A SOFTWARE VISUALIZATION-BASED APPROACH FOR UNDERSTANDING AND ANALYZING INCREMENTAL IMPLEMENTATIONS OF COMPLEX GRAPH-BASED ALGORITHMS

by

Jiaxin Sun

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer Graphic Technology

West Lafayette, Indiana

May 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. David Whittinghill, Chair

Department of Computer Graphic Technology

Dr. Tim McGraw

Department of Computer Graphic Technology

Dr. Esteban Garcia Bravo

Department of Computer Graphic Technology

Approved by:

Dr. Nicoletta Adamo-Villani

Head of the Graduate Program

This thesis is dedicated to my advisor Dr.David Whittinghill for his professional guidance and patient revision, Dr.Tim McGraw's great help on the revision of my thesis and Dr.Esteban Garcia Bravo's creative suggestions on the design of my system.

ACKNOWLEDGMENTS

First, I can't thank my advisor Dr.David Whittinghill more. He spent huge effort in helping me find an interesting and valuable research topic and shape it to be an appropriate thesis. I had a great experience and acquired a lot of innovative knowledge while doing the research because of his company along.

Besides, I sincerely appreciate Dr.Tim Mcgraw's patient revision of my thesis. The quality of my thesis is greatly increased with the help of his detailed revision. In addition, Dr.Esteban Garcia Bravo offered me very creative and innovative suggestions and insight on the design of my system. It's impossible for my system to have good usability and cool user interface.

Last but not least, my gratitude goes to students in Computer Graphics Technology who actively participated in my testing of the AV system. Their valuable responses contribute to the interesting conclusion of this thesis

Author

Jiaxin Sun

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AV Algorithm Visualization

ABSTRACT

Algorithm has always been a challenging topic for students to learn because of its high level of abstraction. To provide visual aid for algorithm education, many algorithm visualization systems have been designed, developed, and evaluated for the last two decades. However, neither the topics covered nor the interactivity of most AV systems are satisfying. This problem is presented in detail in chapter 2. As a result, this research aims to design, implement and evaluate a compiler-based algorithm visualization system on complex graph algorithm implementation with the assumption that it can help students build both confidence and competence in understanding it. This system is designed and developed according to the method in chapter 3. To test the hypothesis, a comparison experiment on 10 students in the Computer Graphics Technology department is conducted. The complete test protocol can be found in chapter 3.4, and the result can be found in chapter 4. Based on the limited number of subjects' testing data, a rough conclusion is made that this AV system has only a slight positive effect on subjects' confidence and competence in understanding complex graph algorithm's implementation, and its usability is acceptable. However, a concrete conclusion can only be reached if the testing is conducted to a larger group of subjects. In addition to the objective testing data, some interesting subjective observations, which are listed in chapter 5.2 are also made while doing the test. These observations indicate that algorithm visualization may more of a tool to examine users' understanding of the implementation than a tool to help them learn it.

CHAPTER 1. INTRODUCTION

Algorithm education has always been a tough task for novice(Vrachnos & Jimoyiannis, 2014), especially for advanced algorithm education. In order to solve this challenge, knowledge visualization technologies have been applied to this area, and systems called ”algorithm visualization system” have been experimented and developed over the years. However, these AV systems have failed to catch the mainstream of algorithm education for some reasons. (Hundhausen et al., 2002) This chapter will present this problem and introduce the purpose of this research, which is to develop a compiler-based AV system to visualize advanced graph-based algorithms in order to improve this failure.

1.1 Problem

AV systems are created to serve as a visual aid for computer science education. However, their pedagogy effect is far from satisfying. First, most AV systems are of low quality and only cover easy algorithms problems.(Shaffer et al., 2010) To master algorithm, students need a vast amount of practicing of various complex algorithm problems. Since these real-world algorithm problems are not included in AV systems, it’s natural neither educators nor learners will utilize AV systems. The second severe problem is that most AV systems are interface-based or script-based. (Urquiza-Fuentes & Velázquez-Iturbide, 2009) There are very few compiler-based systems that allow users to visualize their solution to the algorithm in real programming languages. For algorithm education, the final goal is that students are able to design and implement solutions to solve real algorithm problems, and a compiler-based system is very effective for helping students understand and analyze their solution. Without compiler-based visual aid, students usually have a hard time debugging and optimizing their solutions. A picture is shown below to illustrate how they debug solution without visual aid. The lack of compiler-based AV systems is a critical reason for the unsatisfying state of the AV system.

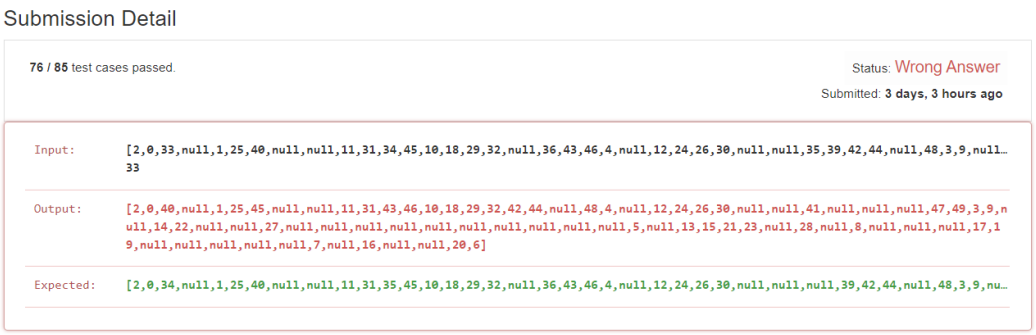


Figure 1.1. Representation of a tree in leetcode

1.2 Significance

In 2015 there are 216,228 students graduate from computer science, and the number keeps growing. Algorithm is a compulsory course for every student in computer science. Not only students in computer science but some students in majors such as electrical engineering, electrical computer engineering are also required to learn algorithms. Therefore an AV system that can help students understand and analyze algorithms will be beneficial to a vast amount of students.

1.3 The Purpose

The purpose is of this research is to design, develop and evaluate a compiler-based AV system which can visualize complex graph-based AV system and testing whether it can help students understand and analyze their implementations to complex algorithm problems or not.

1.4 Hypotheses

This research is conducted based on the hypothesis that the algorithm visualization has a significant positive effect on helping students build confidence and competence in understanding complex graph algorithm's implementation. Besides, it should also have good usability.

1.5 Assumptions

The subjects in the testing will be active and engaged while doing the testing.

1.6 Delimitations

The delimitations of this reserach are listed below:

- The potential users of this AV system will only be people who already have received some levels of algorithm education before.
- Instead of being able to visualize all graph algorithms' implementation, this algorithm visualization system will only visualize Tarjan-Bridge algorithm. The reason to choose this algorithm can be found in chapter 3.1.
- This system will only compile Javascript based solution.
- This system will only be developed in windows operating system.
- The testing of this system will only be conducted to a group of 10 subjects in Computer Graphics Technology department due to the fact that the number of students with algorithm education in this department is not large.

1.7 Limitations

The ideal subjects of this research will be students who have taken an algorithm course before and are not exposed to the Tarjan algorithm. In addition, they are supposed to have a similar level of algorithm education and learning ability. However, these two metrics are very hard to measure. Therefore, there's a possibility that subjects may actually differ a lot in algorithm ability and learning ability. This is a big limitation of this research that may affect the result of it.

1.8 Definitions

Algorithm visualization is a ”subclass of software visualization concerned with with illustrating computer algorithms in terms of their high-level operations, usually for the purpose of enhancing computer science students’ understanding of the algorithms’ procedural behavior.” (Hundhausen et al., 2002, p. 5)

Compiler-based algorithm visualization system is the one that allows user to compile and visualize the algorithm implementation in real time.Urquiza-Fuentes and Velázquez-Iturbide (2009)

Tarjan algorithm is a series of algorithms to find ”strong connected component” in a directed graph. Tarjan Bridge algorithm is one of them which aims to find the ”bridge” in a undirected graph using depth first search. (Tarjan, 1972)

1.9 Summary

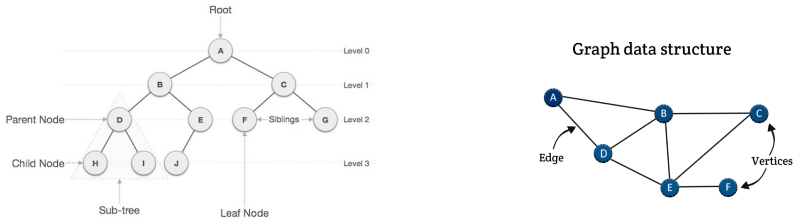
There is a gap between the need for a compiler-based AV system to visualize complex algorithm problems and the lack of such systems among existing AV systems. Therefore such a system will be experimented and developed in this research.

CHAPTER 2. REVEIW OF LITERATURE

Visualization technologies have been applied to many domains, and they have already been applied to algorithm education, and a new domain called algorithm visualization (AV) appeared two decades ago. Many AV systems have been developed and evaluated for general educational purposes. In this section, literature is reviewed to study this discipline. First, the concept of a graph-based algorithm, knowledge visualization, and algorithm visualization are introduced. Then the state and problem of existing AV systems are found.

2.1 Definition of graph-based algorithm and its educational methods

In computer science, graph is an abstract data structure implemented by a directed graph and undirected graph. Graph is defined as ”a type of data structure in which pairs of vertices are connected by edges”.(Anderson, 2012, p. 10) Edges can also be assigned value, and such a graph is called a weighted graph. Graphs can also be distinguished by whether it contains a cycle or not. A directed graph without cycles is also known as a tree. The most common trees are binary tree, binary search tree, AVL tree, and heaps.



(a) Binary tree (b) Undirected graph

Figure 2.1. Examples of tree and graph data structure

Graph-based algorithms are algorithms that are developed based on graph data structures. One of the most typical graph-based algorithm problem is graph traversal. One important algorithm to traverse the graph is the depth-first-search (DFS) algorithm. DFS algorithm traverses the graph following the rue that ”always chooses an edge emanating from the vertex most recently reached, which still has unexplored edges.” (Tarjan, 1972, p. 147) Another classic graph-based problem is pathfinding. Path-finding is known as ”an important problem for many applications, including network traffic, robot planning, military simulations, and computer games.” (Yap, 2002, p. 1) It refers to the process of finding a path–sometimes the shortest path– between a node and a target node in a graph data structure. A* algorithm is the best-known one to solve pathfinding problem which traverses the graph based on the heuristics distance to the target node.(Algfoor, Sunar, & Kolivand, 2015) It keeps examining whether the current node is the goal node while traversing. If it's the goal node, the algorithm is finished, and the shortest path is returned, and otherwise, it labels all surrounding nodes for further exploration. (Cui & Shi, 2011)

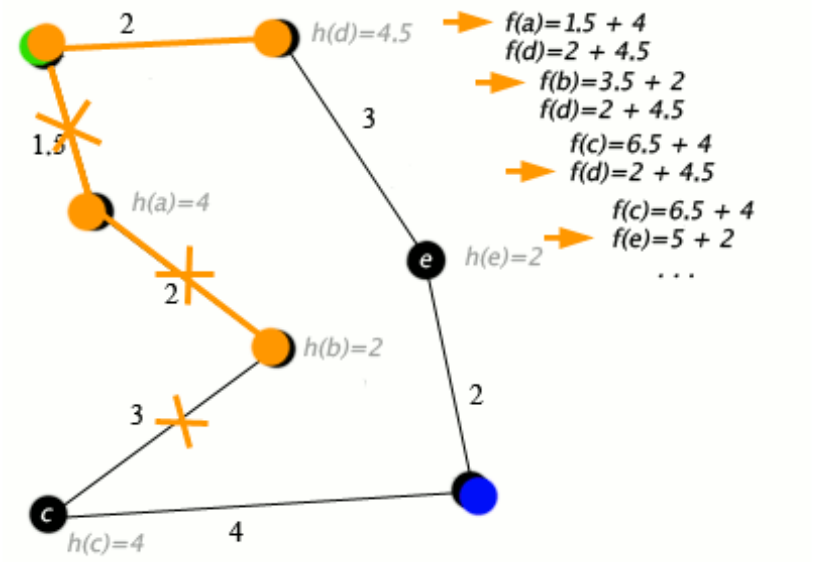


Figure 2.2. Example of A* algorithm

Graph is also considered as a non-linear data structure. Compared with linear data structures such as array, list, queue, and stack, graph and graph based algorithms need more visual aid in education because of their spatial complexity. In fact, both educators and learners tend to draw the graphical representation of trees and graphs when they teach and study this topic. Therefore a graph-based AV system can be helpful for graph-based algorithm education.

2.2 Introduction to knowledge visualization

Information visualization is defined as ”computer generated interactive graphical representations of information.” This discipline attracts a vast amount of researchers because information visualization has the magic to uncover the hidden important and informative patterns of information. (Ware, 2012) Information visualization techniques are widely applied to vast amount of areas. Herman, Melançon, and Marshall (2000) made a collection of areas that graph visualization (here graph refers to a visualization method instead of a data structure) that can be applied to. They are ”computer system file hierarchy, web site maps and browsing history, evolutionary trees, phylogenetic trees, molecular maps, genetic maps, biochemical pathways, and protein functions, object-oriented systems, data-structure, data flow diagrams, real-time systems, subroutine-call graphs, entity-relationship diagrams, semantic networks, and knowledge-representation diagrams, project management, logic programming, VLSI, virtual reality, and document management systems.”(Herman et al., 2000, p. 1) Rohrer and Swing (1997) presented several web-based types of information that have been visualized: Hierarchical information such as organization structure, computer file systems, interlinked Web hierarchies, and communication hierarchies, network information including computer network topology and network traffic monitoring, content-based document clustering, visual web search, and information space metaphors. Some of the examples are shown below.

One important type of information visualization is knowledge visualization. It's defined as "using visual representations to create and transfer knowledge between different individuals." (Eppler & Burkhard, 2004, p. 3) It's growing into an independent discipline because of its ability to help people to face the rapidly increasing knowledge.(Da & Jianping, 2009) Contero, Naya, Company, Saorín, and Conesa (2005) also identified the importance of visualization in engineering education as to improve the spatial abilities of students. Knowledge visualization proofed to be effective in education. For instance, Ifenthaler (2014) developed a web-based system to analyze natural language and represent graphical knowledge, and the tool proofed to be reliable and valid.

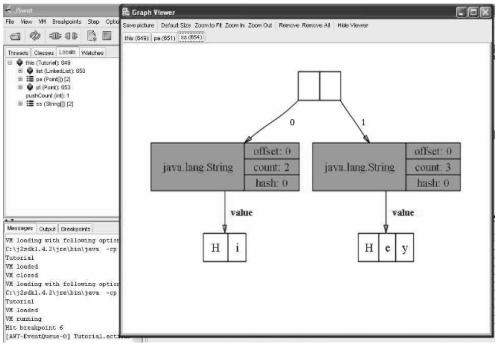
2.3 Existing graph-based algorithm visualization

Since knowledge visualization technique has successfully applied to many other educational disciplinary, this chapter will examine the possibility of usage of knowledge visualization technique on algorithm visualization.

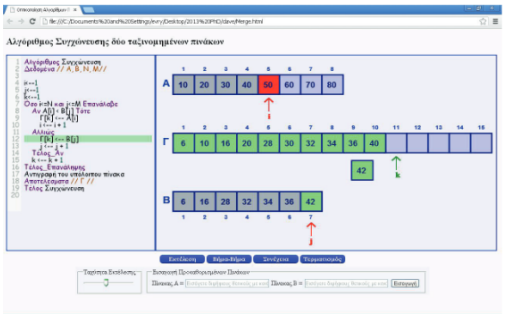
2.3.1 AV system in general

Knowledge visualization techniques have already been applied to visualize algorithms for years, and an interdisciplinary field between knowledge visualization and computer science education identified as "algorithm visualization" appeared for a very long time. Algorithm visualization is defined as "Algorithm visualization is a subclass of software visualization concerned with illustrating computer algorithms in terms of their high-level operations, usually for the purpose of enhancing computer science students' understanding of the algorithms' procedural behavior." (Hundhausen et al., 2002, p. 5) In fact, algorithm visualization can be treated as a kind of knowledge visualization, as mentioned in section 2.2 because most AV systems are developed in order to serve as an algorithm education aid.

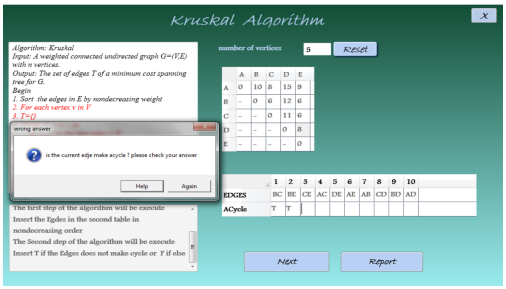
So far, at least hundreds of AV applications have been developed and evaluated.(Shaffer et al., 2010) Urquiza-Fuentes and Velázquez-Iturbide (2009) classified program and algorithm visualization systems into 3 classes based on the types of interactivity: script-based systems, interface based systems and compiler-based system. Since program visualization is very similar to algorithm visualization, this classification method can also be applied to algorithm visualization. Script-based AV systems allow users to interact with the system with script languages. Interface-based AV systems enable users to interact with the system with user interface. Compiler-based systems can compile users' code in real programming languages and visualize users' solutions. Some AV systems fit multiple classes. Examples of AV systems of each class are shown below.



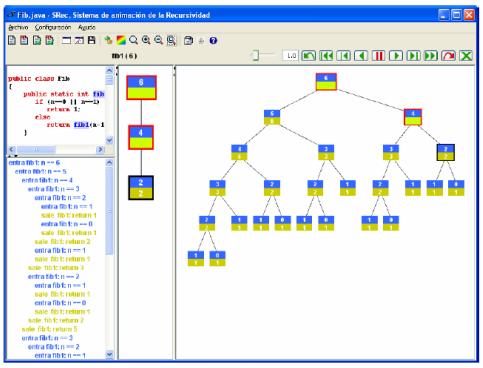
(a) LJV, a script-based data structure visualization system (Hamer, 2004)



(b) DAVE, a script+interface-based AV system (Vrachnos & Jimoyiannis, 2014)



(c) A interface-based AV system for debugging minimum spanning tree Khedr and Bahig (2017)



(d) SRec, a compiler-based AV system to visualize recursion (Velázquez-Iturbide et al., 2008)

Figure 2.3. Examples of AV systems in each class

Shaffer et al. (2010) grouped AV systems according to the algorithms visualized. They studied over 500 AV systems and counted how many AV systems on each algorithm topics.

Table 2.1. *Counts of AV systems on different data structures and algorithms*

(a) Counts of AV systems on different data structures (Shaffer et al., 2010)

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

(b) Counts of AV systems on different algorithms (Shaffer et al., 2010)

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

Besides the classification of AV systems, other research studied different aspects of AV systems. Karavirta and Shaffer (2013) developed a javascript library for algorithm visualization. Scott Grissom conducted an experiment to find the correlation between the level of student engagement on AV tool and learning effect.(Grissom, McNally, & Naps, 2003) The AV system they use is to visualize a simple sorting algorithm. The conclusion from this experiment is that learning increases as the level of student engagement increases. Hundhausen et al. (2002) conducted a meta-study on 24 experimental studies of AV and found that how students use AV system is much more important than the AV system itself.

2.3.2 Graph-based AV system

As shown in section 2.3.1, AV systems can be classified by algorithms and data structures visualized. Graph-based AV systems refer to AV systems focus on visualizing graph data structures and algorithms based on graph data-structures. Teresco et al. (2018) developed a map-based AV tool with data from the Travel Mapping project. This AV system aims to visualize algorithms related to graphs such as sequential search, graph traversals, and Dijkstra's algorithm. The most distinguishing aspect of this AV tool is that it used data from the real-world instead of small, synthetic graphs.

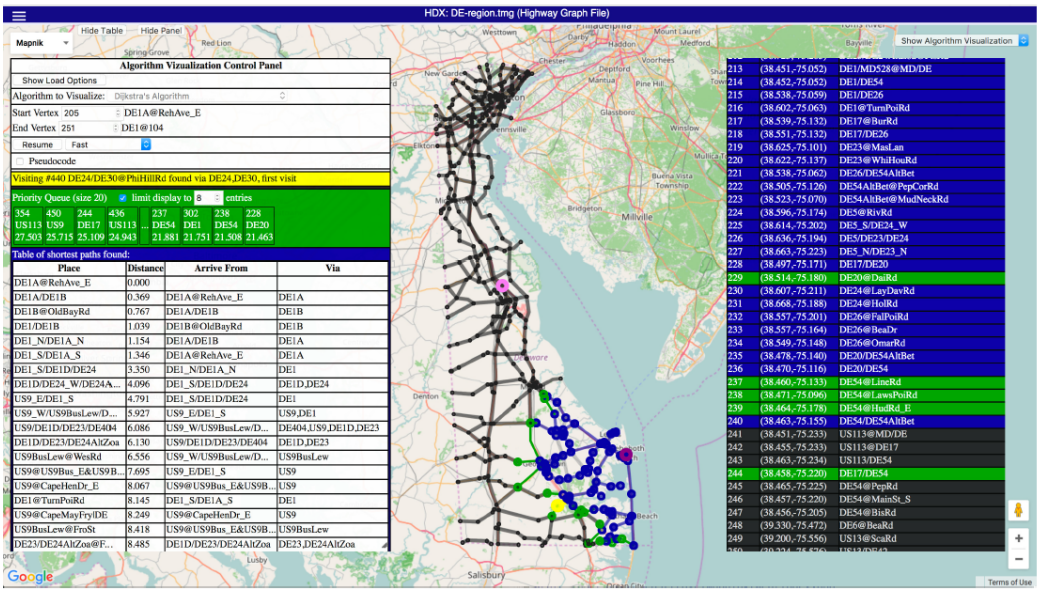


Figure 2.4. Map based AV system with real world data (Teresco et al., 2018)

Khedr and Bahig (2017) created a tool for debugging and learning minimal spanning tree as shown in figure 2.4.(c). The system has both a debugging mode and learning mode. It provides users with the explanation of Kruskal algorithm to find the minimum spanning tree from a graph and use a matrix to represent a graph. Users should choose the correct edge and vertices based on Kruskal algorithm, and the program will alert users if the edge and vertices are not correct.

Karavirta (2007) made an XML-based data structure system for educational purposes. It allows users to interacts with a graphical representation of data structures with XML languages. This AV system merely allows users to interact with tree data structures without related algorithms.

```
<delete target="BST">
  <key value="C"/>
  <elementary>
    <remove target="nodeC"/>
    <remove target="edgeCA"/>
    <replace target="edgeMC">
      <edge from="nodeM" to="nodeA"/>
    </replace>
  </elementary>
</delete>
```

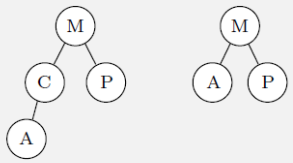


Figure 2.5. XML-based AV system (Karavirta, 2007)

2.4 Problems of existing graph-based AV systems

2.4.1 Lean towards easy topic

Although AV has been studied for over two decades and hundreds of AV systems have been developed (Shaffer et al., 2010), there are still some severe problems. ”Despite the intuitive appeal of the AV techniques, it has failed to catch on in mainstream computer science education.” (Hundhausen et al., 2002) One serious problem is that most AV systems tend to cover only easy topics. (Shaffer et al., 2010), presented the current state of research on AV by studying over 500 AV systems. The research studied their availability, implementation, range of algorithms covered, dissemination, author, time, existence, and license. The most important conclusion is most AV systems' quality is low and the algorithm topics covered are easy topics. Systems that can visualize easy algorithms are valuable for the novice, but for people with basic algorithm knowledge and need advanced algorithm education, these systems are not helpful. Especially in real algorithm interviews, the algorithms questions tend to be much more difficult and complex than basic algorithm problems covered in existing AV systems, as shown in figure 2.8. For advanced algorithm problems– especially for graph-based problems – visual aid is very beneficial for people to understand how algorithm works behind the scene. Unfortunately, so far, the most popular visual aid for advanced algorithm problems are still traditional visualization, such as blackboard and paper-based visualization, because they are not covered in current visualization systems. In fact, most research on AV doesn't even include the scenario of advanced algorithm education and preparing for technical interviews in scenarios of usage of AV systems, as shown in figure 2.9. Therefore a gap is shown between the need for systems that can visualize advanced algorithm problems for experienced users and the existing AV systems that only cover easy algorithm problems.

124. Binary Tree Maximum Path Sum

Hard 2135 166 Favorite Share

Given a **non-empty** binary tree, find the maximum path sum.

For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain **at least one node** and does not need to go through the root.

Example 1:

```
Input: [1,2,3]

       1
      / \
     2   3

Output: 6
```

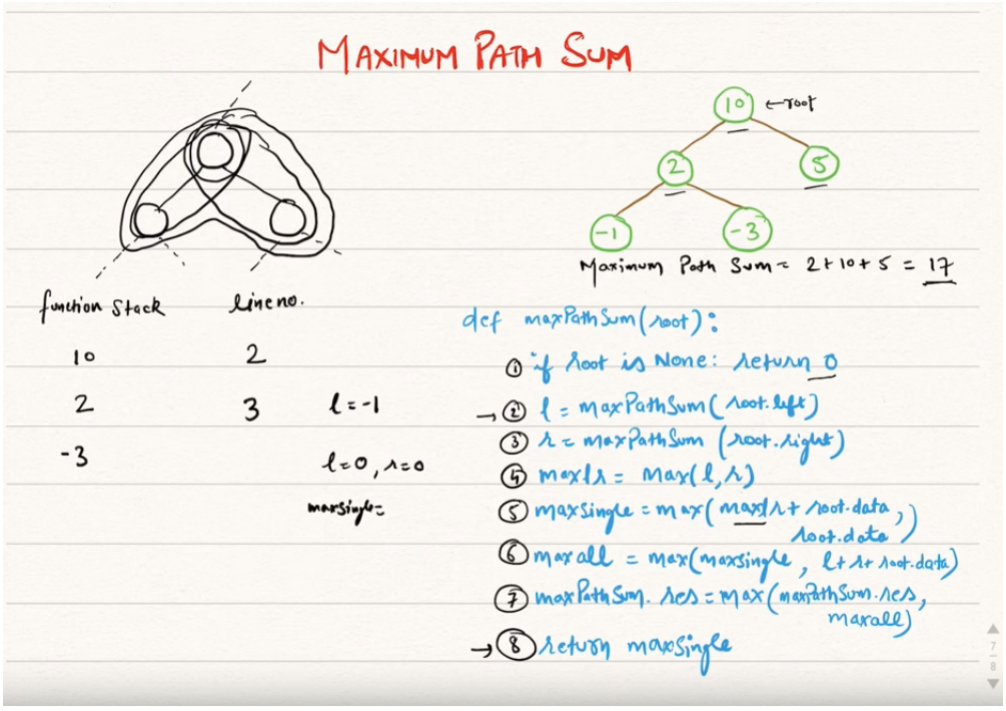
Example 2:

```
Input: [-10,9,20,null,null,15,7]

   -10
   / \
  9  20
    /  \
   15   7

Output: 42
```

(a) Binary Tree Maximum Path Sum problem in leetcode



(b) Visual Explanation of solution to Binary Tree Maximum Path Sum problem

Figure 2.6. A example of algorithm question in real technical interview

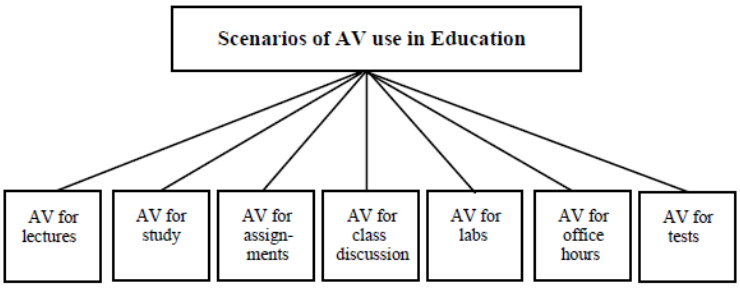


Figure 2.7. Incomplete scenarios of AV usage in education (Hundhausen et al., 2002)

2.4.2 Lacking compiler-based AV system

The second problem is the lack of compiler-based systems. As shown in section 2.3.1, AV systems can be classified as script-based, interface-based, and compiler-based AV systems (Urquiza-Fuentes & Velázquez-Iturbide, 2009). For advanced algorithm education compiler-based AV systems are more beneficial than the other two types of AV systems since compiler-based systems are better for the ”changing, construction and presenting engagement levels of algorithm education”.(Urquiza-Fuentes & Velázquez-Iturbide, 2009, p. 20) However, there are very few compiler-based AV systems with good quality. Urquiza-Fuentes and Velázquez-Iturbide (2009) presented a few compiler-based AV-systems : SRec (Velázquez-Iturbide et al., 2008), MAVIS (Koifman, Shimshoni, & Tal, 2008). SRec is designed to visualize recursive functions written in java, as shown in figure 2.4(d). The recursion is visualized with incremental animation, which allows you to play the animation back and forward. Users need to first mark the recursion function to be visualized in a java file and then run the visualization system. However, the test case that runs in this AV system is only a function to calculate the 6th Fibonacci number. With such an easy test case, the system can not prove to be capable of visualizing complex recursive algorithms. In fact, recursive dynamic programming algorithm problems are considered to be both popular and difficult among all algorithm problems. Therefore such an AV system cannot help too much in recursive algorithm education.

2.5 Summary

After a systematic study of existing AV systems, two major problems are found: algorithm topics covered are too easy and lacking compiler-based AV systems. At least in advanced algorithm education, AV systems are not taken into account because of these two problems. Therefore the need for developing compiler-based AV systems for advanced algorithms visualizations emerges.

CHAPTER 3. RESEARCH METHODOLOGY

In order to study a software visualization-based approach for understanding and to analyze incremental implementations of complex graph-based data structures and algorithms, a system to visualize complex graph-based algorithm problems will be developed, and its design, development, and evaluation method will be presented in this chapter. First of all, all the functions and features will be carefully designed based on user experience principles and processes. A prototype will be designed based on the user journey map and will be examined by small-scale preliminary testing. After that, the detailed program architecture will be developed, and libraries and API utilized will be shown and explained.
Finally, a questionnaire-based evaluation will be conducted to examine the effect of this AV system.

3.1 Algorithm Chosen to Visualize

As explained in the delimitation section, only Tarjan-Bridge algorithm's implementation is chosen to visualize because it's impossible to visualize all graph algorithms. It is an important algorithm in graph theory. As explained in the definition, this algorithm finds a "bridge" of an undirected graph. Bridge is a critical connection of a graph, without which the graph will become separated from connected. Tarjan-Bridge algorithm utilizes depth first search to find bridges in linear time.

#	Title	Solution	Acceptance	Difficulty	Frequency
1	Two Sum		45.2%	Easy	
200	Number of Islands		45.1%	Medium	
146	LRU Cache		30.4%	Medium	
1192	Critical Connections in a Network		48.9%	Hard	
2	Add Two Numbers		33.0%	Medium	
42	Trapping Rain Water		47.2%	Hard	

Figure 3.1. The frequency of Critical Connection problem (need Tarjan-Bridge algorithm to solve) in leetcode among 1300 problems

There are 3 major reasons why this algorithm is chosen in this research. First of all, although there are many AV systems that are designed to visualize the graph algorithm, this algorithm has not been visualized in any AV system yet. Second, this is a relatively complex algorithm, and the implementation of it is very tricky. In the coding platform such as leetcode, people make lots of posts to ask the details of the implementation. Besides, this algorithm is a very popular interview question. Software engineer applicants of big technology companies, such as Amazon, have been asked of this question many times. Therefore, as shown in figure 3.1, its frequency shown in the interview is in the top 5 of all 1300 questions. Based on all the important facts above, this algorithm is very suitable for this research, and a visualization system for this algorithm is in need.

3.2 Design of the graph-based visualization system

The design process of this system consists of 4 parts, user journey map, prototype design, preliminary testing, and system design. They will be described in detail in this chapter.

3.2.1 User journey map

The user journey map is conducted in order to find the pain point of the process of learning algorithm implementation, and the prototype will be designed to solve this pain point. Journey maps and the personas are shown below. The journey clearly showed that for algorithm learners, the most time consuming and frustrating process is to debug algorithm errors since the only visual aid they have is paper. On the other hand, for users with visual aid from AV systems, the usability of the debugging algorithm is improved significantly. Furthermore, with the visual aid of AV system, users are more likely to come up with an optimized solution. Therefore, a compiler-based AV system is necessary, and the prototype should be able to compile users' solutions and visualize solutions.

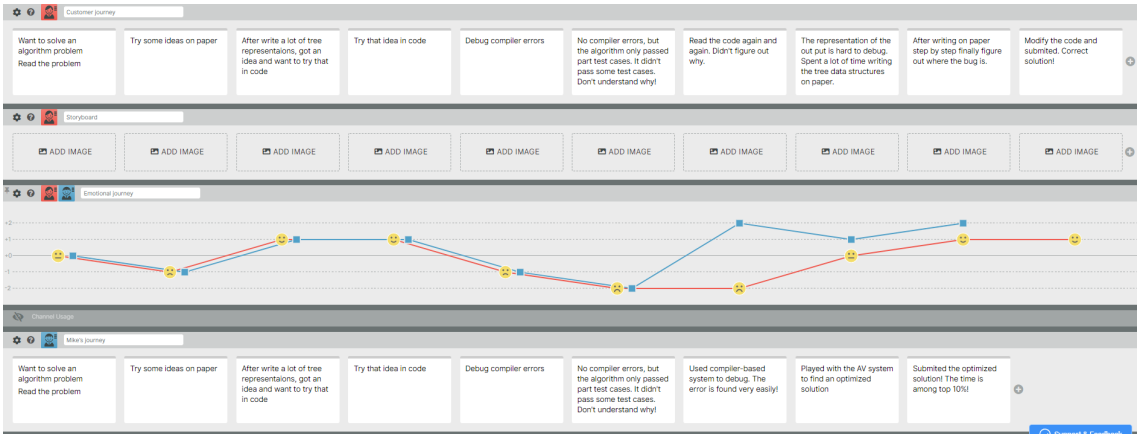
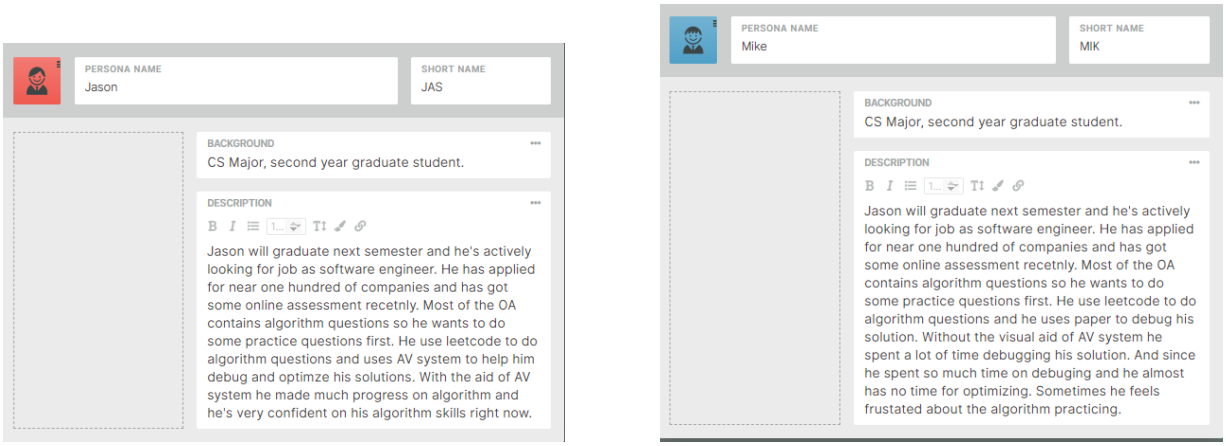


Figure 3.2. User Journey Map of algorithm learning process

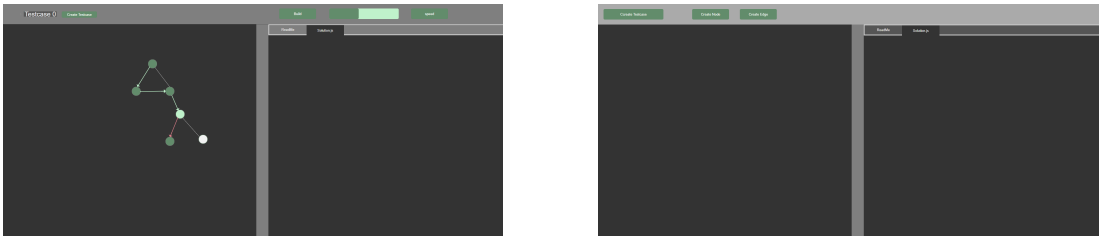


(a) (b)

Figure 3.3. Personas

3.2.2 Prototype Design

A prototype of the compiler-based AV system on Tarjan algorithms is designed based-on the pain points found in the user journey with the help of Adobe Axure. This prototype contains 2 sub-pages: main page and create page. The main page consists of 3 components: header component, visualization component, and code editor. The header component allows users to choose or create new test cases and build and play visualization animation. The visualization animation will be played in the visualization component. Users can modify the code in the code in the code editor and compile it in real-time. When the user clicks the ”Create testcase button” the application will be redirected to create page as shown in figure 3.4. Here the user is asked to create a connected undirected graph. In ”Create Node” mode the user can click on the component to create a node, and in ”Create Edge” mode user can click two nodes to create an edge. After a valid test case is created user can click create ”Create testcase” button to go back to main page, and the system will add the test case the user just created into the testcase list and allow the user to build and run the visualization on it.



(a) Prototype's main page (b) Prototype's create page

Figure 3.4. Prototype design

3.2.3 Preliminary Testing and refinement

A small-scale preliminary testing will be conducted to examine the usability of this prototype design. 3 people from the Computer Graphic Department with basic algorithm education are chosen to participate in the testing. The testing is designed as follows: each participant is provided with the prototype exported from Axure and is taught how to use this system. After that, they are asked to do a learning task with this system. The learning task asks them to create a test case, build and visualize it. In the whole process, they are asked to think aloud, and their responses and activities is recorded in order to refine the prototype.

During the preliminary testing, there are two things that are commonly mentioned by the participates. First, the shapes and colors of nodes and edges are designed to express the different statuses of them, but the participants don't understand. In order to fix that, a legend bar is needed in the finalized design. Second, as the incremental visualization plays, the changing of the state of nodes or edges is not clearly expressed. To make it clear, a pop-up component with an explanation of changing state will be associated with each change. Besides, some minor changes in the user interface are also mentioned. For example, some of them think that in the create page the code editor is not needed. All of these responses will be taken into consideration in the final design so that the system will have better usability.

3.2.4 System Design

To make sure the system is developed correctly and efficiently, the system is designed carefully before the development. The AV system includes a front-end client-side application and a back-end server. The front-end contains two routes: visualization route and create a route. The overall system design is illustrated in figure 3.5.

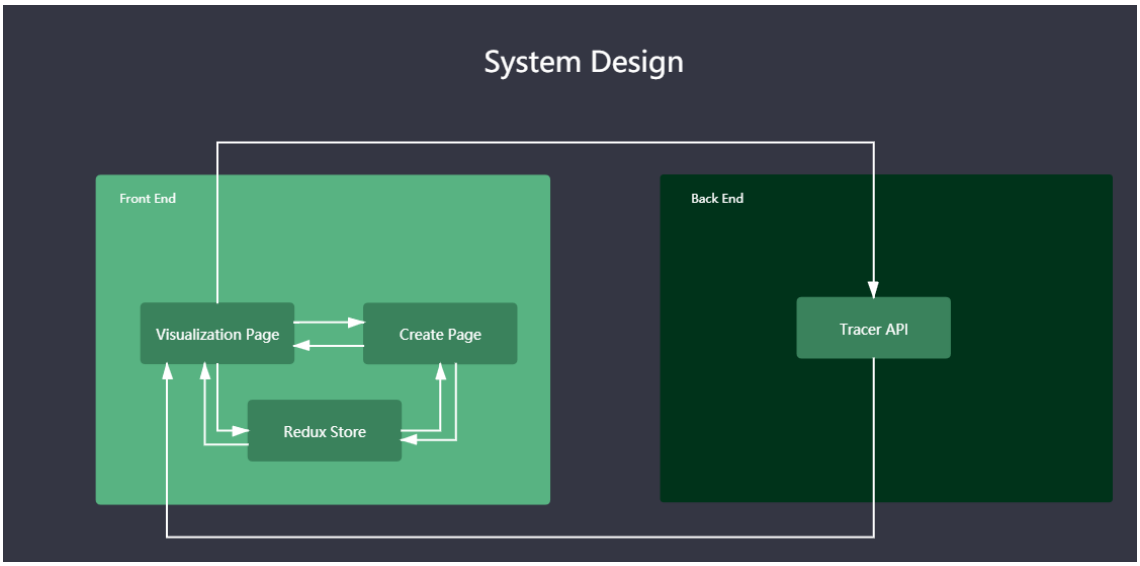


Figure 3.5. System Design

The front-end is developed as a typical react-redux application. Each rendering component extends react's base component. A redux store is attached to the front-end application to store the central data. The system design of the visualization route is shown in figure 3.6. This is the most important and complicated part of this system. The key functionality is building the code in the code editor and play visualization. The build function will send the code to the server as a string. The server will run the code with the help of the inherent javascript function and generate a command array that stores all the visualization commands. The commands will be sent to the front-end and then be stored in the redux store. When the user plays the visualization, the command corresponding to the current playing step will be applied to the visualization component, and the graph render of it will render the visualization accordingly.

The create page is relatively simple. When users create a graph using the user interface, a matrix representation of the graph is stored in the local state variable. When the creation is done and confirmed, the matrix will be converted to a string and then sent to the redux store. The design is shown in figure 3.7. The back-end server is built so that the rendering and logic of the system are separated. The server contains the key API of this system. It contains the graph tracer API to run the code received from the front and generate the commands array to send back to the front-end.

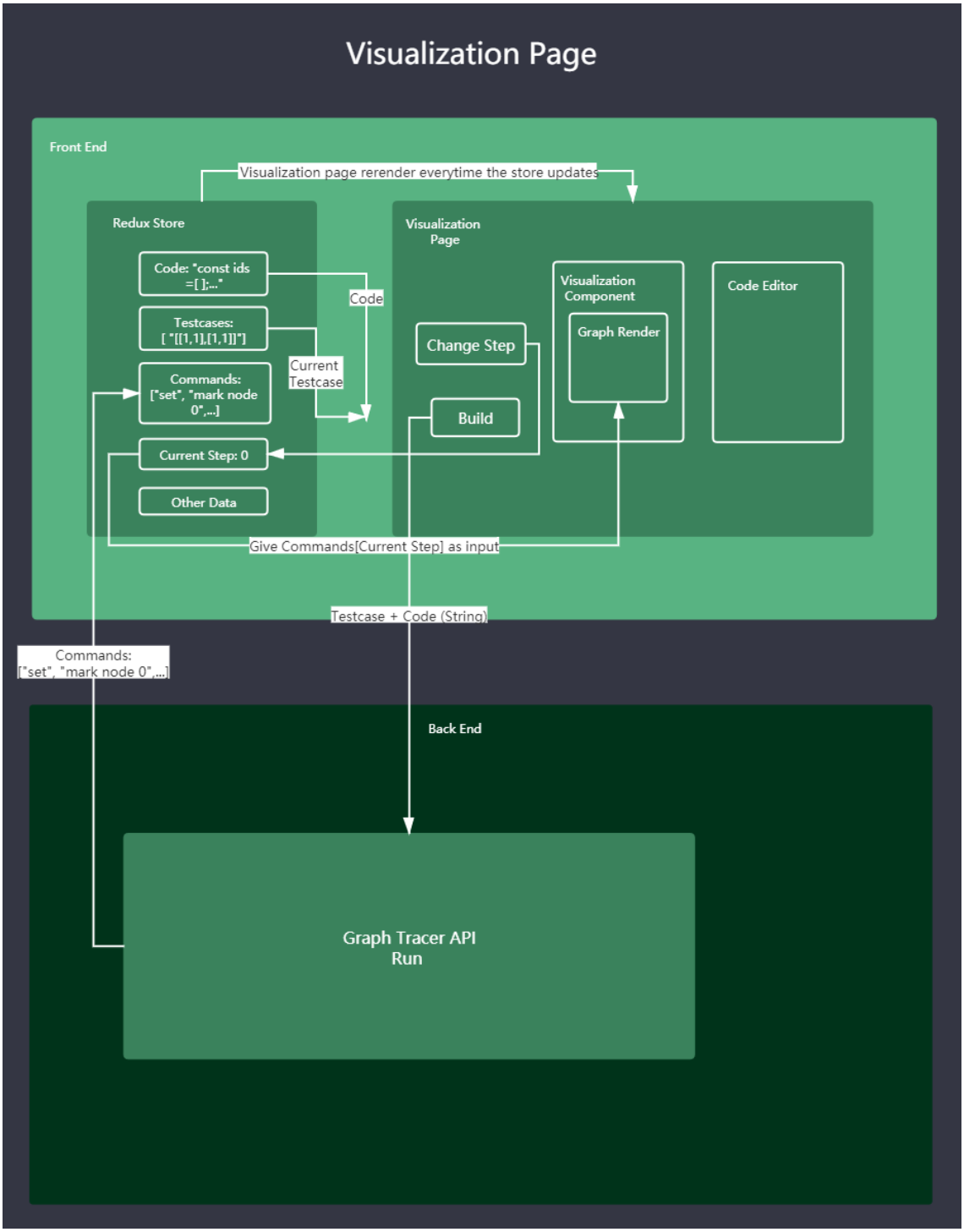


Figure 3.6. Visualization Route

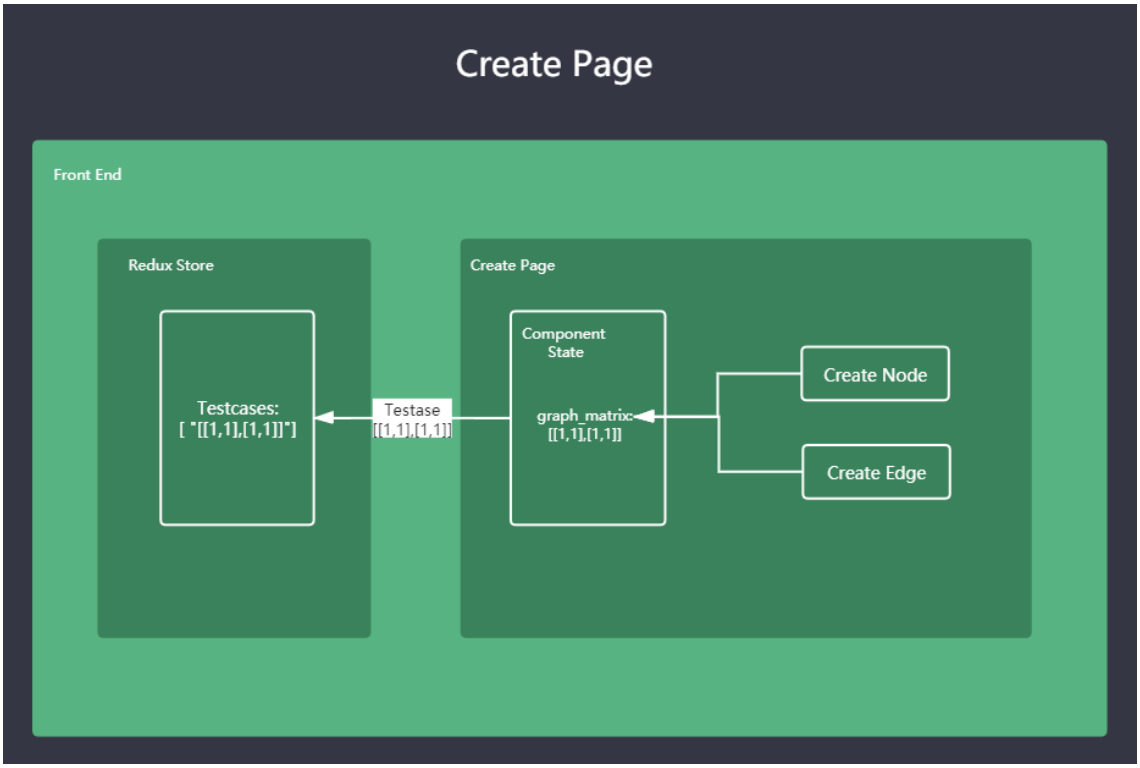


Figure 3.7. Create Route

3.3 Development of the graph-based visualization system

3.3.1 Technologies utilized

This AV system includes a front-end application and a back-end server. To develop the front end application, the framework React and Redux is utilized because they are among the most popular and powerful front-end framework. Some plugins of react, such as React-Ace that is used in the code editor component, react-router, which is used to create a router, are also utilized. React-Semantic-UI is also used in some UI elements such as pop up component. The back-end server is built with node.js and express, which are the most common server frameworks. Besides, the server-side API is written in typescript and then compiled to javascript because typescript provides type checking and can avoid errors.

3.3.2 System Components

As explained in the system design section, the developed system has two pages: visualization page and the create page. Both pages follow the header-content style and similar theme.

Figure 3.8 shows the visualization page of the system. The header contains three parts: test case control, visualization mode control, and the visualization player. The system is designed to encourage the user to create test cases of the algorithm by themselves, and the test case control part allows the user to create new test cases and choose the currently built test case. When users click the ”create new test case” button the application is redirected to the create page to build a test case, which will be described later. On the right of test case control part, user can choose what information to show on the visualization panel. For example, the user can turn on or off the pop up showing all variables value of each node by clicking the ”show full info” button. The player contains all the user interface to play the visualization. Users can choose to let it play automatically or go through each step manually.

Below the header is the content of the system. It consists of three panels: visualization legend, visualization, and code editor. The visualization is designed in a way that the states of each node and edge are differentiated by their color and shape. Users can always refer to detailed state information. The implementation of the Tarjan-Bridge algorithm in Javascript is provided in the code editor. Besides code for Tarjan-Bridge algorithm, the visualization code is also ’hided’ in the code. This visualization code is used for tracer API to generate commands. For example, at the beginning of critical connection function, the code of marking current node is hidden inside ”//visualize” so that every time a new recursive call is activated, the currently visiting node will appear flash and yellow. This code is carefully pre-designed. Since this system can compile and generate visualization commands in real-time, it enables the user to modify the code and visualize the modified code as long as there is no compile or run time errors. Given time, it’s even possible for the user to write their own visualization code.

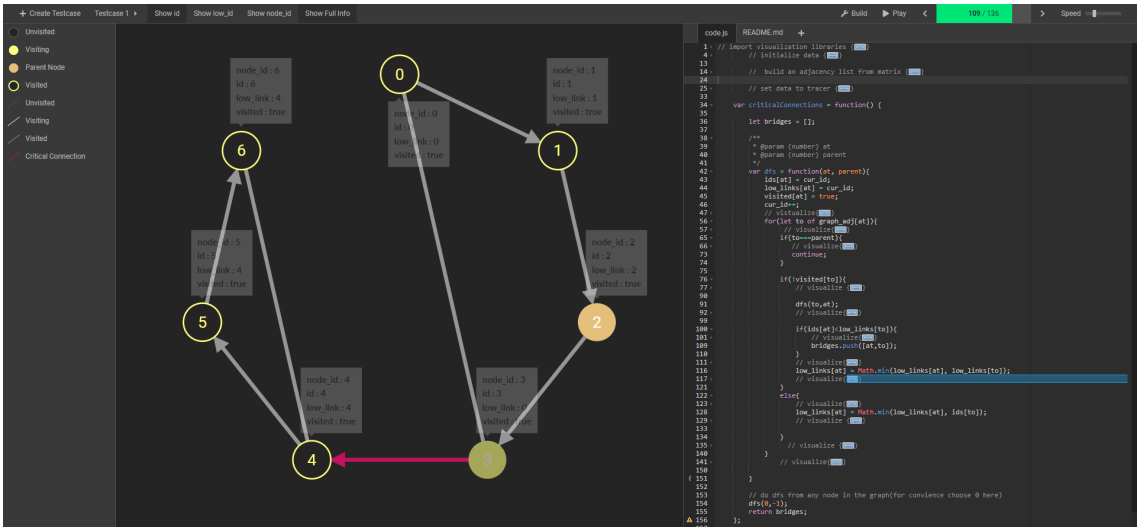


Figure 3.8. Visualization Page

This visualization is carefully designed to follow the logic of Tranjan-Bridge algorithm in order to make the incremental visualization understandable. The algorithm finds the bridges of a connected undirect graph by doing a depth first search on the graph and keep updating the variables of each node. When he updated variable meets certain conditions, it yells that a bridge is found. Therefore, the visualization is designed based on this. First of all, each node is accompanied by a pop-up panel to show the values of its variable and how are the updated as the depth first search goes on. In addition, each step a pop up will appears along with the node or edge that is being executed. As shown in the figure, when the algorithm finds a bridge, a pop up will appear near this bridge, telling you why this is a bridge. These pop-up messages contain all the information for the user to understand the algorithm. Last but not least, the state of nodes and edges are shown by their color and shapes. For example, visiting nodes and visited ones are of the same color but different fill mode so that the logic behind it can be easily seen.

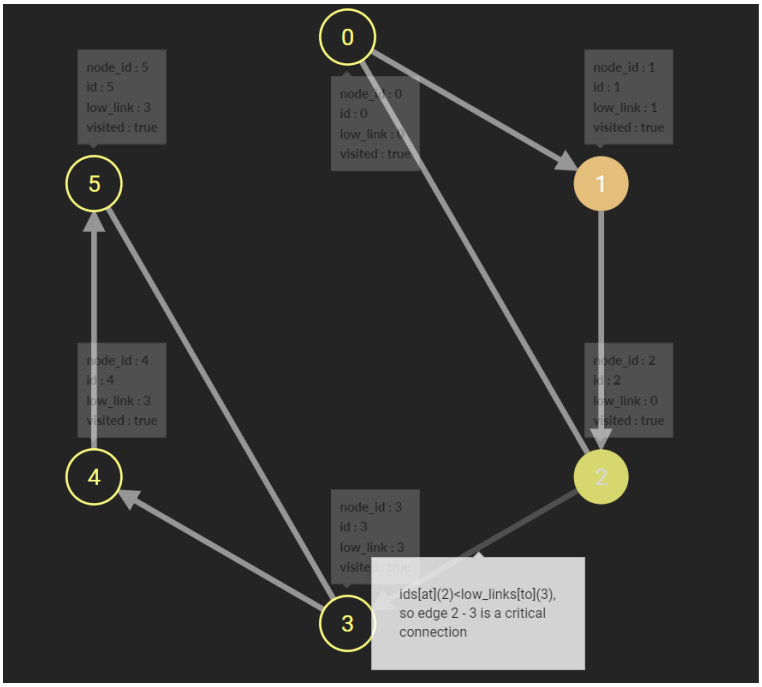


Figure 3.9. Message explaining why the algorithm claim this edge as a bridge

Create page is shown in figure 3.10. User can create a node by clicking on the canvas and create an edge by clicking on two nodes. Some supporting functions are provided, such as undo the changes and changing the nodes and edges size. When the creation is done, users can click ”create testcase” button to generate a new one and redirect it to the visualization page. Since the input of this algorithm is restricted to be a connected undirected graph, this button will be disabled when the graph is not valid.

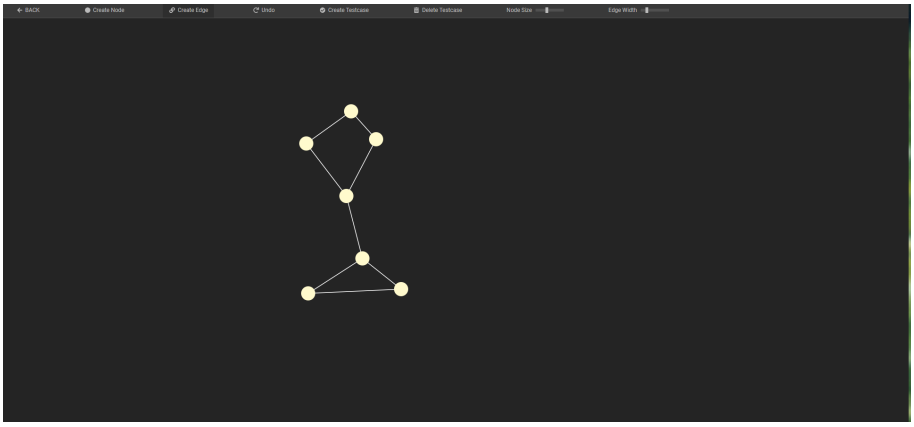


Figure 3.10. Create Page

3.4 Testing

In order to prove the hypothesis that the AV system will help students understand the implementation of complex graph algorithms better, testing is conducted to measure the effect of this AV system on the subject's confidence, competence, and usability. This section will give a detailed explanation of how the test is conducted.

3.4.1 Population and Sample

The population of this research is all the students that have received some basic algorithm education and in the process of learning complex graph algorithms. Due to the limitation explained before, only 10 students who have learned algorithms before in the CGT department have been selected as a sample.

3.4.2 Variables

The effectiveness of this system is explained in three variables: the subject's confidence, competence, and the system's usability. The confidence explains how much the subjects are confident about their understanding of the implementation of this algorithm. It's a subjective metrics. On the other hand, the competence is a totally objective metrics. Subject's competence of understanding of the implementation is measured by objective questions of the implementation. The usability means how user-friendly the system is. The testing method of these variables will be explained in detail in the next section.

3.4.3 Method

In order to measure the three variables listed above, the testing is carefully designed, and the testing method is explained in this section. 10 subjects are randomly divided into two equal groups. One group will be asked to finish some learning tasks in regard to the algorithm implementation after watch the algorithm tutorial video only in 30 minutes, while the other group needs to finish the same tasks after watching the same video and being taught how to use the av system in the same time. For simplicity, these two groups are referred to as group A and group B. The complete test protocol is list below.

a. Establish remote communication with subjects via Zoom meeting.

b. The purpose of this research and the steps of the testing will be introduced to subjects.

c. Subjects in both groups are asked to watch a 12 minutes video tutorial on Tarjan-Bridge Algorithm. This video consists of 3 parts: what problem does this algorithm solve, the theory of this algorithm and the implementation of this algorithm. As soon as subjects start watching the timer of 30 minutes will begin.

d. After finished the video, group B will be taught how to use the AV system. Following that, they will be asked to finish two simple learning tasks. Task one requires them to create a test case that has bridges and run the visualization. Task two is almost the same except for the test case is changed to one without bridges. This process takes about 3 to 5 minutes. Group A is not involved in this step.

e. Subjects in both groups are asked to answer the pre-designed questionnaire in Google Docs. The complete questionnaire can be found in the appendix. While answering questions in it, group A can refer to the video they just watched, and group B can both refer to the video and playing the AV system. Group A will not answer the usability part of the questionnaire.

f. Group B is asked to ”think out loud” whenever they use the AV systems. They are asked to tell the researcher everything they have in mind about the system, such as whether the visualization is clear, whether they find some user interface is hard to use. Their response will be recorded by the researcher to analyze the usability.

g. Subjects submitted the questionnaire when they finish.

After all subjects haven been test, their result will be analyzed. The mean correctness or rates of each question in the questionnaire will be compared. For competence questions, the correctness for the individual question will be compared to see if some questions are more difficult than others. There are three questions about self-efficacy (confidence) and three content-related questions (competence). A comparison will be made to see if one of the experimental conditions is associated with greater or lesser correspondence between confidence and competence, and whether one of the conditions is associated with greater competence /confidence in general. Besides, all responses about the usability of group B acquired from step f of the protocol will be gathered to evaluate the usability of this system.

CHAPTER 4. RESULT

4.1 Confidence Result

Subjects are asked to rate their confidence level on their understanding of algorithm, implementation and analysis. The result of these three questions will be shown in this section.

The confidence in understanding the algorithm of two groups is shown in figure 4.1. If the 5 levels from disagreeing to agree can be rated with a number from 1 to 5, the mean value of group A and group B is 4 and 4.6, respectively.

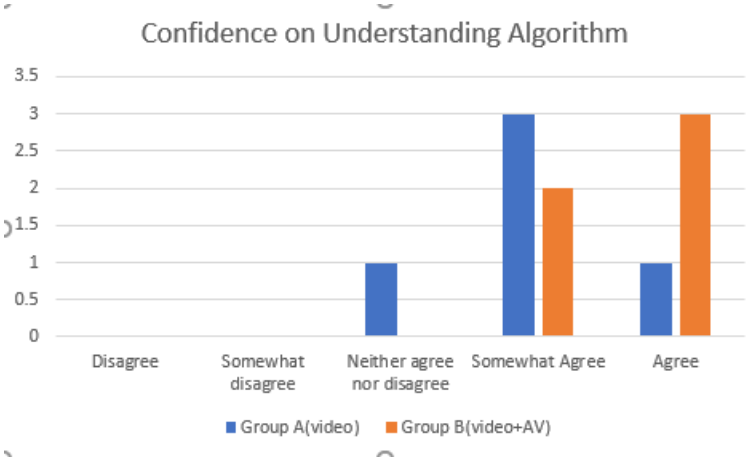


Figure 4.1. Result on understanding algorithm

The confidence in understanding the implementation of two groups is shown in figure 4.2. Similarly, the mean value of group A and group B is 3.8 and 4.6, respectively.

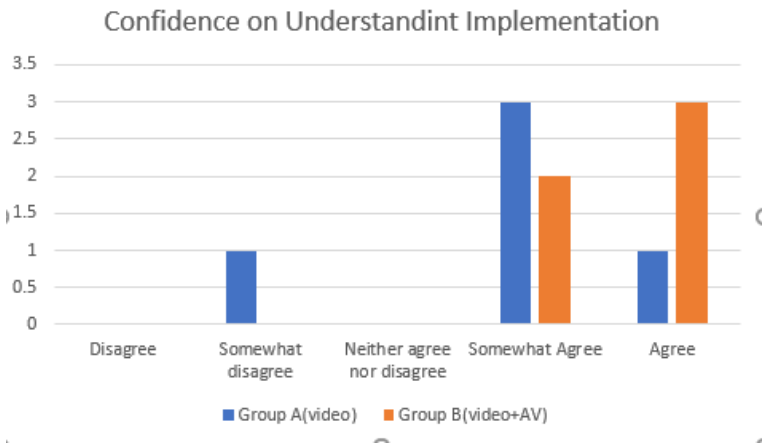
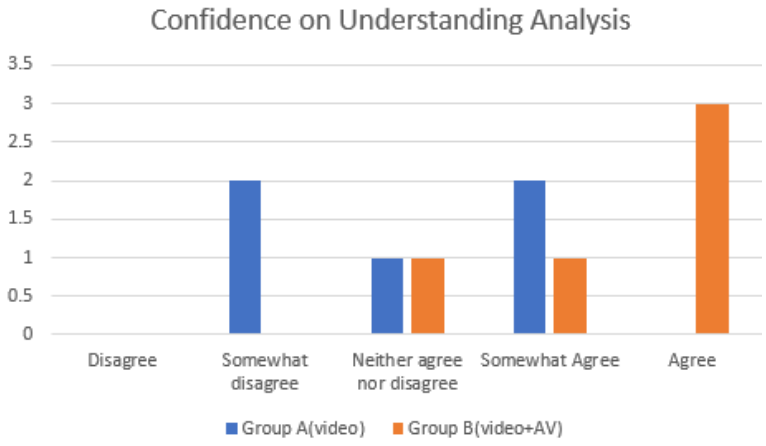


Figure 4.2. Result on understanding implementation

The confidence in understanding the analysis of the algorithm is shown in figure 4.3. Similarly, the mean value of group A and group B is 3 and 4.4, respectively.



4.2 Competence result

In the testing, all subjects are asked to answer 3 questions about the implementation. The count of correct answer of each question of the two group is presented in figure 4.4. The average correct count among all three questions of group A and group B is 1 and 2, respectively.

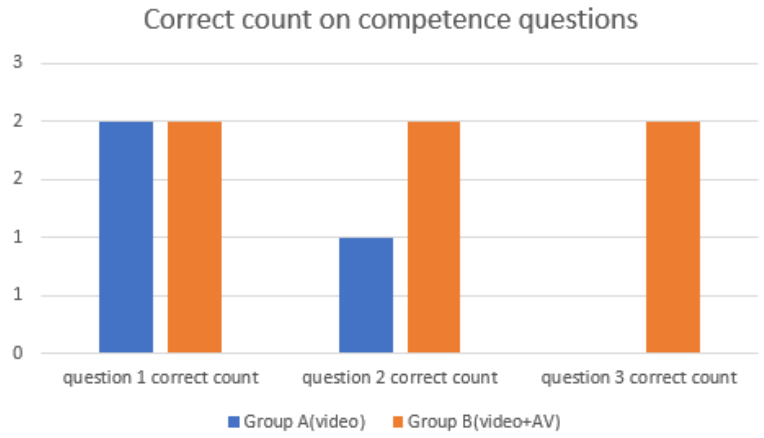
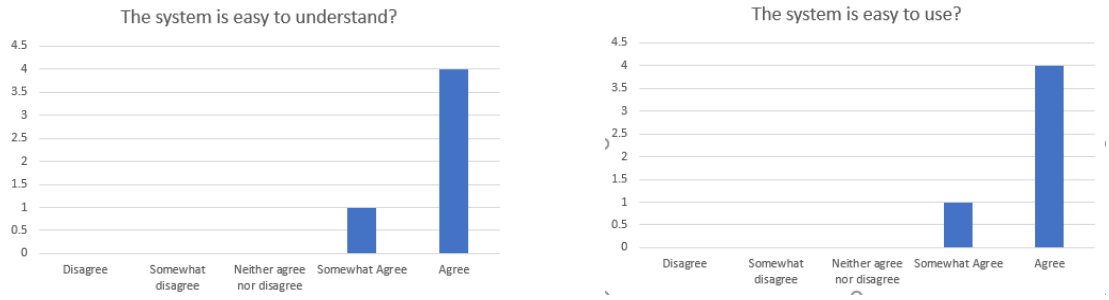


Figure 4.3. Result on competence questions

4.3 Usability result

Each subject is asked to rate how easy the system is to understand and how easy the system is to use. The result is shown in figure 4.5. Besides that, all the usability suggestions retrieved from subjects ”thinking out loud” are collected in figure 4.6. Since some suggestion is mentioned by multiple subjects, the count of each subject is also recorded.



(a) How easy to understand

(b) How easy to use

Figure 4.4. Usability rating result

Table 4.1. *Collections of suggestions*

Usability feed back	Count
The difference between back and create button is confusing	5
The updating of data in the pop up component is not obvious	1
It's not easy to create edge	1
It will be more practical to be able to support importing files as input	1
The representation of graph in create page and visualization page is not consistent	1
Buttons in the header is not big enough	1
The meaning of the arrow is not clear	1

CHAPTER 5. CONCLUSION AND DISCUSSION

5.1 Conclusion

As explained in the limitation, due to the small quantity of the testing subjects and the uneven algorithm ability of them, the research can not reach any concrete conclusions, and only rough conclusions can be reached. Based on the result shown in chapter 4, this system seems to slightly help subjects build both confidence and competence in the understanding of the implementation of the Tarjan-Bridge algorithm. Besides, it's basically user friendly but still need to be improved. In order to get a more convincing outcome, further research should conduct testing on a larger scale of testing subjects with the similar algorithm level.

5.2 Discussion

Besides the conclusions reached from the data of testing, some interesting phenomenon is also observed while doing the testing for the subjects. Since these phenomenons are only subjectively observed, this research will not try to reach any conclusion from them and only discuss them in this section.

The most common phenomenon is that all subjects seem to rely more on their mind than on the AV system when answering some of the competence questions. In fact, all three questions can be easily solved with the help of the AV system. For example, the second question asks what will happen if there's no "parent" variable in the implementation. With the help of the system, they can simply delete the codes in regard to this variable and run the visualization to see the result. However, subjects still tend to think the implementation in their mind even if they are repeatedly told that they can utilize the system to find the answer. After the testing is done, subjects are interviewed about the reason why they prefer their mind than the system, and their answers are different. Some say they no matter how useful the system is if they failed to visualize the implementation in their mind, it's impossible for them to truly understand it. This may provide some insight into why algorithm visualization systems have been studied for decades but still fail to be the mainstream of algorithm education. In other words, algorithm visualization may be more of a tool to check students' understanding of the algorithm than a tool to help them learn it. Others were too confident about the result to use the system while they don't know that they did it wrong. In the future test, researchers may try to tell the subjects which questions they give wrong answers to after that researcher should ask them to try to correct their wrong answers with the help of the system and record whether they can correct them.

Second, the system did help some subjects find correct answers, but it may not actually help them understand the implementation. For example, the last question asks them to find what the return value be if the condition of finding the bridge is slightly changed. Some subjects found that the return value will contain wrong values easily by changing the code in the editor and run the visualization. However, in the interview after the testing, they shared that they don't actually understand why the return value will be wrong. This again may lead to the hypothesis that the AV system can merely be a checking tool inside of a learning tool.

Third, although none of the subjects mentioned this, for recursive algorithms such as Tarjan-Bridge algorithm the visualization of the stack may be helpful. This hypothesis is made due to the fact that think the algorithm should be done when the depth first search reached the last node of the graph. When the visualization still goes on after it reaches the last node, some of them start to question the correctness of the system. In fact, there are functions on the stack that hasn't returned yet, so the algorithm will go back to them. Therefore a visualization of how the stack manipulates the recursive functions may be helpful.

Last but not least, AV systems may not replace traditional algorithm educations media, such as simple video. When answering the competence questions, some subjects prefer to refer to the video than the AV system. As a result, the AV system may not be a replacement of the traditional educational method.

In summary, AV system should be assigned to a less important role in algorithm education. Future research may be conducted to test whether it can become a competent tool to examine students' understanding of algorithm implementation.

REFERENCES

Algfoor, Z. A., Sunar, M. S., & Kolivand, H. (2015). A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, *2015*, 7.

Anderson, E. (2012, August 21). *Techniques for graph data structure management.* Google Patents. (US Patent 8,250,107)

Contero, M., Naya, F., Company, P., Saorín, J. L., & Conesa, J. (2005). Improving visualization skills in engineering education. *IEEE Computer Graphics and Applications*, *25*(5), 24–31.

Cui, X., & Shi, H. (2011). A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, *11*(1), 125–130.

Da, Z., & Jianping, Z. (2009). Knowledge visualization-an approach of knowledge transfer and restructuring in education. In *Proceedings of the 2009 international forum on information technology and applications-volume 03* (pp. 716–719).

Eppler, M., & Burkhard, R. (2004). Knowledge visualization: Towards a new discipline and its fields of applications. *ICAWorking Paper*, *2*, 3–25.

Grissom, S., McNally, M. F., & Naps, T. (2003). Algorithm visualization in cs education: comparing levels of student engagement. In *Proceedings of the 2003 acm symposium on software visualization* (pp. 87–94).

Hamer, J. (2004). A lightweight visualizer for java. In *Proceedings of third progam visualization workshop* (pp. 55–61).

Herman, I., Melançon, G., & Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, *6*(1), 24–43.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, *13*(3), 259–290.

Ifenthaler, D. (2014). Akovia: Automated knowledge visualization and assessment. *Technology, knowledge and learning*, *19*(1-2), 241–248.

Karavirta, V. (2007). Integrating algorithm visualization systems. *Electronic Notes in Theoretical Computer Science*, *178*, 79–87.

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).

Khedr, A. Y., & Bahig, H. M. (2017). Debugging tool to learn algorithms: A case study minimal spanning tree. *International Journal of Emerging Technologies in Learning*, *12*(4).

Koifman, I., Shimshoni, I., & Tal, A. (2008). Mavis: A multi-level algorithm visualization system within a collaborative distance learning environment. *journal of Visual Languages & computing*, *19*(2), 182–202.

Rohrer, R. M., & Swing, E. (1997). Web-based information visualization. *IEEE Computer Graphics and Applications*, *17*(4), 52–59.

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), 9.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), 146–160.

Teresco, J. D., Fathi, R., Ziarek, L., Bamundo, M., Pengu, A., & Tarbay, C. F. (2018). Map-based algorithm visualization with metal highway data. In *Proceedings of the 49th acm technical symposium on computer science education* (pp. 550–555).

Urquiza-Fuentes, J., & Velázquez-Iturbide, J. Á. (2009). A survey of successful evaluations of program visualization and algorithm animation systems. *ACM Transactions on Computing Education (TOCE)*, *9*(2), 9.

Velázquez-Iturbide, J. Á., Pérez-Carrasco, A., & Urquiza-Fuentes, J. (2008). Srec: An animation system of recursion for algorithm courses. In *Acm sigcse bulletin* (Vol. 40, pp. 225–229).

Vrachnos, E., & Jimoyiannis, A. (2014). Design and evaluation of a web-based dynamic algorithm visualization environment for novices. *Procedia Computer Science*, *27*, 229–239.

Ware, C. (2012). *Information visualization: perception for design*. Elsevier.

Yap, P. (2002). Grid-based path-finding. In *Conference of the canadian society for computational studies of intelligence* (pp. 44–55).

APPENDIX A. QUESTIONNAIRE

Algorithm visualization testing

1. How strong is your current understanding of algorithms in general?

Mark only one oval.

	1	2	3	4	5	
Weak	◯	◯	◯	◯	◯	Strong

2. How well do you know about Tarjan algorithm?

Mark only one oval.

	1	2	3	4	5	
Haven't heard about	◯	◯	◯	◯	◯	Know it pretty well

Test

Please rate how well do you understand the Tarjan Algorithm

3. I understand how the theory of the algorithm works.

Mark only one oval.

- ◯ Disagree
- ◯ Somewhat disagree
- ◯ Neither agree nor disagree
- ◯ Somewhat Agree
- ◯ Agree

4. I understand the implementation of this algorithm.

Mark only one oval.

- Disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat Agree
- Agree

5. I understand the time complexity of this algorithm

Mark only one oval.

- Disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat Agree
- Agree

6. What will happen if there isn't "visited" variable in the implementation?

Mark only one oval.

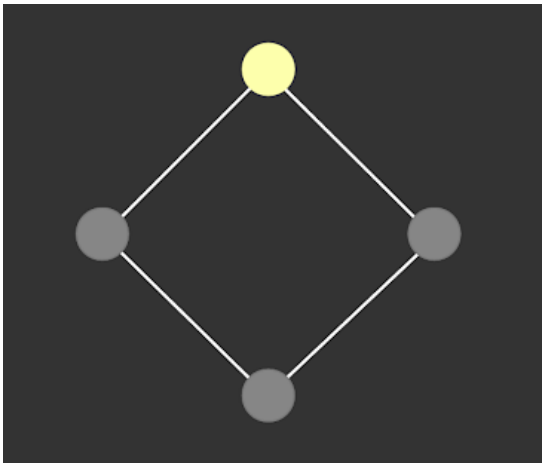
- The result will be wrong.
- Stack will overflow
- The result is still correct but the performance will be worse.
- The result is still right and the performance is still the same.
- None of the above choices

7. What will happen if there isn't "parent" variable in the implementation?

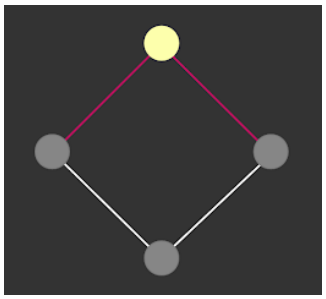
Mark only one oval.

- The return value will still be correct
- The return value will contain edges that are not critical connections
- The return value will always be an empty array
- The return value will be random edges in the graph
- None of the above choices

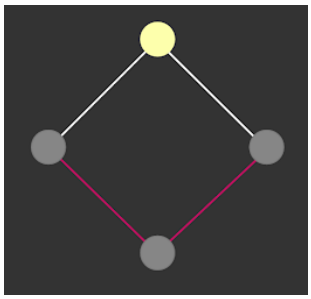
8. This algorithm finds a critical connection when the ids[at] < low_ids[to]. Consider the graph below which doesn't have any critical connections and the algorithm should return an empty array. If the condition is changed to ids[at] <= low_ids[to], what will the return value be ? (The depth first search start from the yellow node)



Mark only one oval.



One of the edges in red color. (Which one will be returned depends on the input)



One of the edges in red color . (Which one will be returned depends on the input)

An empty array.

None of above choices

Post-test

9. It was easy to me to understand HOW the visualization tool operates

Mark only one oval.

	1	2	3	4	5	
Disagree						Agree

10. It was easy to me to use the visualization tool

Mark only one oval.

	1	2	3	4	5	
Disagree						Agree

11. Please share any additional thoughts or suggestions regarding the application, aspects of learning the Tarjan Algorithm, or any other concerns you would like to share with the researcher.