

B.Comp. Dissertation

Teaching Algorithms with Web-based Technologies

By

Erin Teo Yi Ling

Department of Computer Science

School of Computing

National University of Singapore

2014/2015

B.Comp. Dissertation

Teaching Algorithms with Web-based Technologies

By

Erin Teo Yi Ling

Department of Computer Science
School of Computing
National University of Singapore
2014/2015

Project No: H160110

Advisor: Dr. Steven Halim

Deliverables:

- Report: 1 volume

Abstract

VisuAlgo (<http://visualgo.net>) is a web-based learning tool that aims to improve teaching of data structures and algorithms through dynamic interactive visualizations (visualization component), and an automated assessment system (training component). The automated assessment system offers generation of random questions to test students' understanding of data structures and algorithms. This is a development project that focuses on the improvement of the automated quiz system to enhance student learning and to expand adoption of the tool, not only to students across the world, but also to other professors teaching algorithm courses.

Subject Descriptors:

Applied Computing: Education – Computer-assisted instruction

Applied Computing: Education – E-learning

Social and professional topics: Computing education – Computing science education

Information systems: Web applications – Crowdsourcing

Human-centred computing: Visualization – Graph drawings

Keywords:

Web-application, education, visualization, data structures and algorithms

Implementation Software:

HTML5, JavaScript, CSS, PHP, MySQL, Chrome, Firefox

Acknowledgement

I would like to express my utmost gratitude to my supervisor, Dr. Steven Halim for his continuous support and guidance throughout the project.

I am also thankful to the past contributors of VisuAlgo, without whom this project would not be what it is today.

Contents

Abstract.....	iii
Acknowledgement	iv
1 Introduction	1
1.1 About the Project.....	1
1.1.1 Visualization Component.....	1
1.1.2 Training Component	2
1.2 Project Objectives	3
1.3 Summary of Achievements	4
2 Literature Review	5
2.1 Similar Data Structure and Algorithm Quiz Systems	5
2.2 Automatic Feedback on Web-Based Assessment Systems.....	6
2.3 Crowdsourcing	7
3 Improving Quality and Quantity of Questions	9
3.1 Background of the Problem and Objectives.....	9
3.2 Richer Question Bank	9
3.2.1 Adding Questions.....	10
3.2.2 Research on Past Exam Questions	11
3.3 Question Difficulty Calibration.....	12
3.3.1 How Question Difficulty Calibration is done	13
4 Intuitive Feedback and Integration with Visualization Component.....	17
4.1 Background of the Problem and Objectives.....	17
4.1.1 Case Study from CS2010 Semester 1 2014/2015	17
4.2 Usage of Visualization Component to Provide Guidance.....	18
4.3 Providing Useful Feedback	20
5 Using Human-Drawn Graphs	24
5.1 Background of the Problem and Objectives.....	24
5.1.1 The Random Graph Generator	24
5.2 Crowdsourcing for Graphs	26
5.2.1 Graph Drawing in Visualization Component	26
5.2.2 Graph Drawing in Training Component	29
6 Bug Reporting in Training Component.....	31
6.1 Background of the Problem and Objectives.....	31

6.2	Bug Report Feature	32
6.3	False Positives	34
7	Test Administration and Statistics	38
7.1	Background of the Problem and Objectives.....	38
7.2	Finer Test Configuration	38
7.3	Making Test Feature Available to Other Professors	39
7.3.1	Registering Students for Tests	40
7.4	Test Statistics.....	40
8	Conclusion.....	43
8.1	User Feedback	43
8.1.1	Usage of VisuAlgo.....	43
8.1.2	Does VisuAlgo Enhance Learning?.....	43
8.2	Recommendations for Future Work.....	46
	References.....	49
	Appendix A: List of Questions Added.....	51
	Linked List	51
	Recursion.....	51
	Sorting	52
	Hash Table.....	53
	Binary Heap.....	53
	Binary Search Tree.....	54
	AVL Tree	54
	Union Find Disjoint Sets	55
	Graph Data Structures	55
	Graph Traversal.....	56
	Minimum Spanning Tree	57
	Single-Source Shortest Paths	57

1 Introduction

1.1 About the Project

VisuAlgo is a web-based learning tool conceptualized by Dr. Steven Halim in 2011 with the aims of improving teaching of data structures and algorithms through dynamic interactive visualizations. Students from all around the world currently use VisuAlgo, with the website garnering an average of more than 2,000 sessions daily. The site has also been featured on various online media platforms such as reddit (reddit, 2014), Hacker News (Hacker News, 2014), and blogs (Gigazine, 2014), and has received positive reviews from past CS2100 students.

The tool consists of two major components: the visualization component and the training component.

1.1.1 Visualization Component

The visualization component offers teaching of data structures and algorithms through interactive visualizations. The development of this component started in 2011 and it already has an extensive selection of algorithms and data structure visualizations.

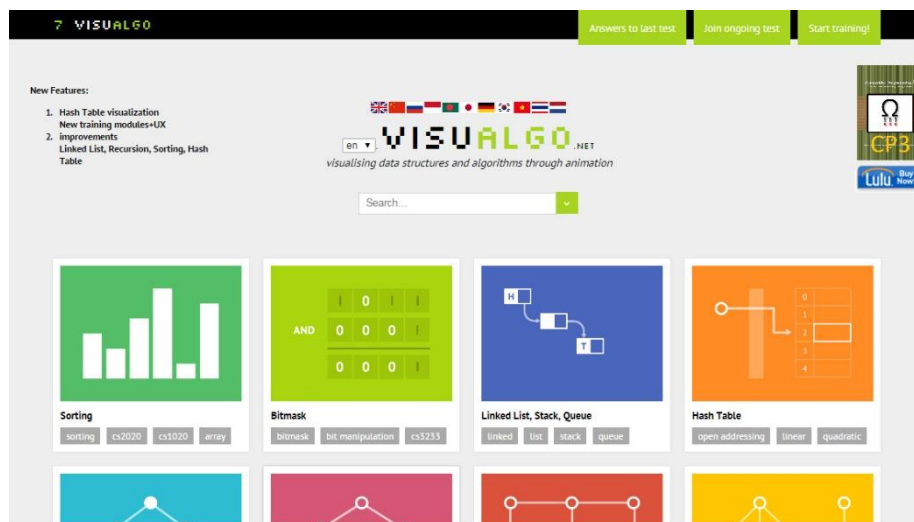


Figure 1: VisuAlgo homepage (Screenshot of visualgo.net, Mar 2015)

It aims to improve learning and teaching of algorithms in the following ways: (Halim, Koh, Loh, & Halim, 2012)

- Having interactive visualizations over static visualizations. Students will have better understanding of an algorithm if they can use their own examples.
- Improve understanding of how algorithms work so a teacher may focus more on advanced applications of algorithms.
- Having a unified interface/tool for visualization of different algorithms so that students have a “one-stop” place for all their learning needs.

1.1.2 Training Component

Just having the visualization component is not enough. Studies such as Brad and Humphreys’ show that working on online graded practice quizzes is positively correlated with student performance on examinations, but sole passive reading is not (Brad & Humphreys, 2001). When students are tested on concepts, any learning gaps or misconceptions will be revealed and they will be able to correct themselves by relearning the concept (e.g. by going back to the visualization component).

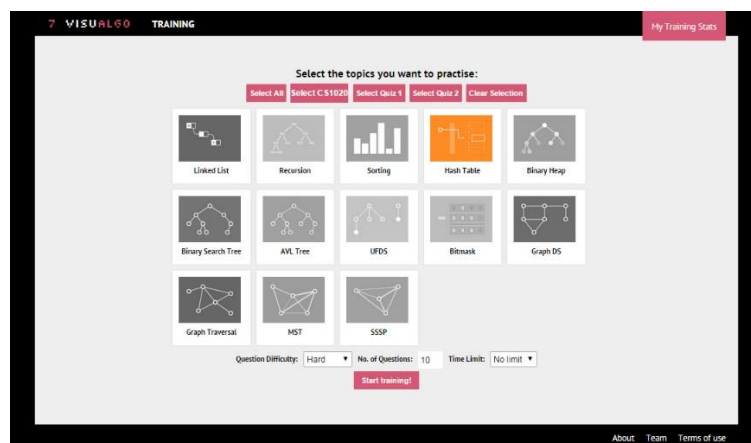


Figure 2: VisuAlgo training page (Screenshot of visualgo.net, Mar 2015)

The training component offers generation of random questions to test students’ understanding of data structures and algorithms. These questions are unlike regular online MCQ questions where the question and the answer are always fixed. The questions have randomized parameters (e.g. the height of the Binary Search Tree presented) and the server verifies the answer automatically. This way, students can keep trying the same question variant but with different parameters, and in order to get the question correct every time, the student has to fully understand the concept and not just memorize the question and answer pair.

One of the limitations of the training component is that the questions asked must have automatically verifiable discrete answers. As such, the training component cannot ask complex design and analysis questions and is not meant to replace written tests crafted by an experienced professor/educator. Instead, the training component's intent is to test and reinforce students' knowledge of the basic concepts of a particular data structure or algorithm such as how the algorithm runs or applications of the algorithm. By doing so, professors can leave the teaching and testing of basic concepts to the tool and concentrate on more interesting and challenging questions to discuss and test during tutorials or examinations.

The training component also includes a test feature that allows professors to create tests with certain parameters (time limit, number of questions, topic, etc.) for students to take part in for grading purposes.

1.2 Project Objectives

The broad goal of VisuAlgo is to use technology to enhance student learning and to expand adoption of tool, not only to students across the world, but also to other professors teaching algorithm courses/modules.

The focus of this project is to improve the quality of VisuAlgo's training component, which is less established than its counterpart (the visualization component), and this will be done by solving the following issues:

- Too few questions in training component
- Inability to control the quality/difficulty of questions generated
- Randomly generated graphs for graph topics are repetitive and are sometimes not appropriate for the question being asked
- Extend the test feature to allow other professors and students to use it
- Training component lacks guidance and feedback to user (only displays the correct answer)
- Visualization component and training component are segregated

In addition to that, my responsibility for this project also includes maintenance of VisuAlgo. Of which involves listening to feedback from students and professors about the tool and developing new features to meet their needs.

1.3 Summary of Achievements

Throughout the course of this project, I have done the following:

- Greatly improved the quality and quantity of questions in the training component
- Better integrated the visualization component with the training component to enhance students' learning
- Implemented intuitive feedback feature for the training component
- Designed and created system for collecting human-drawn graphs from users
- Implemented easy-to-use bug reporting feature
- Fixed numerous bugs reported by users
- Extended the test feature of the training component to allow other professors to use the tool
- Maintained VisuAlgo website together with Dr. Steven Halim
- Worked as a lab teaching assistant for CS2010 2014/2015 Semester 1
- Pitched VisuAlgo together with Dr. Steven Halim to other professors teaching Data Structure and Algorithms modules in NUS.

I also contribute to the project on a regular basis, working an average of 3-4 times per week with the exception of exam periods.



Figure 3: Contribution throughout the project (Source: Github profile, 2015)

2 Literature Review

2.1 Similar Data Structure and Algorithm Quiz Systems

One of the major goals of the project was to improve the training component to better engage users and promote their learning. To achieve this, we study how other existing automatic assessment systems support learning.

Online automatic assessment systems have been developed over the years, with the main aim of reducing the workload of human teachers (Korhonen, Malmi, & Silvasti, 2003) and allowing larger cohorts of students to be taught at a lower cost (Korhonen, Malmi, Myllyselkä, & Scheinin, 2002). These systems support the following functionalities: 1) allows students to access it at any time and place 2) allows resubmission of exercises so students can immediately learn from their mistakes 3) allows personalization of exercises (Korhonen, Malmi, & Silvasti, 2003). The aforementioned functionalities make automatic assessment systems comparable to human guidance (Laakso, Salakoski, & Korhonen, 2005) but should not be made to replace traditional learning environments. Instead, both human guidance and these automatic systems should work together in order to bring a better, more engaging learning experience to students. Laakso et al.'s paper suggests that human guided exercises should be introduced to students first to get them engaged in the course, thereafter; students can guide their own self-learning using web-based learning environments (Laakso, Salakoski, & Korhonen, 2005). In addition, more challenging exercises are difficult to issue in online learning environments because they require more support and time (Korhonen, Malmi, Myllyselkä, & Scheinin, 2002). Massive Open Online Courses are found to have a low completion rate (Parr, 2013) and this is most likely due to the fact that students are simply not motivated to complete the course.

Therefore, while it is possible for our tool to work as a standalone, the benefits will be greater if it is used to support a data structures and algorithms course. Students can use VisuAlgo to direct their own learning and strengthen their understanding of basic concepts, and professors can then better spend their time introducing more interesting and challenging problems, and engaging and motivating students in learning.

TRAKLA2 (Laakso, Salakoski, Korhonen, & Malmi, 2004) is one of the more notable automatic assessment systems that is very similar to VisuAlgo. It is “a system for automatically assessing visual algorithm simulation exercises” (Laakso, Salakoski, Korhonen, & Malmi, 2004). Like VisuAlgo, the system is available online (TRAKLA2 Research Site) but is rather outdated, the last update being in November 2009. It also requires the use of Java applets that requires the Java plugin, which is slow and does not have a good reputation of being secure (Constantin, 2014), to be installed.

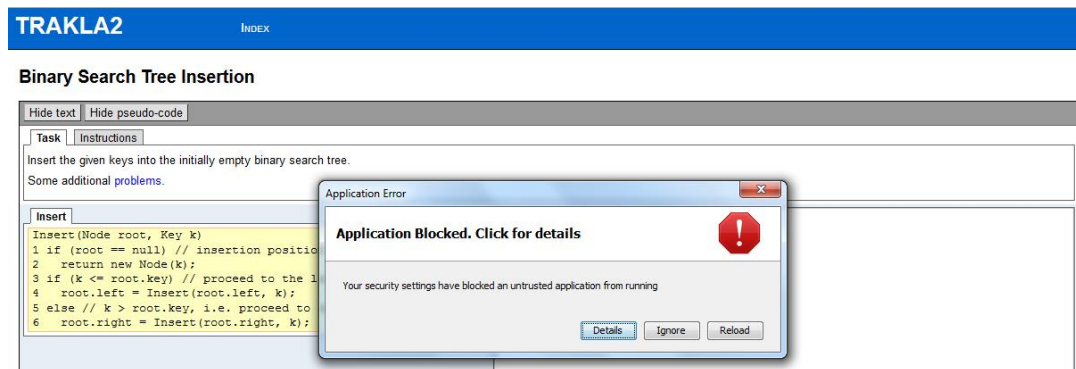


Figure 4: TRAKLA2 blocked by default security settings on Firefox (Screenshot of TRAKLA2, 2015)

TRAKLA2’s assessments mainly require users to simulate various algorithms. Our training component however has a different focus: to validate users’ concepts and understanding of algorithms. The simulation of algorithms is already covered in the visualization component. Visualization of the data structure and assessment are done at the same time in TRAKLA2; the user is given an algorithm to perform and can directly modify and observe the data structure to simulate the given algorithm. This is great for active learning. In VisuAlgo, however, the visualization and training components are segregated and a user is not allowed to modify a given structure in the training component. We may better integrate the training and visualization components such that a structure in the training component can be easily transferred over to the visualization component. Then, in the visualization component, a user may observe how the algorithm works on that particular structure from the training component thus promoting active learning.

2.2 Automatic Feedback on Web-Based Assessment Systems

Automatic assessment systems are not capable of giving feedback on students’ submissions in the same level of sophistication as a teacher (Korhonen, Malmi, Myllyselkä, & Scheinin, 2002). Saikkonen et al. introduce Scheme-Robo, an “automatic assessment system for

Scheme exercises” (Saikkonen, Malmi, & Korhonen, 2001). One of the notable features of Scheme-Robo is that students can be given constructive comments that pinpoint mistakes based on the outcome of their program. A teacher defines comments in a configuration file that describes how solutions should be assessed. Similarly, we can leverage teachers’ knowledge of possible misconceptions or careless mistakes that can be made by students and define comments that will direct students to the correct answer in VisuAlgo’s online training component. Scheme-Robo was considerably well received by students, the only qualm being that error messages given by the system were not adequate (Saikkonen, Malmi, & Korhonen, 2001). However, the quality of error messages significantly depends on the teachers’ configuration file. In our training component, we can record students’ input for questions and analyse common mistakes students make. With this information, we can continually augment the comments given by the training component and give feedback that has the level of detail and sophistication comparable to that of a teacher.

2.3 Crowdsourcing

Crowdsourcing presents a cost-effective method of obtaining content by gathering contributions from the community. Various crowdsourcing solutions have emerged over the years, such as crowdsourcing geographic information to aid in disaster response (Glennon, Goodchild, & Alan, 2010). Leveraging on the idea of crowdsourcing, to present better graphs in the training component we wish to solicit good graph drawings from users of VisuAlgo.

Content obtained by crowdsourcing, however, is not subject to quality control (Glennon, Goodchild, & Alan, 2010). Users may submit “bad” graphs (i.e. graphs that are too trivial or difficult) which are not wanted for questions in our training component. As such, a system must be developed to establish quality control. Majority voting is a basic quality control strategy that involves collecting feedback from multiple human subjects and taking the consensus (Hao, Jia, & Li, 2008). Hao et al. employed this strategy to assess the quality of bounding box annotations drawn by workers on images. In their approach, they have one worker draw bounding boxes on a given image and have another worker verify the correctness of the bounding box. By majority vote, boxes that have strong consensus are considered to be “good” boxes. Their results show that accurate bounding box annotations can be collected through crowdsourcing with quality control mechanisms. Using the strategy of majority voting for quality control, we provide users with the ability to rate graphs.

There are other issues with crowdsourcing such as spammers (Hao, Jia, & Li, 2008). Spammers can compromise the validity of graph ratings by marking all graphs as good/bad. Another implication of crowdsourcing is that “information in which many people have an interest will be more accurate than information that is of interest to only a few” (Glennon, Goodchild, & Alan, 2010). The success of crowdsourcing for collecting good graphs depends highly on users’ motivation to draw good graphs and rate them.

3 Improving Quality and Quantity of Questions

3.1 Background of the Problem and Objectives

One of the priorities of this project is to increase significantly the number of questions provided by the training component as a quiz system's quality is highly dependent on the quality of its question bank. Offering a more substantial amount of questions that cover a broad range of concepts (even concepts that are not explicitly taught in class but are related to the topic at hand), can also greatly improve a student's understanding of a topic especially if the student is diligent enough to attempt the questions repeatedly to cover and master all questions.

To improve the quality of questions asked by the training component, we have to generate questions of an appropriate level of difficulty. Without proper calibration, sometimes either questions that are too trivial (e.g. find the MST of a tree) or too hard/tedious (perform Dijkstra's algorithm on a large dense graph) are randomly generated.

Other than the above, in this project, we aim to add new features to the training component to make it more comprehensive such as new question input methods.

3.2 Richer Question Bank

Throughout the course of this project, the number of questions in the training component has increased by more than 300% (from 8 topics and 41 question types to 12 topics and 129 question types). The summary is given below:

Topic	No. of Qns (Previous, Apr 2014)	No. of Qns (Current, Mar 2015)	Difference
Linked List	0	5	+5
Recursion	0	2	+2
Sorting	0	10	+10
Hash Table	0	7	+7
Binary Heap	10	16	+6
Binary Search Tree	12	18	+6
AVL Tree	1	7	+6
Union Find Disjoint Sets	3	6	+3
Bitmask	4	4	-
Graph Data Structures	3	16	+13
Graph Traversal	2	17	+15

Minimum Spanning Tree	3	9	+6
Single Source Shortest Paths	3	12	+9

Table 1: Summary of Questions Added (Refer to Appendix A for full list)

In addition to enriching the question bank, three new question input methods were implemented (subset of edges, multiple-select/multi-response, and graph drawing). New question input types allow us to ask a broader range of questions.

3.2.1 Adding Questions

Adding a question to the training component involves the following steps:

1. Deciding on a question to ask. This involves research on the types of questions asked in data structure and algorithms courses and picking out the ones that are appropriate (see Section 3.2.2 for more details).
2. Determining if the question to add falls into easy, medium, or hard difficulty (a question may fall into more than one difficulty). In VisuAlgo's training component, questions are classified into these three difficulties so that students will be able to try out easy questions when they first learn the data structure/algorithm and slowly move up to medium and hard difficulties once they have a deeper understanding of the concepts. For example the question "How many structurally different BSTs can you form with X distinct elements?" would be considered a medium/hard question because it requires deeper thinking and in-depth understanding of the Binary Search Tree data structure.
3. Writing the question generator. This includes defining the question wording, defining what parameters to randomize and how to randomize them, calibrating the question for different difficulties (easy, medium, or hard), and defining properties of the graph to be displayed (if applicable). This may also involve extending the graph generator if the graph generator does not provide a graph with the properties needed.
4. Writing the question answerer. This includes writing an algorithm that proposes a model answer for the question asked. Sometimes model answers are unable to be generated or are too complex and slow to generate. In such a case, this step can be skipped but the answer verifier must give appropriate feedback to the user.
5. Writing the answer verifier. This includes writing an algorithm that verifies the correctness of the submitted answer. The verifier may also include logic to offer

students appropriate feedback on why they got the question wrong and how they should correct it (see Section 4 for more details).

Each question took me about 15 minutes to 2 hours (average of an hour) to implement depending on its complexity (excluding the time it takes researching and constructing potential questions to add). Although it takes quite a bit of time to add a new question, it is a one-time effort. Once the question is added, generating the question, verifying students' answers, and offering feedback is all automated and requires no human effort.

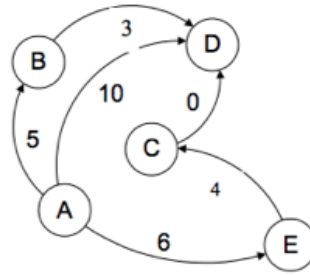
3.2.2 Research on Past Exam Questions

Numerous past examination papers from NUS computing courses such as CS1010, CS1020, CS1102, CS2010, and CS2020 as well as papers from other universities were studied to observe the questions commonly asked to test a students' understanding of the taught data structures and algorithms. Competitive Programming 3 (Halim & Halim, 2013) was also an excellent source of inspiration for the addition of more difficult/interesting questions.

Questions that can be pseudo-randomized with parameters and that have discrete answers are ideal for the question bank. For example the question "Suppose vertex $|value|$ is removed from the following graph. How many components are there in the resultant graph?" – where $|value|$ and graph presented can be randomized. These kinds of questions are usually asked to test a student's basic understanding of concepts in past year examination papers but can now be automatically generated and verified by the training component. This aids students' self-learning as they may continually resubmit exercises and get instant feedback on their mistakes thus strengthening their understanding of concepts.

The following is a sample final exam question from the course CSE326 (Data Structures) from the University of Washington:

Use the following graph for this problem. Where needed and not determined by the algorithm, assume that any algorithm begins at node A.



b) (2 pts) Give two valid topological orderings of the nodes in the graph.

Figure 5: Sample Exam Question on Topological Sort (source: <http://courses.cs.washington.edu/courses/cse332/10sp/oldExams/sample-final-cse326.pdf>)

A similar question can now be asked in VisuAlgo's training component:

3. Click the sequence of vertices that result in a valid topological sort of the graph. If the graph has no valid topological sort, select 'No Answer'.

No answer

Figure 6: Sample Question from VisuAlgo on Topological Sort (source: visualgo.net)

3.3 Question Difficulty Calibration

One of the issues with the training component is that questions are generated randomly and as a result, we sometimes get questions that are too trivial or difficult to solve. This presents a problem in the setting of online quizzes (using the training component's test feature) because lecturers are not able to choose the level of difficulty of their quizzes and may end up with questions that are not appropriate for their quiz. An example of a trivial question is shown below. The question asks for the shortest path between vertex 4 and 5, however it is obvious to see that there is no path between the two vertices and deriving the correct answer to the question is simple.

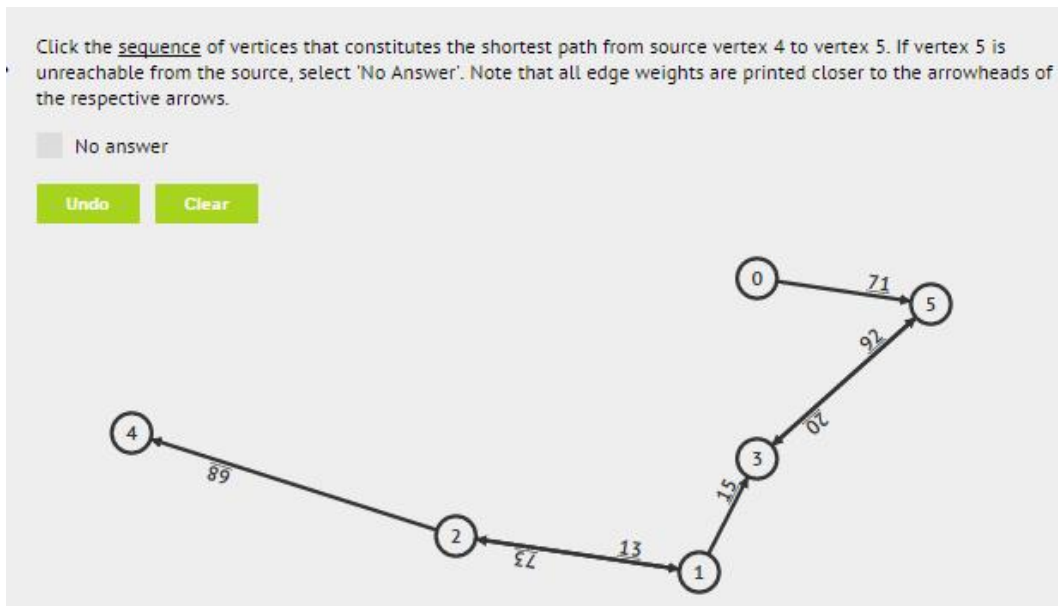


Figure 7: Example of a Trivial Question (Screenshot of visualgo.net, 2014)

Such questions are not meaningful and not ideal for testing purposes. We want to be able to control the level of difficulty for each question. To do so, we calibrate the parameters of each question to provide an easy, medium, and hard version.

3.3.1 How Question Difficulty Calibration is done

Generally, a question is “harder” if there are more steps involved in deriving the answer or it requires a deeper level of understanding. Some broad heuristics can be applied to calibrate questions; for example, a question is likely to be more difficult if the data structure presented is larger/more complex. Question specific heuristics can also be defined for calibration; for example, there will be more steps involved when searching for a node deeper in the binary search tree. Using these heuristics, we can generate questions of varying difficulties.

Throughout the course of the project, all questions were calibrated with to have varying levels of difficulty (where applicable). The following table illustrates the differences in the 3 levels for the question “Given the BST as shown in the picture, click the sequence of vertices that are visited by Search(|value|).”

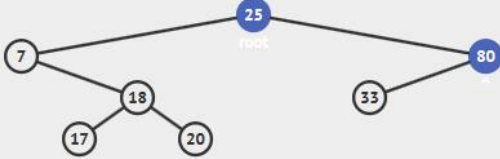
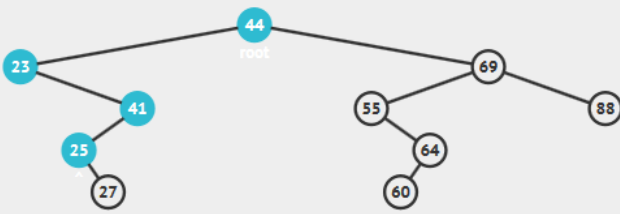
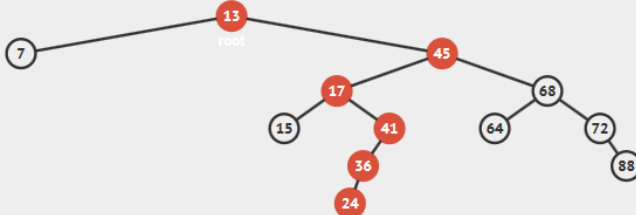
<p>Given the BST as shown in the picture, click the <u>sequence</u> of vertices that are visited by Search(80).</p> <p>Undo Clear</p> <p>Your answer is: 25 , 80</p> 	<p>Difficulty: Easy Small graph, few steps to arrive at answer.</p>
<p>Given the BST as shown in the picture, click the <u>sequence</u> of vertices that are visited by Search(25).</p> <p>Undo Clear</p> <p>Your answer is: 44 , 23 , 41 , 25</p> 	<p>Difficulty: Medium Larger graph, picks a node deeper in the tree, more steps to arrive at answer.</p>
<p>Given the BST as shown in the picture, click the <u>sequence</u> of vertices that are visited by Search(24).</p> <p>Undo Clear</p> <p>Your answer is: 13 , 45 , 17 , 41 , 36 , 24</p> 	<p>Difficulty: Hard Even larger graph, picks a node deeper in the tree, more steps to arrive at answer.</p>

Table 2: Applying the heuristics size of BST and depth of node to calibrate question difficulty

We can see that by just applying some rules, we can calibrate the difficulty of the questions relatively well. The calibration of the difficulty can be better fine-tuned with the addition of more heuristics.

The example above illustrates how we can make the question more difficult by adjusting the parameters so that deriving the answer requires more steps. We can also tweak the difficulty of the question such that solving harder difficulties requires a more in-depth understanding of the concepts. Take for example a basic question on the topic single-source shortest paths.

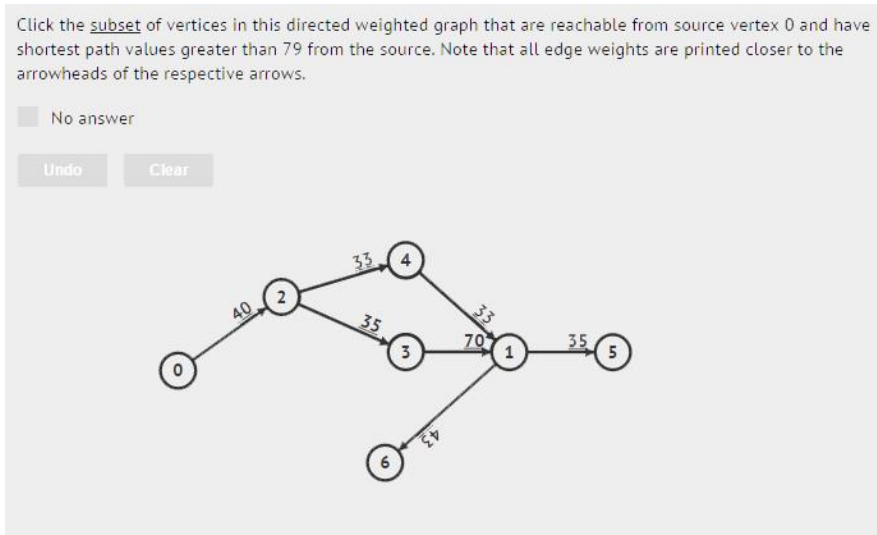


Figure 8: Easy version of the SSSP question (Screenshot from visualgo.net, 2015)

The graph in the easy version of the question would have few alternate paths from the source vertex to other vertices and it is probably easy to calculate the shortest paths without using an algorithm because there are few paths to compare for each vertex.

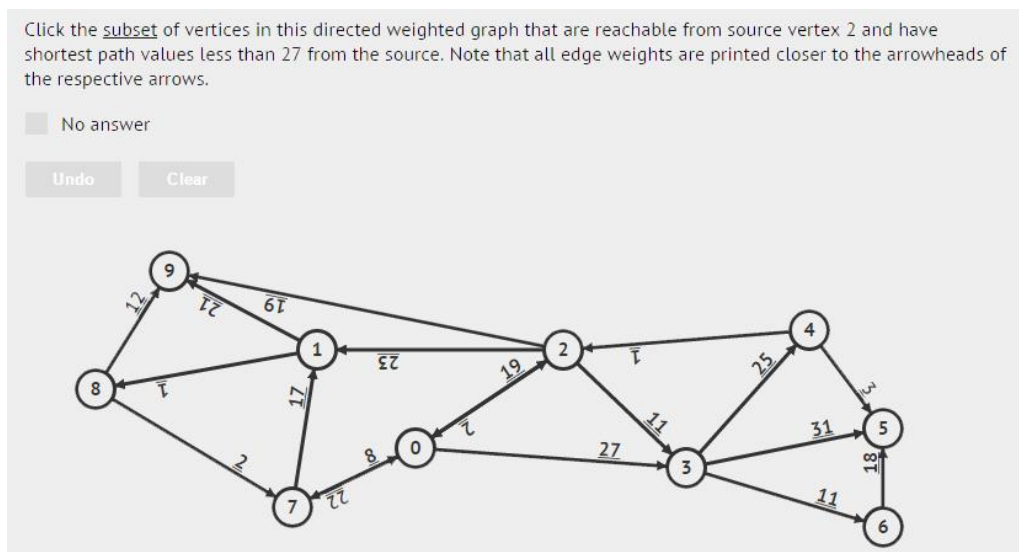


Figure 9: Medium version of the SSSP question (Screenshot from visualgo.net, 2015)

The graph in the medium version of the question would have many alternate paths from the source vertex to other vertices. Then it is not enough to simply look the graph to solve the question, the student must now be able to correctly apply any single-source shortest path algorithm.

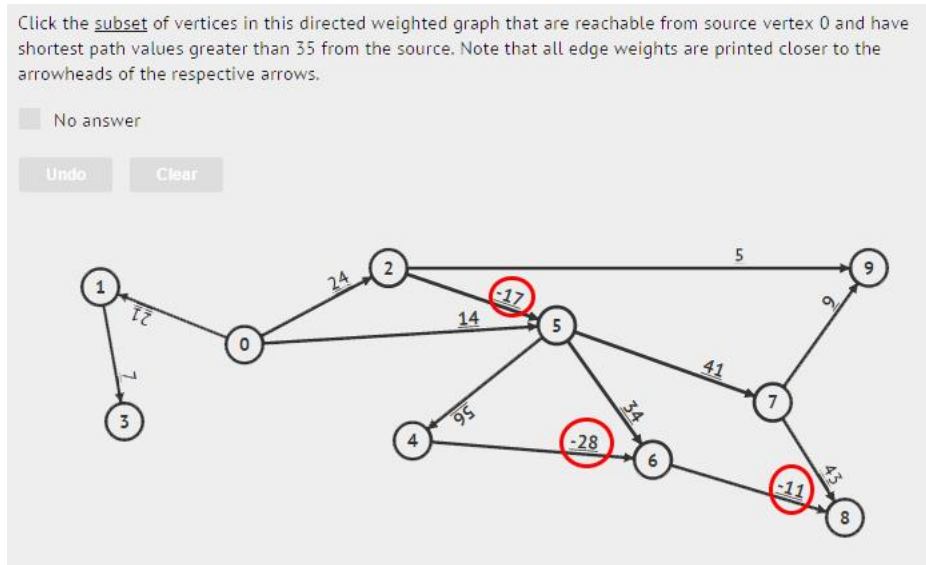


Figure 10: Hard version of the SSSP question (Edited screenshot from visualgo.net, 2015)

The graph in the hard version of the question is similar in structure to the one in the medium version where there are many alternate paths from the source vertex to other vertices so an algorithm must be run to solve the problem. However now, the graph includes negative weights (circled in red in the above image). Therefore, the student must not only apply a single-source shortest path algorithm but also know which ones are appropriate to use. That is, the student must know that Original Dijkstra's algorithm may not produce the correct shortest paths and must use an alternative algorithm to solve the problem.

As shown above, when the level of difficulty increases, the question requires the student to have a deeper understanding of concepts to solve. A student can now attempt questions based on their level of understanding and be introduced to more in-depth concepts as they move to a higher difficulty. This allows them to build their knowledge incrementally.

By adding question difficulty calibration to the training component, we now get more interesting and appropriate questions that are not too trivial or difficult. This will not only benefit lecturers by making it easier to adjust the difficulty of online quizzes, but also students by allowing them to attempt exercises based on their level of understanding.

4 Intuitive Feedback and Integration with Visualization Component

4.1 Background of the Problem and Objectives

The training component is only able to tell the student the following things about their submissions:

1. Whether the answer submitted by the student was correct
2. What the model/correct answer is

While this is adequate in most cases, we want the training component to be able to offer useful comments and feedback that will pinpoint where the student went wrong and to help to direct students to the correct answer, making the tool act more like a teacher.

4.1.1 Case Study from CS2010 Semester 1 2014/2015

When students get a question wrong in the training component, they would like to know what went wrong, and how to correct their mistakes/misconceptions. During the second half of CS2010 Semester 1 2014/2015, an online graded quiz was held. Before the graded quiz, students prepared themselves by trying out questions in the training mode. Over this period, the lecturer of the module, as well as teaching assistants received many e-mails and messages from students asking, “Why did I get this question wrong?”

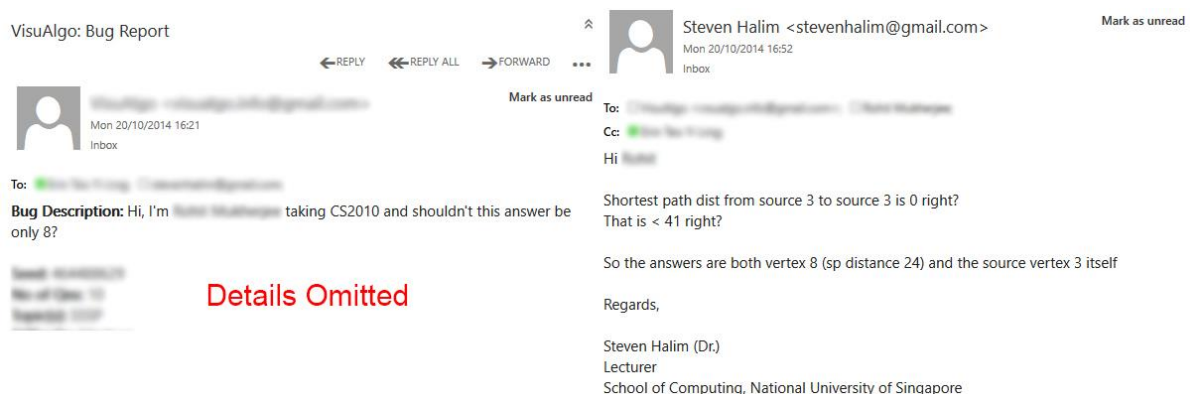


Figure 11: Correspondence between student and lecturer (Edited screenshot taken with permission from Dr. Steven Halim, 2014)

While it is good that students are able to test themselves using the training component and find out that they have a certain misconception or wrong understanding of a particular topic, oftentimes students may not be able to figure out what went wrong themselves and need guidance to correct their mistakes.

Instead of getting educators to do all the work of highlighting what went wrong and guiding students to the correct answer for questions in the training component, we want the training component to provide this service. As mentioned in Section 1.1.2 – Training Component, we want to be able to have professors leave the teaching and testing of basic concepts to VisuAlgo and concentrate on more interesting and challenging questions and concepts.

4.2 Usage of Visualization Component to Provide Guidance

The visualization component provides a collection of interactive data structure and algorithm visualizations for students to learn how these data structures and algorithms work. Before, if students get a question wrong in the training component, they may refer back to the appropriate data structure/algorithm visualization in the visualization component to relearn concepts. However, the visualization and training components are segregated and in order to use the visualization tool to simulate how the algorithm runs on the data structure presented by a question in the training component is tedious. For example, if a student wants to visualize how BFS runs on a general graph presented by the training component, he/she has to go through the following steps:

1. Open the visualization component in a new window/tab
2. Navigate to the appropriate visualization on the list (Graph Traversal)
3. Use the draw graph functionality to manually copy the graph shown in the question to the visualization component
4. Run the algorithm (BFS)

We want to better utilize the visualization component to provide guidance and teaching to students. To do this, we provide a link between the two components and allow data structures to be transferred from the training component to the visualization component. When students get a question wrong in the training component, the training component will provide them with a link that opens the visualization component in a new window. Then by studying the visualization of the algorithm, students will be able to pinpoint their mistakes and correct

them. Providing a link to the visualization component from the training component will also promote the use of the visualization component for learning and review.

Throughout the course of this project, we have managed to provide this link for all questions in the training component that have corresponding visualizations. In the following figure, a student gets a selection sort question wrong and is prompted to try out the visualization (highlighted in red in the image).

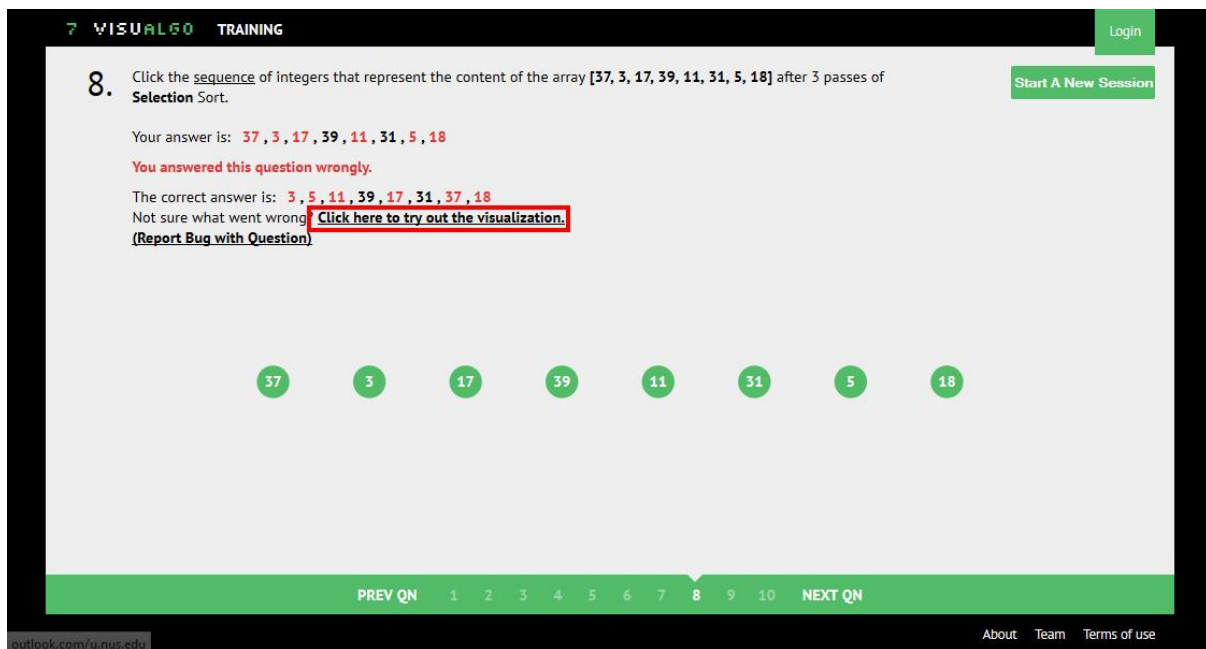


Figure 12: Student gets a selection sort question wrong during training (Screenshot from visualgo.net, 2015)



Figure 13: Link opens selection sort visualization with array presented by question (Screenshot from visualgo.net, 2015)

Upon clicking the link, the selection sort visualization opens in a new window with the exact array presented by the question and the student is able to try out the visualization of the algorithm.

The transfer of data structures from the training component to the visualization component is done using URL query string. The training component generates a URL with an appropriate query string containing parameters such as the data structure. The link opens the appropriate visualization page and the page processes the query string to display the data structure and other parameters. For the above figure, the URL generated by the training component is <http://visualgo.net/sorting.html?create=37,3,17,39,11,31,5,18&mode=Selection>. The sorting visualization page extracts the data structure (array) and sorting mode (selection) from the query string and presents the appropriate visualization. Because we are using URL query string, a student may also bookmark the URL and review the example at another time.

4.3 Providing Useful Feedback

For explaining an answer to the student, providing a visualization would be ideal because it is detailed and easy to understand. However, not all questions in the training component have a related visualization. Other than using the visualization component to provide guidance, we can also provide students with short but useful feedback that will pinpoint their mistakes or

explain why the answer is so. Sometimes a short explanation is sufficient for the user to understand their mistakes and it takes much less time than watching and studying the entire animation of the algorithm using the visualization component.

The following undirected weighted graph only has one unique Minimum Spanning Tree. True or False?

True
 False

You answered this question wrongly.

The correct answer is: **True**
The graph below only has distinct weights. Thus, the MST will be unique.
[\(Report Bug with Question\)](#)

```
graph LR; 2 ---|6| 4; 4 ---|11| 3; 3 ---|15| 0; 0 ---|52| 1; 1 ---|43| 5; 4 ---|19| 5;
```

Figure 14: Example of explanation provided by training component (Screenshot of visualgo.net, 2015)

The above figure shows an example of feedback that can be provided by the training component. When a student gets a question wrong, the training component not only provides the correct answer but also an explanation of why the answer is so.

We can also provide feedback based on common mistakes or misconceptions students may have. For example in the question below, it is common for students to incorrectly answer with the Minimum Spanning Tree instead of the Second Best Minimum Spanning Tree.

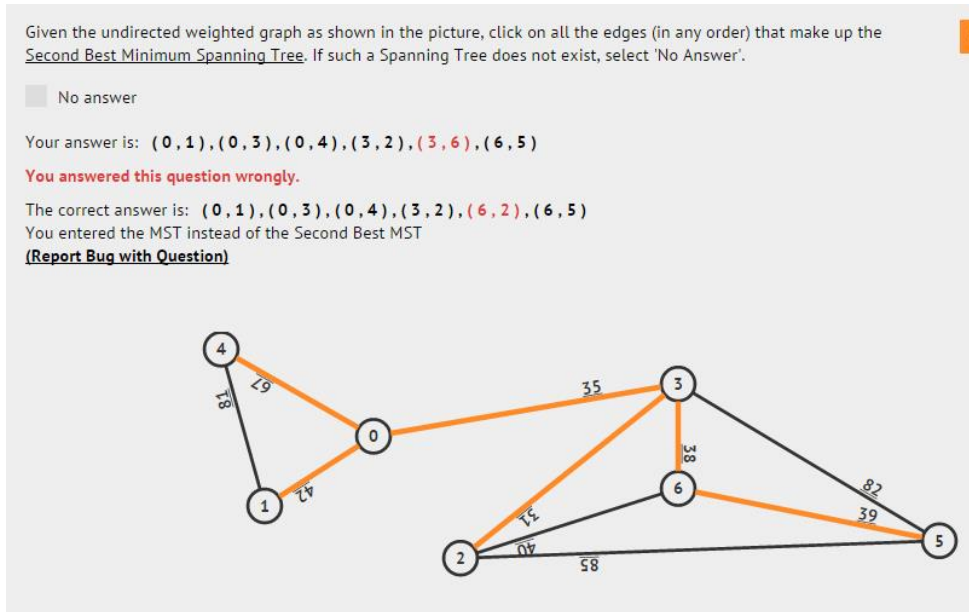


Figure 15: Example of feedback provided by training component (Screenshot of visual.net, 2015)

Students can better learn from their mistakes from the feedback and guidance provided by the training component. The amount of instances where students have to go to their lecturers/professors for help on these basic questions would also decrease. The figure below shows a post on the NUS CS2010 Facebook Group in which a teaching assistant replies to a student's query about a question in VisuAlgo's training component.

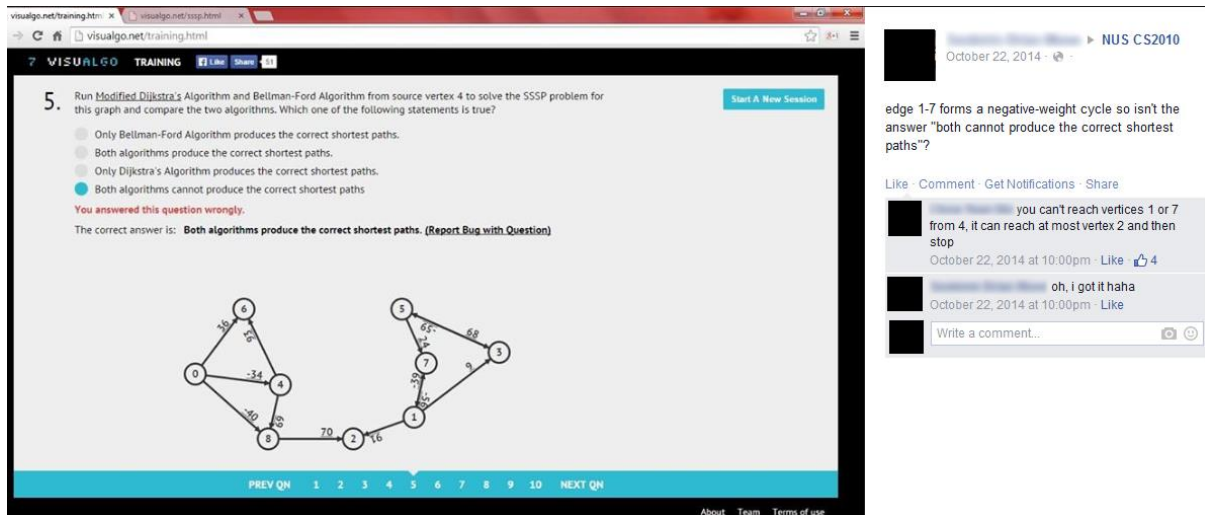


Figure 16: Teaching assistant replying to student's query on CS2010 Facebook Group (Screenshot of facebook.com, 2014)

Run Modified Dijkstra's Algorithm and Bellman-Ford Algorithm from source vertex 2 to solve the SSSP problem for this graph and compare the two algorithms. Which one of the following statements is true?

- Only Bellman-Ford Algorithm produces the correct shortest paths.
- Both algorithms produce the correct shortest paths.
- Only Dijkstra's Algorithm produces the correct shortest paths.
- Both algorithms cannot produce the correct shortest paths

You answered this question wrongly.

The correct answer is: **Both algorithms produce the correct shortest paths.**

The graph contains a negative cycle but is unreachable from the source vertex. Both algorithms will produce the correct answer.

(Report Bug with Question)

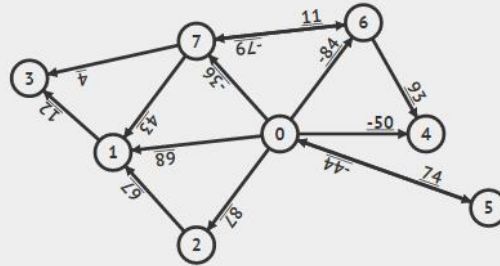


Figure 17: Example of explanation provided by training component of same question (Screenshot of visualgo.net, 2015)

Now the training component can give similar feedback as the teaching assistant.

5 Using Human-Drawn Graphs

5.1 Background of the Problem and Objectives

We want questions in the training tool to be meaningful for learning and testing purposes. Therefore, the questions have to be of a reasonable level of difficulty. To achieve this, we have to generate appropriate parameters for the question. In Section 3.3, we have shown how we can adjust the generation of random parameters using heuristics to calibrate question difficulty. However, the calibration of difficulty in graph topics such as single-source shortest paths is more difficult because we are dealing with general graphs unlike Binary Search Tree, Binary Heap, and Union Find Disjoint sets where the structures are well defined. For questions on graph topics, the quality of the question depends highly on the graph presented. Therefore, we want the graph generator in the training component to be able to accomplish the following:

1. Generate graphs based on certain conditions (e.g. good for illustrating single-source shortest path questions)
2. Draw the graph such that it is understandable, usable, and aesthetically pleasing
3. Generate a multitude of different random graphs so that students do not keep seeing the same graph over and over when attempting questions

However, the above requirements (especially 1 and 2) are hard to do quickly and automatically. Graph drawing itself is a rich topic with many open problems (Brandenburg, Eppstein, Goodrich, Kobourov, Liotta, & Mutzel, 2004) (Battista, Eades, Tamassia, & Tollis, 1994) and is not within the scope of this project. It is also difficult to verify and prove that the automatically generated random graph is always appropriate for the question.

5.1.1 The Random Graph Generator

The random graph generator was implemented to generate graphs with appropriate properties for questions in the training component.

Suppose we want to generate a graph for a Minimum Spanning Tree question. This is how the random graph generator would work:

1. The training component asks the random graph generator for an undirected, weighted, connected graph with N vertices. The graph must also have distinct edge weights.
2. The random graph generator selects a random graph template (usually a large graph) that is undirected and connected.
3. The random graph generator randomly deletes vertices in the graph while checking that the conditions (connected) are still satisfied.
4. The remaining vertices in the graph are numbered from 0 to $N - 1$ randomly.
5. The edges are assigned random distinct weights.

This method produces graphs that are understandable, usable, and aesthetically pleasing as it is based on pre-made graph templates that are already tested and reviewed. However, we observe that this method tends to produce graphs that are very similar. This outcome is expected as the graphs are based on the same templates.

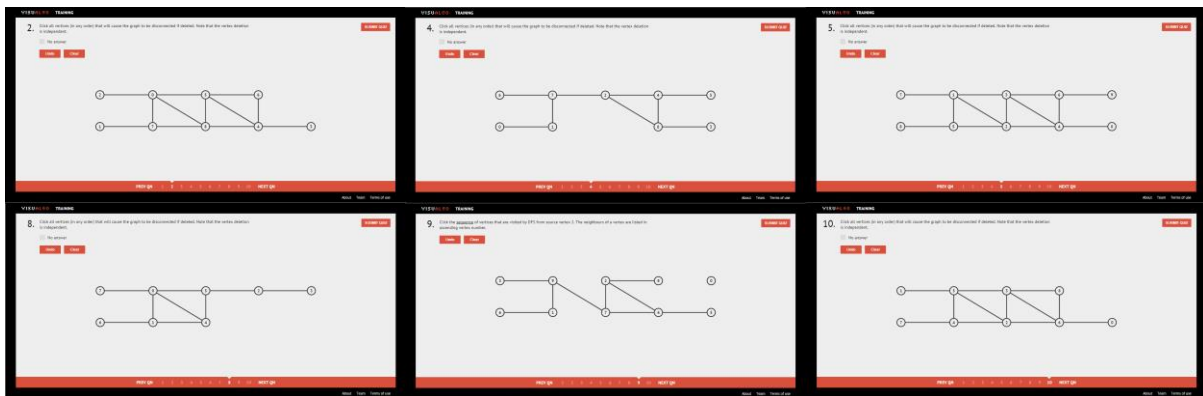


Figure 18: A similar graph structure appeared in 6/10 questions generated by the training component. (Screenshots of visualgo.net, 2014)

Another issue is that we cannot control the difficulty of the graphs generated. Because vertices are deleted and edge weights are assigned in a random fashion, graphs that are generated are not always of appropriate difficulty.

Instead, what we propose is to use human-drawn graphs. Intuitively, humans draw graphs that are reasonably aesthetically pleasing, understandable, usable, and related to the algorithm/topic at hand. Humans are also able to conceive graphs that reveal corner cases or interesting cases for algorithms from experience and understanding of the algorithm. When such corner cases are introduced in questions, it requires the student to reflect and think deeper about the algorithm thus strengthening their knowledge of concepts. Such corner cases

are hard to generate randomly unless very specific heuristics are implemented. However, if rules are too stringent, the randomly generated graphs will be too similar to one another.

5.2 Crowdsourcing for Graphs

We want to collect a large set of suitable human-drawn graphs to present for questions in the training component. Ideally, we want enough graphs such that when a student attempts a particular question/topic in the training component, they see a different graph each time. Some past CS2010 students have stated that when preparing for the graded online quiz, they attempted the same questions more than 10 times. Therefore, we want to collect about 100 graphs for each different topic/graph type. To obtain this many human-drawn graphs, we propose the use of crowdsourcing.

5.2.1 Graph Drawing in Visualization Component

Before soliciting human-drawn graphs, a proper interface has to be set up to allow users to easily draw graphs and submit them to VisuAlgo. A graph database also needs to be set up to allow storage of the user-drawn graphs so that the training component can use them later. We allow users to draw their own graphs in the visualization component so that they are not restricted to visualizing algorithms on template graphs. The users then have the option to submit their graph to VisuAlgo. Students that were engaged to help with developing and improving VisuAlgo; Khac Tung and Jonathan, made VisuAlgo's graph drawing feature possible. My responsibility was to accept user-submitted graphs from the graph-drawing feature and store them in the graph database in a way that the training component can easily retrieve them later.

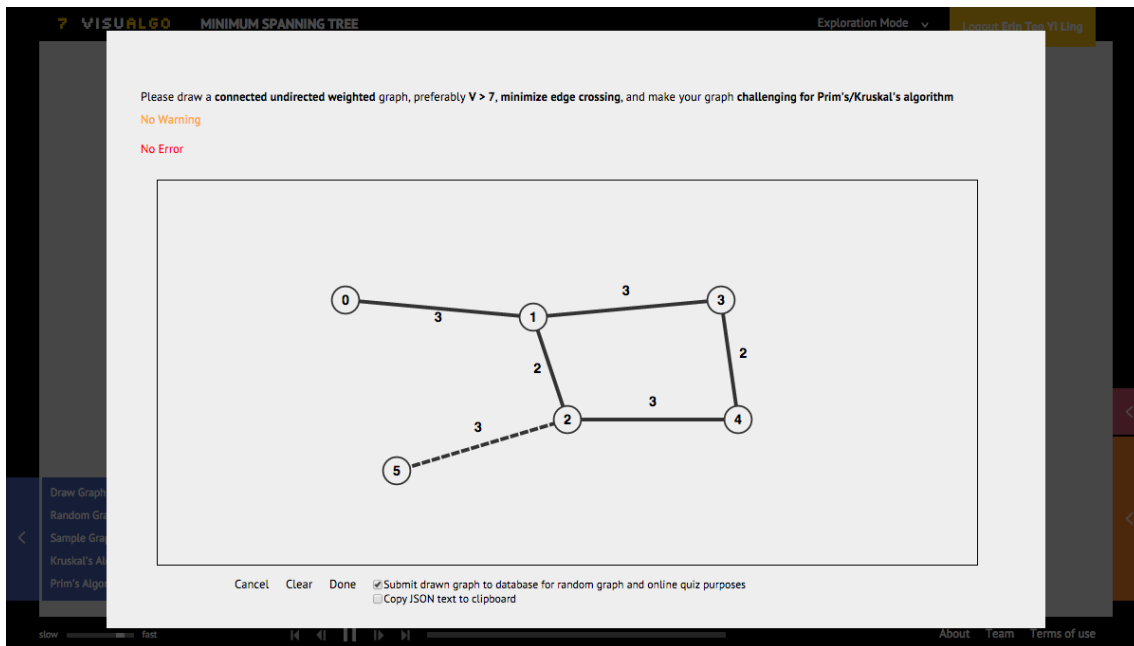


Figure 19: Interface for graph drawing in the Minimum Spanning Tree visualization (Screenshot of visualgo.net, 2015)

Firstly, all graphs submitted by users have to be verified and classified. To make sure the graph submitted is appropriate for the training component, the graph has to be verified before it is saved to the graph database. Verification generally involves making sure the graph submitted is not erroneous (e.g. the graph is submitted in the correct format) and is of a certain quality. The graphs submitted also need to be classified. This is to allow the training component to quickly retrieve graphs with specific properties that are appropriate for certain questions. Several graph algorithms are run on the graph to check if they fulfil certain properties (e.g. bipartite, connected, cyclic, distinct edge weights) and thereafter, the graph is categorized by its properties and stored in the database for quick retrieval. This way, the training component will be able to present graphs for questions in a much quicker fashion than running a random graph generation algorithm for each question.

Regrettably, not all graphs submitted by users are appropriate for the training component. The submitted graph has to be vetted by a proficient and trusted individual (i.e. the system's administrators) in order to determine if it is appropriate for the questions in the training component. To support this, an interface for viewing submitted graphs and committing them to a list of appropriate graphs for the training component (henceforth referred to as committed graphs) was implemented. Admittedly, this process is rather costly because it requires a lot of manual effort and time on the administrators' part, especially if there is an excessive number of submitted graphs to vet. However, human effort is vital to ensuring all graphs presented by

the training component are appropriate for the topic, understandable, usable, and aesthetically pleasing because a computer cannot accurately gauge properties such as whether a graph can be understood by a human. By vetting all the committed graphs, we will also have the utmost confidence that only quality graphs are presented by the training component.

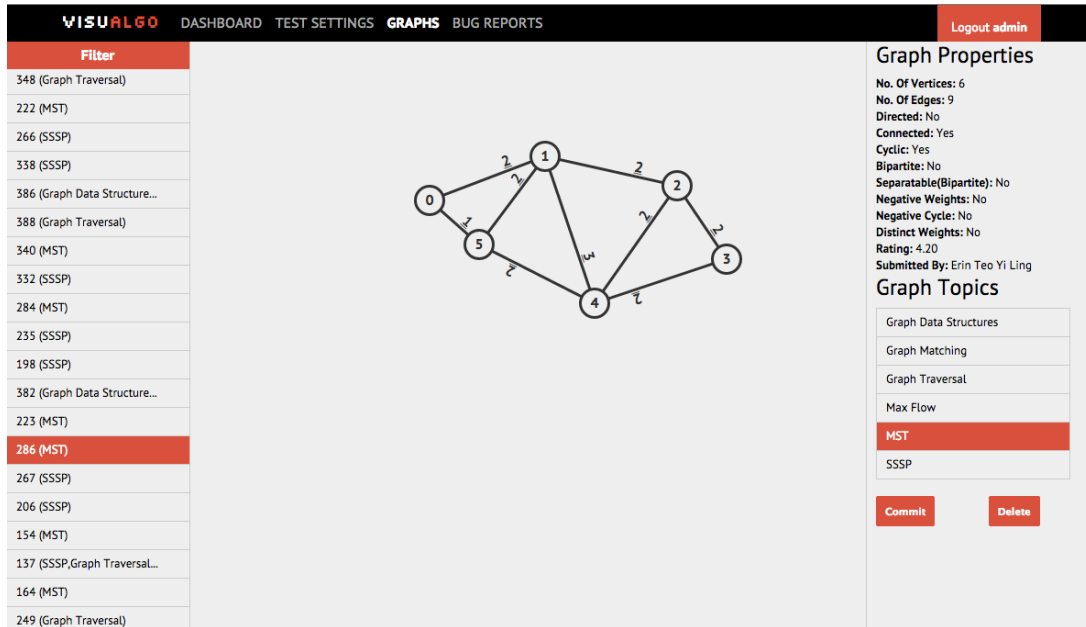


Figure 20: Interface for administrators to review user-submitted graphs and commit appropriate ones to be used in the training component (Screenshot of visualgo.net, 2015)

To ease the process of manually reviewing graphs, we put in place some quality control mechanisms to filter the submitted graphs. For example, we immediately reject duplicate graphs and graphs that are deemed too simple (e.g. null graphs). We also employ majority voting to assess the quality of the submitted graphs. This is done by getting users of the system to rate submitted graphs, and graphs that receive a higher rating are more likely to be of a better quality. Administrators can just look at graphs that are considered as “good” by the public and graphs with low ratings will automatically be rejected.

As mentioned previously in Section 2.3, the success of crowdsourcing for collecting good graphs depends highly on users’ motivation to draw good graphs. Unfortunately, the amount of graphs submitted by users from the visualization component as well as the quality of those graphs were not up to our expectations. We posit that one of the reasons for this is that template graphs are already available in the visualization component for users to run the algorithm on and there is little reason for users to draw their own. There is also little guidance as to what kind of graphs a user should draw so that it constitutes as a “good” graph for the

questions in the training component. As shown in the image below, graphs submitted by users tend to be either too complex or too simple to be used in the training component.

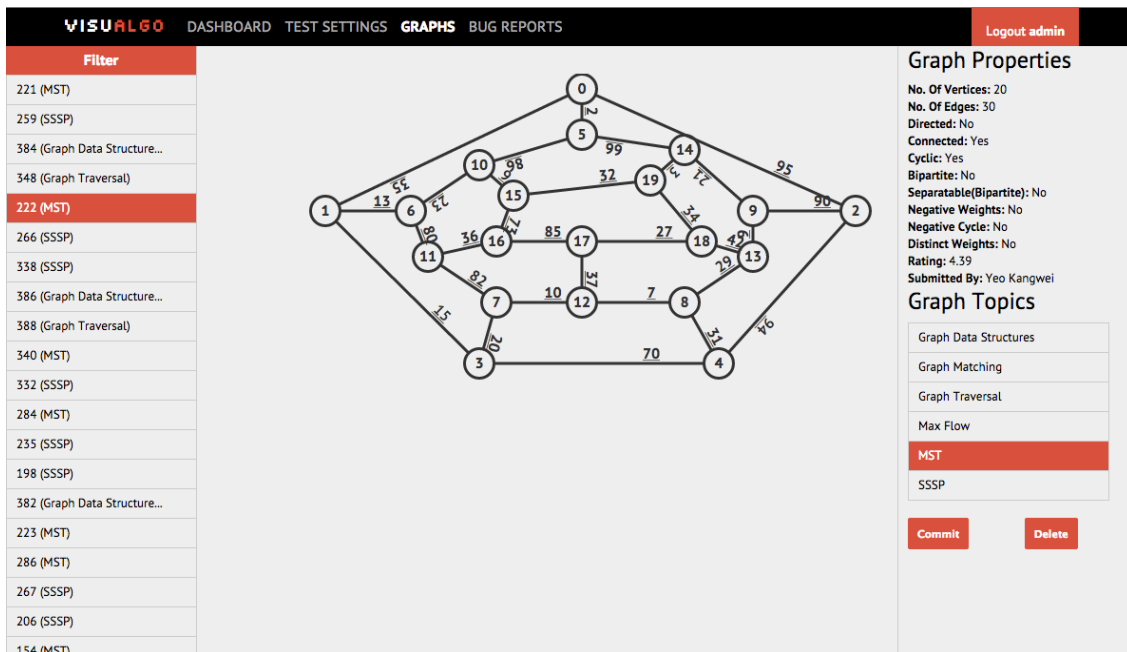


Figure 21: Example of a complex graph submitted by a user (Screenshot of visualgo.net, 2015)

5.2.2 Graph Drawing in Training Component

In order to obtain graphs that we deem suitable for questions in the training component, we require users who draw graphs to follow rules and guidelines. Certain rules can be put in place in the graph drawing tool such as “the graph must be connected” for the minimum spanning tree visualization. However, rules cannot be too stringent and restrictive in the visualization component. If overly restrictive rules are enforced, it would defeat the purpose of having the graph drawing tool in the visualization component, which is to allow users to draw any graph they would like and to run and visualize the algorithm on the graph.

Instead, we solicit graphs from users by utilizing graph drawing questions in the training component. With graph drawing questions, we can ask users to draw graphs that satisfy certain conditions and criteria. The conditions and properties prescribed by the question can be adjusted such that users draw graphs with desirable properties that can be used in the training component. Then when a user gets the question right (i.e. the graph drawn fulfills the conditions stated), the graph gets submitted to the graph database. The administrator’s work of vetting graphs also becomes easier because the administrator just needs to make sure the graph is understandable by humans as only graphs that suit our needs get submitted.

Over the course of the project, we've managed to integrate the graph drawing component from the visualization component into the training component, and implement 11 graph drawing questions.

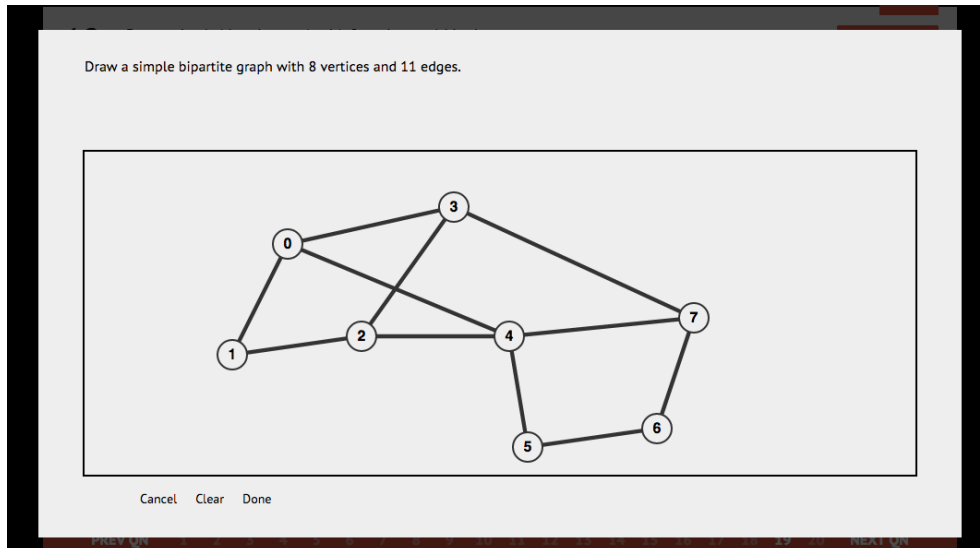


Figure 22: Example of graph drawing question in the training component (Screenshot of visualgo.net, 2015)

The figure above illustrates how we can acquire very specific types of graphs from users. If the graph database is lacking of a certain type of graph, we can easily craft a graph drawing question that requires users to draw that type of graph. For example, for topological sort questions, we require a collection of directed acyclic graphs. To acquire such graphs from users, we can simply formulate a question as shown below.

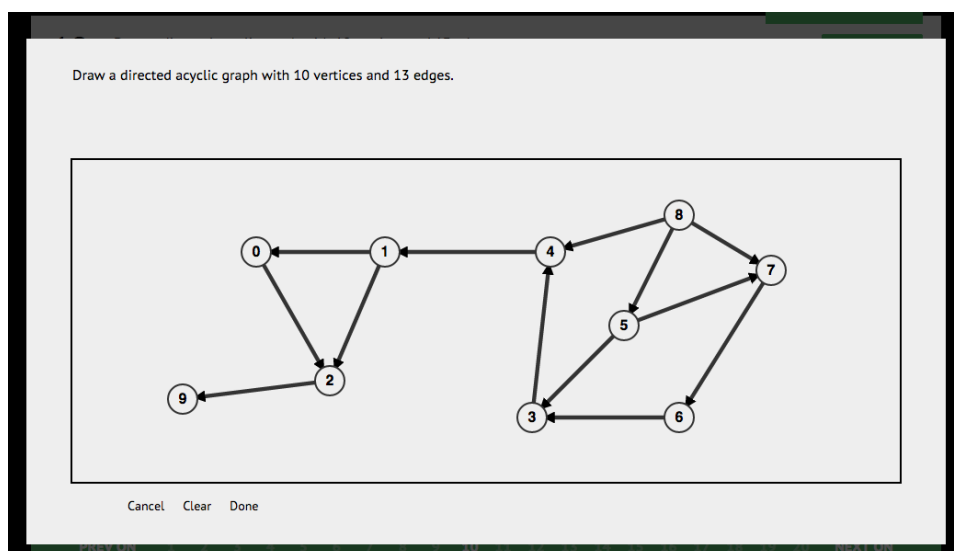


Figure 23: Example of graph drawing question in the training component (Screenshot of visualgo.net, 2015)

6 Bug Reporting in Training Component

6.1 Background of the Problem and Objectives

People all over the world are currently using VisuAlgo. According to data obtained from Google Analytics, we saw that from 1 August 2014 to 31 October 2014, VisuAlgo had 253,687 hits (average of ~2800 hits per day) originating from various areas such as the United States (40K hits), India (26K hits), China (26K hits) and Singapore (18K hits). Bugs will deter people from using our tool and may even confuse people (for example returning the wrong answer in the training component). In this project, we aim to make the system as bug-free as possible. One way to easily spot and remove bugs is for users to report them as soon as they come across one. As such, we want to provide a reliable and easy to use bug-reporting feature.

Previously, when users spot a bug in VisuAlgo, they would send an email to Dr. Steven Halim describing the bug with attached screenshots of the bug. Below is an example of an email describing a bug in the training component.

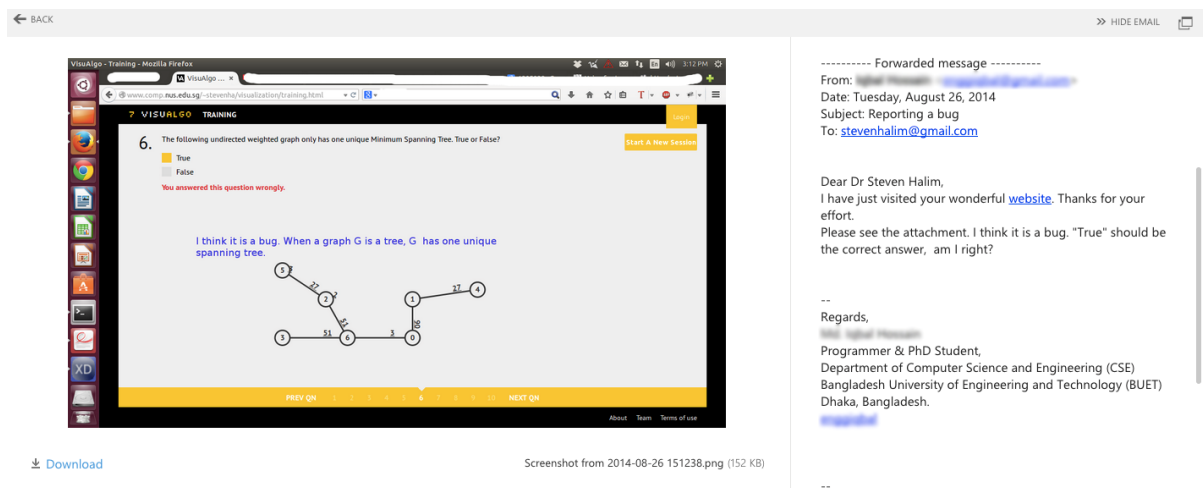


Figure 24: Email describing a bug in training component (Screenshot of outlook.com, 2015)

Oftentimes, knowing the existence of a bug is not enough to fix it. We need to replicate the bug on our own environment (if we do not replicate the bug, we also cannot tell if we have fixed it). Generating the exact same question as the one presented in the bug report is almost impossible because question parameters are generated randomly. This makes the replication of bugs more difficult.

6.2 Bug Report Feature

We want to make it simpler for users to report bugs. Instead of going through the entire process of looking for Dr. Steven Halim’s email address, writing an email detailing the bug, and capturing and editing screenshots of the bug, users can report a bug simply by clicking a button. When the process of reporting a bug is simpler for users, they are more likely to report bugs that they come across as well.

In the training component, a student would report something as a bug when they get a question wrong and think that there is a discrepancy between the model answer given by the training component and their own understanding of the question and topic. In the figure below, there is clearly a path between vertex 2 and 9. However, the training component erroneously gives zero as the correct answer. Users can easily report this error as a bug by clicking the “Report Bug with Question” button (highlighted in red in the image below).

The screenshot shows the Visualgo Training interface. At the top, it says "7 VISUALGO TRAINING" and has a "Login" button. The main content area displays a question: "1. In the following directed acyclic graph, how many simple paths are there from vertex 2 to 9?". Below the question is an input field containing the number "2". A red message says "You answered this question wrongly." Below that, it says "The correct answer is: 0" followed by a red-bordered button labeled "Report Bug with Question". Below the text is a directed graph with 10 vertices (0-9) and several edges. The graph is a directed acyclic graph (DAG). The edges are: 0 to 1, 0 to 2, 0 to 3, 0 to 4, 0 to 5, 0 to 6, 0 to 7, 0 to 8, 0 to 9, 1 to 2, 1 to 3, 1 to 4, 1 to 5, 1 to 6, 1 to 7, 2 to 3, 2 to 4, 2 to 5, 2 to 6, 2 to 7, 2 to 8, 2 to 9, 3 to 4, 3 to 5, 3 to 6, 3 to 7, 3 to 8, 3 to 9, 4 to 5, 4 to 6, 4 to 7, 4 to 8, 4 to 9, 5 to 6, 5 to 7, 5 to 8, 5 to 9, 6 to 7, 6 to 8, 6 to 9, 7 to 8, 7 to 9, 8 to 9.

At the bottom of the interface, there are navigation buttons: "PREV QN", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "NEXT QN". In the bottom right corner, there are links for "About", "Team", and "Terms of use".

Figure 25: User notices a bug in the training component (Screenshot of visualgo.net, 2015)

Upon clicking the button, users are prompted to enter a description of the bug before submitting the bug report. When the training component receives a bug report, it saves the bug report in the database and sends an email containing the bug report to the administrators and developers of VisuAlgo.

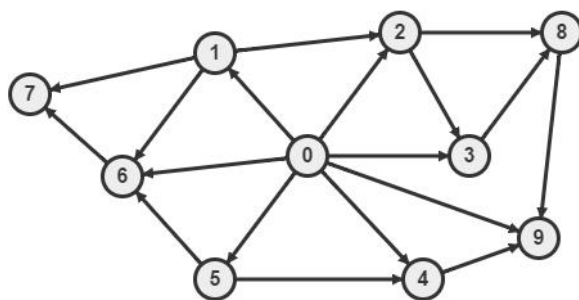
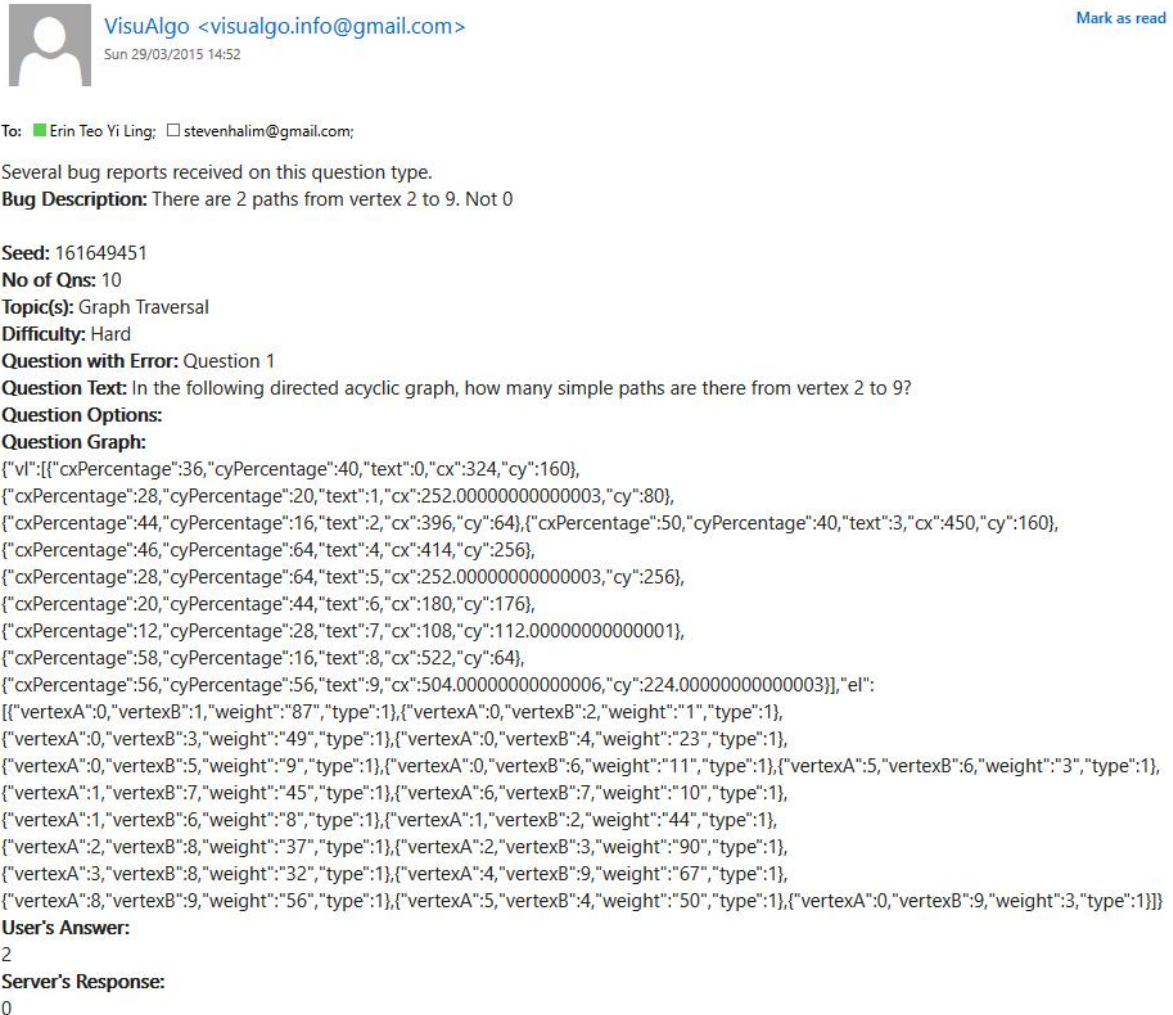


Figure 26: Sample email of bug report sent to administrators and developers of VisuAlgo

The bug report would include the following:

1. The seed used to generate the questions
2. The number of questions used to generate the questions
3. The topics selected by the user
4. The difficulty selected by the user
5. The question where the error was found
6. Details about the question such as the question text, question options, and the questions graph (the JSON and as well as an attached picture of the graph)
7. The user's answer and the answer given by the training component

With the seed, number of questions, topics, and difficulty we can generate the exact same question as the one seen by the user and replicate the bug. This makes it easier and quicker for developers of VisuAlgo to fix reported errors. All bug reports are archived and can be viewed in the administrator's control panel. When bugs are fixed, administrators can mark them as fixed here.



Fixed	Reported On	Topic	Qn Type	Message
<input type="checkbox"/>	3/29/2015 14:52:03	Graph Traversal	countPathsDag	There are 2 paths from vertex 2 to 9. Not 0
<input type="checkbox"/>	3/29/2015 12:52:00	Graph Traversal	countPathsDag	Why 0??
<input type="checkbox"/>	3/29/2015 12:46:32	Graph Traversal	countPathsDag	The answer should be 2.
<input type="checkbox"/>	3/29/2015 12:31:19	Graph Traversal	countPathsDag	Vertex 4 and 6 are connected by a single path. So shouldn't the answer be 1?
<input type="checkbox"/>	3/27/2015 13:55:40	Recursion	recursionComplexity1	variable sum is never defined

Figure 27: Bug Reports tab in administrator's control panel (Screenshot of visualgo.net, 2015)

6.3 False Positives

When the bug-reporting feature was launched, we received many bug reports from students, majority of which were false positives. Students report errors when the answers presented by the training component do not reflect their understanding of the topic. However, when students have misconceptions about the data structure or algorithm, they may wrongly report correct behaviour as a bug. An example of a false bug report is shown below.

Bug Description: Hi Steven & Erin, I'm pretty sure I read the question correctly, 3 components with minimum weight. Seems like the answer is the inverse of my answer. Thanks!

Seed: 954333510

No of Qns: 10

Topic(s): Graphs, GraphTraversal, MST, SSSP

Difficulty: Hard

Question with Error: Question 8

Question Text: Select the edges (in any order) that form 3 connected components and the total weight of these components is **Minimum**.

Question Options:

Question Graph:

```
{\"v\": [{\"cxPercentage\": 42, \"cyPercentage\": 20, \"text\": \"0\", \"cx\": 378, \"cy\": 80}, {\"cxPercentage\": 58, \"cyPercentage\": 40, \"text\": \"1\", \"cx\": 522, \"cy\": 160}, {\"cxPercentage\": 56, \"cyPercentage\": 76, \"text\": \"2\", \"cx\": 504.00000000000006, \"cy\": 304}, {\"cxPercentage\": 68, \"cyPercentage\": 8, \"text\": \"3\", \"cx\": 612, \"cy\": 32}, {\"cxPercentage\": 46, \"cyPercentage\": 52, \"text\": \"4\", \"cx\": 414, \"cy\": 208}, {\"cxPercentage\": 54, \"cyPercentage\": 12, \"text\": \"5\", \"cx\": 486.00000000000006, \"cy\": 48}, {\"cxPercentage\": 28, \"cyPercentage\": 40, \"text\": \"6\", \"cx\": 252.00000000000003, \"cy\": 160}], \"e\": [{\"id\": 0, \"vertexA\": 4, \"vertexB\": 1, \"weight\": 70, \"displayWeight\": true, \"type\": 0}, {\"id\": 1, \"vertexA\": 1, \"vertexB\": 5, \"weight\": 77, \"displayWeight\": true, \"type\": 0}, {\"id\": 2, \"vertexA\": 4, \"vertexB\": 0, \"weight\": 38, \"displayWeight\": true, \"type\": 0}, {\"id\": 3, \"vertexA\": 0, \"vertexB\": 5, \"weight\": 59, \"displayWeight\": true, \"type\": 0}, {\"id\": 4, \"vertexA\": 6, \"vertexB\": 0, \"weight\": 11, \"displayWeight\": true, \"type\": 0}, {\"id\": 5, \"vertexA\": 6, \"vertexB\": 4, \"weight\": 52, \"displayWeight\": true, \"type\": 0}, {\"id\": 6, \"vertexA\": 4, \"vertexB\": 1, \"weight\": 70, \"displayWeight\": true, \"type\": 0}, {\"id\": 7, \"vertexA\": 4, \"vertexB\": 2, \"weight\": 31, \"displayWeight\": true, \"type\": 0}, {\"id\": 8, \"vertexA\": 4, \"vertexB\": 2, \"weight\": 21, \"displayWeight\": true, \"type\": 0}, {\"id\": 9, \"vertexA\": 2, \"vertexB\": 1, \"weight\": 21, \"displayWeight\": true, \"type\": 0}, {\"id\": 10, \"vertexA\": 5, \"vertexB\": 3, \"weight\": 8, \"displayWeight\": true, \"type\": 0}, {\"id\": 11, \"vertexA\": 3, \"vertexB\": 1, \"weight\": 73, \"displayWeight\": true, \"type\": 0}], \"type\": 0}}
```

User's Answer:

[[1,5],[3,1],[4,1],[0,5],[6,4],[4,0]]

Server's Response:

[5,3,6,0,2,1,4,2]

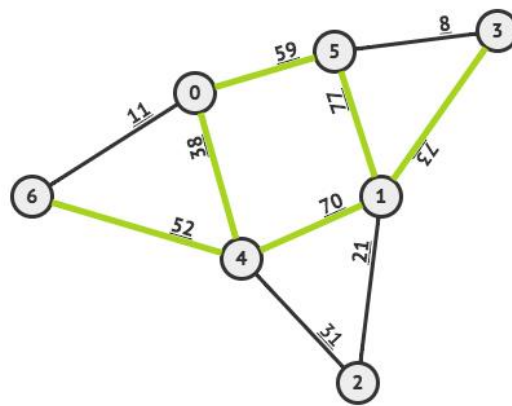


Figure 28: An example of a false bug report from user

Bug reports, especially received in the form of an email, create a sense of urgency and concern in developers. We do not want developers to be falsely alarmed and overwhelmed by these incorrect bug reports from users. As such, we only want developers and administrators to only be alerted of a bug via email when there is truly an error in the system. To do this, we only alert administrators and developers of a bug via email when more than one user has reported the “same bug”. By the term “same bug”, we mean that the error comes from the same question type. So for instance, if more than one user has reported a bug in a Minimum Spanning Tree question on Kruskal’s algorithm, an email containing the bug report will be sent to VisuAlgo’s administrators and developers. The intuition behind this is if more than one person reports a bug in the question, then the discrepancy between the answer given by the training component and the user’s answer is less likely to be a misunderstanding on the user’s part but a real error in the system.

All bug reports from users, especially false positives, give us useful insight on how our users interact with the system and we may use this information to make the system better for our users. When a user submits a false bug report, it reveals to us certain misconceptions and common mistakes users have when attempting a particular question in the training component. With this knowledge, we may add additional logic to improve the feedback that the training component provides for that particular question (See Section 4). False positive bug reports may also reveal shortcomings in the user interface of VisuAlgo. For instance, in the bug report shown below, we can gather that the user thinks that the system marks his answer as incorrect because it regards the edge [2, 3] as different from [3, 2] when they are actually the same edge. In fact, his answer is marked as incorrect because it differs with the system's answer by the last edge.

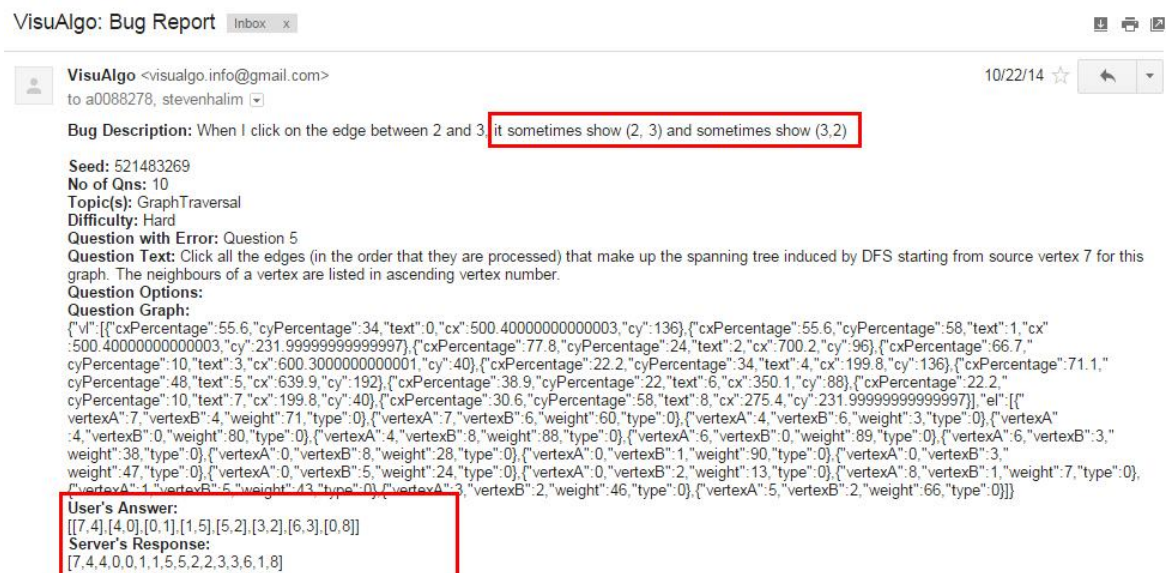


Figure 29: An example of a false bug report from user

The above false positive bug report made us understand that it was difficult for users to spot the difference between their own answer and the training component's answer, specifically when the answer is a long list of vertices/edges. As shown in the figure below, this led us to improve the interface for the training component such that it highlights the differences between the two answers making it easier for the users to identify their mistakes.

Click all the edges (in the order that they are processed) that make up the spanning tree induced by DFS starting from source vertex 5 for this graph. The neighbours of a vertex are listed in ascending vertex number.

No answer

Your answer is: (3, 5), (1, 3), (0, 1), (0, 4), (4, 8), (8, 7), (7, 9), (2, 3), (5, 6)

You answered this question wrongly.

The correct answer is: (5, 3), (3, 1), (1, 0), (0, 4), (4, 8), (8, 7), (7, 9), (3, 2), (2, 6)

Not sure what went wrong? [Click here to try out the visualization.](#)

[\(Report Bug with Question\)](#)

Figure 30: Highlighting of differences between user's answer and system's answer (Screenshot of visualgo.net, 2015)

As a developer of VisuAlgo, I felt the bug reports from users were very helpful as the reports revealed special corner cases that I did not think about and handle in the code.

7 Test Administration and Statistics

7.1 Background of the Problem and Objectives

In April 2014, only one lecturer could use VisuAlgo's test feature: Dr. Steven Halim. The test feature also only allowed for one test to be running at any given time. We want to extend the system so that other professors/lecturers teaching data structures and algorithms courses/modules may use the test feature to hold graded online quizzes for students.

The test feature also allows very little configuration and calibration. The professor is only allowed to set the test seed, total number of questions, test time limit, and the topics.

We also want to offer professors more statistics and information about students' performance on tests. By offering statistics on students such as their scores in the quiz, for example, professors can gauge students' performances, gain insights about their weaker topics or misconceptions, and take action based on that information.

7.2 Finer Test Configuration

The test feature now allows for more options and calibration of tests to suit lecturers' assessment goals. The lecturer is now allowed to specify the number of questions for each topic that he includes in his test and the test difficulty (Easy, Medium, or Hard).

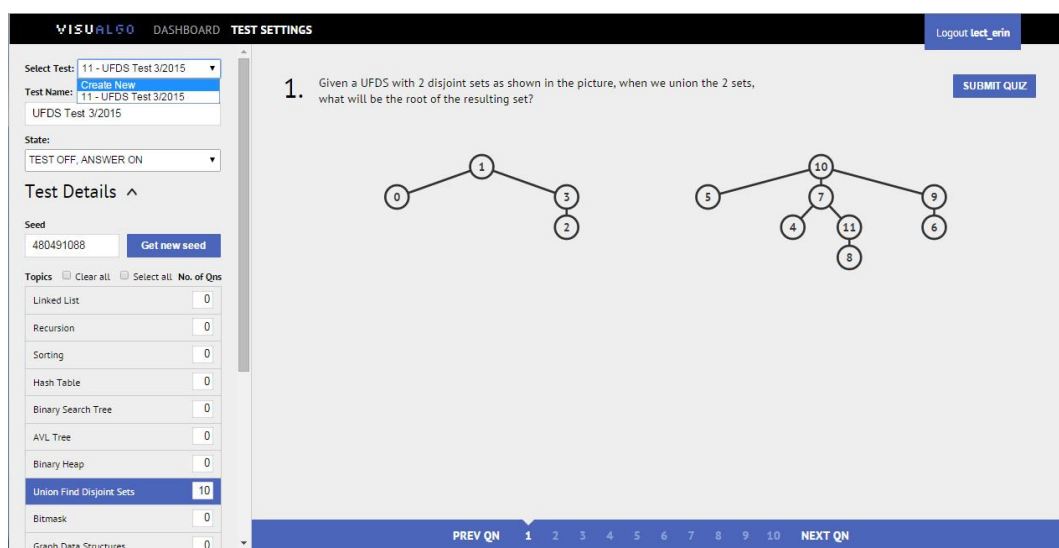


Figure 31: Test control panel for lecturers to create and modify tests (Screenshot of visualgo.net, 2015)

7.3 Making Test Feature Available to Other Professors

The database schema (shown in the figure below) was not designed to accommodate multiple professors and multiple tests. The test_config table only stores one test configuration. The user table stores student's information and the test table stores students' test scores and other test related information.

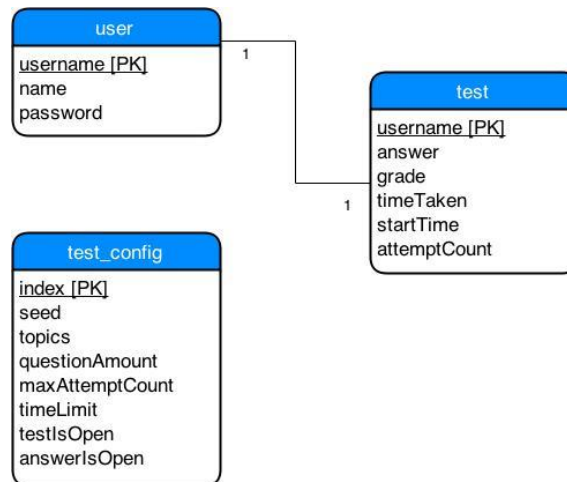


Figure 32: Previous database schema for test feature

To accommodate multiple professors and multiple tests, the schema was redesigned as such:

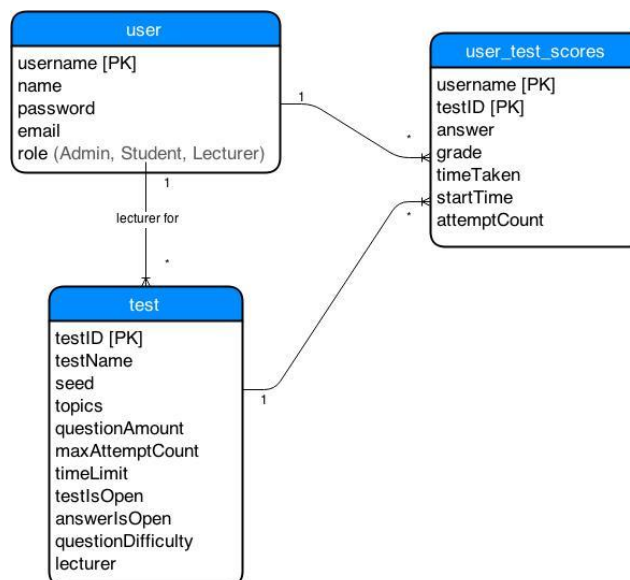


Figure 33: Database schema for test feature

Now professors who want to use VisuAlgo's test feature simply have to approach us to register a Lecturer account with us. Upon doing so, they may create as many tests as they wish.

7.3.1 Registering Students for Tests

Before, only one group of students, the students of CS2010 of that current semester, were allowed to attempt the test hosted by Dr. Steven Halim. To register students for a test, a lecturer has to upload a comma-separated values (CSV) file with the fields “username”, “name”, and “password” (the password is randomly generated by the lecturer). Thereafter, a lecturer would have to send an email to all students notifying them of their username and password for VisuAlgo. Students are not allowed to change their password and found it hard to remember the passwords generated for them.

Now, many different groups of students can take tests created by their respective lecturers. To register students for a test, similar to previously, a lecturer has to upload a comma-separated values (CSV) file. This time, the CSV file should contain an additional field: “email”. When the students are registered, they will automatically receive an email from VisuAlgo detailing their username, password, as well as where and how they can change their password.

7.4 Test Statistics

Users of VisuAlgo not only include students but teachers as well. As such, we want to provide teachers who use the tool, useful statistics about their students’ performance. Previously, the test feature had very limited statistics; only reporting the test scores of students and the overall average test score. As Dr. Halim is the only teacher using VisuAlgo to hold online quizzes thus far, he offered his views on the kind of statistics teachers (in general) would be interested in.

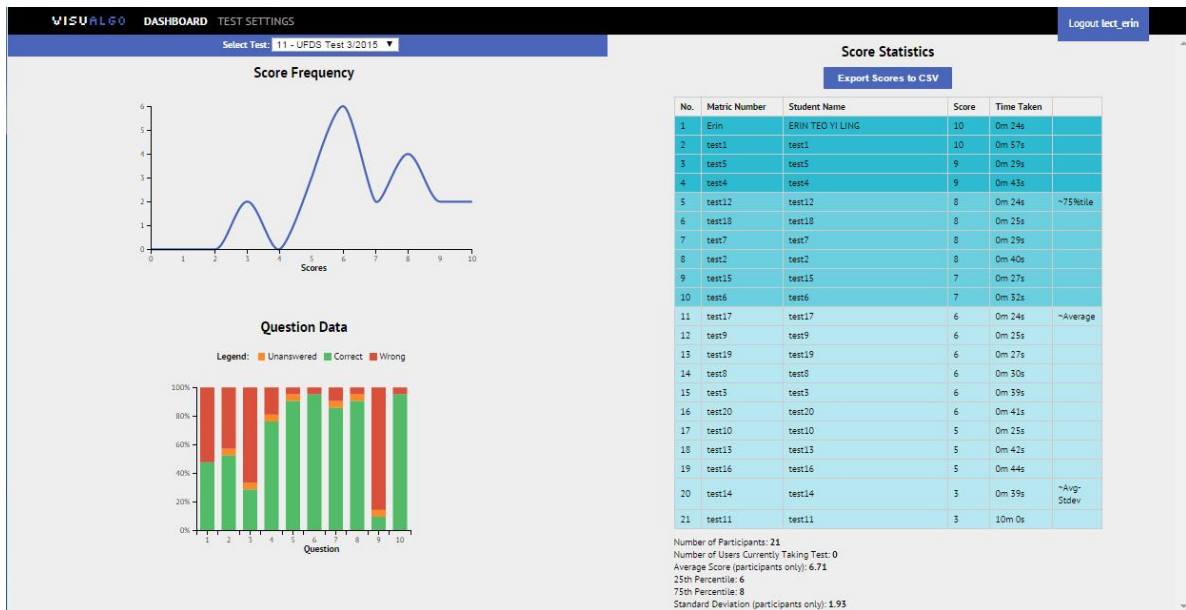


Figure 34: Dashboard for lecturers to view test statistics (Screenshot of visualgo.net, 2015)

The dashboard includes a score frequency curve that allows lecturers to have a quick overview of students' scores and performance on the test. Basic statistics such as the standard deviation of scores, and percentiles were added. The scoreboard was also made to be colour coded to categorize students whose scores fall in different percentiles. This is so that the teacher can easily identify the better/weaker students and take action based on that information.

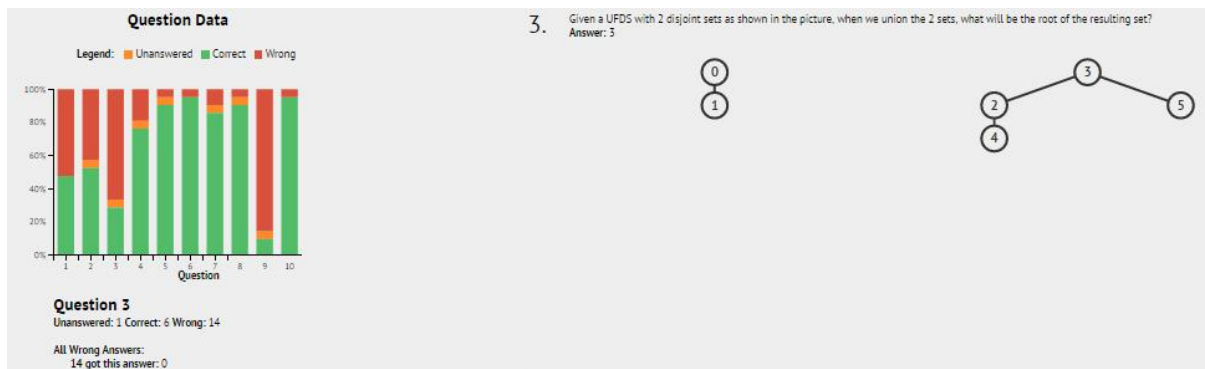


Figure 35: Question Data section in Dashboard (Screenshot of visualgo.net, 2015)

Detailed statistics on the students' performance on each question was recorded and is presented in the above graph (labelled Question Data). The graph gives teachers insights on the questions students found more difficult. In the figure above, we see that most students struggled with Question 3 and 9 in the test as majority of the students got these questions wrong. Clicking on a particular question gives the teacher more details such as how many students got the question wrong and the wrong answers that they submitted. It also displays

the question and model answer on the right side for the lecturer's reference. By studying the statistics provided here, teachers can pinpoint any learning gaps or misconceptions students may have and address them.

8 Conclusion

8.1 User Feedback

After CS2010 Semester 1 2014/2015 Online Quiz 1, a mid-semester survey was conducted to gather students' feedback on VisuAlgo and the module in general. The survey had high response rates; 116 out of 155 students taking CS2010 (75% of students) participated in the survey.

8.1.1 Usage of VisuAlgo

53.8% of students claim that they use the tool at least once a week. They often use the tool to revise topics after lectures, to clarify concepts they do not understand, or to practice questions before a quiz. With the online quiz, students had more motivation to learn using the tool. By looking at google analytics data, we see that during the week of the online quiz (Sep 14 2014 – Sep 18 2014) the usage of the tool spiked, there were a total of 2061 sessions in Singapore with 80.45% of sessions coming from returning users. These sessions lasted an average of 14 minutes and 25 seconds, which is much longer than the average session duration of the site (2 minutes and 32 seconds for August 2014).

8.1.2 Does VisuAlgo Enhance Learning?

Students unanimously felt that having VisuAlgo enhanced their learning. 56.1% of students thought that both the visualization and training component equally enhanced their learning and 36.6% of students felt that the visualizations were more useful for learning than the training. Below are a few of the select comments made by students who thought both the visualization and training components helped their learning:

1	“VisuAlgo has enabled better visualisation and mechanics of algorithms. Trainings are also very useful to help reinforce understanding as well as to give us a better idea of the coming quiz.”
2	“Compared to reading a whole chunk of words and trying to absorb and understand it, the visualization aid, pseudo-code and description is better and much easier to understand. The online quiz and training mode enhanced my understanding of the various topics. Questions like, “What is the minimum number of vertices in an

	AVL tree of height H?" (where the concepts are mentioned in lecture but I didn't give much thought about it) "forces" me to search for the answers and as a result it allowed me to remember it better.”
3	“I feel that algorithms are best understood when you can see them in action and VisuAlgo does that very well. It helped me understand algorithms much quicker than if it were just explained with words. The online quiz helped me to refine my understanding of the concepts that were taught to me. It also helped me to identify and rectify the areas that I was lacking in. ”

Figure 36: Selected comments from students about VisuAlgo

We can see that with both the visualization and training components, students’ learning process changes from linear to cyclical. In the linear learning process, students learn about an algorithm and continue to revise it before they sit for their test and after the test concludes, the student receives their grade.

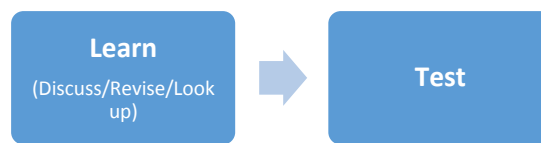


Figure 37: Linear Learning Process

When the students’ learning process is cyclical, students first learn about an algorithm from class and then test their understanding of it. Their training results will reveal any misconceptions or concepts they are unaware of. They can then go back to learning where they clarify their misconceptions and bridge any learning gaps before testing themselves again. This process encourages reflective thinking (Karavirta, Korhonen, & Malmi, 2005) and can provide strong support for learning.

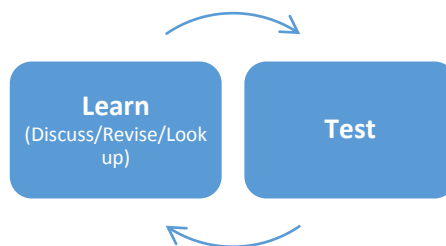


Figure 38: Cyclic Learning Process

Two trial online quizzes were held on 13 September 2014 and 17 September 2014 before the actual graded online quiz on 18 September 2014. During the period of Sep 14 2014 – Sep 18 2014, VisuAlgo was observed to be used much more than usual. Out of the 155 students taking CS2010, 46 attempted the two trial quizzes as well as the graded online quiz. Below are the average quiz scores for the two trial online quizzes and the graded online quiz for these 46 students.

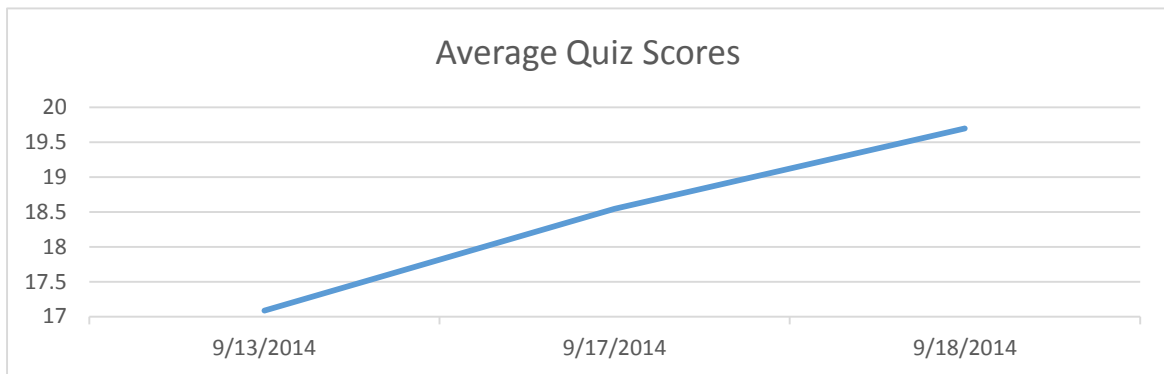


Figure 39: Average quiz scores of 46 of out of 155 students who took all three quizzes

Their individual quiz scores are shown in the following graph.

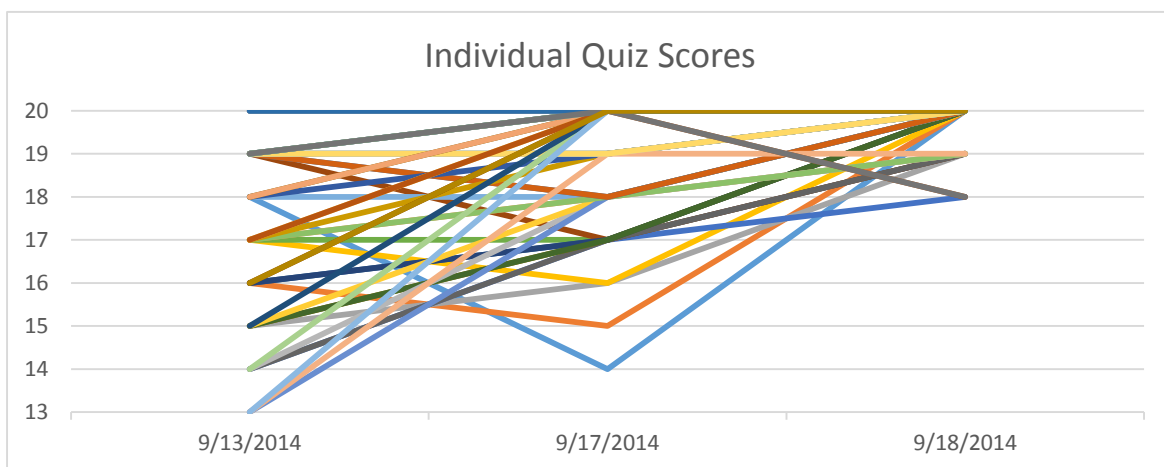


Figure 40: Quiz scores of 46 out of 155 students who took all three quizzes

From the above charts, we see that most students' scores experience a trend of increasing over time. There are some rare drops in scores among these 46 students, however, these are probably due to careless mistakes and not lack of understanding. This shows that as students continually use the tool over time, they gain a better understanding of concepts. This steady trend of improvement can be attributed to cyclical learning where students can better self-direct their learning by testing themselves and reaffirming concepts. Without the training component, this will not be possible.

As mentioned previously, VisuAlgo works best if it is used to support a data structures and algorithms course. Students can use VisuAlgo to direct their own learning and strengthen their understanding of basic concepts, and professors can then better spend their time introducing more interesting and challenging problems, and engaging and motivating students in learning. In order to fully utilize VisuAlgo for teaching a data structures and algorithms course, it is not enough to simply tell the students that such a tool exists, educators must integrate the tool in their teaching. It is also important that professors make practicing of questions with the training component mandatory. This can be done by setting up a graded quiz. The graded quiz can be of a very small percentage of students' grade ($\leq 5\%$). If not made mandatory, it is unlikely that students will be active in practicing questions (Saikkonen, Malmi, & Korhonen, 2001).

When educators introduce VisuAlgo's training component to students, they should reinforce that the questions presented in the training component are meant only to test basic concepts and are not representative of the questions that can be asked in written examinations. If not, the Dunning-Kruger effect where "unskilled individuals suffer from illusory superiority, mistakenly assessing their ability to be much higher than is accurate" (Kruger & Dunning, 1999) might be observed in students.

8.2 Recommendations for Future Work

Website Performance

VisuAlgo is not optimized for performance. The main page is especially graphic-heavy and takes quite a while to load especially on slower internet speeds. Many solutions may be applied to solve this problem such as the use of lazy loading, image compression, and sprites. The website also makes many calls to different CSS and JavaScript files. These CSS and JavaScript files may be combined and minified to produce a faster loading speed. There may be other redundancies or bottlenecks that may be identified and optimized to achieve a faster website performance.

Improve Quantity of Questions in Question Bank

The quantity of questions have improved tremendously over the course of this project. However, there are only a few questions on CS1020 (Data Structures and Algorithms) topics:

Linked List, Recursion, Sorting, and Hashing. There are also many visualizations that do not have a respective set of questions in the training component (e.g. Suffix Array/Tree).

Add More Advanced Visualizations

The visualization component has a wide range of visualizations spanning numerous topics. We can extend it further to provide an even more extensive library of visualizations that are not available anywhere else on the web. The following list of visualizations may be added:

1. Hashing – add separate chaining, or other various techniques
2. Linear Search and Binary Search
3. Shortest Path Faster Algorithm (SPFA)
4. Directed Minimum Spanning Tree (Chu–Liu/Edmonds' algorithm)
5. Many More

Better Testing Practices

At the moment, all testing is done manually. This is extremely inefficient and time-consuming to carry out. We want to put in place automated software testing by using automated testing tools to compare actual and expected behaviour and report any failures. The benefits of automated tests over manual testing is obvious. Automated tests can be run repeatedly without manual effort and will greatly speed up development time.

Security

NUS has recently update their Standard Operating Procedure (SOP) for Assessment and the security aspect of the current system is lacking. Security was not a priority for this project. While we did look into and fix some issues regarding SQL Injection and encryption of passwords, there are still security loopholes in the system that may be breached. In the future, we may look into issues such as:

1. Establishing an encrypted link between client and server (by obtaining an SSL certificate)
2. Backing up of database or obtaining a backup server
3. Other security issues that may have been overlooked such as session hijacking or insecure code

Increase Engagement and Motivation of Users

We can increase engagement of users using social tools. By improving the social aspect of VisuAlgo, we can start to establish a community that help each other to learn by discussion or support. To increase the engagement of VisuAlgo with students all across the globe, Wang Zi, a UROP student who is working under Dr. Steven Halim, is translating VisuAlgo to multiple languages to improve VisuAlgo's page rank on various search engines and search queries.

Gamification can be used to increase users' self-contributions (Huotari & Hamari, 2012). It has been applied to many areas to improve user engagement (Hamari, 2013). We can continue to explore how to apply gamification strategies to better engage users and provide them with a better learning experience.

References

- Battista, G. D., Eades, P., Tamassia, R., & Tollis, I. G. (1994). Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 235-282 .
- Brad, D., & Humphreys, C. (2001). Evaluation of Computer-Assisted Instruction in Principles of Economics. *Educational Technology & Society* 4.
- Brandenburg, F., Eppstein, D., Goodrich, M. T., Kobourov, S., Liotta, G., & Mutzel, P. (2004). Selected Open Problems in Graph Drawing. *Lecture Notes in Computer Science Volume 2912*, 515-539.
- Constantin, L. (14 Mar, 2014). *Researchers: Java's security problems unlikely to be resolved soon*. Retrieved from PCWorld:
<http://www.pcworld.com/article/2030778/researchers-javas-security-problems-unlikely-to-be-resolved-soon.html>
- Gigazine*. (Aug, 2014). Retrieved from Gigazine: <http://gigazine.net/news/20140819-visualgo/>
- Glennon, Goodchild, M. F., & Alan, J. (2010). Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth*, 231-241.
- Hacker News*. (2014). Retrieved from Hacker News:
<https://news.ycombinator.com/item?id=8194662>
- Halim, S., & Halim, F. (2013). *Competitive Programming 3 The New Lower Bound of Programming Contests*. Singapore: Lulu.
- Halim, S., Koh, Z. C., Loh, V., & Halim, F. (2012). *Learning Algorithms with Unified and Interactive Web-Based Visualization*.
- Hamari, J. (2013). Transforming Homo Economicus into Homo Ludens: A Field Experiment on Gamification in a Utilitarian Peer-To-Peer Trading Service. *Electronic Commerce Research and Applications* 12, 236-245.
- Hao, S., Jia, D., & Li, F.-F. (2008). *Crowdsourcing Annotations for Visual Object Detection*.
- Huotari, K., & Hamari, J. (2012). Defining Gamification - A Service Marketing Perspective. *Proceedings of the 16th International Academic MindTrek Conference 2012*, 3-5.
- Karacapilidis, N. (2014). Solutions and Innovations in Web-Based Technologies for Augmented Learning: Improved Platforms, Tools, and Applications.
- Karavirta, V., Korhonen, A., & Malmi, L. (2005). Different Learners Need Different Resubmission Policies in Automatic Assessment Systems. *Proceedings of the 5th Annual Finnish / Baltic Sea Conference on Computer Science Education*, (pp. 95–102).

- Korhonen, A., Malmi, L., & Silvasti, P. (2003). TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. *Proceedings of Kolin Kolistelut / Koli Calling – Third Annual Baltic Conference on Computer Science Education*, 48–56.
- Korhonen, A., Malmi, L., Myllyselkä, P., & Scheinin, P. (2002). Does it Make a Difference if Students Exercise on the Web. *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'02*, 121–124.
- Kruger, J., & Dunning, D. (1999). Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments. *Journal of Personality and Social Psychology*. Retrieved 2015, from Wikipedia: http://en.wikipedia.org/wiki/Dunning%E2%80%93Kruger_effect
- Laakso, M.-J., Salakoski, T., & Korhonen, A. (2005). The Feasibility of Automatic Assessment and Feedback. *Proceedings of Cognition and Exploratory Learning in Digital Age (CELDA 2005)*, 113–122.
- Laakso, M.-J., Salakoski, T., Korhonen, A., & Malmi, L. (2004). Automatic Assessment of Exercises for Algorithms and Data Structures – a Case Study with TRAKLA2. *Proceedings of Kolin Kolistelut / Koli Calling – Fourth Finnish/Baltic Sea Conference on Computer Science Education*, 28–36.
- Parr, C. (2013). *Mooc completion rates 'below 7%'*. Retrieved from Times Higher Education: <http://www.timeshighereducation.co.uk/news/mooc-completion-rates-below-7/2003710.article>
- reddit*. (19 Aug, 2014). Retrieved from reddit: http://www.reddit.com/r/learnprogramming/comments/2dznos/visualgo_animated_data_structure_and_algorithm/
- Saikkonen, R., Malmi, L., & Korhonen, A. (2001). Fully Automatic Assessment of Programming Exercises. *Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'01*, 133–136.
- TRAKLA2 Research Site*. (n.d.). Retrieved from TRAKLA2 Research Site: <http://www.cse.hut.fi/en/research/SVG/TRAKLA2/>

Appendix A: List of Questions Added

Linked List

1. Suppose `n` points to a node in a linked list. What expression will be true when `n` points to the tail node?
2. In the pseudocode program below, `list` is a Singly Linked List. The function `populateList()` adds the integers `[elements]` to `list` sequentially. What is the output of the program? Select 'No Answer' if the program results in an error.
`populateList();`
`int sum = 0;`
`Node n = list.head;`
`|program|`
`print sum;`
3. Consider the singly linked list below. Which of the following options represent the resulting linked list if you insert `|element|` to the `|position|` of the linked list?
4. The singly linked list below represents a stack `st` where `st.peek() = |firstValue|`. Suppose the following operations are performed:
`|operations|`
What is the output of `st.peek()`? Select 'No Answer' if the program results in an error.
5. The singly linked list below represents a queue `q` where `q.peek() = |firstValue|`. Suppose the following operations are performed:
`|operations|`
What is the output of `q.peek()`? Select 'No Answer' if the program results in an error.

Recursion

1. Consider the following pseudocode:
`testFunction(n) {`
 `if (n == 0) return 0;`
 `else if (n == 1) return 1;`
 `else return testFunction(n - 1) + testFunction(n - 2);`
`}`
What is the output of `testFunction(|amt|)`?

2. What is the time complexity of the following pseudocode?

```
int doSomethingCool(int n) {  
    if (n < 3) return 0;  
    int num = 5;  
    for (int j=0; j<n+1; j|firstLoopIncrement)  
        num += 1;<br>  
    for (int k=n; k>0; k|secondLoopIncrement)  
        num += 1;  
    return doSomethingCool(|rec|) + num;  
}
```

Sorting

1. Which of these sorting algorithms are guaranteed to run in $n \log n$?
2. An array of $|amt|$ integers is being sorted using Quicksort. Suppose the algorithm has just finished the first partitioning and pivot swapping thus changing the content of the original array to the following: $[array]$
From the above resulting array, how many integers could have been the pivot? Note: elements $==$ pivot are partitioned to the right.
3. How many comparisons are required to sort an array of $|amt|$ integers: $[array]$ using Bubble Sort?
4. How many comparisons are required to sort an array of $|amt|$ integers: $[array]$ using Selection Sort?
5. How many comparisons are required to sort an array of $|amt|$ integers: $[array]$ using Insertion Sort?
6. How many [comparisons/passes] are required to sort an array of $|amt|$ integers: $[array]$ using Improved/Enhanced Bubble Sort (Bubble Sort terminates when there are no more swaps during the pass)?
7. Click the sequence of integers that represent the content of the array $[array]$ after $|passes|$ passes of Insertion Sort.
8. Click the sequence of integers that represent the content of the array $[array]$ after $|passes|$ passes of Bubble Sort.
9. Click the sequence of integers that represent the content of the array $[array]$ after $|passes|$ passes of Selection Sort.

10. How many swaps are required to sort an array of $|amt|$ integers: $[array]$ using Bubble Sort?

Hash Table

1. Suppose you have a linear probe hash table of size $|amt|$ with the hash function $k \% |amt|$. Insert the keys: $|sequence|$ into the table (in order). Is this the correct resulting hash table after insertion: $|table|$?
2. Suppose you have a quadratic probe hash table of size $|amt|$ with the hash function $k \% |amt|$. Insert the keys $|sequence|$ into the table (in order). Is this the correct resulting hash table after insertion: $|table|$?
3. Suppose you have a hash table of size $|amt|$ with the hash function $k \% |amt|$ and secondary hash function $|secHash| - k \% |secHash|$ to resolve collisions. Insert the keys: $|sequence|$ into the table (in order). Is this the correct resulting hash table after insertion: $|table|$?
4. Consider a linear probe hash table of length $m = |amt|$ with the hash function: $h(key) = key \bmod m$. Click the sequence for inserting keys $|keys|$ such that the resulting content of the hash table is:
 $|table|$
5. Suppose you have a linear probe hash table of size $|amt|$ with the hash function $k \% |amt|$. Click the sequence of integers that represent the probe sequence if you delete $|element|$ from the following hash table: $|table|$
6. Suppose you have a quadratic probe hash table of size $|amt|$ with the hash function $k \% |amt|$. Click the sequence of integers that represent the probe sequence if you delete $|element|$ from the following hash table: $|table|$
7. Suppose you have a hash table of size $|amt|$ with the hash function $k \% |amt|$ and secondary hash function $|secHash| - k \% |secHash|$. Click the sequence of integers that represent the probe sequence if you delete $|element|$ from the following hash table:
 $|table|$

Binary Heap

1. What is the $[minimum/maximum]$ number of comparisons between heap elements required to construct a max heap of $|amt|$ elements using the $O(n)$ BuildHeap(array)?

2. What is the maximum number of swaps between heap elements required to construct a max heap of $|amt|$ elements using the $O(n)$ BuildHeap(array)?
3. True or False: The [smallest/second smallest] element in a Binary Max Heap that contains > 3 distinct integers is always at one of the leaves.
4. True or False: The [second largest/third largest] element in a Binary Max Heap that contains > 3 distinct integers is always one of the children of the root?
5. True or False: An array A of n distinct integers that are sorted in descending order forms a valid Binary Max Heap. Assume that A[0] is not used and the array values occupy index [1:n].
6. True or False: Given a Binary Max Heap, calling ShiftDown(i) $\forall i > \text{heapsize}/2$ will never change anything in the Binary Max Heap.

Binary Search Tree

1. True or False: The smallest element in any non-empty BST always has [no left child/no right child/a successor/no predecessor].
2. True or False: The largest element in any non-empty BST always has [no right child/a parent].
3. Suppose that we have all distinct integers inside a BST of unknown structure and we want to search for the integer |value|. True or False: It is possible to have a search sequence as follows: |sequence|.
4. How many structurally different BSTs can you form with $|amt|$ distinct elements?
5. True or False: The insert operation in BST is always not commutative in the sense that inserting two distinct elements x and then y into an existing BST (not necessarily balanced) always produce structurally different BST as inserting y and then x.
6. True or False: The delete operation in BST is always commutative in the sense that deleting x and then y from an existing BST (not necessarily balanced) always produces structurally the same BST as deleting y and then x. Note that $x \neq y$ and both x and y exist in the BST.

AVL Tree

1. What is the [minimum/maximum] possible height of a BST with $|amt|$ elements?
2. What is the minimum number of vertices in an AVL tree of height $|amt|$?

3. What is the maximum number of vertices in an AVL tree of height $|amt|$?
4. An integer $|value|$ is going to be inserted into the AVL tree below. Will the insertion result in any rotations?
5. Will the deletion of $|value|$ in the following AVL tree result in any rotations?
6. What is the balance factor of the vertex $|value|$? (Note that the given BST is not necessarily balanced)

Union Find Disjoint Sets

1. Given a UFDS with 2 disjoint sets as shown in the picture, when we union the 2 sets, what will be the root of the resulting set?
2. Given $n = |amt|$ disjoint sets initially in a UFDS, is it possible to call `unionSet(i, j)` and/or `findSet(i)` operations to get a single tree with actual height $|height|$ that represents a certain set? Both path compression and union by rank heuristics are used.
3. Given the UFDS as shown in the picture, how many disjoint sets are there?

Graph Data Structures

1. What is the [in/out] degree of vertex $|value|$ in the graph?
2. How many entries are there in the Edge List for this graph?
3. Select the statement(s) that are true for the graph below.
4. The following are representations of entries in an Adjacency List. Select the entry(ies) that belong to the Adjacency List of the graph below.
5. Suppose this graph is stored in an Adjacency Matrix 'adjMat'. What is the value of `adjMat[|start|][|end|]`? Note: if a cell is empty, its value will be zero.
6. What is the sum of the degrees of all the vertices in this graph?
7. Which graph data structure(s) should you use to store a graph with 10000 vertices and 10000 edges? Suppose your computer only has enough memory to store 50000 entries.
8. Which graph data structure(s) should you use to store a graph with 100 vertices and the existence of `edge(u, v)` is frequently asked?
9. Which graph data structure(s) should you use to store a graph with 200 vertices, 19900 edges, and the neighbours are frequently enumerated? Suppose your computer only has enough memory to store 50000 entries.

10. Which graph data structure(s) should you use to store a graph with 200 vertices, 19900 edges, and the edges need to be sorted? Suppose your computer only has enough memory to store 50000 entries.
11. Draw a complete graph with |amt| vertices.
12. Draw a tree with |amt| vertices.
13. Draw a directed acyclic graph with |amtVertices| vertices and |amtEdges| edges.

Graph Traversal

1. Is the graph in the picture a bipartite graph?
2. Is the graph in the picture a tree?
3. Click the sequence of vertices that result in a valid topological sort of the graph. If the graph has no valid topological sort, select 'No Answer'.
4. Click all the edges (in the order that they are processed) that make up the spanning tree induced by [DFS/BFS] starting from source vertex |value| for this graph. The neighbours of a vertex are listed in ascending vertex number.
5. Consider the following partial pseudocode for DFS: |algorithm|
Suppose this code is run on the following BST starting from the root, click the sequence of vertices in the order that they are printed. Assume that the left child is always visited before the right child.
6. Select the option(s) that represent a valid simple path for the graph below.
7. In the following directed acyclic graph, how many simple paths are there from vertex |from| to |to|?
8. Suppose vertex |value| is removed from the following graph. How many components are there in the resultant graph?
9. Is |sequence| a valid topological sort of the graph?
10. How many different spanning trees are there in a complete graph with |amt| vertices?
11. Click all the edges that must belong to every spanning tree of the graph shown below.
12. DFS and BFS |subtype| run in |complexity| on:
13. Draw a simple connected graph with |amtVertices| vertices and |amtEdges| edges such that the graph contains |criteria| [bridges/cut vertices].
14. Draw a simple bipartite graph with |amtVertices| vertices and |amtEdges| edges.
15. Draw a simple directed graph with |criteria| strongly connected components using |amtVertices| vertices and |amtEdges| directed edges. Your graph must be drawn such

that by replacing all of its directed edges with undirected edges, a connected undirected graph is produced.

Minimum Spanning Tree

1. Given the undirected weighted graph as shown in the picture, click on all the edges (in any order) that make up the Second Best Minimum Spanning Tree. If such a Spanning Tree does not exist, select 'No Answer'.
2. The following undirected weighted graph only has one unique Minimum Spanning Tree. True or False?
3. The following undirected weighted graph represents the road network of Country A. Vertices represent cities, edges represent roads, and edge weights represent the cost of maintaining a road. To reduce the cost of maintenance, Country A has decided to demolish some roads such that the resulting road network still connects all cities but has the MINimum maintenance cost. Note that roads |edges| cannot be demolished because they are of historical importance. Click on all the roads (in any order) that make up the new road network.
4. Select the edges (in any order) that form |amt| connected components and the total weight of these components is MINimum.
5. Select the edges (in any order) that do not belong to any [MINimum/MAXimum] Spanning Tree of this graph. If no such edge exists, select 'No Answer'.
6. Draw a simple connected weighted undirected graph with |amtEdges| edges and |amtVertices| vertices so that the optimized Kruskal's algorithm have to examine all |amtEdges| edges before stopping with the MST. The weights of the edges have to be distinct.

Single-Source Shortest Paths

1. Click the sequence of vertices such that when all outgoing edges of these vertices are relaxed in this order using Bellman-Ford algorithm, the SSSP problem can be solved in $O(V+E)$ time.
2. Run BFS on the following graph from source vertex |source|. Does it produce the correct shortest path values at all vertices reachable from the source vertex?

3. Run Modified Dijkstra's Algorithm on the following graph on all vertices. Does it produce the correct shortest paths for each vertex?

4. Given the shortened pseudocode for Dijkstra's algorithm:

```
priority_queue pq; pq.enqueue(D[s], s);
while (pq is not empty) {
    (d, u) = pq.dequeue();
    if (d == D[u]) {
        for each vertex v adjacent to u { relax (u, v) and add v to pq if relaxed }}}
```

If the Priority Queue highlighted above was changed to a [Queue/Linked List/Stack/Heap/BST/Hash Table], would Dijkstra's algorithm still work for the following type of graph?

5. After $|amt|$ passes of the Bellman-Ford Algorithm on the following graph from source vertex $|source|$, what is the value of $D[|value|]$? Suppose the edges are relaxed in the order: $|edges|$. If $D[|value|]$ is INFINITY, select 'No Answer'. Note that all edge weights are printed closer to the arrowheads of the respective arrows.

6. Run [Modified Dijkstra's Algorithm/Original Dijkstra's algorithm] and Bellman-Ford Algorithm from source vertex $|value|$ to solve the SSSP problem for this graph and compare the two algorithms. Which one of the following statements is true?

7. Draw a simple connected weighted directed graph with $|amtVertices|$ vertices and at most $|amtEdges|$ edges such that running Modified Dijkstra's algorithm from source vertex 0 relaxes at least $|threshold|$ edges to get the correct shortest paths. Your graph cannot contain a negative cycle.

8. Draw a simple connected weighted directed graph with $|amtVertices|$ vertices such that running Optimized Bellman-Ford algorithm with source vertex 0 goes through at least $|threshold|$ passes to get the correct shortest paths. Your graph cannot contain a negative cycle. Note that all edges are processed according to the adjacency list i.e. all outgoing edges from vertex 0 is processed then edges from vertex 1 and so on.

9. Draw a simple connected weighted directed graph with $|amtVertices|$ vertices and $|amtEdges|$ directed edges such that running Modified Dijkstra's from source vertex 0 makes the algorithm run indefinitely. The weights of the edges have to be distinct.