A Literature Review on Tools for Learning Data Structures

Shirin Patel
University of Cape Town
CSC4000W - Honours Project
May 15, 2014

Abstract

Data structures are a conceptually demanding topic which confronts many Computer Science students early in their course. The topic has a strong conceptual basis and often proves difficult for many to grasp. A number of previous studies have examined that the use of interaction and visualization within the systems can motivate a student to engage in the learning process. This literature review investigates the effectiveness of these systems that were and are being used today for teaching and learning of data structures to novice Computer Science students. It also explores the different techniques that are used to develop the intelligent tutoring systems and concludes on presenting which techniques are most effective.

Introduction

Learning data structures is a complex issue for many students and complexity is defined as a problem that can have several solutions. Some of the complex problems in the computational areas include; teaching programming, algorithms and data structures, among others. Data structure is an important subject for 2nd year students at UCT, but it is difficult to work with them due to their abstract nature. Interactive tools are created to make data structures learning easy and interesting. While using these tools, they understand how the algorithm works and how the operations (insertion, deletion, searching, traversal, merging, etc.) are executed.

This study reports on various papers on how researchers/Computer Scientists have gone about to tackle the problems faced by students in learning data structures and various approaches that are taken to make data structures learning more fun and effective. This literature review includes various tools, technologies, advantages of using these technologies and suggestions on the techniques that can be combined for effective learning of data structures. Before discussing the approaches, I will review problems faced in learning and teaching data structures and why the data structures are being taught in the first place.

We will ultimately attempt to answer the following research question: What functionalities and usability aspects can be used in an intelligent tutoring system to make data structures learning fun, interactive and effective and will eventually improve students’ knowledge on data structures?
Problems faced

The first difficulty identified by teachers and tutors, was the low motivation of students according to Mesialo, et al: “What on earth do these (theoretical) algorithms and data structures have to do with my future (practical) job?” The second difficulty talked about was how tricky data structures can be and how they often remained abstract to students. Third stated that the assignments were done individually and students were not encouraged to cooperate. Lastly, the problem settings were closed: so for example, a student might be asked to implement the depth-first search algorithm. This made the assignments distant from any research or real problem setting.

The most common approach at teaching data structures requires some pictures with explanations and most tertiary institutions such as the University of Cape Town makes use this approach. In this context, students learn through observation but this is not as effective since it is not one-on-one approach, in that, student cannot learn in their own time. A number of prior studies have found that using algorithm visualization in a classroom had no significant effect on student’s performance.

Reasons for teaching/learning data structures

There were 5 categories of instructor rationale identified for the purposes of teaching data structures (Raymond et al, 2004), but there were merged to 4 in this context:

1. Developing Transferable thinking: Here, data structures were described as a vehicle for developing thinking and problem solving skills. “The design of a data structure is like the solution to a riddle: the process of developing the answer is as important as the answer itself.”

2. Improving programming skills: Implementation of data structures is used to improve programming skills of students, especially their dexterity with recursion and pointers. “...reading and using the code without having written something similar is like watching Olympic ping pong on TV. It sure looks easy, even somewhat repetitious; however, the level of precision is only experienced by trying to do the same.”

3. Knowing “what’s under the hood”: Students often use libraries to implement data structures and algorithms. This category acknowledges a place for learning the libraries and modifying them to serve their purpose. “A graduate should be convinced that fancy technology is understandable, and adjustable; they should feel that they can be masters of the magic that the Wizard hides behind the curtain.”

4. Component Thinking: This category puts emphasis on the importance of student learning component engineering principles, such as black-box interaction and code re-use. “Software Engineering is moving away from emphasis on the creation of code, toward emphasis on components and code reuse.” This avoids the process of re-inventing the wheel.
All of the above reasons gave a strong indication on how important it was for the students to learn data structures as part of their Undergrad Degree.

**Tools and their Outcomes**

There have been a number of tools developed for the purpose of teaching and learning data structures. We will look at those specific to our problem domain.

**The DSL Tool**

An interactive visualization tool, the Data Structures Learning (DSL) discussed in Alhousban (2011) paper, was developed and used first in a short mini study that showed that, used together with visualizations of algorithms, and aural instructions, it produced faster student response times than did textual instructions. This result suggested that the additional use of the auditory sensory channel reduced the cognitive load of the student. It was then used in a study over two academic terms in which students studying data structures module were offered the opportunity to use the DSL tool with either aural or textual instructions.

The collected data showed that the DSL tool was extensively used by weaker students. A comparison was made of the students’ DSL use with their end-of-year assessment marks which revealed that academically weaker students had tended to use the tool most. From the evaluation of the DSL tool, it was concluded that less able students were keen to use any useful and available instrument to aid their understanding, especially of difficult concepts.

**The Vedya Tool**

Vedya tool discussed in Segura P et al (2008) paper was used for visualizations of data structures and algorithm schemes. The pedagogical aim of Vedya was to facilitate the student’s grasp of the target procedures of education in Computer Science by means of interactive learning, in order to facilitate teamwork and communication between teachers and students. It managed the administration of the individual students’ homework, including generation of exercises, tests, grading the deliverable homework, and storing achieved results. It covered the most common data structures: stacks queues, binary search trees, AVL trees, priority queues, sorting and hash tables.

It made use of interesting visualization techniques through a maze representation. For example, when the user inserted an item in a queue, the truck was shown to throw the item on top of the maze. When extracting an item, the end of the maze opened and the first item would fall down. The use of the maze illustrated that items could not jump over the previous ones and the fact that in a queue, items were extracted in the order that they came in.

Vedya tool helped students benefit from complementary and interactive material, facilitating the intuitive comprehension of most typical operations of classical data structures without any restriction of time or material.

**Sketch based Interfaces**

We are entering a new era of teaching, learning and computing. Adamchik’s (2011) vision was to design a pen based computing environment in which student itself would draw data structure using tablet and stylus. For this, they had to develop an interface with four key parts:
Stroke recognition and beautification,
The association of strokes to an underlying domain-dependent data structure,
The animation of the algorithms, and
The verification of those algorithms.

The idea was to draw a particular instance of data structure using stylus and invoke an algorithm to animate over this data structure. Two study modes were identified according to students’ cognitive learning style. The first mode was the initial learning of the algorithm by stepping through it either using textual description or graphical approach. In the second study mode, students actively participated in the learning process. An example would be student tested themselves in preparation for the exam.

This system was developed after much evaluation on the how students learned data structures and algorithms. The evaluations indicated that the algorithm animations had no significant effect if the students were just observers. Student must be actively engaged in order to learn. Therefore, electronic education technology integrated with a Tablet PC environment had a fundamental influence since it allowed expression and the exchange of ideas in a highly interactive atmosphere. Using pen-based gesture interface, the system promoted the students’ intuition for both problem solving and algorithm thinking.

**VIDSAA Tool**

Kacha & Ron (2006) reported on a project which had developed a range of learning objects\(^1\) that helped students to learn about the different data structures and the algorithms by which they were controlled, called VIDSAA (Visualization in Data Structure and Algorithms). In this tool they covered data structures like Lists, Stacks, Queues, Trees, Sorting & Searching and Graph. There were a total of 55 animations. Each animation provided an interactive representation of the algorithms and data structures which the student could control. In this, they visualized the data structure using programs and flow chart representation.

The use of play, stop, forward and backward buttons enabled the student to step through the processing and to observe the changes to the variables and the relevant data. This feature was intended to enable the students to pause and think before watching a further step of the animation and this was intended to provide an opportunity for students to become active learners. VIDSAA was designed with a capacity for installation onto a student’s computer for the use of outside the classroom activities as part of their independent learning.

**Different approaches to learning Data Structures**

Three most popular methods were identified that are being used today to make data structures for both learning and teaching, effective and fun. The tools mentioned above integrated some or all of these methods in their systems. We will discuss how each approach has been useful and which were successful.

\(^1\) Modular resource, usually digital and web-based, that can be used and re-used to support learning activities.
**Multimedia-interactive Systems**

Andrade et al (2008) compared three different teaching approaches, namely, traditional teacher-led instruction, Web pages and a multimedia-interactive system to determine which method works best. Three different groups were involved, each group using a different approach. This system was specifically designed to teach binary trees.

The Web page contained hyperlinks to navigate through the information and examples. The information was shown by text and images. There was no animation and interactivity in the Web page. In this case the instructor had to check and evaluate the students’ exercises. The multimedia-interactive system kept the same webpage, but there were some interaction differences: animation and sounds as explanatory sections of each topic (insertion, deletion and searching nodes). Also, this system included interactivity where students had the ability to answer interactive exercises by moving data and images. In this case the instructor did not check and evaluate the students’ exercises, the system did this activity.

The first two groups of students were tested based on a written test and the multimedia group students were tested on the system. After the results were evaluated, it was concluded that multimedia could effectively be used to help students learn data structures specifically binary trees.

**Learning through Graphics**

τέχνη (or TEXNH) project discussed by Duchowski & Davis (2007) presented a first-time implementation of a data structures and algorithms course based on a specific computer graphics problem, namely surface reconstruction from unorganized points, as the teaching medium. The primary goal of this project was to incorporate graphics projects in their undergrad computer science degree curriculum. The graphics projects ranged from traditional graphics problems such as ray tracing\(^2\) to cutting edge implementations.

In first year curriculum, image processing was introduced to teach the students how data structures algorithms are applied on an image. In second year, ray tracer had to be implemented because it covered broad range of computer science concepts; provided visual feedback; allowed program correctness to be identified immediately; and lead to the understanding and implementation of OO (Object Oriented) paradigm.

These concepts of image processing and rendering lead students to understand and learn two-dimensional arrays and dynamic memory allocation. Although from a graphical point of view this attempt had its moment of success as it was entirely based on developing problem solving skills, but it was not focused on learning the basics of data structures in particular. And therefore this approach seemed overwhelming and too heavy for an introductory course.

\(^2\) A technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.
**Game-based learning**

“Active student learning is especially important in an introductory data structures course where student learn the fundamentals of programming”. Lawrence (2003) described a project for learning data structures based on the idea of competitive programming. It motivated student learning by allowing students to evaluate and improve their programs throughout an assignment by competing their code against instructor-defined code and the code of other students in a tournament environment.

The ability of students to evaluate their code against others encouraged them to spend more effort in its development. In the project, described in this paper, students wrote their code that became the “artificial intelligence” for the computer to play a game. The student code was then uploaded to a server that allowed the students to challenge each other’s code. The interface was entirely web-based. Students were exposed to fundamental programming constructs, such as lists, stacks, queues, and trees.

Games captured student interest because they were fun and exciting, and students tended to learn more when they were actively engaged by the subject. This was most innovative method used to teach/learn data structures so far in any Computer Science course.

**Conclusions**

There are many factors which can influence the success of learning objects as support for student learning in university settings. Students will often need to be motivated and encouraged to use the independent learning resources. Also, it is unclear as to what forms of learning support best facilitate their use.

However, it is clear from the papers considered in this review that when designing tutoring systems, interaction with the system is the most important thing to consider for engaging students in the learning process. These interactions include: the use of multimedia like text supported with aural instructions, animations like the maze representation used in the Vedya tool, and tutorial videos for familiarization with the system.

The graphical approach seemed promising, but it is ineffective in this particular context since it uses much advanced techniques like image processing to teach data structures and misses the whole point of the learning process. The gaming approach described by Lawrence (2003) in his paper is a very interesting thing to consider when designing tutoring systems since it engages the students in the learning process. The motivation behind seeing other students’ code by contrasting it with their own code would eventually improve the students programming skills and will encourage them to work harder.
Bibliography


