Data Structures taught in a Web-based Environment

Neha Kumar (neha@cory.eecs.berkeley.edu)

Abstract

This paper suggests modifications in the traditional approach followed to instruct Computer Science students and discusses a new web-based medium of imparting the same education. It focuses on the design of curriculum for a lower-division Data Structures course, tailored to be taught using an online learning environment for students. It also elaborates on the benefits of this approach in comparison with the traditional form of instruction of the course and discusses research questions yet to be explored in the near future.

Background

The traditional approach to Computer Science education, which requires a student to attend lectures, a laboratory section, and/or a discussion section, has been in practice for several decades. Technological advancements in the field of computers enabling widespread use of the Internet make it worth the while, however, to explore the efficacy of other possible approaches towards this end – the Internet itself. The WISE (Web-based Inquiry Science Environment) tool developed by the School of Education at University of California, Berkeley offers one such approach.

WISE enables students to learn in an online environment with its inclusion of several different activities such as online reading, real-time online discussions, brainstorming sessions, online note taking etc. Instead of passively listening to lectures, WISE allows the students to participate more actively in the learning process. This increased participation is made possible by the extensive nature of the activities the package has to offer, essentially allowing its learners to do all they could in a classroom learning environ and much more.

The class of people benefiting from this mode of teaching includes not just the students, but the instructors as well. Using WISE, instructors are able to monitor the progress of their students by checking on their level and quality of participation periodically. This allows them to identify their weak and strong pupils with greater ease, recognizing problem areas faster and addressing them better, thus performing better as teachers. Minimal response time and constant feedback enrich the roles of both the student and the teacher.

The WISE tool for instruction consists of three components: a course builder, which allows the instructor to add to the online database of student exercises and activities; an online learning environment that delivers activities to the student and stores their work; and a course portal that serves as the students' primary interface to the system.

The main focus of this paper is to discuss how WISE fits in with imparting the Computer Science curriculum to undergraduate students, specifically with respect to the Data Structures course offering at UC Berkeley: CS 61B. It also discusses the adaptation of the 61B curriculum to suit the framework of WISE.

Related Work

During the course of the Summer 2002 session at UC Berkeley, the Computer Science department's introductory course in programming, CS 3, was taught using the WISE learning environment. This was done with the intention of testing out a new version of WISE that had been developed over the previous semester to judge its potential as an effective teaching tool for CS 3.

The class typically consists of students unfamiliar with the concept of programming, without much experience with computers. In addition, since more of them are freshmen, CS 3 also ends up being one of the first courses they take at college. Taking these factors into account, it becomes the responsibility of the instructor to pay cautious attention to the pace of the course and ensure that there are no large jumps in the difficulty levels of the course material.

The content of the course, as it has been taught over the years, includes functional programming, recursive programming, higher-order recursive techniques and lists, in that order of instruction. The session ends with the students writing approximately 200 lines of code in what constitutes for most their first actual Computer Science project. This is intended to prepare them for harder course projects to come in semesters that follow.

Traditionally taught in four hours of lecture, two hours of discussion (supervised by a Teaching Assistant) and four hours of mandatory laboratory work during a regular summer session, the course was modified this summer to include fourteen hours of laboratory sessions in a week structured thus:

- quizzes testing the previous day's material
- programming tasks – composing and analyzing programs
- online reading assignments that presented concise summaries of the text material
- collaboration with other students

The results at the end of the offering, in the form of a final exam for the students, were quite positive. The same exam administered under the classroom mode of instruction in 1994 had yielded an average score of 25.8 out of 60. The summer group of students was able to raise this average by 7.1 points to 32.9. The course evaluations were also overwhelmingly positive in favor of the WISE learning medium and the instructors. One student commented:

"I like the way lectures are combined with problems online. I am learning more about CS than I ever did before. This method of teaching is very practical and works a lot better than regular lecture/lab/discussion sections. … the professor is always there ready to answer any question you may have. People are not afraid to ask the stupid question because it does not interrupt anyone else's progress. I really support this type of teaching."

Building enthusiasm for a course in the minds of the students is a challenging task. To receive encouraging feedback from students thus is indeed a strong indicator of the success of such an experiment.

Design

The benefits of this online style of instruction naturally lead one to wonder how far we can take this approach while it continues to yield positive results. So over the summer we planned to take it one step further and adopt this style for the instruction of a pre-requisite course for Computer Science majors at UC Berkeley: Data Structures (or CS 61B).

CS 61B and CS 3 are different in many aspects that make it an interesting task to explore whether the former can be tailored to the WISE way of instruction analogous to that of the latter. More importantly, can it achieve the same level of success?

Students taking this course on Data Structures, unlike in CS 3, are expected to be equipped with considerable previous programming experience at the time of enrolment. The size of their programs is no longer limited to 200 lines, and is split into a larger conglomeration of files. Development of WISE for the instruction of this course must allow for this.

There is a large leap in the difficulty level of Data Structures concepts as well. Designed to familiarize the students with key concepts in the realm of Computer Science, the course is undeniably much more complex to learn than is CS 3. There are non-trivial algorithms that need to be understood and implemented, as well as intricacies in the data structures taught that need delving into.

Design is of key importance in 61B. This is the class in which students have traditionally been taught to design large projects and in which they learn the skills of modular programming. This is one aspect that students would prefer human instruction for.

With all of the above-mentioned differences, it remains to be seen how well WISE will fit the curriculum. The positive aspects that make this experiment worthwhile include all those that made CS 3 worth the effort, as was demonstrated by the results of the summer experiment. In addition, the concepts that are taught in CS 61B are few in number but cover several fine details amenable to being incorporated into independent WISE activities. Moreover, modular programming skills can be inculcated by asking students to organize their modules into various WISE activities. By answering specific design questions before they embark on the implementation of the project itself, the instructor can check to see that they are on the right track. The online collaborative tools can also help students view differing perspectives on the design process, where they might otherwise have been averse to walking up to every other student in lab and asking for their views.

As taught traditionally, Data Structures broadly proceeds along the lines of the following topics:

- Introduction to Java
- Activation Records
- Inheritance in Java
- Exceptions in Java
- Testing methods
- Linked Lists & Arrays
- Game-tree Search
- Asymptotic Analysis
- Hash Tables
- Trees

- Priority Queues/ Binary Heaps
- Binary Trees
- 2-3-4 Trees
- Splay Trees
- Graphs
- Disjoint Sets
- Sorting Algorithms
- Randomized Analysis

After these broad topics are identified, they need to be represented as WISE components. Each of the above topics is designed to be a WISE Project or, depending on the extensive nature of the topic, a series of WISE Projects. Every Project consists of various activities which may progress in the following typical manner:

- Display of text introducing the concept
- Note-taking by the students
- Collective brainstorming
- Exercises to illustrate each new concept studied
- Online discussions asking the students to answer fundamental questions on the topic

Graphs: A Case Study

A case study on graphs and how it is taught in CS 61B would serve to illustrate the format of this WISE teaching tool. The study progresses in the steps outlined below:

Project: Graphs

- Introduction
- Terms to Know
- Representation: Adjacency Lists and Matrices
- Traversal: DFS and BFS
- Homework: Modeling a Map of Berkeley

First, it is important for the instructor and the student to see eye to eye on the significance of the course topic. The introduction of graphs to the students, therefore, must necessarily illustrate that graphs are a critical part of the study of Computer Science and are revisited in various spheres of Computer Science, such as Networking, Compilers, Algorithms etc.

This introduction would best be followed by an overview of the terminology used in reference to graphs that a majority of students would be unfamiliar with, e.g. degree of a vertex, how directed graphs are different from undirected graphs etc. The idea is to provide the students with background information on graphs as well as to help them 'break into' the topic and feel comfortable with it.

After learning what graphs are and how they look in 'picture' form, students should be taught how this picture can then be translated into the language of the computer, i.e. using an adjacency list and/or an adjacency matrix. Exercises under this sub-topic would help demonstrate the idea with further clarity and students can be asked to convert a graph from a picture to a list or a matrix and vice versa, testing them at various difficulty levels. Understanding when one representation works out to be more efficient than another representation is best illustrated to them as they see the difference in the two representations. In particular, giving them a sparse graph and asking them to provide its representation in both forms, then giving them a populated graph and asking for its representation would drive home the point. The greater the room for exercises, the stronger their foundation of knowledge in the subject, and the better they learn.

Graphs become much more complex to deal with when one has to learn to traverse them. This is where Depth-First Search and Breadth-First Search enter the picture. At this stage, students are not quite as familiar with writing recursive programs as they need to be to tackle traversal of graphs. An abundance of exercises asking students to conduct DFS and BFS on various graphs helps them to understand the algorithms to multiple levels of recursion. Again, the student gains expertise in the topic as he/she learns by experimentation and implementation rather than simply listening to the theory of graph traversal.

At an introductory level, students are best off not being fed an overdose of information that they end up finding hard to digest. At this stage in the project they are well-educated with regards to the critical concepts underlying graphs. As a homework exercise or a mini-project, they can be asked to model a street map of Berkeley – starting with interpreting words (information regarding street names and intersections) in terms of pictures (in the form of a directed/undirected graph). This can be followed by asking them to provide both an adjacency list and an adjacency matrix representation of the graph, as well as their evaluation of which would be more efficient. Specific examples of DFS and BFS conducted on the graph could be asked, testing them thus on the entire breadth of their knowledge of the topic. Such a case study proves useful to the students as it puts the various segments they have learned about into perspective.

This sample case-study illustrates a bottom-up approach consisting of a variety of WISE "activities" and "steps". In WISE we would represent it as follows:

<u>Project:</u> Day 1 of Graphs
    Graphs – an introduction
<u>Activity:</u> Introduction
    <u>Step 1:</u> Reading
    What are graphs?
    Why are graphs important?
<u>Activity:</u> Terms to Know
    <u>Step 1:</u> Reading
    Definitions
<u>Activity:</u> Representation of Graphs
    <u>Step 1:</u> Reading
    What are adjacency matrices?
    <u>Step 2:</u> Exercises
    Convert graphs to matrices
    Convert matrices to graphs
    <u>Step 3:</u> Reading
    What are adjacency lists?
    <u>Step 4:</u> Exercises
    Convert graphs to lists
    Convert lists to graphs
    <u>Step 5:</u> Exercises
    Compare lists to matrices
    Which to use and when
<u>Project: Day 2 of Graphs</u>
    Graphs – contd.
<u>Activity:</u> Traversal of Graphs
    <u>Step 1:</u> Reading
    Depth-First Search
    <u>Step 2:</u> Exercises
    Perform DFS on various graphs
    Given DFS sequence, give graph
    <u>Step 3:</u> Reading
    Breadth-First Search
    <u>Step 4:</u> Exercises
    Perform BFS on various graphs
    Given BFS sequence, give graph
    <u>Step 5:</u> Exercises
    Compare the two
    Which to use and when
<u>Activity:</u> Application – Modeling a

Graph of Berkeley

<u>Step 1</u>: Exercise
Convert words to pictures
<u>Step 2</u>: Exercise
Represent picture as a list/ matrix
<u>Step 3</u>: Exercises
Perform DFS on the graph
Perform BFS on the graph

An identical "Project – Activity – Step" format can be followed for each of the topics of instruction, interspersed with activities for the students to do and learn from as they proceed.

Discussion

The WISE layout of activities and learning material proves more effective in teaching the students because of the change in emphasis of listening alone vs. doing. As the students learn a new topic, WISE activities keep them busy until they are at ease with the new knowledge that they have gained. This easily compares favorably to the classroom model where students put in hours of struggle to do homework and projects when their foundation of background knowledge is too weak to allow for the same level of comfort. Learning by listening cannot compare in effectiveness to the hands-on experience they get in lab.

The transition from the traditional model of teaching to the WISE model is not an easy one. To make it completely hassle-free for the students we have to ensure that the activities match up suitably to the instruction of the Data Structures course, given all the differences in this and CS 3 listed above. The team of WISE developers is working closely with the curriculum design group to adapt WISE to this course in particular and to build tools that will offer students the optimal learning environment for CS 61B.

Our plans for the immediate future involve the completion of this project – in the form of a WISE course offering of CS 61B – followed by subsequent tailoring of WISE to other lower-division undergraduate CS courses as well. In the longer run, these courses are being designed to be taught at the upcoming UC Merced campus as part of their Engineering School curriculum for undergraduates.

Online offerings of these courses benefit not only the students attending the classes and learning in lab; this mode of instruction has considerable scope for distance learning as well. Students in community colleges and other smaller schools can be organized in sections running labs synchronously in their remote settings. In this manner, a single course offering ends up benefiting many more students than the limited classroom settings did in the past.

The accessibility and widespread use of the Internet are the primary factors that favor this educational setting. The role of human interaction cannot be dismissed, but should complement the role that the web-based environment has to play. An effective interplay between these two roles can improve the learning ability of the students considerably.

References

[1] Clancy, M., Linn, Ryan. C, M., Titterton, N., and Slotta, J. "New Roles for Students, Instructors, and Computers in a Lab-based Introductory Programming Course"