure 10: Experiment 3 Experienced User Graphs and Game Trees

After the user completed the Graphs and Game Trees module they were asked what prompted their answer choices. They explained that they used the intuition developed from the algorithm visualization to select the best answer. When we asked them how they would approach the problem if they did not have a visualization they said they would need to rethink how to approach the problem. These observations indicate that a degree of intuition was developed from presenting the user with an algorithm visualization.

The distinguishing characteristics of these experiments show that a simple picture is not enough to demonstrate the mechanics of an algorithm. Dynamic algorithm visualizations seem to be more effective at helping inexperienced users develop intuition. In the case of the experienced user, we saw that they tend to remember the visualization when answering pseudo code based questions.

This evidence supports our hypothesis. However, many factors still remain yet to be determined. Future experiments would include testing users knowledge after an extended duration of time. We must also take the testing size into consideration. Larger testing sizes could yield different results.

7 Conclusion

In this study we set out to get an understanding of why some studies on the effectiveness of algorithm visualizations on student learning yielded unfavorable results, despite being well received by students and teachers alike as favorable teaching tools. From this endeavor, three additional questions related to algorithm visualizations and student learning arose. Research has shown that how the visualizations are presented, as opposed to what visualizations are presented, is the determining factor for AV effectiveness (question 1). Although allowing students to construct their own AV causes them to spend more time with the content, it is not always beneficial as the creation process might be a distraction (question 2). Lastly, the most important factor in AV effectiveness as determined by studies is engagement (question 3).

With this information in mind, we were able to design our own AV to test whether or not our initial hypothesis stands; algorithmic intuition is achievable through algorithm visualization. Our entire project is entirely open source. It is available on github.

"The theoretical foundations for creating effective AVs appear to be steadily improving. More articles are appearing regarding effective use of AVs in courses, and a body of work has begun to form regarding how to develop effective AVs in the first place." (C. A. Shaffer et al.)

[Hansen et al., 2002] [Shaffer et al., 2010] [Robles, 2014]

References

S. Hansen, N. Hari. Narayanan, and M. Hegarty. Designing educationally effective algorithm visualizations. *Journal of Visual Languages and Computing*, 13(3):291–317, 2002.

David Robles. mauler: mauler javascript library. https://github.com/davidrobles/mauler, 2014.

C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, 10(3):22, 2010.

Statutory Declaration

I hereby declare that the paper presented is my own work and that I have not called upon the help of a third party. In addition, I affirm that neither I nor anybody else has submitted this paper or parts of it to obtain credits elsewhere before. I have clearly marked and acknowledged all quotations or references that have been taken from the works of others. All secondary literature and other sources are marked and listed in the bibliography.

_____

Signature

_____

Place, Date