

2008

## Investigating the Effectiveness of Active Interaction Tools on Student Learning

Lurie Andrei  
*San Jose State University*

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_projects](https://scholarworks.sjsu.edu/etd_projects)



Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Andrei, Lurie, "Investigating the Effectiveness of Active Interaction Tools on Student Learning" (2008).  
*Master's Projects*. 80.

DOI: <https://doi.org/10.31979/etd.qhcz-5vus>  
[https://scholarworks.sjsu.edu/etd\\_projects/80](https://scholarworks.sjsu.edu/etd_projects/80)

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact [scholarworks@sjsu.edu](mailto:scholarworks@sjsu.edu).

**INVESTIGATING THE EFFECTIVENESS OF ACTIVE INTERACTION TOOLS  
ON STUDENT LEARNING**

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By Andrei Lurie

August 2008

© 2008

Andrei Lurie

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

---

Dr. David Taylor

---

Dr. Cay Horstmann

---

Mr. Menko Johnson

APPROVED FOR THE UNIVERSITY

---

## ABSTRACT

In this project, we investigate the effectiveness of active interaction animation tools for learning. We limit our scope to a particular computer science course that teaches graph algorithms on an undergraduate level. More specifically, we evaluate student understanding of basic graph algorithms when two kinds of interactive animation tools are used by the students to learn the algorithms: *active* interaction and *passive* interaction. We hypothesize that animations which engage students in active interaction are more effective and more beneficial to learning and comprehension than the animations which do not explicitly engage students in active interaction. We conduct an experiment and study the effects of these two kinds of interactive animation on learning effectiveness.

## **ACKNOWLEDGEMENTS**

I would like to thank my advisor, Dr. David Taylor, for his guidance, invaluable insight, and bright ideas, as well as for spending countless hours of his time reviewing drafts of this report.

I would like to thank my committee members Dr. Cay Horstmann and Mr. Menko Johnson for their time reviewing this report, participating in the defense, and for their valuable comments.

I would like to thank my parents, Galina and Ted Lurie, for their encouragement and support.

And most of all, I would like to thank my lovely wife, Nicole, for her patience, motivation, and for sacrificing our family time, allowing me to spend time on this project.

## TABLE OF CONTENTS

1	Introduction.....	1
1.1	Background and definitions.....	1
1.2	Prior work.....	3
1.3	Project proposal.....	5
2	Experiment design.....	7
3	Experiment setup.....	11
3.1	Interactive Tools.....	12
3.1.1	Tool1.....	13
3.1.2	Tool2.....	15
3.2	Animation Tools.....	18
3.2.1	Animation tool1.....	19
3.2.2	Animation tool2.....	20
3.2.3	Animation tool3.....	21
3.3	Moodle.....	22
3.4	Exercises.....	26
3.4.1	Breadth-First Search.....	29
3.4.2	Depth-First Search.....	31
3.4.3	Kruskal's Algorithm.....	32
3.4.4	Prim's Algorithm.....	34
3.4.5	Bellman-Ford Algorithm.....	35
3.4.6	Dijkstra's Algorithm.....	37
4	Results and Analysis.....	39
5	Discussion.....	48
5.1	Exercise 1 (Breadth-First Search).....	49
5.2	Exercise 2 (Depth-First Search).....	49
5.3	Exercise 3 (Kruskal's algorithm).....	49
5.4	Exercise 4 (Prim's algorithm).....	49
5.5	Exercise 5 (Bellman-Ford algorithm).....	50
5.6	Exercise 6 (Dijkstra's algorithm).....	50
5.7	Combined Exercises.....	50
6	Limitations of the study.....	52
7	Conclusion and Future work.....	53
8	References.....	54
	Appendix A: Example of the exercise flow.....	56
	Appendix B: Student consent form.....	66
	Appendix C: Experiment data.....	69
	Appendix D: Survey.....	101
	Appendix E: Adding exercises to Moodle.....	108
	Appendix F: Design reference for exercise support in Moodle.....	113

## LIST OF TABLES, LISTINGS, AND FIGURES

### Figures

Figure 1: Instance of the Bellman-Ford algorithm problem.....	14
Figure 2: Completed instance of the Bellman-Ford algorithm problem.....	15
Figure 3: Instance of the Kruskal's algorithm problem.....	17
Figure 4: Grading mode for the Kruskal's algorithm problem instance.....	18
Figure 5: Thomas Wolf's animation applet.....	20
Figure 6: Mustafa Incel's animation applet.....	21
Figure 7: Animation applet by Rensselaer Polytechnic Institute.....	22
Figure 8: CS146 course main page with six exercises.....	35
Figure 9: Detailed exercise report for a student.....	36
Figure 10: state diagram for the exercise flow.....	40
Figure 11: Selecting vertices to dequeue/enqueue.....	41
Figure 12: Selecting edges for the output vertices.....	42
Figure 13: DFS problem in interactive tool2.....	43
Figure 14: Kruskal's problem in interactive tool2.....	44
Figure 15: Prim's algorithm in interactive tool2.....	46
Figure 16: Bellman-Ford algorithm in interactive tool1.....	48
Figure 17: Dijkstra's algorithm in interactive tool2.....	49
Figure 18: Data and the histogram for Prim's exercise, Question 1.....	52
Figure 19: Data and the histogram for Prim's exercise, Question 2.....	52
Figure 20: Data and the histogram for Prim's exercise, Question 3.....	53
Figure 21: Statistics for Prim's algorithm exercise.....	53
Figure 22: Normalized version of data in Figure 21.....	54
Figure 23: Statistics for all experiments combined.....	54
Figure 24: Statistics for time spent on phase2 of the Prim's exercise.....	55
Figure 25: Score histogram for all six exercises combined.....	63

### Tables

Table 1: Experiment design factors.....	9
Table 2: Result of the t-test.....	44
Table 3: Result of the "possible improvement" statistic.....	45
Table 4: Result of the "possible improvement" statistic.....	45
Table 5: Result of the correlation test.....	47
Table 6: Statistics summary.....	48



# 1 Introduction

## 1.1 Background and definitions

Visualization is often employed by educators to aid students in understanding nontrivial concepts. The old saying that a picture is worth a thousand words is especially pertinent to the study of abstract subjects such as computer science. In particular, the study of graph algorithms involves several abstract concepts, and it would be next to impossible to teach and fully comprehend such concepts without utilizing some sort of visualization.

It is generally accepted that graphics can facilitate comprehension, learning, memory, and inference [1], and it is clear that it is more beneficial to the student when a verbal explanation of some abstract concept is complemented by a visual presentation. However, there is no consensus on what types of visualizations really improve insight and facilitate learning.

With the advance of technology, the use of visualization in academia evolved from drawings on the chalk board to overhead transparencies, to static presentation graphics, to animated presentation graphics, and most recently, to the use of the interactive tools that allow students to see the demonstration of the concept at their own pace. Unfortunately, over the years, much more research effort has been spent on the technological aspects of software visualization (especially algorithm animation) than on the pedagogical effects of such systems [2]. This resulted in the disconnect between technical advances in software visualization and the understanding on how this technology can and should be applied to maximize the learning effectiveness. For instance, there is a plethora of algorithm visualization tools available to educators today (see report by [3]), but it is not known which of these tools enhance student learning more effectively. Furthermore, it is not clear how to improve these tools so that student learning is maximized. To this end, several studies have been done in an attempt to show the effectiveness (or ineffectiveness) of certain visualization types [1, 2, 6, 12, 13] (see Section 1.2 for the discussion of prior work).

Animation has been one of the popular targets of such studies; several research attempts have compared animations to static graphics theorizing that animated graphics would be more efficient for learning [7, 8, 9, 10, 11]. However, results of these studies are mixed. Some did not discover any noticeable improvement [10], and the ones that did were often not making an accurate comparison [11], because the animations involved interaction or contained more information than did static graphics. In other words, the animation aspect was not well isolated and was combined with some other factors that could have influenced the outcome of the study. Because of this, it is uncertain whether animation by itself contributes to learning effectiveness, or whether there are some other factors (perhaps directly or indirectly linked to animation) that influence the learning

effectiveness.

Furthermore, most animations have evolved into tools that allow the user to interact with them by: pausing the animation, controlling the the animation speed, progressing through the animation one step at a time. Some tools even provide a way for the user to change how animation is presented. For instance, user can: zoom in and out, pick alternative views, move or modify displayed objects, etc. We refer to such type of visualization tools as interactive animations, and it is on this kind of tools that we find it interesting to focus our research.

In this project we study how effective the interactive animation tools are on student comprehension. For the purpose of our study, we classify the interactive animation tools into two categories:

- **passive interaction animation tools** (hereafter “animations” or “animation tools”)

These are the animation tools that do not just display the animation of the subject matter, but allow user to control how the animation is presented and carried out. For instance, the tool may allow user to pause the animation, control the speed at which the animation occurs, zoom in and out of the view, rearrange animated objects. Such tools are interactive by definition, since they, indeed, provide some user interface. However, they allow user to control only **how** the animation is represented and/or the pace with which the animation occurs. The tools still have full control over which objects are being animated and over the sequence in which animation occurs. In classifying the interactive animation tools of this nature, we chose the term *passive* to reflect the reality of how these tools are being used by the students. The tools are essentially a one way information delivery, similar to a movie, where students have very limited intellectual engagement and simply sit back and watch the presented information. An overwhelming majority of the algorithm visualization tools fall under this category.

- **active interaction animation tools** (hereafter “interactive animation” or “interactive tools”)

These animation tools take interaction to the next level. In addition to providing the same controls as passive interaction tools, they allow user to control **what** objects are being animated and they allow user to conduct the animation (i.e. control the sequence in which animation occurs). In classifying the interactive animation tools of this nature, we chose the term *active* to reflect the main characteristic of these tools that enables active participation from the user. The tools transform the user from a mere observer to an active participant.

## 1.2 Prior work

Significant amount of research has been done on software visualization, and even more effort has been spent on creation of visualization tools. For instance, at the time of writing this report, there were close to 400 cataloged visualization tools just for algorithm animation [5]. Unfortunately, pedagogical aspects of software visualization has not received nearly as much attention. Much less effort has been spent in assessing the algorithm animation tools and furthering our understanding of what makes them more effective, despite the fact that algorithms comprise an essential part of computer science. Perhaps, this could be attributed to the intuition that animations, being dynamic, must have some natural pedagogical benefits when they are applied to learning concepts and processes that change with time (such as algorithms), and therefore, not as much research has been done on what seemed to be an obvious matter. Nevertheless, a number of research studies (notably Stasko, et al. [2, 6, 13]) have been carried out that aimed at understanding the pedagogical aspects of algorithm animation tools. Results of those studies are mixed and demonstrate that the subject matter is more complicated than it appears.

Some early studies looked at how effective the static graphics are when compared to learning from text. The results failed to identify a clear advantage of graphics over the text [7, 8, 9, 1]. This shows that despite the intuition, graphics may not be suitable in certain situations, and that only carefully designed graphics could benefit learning.

Later studies concentrated on learning effectiveness of animations, however, the results were mixed. For example, [10] analyzed performance of the post-test for two groups of students: one that learned the algorithm from textual form only and the other that learned the algorithm from textual form and the animation of the algorithm. The study found no significant difference in the post-test performance of the two groups. The study from [11] found a significant benefit of animations compared to static graphics. However, the animations in this study contained critical information about the concept which static graphics did not have. Therefore the benefit could have come from this additional information rather than the animation itself [1].

Extensive work in the area of algorithm animation effectiveness has been done by Stasko et al. [2, 6, 13]. In [13] the authors found that students who used the animation tool interactively by creating their own graphs for the animation tool performed slightly better on the post-tests than the students who passively watched the animation.

Of particular interest to us was work done in [2]. In that experiment, students were divided into four groups associated with the type of the learning tools available to the students:

- no-animation/no-prediction  
Students were presented with the text description of an algorithm and were given 10 minutes to study it.
- animation/no-prediction  
Students were presented with the text description of an algorithm and were given 5 minutes to study it. Then spent 5 minutes watching the algorithm animation.
- no-animation/prediction  
Students were presented with the text description of an algorithm and were given 5 minutes to study it. Then spent 5 minutes making predictions on the static graphs with immediate feedback from the experimenter.
- animation/prediction  
Students were presented with the text description of an algorithm and were given 5 minutes to study it. Then spent 5 minutes making explicit predictions while watching the algorithm animation.

A post-test was performed to test students' knowledge of the given algorithm. The goal of the study was to examine whether animations of algorithms would help students learn the algorithms more effectively. However, the results showed that students in the no-animation/prediction group learned algorithms as effectively as the students in the animation/no-prediction group, and even as effectively as the students in the animation/prediction group. This suggests that prediction is the key factor in learning effectiveness. This aligns well with our belief in the effectiveness of active interaction animation tools, since animation tools that provide user interface for prediction fall into our definition of active interaction animation tools (see Section 1.1. Background and definitions).

The results of [2] also showed that students in the no-animation/no-prediction group did not learn as effectively as the students in the other three groups, and that there were no significant differences in learning effectiveness for the students in those three groups. The authors hypothesize that perhaps the animation encourages learners to predict the algorithm's behavior, and therefore, helps to increase the learning effectiveness. This is a plausible argument, yet curious and apt learners may be encouraged to predict the algorithm's behavior even without using animation. Furthermore, not all students are motivated learners, and therefore, animation by itself might not encourage them to actively participate in the learning process. Nevertheless, the observations made in [2] are important and may help to create more effective animations (for instance, designing animation in the way that would encourage prediction).

Another relevant work to our study is [12], where students were divided into two groups:

- Students using the interactive animation tool  
The interactive aspects of the tool were “Show me” mode and “I’ll Try” mode. With “Show me” mode students were able to see the solution, and with “I’ll Try” mode, students were able to predict the solution.
- Students using the non-interactive animation tool

Result of the study was surprising. The group that was using the interactive animation tool performed slightly worse on the post-test than the group that was using the non-interactive animation tool. Authors suggest that, most likely, the students using the interactive tool were treating the tool as a video game or engaged into a guessing game. In other words, they were not utilizing the tool effectively. The authors did not make it clear, however, how many students who used their interactive tool were actually making use of the “I’ll try” mode and to what degree (i.e. extensively or just a few times).

### **1.3 Project proposal**

The main goal of this project is to study the learning effectiveness of active interaction animations. A number of animation studies have been conducted in the past. The common goal of all such studies was to examine the pedagogical effects of software visualization and to better understand the key aspects of creating effective tools for learning. However the studies tended to concentrate on comparing animations to static graphics or attempted to demonstrate that animations alone increase learning effectiveness. Unfortunately, the results of the majority of those studies were mixed and did not support the hypothesis with clear certainty. The most pertinent work to our project would be [2] (see Section 1.2. Prior Work). Its result suggests that active interaction (referred to as “prediction” in their paper) is a key component in learning effectiveness. However, it further theorized that animations without active interaction also aid learning because they encourage active interaction. In other words, the paper seems to suggest that animation itself has an implicit active interaction aspect. For example, the paper noted that several students voluntarily chose to predict the algorithm behavior while using the animation. However, it could be argued that student curiosity and passion for learning encourages prediction, and the animation is only giving an outlet for those predictions rather than actively encourages them. Students that are not very excited about the algorithm or the tool are less likely to engage in such active interaction and would passively watch the animation take place. This is further supported by the result of our survey (see Appendix D, question 6). We argue that an effective learning tool would have to offer more benefit to all students, no matter how enthusiastic they are about the subject.

Because interaction has been shown to facilitate learning and improve comprehension [4, 1], and making predictions (i.e. actively interacting) was

identified as being the key to effective learning [2], it would seem that animations that require students to actively interact should be more effective for learning than the animations that don't require active interaction. This is the hypothesis we would like to test in this project. In other words, we conduct an experiment to study the learning effectiveness of the interactive tools (i.e. active interaction) and compare it to the learning effectiveness of the animations (i.e. passive interaction). We theorize that interactive tools will have greater learning effectiveness. Even though we are not the first to utilize interactive tools and animations in one study, to the best of our knowledge, our formulation of the hypothesis (i.e. That animations that require active participation from the student are more effective than animation that do not) is slightly different from the ones in prior studies. We also tried to design an accurate study given the limited number of students available for our pilot program. For each exercise, students will use both an active interaction tool and an animation, though the order is chosen at random. Student will answer topic questions before using either tool, after the first tool, and after both tools. Over six exercises (each exercise covering one algorithm), we use three different animation tools and two versions of an interactive tool being developed at SJSU.

## 2 Experiment design

The experiment was designed to investigate the effect of active interaction algorithm animation on student learning. Following is a brief summary of the experiment's characteristics, followed by a more detailed coverage of the experiment's design:

- **Demographics**

Participants were the students from two sections of the introductory algorithms course (CS146: Data Structures and Algorithms) taught at the San Jose State University in the Spring 2008 semester. There were a total of 32 students.

- **Content**

Six algorithms were used in the experiment: Breadth-First Search, Depth-First Search, Kruskal's algorithm, Prim's algorithm, Bellman-Ford algorithm, and Dijkstra's algorithm.

- **Format**

Students were asked to complete six exercises (one for each algorithm). All six exercises had identical structure which is outlined in details below. In a nutshell, an exercise consisted of five phases: Question 1, Tool 1, Question 2, Tool 2, Question 3. In the question phase students were presented with a question about the algorithm, and in the Tool phase students used active interaction animation tool and passive interaction animation tool (the order in which the two tools were used by a particular student was decided at random).

Exercises were put on the department's web server and were available to students through the web browser. To take an exercise students had to log on to the course management system and follow a link to the exercise.

- **Data Collection**

Various data were collected for each exercise. In the beginning and end of each phase a log was saved that identified the student id, phase number, and the timestamp. In addition, for each "question" phase, the answer and the elapsed time were saved for each student, and for each "tool" phase the number of attempts and their elapsed time were saved for each student. All data were saved into relational database system.

The experiment consisted of a series of exercises, with each exercise covering a particular algorithm. Each exercise consisted of five phases outlined below:

- **Phase 1**

During this phase, student was presented with a question covering the algorithm. Student was shown a picture of a graph, asked a question about what the algorithm would do on that graph instance, and asked to submit

the answer in the provided HTML form. The student was given 24 minutes to complete the question, and the answer along with the elapsed time it took the student to answer were recorded.

This phase is designed to measure the student's initial understanding of the algorithm which had been shown in class.

- **Phase 2**  
During this phase, student was presented with one of the two tools, either the interactive tool or the animation tool. Which tool was given was determined at random. Student could spend unlimited time on the tool, hopefully using it many times to further his/her understanding of the algorithm. The total accumulated time spent on the tool was recorded.
- **Phase 3**  
This phase is identical to phase 1, except a different question was asked. This phase is designed to measure the effect (if any) of the first tool on student's comprehension of the algorithm.
- **Phase 4**  
This phase is identical to phase 2, except a second of the two tools is presented to the student (e.g. If interactive tool was picked in phase 2, the animation tool will be chosen for phase 4).
- **Phase 5**  
This phase is identical to phase 1, except a different question was asked. This phase is designed to measure the effect of the second tool on the student's comprehension of the algorithm.

The “question” phases (phase 1, phase 3, and phase 5) had several restrictions. First, we deemed it necessary to limit the time allowed to complete the question in order to ensure that the answer reflects the student's current understanding of the algorithm. That is, we wanted to avoid instances where a student, after seeing the question, would have had time to go back to the textbook or some web site to do additional studying relevant to the question. The time limit for the question was set at about 24 minutes, with questions designed to be answerable within 10 minutes. The 24 minute limit was also due to the fact that it was the default session timeout value on the web server used for our experiment.

Secondly, students were not allowed to interrupt the question web page (for instance, by reloading the page, going back a page, or closing the browser's window). Any interruption would count as an incorrect answer. This restriction was deemed necessary to help ensure proper experiment etiquette. For instance, we wanted to avoid the scenario where student would see the question and then go back to use one of the animation tools. Not having this restriction would also cause difficulty in recording an accurate time it took the student to answer the question. Since the student could have, potentially, copied the



question, exited the browser, then worked on the problem off-line, coming back to the question page later to submit the answer.

Before loading the question page, students were shown a warning page that explained to them about the aforementioned restrictions. Furthermore, any attempt to interrupt the question page, would result in a confirmation window popping up that warned them that leaving the page would result in their answer being marked as an incorrect answer.

We chose not to limit the time students could spend on the “tool” phases (phase 2 and phase 4) because our goal was to measure the tool's effect on learning. Limiting student's interaction time with the tool could potentially interfere with the measurement by not allowing students to have enough time to learn and get the best benefit. Besides, if the tool were ineffective, it would not matter how long it would be used by the student; the student would be unlikely to improve his/her understanding of the algorithm by using the tool for a longer time.

To analyze the design of our experiment we refer to [14], which discusses general issues with experiments on algorithm animations and lists seven animation-specific design factors that a good experiment should address. Below is the table that describes the seven design factors, their definition from [14] and how they were addressed in our experiment.

<b><i>Factor</i></b>	<b><i>Definition</i></b>	<b><i>Notes</i></b>
Usability	The system should be sufficiently mature so that it produces little or no student feedback on environment problems or envisioned system enhancements.	Overall there were no major negative feedback from the participants. A few minor issues came during one of the exercises but were resolved before the next exercise.
Animation quality	Instructor input should guide animation design, based on pedagogic experience.	Designs for both of the interactive tools used in our experiment were lead by the CS Professors at SJSU. The system, however, is in development and has concentrated less on animations than on the interactive quality.

Training	Students should receive hands- on instruction for operating the animation environment, especially learning and study. skills	Unfortunately, time constraints and other circumstances prevented us from being able to provide hands-on training to participants.
Logistics	The animation platform should be widely and easily accessible.	Experiment was run over the internet and all tools were implemented as Java applets.
Animation use type	The experiment should target one specific use of animations.	Specific target was the animation of graph algorithms.
Individual differences	Demographic and cognitive factors that may affect response should be factored into results.	The tool type that each student used first was picked at random, providing a true random sample.
Algorithm difficulty	The algorithm being animated should be sufficiently complex that subjects will feel a need for additional support, but not so complex that it cannot be well visualized.	Our experiment consisted of six algorithms ranging in difficulty from elementary to moderate.

**Table1: *Experiment design factors***

Due to the time constraints, we had limited time for the design and planning of the experiment, however we feel it measures up reasonably well against the aforementioned design factors.

### 3 Experiment setup

This section describes how the experiment was setup and carried out. It provides specific details about the tools used in the experiment, the algorithms covered by the exercises, and the environment in which the experiment was carried out.

The setup for running the experiment consisted of several parts which are outlined below and are covered in more detail in the sections that follow.

- **Environment**

One of the main aspects for the experiment setup was the exercise environment. Decisions had to be made on how the exercises would be made available to the students, what format would the exercises have, and how the exercise statistics would be collected.

To make the exercises widely and easily accessible, we made them available over the internet. Each exercise was implemented as a series of web pages, where each web page represented a logical phase of the exercise. The exercise web pages had a predefined progression so that the participant was taken from one phase of the exercise to the next with a click of a corresponding button. Please refer to Appendix A for more details and for the series of screen shots that illustrate the flow of the exercise.

For some exercise phases (question phases) students had to complete the phase as a logical unit of work, meaning that they could not interrupt the phase web page and had to complete the phase within a given time limit. For other phases (tool phases) students were allowed multiple (but not concurrent) sessions, meaning that they could exit the phase web page and return to it at a later time. There was no restriction on how many times students could enter/exit the phase and there was no restriction on the time student could spend on the phase.

All student activities were logged, and various statistics collected. Both logs and statistics were saved into a relational database (PostgreSQL). After completing an exercise, student would not be allowed to repeat that exercise again (otherwise the data would be inaccurate).

To minimize the amount of work, the exercises were deployed using the course management system (Moodle). The advantage of doing so was that we would not need to implement the user management system and could take advantage of several course administration features that course management systems offer. It also made it possible to reuse code for HTML form processing, HTML formatting, logging, and database access.

- **Exercises**

For the experiment, we wanted to collect data for six different algorithms. Thus, it was natural to break up the experiment into six independent parts, each part covering one algorithm. In a sense, our experiment was really a set of six smaller experiments. We refer to these experiments as “exercises” since they were presented to the students as a homework exercises that were graded.

- **Questions**

Each exercise had three question phases. Therefore, we needed to compose three different questions per algorithm. Questions had to be of comparable difficulty. All questions had the same format in the sense that some graph was given to the student and he/she was asked about an algorithm's behavior on that graph. For the answer portion, student was asked to provide some specific information that would tell us whether the student ran the algorithm correctly and how well he/she understands the algorithm. Thus, the answer part was naturally different for each algorithm.

- **Tools**

Given that the exercises were accessed through the internet via a course management system, a natural choice for the tool format was a Java applet. Thus, for each exercise, a corresponding tool was embedded into the tool page as an applet.

### **3.1 Interactive Tools**

To recap earlier discussion (see Section 1.1), when we refer to interactive tools, we refer to active interaction algorithm animation tools. These tools not only show the animation of an algorithm but also allow the user to engage in active learning. The most common feature of interactive tools is allowing the user to direct the animation flow by telling the tool what to do next. Intuitively, this is a powerful learning feature, and is the subject of our study.

Surprisingly, the number of active interaction algorithm animation tools is small. In fact, at the time of writing this report, we are aware of only two such tools that are rich in interactive features and cover many algorithms: IDSV (Interactive Data Structures Visualization) [15], TRAKLA2 [16]. Also, a new tool - Framework for Active Learning [17] is being currently developed here at San Jose State University.

For our experiment we decided to use the tool that is being developed at our university. It was a good opportunity to try the tool with a wider audience and get some feedback from the students, which could be valuable to the tool designers. At the time of experiment design and preparation, there were two preliminary

versions of the interactive tool, each having a slightly different approach to the interface, features, and level of interaction. For instance, one of the versions, similar to TRAKLA2, allows students to make incorrect moves and provides the feedback showing correct moves along side the student's moves. Another version of the tool, similar to IDSV, prevents incorrect moves by immediately displaying an error message and giving a hint about what the correct move should be. We used both preliminary versions in our experiment since they were developed in parallel and covered different algorithms. This allowed us to include more algorithms in our experiment. Perhaps the experiment data could also be used by the tool designers to help shape the future direction of the tool development.

In this report we will refer to one of the versions of the tool as tool1 and the other version as tool2. This numbering is arbitrary and does not imply that one of the versions is more significant than the other, nor does it portray any preference for one version or the other. These are simply two versions of the interactive tool.

Following are brief descriptions of the two versions of interactive tool that we used in the experiment.

### **3.1.1 Tool1**

The framework is the result of Sean Sharma's Thesis work [17] under guidance of Dr. Horstmann. In the software system implemented under this framework, user is presented with the visualization of a problem, covering a particular algorithm, and with the action choices to apply the algorithm to visualization. The user is required to solve the problem by using specific actions at appropriate time to manipulate the appropriate visualization objects. In a sense, the user is acting on the behalf of the algorithm and directing the animation step by step as if the animation was run by the algorithm. When user instructs the system to make the next move, the action is checked against the algorithm. If the move does not correspond to what would have been done by the algorithm, an error is issued informing the user of the incorrect move. Otherwise, user's direction is carried out by the system by animating the step of the algorithm. User continues to instruct the system step by step until the problem is completed. The framework provides user with the ability to ask the system to make the next move via Show Next Step mode, similar to how it is done in the IDSV.

The usefulness of this feature is twofold. It could be used by the user as 'help', perhaps once the user reaches a point where he/she is not sure what the correct move would be. Or it could be used to run the animation of the algorithm entirely from start to finish, perhaps by the students less familiar with the algorithm and wishing to see some animated examples first. Note that if used in this manner,

the system would not be utilized effectively and the active interaction aspect would be lost.

Figure 1 below illustrates the instance of the Bellman-Ford shortest path algorithm problem.

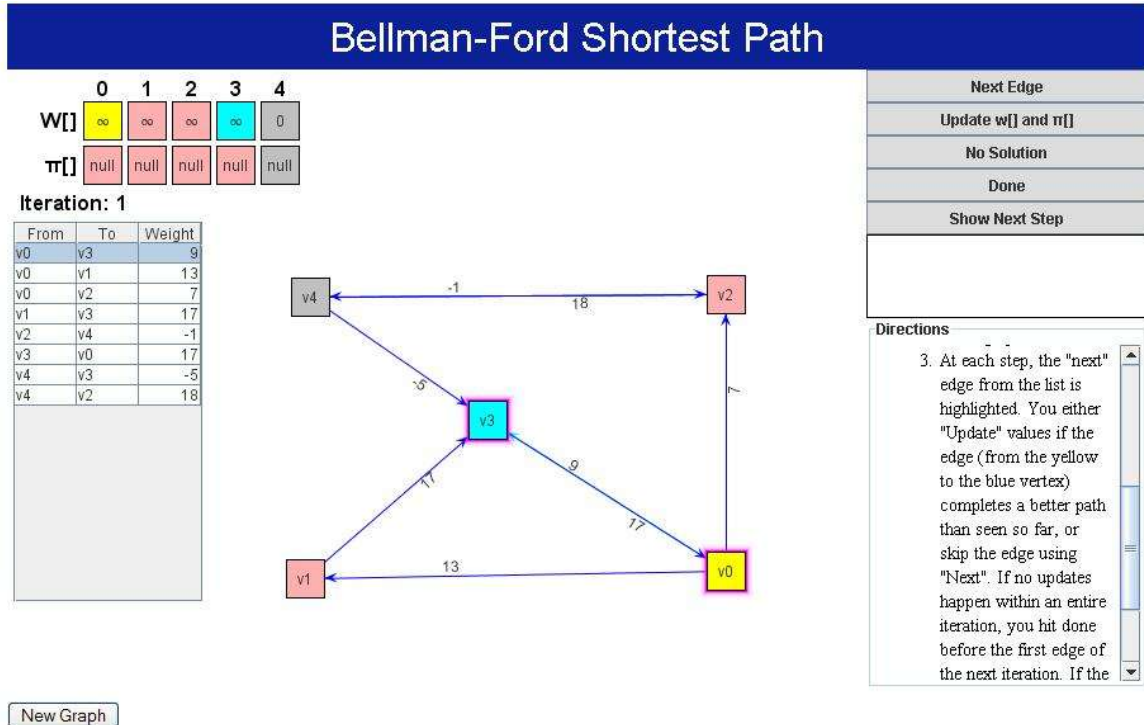
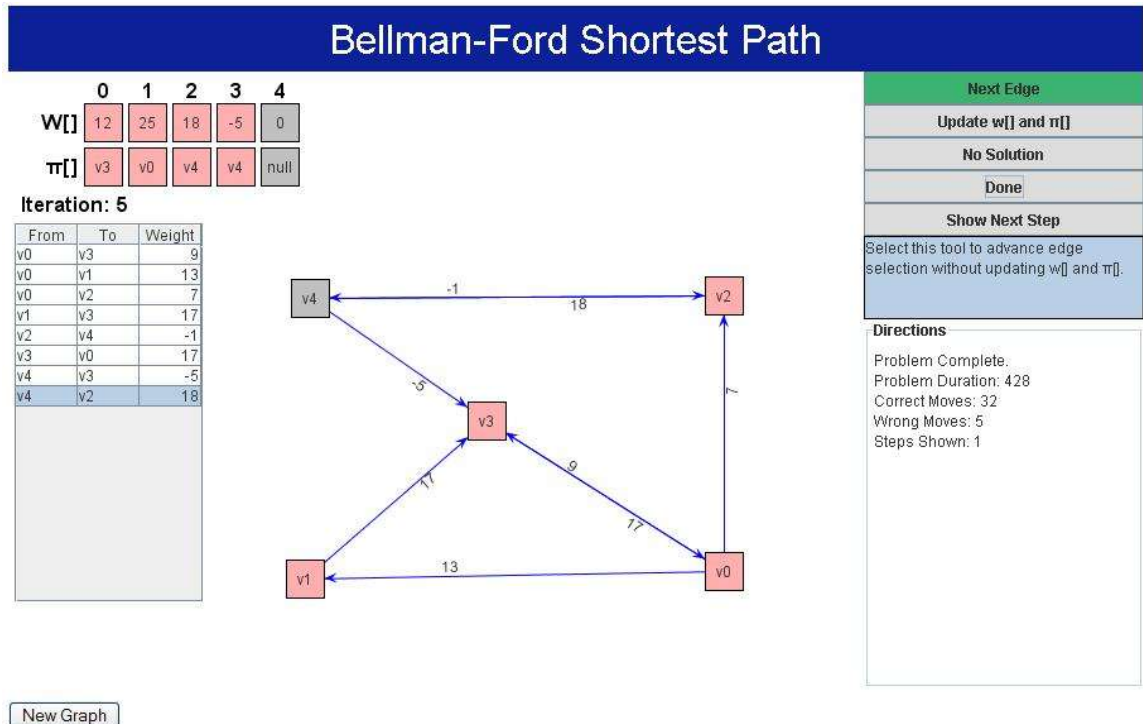


Figure 1: Instance of the Bellman-Ford algorithm problem

Available actions are presented on the right hand side of the tool's window along with some directions. User is expected to select appropriate action (“Next Edge” or “Update w[] and π[]”) for each step until problem is completed. Note that the framework allows the problem creator to control the level of user interaction for the problem. For instance, in the example shown in Figure 1, the next edge is automatically selected when user's action is “Next Edge”, however the problem could be easily setup so that the user would have to pick the next edge manually by clicking on it.

The framework automatically keeps track of the basic statistics such as: the number of correct moves, the number of incorrect moves, the number of times Show Next Step mode has been used, and the elapsed time it takes the user to finish the instance of the problem. These statistics are shown to the user when

the problem is completed. As an example, Figure 2 below shows the completed instance of the problem from Figure 1.



**Figure 2: Completed instance of the Bellman-Ford algorithm problem**

Note that this tool's design idea was to make creating exercises easy, with only minor modifications to the algorithm.

We chose to use this tool for two exercises of our experiment: Breadth-First search algorithm, and the Bellman-Ford algorithm. In the remainder of this report we will refer to this tool as interactive tool1.

### 3.1.2 Tool2

This framework was developed by Edward Yin under guidance of Dr. Taylor. The framework has somewhat different objectives compared to the tool1. It puts more emphasis on the automated feedback and automated grading, allowing students more freedom to "go wrong" and catch their own mistakes. This tool allows more general interaction than tool1, at the cost of the exercises being more complex to write. At the time of conducting our experiment (and writing this

report), the framework was still under development but had enough functionality to be used for our purpose. This framework is similar to the TRAKLA2 framework.

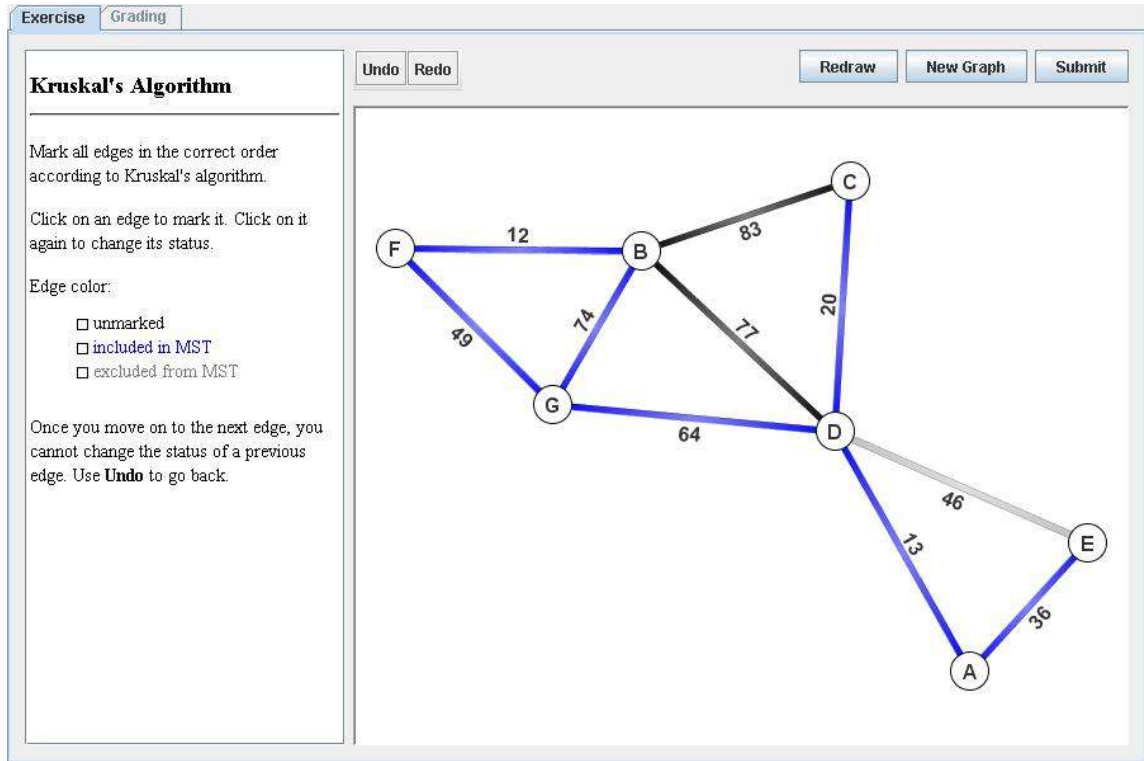
In the software system implemented under this framework, a user is presented with the visualization of a problem, covering a particular algorithm, and with the action choices to apply the algorithm to visualization. The user is required to solve the problem by using specific actions at appropriate time to manipulate the appropriate visualization objects. Just as in the framework of Tool1, the user is acting on the behalf of the algorithm and directing the animation step by step as if the animation was run by the algorithm. When user instructs the system to make the next move, the action is registered in the internal structure and user's direction is carried out. Note that unlike interaction tool1, the action is not checked against the algorithm until the instance is submitted for automated grading. Thus, no errors are issued for the incorrect moves during the exercise. User continues to instruct the system step by step until the problem is completed.

The framework provides user with the ability to undo/redo the moves, which is an essential feature when no errors are issued for invalid moves right away. Should the user make a mistake at some step, he/she may be able to realize this at the later time (perhaps when a problem state is reached where something does not make sense). User then could use the undo feature to go back to the step where mistake was made and make a correct move.

The framework does not provide user with the ability to ask the system to make the next move. Instead, it provides are more advanced feedback feature. Once the Submit action is picked by the user, the system goes into the grading mode where the correct moves are compared side by side with the moves made by the user. Moreover, user is able to return to the exercise from the grading mode and freely toggle between the modes. This powerful feature is similar to TRAKLA2 and can be used in several ways. One way is to use it as a validation of the submitted work. User would work on the problem to completion (or to certain point), then go to the grading mode and check the submitted moves for correctness. Another way this feature could be utilized is to emulate the Show Next Step mode of other tools. To see the next move (or series of moves), user would switch to the grading mode, step to the point where he/she is currently at in the problem instance, and then make one or more steps to see the next correct moves. On the extreme side, the system could be used to just see the animation of the algorithm from start to finish without having to provide correct moves. This can be achieved by making an arbitrary first move and then switching to the grading mode and stepping through the problem instance watching the animation on the side of the window where the correct moves are displayed. Note that if used in this manner, the system would not be utilized effectively and the active interaction aspect would be lost.



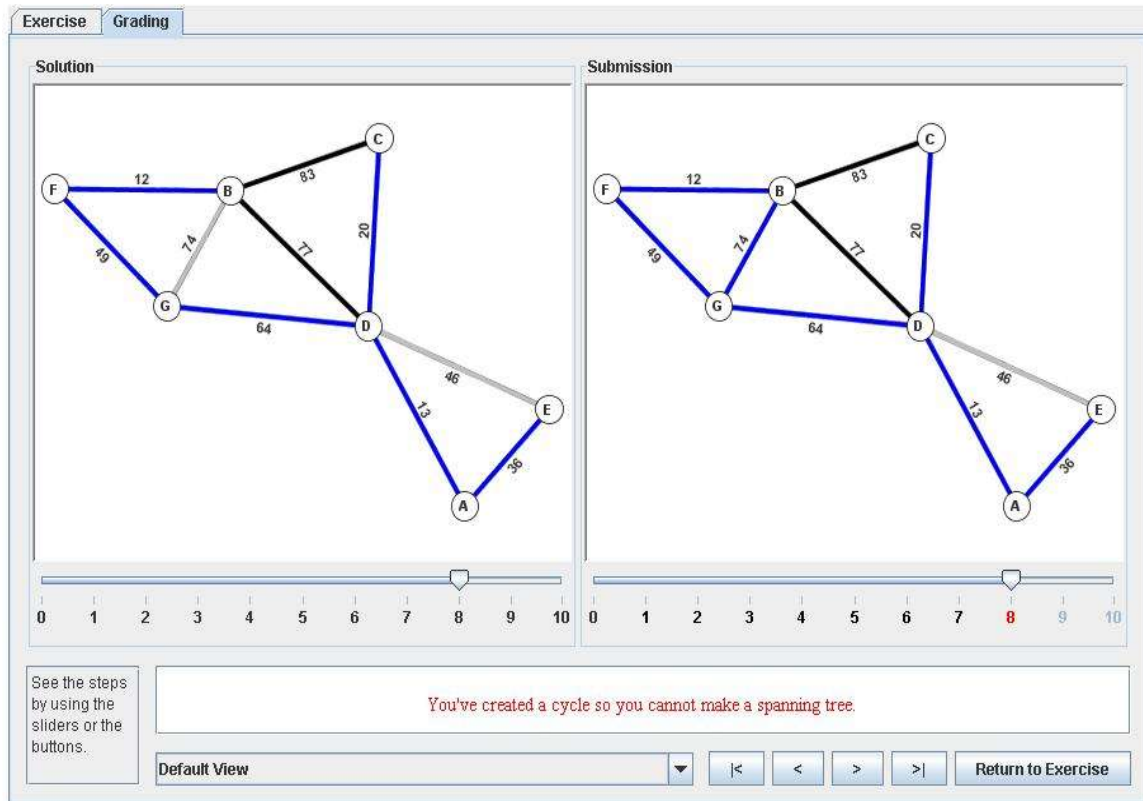
Figure 3 below illustrates the instance of the Kruskal's minimum spanning tree algorithm problem.



**Figure 3: Instance of the Kruskal's algorithm problem**

At the time of our experiment, the framework did not automatically collect user statistics. Later phases of the framework design plan to implement collection of various statistics such as number of correct moves, number of incorrect moves, number of times the Undo, Redo, Submit, and New Graph actions were taken, elapsed time of the exercise, etc. To facilitate our experiment, the framework added the ability to calculate the elapsed time it takes the user to finish the instance of the problem.

Figure 4 below demonstrates the grading mode. It shows the error made by the user on step number 8. User is able to go back, undo the steps back to the step number 7, make the correct move number 8 this time around, and then resume making moves.



**Figure 4: Grading mode for the Kruskal's algorithm problem instance**

We chose to use this tool for four exercises of our experiment: Depth-First search algorithm, Kruskal's algorithm, Prim's algorithm, and Dijkstra's algorithm. In the remainder of this report we will refer to this tool as interactive tool2.

### 3.2 Animation Tools

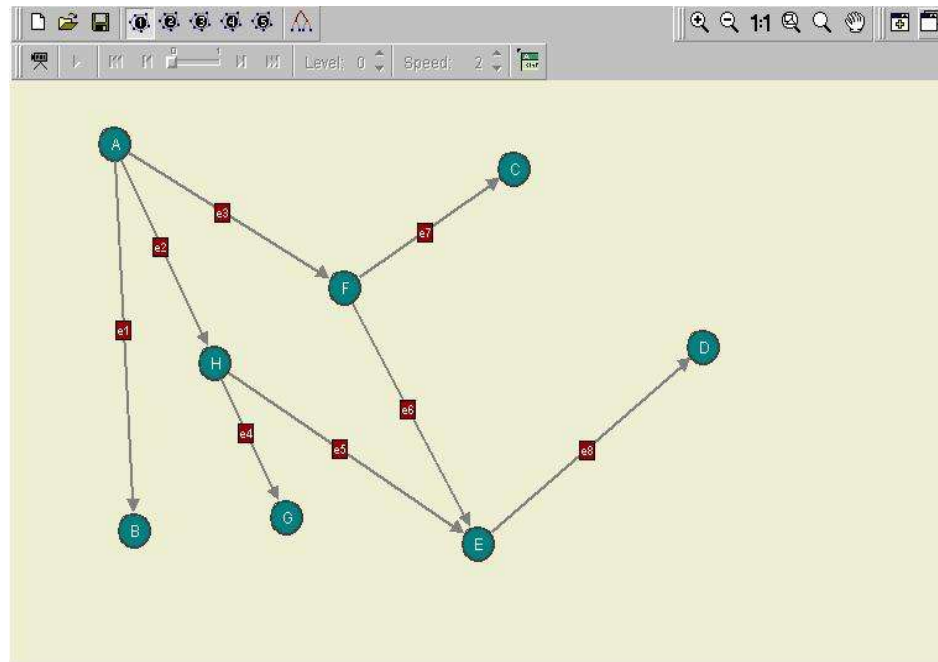
To recap earlier discussion (see Section 1.1), when we refer to animation tools, we refer to passive interaction algorithm animation tools. These tools have some basic interactive features, most common of which allows the user to control the animation flow by pausing, stepping, or running the animation at specified speed. Several algorithm animation tools introduced the ability for the user to create his/her own input to the tool to animate. For instance, graph animation tools allow user to build new graphs from scratch and then run animation on them. This is a good step towards active interaction, however it is not clear how beneficial this feature is for unmotivated students or students who understand very little of the algorithm they are trying to learn. After all, if one does not understand the basics

of the algorithm, how can he/she construct a meaningful graph that would address the peculiarities of the algorithm one is trying to learn. Perhaps this feature is most useful to those who understand algorithm well but are unsure about certain instances or special cases of applying the algorithm. In that case it is great to have the feature that allows one to construct the graph to test that particular aspect of the algorithm. We chose not to classify animation tools with this “build your own graph” feature as active interaction animation tools.

To our surprise, finding animation tools for our exercises turned out not to be as trivial as we expected. According to the report in [3], there are hundreds of algorithm animation tools available on the internet. The authors maintain a catalog of several hundred algorithm visualizations [3,5]. However when we visited the catalog, we found that only 39 of those were graph algorithms, and about a quarter of those were traversals (Breadth-First Search, Depth-First search). In addition, we needed animations that would be easy to embed into a web page so that we could keep track of certain statistics (like elapsed time and number of uses). We were not able to find one animation tool that would match our criteria and cover all algorithms in our experiment, thus, we picked three animation tools that seemed of a good quality to us and were easy to embed into a web page. Following is a brief overview of each one.

### **3.2.1 Animation tool1**

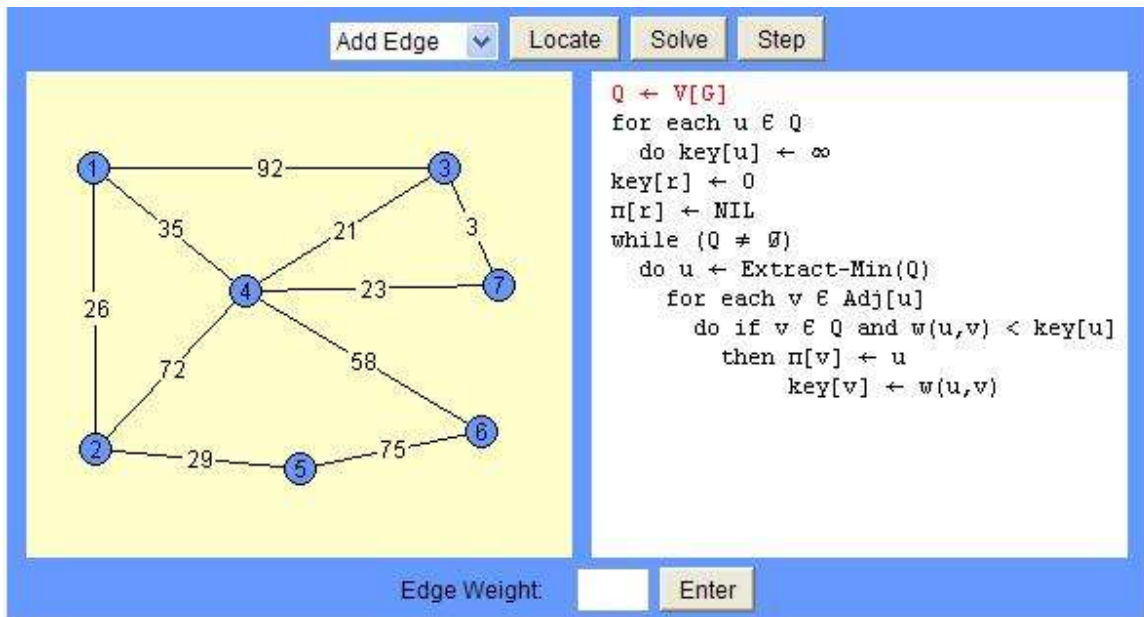
For our first exercise (Breadth-First Search) we chose the animation tool developed by Thomas Wolf [18]. Figure 5 below shows a screen shot of the applet. This animation tool had predefined graphs constructed by us using this same tool (the tool allows saving created graphs on local disk). It also allowed users to construct their own graphs. We refer to this tool as animation tool1.



**Figure 5: Thomas Wolf's animation applet**

### 3.2.2 Animation tool2

For our fourth exercise (Prim's algorithm) we chose the animation tool developed by Mustafa Incel [19]. Unfortunately, this tool did not support predefined graphs, so users had to create their own. The other Prim's algorithm animation tool we looked at had predefined graphs but did not allow creation of the graphs by the user and had no annotations making it hard for those who don't know the algorithm well to follow the animation. The tool developed by Mr. Incel had a nice feature that showed a highlighted line progressing through the pseudocode of the algorithm, allowing user to make better connection between the animation and the algorithm. Figure 6 shows a screen shot of this animation tool. We refer to this tool as animation tool2.

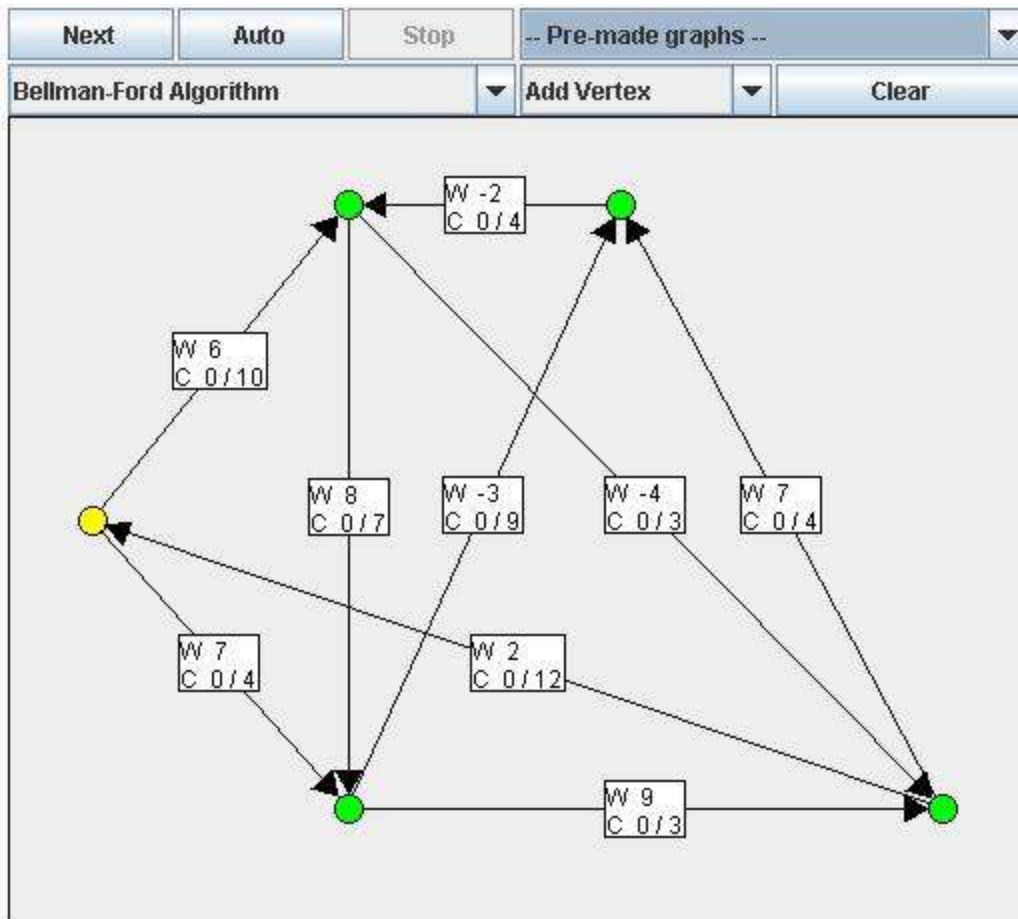


**Figure 6: Mustafa Incel's animation applet**

### 3.2.3 Animation tool3

For the rest of our exercises we chose the animation tool developed at Rensselaer Polytechnic Institute [20]. This tool has predefined graphs and also allows for creation of customized graphs. The interface is straight forward and intuitive. We found this tool to be the best of all other animation applets that we have looked at, needing no real explanation for students to use, and were fortunate to be able to use it for most of our exercises.

Figure 7 below shows a screen shot of this animation tool. We refer to this tool as animation tool3.



© 1998 Rensselaer Polytechnic Institute

**Figure 7: Animation applet by Rensselaer Polytechnic Institute**

### 3.3 Moodle

The experiment was deployed in the Moodle environment. Moodle is a course management system (CMS). It is a free, open source software platform that allows educators to create and manage online courses and learning communities [21]. Moodle is developed in PHP and its modular design allows for easy extension.

Our choice of Moodle as a deployment platform was twofold. Most importantly, we needed to make exercises to be easily accessible and maintained, as well as track student activities, all without investing into developing a framework from the ground up. A secondary goal was to design the exercises in a way that would

allow Computer Science professors at San Jose State University to incorporate the exercises and interactive tools into their curriculum for teaching algorithm courses. Moodle's abundant features, modular design, and open source nature allowed us to meet both goals.

The following Moodle features were utilized:

- **User management**  
Each participating student was given a user id and was enrolled into a CS146 course created specifically for the experiment. Recall that the population sample for our experiment consisted of all student enrolled into the two traditional CS146 course sessions taught at the SJSU in Spring 2008 semester. The professor teaching both sections of the traditional CS146 course (Dr. Taylor) was assigned a teacher role in this online course.
- **Course management**  
An online CS146 course was created to carry out our experiment. The course had a topic format, with each topic being an exercise for a particular algorithm. The course teacher (Dr. Taylor) made each exercise available to students after covering associated algorithm in class. Each exercise was assigned as a graded homework and had a due date of approximately one week after the assigned date.
- **Database access**  
A new database table was created to store the exercise statistics, and several existing tables were used for the experiment. We were able to utilize the existing code for interacting with the database (e.g. querying and modifying the tables).
- **Logging**  
We were able to take a full advantage of existing logging capabilities. Moodle automatically logs every activity, and therefore, we could extrapolate certain statistics from the log records in addition to the statistics we explicitly recorded. These logs proved to be useful in diagnosing some issues related to student activity (e.g. session timeouts due to user inactivity).
- **HTML support**  
We were able to significantly reduce the amount of coding effort by reusing existing functions for HTML formatting and form processing.

Our experiment setup in Moodle was as follows:

- 1) CS146 course was created in Moodle, and user ids with initial passwords were created for each student.
- 2) Each student was sent an e-mail note from the instructor, identifying the

web site link to the course and the student's user id along with the temporary initial password. Students were required to change their initial password at the time of the very first log in to Moodle.

- 3) When logging in for the very first time, students were presented with a consent form. This form was a fulfilment of the university's requirement as applied to all studies conducted on students. The form explained the nature of the experiment and provided the option for a student to opt out from the experiment. The choice of the student to participate or opt out was not made known to the instructor with the caveat that anonymous consent or rejection would be discernible. Statistics collected for those who opted out would not be included into analysis (however, all students elected to participate). The consent form can be seen in its entirety in Appendix B.
- 4) After logging in (and completing the consent form in the case of the very first log in), students would come to the main page of the course where the links to the exercise were presented. Students could navigate to the desired exercises or other features of the course. Each exercise was made available to students as the associated algorithm was covered in the class lecture. Figure 8 below, shows the main course page with all six exercise available to students.
- 5) In addition to exercise links for students, course activity report was also modified so that the course teacher and the experiment administrator could easily see the progress of each exercise. The activity reports showed, for each exercise, a list of students who attempted the exercise. In addition, for each student who attempted the exercise, some specific information could be obtained (by clicking on the student name) that showed the exercise phases completed by the student at that time and the phases statistics. For example, student's answers for the question phases could be seen, and/or elapsed times for using the tools could be seen. Figure 9 below gives an example.



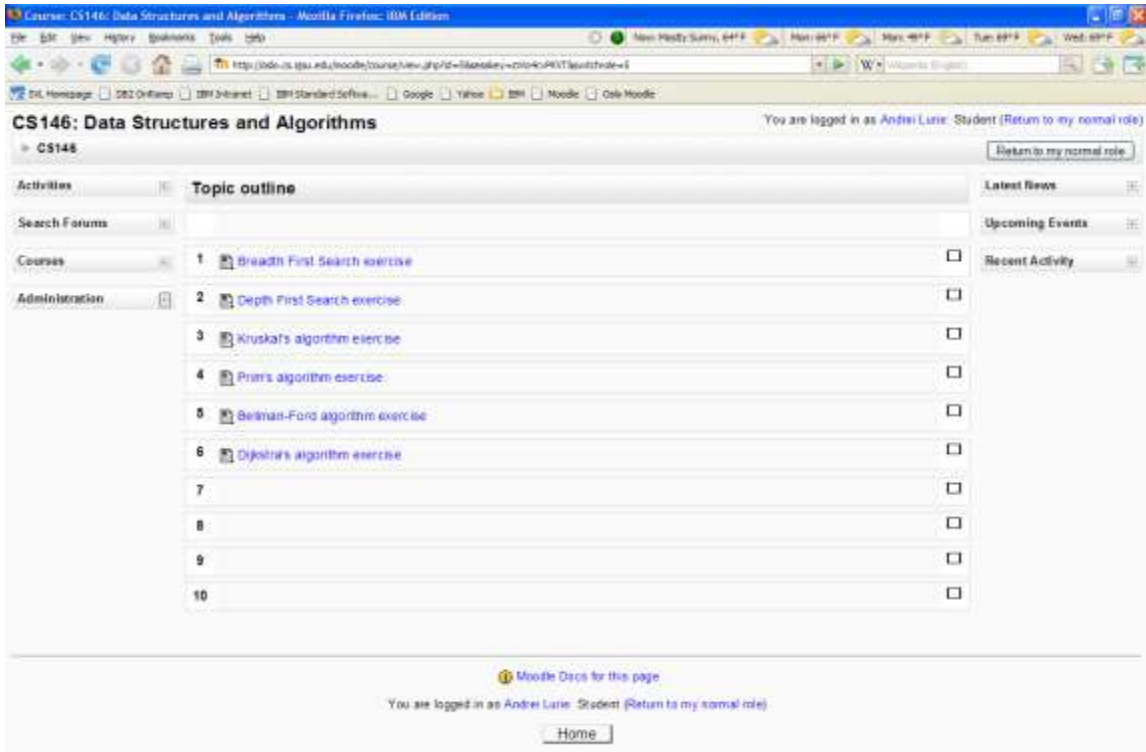


Figure 8: CS146 course main page with six exercises



Figure 9: Detailed exercise report for a student

### 3.4 Exercises

The goal of each exercise was to measure the effectiveness of the active interaction tool on student learning and to compare this effectiveness to the one of the animation tool. Unlike many studies done in the past, we chose not to have a screening test for dividing participants into different groups. Instead, we decided to randomly assign students into groups. One group was the students who were given interactive tool as the first tool to use, and the other group was the students who were given the animation tool first. These groups were decided randomly for each exercise. Thus, the first question of the exercise served as a baseline for each student, and the second question was used to measure the effect of the tool.

Furthermore, the experiment participants were asked to use a second tool (whichever one was not picked as the first tool) after answering the second question. For example, if an interactive tool was picked for the student first, the animation tool would be picked for the second tool. The third question was asked after a student had used the second tool to study. The addition of this “second tool” and “third question” phases were mainly added to ensure the exercise fairness for the students. Because the exercises were used as graded assignments and the algorithms practiced in these exercises were tested on the midterms, it would be unfair to give students different tools to learn the algorithms. Therefore, the experiment was designed to give all students equal opportunity for studying the algorithms. Statistics collected from these additional phases could potentially be used for additional analysis.

The structure of all six exercises was the same. The only difference between the exercises was the choice of tools and the covered algorithm. The flow of the exercise consisted of the series of web pages that were loaded in predefined order. Below is the classification of the web pages that made up the exercise:

- **introduction page**  
The introduction page informed the student about the algorithm the exercise covered, and gave detailed instructions about each phase of the exercise. A “Start” button was provided below the directions which would take the student to the next page.
- **pre-question page**  
The pre-question page informed the student that the next page is the question page and that the question would need to be answered within 24 minutes without any web page interruptions such as reloading the page, or leaving the page by navigating the browser back or closing the browser window. Even though the introduction page provided the same information, we deemed it necessary to remind the student about these restrictions before each question. A “Next” button was provided below the information text which would take the student to the next page.

- **question page**  
The question page contained a question that tested the student's understanding of the algorithm. HTML form was provided for the student to enter and submit the answer. Question structure is described in more details in the specific exercise sections that follow.  
Note that, as mentioned before, the question page could not be interrupted. A warning message box would open if the student attempted any interruption. If, after the warning, the student wished to exit anyway, the answer for the current question would be saved as incorrect (“empty” answer string).
- **tool page**  
The tool page contained an applet of the particular tool (interactive tool or animation tool). A “Done” button was provided below the applet which would be used by the student to terminate the tool phase and proceed to the next page.
- **final page**  
The final page was displayed after the last phase (the third question) was completed by the student. The page informed the student that the exercise was completed and also had an embedded interactive tool used for the exercise. The interactive tool was given on the final page for those students who would want to use the tool even after completing the exercise (perhaps to study for the class test or to voluntarily use the tool to further their understanding of the algorithm).

Following is the overview of the exercise flow as it would occur on the very first time the student attempted the exercise. Some notes are included that describe what the flow would be for the repeated exercise attempts by the student (refer to Appendix A for the example screen shots):

- When a student clicked on the exercise link in the Moodle CS146 course, the he/she would be presented with the introduction page.  
All subsequent attempts of the exercise would also start with this page (for the sake of reminding the student about the logistics of the exercise).
- When a student clicked the “Start” button on the introduction page, he/she would be taken to the pre-question page of the first question. If this was not the first attempt of the exercise, the student would be taken to the last page he/she was on before quitting the exercise, with one exception: if the last page the student was on before exiting the exercise was the question page, then the page the student would be taken would be the page of the next phase after the question (it would be one of the tool's pages or the final page).

- When a student clicked “Next” button on the pre-question page for the first question, he/she would be taken to the first question page.
- After submitting the answer to the first question, the student would be taken to the first tool page. At this time, either the interactive tool or the animation tool would be loaded. The choice was determined by using PHP built-in random number generation function (mt\_rand) to get a random number. If the number was even, interactive tool would be chosen, otherwise the animation tool would be chosen. Students were encouraged to spend as much time as they need using the tool. By design, this would be the most common page where students would exit the exercise and come back to when resuming the exercise at a later time. A “Done” button was provided on the bottom of the page and student were instructed to click it only when they wished to terminate the tool phase and proceed to the next phase of the exercise.
- When a student clicked “Done” button, he/she would be taken to the pre-question page for the second question.
- When a student clicked “Next” button, he/she would be taken to the second question page.
- After submitting the answer to the second question, the student would be taken to the second tool page. The tool used on this page would be the one not picked for the first tool page (e.g. if intercation tool was picked for the first tool page, animation tool would be picked for the second tool page, and vice versa).
- When a student clicked “Done” button, he/she would be taken to the pre-question page of the third question.
- After submitting the answer to the third question, the student would be taken to the final page of the exercise. The exercise would be marked completed for the student. Any subsequent attempts of this exercise would bring the student directly to this final page.

Figure 10 below shows a diagram representing the exercise flow. The numbers next to the transition arrows to the Exit state enumerate different exit points, and the numbers next to the transition arrows from the Introduction state shows which state will be entered if the exercise were terminated at the corresponding exit point. For example, if the exercise were terminated at Question 1 phase or at tool1 phase, it would resume at tool1 phase, and if the exercise were terminated during pre-question 2 phase, it would be resumed at the pre-question 2 phase.

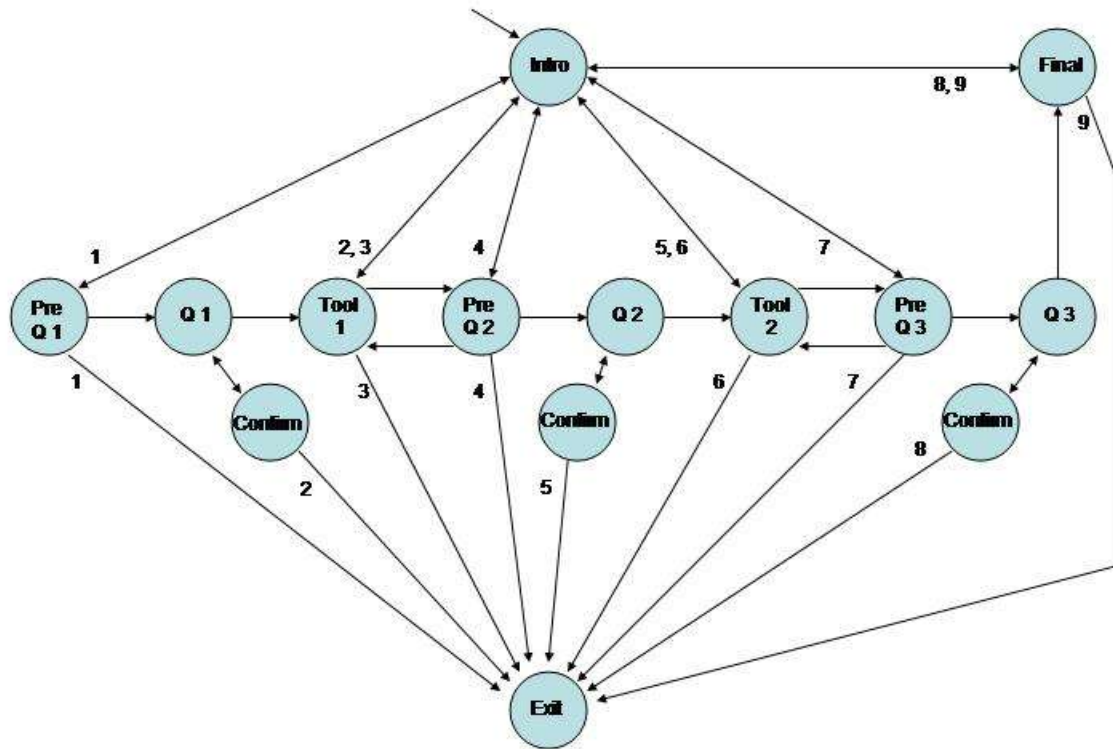


Figure 10: state diagram for the exercise flow

### 3.4.1 Breadth-First Search

Exercise number one covered the Breadth-First Search algorithm.

For the question phases of the exercise, the following question structure was used:

#### Question

Given the following graph, perform a breadth first search starting from vertex A (break all ties alphabetically). Compute the order of exploration of all vertices. Provide the answer in the form of a comma-separated vertex ids in the order of exploration (e.g.: A,B,C,D,E,F,G)

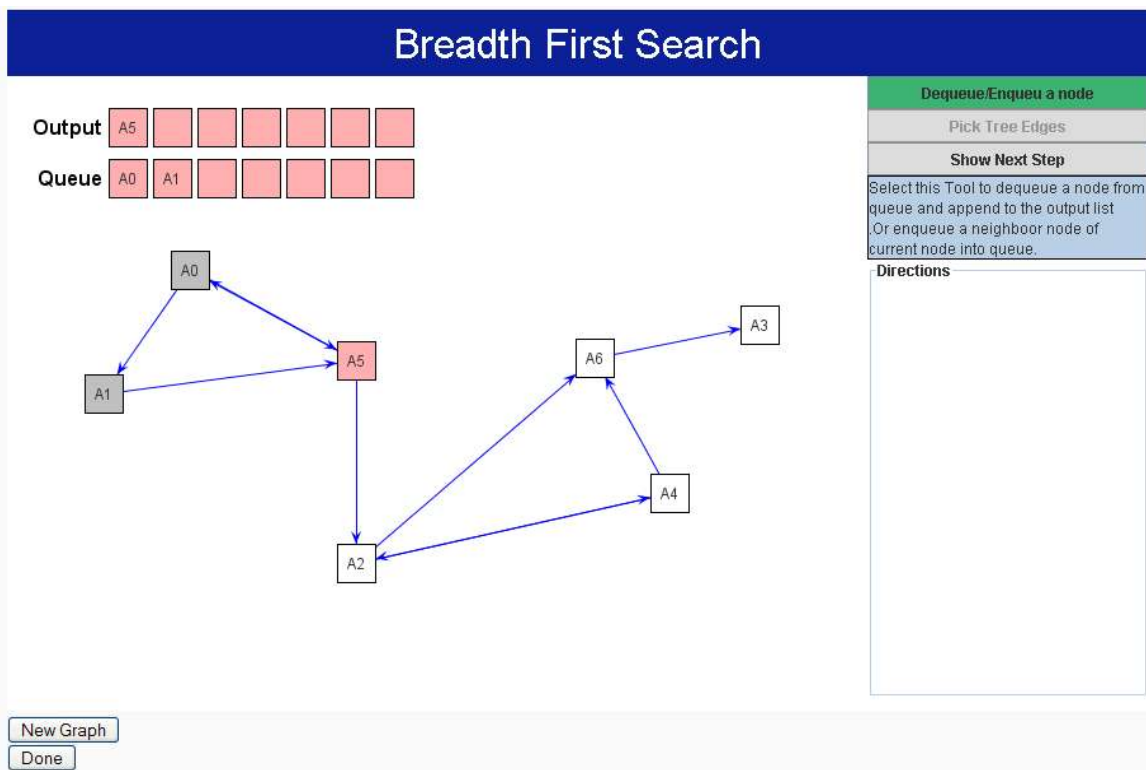
#### Answer format

A simple text box was provided for the student to enter the answer string.

For the animation tool, animation tool1 was used. Using this tool, students could observe the animation of the BFS algorithm being performed on several pre-

made graphs or students could construct their own graphs and watch the algorithm being performed on those graphs.

For the interactive tool, interactive tool1 was used. In this tool, students were presented with the randomly generated graph and were asked to direct the tool to perform the BFS algorithm on that graph. Student had to direct the tool by first clicking the vertices of the graph in the correct order to dequeue a vertex into the output list or to enqueue a neighbor vertex of the current vertex. Then, after all vertices were processed (dequeued into the output list), students needed to pick the correct tree edges to the current vertex selected by the tool from the output list. Figures 11 and 12 below illustrate these two phases (for two different graph instances).



**Figure 11: Selecting vertices to dequeue/enqueue**

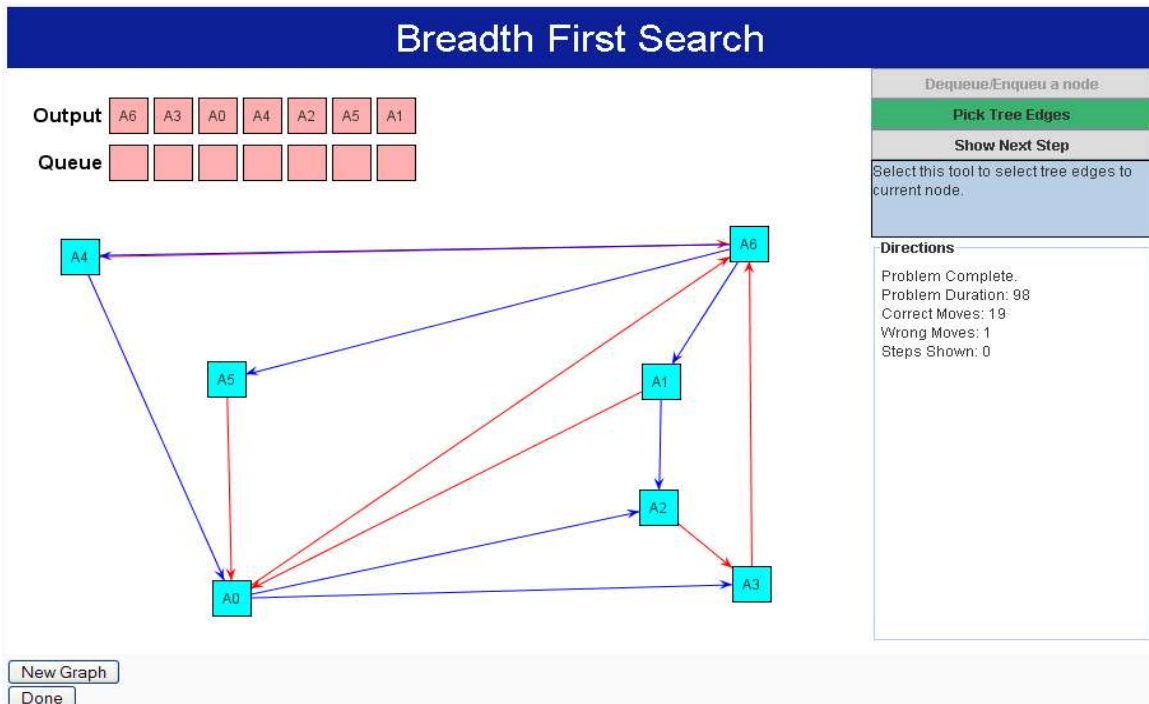


Figure 12: Selecting edges for the output vertices

### 3.4.2 Depth-First Search

Exercise number two covered the Depth-First Search algorithm.

For the question phases of the exercise, the following question structure was used:

#### Question

Given the following graph, perform a depth first search starting from vertex A (break all ties alphabetically). Compute discovery/finish time for each vertex, and label each edge as tree, back, forward, or cross. Provide the answer in the form entries below.

#### Answer format

For each vertex, two simple text boxes were provided for the student to enter the discovery and finish times of the vertex. For each edge, a drop down selection list was provided for the student to pick the correct edge classification. The choices were: "Tree", "Back", "Forward", "Cross".

For the animation tool, animation tool3 was used. Using this tool, students could observe the animation of the DFS algorithm being performed on pre-made graph or students could construct their own graphs and watch the algorithm being performed on those graphs.

For the interactive tool, interactive tool2 was used. In this tool, students were presented with the randomly generated graph and were asked to direct the tool to perform the DFS algorithm on that graph. Students had to direct the tool by first clicking the current vertex's edge appropriate number of times to classify it (each click of the edge cycled through the possible classification types). Once the edge was classified, the student had to make a decision whether the vertex on the other end of the edge needs to be entered. Vertices were discovered by clicking on them once and vertices were finished by clicking in them again. Figure 13 below shows an example of the DFS problem.

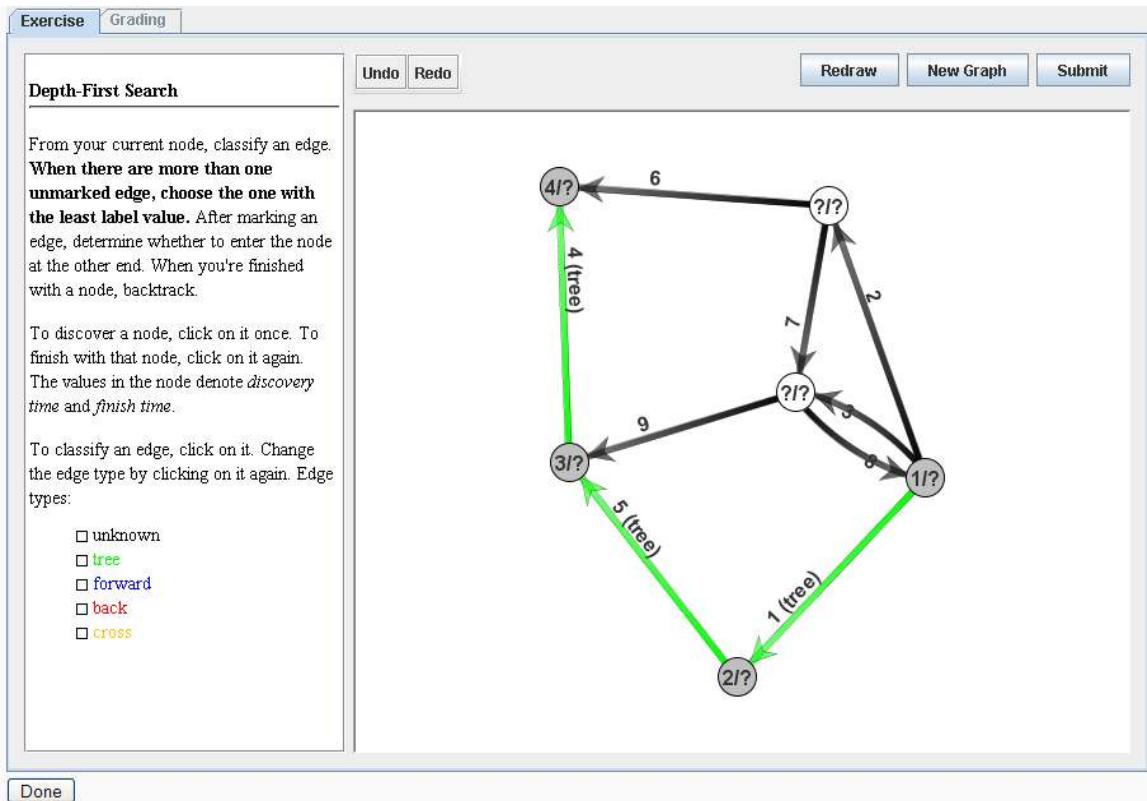


Figure 13: DFS problem in interactive tool2

### 3.4.3 Kruskal's Algorithm

Exercise number three covered the Kruskal's Minimum Spanning Tree algorithm. For the question phases of the exercise, the following question structure was used:



### Question

Given the following graph, find the minimum spanning tree using Kruskal's algorithm. Provide the answer in the form of a comma-separated edges in the order of them being added to the tree (e.g.: AB,FE,BC)

### Answer format

A simple text box was provided for the student to enter the answer string.

For the animation tool, animation tool3 was used. Using this tool, students could observe the animation of the Kruskal's algorithm being performed on pre-made graph or students could construct their own graphs and watch the algorithm being performed on those graphs.

For the interactive tool, interactive tool2 was used. In this tool, students were presented with the randomly generated graph and were asked to direct the tool to perform the Kruskal's algorithm on that graph. The student had to direct the tool by clicking the edges in the correct order appropriate number of times to classify them as either included into the tree or excluded from the tree. Figure 14 below shows an example of the Kruskal's problem.

The screenshot shows an interactive tool for Kruskal's Algorithm. The interface includes a title bar with 'Exercise' and 'Grading' tabs. Below the title bar are buttons for 'Undo', 'Redo', 'Redraw', 'New Graph', and 'Submit'. The main area is divided into two panels. The left panel, titled 'Kruskal's Algorithm', contains instructions: 'Mark all edges in the correct order according to Kruskal's algorithm. Click on an edge to mark it. Click on it again to change its status.' It also includes a legend for 'Edge color:' with three options: 'unmarked' (grey), 'included in MST' (blue), and 'excluded from MST' (black). A note at the bottom of the left panel states: 'Once you move on to the next edge, you cannot change the status of a previous edge. Use Undo to go back.' The right panel displays a graph with 7 nodes (A, B, C, D, E, F, G) and 10 weighted edges. The edges and their weights are: (B,C) weight 16, (C,A) weight 63, (A,D) weight 27, (D,G) weight 31, (G,E) weight 70, (E,F) weight 37, (F,D) weight 13, (D,B) weight 75, (C,D) weight 38, and (B,F) weight 75. The current state shows edges (B,C), (C,A), (A,D), (D,G), (G,E), (E,F), and (F,D) in blue, (B,F) in black, and (C,D) in grey. A 'Done' button is located at the bottom left of the interface.

Figure 14: Kruskal's problem in interactive tool2.

### 3.4.4 Prim's Algorithm

Exercise number four covered the Prim's Minimum Spanning Tree algorithm.

For the question phases of the exercise, the following question structure was used:

#### **Question**

Given the following graph, perform Prim's algorithm starting at vertex A. For the answer, provide for each vertex a comma-separated list of all its intermediately stored weights (keys)(e.g.: Vertex A: 50, 9, 2)

#### **Answer format**

For each vertex, a simple text box was provided for the student to enter the answer string.

For the animation tool, animation tool2 was used. Using this tool, students could observe the animation of the Prim's algorithm (as well as see the flow through the pseudocode) being performed on graphs constructed by the student.

For the interactive tool, interactive tool2 was used. In this tool, students were presented with the randomly generated graph and were asked to direct the tool to perform the Prim's algorithm on that graph. Student had to direct the tool by first clicking the correct vertices that needed to be updated. Once all required vertices were updated, student needed to click on the correct edge to be added to the tree. Figure 15 below shows an example of the Prim's problem.

Exercise Grading

Undo Redo Redraw New Graph Submit

### Prim's Algorithm

Repeat the following until the spanning tree is complete:

1. Click on each of the vertices that need updating. Note that in Prim's algorithm, vertices to be updated are neighbors of the vertex most recently added to the tree. Therefore, **clicking on a non-adjacent vertex has no effect**. The value to be stored is the weight of the connecting edge.
2. Once all vertices that need updating are updated, click on an edge to add it to the tree. Note that **clicking on an edge not connected to the tree has no effect**.

Color hints:

- being considered
- in the tree
- not in the tree

Done

Figure 15: Prim's algorithm in interactive tool2.

### 3.4.5 Bellman-Ford Algorithm

Exercise number five covered the Bellman-Ford Shortest Path algorithm.

For the question phases of the exercise, the following question structure was used:

#### Question

Given the following graph, perform Bellman-Ford algorithm starting at vertex A and assuming that in each iteration the edges are relaxed in the following order: BC,AF,AD,CB,DB,DC,DE,FD,FE,ED

Provide for each vertex its intermediate weights as well as the iteration number when that weight was set.

Provide the answer in the format: weight#iteration. For example, if vertex B had weight of 5 set during iteration 1, and then weight of 4 and then 1 set during iteration 2, the format would be: 5#1, 4#2, 1#2

Note that initial weights are already provided for you in the answer.

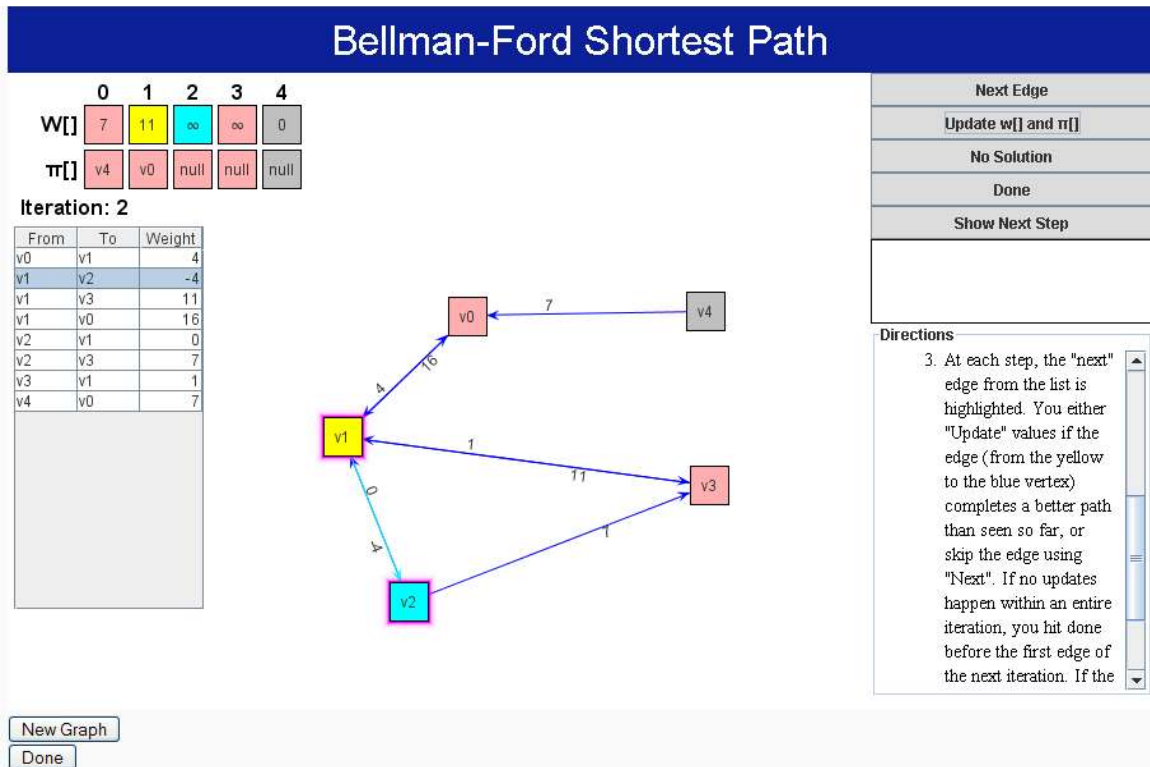
Also provide a comma-separated list of edges of the shortest path's tree in the order in which the algorithm adds them (e.g.: AD, DC, DB)

**Answer format**

For each vertex, a simple text box was provided for the student to enter the answer string. Initial weight 0#0 was provided for vertex A, and initial weight inf#0 was provided for all other vertices.

For the animation tool, animation tool3 was used. Using this tool, students could observe the animation of the Bellman-Ford algorithm on the pre-made graph or students could construct their own graphs and watch the algorithm being performed on those graphs.

For the interactive tool, interactive tool1 was used. In this tool, students were presented with the randomly generated graph and were asked to direct the tool to perform the Bellman-Ford algorithm on that graph. Student had to direct the tool by either selecting "Next" for the tool to pick the next edge or by selecting "Update" to update the vertex weight if the current edge completed a better path than seen so far. Once no updates have happened through the full iteration through the edges, the student had to pick "Done" to signal the termination of the algorithm to the tool. Figure 16 below shows an example of the Bellman-Ford algorithm.



**Figure 16: Bellman-Ford algorithm in interactive tool1.**

### 3.4.6 Dijkstra's Algorithm

Exercise number six covered the Dijkstra's Shortest Path algorithm.

For the question phases of the exercise, the following question structure was used:

**Question**

Given the following graph, perform Dijkstra's algorithm starting at vertex A.

For the answer, provide for each vertex a comma-separated list of all its intermediately stored weights (e.g.: Vertex A: 50, 9, 2).

Also provide a comma-separated list of edges of the shortest path's tree in the order in which the algorithm adds them (e.g.: AB, BC, CD)

**Answer format**

For each vertex, a simple text box was provided for the student to enter the answer string. Additional text box was provided for the student to enter the edges of the shortest path tree.

For the animation tool, animation tool3 was used. Using this tool, students could observe the animation of the Dijkstra's algorithm being performed on the pre-made graph or students could construct their own graphs and watch the algorithm being performed on those graphs.

For the interactive tool, interactive tool2 was used. In this tool, students were presented with the randomly generated graph and were asked to direct the tool to perform the Dijkstra's algorithm on that graph. Student had to direct the tool by first highlighting an incident edge from the current vertex by clicking on the edge once. Then, if the vertex at the other end of the edge needed to be updated, student had to click on that vertex, otherwise, student had to click on the edge to eliminate it from the shortest path tree. Once current vertex processing was done, student had to select the next vertex to process. Figure 17 below shows an example of the Dijkstra's problem.

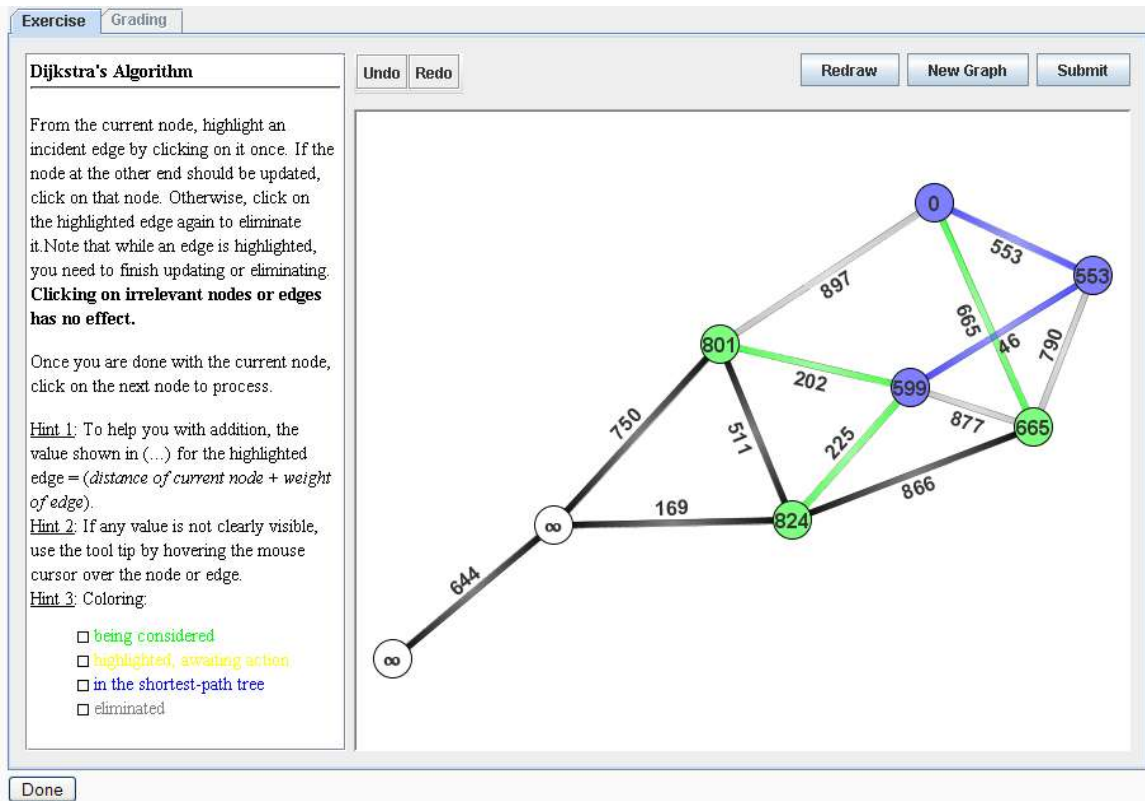


Figure 17: Dijkstra's algorithm in interactive tool2.

## 4 Results and Analysis

In this section we present the data collected from the experiment and describe the statistical analysis we conducted. The discussion and conclusions we drew from the analysis are presented in the sections that follow.

For each exercise of the experiment we collected the following statistics:

- **Answer score for each of the three questions**  
Answers were graded by the professor teaching the CS146 course. Exercises were graded without considering which tool (animation or interactive tool) was seen first for that student. All scores had an integer value from 0 (zero) to either 3 (three) or 6 (six) depending on the exercise. Partial credit was given as deemed appropriate by the professor. The following exercises had a maximum score of 3: Breadth-first Search, Kruskal's algorithm, Prim's algorithm. Other exercises had a maximum score of 6.
- **Time spent on each tool phase**  
Time (in seconds) for the duration of each phase was saved. The time for tool phases (phase 2 and phase 4) were of the most interest to us.

Complete data and graphs for our experiment and analysis are presented in Appendix C. It takes many pages, and therefore, to avoid exploding this section with tables and graphs, we only present a subset of the data here – mostly to provide some visual support for the topics covered in this section.

In summary, we performed the following analysis for each of the six exercises:

- For each exercise, we moved students results into two groups. First group was for the students who happened to see the interactive tool first (i.e. Interactive tool was picked for phase 2 of the exercise, and animation was used in phase 4). We will refer to this group as **interaction group**. The second group was for the students who saw the animation first (i.e. animation was picked for phase 2 of the exercise, and interactive tool was used in phase 4). We refer to this group as **animation group**.
- We created multiple sets of data, each set representing different counting approach. One set contained data for all students which took the exercise, even though some of them might not have answered all the questions. We treated such skipped questions as having a score of 0 (for example, assuming that question was skipped due to not knowing how to answer, and thus this could be treated as if the question was answered incorrectly with no partial credit). We refer to this data set as **complete set**. Note that we find this set to be of limited usefulness to our study, because we cannot be certain that the question was skipped by the student due to not knowing how to answer. However, we still performed some analysis on this data set,

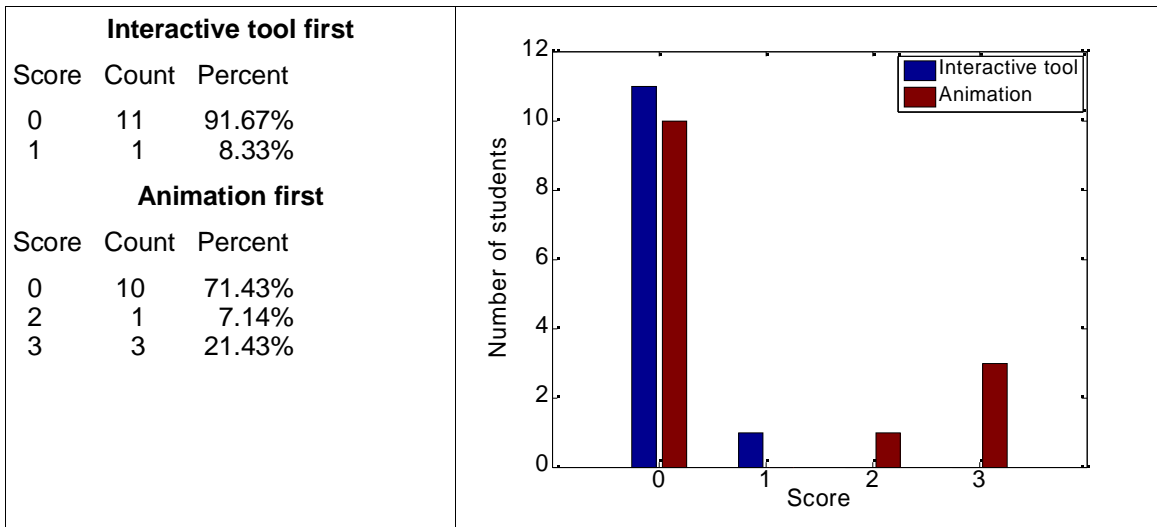
mostly for completeness reasons.

The second set contained data only for students who took the exercise and had answered all of the questions. That is, we eliminated the data for the students who skipped at least one question. We refer to this set as **clean set**.

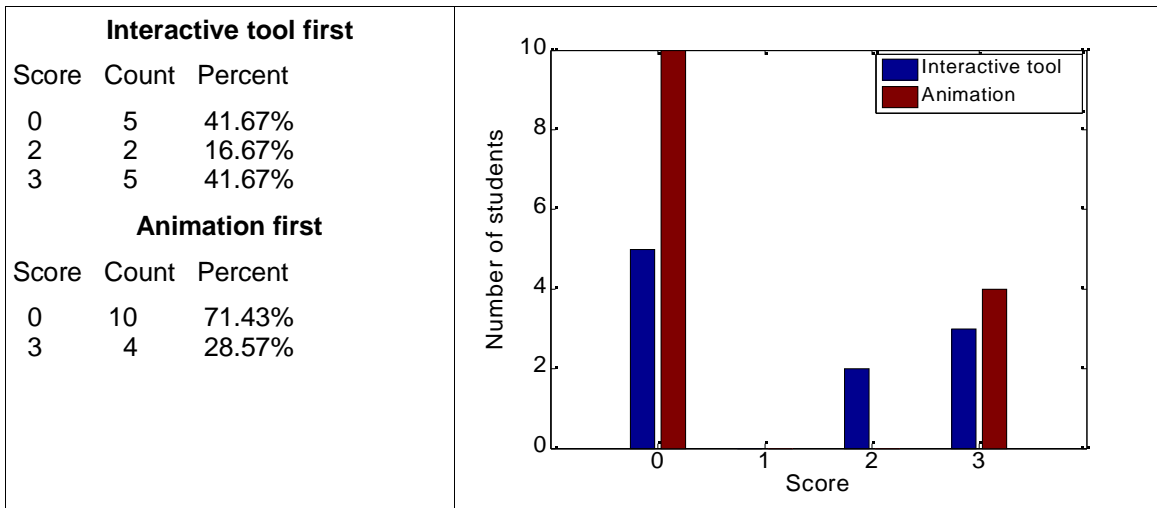
- We further categorized the data into **normalized** and **un-normalized** sets. Un-normalized set contain the data “as is”. That is, the score values were integers from 0 to 3 (or 0 to 6 for some exercises) as originally assigned by the professor's grading. Normalized set contained the data normalized in such a way that the mean for all scores for the exercise was 0 and the standard deviation was 1. We did this by first calculating the mean  $m$  and standard deviation  $s$  for the entire set (both groups) and then changing each score to new score' =  $(\text{score} - m) / s$ .
- We calculated the mean, mean confidence interval, and standard deviation of the answer scores for each of the three questions. We did this for each group and each set. In other words we ran the calculations four times per group per question:
  1. Un-normalized data, complete set
  2. Un-normalized data, clean set
  3. Normalized data, complete set
  4. Normalized data, clean set (this is our preferred set)
- We calculated mean, mean confidence interval, and standard deviation of the answer scores for all exercises combined. This was done on the normalized data, for complete and clean sets.
- We calculated mean, median, and trimmed mean for the time spent on each tool by each student.
- We performed a t-test to see the significance of the difference between means for Question 1 and Question 2 of the combined exercises.
- We looked at the ratio of score improvement over the maximum possible improvement for a given exercise.
- We ran the correlation test to find out if there is a correlation between the time spent on the tool and the score improvement.



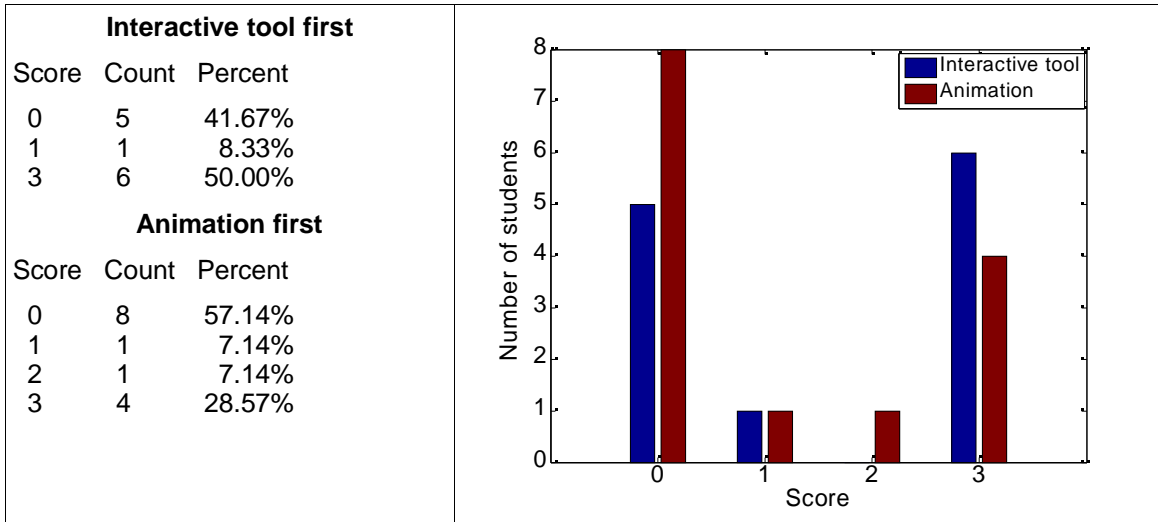
Appendix C contains the data tables and histogram graphs for all six exercises. Here, for the sake of brevity, we cover only one of the exercises, the Prim's algorithm. Figures 18, 19, and 20 below show the tabulated data and the histograms for questions 1, 2, and 3 accordingly (clean data set). The data is split into two groups: students who saw interactive tool first (i.e. phase 2 of the exercise) and students who saw the animation first (i.e. Interactive tool was seen in phase 4 of the exercise).



**Figure 18: Data and the histogram for Prim's exercise, Question 1**

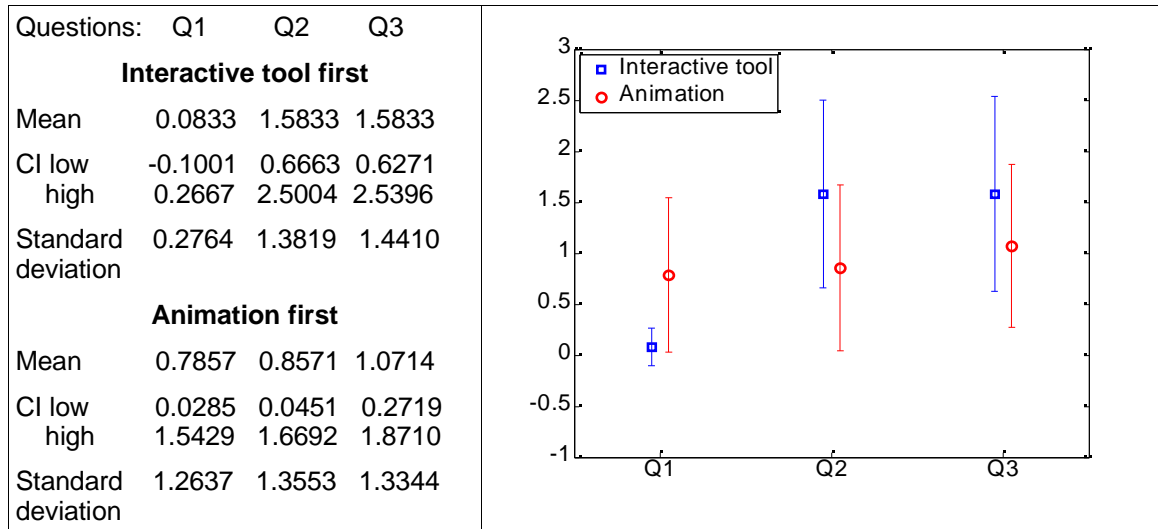


**Figure 19: Data and the histogram for Prim's exercise, Question 2**



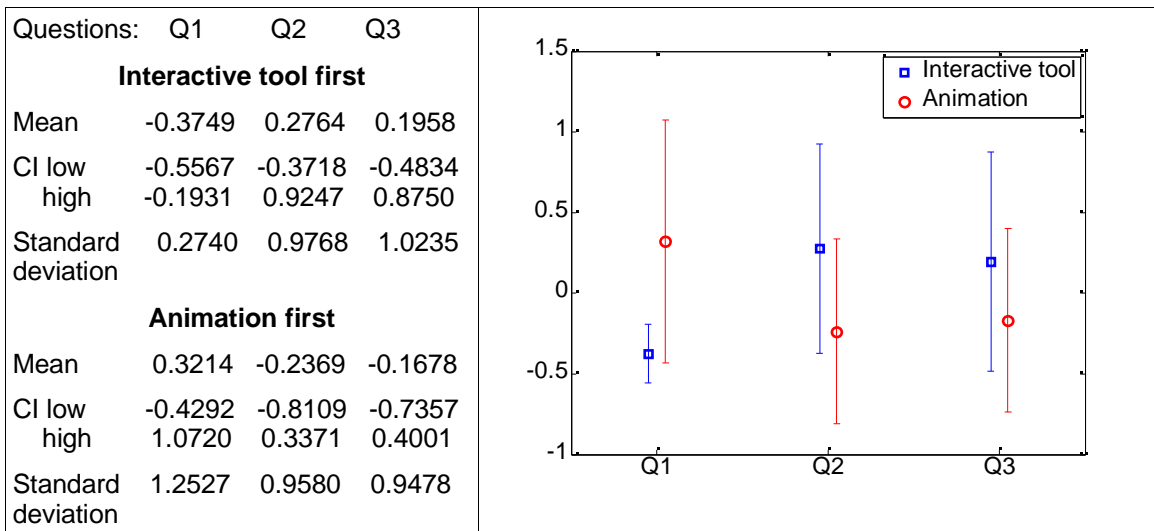
**Figure 20: Data and the histogram for Prim's exercise, Question 3**

Figure 21 shows the mean, confidence interval for mean, and standard deviation for the Prim's exercise.



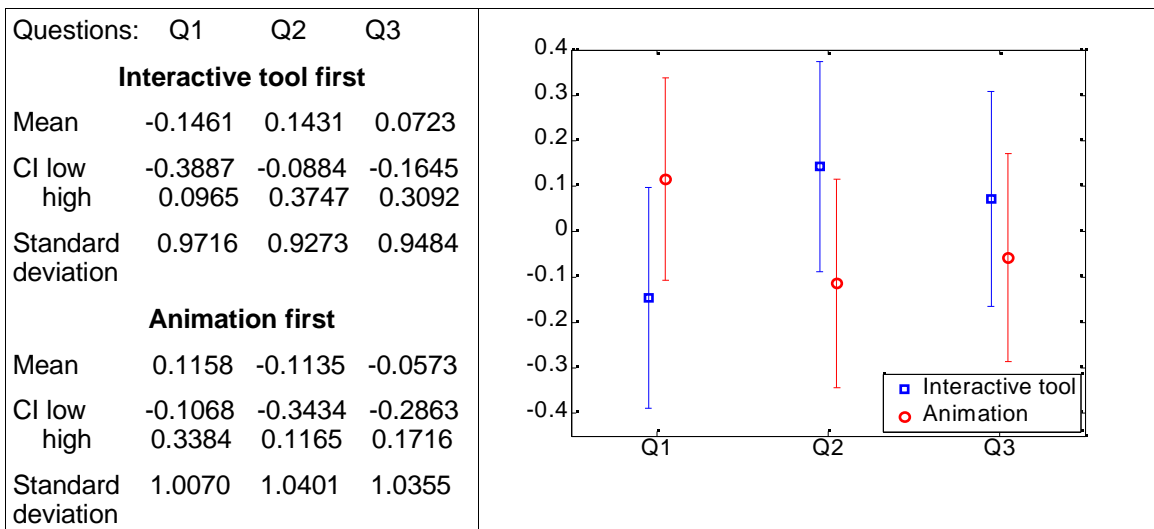
**Figure 21: Statistics for Prim's algorithm exercise**

Next, we normalized the data for each exercise so that the mean would be zero and the standard deviation would be one. This allowed us to combine the data from all six exercises into one data set. Figure 22 shows the mean, confidence intervals, and standard deviation for normalized version for Prim's exercise (normalized version of Figure 21).



**Figure 22: Normalized version of data in Figure 21**

Figure 23 shows the mean, confidence interval, and standard deviation for all six exercises combined.



**Figure 23: Statistics for all experiments combined**

Figure 23 is the most important result of our project. It demonstrates that students who used the interactive tool first significantly improved after using the tool. We performed a t-test to find the significance of this improvement. That is, we ran a t-test on the difference between the mean score for Question 1 and the mean score of Question 2. The results are shown in the Table 2 below. The results indicate that the null hypothesis can be rejected at the common significance level  $\alpha = 0.05$  (can be rejected even at  $\alpha = 0.02$  significance level), since the *p-value* falls below  $\alpha$ , and the 95% confidence interval on the mean does not contain 0.

(*p-value* is the probability, under the null hypothesis, of observing a value as extreme or more extreme than the test statistic).

t statistic value	<i>p-value</i>	Degrees of freedom	Standard deviation	Confidence interval on mean
-2.3989	0.0194	64	0.9721	[-0.5301, -0.0484]

**Table 2: Result of the t-test**

When analyzing the data for each exercise, we noticed that for some exercises the students don't start out with relatively equal scores. For instance, in exercise 1, the Question 1 mean score for the interactive tool group is 2.5385 and the mean score for the animation group is 1.8125. Exercise 3 is the reverse situation where the mean score for the interactive tool group is 1.8182 and the one for the animation group is 2.7333. Also, the number of students in the two groups is not the same. Thus, one way to get a rough idea about the effects of the tools is to look at the ratio of score improvement over the maximum possible improvement. That is, we look at the result of:  $(MQ2 - MQ1) / (N - MQ1)$ , where MQ1 is the mean score for Question 1, MQ2 is the mean score for Question 2, and N is the maximum possible score for the exercise. The results of this calculations are shown in the Table 3 below. Note that when calculating this for all exercises combined, we doubled the scores for the exercises that had a maximum score of 3 so that the maximum score (the score of 6) becomes the same for all exercises.

Exercise	Interactive tool group	Animation group
1	0.1667	0.4737
2	0.3571	0.0294
3	0.6923	-0.5000
4	0.5143	0.0323
5	0.7500	0.2931
6	0.4231	0.1579
<b>All</b>	<b>0.4952</b>	<b>0.1689</b>

**Table 3: Result of the “possible improvement” statistic**

A slight modification to the above calculation is: (number of students who got less than 6 on Q1 but got 6 on Q2) / (number of students who got less than 6 on Q1), which shows the ratio of students who improved to the perfect score using the tool over the total number of students who could have improved to the perfect score. The result is in the Table 4 below.

Exercise	Interactive tool group	Animation group
1	$1/3 = 0.3333$	$4/8 = 0.5$
2	$1/9 = 0.1111$	$3/12 = 0.25$
3	$3/5 = 0.6$	$1/2 = 0.5$
4	$5/12 = 0.4167$	$1/11 = 0.0909$
5	$2/5 = 0.4$	$4/13 = 0.3077$
6	$4/11 = 0.3636$	$3/6 = 0.5$
<b>All</b>	<b><math>16/45 = 0.3556</math></b>	<b><math>16/52 = 0.3077</math></b>

**Table 4: Result of the “possible improvement” statistic**

Even though both calculations above are rather rough, they still seem to suggest that the interactive tool is more effective than the animation.

Complete data about the times for the tool phases is presented in Appendix C. Here, we provide a sample, again for the Prim's algorithm exercise. Figure 24 shows time spent by each student on the tool in phase 2 (i.e. time spent on the first tool seen by the student).

Times by each student (sorted)			Basic statistics for the times		
Student #	Interactive tool	Animation		Interactive tool	Animation
1	99	43	Mean	582.8333	511.7857
2	145	48	Median	446.0000	181.0000
3	210	76	Trimmed	499.7000	293.5000
4	218	84	mean (5%)		
5	397	137			
6	402	162			
7	490	173			
8	569	189			
9	577	296			
10	645	525			
11	1344	532			
12	1898	642			
13		658			
14		3600*			

\* - all times were capped at time = max(3600, time)

**Figure 24: Statistics for time spent on phase2 of the Prim's exercise**

We ran the correlation tests for each of the exercises, as well as for the combined exercises, however the results showed no correlation between the time spent on the tool and the change in the score. The data for the correlation coefficients is in the Table 5 below. The only significant correlations found were for interactive tool in exercise 2 and the animation tool in exercise 6. The data for combined exercises does show a significant correlation for the interactive tool, however, the relation is rather weak (0.3) and is likely due to the strong correlation (0.77) for the exercise 2 offsetting the rest of the exercises.

Exercise	Interactive tool group coefficient ( <i>p-value</i> )	Animation group coefficient ( <i>p-value</i> )
1	0.2199 (0.4703)	-0.1950 (0.4692)
2	0.7703 (0.0055)	-0.4210 (0.1520)
3	-0.0119 (0.9723)	-0.0646 (0.8190)
4	-0.0686 (0.8323)	0.0459 (0.8761)
5	0.5801 (0.3052)	0.3586 (0.2081)
6	0.0387 (0.9002)	0.7173 (0.0195)
<b>All</b>	<b>0.3033 (0.0140)</b>	<b>0.1066 (0.3406)</b>

**Table 5: Result of the correlation test**

## 5 Discussion

In general, we found the results very encouraging. They seem to support our hypothesis that active interactive tools are more effective in student learning of algorithms than animation (passive interaction). Looking at the data for individual exercises, for some of them we were not always certain whether or not the data supports our hypothesis, however, for others, as well as looking at all exercises combined, we do see a trend that favors our hypothesis.

Below, we present a table which summarizes the core statistics for our experiment. The summary is for the normalized, clean data. The table includes Percentage Increase (PI) value, which is the ratio of the score improvement from one question to the next over the total possible improvement. The PI was calculated using the formula:  $PI_n = (MQ_{n+1} - MQ_n) / (Q_{max} - MQ_n)$ , where  $MQ_n$  is the Question n score mean,  $MQ_{n+1}$  is the Question n+1 score mean, and  $Q_{max}$  is the maximum possible score. Some comments and discussion follows.

<b>Exercise</b>	<b>Groups</b>	<b>N</b>	<b>MQ1</b>	<b>MQ2</b>	<b>MQ3</b>	<b>PI1</b>	<b>PI2</b>
<b>1</b>	<b>Interaction</b>	13	0.33	0.14	0.09	0.17	-0.2
	<b>Animation</b>	16	-0.27	-0.11	-0.08	0.47	0
<b>2</b>	<b>Interaction</b>	11	0.02	0.23	0.19	0.36	-0.11
	<b>Animation</b>	13	-0.02	-0.2	-0.16	0.03	0.03
<b>3</b>	<b>Interaction</b>	11	-0.45	0.03	0.02	0.69	0.5
	<b>Animation</b>	15	0.33	-0.02	-0.01	0.5	0.5
<b>4</b>	<b>Interaction</b>	12	-0.37	0.28	0.2	0.51	0
	<b>Animation</b>	14	0.32	-0.24	-0.17	0.03	0.1
<b>5</b>	<b>Interaction</b>	5	0.06	0.61	0.36	0.75	-0.4
	<b>Animation</b>	14	-0.02	-0.22	-0.13	0.29	0.17
<b>6</b>	<b>Interaction</b>	13	-0.37	-0.13	-0.23	0.42	-0.2
	<b>Animation</b>	10	0.48	0.17	0.29	0.16	-0.06
<b>All</b>	<b>Interaction</b>	65	-0.15	0.14	0.07	0.5	-0.08
	<b>Animation</b>	82	0.12	-0.11	-0.06	0.17	0.1

**Table 6: Statistics summary (N – number of students in the group, |  
 MQ1 – Question 1 score mean,  
 MQ2 – Question 2 score mean,  
 MQ3 – Question 3 score mean,  
 PI1 – PI for Question 1 to Question 2  
 PI2 – PI for Question 2 to Question 3 )**



### **5.1 Exercise 1 (Breadth-First Search)**

Exercise 1 (as well as Exercise 3) is one of the simplest of all six exercises. Hence, it was not surprising to see many students getting the very first question correct. Unfortunately, the Question 1 score distribution for this exercise was significantly skewed. A majority of students in the interactive group (76.92% or 10 students) got the 1<sup>st</sup> question correct. Only 50% (8 students) in the animation group answered correctly. Looking at the data for this exercise (see Table 6 above and Appendix C, section C1), our analysis does not show the interactive tool to be significantly more effective than the animation tool for this exercise. However, the data shows that the interactive tool still had a positive effect on student learning.

### **5.2 Exercise 2 (Depth-First Search)**

Our analysis shows that the interactive tool was likely more effective than the animation for this particular exercise. This observation is supported by analyzing the data set mean (normalized clean data set). Both groups performed relatively equally on Question 1, however, the interactive group did significantly better than the animation group on Question 2 (after using the tool). Further, the Percentage Increase (see PI1 column in the Table 6 above) for the interactive group is an order of magnitude higher than that of the animation group.

### **5.3 Exercise 3 (Kruskal's algorithm)**

Similar to Exercise 1, this exercise is one of the simplest of all six exercises. Just as in Exercise 1, we see many students answered correctly to Question 1, except this time the skew is in the animation group. 13 students (86.67%) answered correctly in the animation group, whereas only 6 students (50%) answered correctly in the interaction group. Our analysis shows that the interactive tool was likely more effective than the animation for this particular exercise. The interactive group started out with much worse scores than the animation group, however, after using the tool, the score mean for the interaction group became higher than the mean of the animation group. Surprisingly, Percentage Increase (see PI columns in the Table 6 above) for the animation group is negative, indicating that the group's scores actually got worse after using the tool. This may be a side effect of the small sample size, or perhaps the tool somehow confused the students.

### **5.4 Exercise 4 (Prim's algorithm)**

For this exercise, a lot of students received a score of 0 on the Question 1; 11 students (91.67%) in the interaction group and 10 students (71.43%) in the animation group. Similar to most of our other exercises, data analysis suggests that the interactive tool was likely more effective than the animation.

### 5.5 Exercise 5 (Bellman-Ford algorithm)

For this exercise, we got a somewhat unexpected imbalance in the number of students for the two groups (5 vs. 14). However, in line with most other exercises (especially Exercise 2), the data seems to indicate that the interactive tool was more effective than animation for this exercise.

### 5.6 Exercise 6 (Dijkstra's algorithm)

Data for this exercise is again somewhat unbalanced. 7 students (53.85%) in the interaction group had score of 0 for Question 1, while the animation group had only 1 student (10%) with zero score and 7 students (70%) did better than average (2.9). The analysis shows, again, the interactive tool being slightly more effective than animation.

### 5.7 Combined Exercises

Analyzing the data for all exercises combined reinforces the trend we observed from looking at the individual exercises. Comparing the normalized mean scores for questions 1 and 2 for both groups, looking at the Percentage Increase (see PI columns in the Table 6 above), as well as several other statistics, all demonstrate the apparent effectiveness of the interactive tool over animation. We next consider how many students scored above average (normalized score > 0) for each question. Figure 25 below gives such a histogram. We can see that interaction group had a significant improvement (about 20%) going from Question 1 to Question 2, whereas the animation group has very little change.

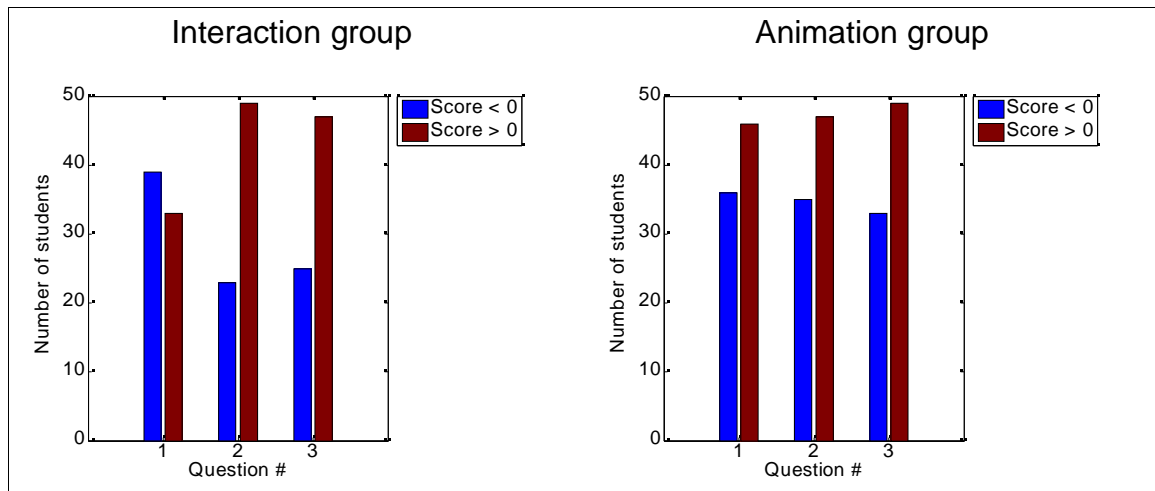


Figure 25: Score histogram for all six exercises combined

We also combined the exercises in two other ways. We combined exercises 1 and 5 (where interactive tool1 was used), and then exercises 2, 3, 4, and 6 (where interactive tool2 was used). This was done for our secondary analysis about the versions of the interactive tool. These combinations are not necessary to our main analysis but could be useful to the interactive tool designers at SJSU. The data for these combinations can be found in Appendix C. The analysis shows that interactive tool2 appears to be significantly more effective than the animation, and interactive tool1 looks very similar but slightly better in effectiveness than the animation. However, results for interactive tool1 are less certain because: 1) only two exercises used the tool (only 18 data points), and 2) one of the exercises was the BFS, which we believe may be too simple for a small sample size we had available to our experiment.

## 6 Limitations of the study

There were several limitations in our experiment that we would like to mention.

- Most importantly, we had a relatively small sample size. Due to time constraints, it was difficult for us to come up with a broader audience for our study. As the result, some results are not certain. For instance, exercise 5 had only 5 students in the interaction group. It is difficult to infer from such a small sample size. However, combining all exercises together gave us a reasonable sample size of 147, and we are more certain about the conclusions we drew from analyzing this data set.
- Some exercises might have been too simple for the experiment (for instance, perhaps the questions were not hard enough or the nature of the algorithm was trivial), resulting in majority of students receiving a perfect score on all three questions. BFS and Kruskal's exercises certainly appear to fall into this category.
- Ideally, we would have liked to provide a basic tutorial for all the tools used in our experiment so that it would be more clear to every participant how to operate the tools. This would minimize possible confusion or questions about the directions. Due to time constraint, we were unable to provide such assistance.
- Again due to the time allocated for the project, we could not create our own animations or modify some existing open source animations to suit our needs. As the result, we had to pick three different animation applets and did not have the ability to collect more detailed statistics about the animation use (e.g. Number of problem instances ran, number of graphs created by the user, etc.).
- For each exercise, the information content was different between the animation and the interactive tool. Given more time, we could have used the animation only mode of the tool1 framework as the animation tool for the exercises.

We believe these limitations are not critical to the experiment in the sense that they do not invalidate the results of the study, although removing them would add to the certainty of the study's results.

## 7 Conclusion and future work

Based on the data we have collected and our analysis, we have an indication that the active interaction tools are more effective in learning graph algorithms than the passive interaction tools. On average, students who used active interaction tool first, demonstrated a bigger improvement in understanding of the algorithm when compared to the students who used passive interaction (animation) tool. Our analysis shows that this difference is significant. Furthermore, students in the animation group improve more than the interactive group from Question 2 to Question 3, when the animation group got the interactive tool.

Clearly, there is much more that could be done to advance our study. Sometimes, it takes months (perhaps years) to fine tune and improve a study. After analyzing the data we have collected, we definitely learned more about what it takes to design a solid experiment. If we were to conduct the experiment again, there are certain things we would change and not do again. For instance, we would attempt to ensure a more even distribution among the two groups (interactive group and animation group). We would attempt to get a bigger sample size or include more exercises. Perhaps the question phase could be redesigned so that the questions are not fixed and would be randomly chosen. We would attempt to use the **same** presentation for both the animation and the interactive tool (for instance, this could be done by using tool1 without interactive feature enabled for animation and with the interactive feature enabled for the interactive tool).

Also, even with the data we have collected, many more interesting statistics could be calculated, such as: different correlation tests that involve time spent on each question and the scores, comparing Question 2 and Question 3 results (for instance, assuming that animation was used in phase 2, is there still a significant benefit for using the interaction tool in phase 4, etc.). We would also like the interactive tool to be more sophisticated before conducting more tests. For instance, TRAKLA2 is now more developed than when we started (it has larger library of problems, has more interactive features like answer feedback, etc.). Nevertheless, the results of this study are certainly promising enough to encourage further studies.

## 8 References

- [1] Tversky, B., Morrison, J., B. Animation: can it facilitate?, *International Journal of Human-Computer Studies*, v.57 n.4, p.247-262, October 2002
- [2] Michael D. Byrne , Richard Catrambone , John T. Stasko, Evaluating animations as student aids in learning computer algorithms, *Computers & Education*, v.33 n.4, p.253-278, December 1999
- [3] Shaffer, C. A., Cooper, M., & Edwards, H. S., Algorithm Visualization: A Report on the state of the field. *Proceedings of the thirty-eighth SIGCSE technical symposium on Computer science education*, p. 150-154, March 07-11, 2007, Convington, Kentucky, United States
- [4] Ferguson, E. L. & Hegarty, M. (1995). Learning with real machines or diagrams: application of knowledge to real-world problems. *Cognition and Instruction*,13, 129-160.
- [5] <http://algoviz.cs.vt.edu/AlgovizWiki/>
- [6] Stasko, J.T., Badre, A., & Lewis, C. (1993). Do algorithm animations assist learning? An empirical study and analysis. *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, 61-66.
- [7] Levin, J. R. & Lesgold, A. M. (1978). On pictures in prose. *Educational Communication and Technology*, 26, 233–243.
- [8] Levie, W. H. & Lentz, R. (1982). Effects of text illustrations: a review of research. *Educational Communication and Technology*, 30, 195–232.
- [9] Mandl, A. & Levin, J. Eds. (1989). *Knowledge Acquisition from Text and Pictures*. Amsterdam: North-Holland.
- [10] John Stasko , Albert Badre , Clayton Lewis, Do algorithm animations assist learning?: an empirical study and analysis, *Proceedings of the INTERCHI '93 conference on Human factors in computing systems*, p.61-66, May 1993, Amsterdam, The Netherlands
- [11] Park, O., & Gittelmann, S.S. (1992). Selective use of animation and feedback in computer-based instruction. *Educational Technology Research & Development*, 40(4), 27-38.

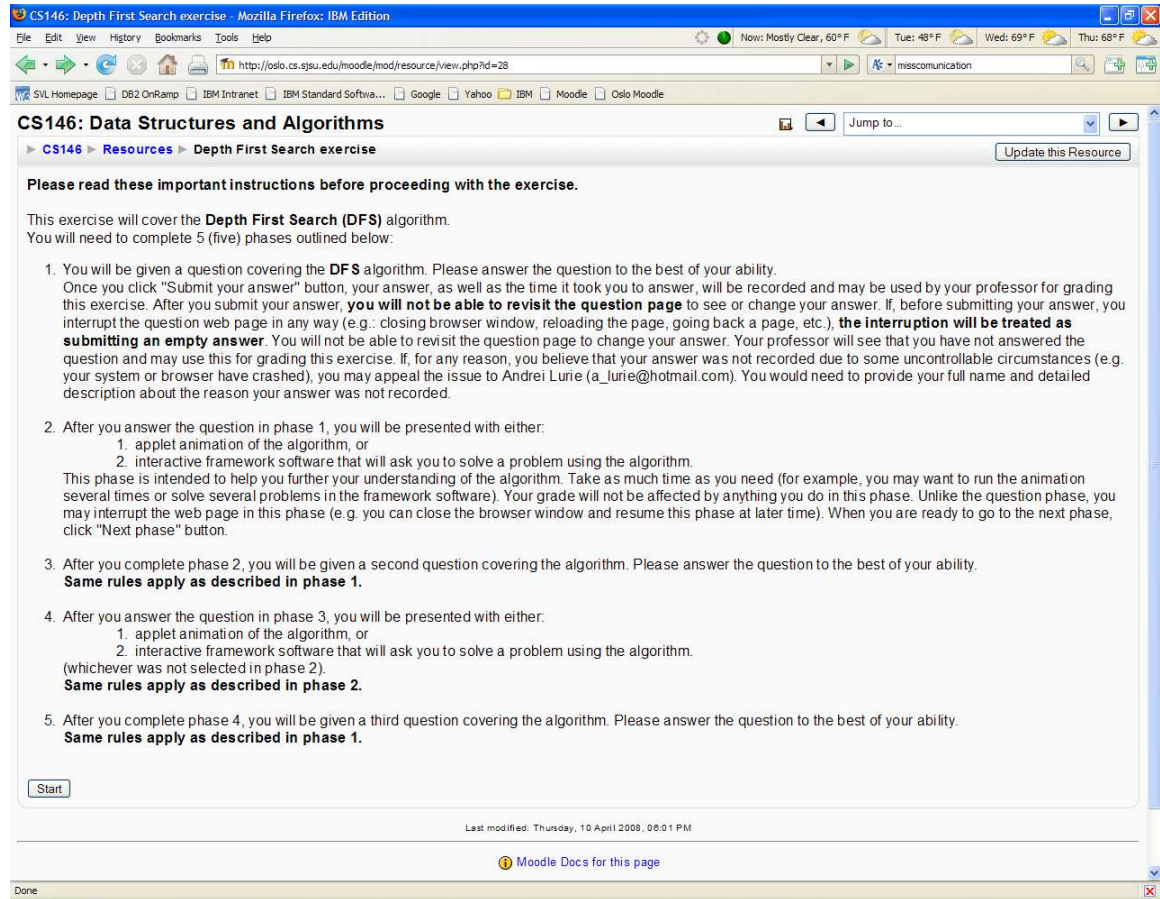
- [12] Duane J. Jarc , Michael B. Feldman , Rachelle S. Heller, Assessing the benefits of interactive prediction using Web-based algorithm animation courseware, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, p.377-381, March 07-12, 2000, Austin, Texas, United States
- [13] Lawrence, A.W., Badre, A.M., & Stasko, J.T. (1994). Empirically evaluating the use of animations to teach algorithms. *Proceedings of the 1994 IEEE Symposium on Visual Languages*, 48-54.
- [14] Gurka J., & Citrin, W. (1996). Testing Effectiveness of Algorithm Animation. *Proceedings of the IEEE International Symposium on Visual Languages*, 182-189.
- [15] <http://nova.umuc.edu/~jarc/idsv>
- [16] <http://www.cs.hut.fi/Research/TRAKLA2/index.shtml>
- [17] Sean Sharma (2008). A Framework for active learning. Thesis, SJSU.
- [18] <http://www.lupinho.de/gishur/html/BFSApplet.html>
- [19] <http://www.mincel.com/java/prim.html>
- [20] <http://links.math.rpi.edu/applets/appindex/graphtheory.html>
- [21] <http://moodle.org/>

## Appendix A: Example of the exercise flow

This appendix contains a collection of the screen shots that illustrate the exercise flow as described in section 3.4.

Figures are presented in the same order as would be seen by the exercise participant. Depth-First search exercise was used as an example, however the flow of all exercises is identical.

Figure A1, shows the introduction page.



The screenshot shows a web browser window with the following content:

- Browser title: CS146: Depth First Search exercise - Mozilla Firefox: IBM Edition
- Address bar: <http://oslo.cs.sjsu.edu/moodle/mod/resource/view.php?id=28>
- Page title: CS146: Data Structures and Algorithms
- Breadcrumbs: CS146 > Resources > Depth First Search exercise
- Buttons: Update this Resource, Jump to...
- Text: **Please read these important instructions before proceeding with the exercise.**
- Text: This exercise will cover the **Depth First Search (DFS)** algorithm. You will need to complete 5 (five) phases outlined below:
- List of 5 phases with detailed instructions and rules for each phase.
- Start button
- Footer: Last modified: Thursday, 10 April 2008, 08:01 PM; Moodle Docs for this page

Figure A1: introduction page



Figure A2 shows the pre-question page for the Question 1.

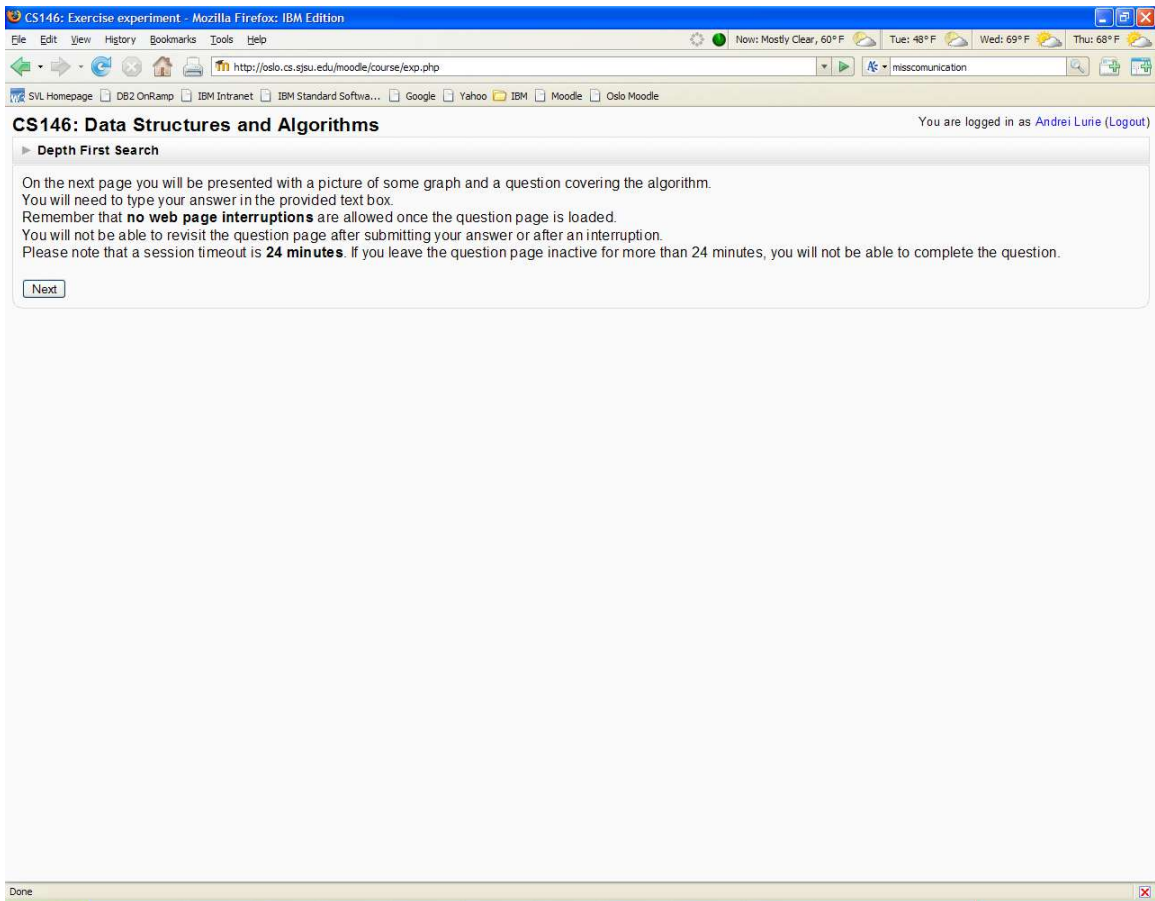


Figure A2: pre-question page

The two figures on the following page show the Question 1 page. The full page could not be fit on the screen, thus, we split it into two screen shots. The first figure shows the top portion of the page (with the question and the graph) and the second figure shows the rest of the page (with answer form and the submit button).

Since the pages for Question 2 and Question 3 have identical answer format, only the top portion of the page (with the question and the graph) will be shown for these questions.

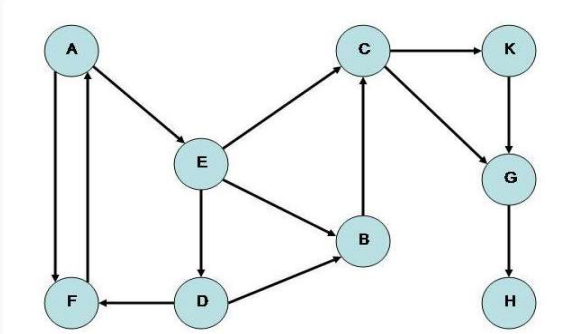
**CS146: Data Structures and Algorithms**

You are logged in as Andrei Lunie (Logout)

**Depth First Search**

**Question 1:**

Given the following graph, perform a depth first search starting from vertex A (break all ties alphabetically). Compute discovery/finish time for each vertex, and label each edge as tree, back, forward, or cross.  
 Provide the answer in the form entries below.



Answer:

**Vertex discovery and finish times:**

Vertex A Discovery time:  Finish time:   
 Vertex B Discovery time:  Finish time:   
 Vertex C Discovery time:  Finish time:   
 Vertex D Discovery time:  Finish time:   
 Vertex E Discovery time:  Finish time:   
 Vertex F Discovery time:  Finish time:   
 Vertex G Discovery time:  Finish time:   
 Vertex H Discovery time:  Finish time:   
 Vertex K Discovery time:  Finish time:

Answer:

**Vertex discovery and finish times:**

Vertex A Discovery time:  Finish time:   
 Vertex B Discovery time:  Finish time:   
 Vertex C Discovery time:  Finish time:   
 Vertex D Discovery time:  Finish time:   
 Vertex E Discovery time:  Finish time:   
 Vertex F Discovery time:  Finish time:   
 Vertex G Discovery time:  Finish time:   
 Vertex H Discovery time:  Finish time:   
 Vertex K Discovery time:  Finish time:

**Edge classification:**

Edge AE Type:   
 Edge AF Type:   
 Edge BC Type:   
 Edge CG Type:   
 Edge CK Type:   
 Edge DB Type:   
 Edge DF Type:   
 Edge EB Type:   
 Edge EC Type:   
 Edge ED Type:   
 Edge FA Type:   
 Edge GH Type:   
 Edge KG Type:

Submit Your Answer

Figure A3 shows the page for the first tool. Recall that the tool type (interactive or animation) is chosen randomly. In this example, interactive tool was picked first.

The screenshot shows a web browser window titled "CS146: Exercise experiment - Mozilla Firefox: IBM Edition". The address bar shows the URL "http://oslo.cs.sjsu.edu/moodle/course/exp.php". The page content is titled "Depth First Search" and "Interactive Tool".

**Interactive Tool**

On the exercise page, hit 'Redraw' for a different layout of the same graph, hit 'New Graph' to try the exercise on a different graph, and hit 'Submit' or 'Grading' for feedback on your answer.  
On the feedback page, you can step through your moves and the correct moves, and then return to the Exercise page, to experiment more with the same graph or new graphs, by hitting the 'Exercise' or 'Return to Exercise' buttons.  
Hit 'Done' to finish using this tool.

**Exercise** | Grading

**Depth-First Search**

From your current node, classify an edge.  
**When there are more than one unmarked edge, choose the one with the least label value.** After marking an edge, determine whether to enter the node at the other end. When you're finished with a node, backtrack.

To discover a node, click on it once. To finish with that node, click on it again. The values in the node denote *discovery time* and *finish time*.

To classify an edge, click on it. Change the edge type by clicking on it again. Edge types:

- unknown
- tree
- forward
- back
- cross

Buttons: Undo, Redo, Redraw, New Graph, Submit

The graph consists of six nodes, each labeled with "??". The nodes are connected by edges with labels 1 through 6. Node 1 is at the top right, node 2 is to its left, node 3 is below node 2, node 4 is to the right of node 3, node 5 is to the left of node 3, and node 6 is below node 4. Edges are labeled: 1 (1-2), 2 (2-3), 3 (3-4), 4 (4-6), 5 (3-5), 6 (4-5).

Done

Applet edu.sjsu.cs.lpp.moodle.LPPApplet started

Figure A3: interactive tool

Figure A4 shows the pre-question page for Question 2

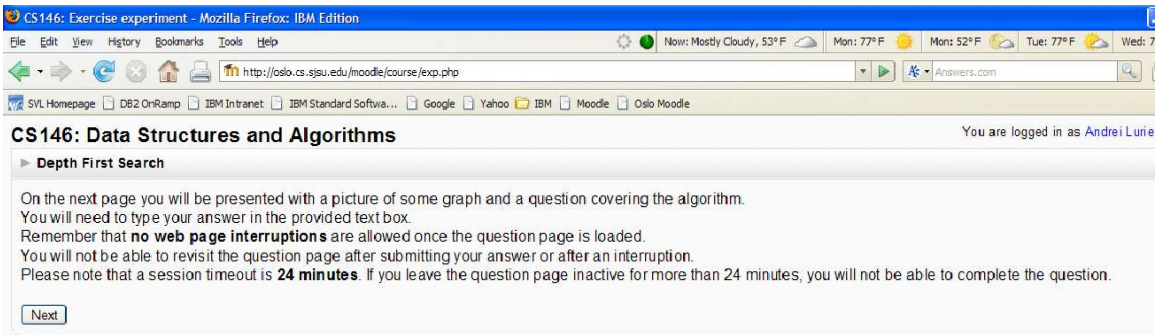


Figure A4: pre-question page

Figure A5 shows the top portion of the page for Question 2

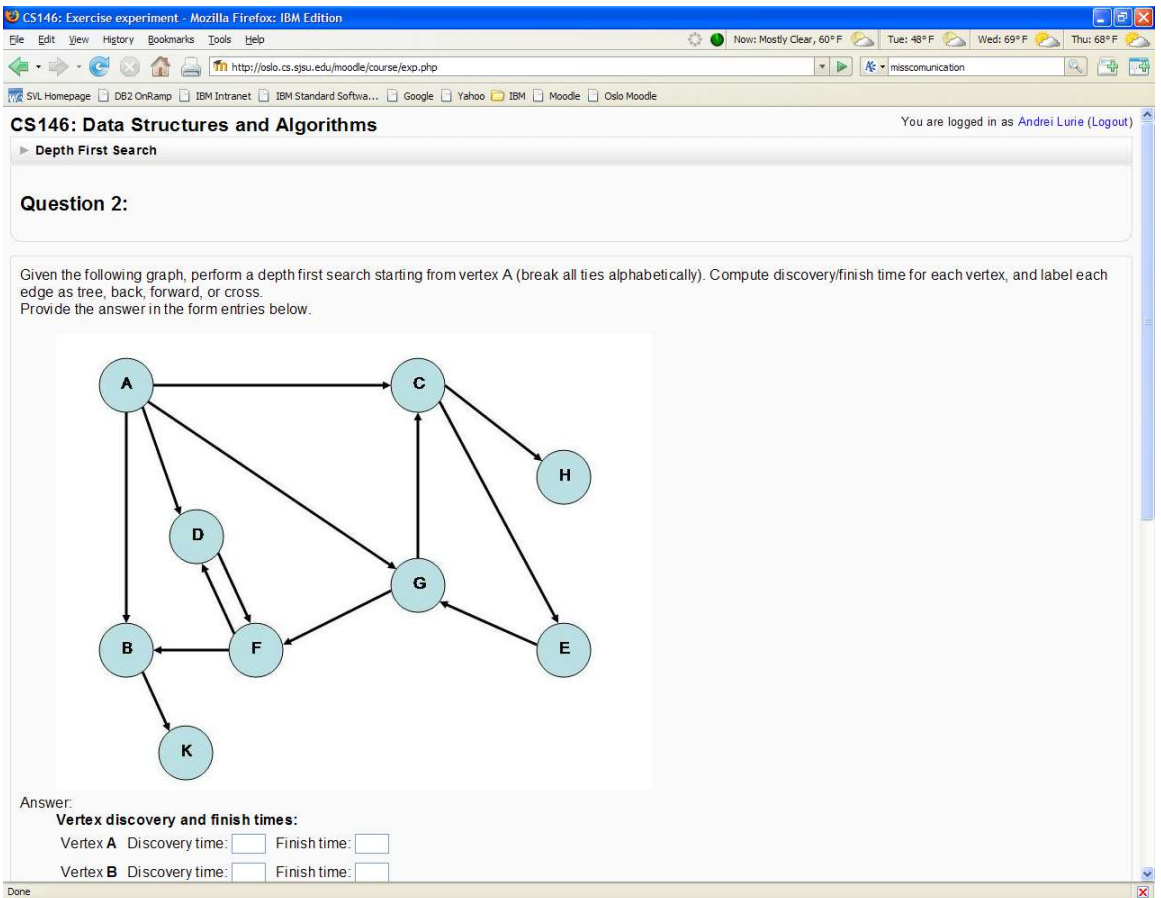


Figure A5: Question 2 page

Figure A6 shows the page for the second tool (which is an animation tool since the interaction tool was chosen as the first tool)

**CS146: Data Structures and Algorithms** You are logged in as [Andrei Lurie](#) (Logout)

▶ **Depth First Search**

**Animation Tool**

To work with predefined graph, select DFS from the drop down list '-- Pre-made graphs --' (top right corner of the applet). Since this is a Depth First Search exercise, please make sure you select **'Depth-first Search'** algorithm from the drop down list in the applet (under the 'Next' button). You can build your own graph by selecting a mouse action from the drop down list (to the left of the 'Clear' button). Once you select an action (default is 'Add Vertex'), clicking the mouse will perform that action. Use 'Clear' button to get fresh screen. You can either run the entire animation ('Auto' button) or step through the animation one step at the time ('Next' button). Click 'Done' button when you are done with this tool and are ready to go to the next phase of the exercise.

Next Auto Stop -- Pre-made graphs --

Ford-Fulkerson Algorithm Add Vertex Clear

© 1998 Rensselaer Polytechnic Institute

Done

**Figure A6: animation tool**

Figure A7 shows the pre-question page for Question 3

**CS146: Data Structures and Algorithms** You are logged in as [Andrei Lurie](#)

▶ **Depth First Search**

On the next page you will be presented with a picture of some graph and a question covering the algorithm. You will need to type your answer in the provided text box. Remember that **no web page interruptions** are allowed once the question page is loaded. You will not be able to revisit the question page after submitting your answer or after an interruption. Please note that a session timeout is **24 minutes**. If you leave the question page inactive for more than 24 minutes, you will not be able to complete the question.

Next

**Figure A7: pre-question page**

Figure A8 shows the top portion of the page for Question 3

CS146: Exercise experiment - Mozilla Firefox: IBM Edition

File Edit View History Bookmarks Tools Help

Now: Mostly Clear, 60° F Tue: 48° F Wed: 69° F Thu: 68° F

http://oslo.cs.sjsu.edu/moodle/course/exp.php

Syll. Homepage DB2 OnRamp IBM Intranet IBM Standard Softwa... Google Yahoo IBM Moodle Oslo Moodle

**CS146: Data Structures and Algorithms** You are logged in as Andrei Lurie (Logout)

▶ Depth First Search

**Question 3:**

Given the following graph, perform a depth first search starting from vertex A (break all ties alphabetically). Compute discovery/finish time for each vertex, and label each edge as tree, back, forward, or cross.  
Provide the answer in the form entries below.

```
graph TD; A((A)) --> C((C)); A((A)) --> G((G)); A((A)) --> D((D)); D((D)) --> B((B)); D((D)) --> F((F)); B((B)) --> F((F)); F((F)) --> G((G)); G((G)) --> C((C)); G((G)) --> H((H)); H((H)) --> F((F)); H((H)) --> E((E)); E((E)) --> K((K)); K((K)) --> E((E)); E((E)) --> G((G));
```

Answer:

**Vertex discovery and finish times:**

Vertex A Discovery time:  Finish time:

Vertex B Discovery time:  Finish time:

Vertex C Discovery time:  Finish time:

Vertex D Discovery time:  Finish time:

Vertex E Discovery time:  Finish time:

Vertex F Discovery time:  Finish time:

Vertex G Discovery time:  Finish time:

Vertex H Discovery time:  Finish time:

Vertex K Discovery time:  Finish time:

Figure A8: Question 3 page

Figure A9 shows the final page

The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://oslo.cs.sjsu.edu/moodle/course/exp.php`. The page title is "CS146: Data Structures and Algorithms" and the user is logged in as "Andrei Lurie".

The main content area shows a "Depth First Search" exercise completion message: "Congratulations. You have completed the exercise." Below this is a button labeled "Go back to the course page".

Below the message, there is a paragraph explaining that the user can reset the exercise. It states: "Because you have teacher privileges you can reset this exercise for your user id so that you could repeat the exercise. Once you click on 'Reset exercise' button all your statistics for the exercise will be deleted and you will be able to start the exercise anew." Below this is a button labeled "Reset exercise".

Below the paragraph, there is a section titled "Below is the Interactive tool for DFS algorithm in case you'd like to use it." This section contains an interactive tool with the following components:

- Depth-First Search** instructions:
  - From your current node, classify an edge. When there are more than one unmarked edge, choose the one with the least label value. After marking an edge, determine whether to enter the node at the other end. When you're finished with a node, backtrack.
  - To discover a node, click on it once. To finish with that node, click on it again. The values in the node denote *discovery time* and *finish time*.
  - To classify an edge, click on it. Change the edge type by clicking on it again. Edge types:
    - unknown
    - tree
    - forward
    - back
    - cross
- Graph**: A graph with 6 nodes and 9 edges. The nodes are labeled with "?/?". The edges are labeled with numbers 1 through 9. The graph is a directed graph with the following edges:
  - Node 1 (top-left) to Node 2 (top-right) with weight 8.
  - Node 1 to Node 3 (bottom-left) with weight 2.
  - Node 2 to Node 4 (bottom-right) with weight 3.
  - Node 3 to Node 4 with weight 9.
  - Node 4 to Node 5 (middle-right) with weight 6.
  - Node 5 to Node 1 with weight 1.
  - Node 5 to Node 2 with weight 5.
  - Node 5 to Node 4 with weight 7.
  - Node 4 to Node 5 with weight 8.
- Controls**: "Undo", "Redo", "Redraw", "New Graph", and "Submit" buttons.

The status bar at the bottom of the browser window shows "Applet edu.sjsu.cs.lpp.moodle.LPPApplet started".

CS146: Exercise experiment - Mozilla Firefox: IBM Edition

File Edit View History Bookmarks Tools Help

Now: Mostly Clear, 60° F Tue: 48° F Wed: 69° F Thu: 68° F

http://oslo.cs.sjsu.edu/moodle/course/exp.php

SVL Homepage DB2 OnRamp IBM Intranet IBM Standard Softwa... Google Yahoo IBM Moodle Oslo Moodle

CS146: Data Structures and Algorithms You are logged in as Andrei Lurie (Logout)

▶ Depth First Search

You have already completed this exercise.  
You are not allowed to repeat the exercise at this time.  
[Go back to the course page](#)

Because you have teacher privileges, you can reset this exercise for your userid so that you could repeat the exercise. Once you click on "Reset exercise" button all your statistics for the exercise will be deleted and you will be able to start the exercise anew.

[Reset exercise](#)

Below is the Interactive tool for DFS algorithm in case you'd like to use it.

Exercise Grading

Undo Redo Redraw New Graph Submit

**Depth-First Search**

From your current node, classify an edge. **When there are more than one unmarked edge, choose the one with the least label value.** After marking an edge, determine whether to enter the node at the other end. When you're finished with a node, backtrack.

To discover a node, click on it once. To finish with that node, click on it again. The values in the node denote *discovery time* and *finish time*.

To classify an edge, click on it. Change the edge type by clicking on it again. Edge types:

- unknown
- tree
- forward
- back
- cross

Applet edu.sjsu.cs.lpp.moodle.LPPApplet started

Figure A9: final page

Figures A10 and A11 illustrate a warning message that would appear if the question page were interrupted. This gave the students an additional warning besides the warnings on each pre-question page and on the first page of the exercise.



CS146: Exercise experiment - Mozilla Firefox: IBM Edition

File Edit View History Bookmarks Tools Help

Now: Mostly Clear, 60° F Tue: 48° F Wed: 69° F Thu: 68° F

http://oslo.cs.sjsu.edu/moodle/course/exp.php

SVL Homepage DB2 OnRamp IBM Intranet IBM Standard Softwa... Google Yahoo IBM Moodle Oslo Moodle

**CS146: Data Structures and Algorithms** You are logged in as Andrei Lurie (Logout)

▶ Depth First Search

**Question 1:**

Given the following graph, perform a depth first search starting from vertex A (break all ties alphabetically). Compute discovery/finish time for each vertex, and label each edge as tree, back, forward, or cross. Provide the answer in the form entries below.

Answer:

**Vertex discovery and finish times:**

Vertex A Discovery time:  Finish time:

Vertex B Discovery time:  Finish time:

Vertex C Discovery time:  Finish time:

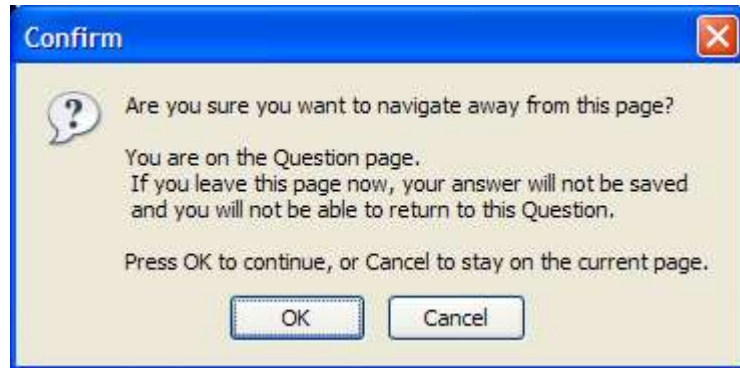
Vertex D Discovery time:  Finish time:

Vertex E Discovery time:  Finish time:

Vertex F Discovery time:  Finish time:

Done

**Figure A10: A warning message displayed when question page is interrupted**



**Figure A11: zoom in of the warning message**

## Appendix B: Student Consent Form

Consent Form (for Adult Participants)  
Agreement to Participate in Research

Responsible Investigator: David Scot Taylor  
Title of Protocol: Study of Active Learning Framework Effectiveness for Teaching Data Structures and Algorithms

1. You have been asked to participate in a research study investigating the effectiveness of software currently under development for teaching the topics of Data Structures and Algorithms.]
2. For several topics taught during the remainder of the semester, you will be given online exercises as homework assignments. For each topic, you will be asked to:
  - a) Complete an exercise for the given topic.
  - b) Use either an animation or the framework software to further your understanding of topic.
  - c) Complete a second exercise, as in 2a, for the problem.
  - d) Use either an animation or the framework software (whichever was not selected in 2b), to further your understanding of the topic.
  - e) Complete a third exercise, as in 2a, for the problem.

Additionally, you may also be asked to fill out an anonymous survey about your thoughts of the software.

The exercises will be graded as homework questions, and statistics will be collected for study of the system's effectiveness. For the purpose of the study, no individual results will be made available, rather aggregate data will be collected. However, for the purpose of homework grades, your data will be used as a regular homework submission.

You may choose to participate or opt out of the study. This will only change whether or not your data is aggregated within the statistics of the study. You are expected to complete your homework exercises either way. If, at any point, you decide that you do not want your grades to be included within the aggregate statistics, please send notification to Andrei Lurie, at **[a\\_lurie@hotmail.com](mailto:a_lurie@hotmail.com)**. He will remove your information from the aggregate statistics, without informing the professor of who has removed themselves.

3. There are no risks anticipated for this study.
4. Information gathered from this study may directly influence the way this course is taught in future semesters. Your willingness to participate should improve future course offerings.
5. If you do not wish to take part in the study, please send notification to Andrei Lurie, at [a\\_lurie@hotmail.com](mailto:a_lurie@hotmail.com), who will remove you from the study statistics without informing the professor of your identity.
6. Although the results of this study may be published, no information that could identify you will be included.
7. No compensation will be given for participation.
8. Questions about this research may be addressed to David Taylor, (408) 924-5124. Complaints about the research may be presented to Department Chair Kenneth Loudon, Department of Computer Science, College of Science, (408) 924-5060. Questions about a research subjects rights, or research-related injury may be presented to Pamela Stacks, Ph.D., Associate Vice President, Graduate Studies and Research, at(408) 924-2480.
9. No service of any kind, to which you are otherwise entitled, will be lost or jeopardized if you choose to not participate in the study.
10. Your consent is being given voluntarily. You may refuse to participate in the entire study or in any part of the study. If you do not participate, there will be no effect on your course grade (the instructor will not even be aware of your lack of participation). Two exceptions: first, if you do not do a given exercise, there will be no data for that exercise to include in the study, and your instructor will thus know that you were not included in the study for that exercise. In this case, you still will not be penalized for not being in the study, though your homework grade will reflect that you have not done that homework assignment. Second, if all students, or no students, participate in the study, the professor will obviously be able to deduct that fact given that the total number of students in the class is known. If you decide to participate in the study, you are free to withdraw at any time without any negative effect on your relations with San Jose State University or with any other participating institutions or agencies. (Upon withdrawal, your previously submitted answers and surveys may still be included in research, as there is no way to "unpublish" results.)

11. **If you are under the age of 18, you should not participate in the study. Please email Andrei Lurie to notify him, or withhold consent upon system login.**
- **The acceptance of a subject to the terms of this document indicates that the subject has been fully informed of the rules of study participation.** By accepting this form, you are giving consent that your information be used within the study. You may make a copy of this consent form if you wish, as it will be available online ([here](#)).
  - **The principle investigator agrees to include consenting subjects in the research and attests that each subject has been fully informed of his or her rights.**

## Appendix C: Experiment Data

Data for each of the exercises, as well as for all exercises combined, is presented in the figures below. Data is divided into two groups: interaction group and animation group. Further, for each group, various data sets are created such as normalized and un-normalized, complete and clean. Refer to Section 4 for more details about the groups and the data sets.

Sections C1 through C6 cover exercises 1 through 6, accordingly. Section C7 covers all exercises combined. Section C8 covers exercises 1 and 5 combined (these are the exercises that used interactive tool1). And Section C9 covers exercises 2, 3, 4, 6 combined (these are the exercises that used interactive tool2).

For each section, we present the following figures:

- Histogram, showing the possible values for the exercise score (one row per value) and the count of students who attained that particular score. The histogram is accompanied by a bar graph, showing, per question, the number of students who attained a score which is below 50% and the number of students who attained a score which is above 50%. The idea behind such statistic is that it is a one way to show the improvement (or degradation) of the group. That is, we look at how many students attained a score worse/better than 50% before and after using the tool. Note that we don't include the students who got a score which is exactly 50% since we are measuring on the scale of worse/better. We should also clarify that by 50% we imply the 50% of the possible score for the exercise. That is, it is not the group score mean. Basically, what the 50% translates to is, for the exercises with a maximum score of 3, we count the number of students attaining a score of less than 2 and the number of students attaining a score of greater than 1, and for the exercises with a maximum score of 6, we count the number of students attaining a score of less than 3 and the number of students attaining a score of greater than 3. We show two figures for the histogram, one for un-normalized complete data set and the other for un-normalized clean data set. We only show figures for the clean data set in Figures C35 and C37.
- Descriptive statistics, showing the mean, confidence interval for mean, and standard deviation for the group. The table is accompanied by a bar graph, showing the confidence interval for mean. We show four figures of this statistics covering all data set combinations: un-normalized complete, un-normalized clean, normalized complete, and normalized clean. Exception are the sections C7 through C9 where only the normalized complete and normalized clean data sets can apply. We only show figures for the clean data set in Figures C36 and C38.

## C1: Exercise 1 (Breadth-First Search)

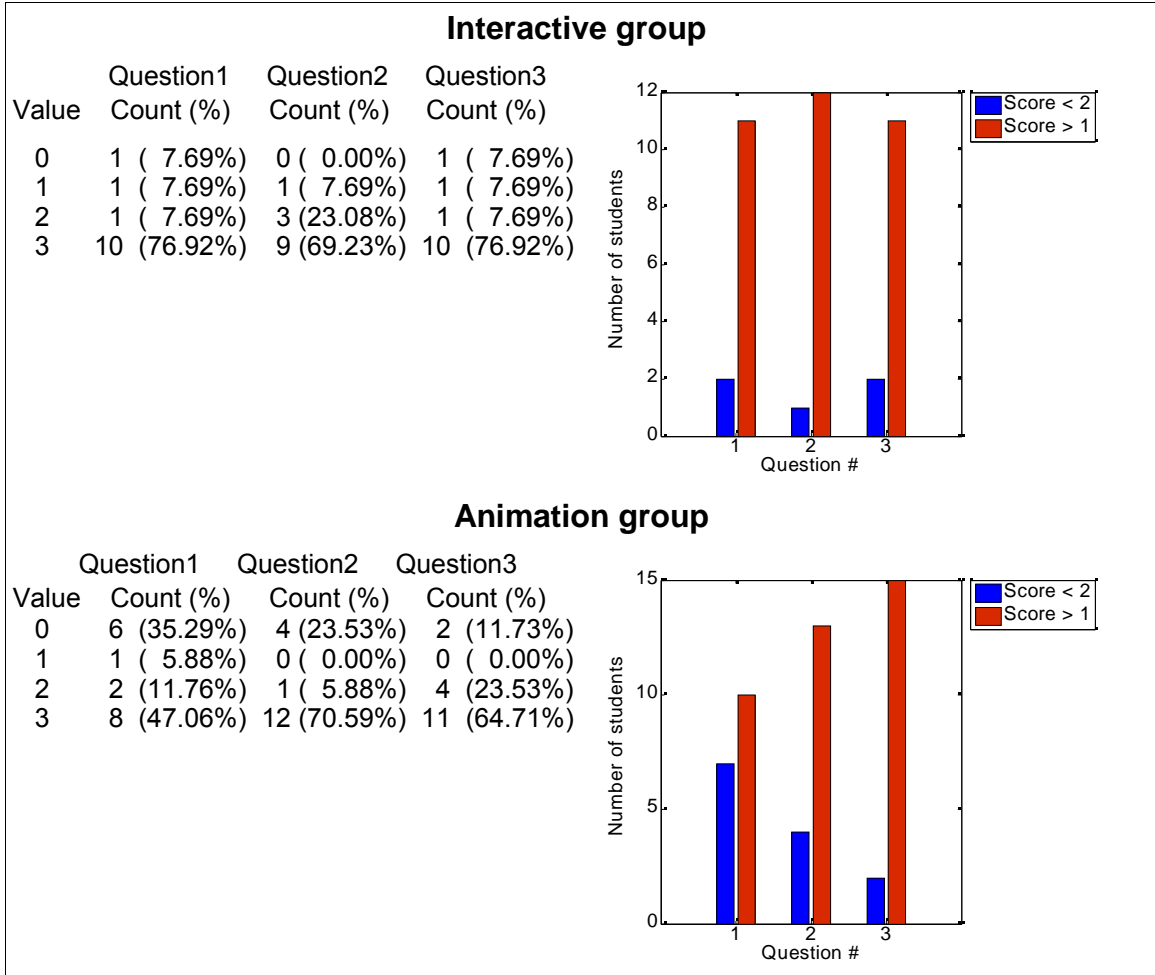
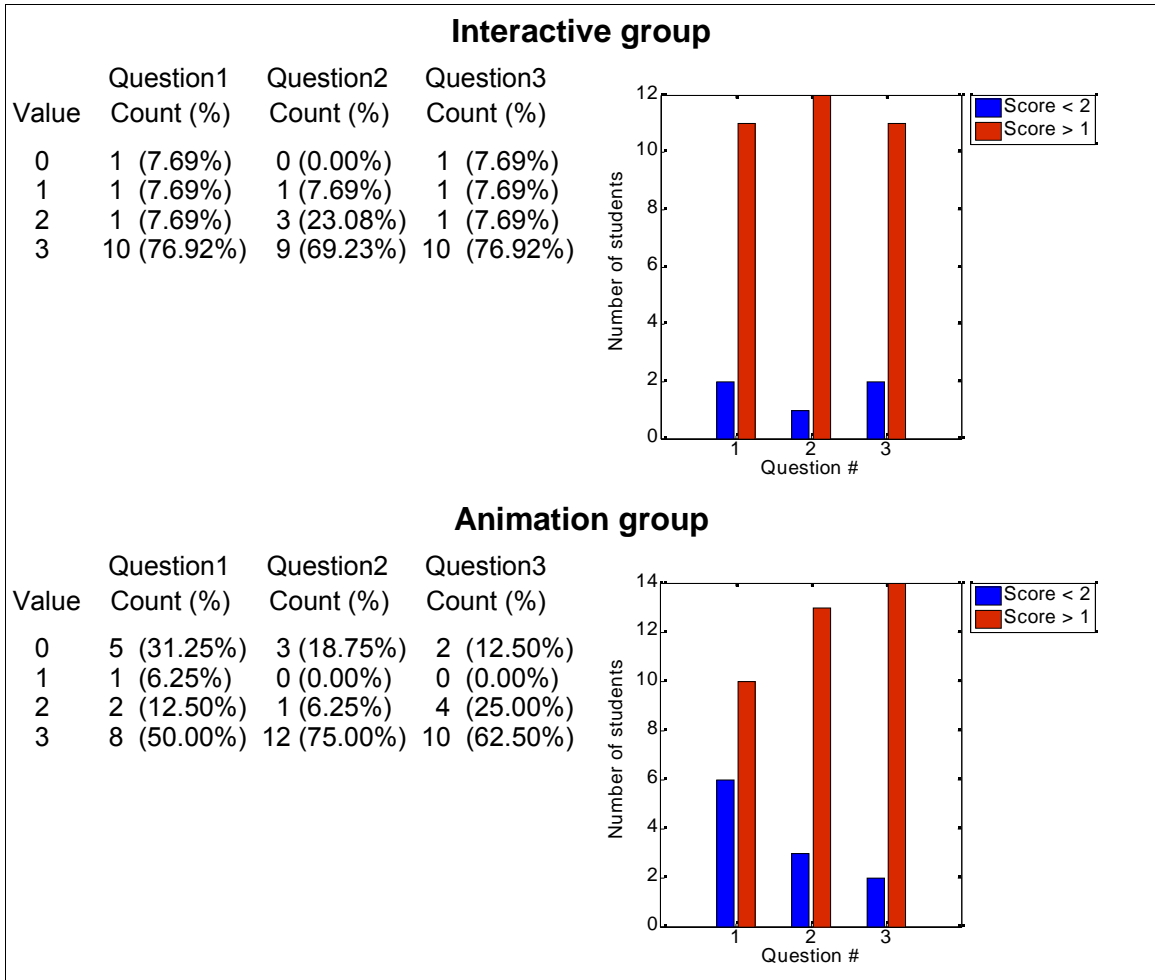
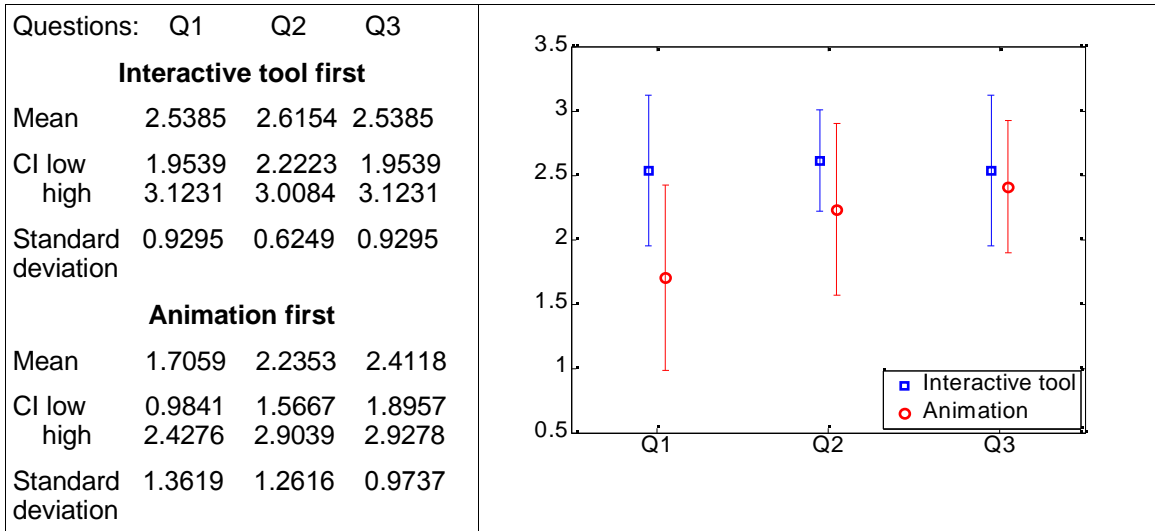


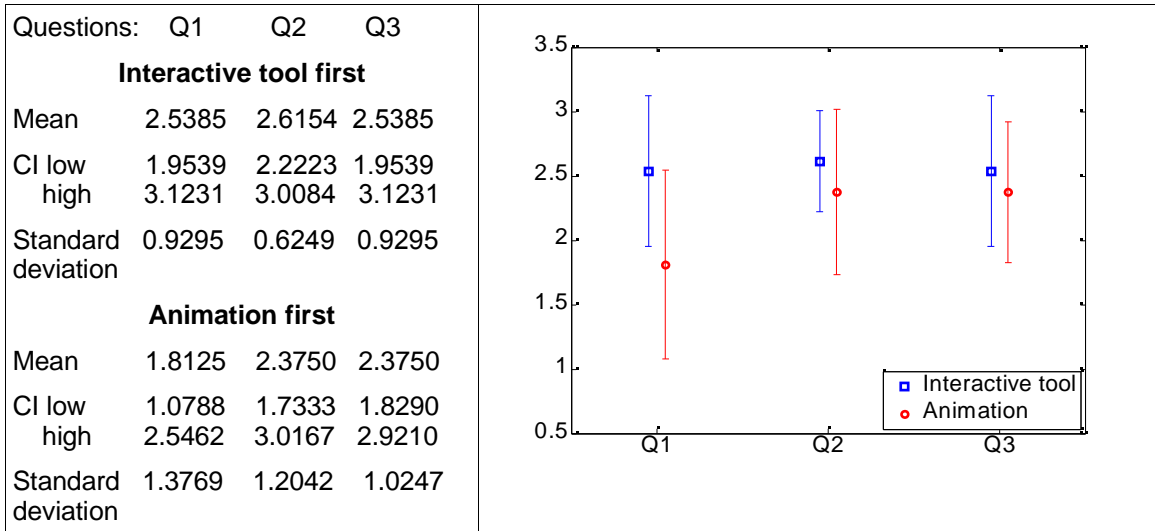
Figure C1: Histograms of complete data set by group (BFS)



**Figure C2: Histograms of clean data set by group (BFS)**

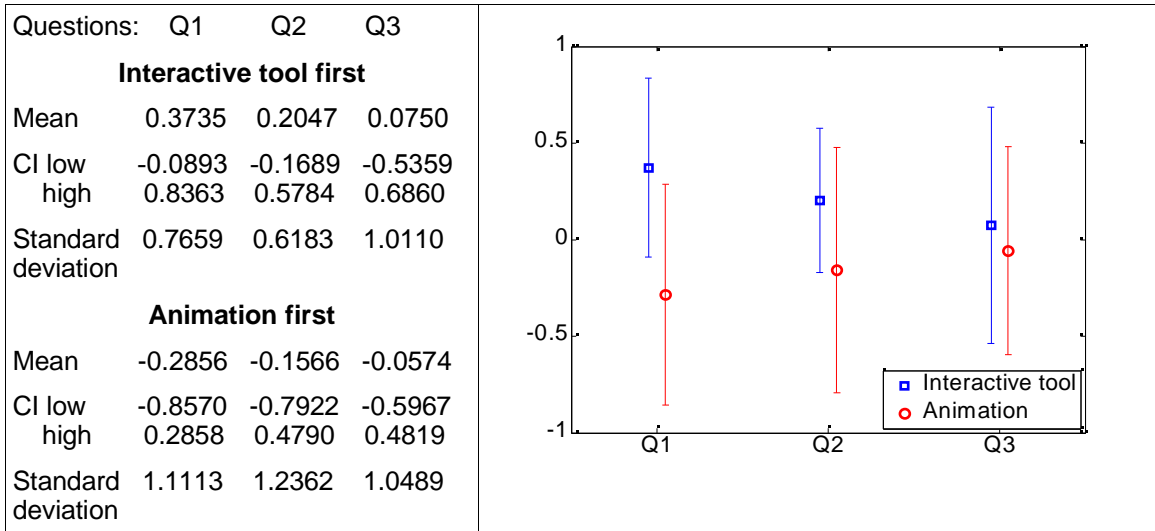


**Figure C3: Statistics for un-normalized, complete data set (BFS)**

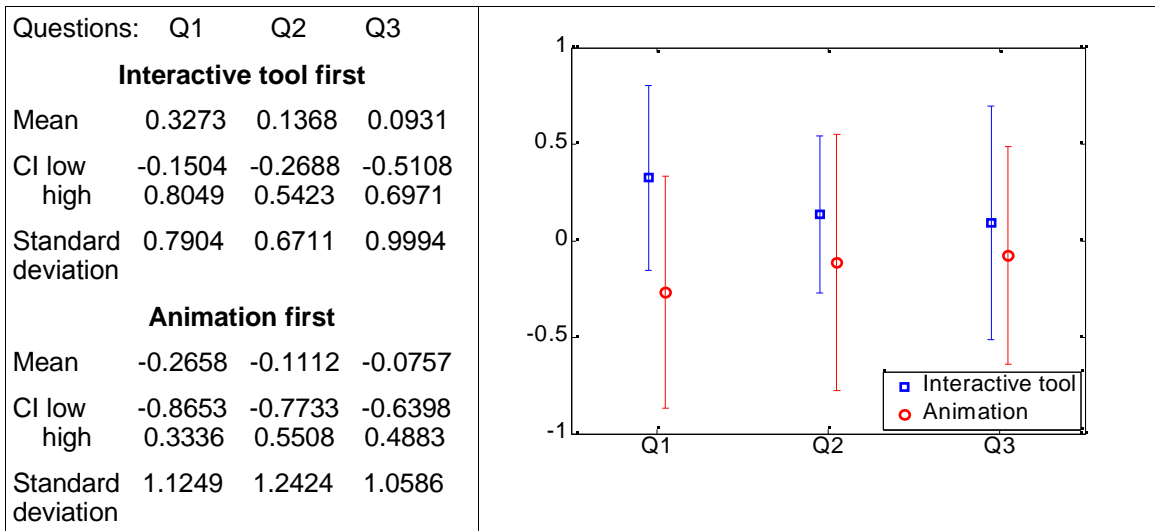


**Figure C4: Basic statistics for un-normalized, clean data set**





**Figure C5: Statistics for normalized, complete data set (BFS)**



**Figure C6: Statistics for normalized, clean data set (BFS)**

## C2: Exercise 2 (Depth-First Search)

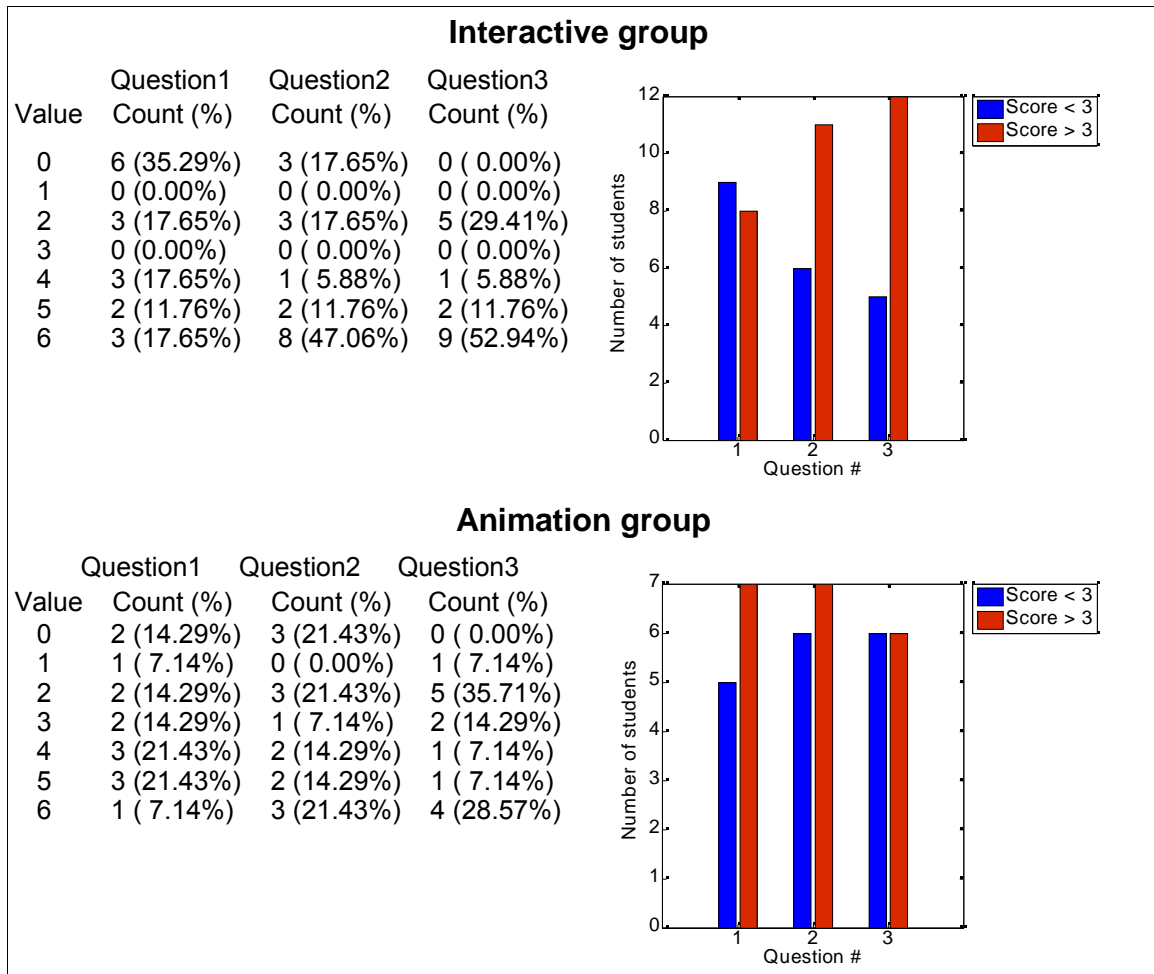
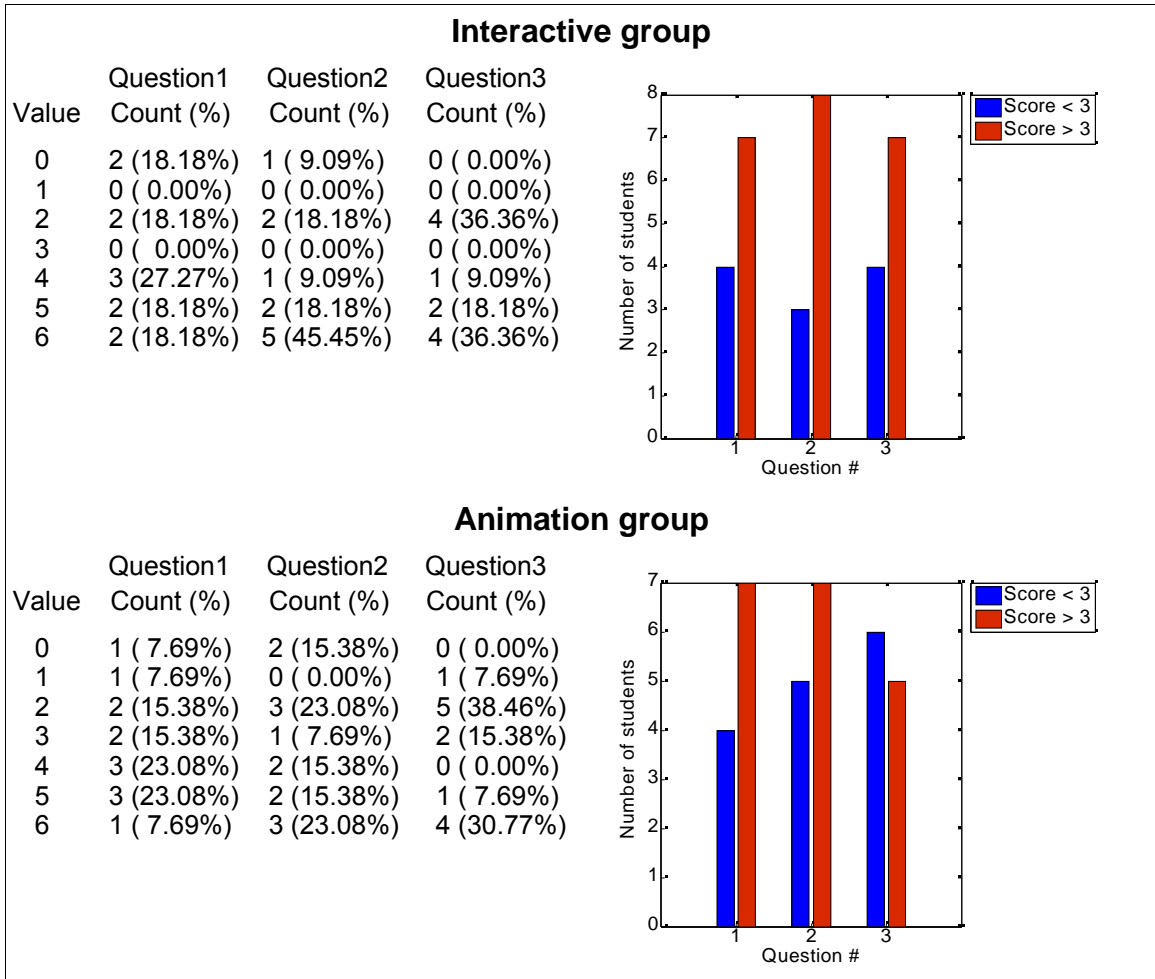
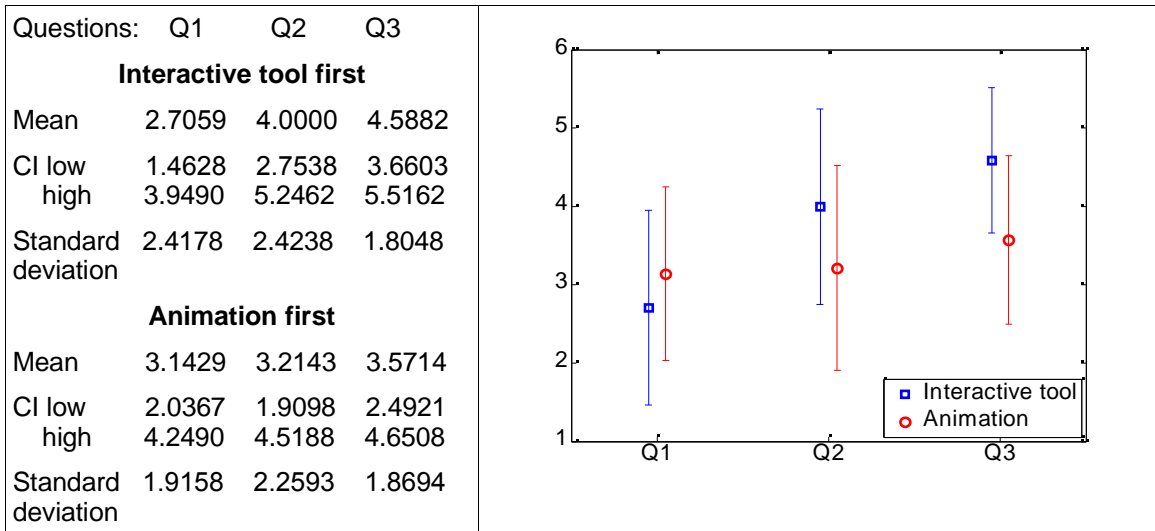


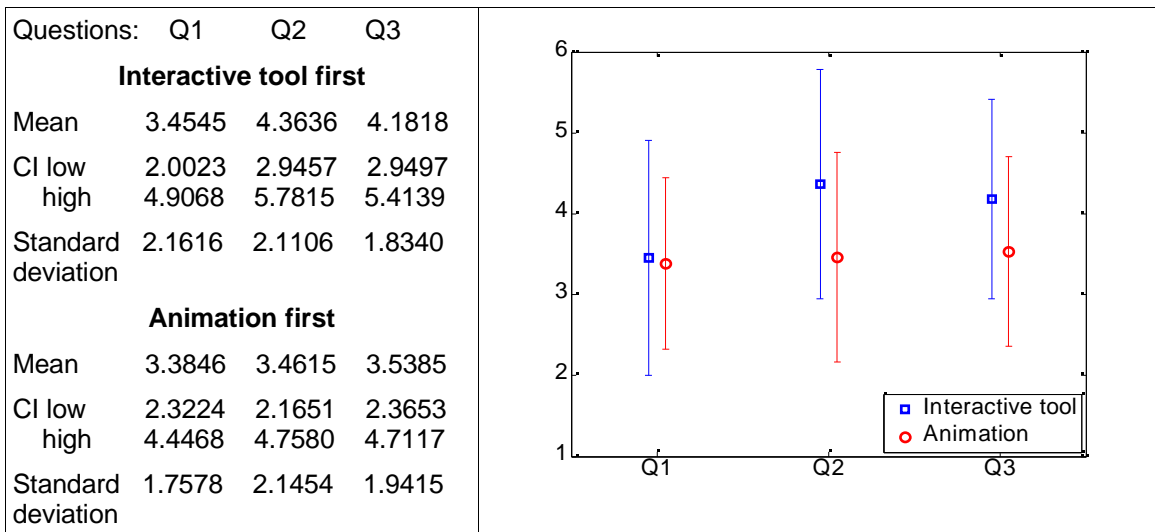
Figure C7: Histograms for complete data set by group (DFS)



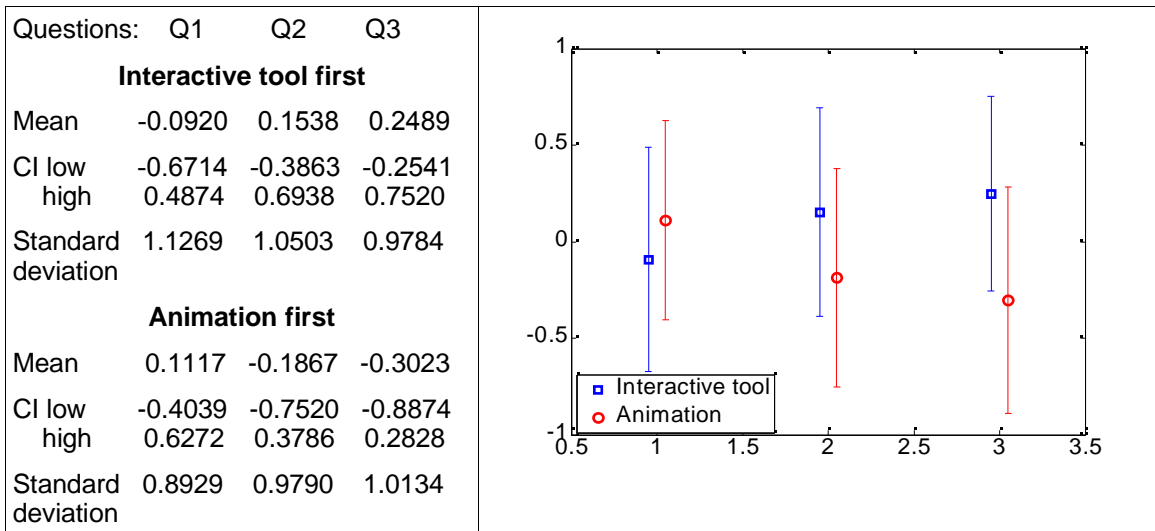
**Figure C8: Histograms for clean data set by group (DFS)**



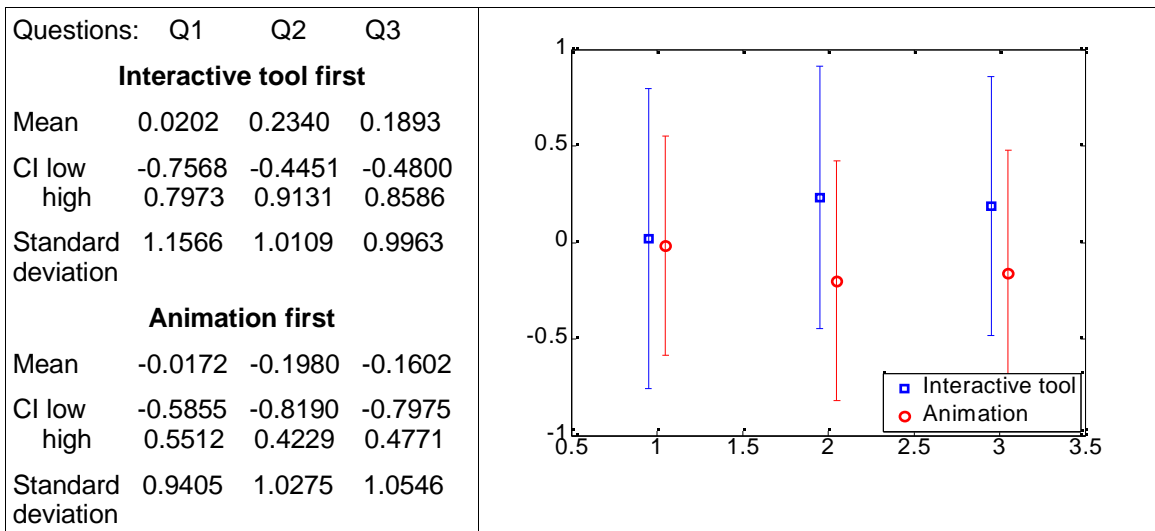
**Figure C9: Statistics for un-normalized, complete data set (DFS)**



**Figure C10: Statistics for un-normalized, clean data set (DFS)**

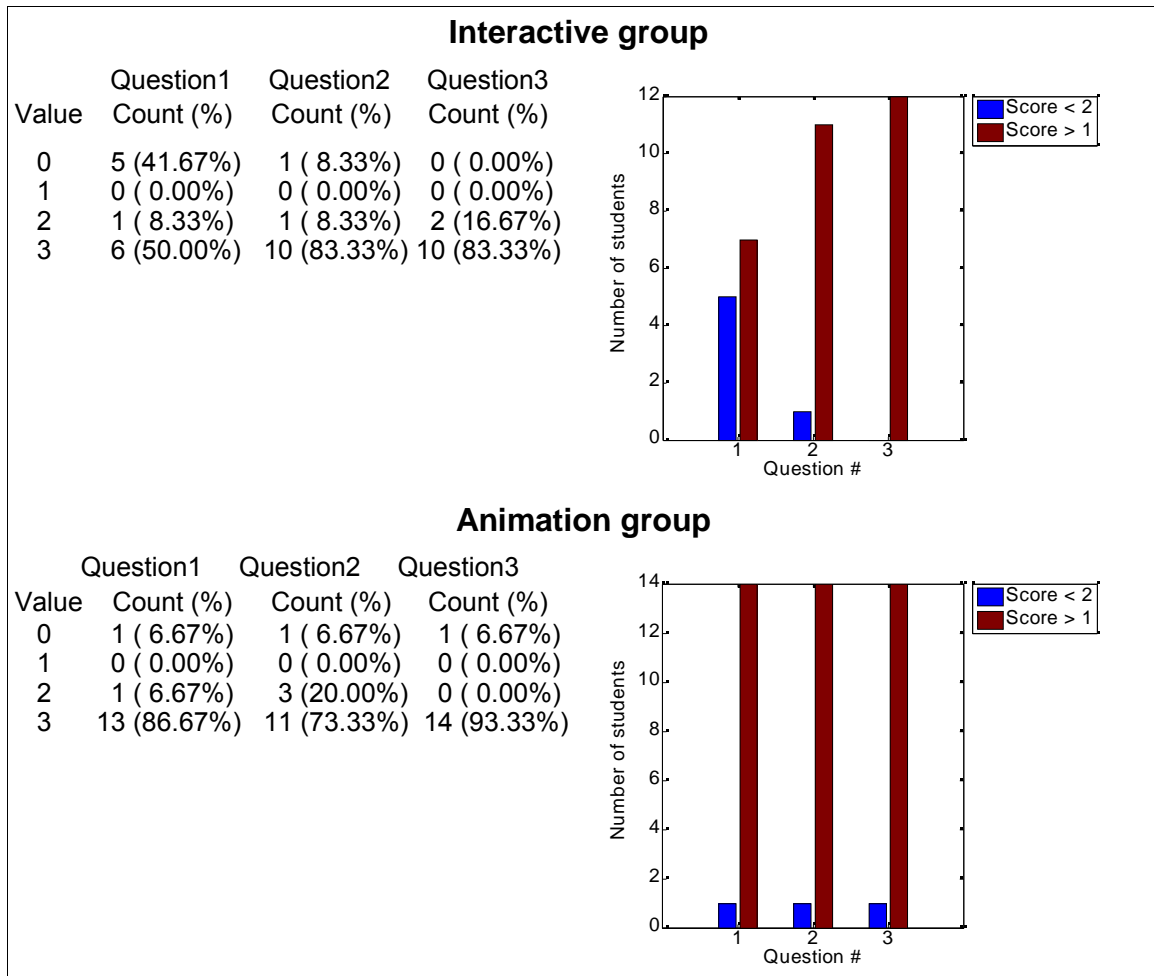


**Figure C11: Statistics for normalized, complete data set (DFS)**

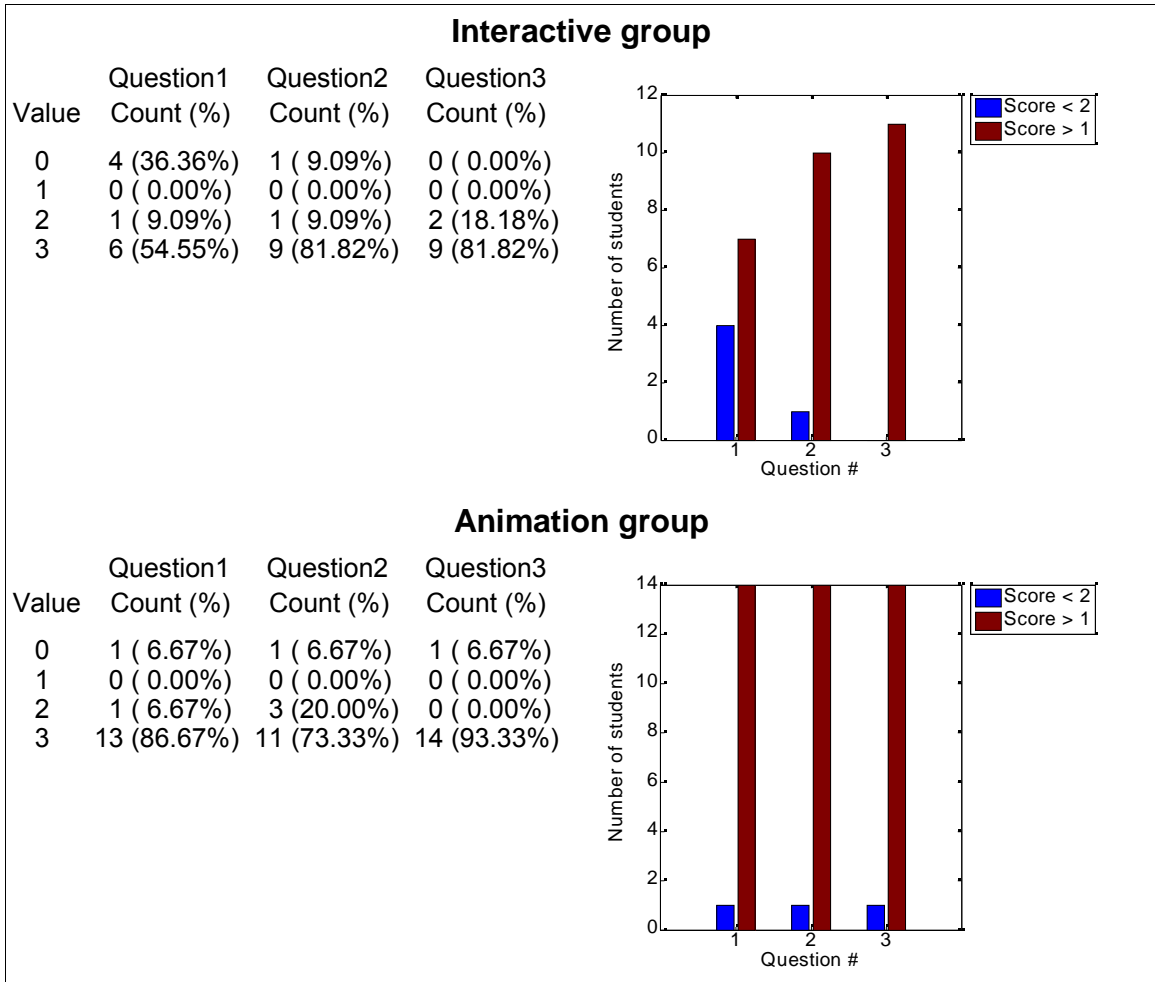


**Figure C12: Statistics for normalized, clean data set (DFS)**

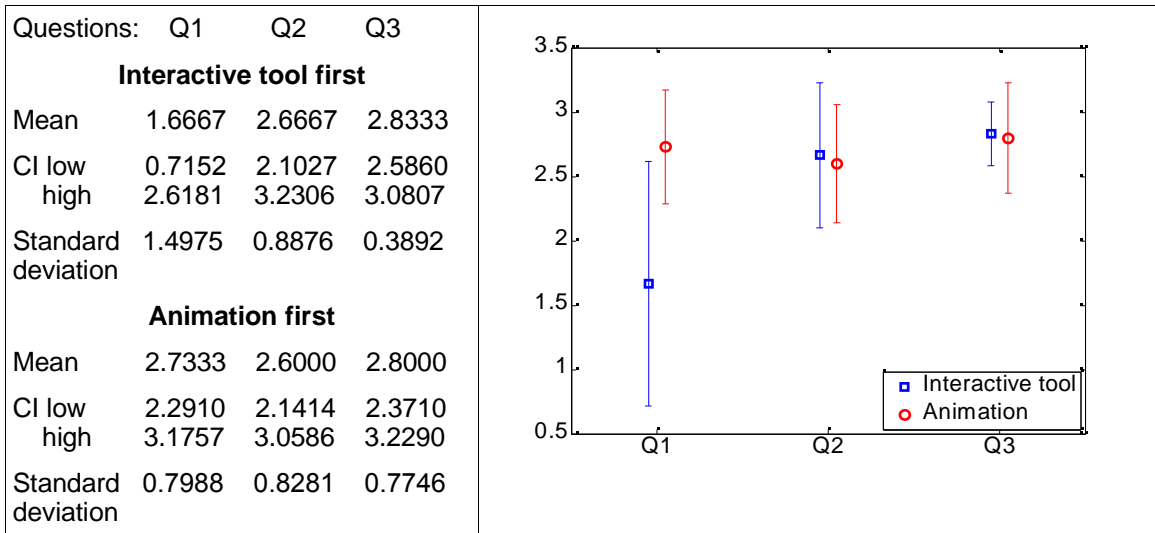
### C3: Exercise 3 (Kruskal's algorithm)



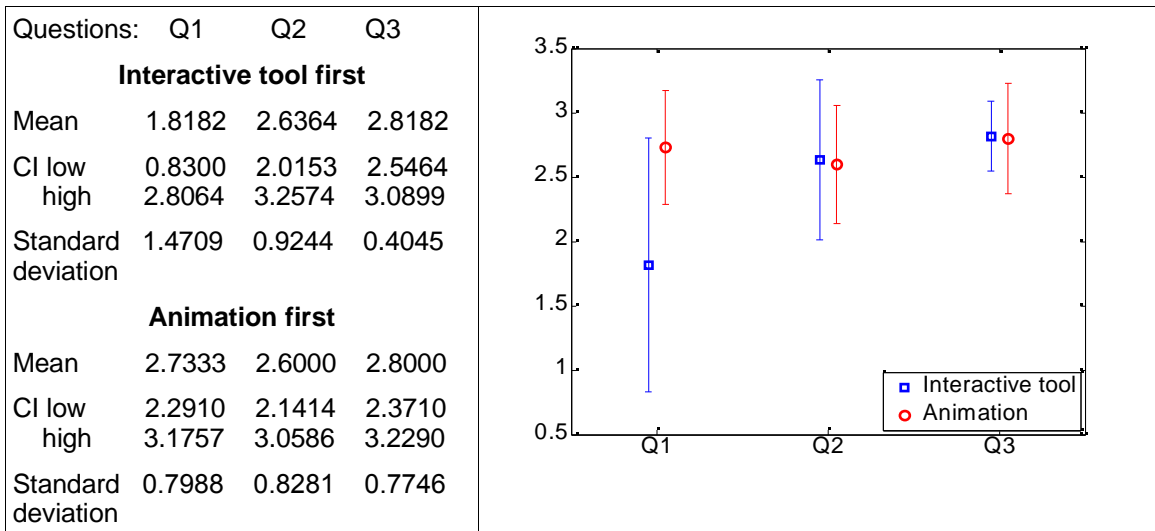
**Figure C13: Histograms for complete data set by group (Kruskal's)**



**Figure C14: Histograms for clean data set by group (Kruskal's)**

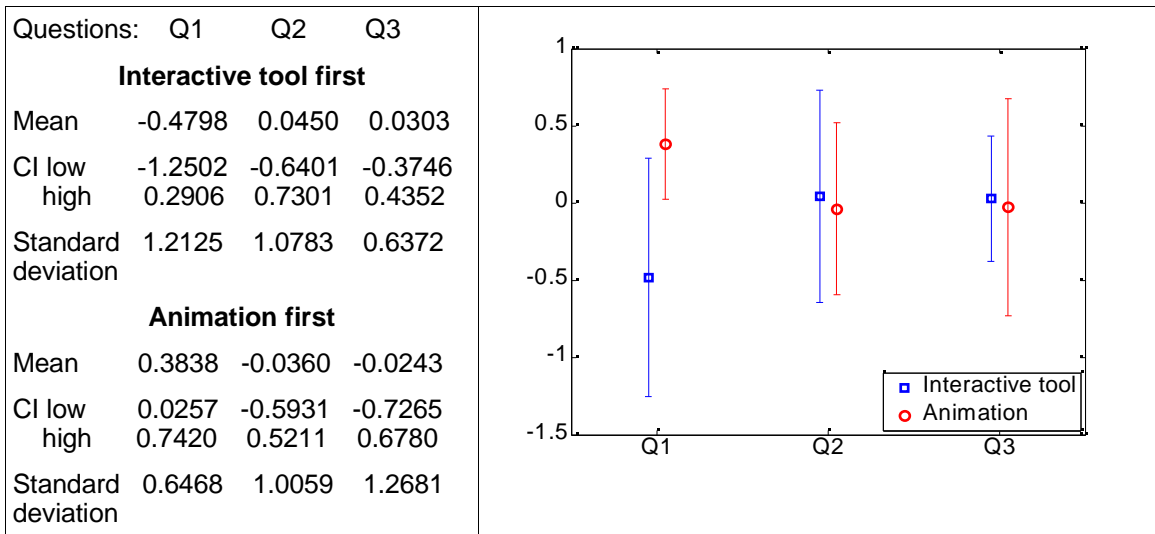


**Figure C15: Statistics for un-normalized, complete data set (Kruskal's)**

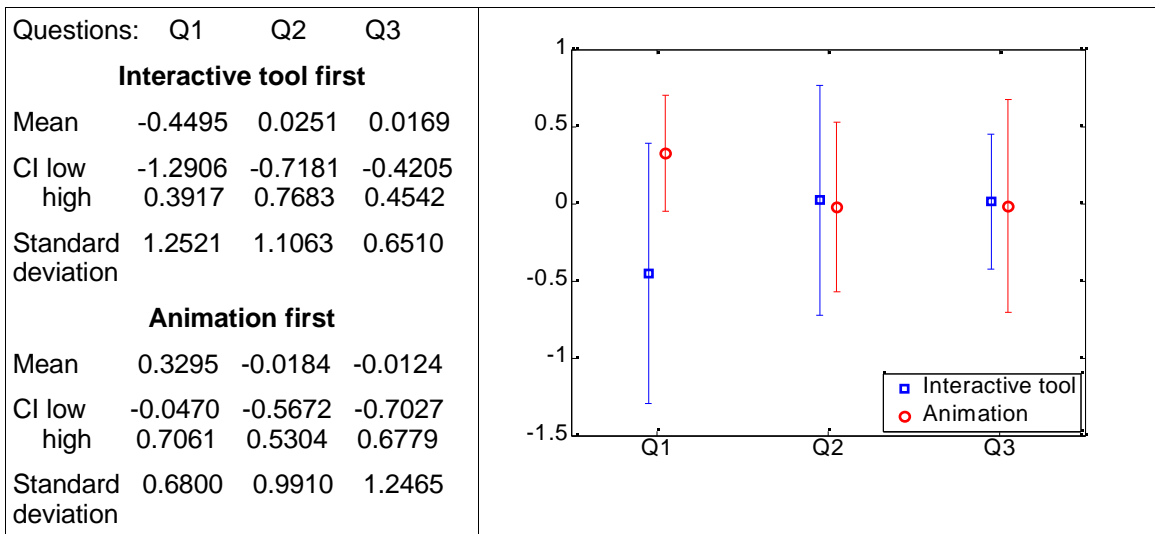


**Figure C16: Statistics for un-normalized, clean data set (Kruskal's)**





**Figure C17: Statistics for normalized, complete data set (Kruskal's)**



**Figure C18: Statistics for normalized, clean data set (Kruskal's)**

### C4: Exercise 4 (Prim's algorithm)

For exercise 4, the complete data set and the clean data set are the same.

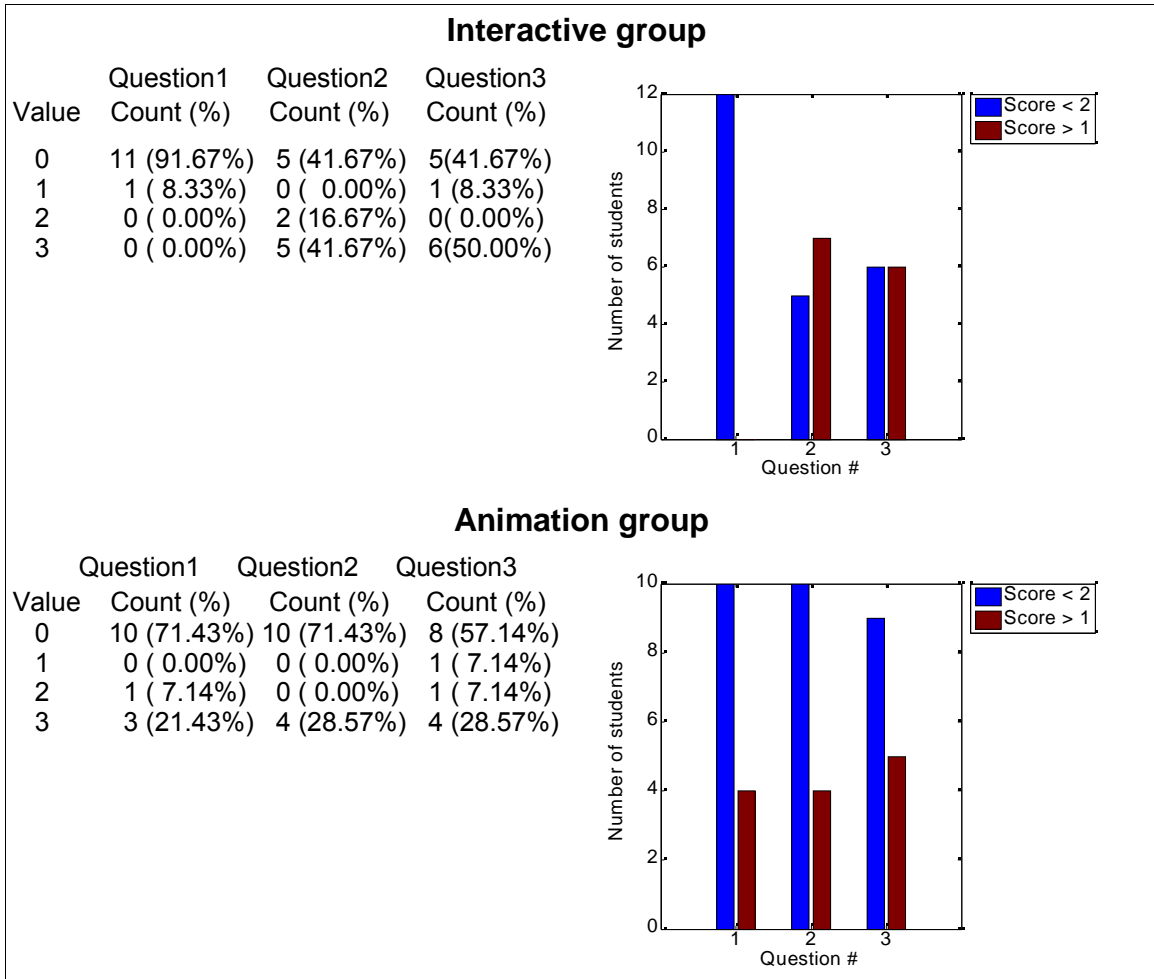
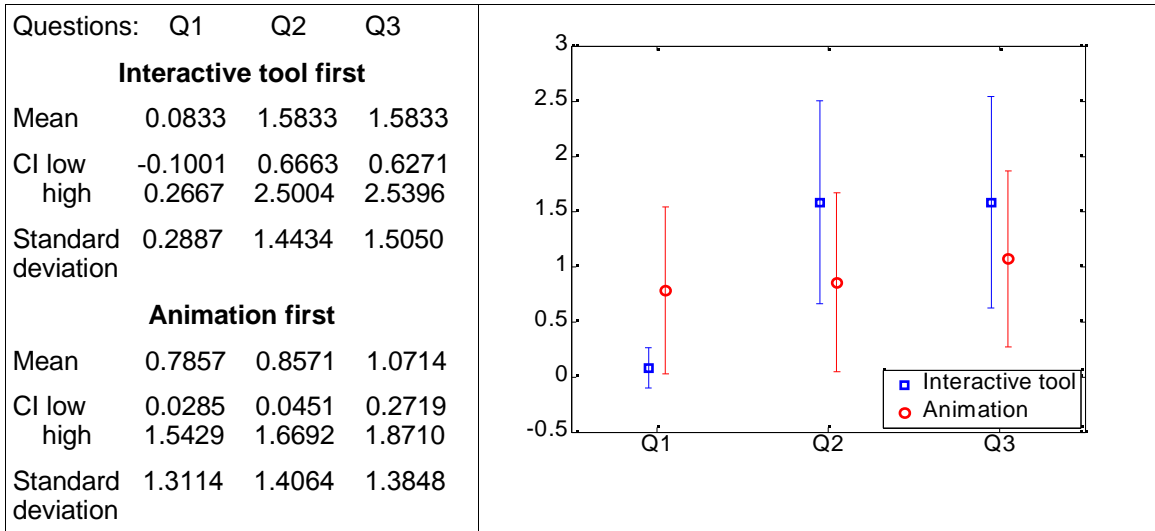
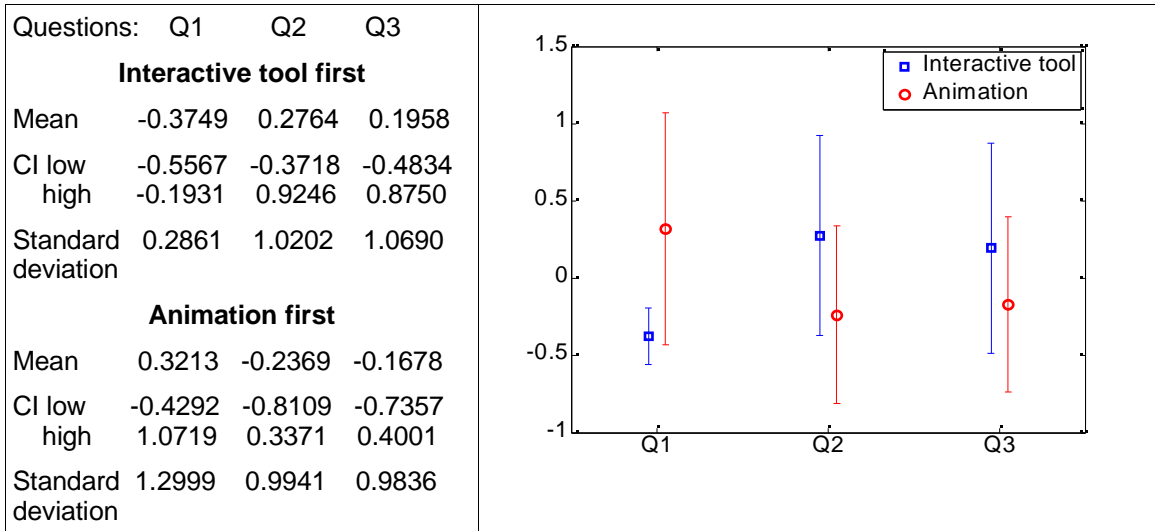


Figure C19: Histograms for complete/clean data set by group (Prim's)

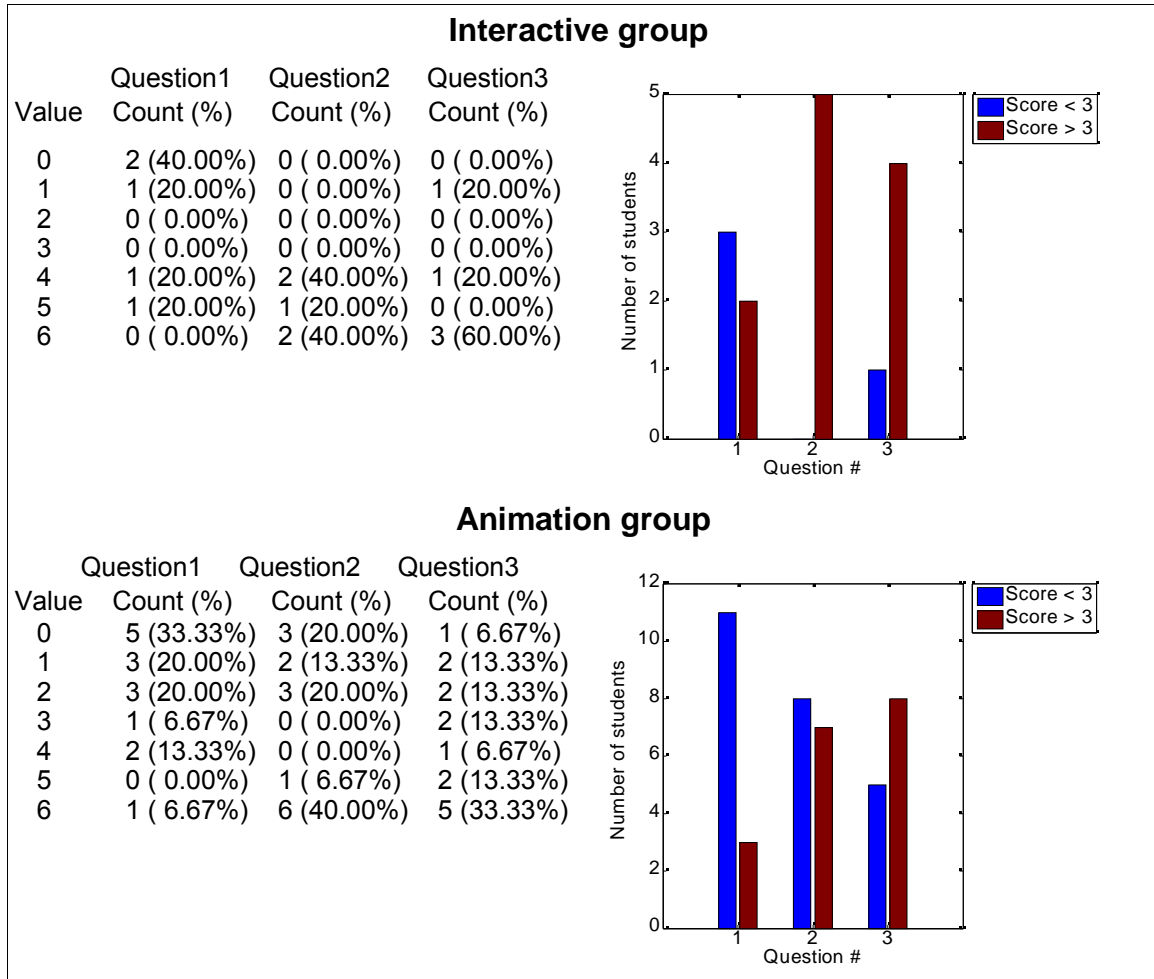


**Figure C20: Statistics for un-normalized data set (Prim's)**

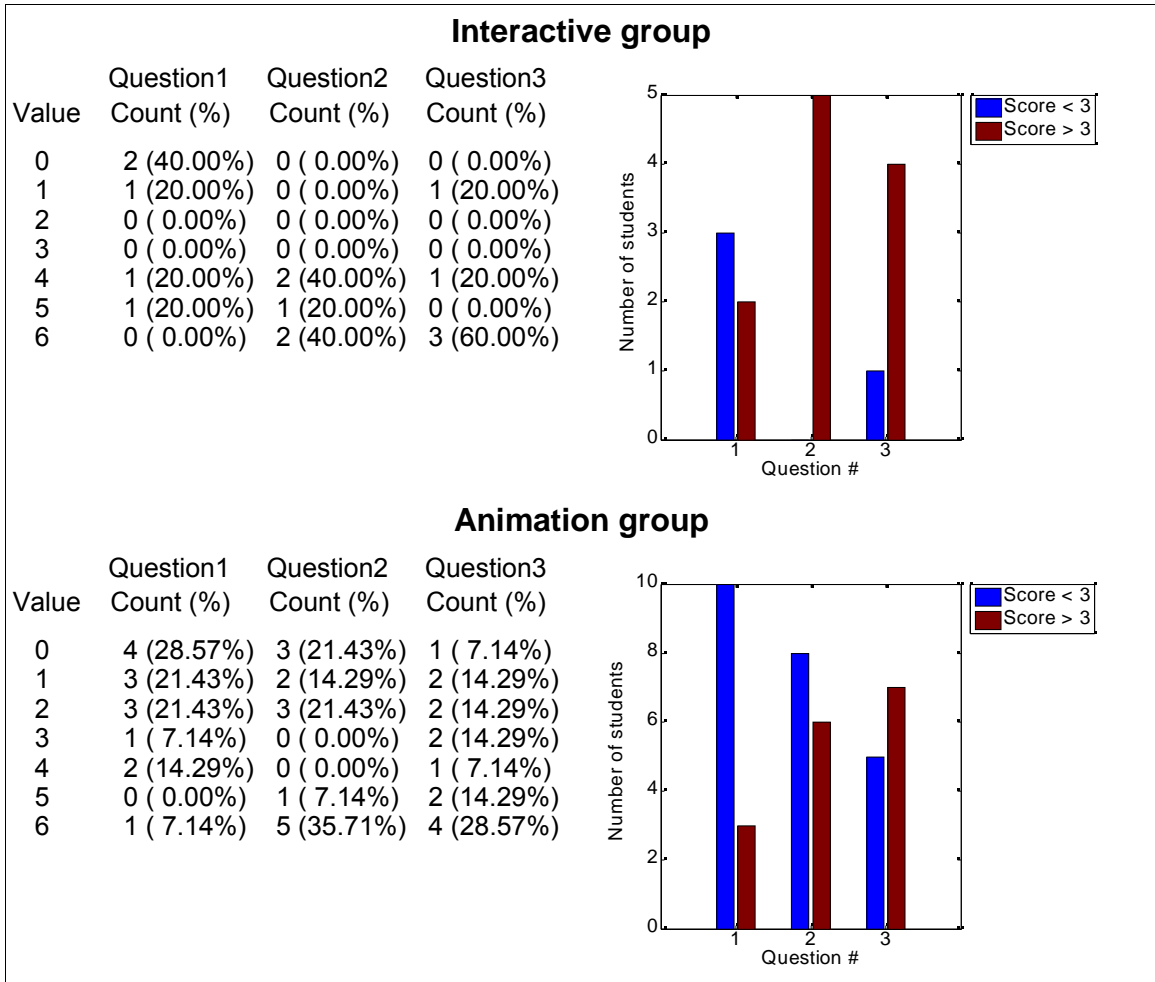


**Figure C21: Statistics for normalized data set (Prim's)**

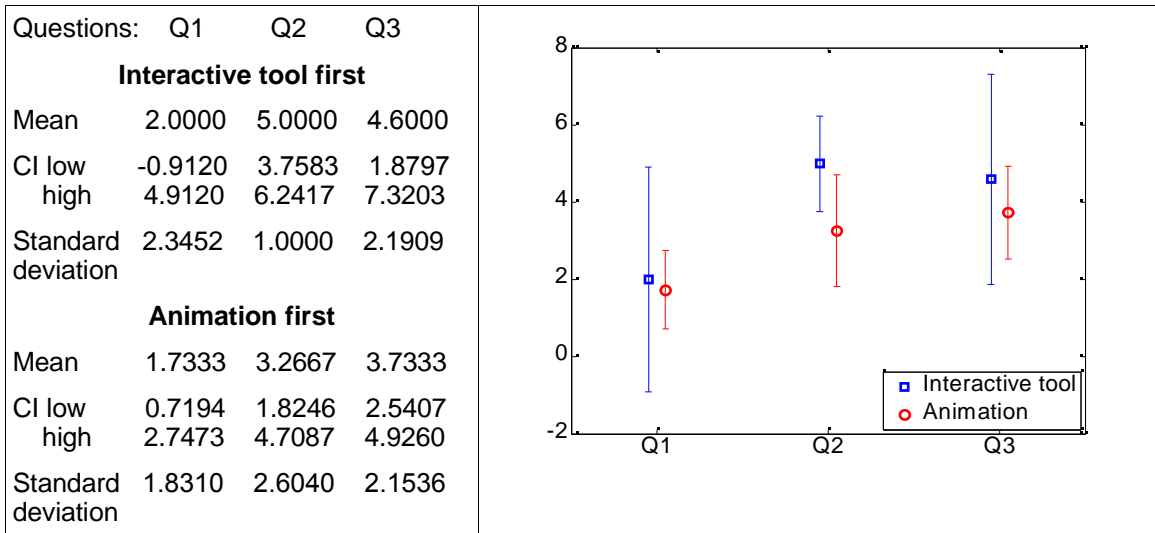
## C5: Exercise 5 (Bellman-Ford algorithm)



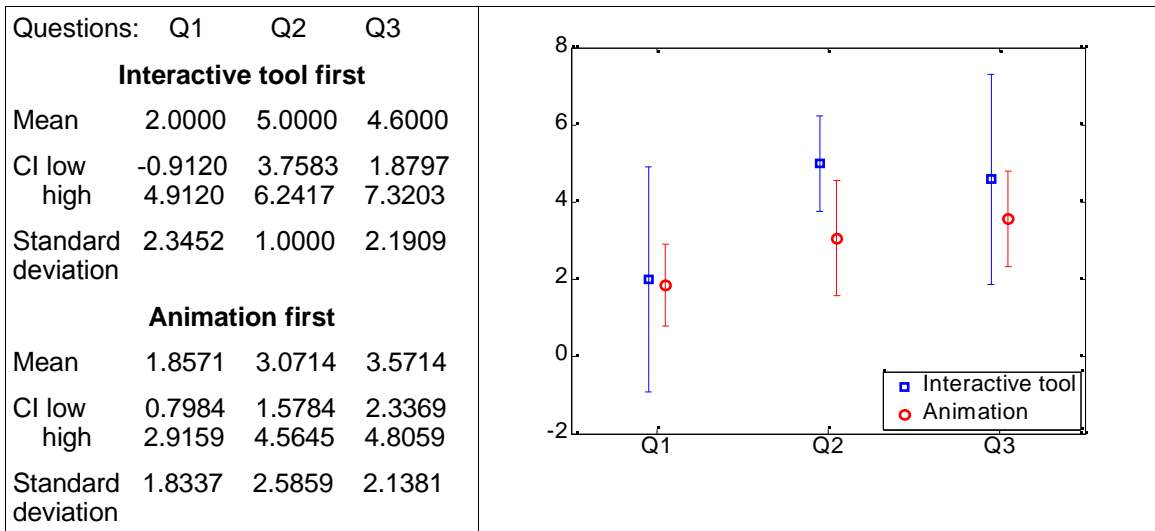
**Figure C22: Histograms for complete data set by group (Bellman-Ford)**



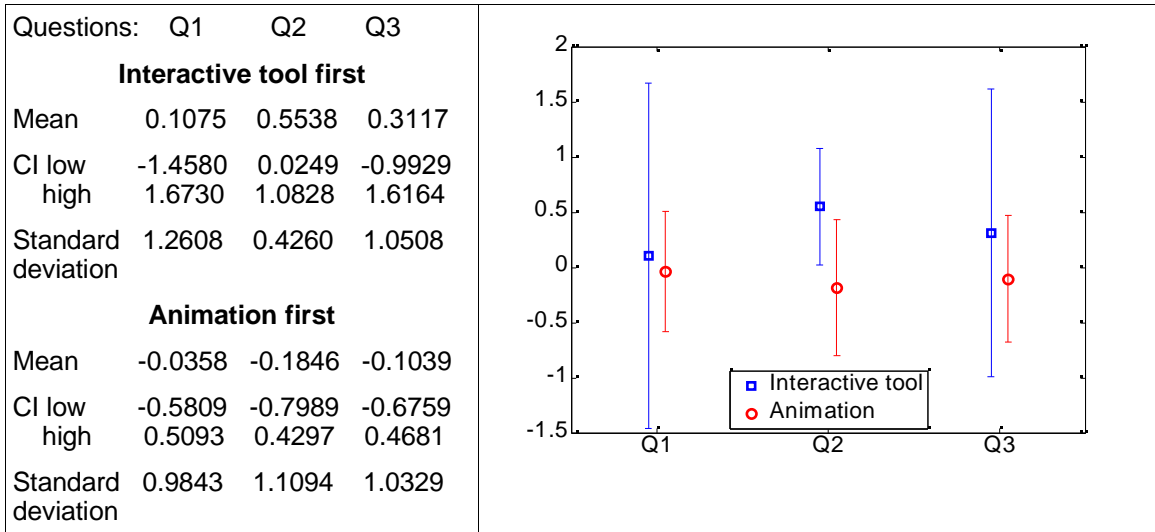
**Figure C23: Histograms for clean data set by group (Bellman-Ford)**



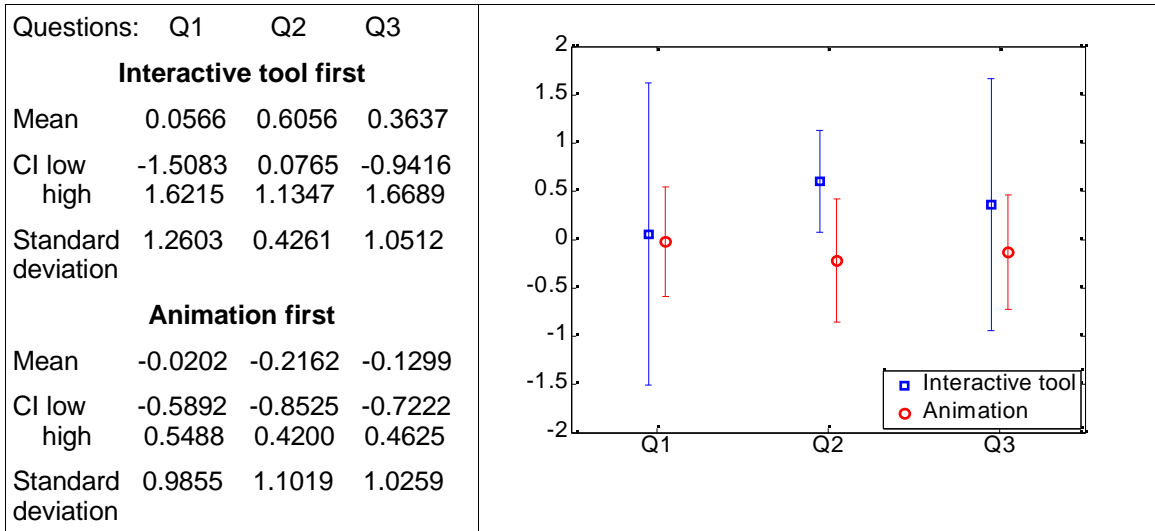
**Figure C24: Statistics for un-normalized, complete data set (Bellman-Ford)**



**Figure C25: Statistics for un-normalized, clean data set (Bellman-Ford)**



**Figure C26: Statistics for normalized, complete data set (Bellman-Ford)**



**Figure C27: Statistics for normalized, clean data set (Bellman-Ford)**

## C6: Exercise 6 (Dijkstra's algorithm)

For exercise 6, the complete data set and the clean data set are the same.

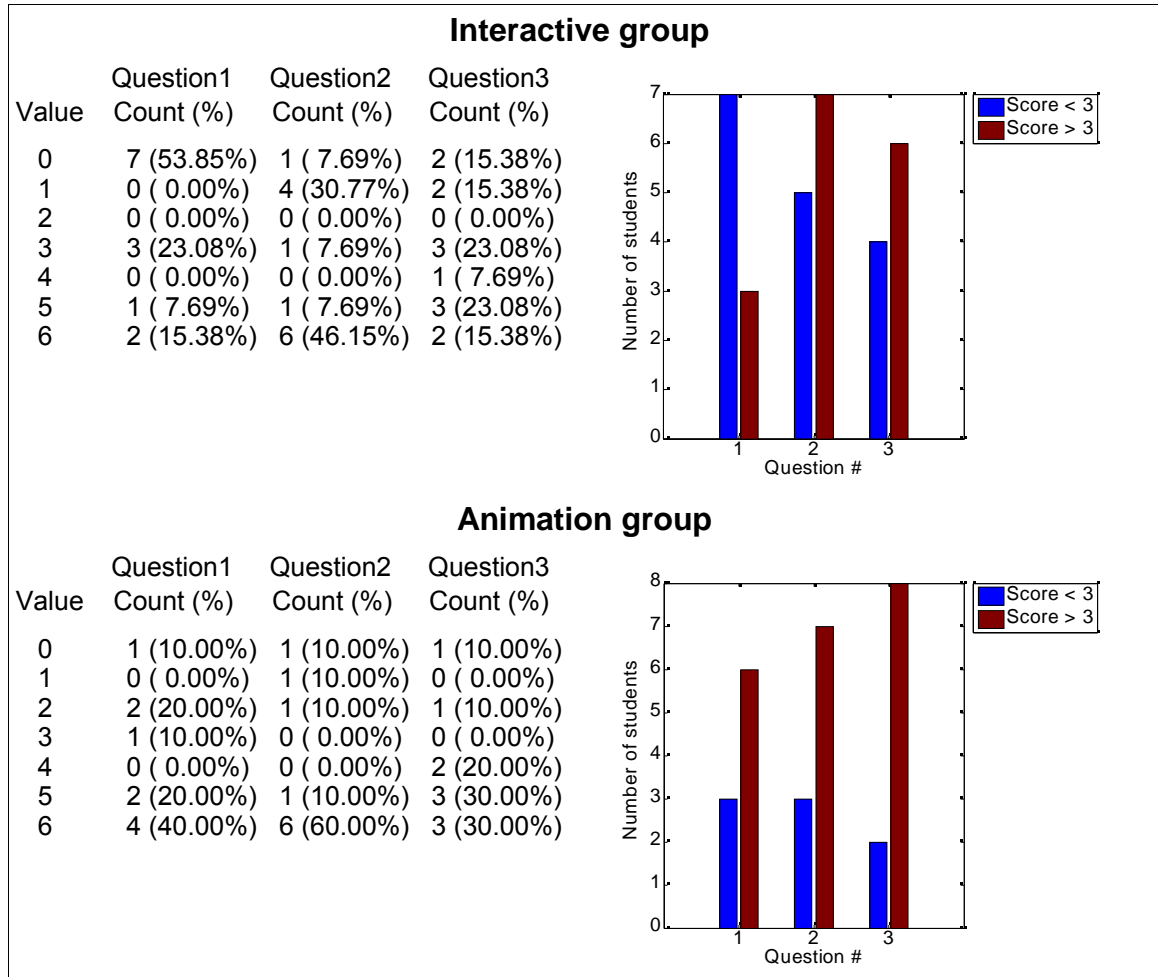
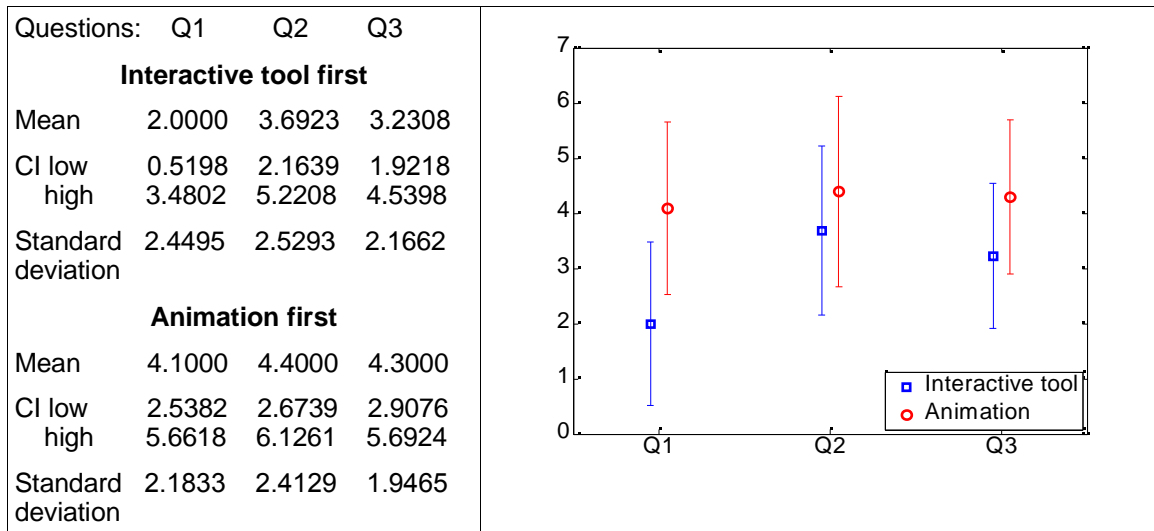
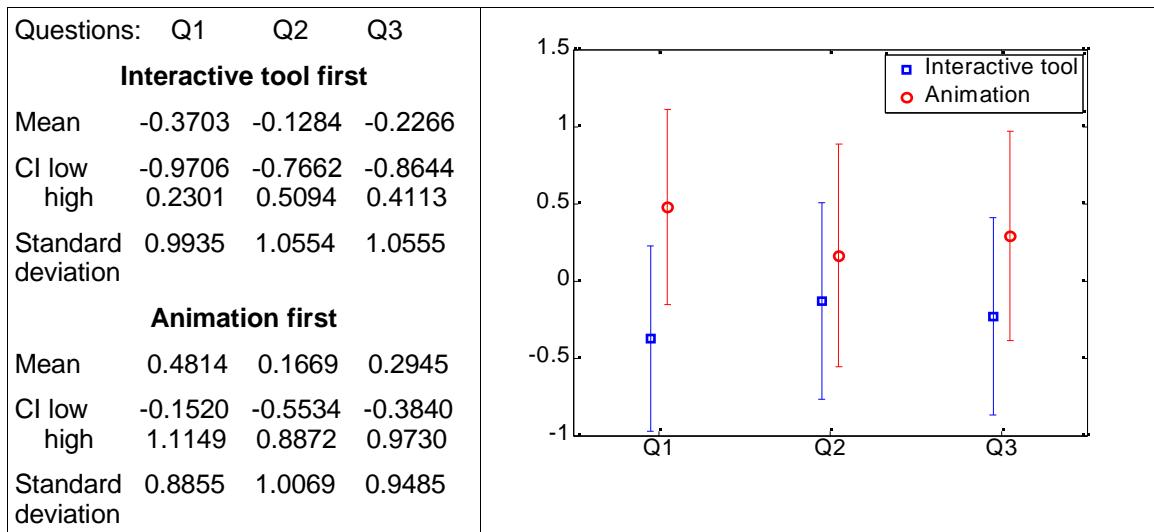


Figure C28: Histograms for complete/clean data set by group (Dijkstra's)





**Figure C29: Statistics for un-normalized data set (Dijkstra's)**



**Figure C30: Statistics for normalized data set (Dijkstra's)**

### C7: All six exercises Combined

Following is the histogram for number of scores below 50% and number of scores above 50% for the normalized data sets of all six exercises combined.

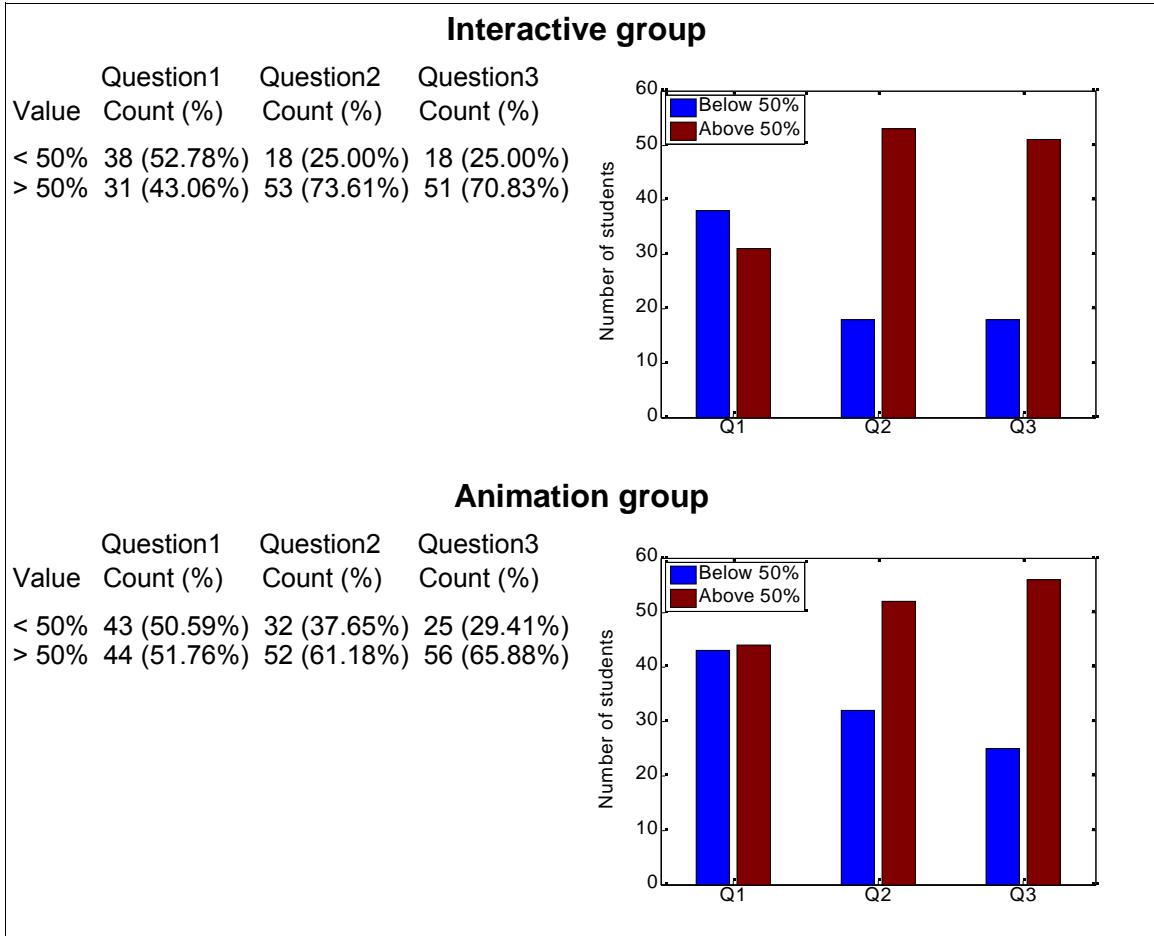
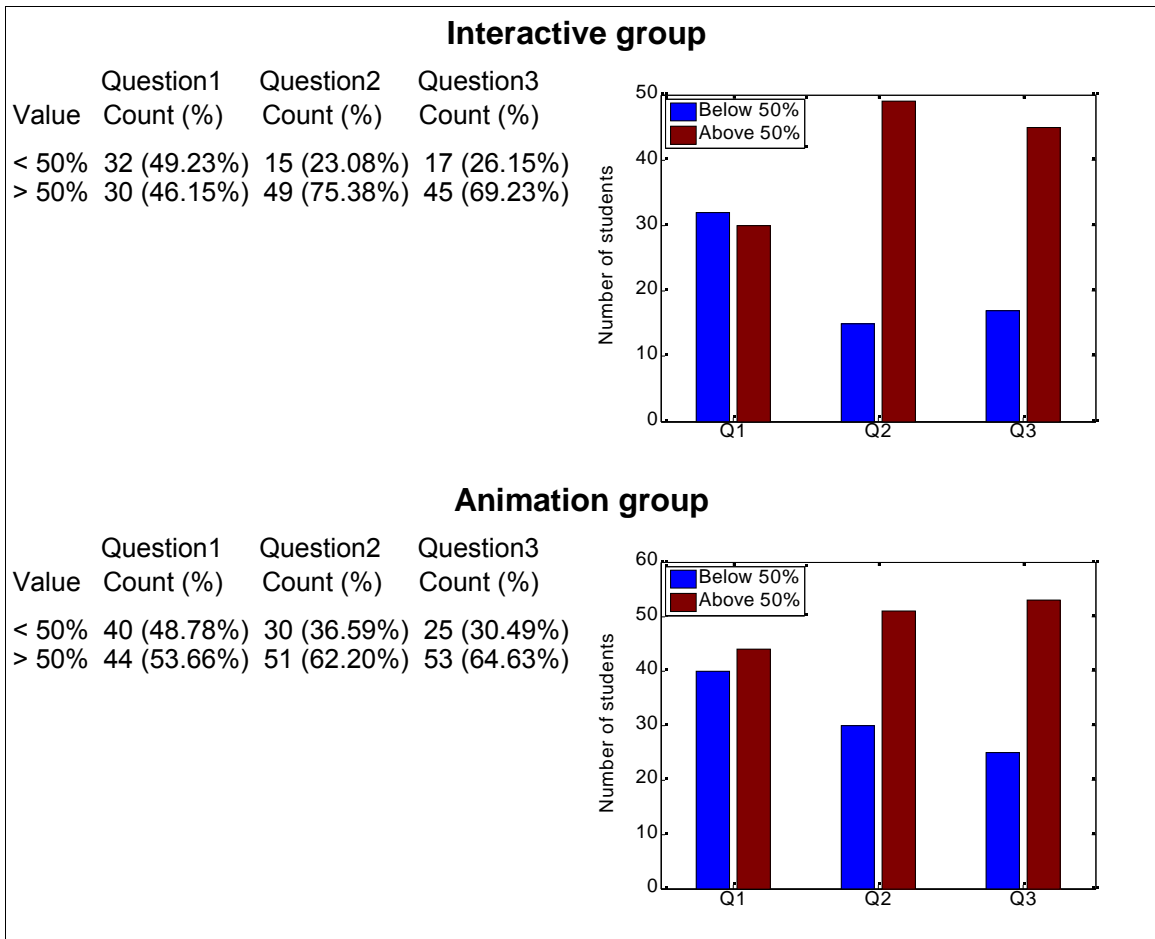
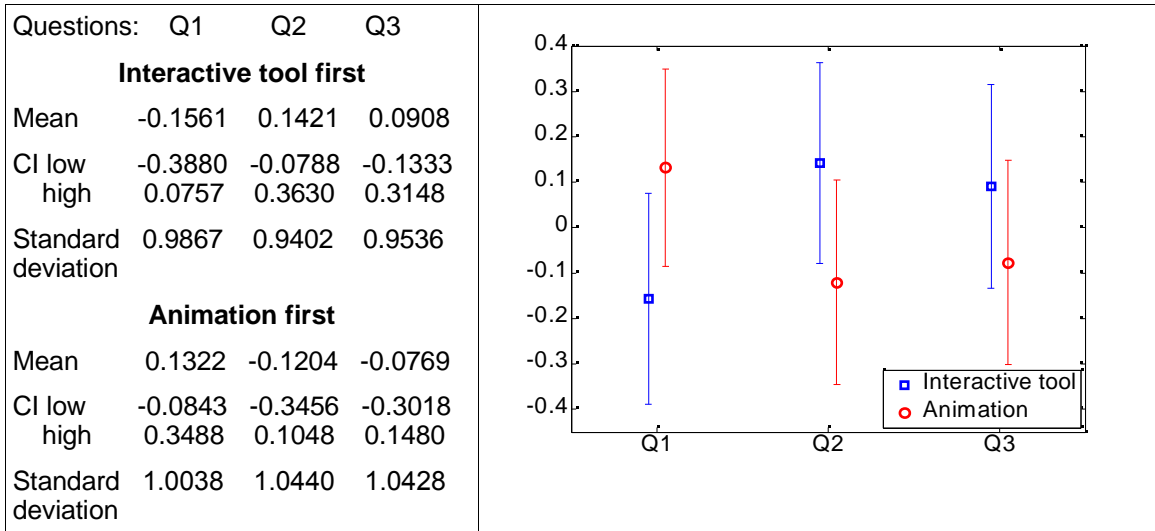


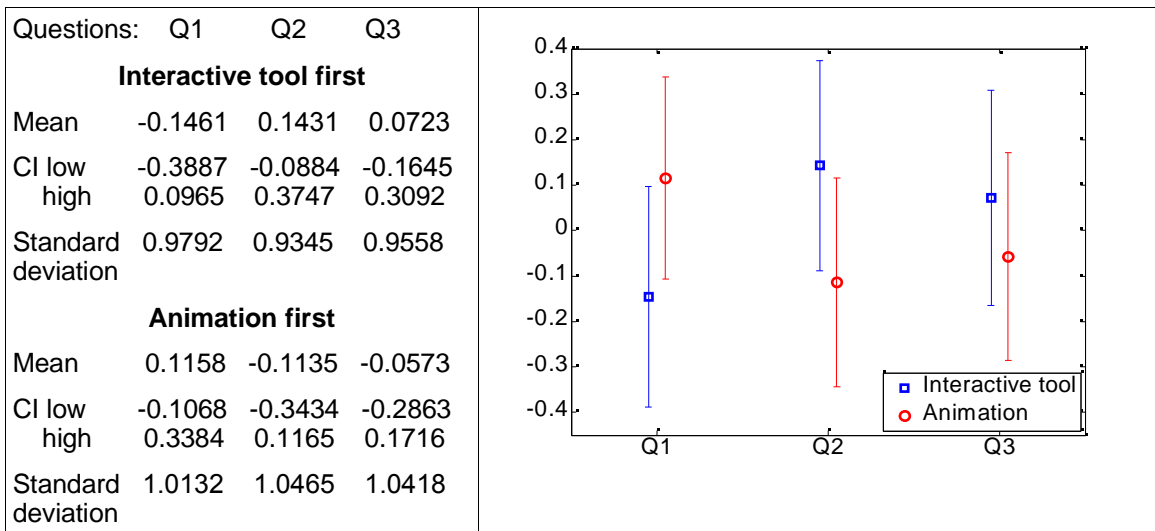
Figure C31: Complete data set (all exercises)



**Figure C32: Clean data set (all exercises)**



**Figure C33: Statistics for normalized, complete data set (all exercises)**



**Figure C34: Statistics for normalized, clean data set (all exercises)**

### C8: Exercises 1 and 5 combined

Following is the histogram for number of scores below 50% and number of scores above 50% for the normalized data sets of exercises 1 and 5 combined.

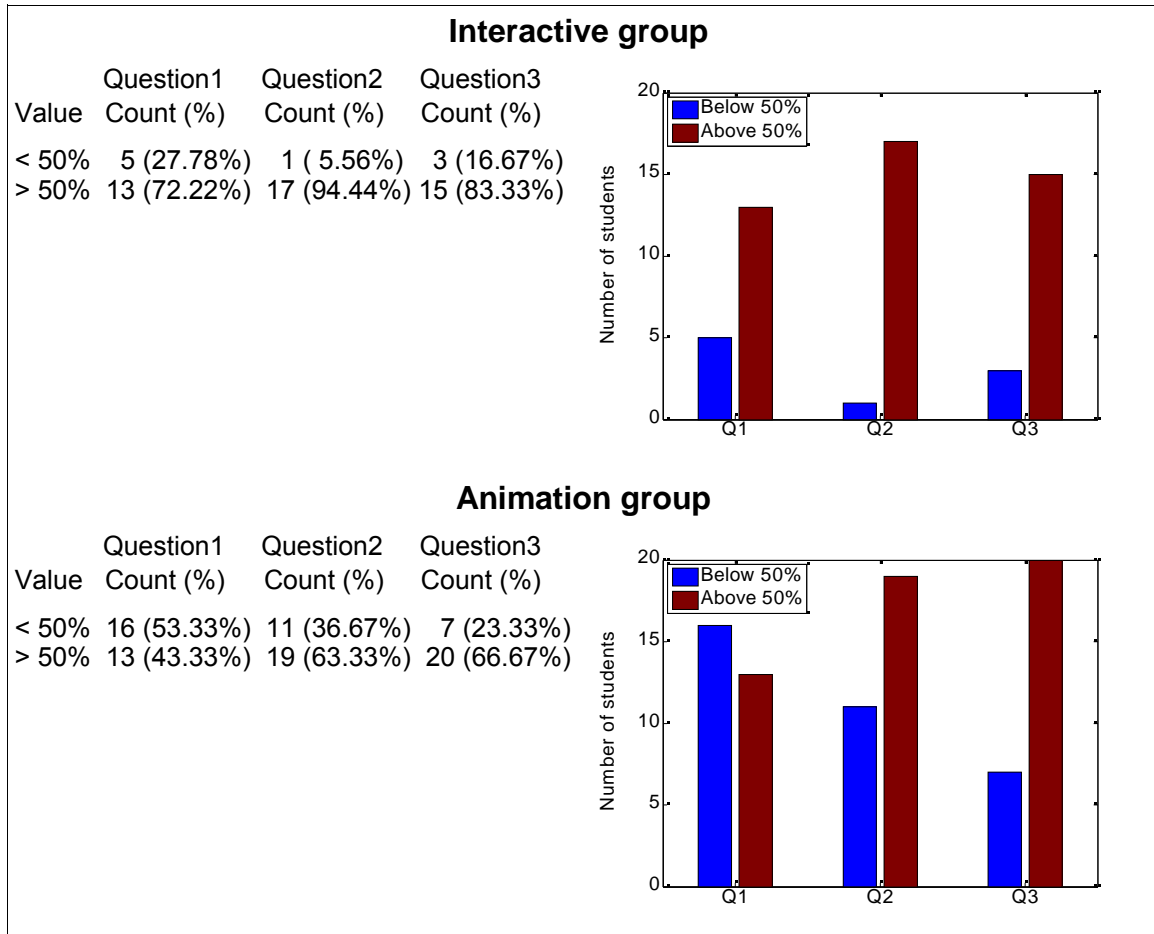
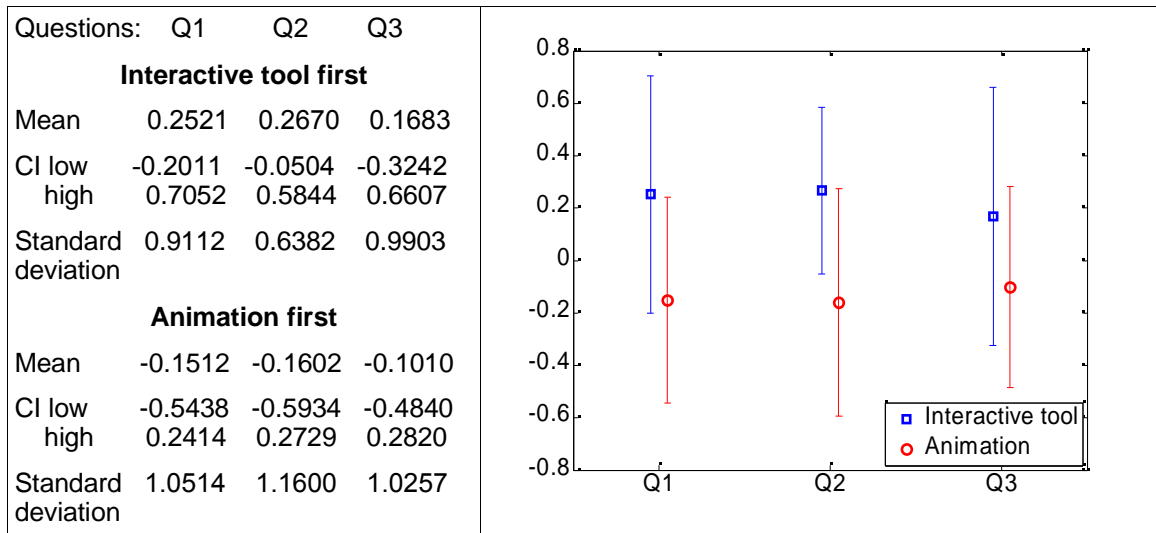


Figure C35: Clean data set (exercises 1 and 5)



**Figure C36: Statistics for normalized, clean data set (exercises 1 and 5)**

### C9: Exercises 2, 3, 4, and 6 combined

Following is the histogram for number of scores below 50% and number of scores above 50% for the normalized data sets of exercises 2, 3, 4, and 6 combined.

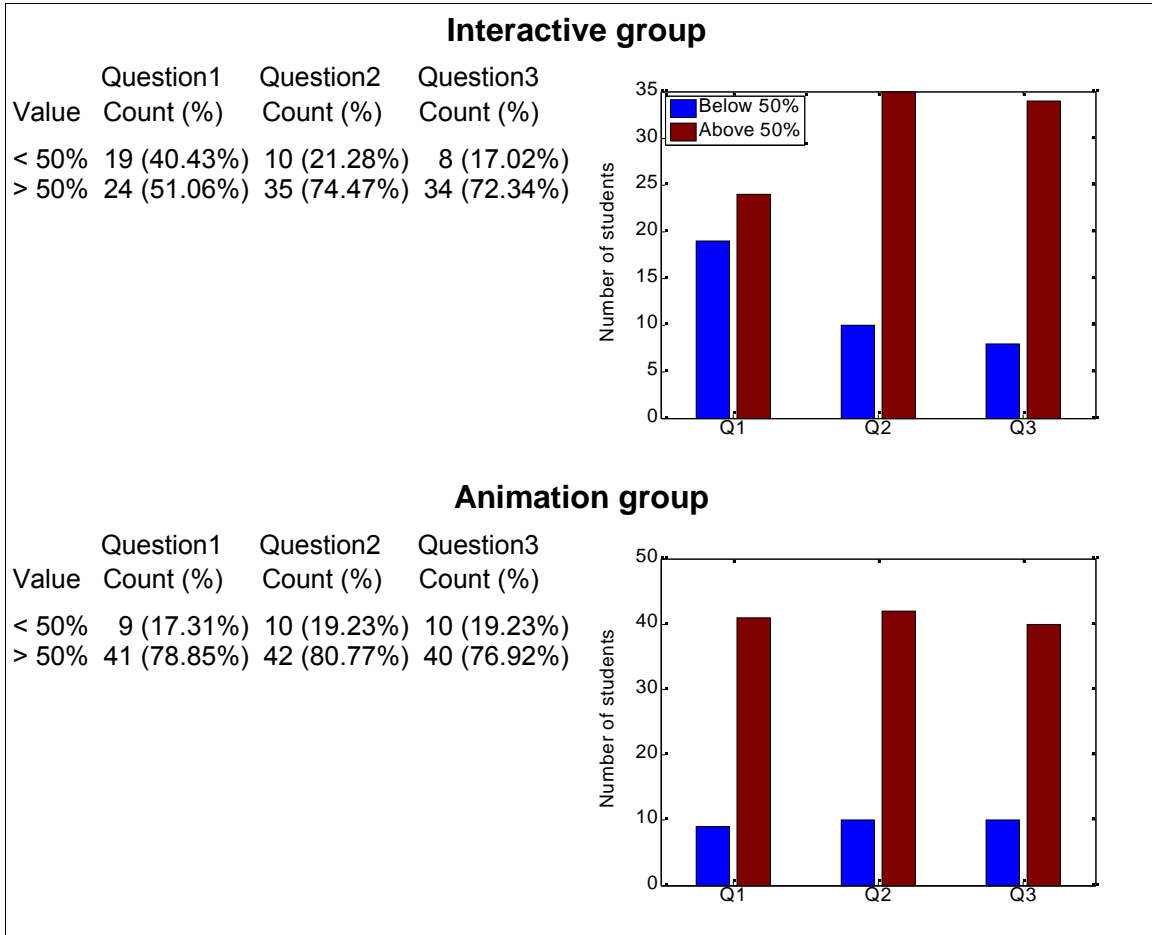
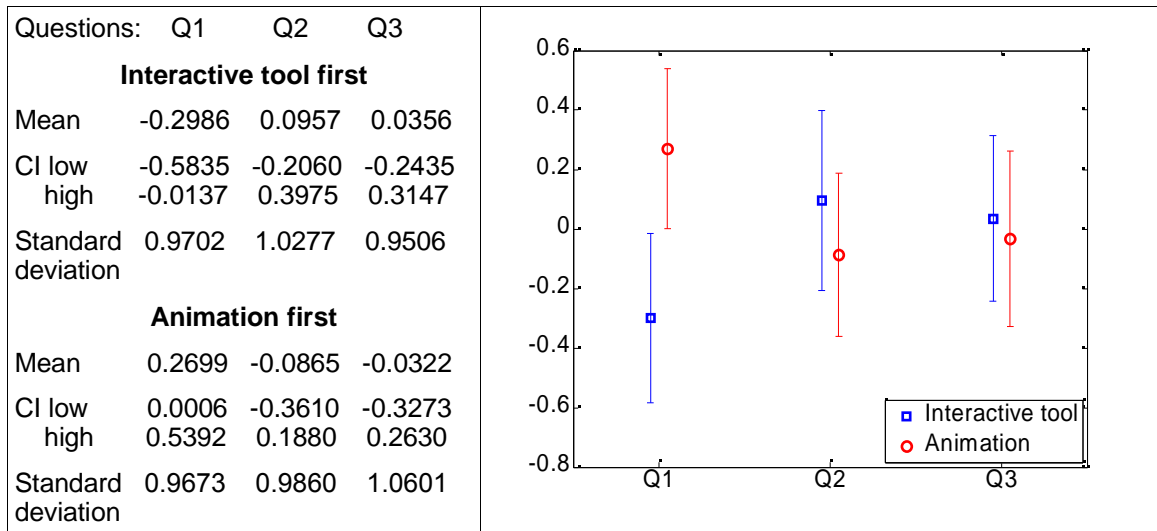


Figure C37: Clean data set (exercises 2,3,4,6)



**Figure C38: Statistics for normalized, clean data set (exercises 2,3,4,6)**



## C10: Time data for the exercises

Following figures show the data we captured related to the time spent on each of the tools.

Times (in seconds) by each student (sorted)			Basic statistics for the times		
#	Interactive tool	Animation		Interactive tool	Animation
1	172	103	Mean	358.3077	369.3529
2	187	124	Median	353.0000	247.0000
3	222	125	Trimmed mean (5%)	350.3636	292.0000
4	238	171			
5	284	190			
6	304	209			
7	353	234			
8	366	235			
9	441	247			
10	470	283			
11	493	285			
12	496	298			
13	632	360			
14		429			
15		569			
16		621			
17		1796			

**Figure C39: Statistics for time spent on first tool (phase2) of BFS exercise**

Times (in seconds) by each student (sorted)			Basic statistics for the times		
#	Interactive tool	Animation		Interactive tool	Animation
1	76	60	Mean	270.7647	382.5714
2	96	93	Median	178.0000	286.0000
3	99	102	Trimmed mean (5%)	200.9333	312.5000
4	125	115			
5	128	127			
6	133	160			
7	136	181			
8	156	391			
9	178	416			
10	185	489			
11	222	526			
12	229	547			
13	253	603			
14	356	1546			
15	358				

16	360	
17	1513	

**Figure C40: Statistics for time spent on first tool (phase2) of DFS exercise**

Times (in seconds) by each student (sorted)			Basic statistics for the times	
#	Interactive tool	Animation	Interactive tool	Animation
1	65	11	Mean	381.2500
2	65	45	Median	169.5000
3	100	45	Trimmed mean (5%)	260.5455
4	120	72		
5	133	84		
6	169	104		
7	170	113		
8	193	124		
9	304	129		
10	440	134		
11	1107	138		
12	1709	141		
13		142		
14		220		
15		412		

**Figure C41: Statistics for time spent on first tool (phase2) of Kruskal's**

Times (in seconds) by each student (sorted)			Basic statistics for the times	
#	Interactive tool	Animation	Interactive tool	Animation
1	99	43	Mean	582.8333
2	145	48	Median	446.0000
3	210	76	Trimmed mean (5%)	499.7000
4	218	84		
5	397	137		
6	402	162		
7	490	173		
8	569	189		
9	577	296		
10	645	525		
11	1344	532		
12	1898	642		
13		658		
14		3600*		

\* - times were trimmed at 3600 seconds max (i.e. time = min(time, 3600))

**Figure C42: Statistics for time spent on first tool (phase2) of Prim's**

Times (in seconds) by each student (sorted)			Basic statistics for the times		
#	Interactive tool	Animation		Interactive tool	Animation
1	542	12	Mean	1156	382.6000
2	581	12	Median	634.0000	212.0000
3	634	16	Trimmed mean (5%)	1156	244
4	1534	67			
5	2489	102			
6		151			
7		165			
8		212			
9		235			
10		308			
11		360			
12		368			
13		508			
14		900			
15		2323			

**Figure C43: Statistics for time spent on first tool (phase2) of Bellman-Ford**

Times (in seconds) by each student (sorted)			Basic statistics for the times		
#	Interactive tool	Animation		Interactive tool	Animation
1	3	4	Mean	210.8462	101.3000
2	78	6	Median	103.0000	66.0000
3	81	7	Trimmed mean (5%)	161.6364	101.3000
4	88	12			
5	93	46			
6	94	86			
7	103	112			
8	125	214			
9	147	246			
10	285	280			
11	290				
12	394				
13	960				

**Figure C44: Statistics for time spent on first tool (phase2) of Dijkstra's**

	Interactive tool	Animation
Mean	407.6528	323.1294
Median	233.5000	171
Trimmed Mean (5%)	336.1250	236.0130

**Table C1: Statistics for first tool of all six exercises combined**

	Interactive tool	Animation
Mean	579.89	375.56
Median	455.5	235
Trimmed Mean (5%)	486.06	281.25

**Table C2: Statistics for exercises 1 and 5 combined**

	Interactive tool	Animation
Mean	350.24	291.47
Median	181.5	137
Trimmed Mean (5%)	284.58	204.83

**Table C3: Statistics for exercises 2, 3, 4, and 6 combined**

## Appendix D: Survey

We conducted a short survey to get students' feedback about the tools they have used in the experiment. Following is the 2-page survey that was given out. The histograms for the feedback data are presented below.

1. For your exercises in this course, you had a chance to use animation applets and two different interactive animation tools. Below are the labeled snapshots of each tool.

<p><b>Tool A:</b></p> <p><b>Bellman-Ford Shortest Path</b></p> <p>W[] 0 1 2 3 4    w[] null null null null null null</p> <p>Iteration: 1</p> <table border="1"> <thead> <tr> <th>From</th> <th>To</th> <th>Weight</th> </tr> </thead> <tbody> <tr><td>v1</td><td>v2</td><td>11</td></tr> <tr><td>v2</td><td>v3</td><td>15</td></tr> <tr><td>v2</td><td>v4</td><td>15</td></tr> <tr><td>v3</td><td>v4</td><td>5</td></tr> <tr><td>v4</td><td>v5</td><td>11</td></tr> <tr><td>v4</td><td>v6</td><td>12</td></tr> </tbody> </table> <p>Next Edge    Update w[] and n[]    No Solution    Done    Show Next Step</p> <p>Directions</p> <ol style="list-style-type: none"> <li>1. Drag nodes as needed to help visibility. Note, for bidirectional edges, the numerical weight is drawn closer to the arrow head of that underlying directed edge.</li> <li>2. The search is from either vertex '0' or '14', as highlighted with initial path weight <math>W=0</math> and a grey vertex.</li> </ol>	From	To	Weight	v1	v2	11	v2	v3	15	v2	v4	15	v3	v4	5	v4	v5	11	v4	v6	12	<p><b>Tool B:</b></p> <p><b>Dijkstra's Algorithm</b></p> <p>From the current node, highlight an incident edge by clicking on it once. If the node at the other end should be updated, click on that node. Otherwise, click on the highlighted edge again to eliminate it. Note that while an edge is highlighted, you need to finish updating or eliminating. <b>Clicking on irrelevant nodes or edges has no effect.</b></p> <p>Once you are done with the current node, click on the next node to process.</p> <p>Hint 1: To help you, with addition, the value shown in ( ) for the highlighted edge = (distance of current node + weight of edge).</p> <p>Hint 2: If any value is not clearly visible, use the tool by hovering the mouse cursor over the node or edge.</p> <p>Hint 3: Cracking</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> being considered</li> <li><input type="checkbox"/> highlighted, awaiting action</li> <li><input type="checkbox"/> in the shortest-path tree</li> <li><input type="checkbox"/> eliminated</li> </ul>
From	To	Weight																				
v1	v2	11																				
v2	v3	15																				
v2	v4	15																				
v3	v4	5																				
v4	v5	11																				
v4	v6	12																				
<p><b>Animation A:</b></p>	<p><b>Animation B:</b></p>																					
<p><b>Animation C:</b></p> <pre> Q ← V[G] for each u ∈ Q   do key[u] ← ∞ key[r] ← 0 n[r] ← NIL while (Q ≠ ∅)   do u ← Extract-Min(Q)   for each v ∈ Adj[u]     do if v ∈ Q and w(u,v) &lt; key[u]        then n[v] ← u            key[v] ← w(u,v)   </pre> <p>Edge Weight: <input type="text"/> Enter</p>	<p>As a reminder, for your six exercises the following tools were used:</p> <p>Breadth-First Search: <b>Tool A and Animation A</b>    Depth-First Search: <b>Tool B and Animation B</b>    Kruskal's Algorithm: <b>Tool B and Animation B</b>    Prim's Algorithm: <b>Tool B and Animation C</b>    Bellman-Ford Algorithm: <b>Tool A and Animation B</b>    Dijkstra's Algorithm: <b>Tool B and Animation B</b></p>																					

Please answer the following questions about the tools and animations by circling the appropriate choice.

**How easy was the tool/animation to use:**

<b>Tool A:</b>	Very easy	Easy	Somewhat difficult	Hard	Did not use the tool
<b>Tool B:</b>	Very easy	Easy	Somewhat difficult	Hard	Did not use the tool
<b>Animation A:</b>	Very easy	Easy	Somewhat difficult	Hard	Did not use the tool
<b>Animation B:</b>	Very easy	Easy	Somewhat difficult	Hard	Did not use the tool
<b>Animation C:</b>	Very easy	Easy	Somewhat difficult	Hard	Did not use the tool

**How clear was the information presented (e.g. how clear were graphics and animation):**

<b>Tool A:</b>	Very clear	Clear	Not so clear	Confusing	Did not use the tool
<b>Tool B:</b>	Very clear	Clear	Not so clear	Confusing	Did not use the tool
<b>Animation A:</b>	Very clear	Clear	Not so clear	Confusing	Did not use the tool
<b>Animation B:</b>	Very clear	Clear	Not so clear	Confusing	Did not use the tool
<b>Animation C:</b>	Very clear	Clear	Not so clear	Confusing	Did not use the tool

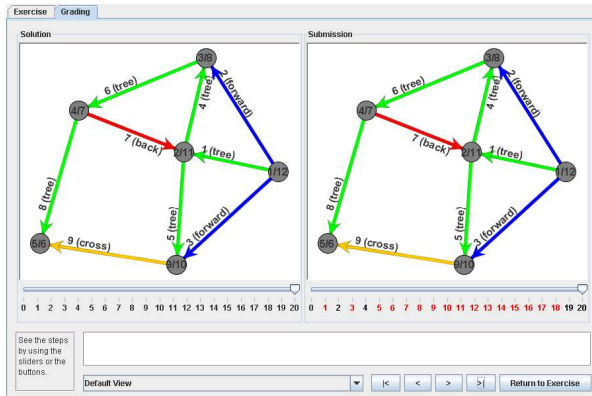
**How effective was this tool for learning the algorithms:**

<b>Tool A:</b>	Very effective	Effective	Somewhat effective	Ineffective	Did not use the tool
<b>Tool B:</b>	Very effective	Effective	Somewhat effective	Ineffective	Did not use the tool
<b>Animation A:</b>	Very effective	Effective	Somewhat effective	Ineffective	Did not use the tool
<b>Animation B:</b>	Very effective	Effective	Somewhat effective	Ineffective	Did not use the tool
<b>Animation C:</b>	Very effective	Effective	Somewhat effective	Ineffective	Did not use the tool

**Would you use this tool to learn new algorithms:**

**Tool A:** Yes No **Tool B:** Yes No **Animation A:** Yes No **Animation B:** Yes No  
**Animation C:** Yes No

2. Tool B, had a feedback feature that compared your answer with the correct answer (see picture below to refresh your memory). Please circle the option that best fits your opinion about this feature:



- Great
- Good
- Neutral
- Not useful
- Bad

3. For your exercises in this course, you had a chance to use several different animation applets that allowed you to create your own graphs. Please circle the option that best fits your use of this feature.

Created my own graphs    Did not create my own graphs    Did not use the applet

**If you created your own graphs, how many did you create on average per exercise:**

More than 10

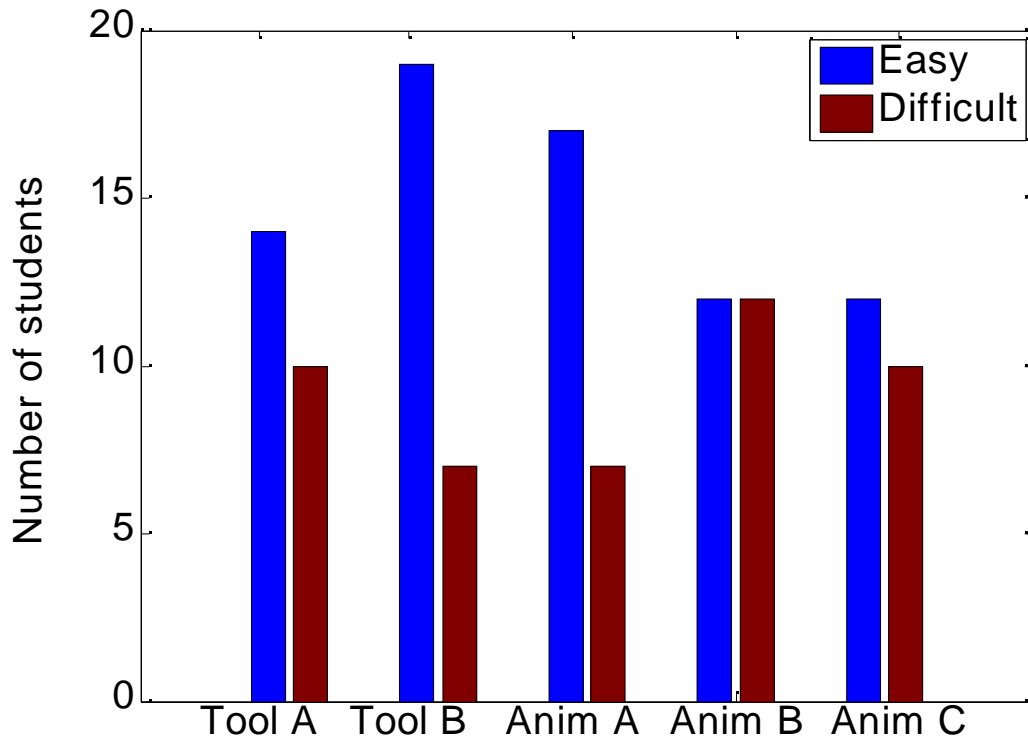
Between 5 and 10

Less than 5

**Question 1: Results**

	Very Easy	Easy	Somewhat difficult	Hard	Did not use
ToolA	3	11	9	1	3
ToolB	9	10	5	2	2
Animation A	4	13	5	2	3
Animation B	1	11	7	5	2
Animation C	2	10	8	2	5

**Table D1: Answers for survey Question 1**



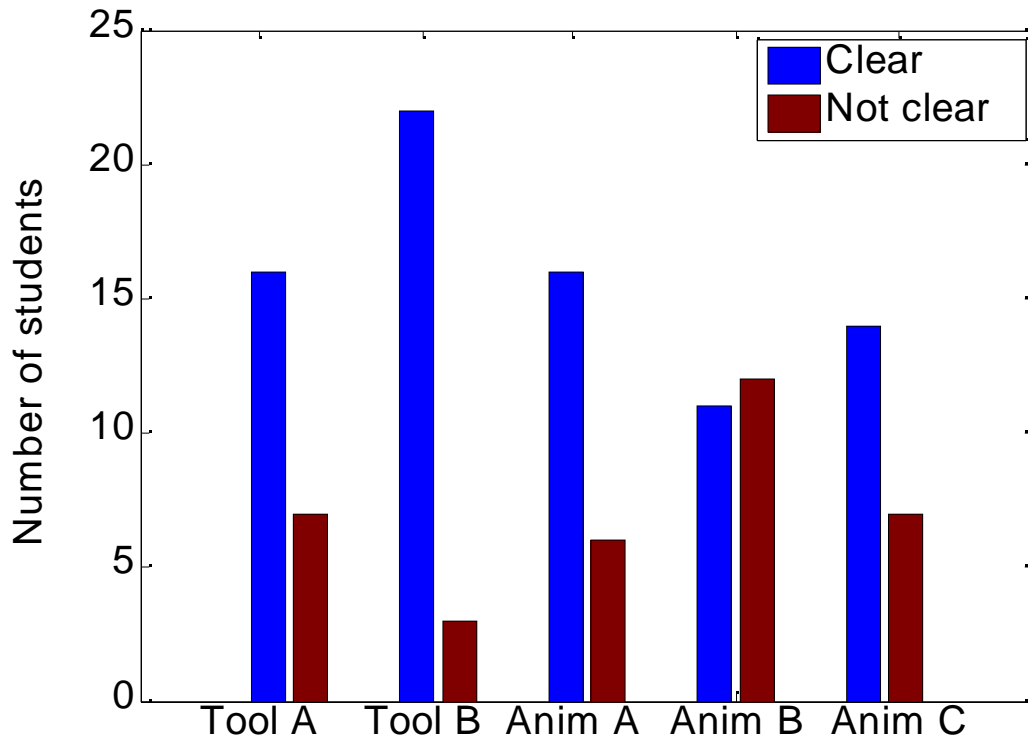
**Figure D1: Graph for Question 1 result**

Question 2: Results.

	Very Clear	Clear	Not so clear	Confusing	Did not use
ToolA	4	12	5	2	3
ToolB	11	11	2	1	2
Animation A	2	14	5	1	3
Animation B	2	9	9	3	2
Animation C	2	12	7	0	5

**Table D2: Answers for survey Question 2**



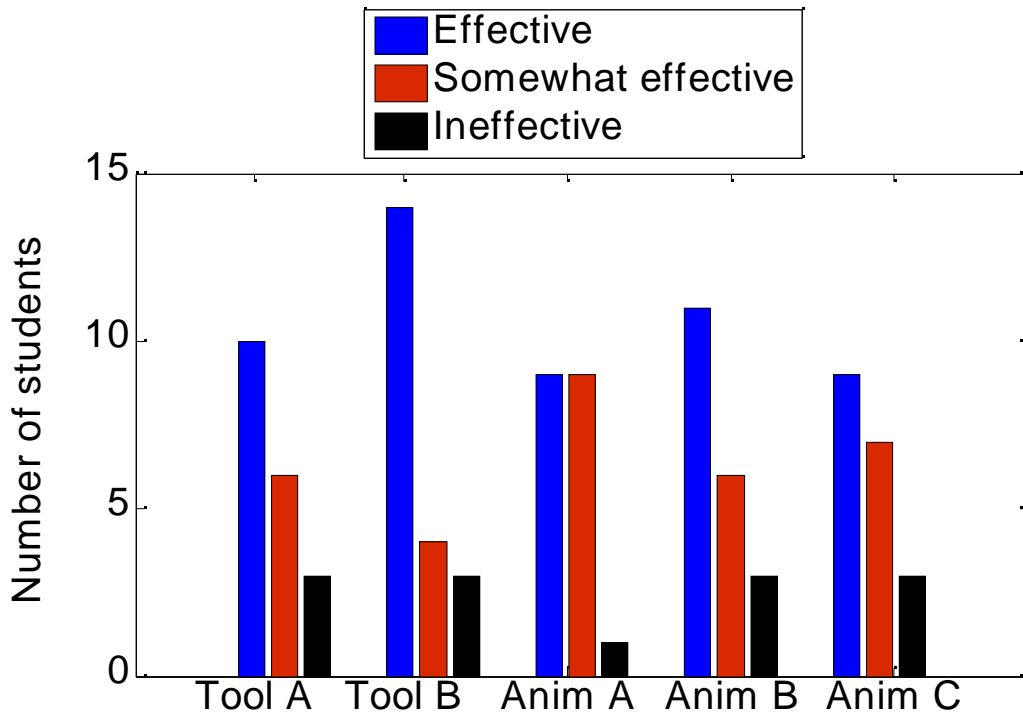


**Figure D2: Graph for Question 2 result**

Question 3: Results.

	Very effective	Effective	Somewhat effective	Ineffective	Did not use
ToolA	6	4	6	3	0
ToolB	9	5	4	3	0
Animation A	4	5	9	1	0
Animation B	4	7	6	3	0
Animation C	3	6	7	3	0

**Table D3: Answers for survey Question 3**



**Figure D3: Graph for Question 3 result**

**Question 4: Results**

	Yes	No
ToolA	12	2
ToolB	13	2
Animation A	11	3
Animation B	9	5
Animation C	7	7

**Table D4: Answers for survey question 4**

Question 5: Results

	Great	Good	Neutral	Not useful
Feedback feature	17	3	1	1

**Table D5: Answers for survey question 5**

Question 6: Results

	Created my own graphs	Did not create graphs
	11*	10

**Table D6: Answers for survey question 6**

\* - Note: 100% of students who created their own graphs, created less than five graphs.

## Appendix E: Adding exercises to Moodle

We chose to deploy our experiment in Moodle (see sections 3 and 3.3. for more details). Thus, Moodle was extended to provide the course teacher with ability to add exercises to the course. The primary goal of this extension was:

- To provide a simple way for the course teacher to add exercise modules to Moodle, where an exercise module is an applet that optionally reports some grading information back to Moodle.
- For Moodle to be able to receive the grading information from the exercise module and record it in the same way it records results of other modules such as quiz, assignment, survey, etc.

Because this extension to Moodle was not the primary objective of our project, only the minimum functionality was implemented, with the intention that this work would be continued by another student (as part of another project or teaching assistance, etc.).

In this appendix, we demonstrate how a course teacher can add exercise module to the course in Moodle. The following appendix F describes the technical aspect of this extension as a reference for someone who would be continuing the extension implementation.

The task of adding an exercise module to the course in Moodle involves very few steps and follows the same Moodle principles common to adding other types of modules such as assignment and quiz. Therefore, a course teacher with prior Moodle experience should not find the process of adding an exercise module too different from the process of adding other type of modules (this was one of the design goals).

Following are the steps the course teacher or the course administrator should take to add an exercise module:

1. Logon to the Moodle system and go to the main page of the desired course. Course teachers and administrators have authority to add content to the course. Therefore, there will be the “Turn editing on” option available, as shown on the figure E1 below. This option could be either a hyper link in the Administration panel (shown on the left side of the figure) or it could be a button (shown on the top right of the figure).

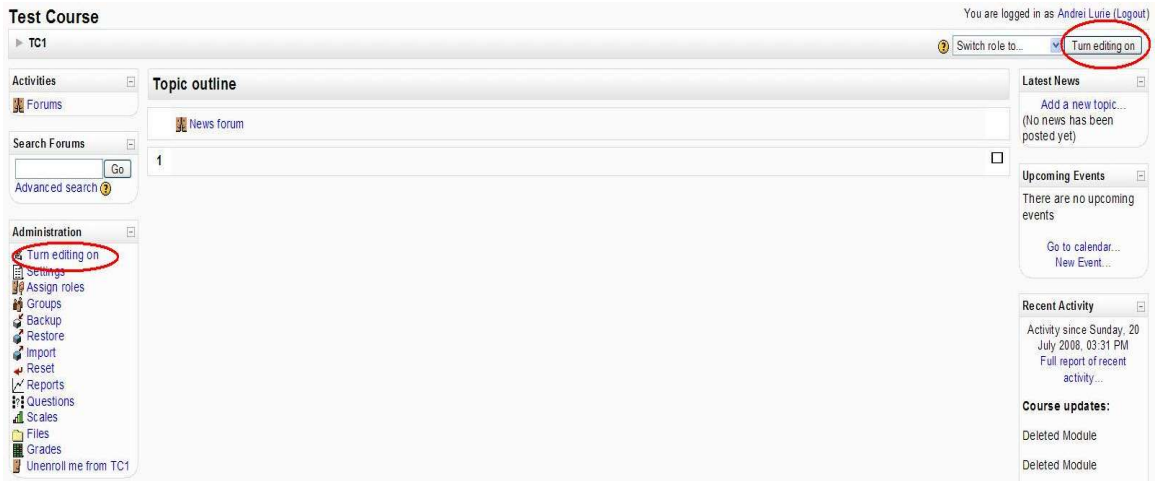


Figure E1: “Turn editing on” options are circled in red.

2. Click the “Turn editing on”. The course page will reload with extra options available. One of the options will be to add an activity to the course, as shown on figure E2 below.

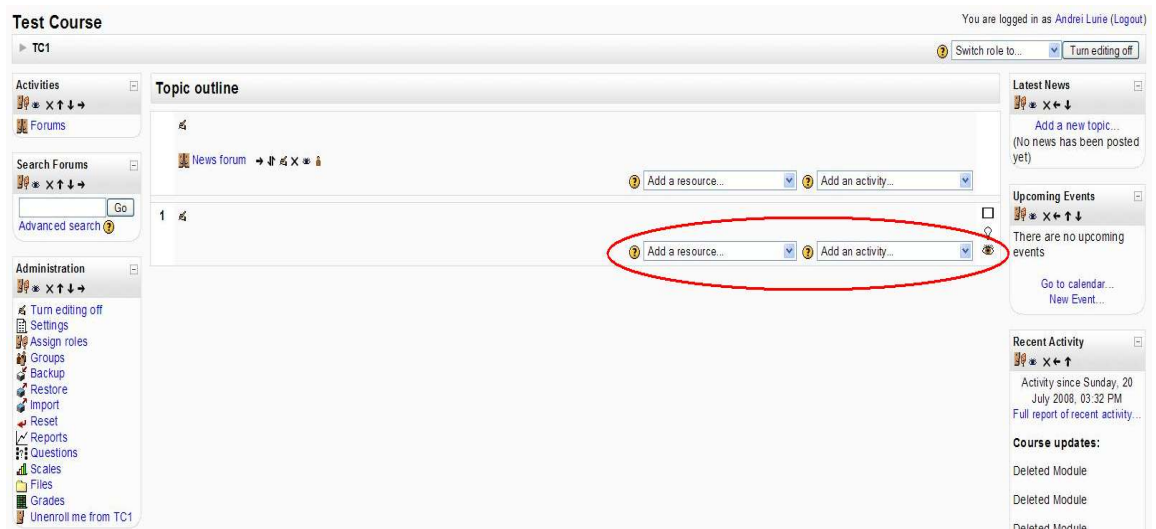
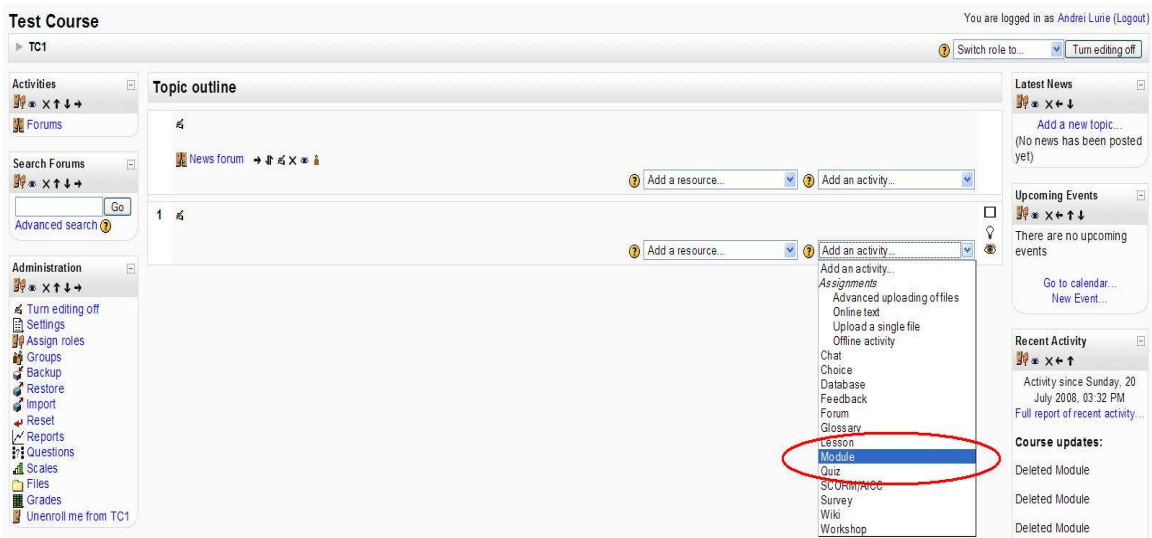


Figure E2: Editing options are circled in red.

3. Select “Module” from the drop down list of activities, as shown in the figure E3 below.



**Figure E3: Module option is circled in red.**

4. The “editing module“ page will be loaded, as shown in the figure E4 below. This page is an HTML form that allows one to specify the exercise parameters such as: the name and description of the module, grading options, timing options, etc. The currently supported fields are:

- **Module Name**  
This is the name of the exercise that will be displayed in the course page (i.e. it will be the name of the link that students click on to get to the exercise). This is a required field.
- **Exercise class name**  
This is the name of the Java class that implements the module. Specify the name without the extension (e.g. If the class name is DepthFirstSearch, specify DepthFirstSearch and not DepthFirstSearch.class or DepthFirstSearch.java)
- **Applet width**  
This option specifies the width of the exercise applet. The value will be copied, unchanged, into the HTML APPLET tag, and therefore the value can be specified in any form supported by the HTML APPLET tag. The default value is 900 pixels.

- **Applet height**  
This option specifies the height of the exercise applet. The value will be copied, unchanged, into the HTML APPLET tag, and therefore the value can be specified in any form supported by the HTML APPLET tag. The default value is 400 pixels.
- **Introduction**  
This option specifies an introduction text for the exercise. The introduction text is displayed when students click on the exercise link, and usually contains description of the exercise and directions.
- **Open the module**  
This option specifies the date and time when the module becomes available to the students. The students will not be able to access the exercise until the specified date and time. This option is disabled by default, meaning that the exercise becomes available immediately after it has been created.
- **Module closes**  
This option specifies the date and time when the module becomes unavailable to the students. The students will not be able to access the exercise after the specified date and time. This option is disabled by default, meaning that the exercise is always available.

Other fields in the form are not currently supported and could be developed in future projects.

## Adding a new Module to topic 1

**General**

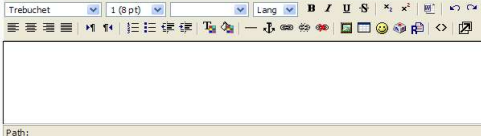
Module Name\*

Exercise class name\*

Applet width

Applet height

Introduction



Path:

**Timing**

Open the Module       Disable

Module closes       Disable

Time limit   Enable

Time delay between first and second attempt

Time delay between later attempts

**Attempts**

Attempts allowed

Each attempt builds on the last

**Grades**

Grade

Grading method

Apply penalties

**Figure E4: Add Module form (top part)**

5. After all desired options are specified, click on “Save changes” button. The exercise should now be created and ready to use.



## Appendix F: Design reference for exercise support in Moodle

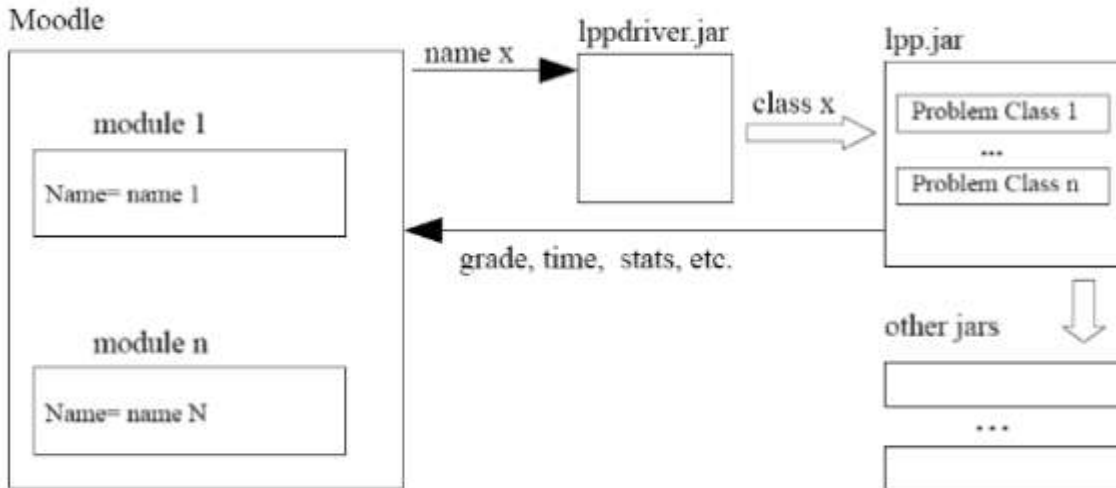
The secondary goal of the project was to provide the Moodle teacher an ability to add exercises developed under the learning productivity framework to Moodle course. One of the main design points was to provide a simple and generic way for teachers to add applets as an exercise assignment; no knowledge of Moodle internals should be required from the teacher.

Because the framework was implemented in Java, we decided that the quickest and most straight forward way would be to extend Moodle to allow course teachers to add exercises as applet modules in the same manner they would add a quiz. Furthermore, we wanted to make the extension generic enough so that any applet, not necessarily the one implemented under the framework, could be added to Moodle. Another important aspect of the design was to make Moodle independent from applet specifics, such as the class and jar names. This was especially important to us since, at the time of our project, the learning productivity framework was under development, and we needed to avoid any applet code changes (especially changes to class names) to require changes in Moodle.

As the result of aforementioned requirements, our design included a middle tier, called the Driver. The Driver is a Java class that has predetermined name which does not change. Moodle always invokes the Driver, passing it some information about the exercise (exercise name, user id, etc.). The Driver dynamically loads the appropriate Java class for executing the exercise. This way, the Moodle side has to be coded once - to invoke the Driver. Afterwards, no matter what changes have been done to the exercise code, Moodle side does not need to change. More specifically, we devised the following scheme:

- The name of the main applet class never changes (the name is LPPApplet). Moodle always references this applet name in the “code” part of the <APPLET> tag (e.g. <APPLET code=”edu.sjsu.cs.lpp.moodle.LPPApplet”... ). This is set up once by the administrator during installation and does not need to be changed.
- When an exercise module is created in Moodle by the teacher, a unique name is specified for each exercise. Typically this unique name is the same name as the java class that implements the corresponding problem (e.g. InsertionSort). The name is passed to the LPPDriver, which, in turn, uses it to determine which problem class to instantiate and run. This indirection allows us to freely change the code on the applet side (even changing the name of the classes) without requiring any changes by the Moodle administrator/teacher! LPPDriver simply maintains the mapping between the problem name used in Moodle and the actual Java class that corresponds to this problem.

This scheme is extremely useful for agile development, and allows more flexibility for migration. Figure F1 below demonstrates the design, as well as the data flow between the components.



**Figure F1: Overall design and data flow**

As shown on the figure F1, Moodle invokes LPPDriver.class that resides in lppdriver.jar. It sends the name of the problem that it wants to invoke (along with other information such as user id). LPPDriver maps the name sent by the Moodle to the Java class associated with the problem, loads the class and invokes it. The code that implements the problem reports the problem results such as grade and time (as well as user id) back to Moodle via the HTTP POST. To simplify the task of reporting problem result back to Moodle and to reuse the code, a Java interface LPPAppletCallback is provided which all problem classes should implement. Once LPPAppletCallback is implemented by the problem class, the reporting to Moodle can be simply done by invoking ReportResult method.

Both LPPDriver and LPPAppletCallback are part of the learning productivity framework, and reside in its own package called moodle (edu.sjsu.cs.lpp.moodle). The compiled jar files need to be copied into the Moodle (into the moodle/mod/lpp directory).

A more current and detailed design documentation is available at [http://mirror4.cvsdude.com/trac/sjsu\\_cs/lpp2007/attachment/wiki/AndreiLuriesWeeklyReport/moodle\\_integration\\_doc.pdf](http://mirror4.cvsdude.com/trac/sjsu_cs/lpp2007/attachment/wiki/AndreiLuriesWeeklyReport/moodle_integration_doc.pdf)