Teaching Portfolio

Spring 2016

David Burlinson
PhD Student
GAANN Computing Scholars Fellow

Computer Science Department

University of North Carolina at Charlotte

Table of Contents

Introduction

The purpose of this document as it currently stands is to present a detailed, structured picture of my experience as a teacher-in-training. As a comprehensive portfolio, it will cover all elements of my education pertaining to this training, and will outline all my endeavors to augment and expand my understanding of effective teaching strategies and practices. In addition to listing the various teaching, professional development, and service activities I have undertaken, this portfolio includes my reflections and thoughts on each step of this journey.

Teaching Philosophy

Instructors of modern computer science must strive to train computer scientists and not simply teach students how to program. In this ever-changing, increasingly multi-disciplinary domain, we must help to equip students with a variety of hard and soft skills, and inspire a holistic understanding of computers and the impact of computing technologies to produce graduates who can integrate into a diverse array of industry, research, and teaching fields. To this end, we should encourage exploration and embrace failure, build confidence in collaboration, and promote critical thinking. We can accomplish this by structuring classes, exercises, and summative and formative assessments around real-world applicability, an appreciation for abstraction, and piecing together the bigger picture.

This approach stems from evidence pointing to the value of a constructivist approach to learning. Instructors use a wide range of methods and tools to convey material, but approaches in which students spend time actively engaged with the concept, problem, or application contribute the most to their comprehension and retention of the material. At its core, all this means is that prudent instructors teach students how to think about problems and material, structure their learning process and workflow, and build intuition for understanding and implementing solutions. Throughout my teaching career, I will seek to combine this active learning approach with the aforementioned focus on inter-disciplinary, real-world applications in and out of the classroom in order to better equip burgeoning scientists and members of the information age.

Having so recently completed my undergraduate degree, I find I am situated in a prime position to empathize with a new generation of computer scientists. My own hardships are still fresh in my mind; I keenly remember my frustrations at missing the forest for the trees, struggling to map a mess of pointers and objects into the structures and algorithms that underpin so much of the discipline. But more importantly, I know the practices I engaged in to be successful and the knowledge I gathered to connect all the dots, and I am steadily learning how best to communicate my experiences, anecdotes, and insights in order to inspire and educate students in an informed way. I will incorporate and build upon modern visualizations and interactive tools for algorithms and data structures, facilitating understanding and retention at all levels of experience. I will regularly tie in individual and collaborative assignments to reinforce teamwork and effective workflow habits in the learning process. In my classroom, I will tackle meaningful computing concepts and problems with lectures, projects, and activities designed to challenge students not only to learn material, but also to engage in dialogue, exploration, and constructive criticism as a group of aspiring computer scientists.

By incorporating my domain experience and appetite for knowledge with modern pedagogical methods and models, I will prepare students to learn from my mistakes, build upon my successes, and develop the expertise to leverage computer science concepts in any facet of our increasingly connected and technology-driven society.

Teaching Experience

Overview

Teaching Assistant – ITCS 2214: Data Structures **Spring 2015**

Instructor: Dr. Kalpathi R. Subramanian
Course size: 40-50 students
Responsibilities:

- Assisted Dr. Subramanian with management of the course Moodle site and preparation and grading of homework and exams
- Held office hours to tutor students and grade interactive demos of programming assignments
- Organized TA Problem Sessions to address knowledge deficits and provide supplemental exercises and instruction for groups of students

AlgoVis Project **Fall 2015**

In lieu of a TA position in the Fall semester and in preparation for teaching ITCS 2214 in Spring 2016, Dr. Subramanian and I began devising an application to explore the use of interactive algorithm visualizations as a teaching and learning tool. We aimed to complete a few modules for basic data structures (Trees and Linked Lists) and lay groundwork for incorporating them into the respective lessons for the course.

Modules give overviews of each concept and provide links to supplemental resources, provide both general examples and examples of specific operations for users to explore and interact with, and assess users' mastery of each data structure and its operations with interactive quizzes.

Instructor – ITCS 2214: Data Structures **Spring 2016**

Course size: 47 students
Course structure: I chose to run 2214 as half lecture and half lab in order to balance practical applications and experience with theory and conceptual understanding. This decision was largely influenced by my experiences in similar courses during my undergrad career, and solidified by the fact that I was allotted a room in which every student had access to a computer.

Responsibilities:

- Created and delivered lectures and lab sessions
- Organized formative and summative assessments
- Graded all exams, and worked closely with a TA to grade homework assignments and programs
- Held office hours to help students and provide supplemental discussions
- Helped set up peer mentoring/study sessions with Dr. Payton's ITCS 4155 students
- Maintained a class Moodle page with links to course material and external resources

Reflections

In my first semester as a teaching assistant, I wanted to ensure I learned as much as I could from the instructor, the students, and the course as a whole. My undergraduate institution had classes that averaged from 3-10 students, so the professors easily handled all the responsibilities that a TA otherwise might have undertaken; 50+ students and the requisite division of responsibility was quite foreign to me at first! Unfortunately a scheduling conflict with another course meant I couldn't attend many of Dr. Subramanian's lectures, but I was able to spend time getting to know the students through my office hours, the Moodle discussion forum, and via email. It took me some time to get used to grading the volume of assessment material, and I had to spend considerable time learning how to assign reasonable and fair grades for the various projects, homework assignments, and tests. After experimenting with a few methods, I settled on an approach that worked well for me. For tests and homework assignments, I generally skimmed through the stack to get a feel for the types of responses students gave, determined how I wanted to break down the points for every question, and then proceeded through all the assignments grading one question at a time for every student. This approach allowed me to keep the context of each particular question in mind for all students, yielding a balanced assessment more quickly than if I had to context switch from question to question for each assignment. I tried to provide reasonable feedback each time I gave less than full marks for any question, but due to time constraints I often condensed common mistakes and suggestions into one email which I would send out via the Moodle discussion forum after every graded assignment. This tactic seemed to work quite well, as it was often these general issues that students would ask me to address during the TA problem sessions I held.

Only a year passed between my TA position and my instructor position in ITCS 2214. The former was good preparation for many elements of the latter, but bearing all the responsibilities of the course and students' success was a new and challenging experience. I spent a great deal of time planning my learning objectives and outcomes, so I had a general picture of how I wanted the semester to play out. I drew fairly heavily from my own experiences learning data structures and also from Dr. Subramanian's advice teaching the same course a number of times in recent years. In anticipation of each week of class, I spent time coalescing information from multiple sources to present each topic concisely and in an approachable fashion given the diverse range of student experiences. For organizing programming assignments and exams I relied on examining similar assessment strategies from a number of professors, adapting elements from many of them based on what I had taught in a given unit. One of the biggest challenges I faced was maintaining student engagement and motivation. Only about half of the students enrolled in the course showed up regularly for class sessions, and less than two thirds of the class regularly submitted programming assignments. I gave regular feedback to students and strongly encouraged them to meet with me to work out potential issues, especially if they missed classes or assignments, but I still found a number of students unwilling to put in requisite time and effort to be fully successful. From my discussions with other faculty at UNC-Charlotte this is not an uncommon experience, however experiencing it firsthand was disillusioning and often difficult to account for when setting up projects and lectures that built upon each other. Overall, most of the students who regularly attended class and submitted assignments developed a strong foundation in data structures, and my focus on asymptotic analysis techniques and application-driven discussion was worth the effort for the students. Finally, I learned a lot about my own organizational strategies and approaches to teaching, and feel far more equipped for my next opportunity to teach a similar type of course.

Professional Development

Courses

Fall 2015

Grad 8001 - Teaching at the College or University Level

Taught by Dr. Judith Krauss, Center for Graduate Life

This course provides an invaluable opportunity to learn about a variety of pedagogical techniques and topics related to teaching at the university level. Some class sessions are devoted to developing a teaching philosophy, a syllabus, and assessment strategies, and in others we discuss a number of articles, books, and ideas about how to structure groups, classwork, and other elements of modern university courses. Overall, Dr. Krauss facilitates a congenial atmosphere in which students can explore these topics, provide constructive feedback to each other, and develop comfort and confidence in their abilities as teachers. I have benefitted greatly from the material and discussions, and strongly recommend this course to any prospective professors at UNC-Charlotte.

Seminars and Workshops

Spring 2015

Developing Your Teaching Philosophy 2/10/2015

Presented by Dr. Judith Krauss, Center for Graduate Life

Summary:

Dr. Krauss began this workshop by stressing that there is no 'right way' to build a teaching philosophy, and that I should attempt to elucidate what **I** believe should happen in a classroom and why (as it pertains to pedagogy, of course). She clarified that a good teaching philosophy can't just be a series of trite statements, but rather should truly fit who I am as an individual while also managing to encompass current trends in teaching; it should be a balance. I learned that I should be very introspective about my experiences with teachers throughout my academic career, isolate the good and bad elements, and then synthesize those ideas into a few discrete core beliefs I hold about teaching. Once I've settled upon 2 or 3 of these, I should then describe, explain, and justify these beliefs in my written philosophy, and attempt to relate them to my specific discipline where possible. Dr. Krauss also said it's important to start and finish the philosophy with something unique to ensure memorability and interest, and that a teaching philosophy probably shouldn't be longer than a page and a half since committees will likely have a huge stack to sift through.

Promoting Better Learning Through Simple, Practical Groupwork, Assessment and Feedback Strategies 3/10/2015

Presented by Dr. Tom Angelo

Summary:

In this workshop, Dr. Angelo shared a number of his recommended approaches to structuring group work and assessment in the classroom. He illustrated his ideas vividly by applying the various techniques on all of us who were in attendance so that we could understand their efficacy firsthand. One of his main points was that the first hour of the first day of a class is the most important one, because it sets the tone for the whole course. During this first hour, the lecturer must establish shared trust, language, and goals in order to make the students comfortable and ensure future engagement, because students need to feel comfortable in order to take risks. Dr. Angelo also shared information about the top three general characteristics students look for in professors: enthusiasm and passion, clarity on language, grading,

and organizational structure of the class, and fairness. For the last item, he suggested that calling on people randomly is the best way; if an instructor asks for volunteers to answer questions, only a small portion of the class will get involved, but if everyone has an equal probability of being called upon, then they are more likely to prepare and less likely to feel singled out. Before wrapping up the workshop, Dr. Angelo discussed cognitive load, saying that instructors should present the same information in multiple ways, base new topics and material on things the students already know, and don't overload students with too many new things in too short a span of time. To address this last point, he recommended a flipped classroom approach in which students gain knowledge of a particular topic outside of class and then take some kind of small assessment quiz.

Student Buy-in for Active Learning 4/3/2015

Presented by Dr. Coral Wayland

Summary:

In this workshop, Dr. Wayland discussed ways to get students engaged, willing to prepare, and involved with a flipped, active classroom. Many of the ideas she shared were ideas I've either experienced from the student perspective or heard discussed by other teachers and lecturers, and a number of them seem like common sense. In particular, there were lots of parallels between what Dr. Wayland said and what Dr. Tom Angelo discussed during the workshop back in March. Dr. Wayland pointed out that a team-based structure makes students more likely to actually prepare and show up for class; they owe it to their team, who otherwise might not get as high a grade on group quizzes. She also indicated that in her experience implementing active learning, it's common that performance goes up, but evaluation scores go down. Students are worried that they're teaching themselves, and resist that if they aren't 'bought in.' They are bound by their expectations of the lecturer being the sage on the stage, so having team/active learning is contrary to their expectations. To combat this, it's important to be transparent from the outset about how course will be different and, more importantly, why. To this end, Dr. Wayland suggested leading students to the conclusion that active learning is a good idea. Her suggestions were to ask groups to discuss why they're in college, share surveys of what employers want from employees, then have the students/teams discuss how the employer's needs apply to their intentions for their education. Then, to further reinforce the pedagogical paradigm, ask them to discuss something they're good at and how they got good at it. When they say that it took effort and repetition, the same idea can be tied back in to the classroom; it makes more sense to spend class time learning to be proficient with some particular knowledge rather than simply absorbing a lecture. Doing all this in the first class session really can set the tone for the semester, especially if the students feel they came to the conclusions naturally.

Problem-based Learning to Promote Student Engagement 4/14/2015

Presented by Dr. Tracy Rock

Summary:

The main purpose of this workshop was to discuss what problem-based learning is and how to structure a course around it. Dr. Rock talked about how critical student involvement and buy-in are, topics that Dr. Angelo and Dr. Wayland touched on in previous workshops I attended. She defined problem-based learning as a subset of project-based learning, pointing out that the former generally tends to be smaller in scope, which fits nicely for something like a Data Structures course with multiple important but fundamentally different segments of information. With regard to actually creating projects in this type of course, Dr. Rock stressed the importance of tying things back to real-world standards and collaborative work. She discussed backward design, in which long-term goals are identified, and decisions about assessment techniques are solidified afterward. I think there are definitely some elements of problem-based learning that could be applied to a course like Data Structures, but the main difficulty would be in ensuring the students gathered enough foundational knowledge to validate its use in the classroom, primarily since the course is currently very theory-heavy.

Fall 2015

Active Learning: Technology in the Active Learning Classroom 11/20/2015

Presented by Sam Eneman and Kurt B. Richter, Ed.D.

Summary:

This workshop gave insight into the future of 21st Century classrooms by exploring the technological capacity of the Kennedy classrooms at UNC Charlotte. I already had a first impression from when I observed Dr. Shehab's Mobile Application Development course in Spring 2015, so it was valuable to have a concept of the technologies in practice prior to hearing more details of their usage and potential. The two rooms can hold 36 and 126 students respectively, and both include a number of tables integrated with laptops, speakers, microphones, dry erase boards, and display screens, making the environment simultaneously conducive to class-wide discussions and demonstrations as well as group work. Properly harnessing all the capabilities of these classrooms clearly takes significant time and effort, a point Dr. Shehab was quick to make when I spoke with him prior to observing his class. Despite the fact that I will be teaching Data Structures in Woodward in Spring 2016, I am tailoring my lectures and assignments toward potentially using the Kennedy classroom in the future. To that end, I intend to roughly split the course meetings between lectures and labs, giving students more opportunities to write code and make progress on programming assignments in an environment where they can collaborate and receive immediate feedback from me and teaching assistants. I hope this will smooth the eventual transition to technology-integrated classrooms and reduce the complexity of adapting to the features they offer.

Master Teacher Observations

Spring 2015

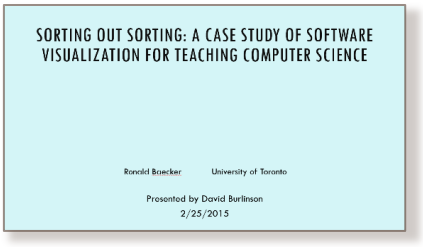
Dr. Mohamed Shehab: Mobile Application Development 4/9/2015

I met with Dr. Shehab a week prior to attending one of his class sessions, and greatly enjoyed hearing his perspective on problem-based learning for computer science courses. He helped solidify my knowledge that flipped classroom and active learning styles are very time-intensive, but his passion for these methods was certainly inspiring. It was interesting to see his approach as a teacher; his style was very laid-back and much more focused on facilitating students' involvement in the programming assignment than on actually lecturing during the 3-hour class session. From my interactions with some of the students, it's clear that they appreciated his style too. He had a good rapport with them, probably due to his easygoing and dynamic involvement, and despite the class being quite large there didn't seem to be any problems with attendance or behavior. All the students I spoke with agreed that the pedagogical approach forced them to spend time between classes learning about the week's topics and discussing ideas as a group, and more importantly they all seemed to recognize the value in this style rather than being frustrated by it. I hope to incorporate some of Dr. Shehab's ideas into the Data Structures course next time I TA or lecture for it. His methods for tying real-world applicability and active learning techniques into the discussion of each structure and algorithm seem very valuable indeed for facilitating student involvement and engagement.

Paper Presentations

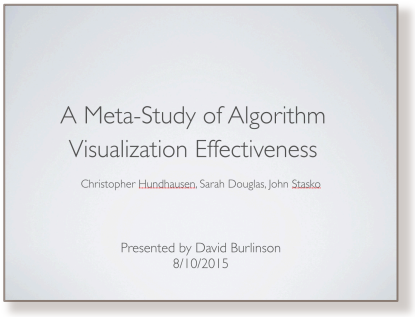
Overview

Spring 2015



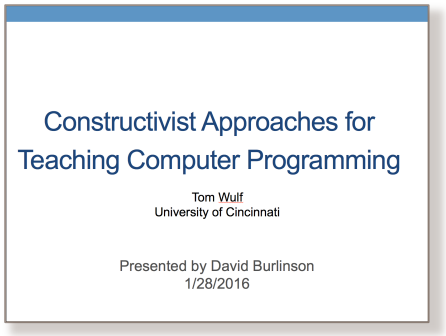
"Sorting Out Sorting: A Case Study of Software Visualization for Teaching Computer Science", by Ronald Baecker, Software Visualization: Programming as a Multimedia Experience, MIT Press, 1998

Fall 2015



Hundhausen, Christopher D., Sarah A. Douglas, and John T. Stasko. "A meta-study of algorithm visualization effectiveness." *Journal of Visual Languages & Computing* 13.3 (2002): 259-290.

Spring 2015



Tom Wulf. 2005. Constructivist approaches for teaching computer programming. In Proceedings of the 6th conference on Information technology education (SIGITE '05). ACM, New York, NY, USA, 245-248.

Reflections

I chose these papers largely due to my interest in visualization and its application in the computer science classroom. I'm quite a visual learner, so I definitely see the value in augmenting instruction with intuitive animations and interactive tools, especially when the domain is relatively abstract to grasp for undergraduates just getting their feet wet with data structures. One such tool, which I brought up during the first presentation, is a web site from the University of San Francisco's Computer Science department showing a series of data structure and algorithm visualizations (http://www.cs.usfca.edu/~galles/visualization/Algorithms.html). Both Dr. Subramanian and I used this site to supplement the lectures given during ITCS 2214 in Spring 2015. Students responded very well to the recursion and tree examples in particular, especially those who were having a hard time grasping what exactly the algorithms and code were doing for these concepts. This site was also our motivation for the AlgoVis tool we began work on in the Fall 2015 semester, and we have been in communication with the professor who created the visualizations as our current work builds upon his examples.
The third paper I presented was a resource I relied upon when building content for my section of 2214 in Spring 2016. I appreciated Dr. Wulf's discussion of practical endeavors for building student competence and certainly found his points about requiring more work from the instructor and the importance of student buy-in to hold true in my experience.

Community Service

Overview

Citizen Schools – Video Game Design

Martin Luther King Jr. Middle School 9/17/2015 – Present

Volunteer Research Assistant

BRIDGES Project 1/2015 - Present

Hour of Code

Reedy Creek Elementary School 3/31/2015

Great Falls Middle School 5/13/2015

Reflections

Hour of Code

Participating in the Hour of Code program was a fantastic opportunity to teach young students about the fundamentals of computer science. I didn't attend the training event held by Tata Consulting Services, but they were happy to allow me to complete the training by myself and attend their outreach event with Reedy Creek Elementary School. It was very rewarding to see the entire classroom thoroughly engaged and utilizing logic and problem-solving skills, so I helped set up another Hour of Code event with some classes from Great Falls Middle School during a trip they took to Winthrop University. Many of the students involved

were from poor socio-economic backgrounds, and this may have been their first glimpse into the world of programming, so it was wonderful to help foster their interest in computing technology as a potential area of study and work.

BRIDGES Project

My involvement with the BRIDGES project tied directly in with my position as a TA for Dr. Subramanian's Data Structures course, which used BRIDGES for all the programming assignments. The tool allowed students to visualize the data structures they were creating to facilitate understanding of the material and to foster retention and engagement overall. It was a useful grading tool as well; a portion of the credit for each programming assignment was given based on the visualizations the students created, so I was able to cut out a significant amount of time I would otherwise have spent poring over code from 50+ students.

My role in the project saw me working on the server side of the project, updating and modifying the visualizations according to the requirements Dr. Subramanian had for his assignments. I learned a lot about Express, Node.js, and Jade, the tools underpinning the BRIDGES server, and d3, the JavaScript library used to build the visualizations. I also gained insight into the process of developing software for use in the classroom and eventual distribution to other lecturers and universities.

In Fall 2015, I have been working with a group of seniors from the Software Design course to oversee their work adding additional functionality for the BRIDGES server. They will implement dynamic animations to display transitions in the states of the visualizations students create with the software.

In Spring 2016, I have continued in an advisory role with the BRIDGES project. I have helped oversee student work on both the client and server side of the application, and have made a number of modifications and updates myself.

Citizen Schools – Video Game Design

Due to my positive experience with both iterations of the Hour of Code program I was involved with in Spring 2015, I decided to seek out similar service opportunities at local schools. I spoke with some members of the UNC-Charlotte chapter of STARS and a local coordinator for Citizen Schools, and decided to join the group using Scratch to teach Video Game Design at Martin Luther King Jr. Middle School. It's a ten-week after-school program during which students learn fundamental elements of computer programming with Scratch, an intuitive drag-and-drop web application. At the end of the ten weeks, students are given the opportunity to showcase their newly acquired knowledge and their unique video games to their friends, family, and community at Citizen Schools' 'WOW!' event.

It's difficult to keep an entire room of students engaged, especially after a long day of school, but it's rewarding to watch the majority of them excitedly exploring concepts they wouldn't otherwise be exposed to in their classrooms and lives. While students don't write any actual code with Scratch, they are given a rich set of features including sprites, loops, conditional branching, and event handling with which to build their games. I'm often surprised with how intuitive some students find these concepts! We spend the first five weeks teaching the students a variety of skills related to game design, from using key input to create and move sprites around the screen to implementing games such as 'guess the number.' The second half of the program is devoted to helping students understand how to create their own games. I've really enjoyed the experience so far, and I'm looking forward to seeing what the students can come up with before the WOW! Event.

Teaching Research and Projects

Overview

Fall 2015

Data Structures Application

This semester I began work on an application for teaching and studying data structures. My goal is to harness modern web technologies to present structured information and engage students with interactive visualizations, and incorporate a number of feedback mechanisms to gamify the learning activities, incentivize repetition and retention, and relay usage statistics to instructors. I am currently working on the back-end structure of the application, and hope to have an entire module on Tree structures ready for use in the Spring 2016 semester. In the longer term, if initial adoption of this application is positive and provides actionable insight, I intend to perform user studies on the efficacy and impact of interactive data structure visualizations as educational resources and learning tools.

Teaching Materials

Syllabus

Data Structures: ITCS 2214 - 002

Spring 2016 Tuesday/Thursday 11:00-12:15 Woodward 140

Instructor:
David Burlinson
dburlins@uncc.edu
www.uncc.edu/~dburlins

Office Hours:
VisCenter Lab | Woodward 437
Tues: 12:30pm – 2:00pm
Wed: 1:00pm – 3:00pm
(Or by appointment)

Grader:
Christina Johnson
cjohn342@uncc.edu

Office Hours:
Woodward Fishbowl
Mon: 12:00pm – 1:30pm
Thurs: 12:00pm – 1:30pm
(Or by appointment)

Course Description

In this course, we will study the theory and implementation of abstract data types (ADTs) including stacks, queues, and both general purpose and specialized trees and graphs. A programming emphasis is on the use of an object-oriented language to implement algorithms related to the various data structures studied including creation, searching, and traversal of ADTs. Time and space complexity of these structures and their related algorithms will be a topic of focus.

This course will feature alternating lectures and lab sessions in order to maximize conceptual and applied understanding of the topics covered.

Materials

Textbook

- Clifford Shaffer, *A Practical Introduction to Data Structures and Algorithm Analysis*, Edition 3.2, Dover Publications.
  - [http://people.cs.vt.edu/~shaffer/Book/JAVA3elatest.pdf]

Supplemental resources

- Visualization Resource
  - [http://www.cs.usfca.edu/~galles/visualization/Algorithms.html]
- BRIDGES
  - [http://bridgesuncc.github.io/]
  - [http://bridges-cs.herokuapp.com/]

Some of the projects in the course will use a new software package, "BRIDGES"; BRIDGES allows real-world datasets to be easily accessible for use in course projects. BRIDGES is under development and currently has the means to access datasets via Twitter and Rotten Tomatoes (Movie Database). We will use BRIDGES in several projects in the course. BRIDGES has Java and C++ support.

Student Goals

Students should be self-motivated and must come to class ready to learn. Active participation in class discussions and activities is strongly encouraged. If you need to miss a class, you must notify the instructor before the class session, and are responsible for catching up on whatever material is missed. The Moodle discussion forum is a good place to ask other students about what occurred in class.

As a student in ITCS 2214, you will:

1. Hone your ability to write and document programs according to specifications
2. Build comfort working in small groups to discuss course material and complete in-class assignments
3. Master conceptual understanding of fundamental data structures: Linked List, Queue, Stack, Tree, Hash Table, Heap, Graph
4. Develop an understanding of space and time complexity for data structures and their algorithms
5. Understand various real-world applications and tradeoffs when implementing data structures to solve problems

Communication

In order to communicate with the instructor, please visit during office hours, email, or schedule an appointment in person or via email. I will do my best to respond in a timely fashion.

If you have any questions about course material, assignments, projects, etc., please feel free to ask during or after class, or post on the class' Moodle discussion forum if you feel the question might benefit the other students.

You may use the Moodle discussion forum to discuss project/homework strategies relating to the work in the course. However, no solutions of any sort should be posted in the forum, as the goal is to ensure each student is capable of performing the work in the course as independently as possible at this early stage of your computer science major.

Assignments and Assessments

Homework – Homework involves written and online assignments on course material. Data structures, operations on these structures, and their time and space complexity will be covered.

Class Participation – This includes attendance, involvement in class discussions, and participation in lab activities.

Programming Projects – Projects will demonstrate mastery of particular topics and data structures. Each project will contain multiple graded sections with increasing difficulty and point value. Weekly lab sessions will provide students an opportunity to work on early project stages.

Midterms – Two midterm exams will be given during the semester. These exams will cover various data structures, operations on these structures, time and space complexity of these operations, and analysis of their application to real-world problems.

Final Exam – A cumulative final exam will be given on the assigned day at the end of the semester. The final will be very similar in structure to the midterm examinations, but will involve more application problems.

Grades

Homework	10%	A: 90% - 100%
Class Participation	5%	B: 80% - 90%
Programming Projects (3-4)	40%	C: 70% - 80%
Midterms (2)	30% (15% each)	D: 60% - 70%
Final Exam (cumulative)	15%	F: < 60%
	100%	

Program Structure and Grading

- All programs must be implemented in Java or C++
- All programs must be well documented to receive full credit
- Each assignment will clearly specify whether or not group work is permitted
- **Late credit**, if permitted, will be clearly specified in the project description
- Program submission guidelines:
  - Java submissions should not be packaged.
  - Each Java file should have a single class; the class name and file name should be the same.
  - Do not zip the files or put them in a separate folder. Upload all files individually into Moodle.
  - C++ programs should follow similar guidelines

University Policies

All students are required to read and abide by The Code of Student Academic Integrity. Violations of The Code of Student Academic Integrity, including plagiarism, will result in disciplinary action as provided in the Code. Definitions and examples of plagiarism are set forth in the Code, which can be found at the following link: http://legal.uncc.edu/policies/up-407

Disability Accommodations

UNC Charlotte is committed to access to education. If you have a disability and need academic accommodations, please provide a letter of accommodation from Disability Services early in the semester. For more information on accommodations, please contact the Office of Disability Services at 704-687-0040 or visit their office in Fretwell 230.

Schedule

Date	Topic	Format, Deliverables
Jan 12th	Syllabus, Pre-test, Background	Lecture, pre-test
Jan 14th	Programming Review (0.0)	Lab, **Program 0.0 due (Saturday Jan 16th)**
Jan 19th	Why Data Structures? Common Data Structures	Lecture
Jan 21st	Programming Review (0.1)	Lab, **Program 0.1 due (Saturday Jan 23rd)**
Jan 26th	Math Background, Algorithm Analysis	Lecture,
Jan 28th	Math Background, Algorithm Analysis	Lecture, Lab
Feb 2nd	Recurrence Relations and Recursion	Lecture
Feb 4th	Recurrence Relations and Recursion	Lecture, Lab
Feb 9th	Lists, generics	Lecture,
Feb 11th	List Project (1.1)	Lab, **homework 1 due**
Feb 16th	Lists, Stacks	Lecture, **Program 1.0 due**
Feb 18th	Bridges, List Project (1.2)	Lab
Feb 23rd	Stacks, Queues	Lecture,
Feb 25th	Stacks, Queues	Lab, **Program 1.1 due**
Mar 1st	Lists, Stacks, Queues – discussion and comparison	Lecture
Mar 3rd	**Exam 1**	**Midterm**
Mar 8th	Spring Break	
Mar 10th		
Mar 15th	Binary Trees	Lecture
Mar 17th	Tree project (2.1)	Lab, **Program 1.2 due**
Mar 22nd	Binary Search, Balanced Search Trees	Lecture,
Mar 24th	Tree project (2.2)	Lab, **Program 2.0 due**
Mar 29th	Heaps	Lecture,
Mar 31st	Tree activities and comparison	Lab, **Program 2.1 due**
Apr 5th	Hash Tables	Lecture
Apr 7th	**Exam 2**	**Midterm, Program 2.2 due (Saturday April 9th)**
Apr 12th	Graphs	Lecture
Apr 14th	Graph Project (3.1)	Lab, **Program 3.0 due**
Apr 19th	Graph Applications	Lecture, **Homework 2 due**
Apr 21st	Bridges, Graph Project (3.2)	Lab, **Program 3.1 due (Saturday Apr 23rd)**
Apr 26th	Advanced Topics	Lecture
Apr 28th	Advanced Topics	Lab
May 3rd	Review	Review, **Program 3.2 due**
May 10th	**Final Exam**	

This document and its contents are subject to the instructor's changes

Lecture Notes

Lecture Notes from April 7th
Balanced Trees and Heaps

These notes are based on the material we discussed in class. If you want further information or need clarification, please read the textbook or check out some resources online.

Tree review discussion

- What are the differences between Trees, Binary Trees, and Binary Search Trees?
- What does the *insert* method look like for a Binary Search tree?
- What is a *path* in a tree?

Balanced trees

What is the complexity of insert/search/remove in a Binary Tree structure, and why?
On average, the complexity for these tree operations is O(log(n)). We begin at the root of the tree and make a decision at each level of the tree until we find the location to insert or the item we're looking for. The number of levels in the tree depends on the order in which values are inserted into the tree.

What's the worst-case scenario and how does it impact search? In other words, how many levels can a tree have after N insertions in the worst case?
*If all N elements are already in sorted order when we read them into the tree, we build a **degenerate** tree with N levels. Consider inserting values 0 ... 10 into a tree, and draw the resultant tree.*

There are a number of flavors of trees that are designed to maintain balance. AVL, Red-Black, and Splay trees are some common examples. For each of these, there is a space/time tradeoff. We maintain more variables (such as balance factor) for every tree node and write slightly more complex methods (such as performing rotations if the balance factor is not -1, 0, or 1) in order to guarantee a maximum height of log(n) for our trees. If we can guarantee our tree has at most log(n) levels, we know our insert/search/delete operations have to perform at most log(n) comparisons.

AVL Trees:

Each node maintains a **balance factor**: height(right) – height(left). For an AVL tree to be balanced, the balance factor must be -1, 0, or 1.

Any balance factor larger than 1 or less than -1 requires one or two **rotations** to reorganize the particular subtree and maintain reasonable balance factors overall in the tree.
Each rotation involves a constant number of steps: we modify child pointers for two nodes. Therefore these rotations operate in O(1) time regardless of the size of the tree.

There are two cases where rotations may be required: insertion and deletion from the AVL tree. These are the two operations that modify the structure of the tree, and might throw off the balance factor somewhere within the tree.

Insertion:
Each return from recursive insertion checks the balance factor at the current level. If the balance factor has become unreasonable, then one of four rotations must be performed: left-left, right-right, left-right, or right-left. The following link describes these rotations with good detail, and also provides a few links for further reading.
http://www.cise.ufl.edu/~nemo/cop3530/AVL-Tree-Rotations.pdf

Overall, AVL trees guarantee a height of log(n) for the tree regardless of the number of nodes added.

Binary Heap

A heap is a complete binary tree satisfying the min- or max-heap property.

Min-heap property: the value of each parent is larger than its child. The root is smallest.
Max-heap property: the value of each parent is smaller than its child. The root is largest.

This is useful when your application continually need the highest priority item, such as in a hospital waiting room, or in thread management in an OS.

Heaps maintain partial order rather than total order over the nodes. Binary Search Trees are totally ordered.

Heaps are always *complete* trees. Therefore the height is minimal: log(n) levels.

In general, Trees can be implemented in arrays or with pointers. Our BST implementation used pointers to the left and right children at every node.

To implement a tree in an array, we place the root at index 1. For any given node at position *k*, its left child is at position k*2 and its right child is at position (k*2) + 1. A node's parent can be found at position k / 2 (integer division ignores the fractional portion of this division). You can draw a simple tree and its related array positions to convince yourself of this relationship.

Since heaps are *complete* trees, they are good candidates for array implementation – every level (except maybe the last level) is full, so the array will always have contiguous numbers without any gaps. Draw a non-complete tree and its related array positions to see this relationship.

Heaps support two primary operations: insert and removeMin (or removeMax, if max heap).

Insert:
Start by considering the first open position in the array, then **percolate up** to restore the heap property.
Continually compare the parent (k/2) to the new value. If the parent is larger than the new value, move the parent to the current position and repeat from the parent position. As soon as the next parent is smaller than the new value, insert the new value at that final position. Percolating up will restore the min heap property.

```
public void insert(Comparable x)
{
        //Insert a new item to the end of the array
        int pos = ++size;

        //Percolate up
        for(; pos > 1 && x.compareTo(heap[pos/2]) < 0; pos = pos/2 )
                heap[pos] = heap[pos/2];

        heap[pos] = x;
}
```

What's the worst case? O(log(n)), since we make one comparison per level and the heap is a complete tree.

RemoveMin:
Store the root of the tree; this is the smallest value in the heap.
To restore the heap property, move the last element to front of array, and then **percolate down.** This is almost exactly the same as percolating up for insertion, and also costs O(log(n)) for the same reasons.

Priority Queues
Heaps can be used to implement a **priority queue**. The uses for priority queues are numerous in computer science; A* for artificial intelligence/robotics/pathfinding, Graph-traversal algorithms, OS implementation for threading and load balancing, time-activated events, prioritizing a list of tasks, etc. The following link is a good resource on priority queues and heap implementations:
http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html

Heaps can also be used to implement a sorting algorithm aptly named **Heapsort**. The algorithm is as follows:
read a sequence of values into a heap,
continually remove min (and reheapify) to obtain an ordered sequence

Heapsort costs O(n log(n)) overall, as both steps perform log(n) operations for n values.

Programming Assignment

ITCS 2214-002 Program 2.1 – Binary Search Trees Spring 2016

Due Thursday April 21st, 2016 at 11:55PM

This assignment is the second and final portion of the Tree project for this course. The primary goal is to use Bridges to read a sequence of Earthquake Tweets from the USGS Earthquake Twitter feed, add some more traversal methods to those already implemented in Program 2.0, and then visualize the results.

Overview

The primary goals of this program are as follows:
(1) Add a few more methods to the BST implementation from Program 2.0
(2) Use the skeleton code to read a list of Tweets and store them in a BST

The **Driver** will create an instance of the BST class, read a sequence of Earthquake tweets (with attributes including magnitude, location, date) from Bridges and store them in the Tree with the earthquake magnitude as the search key, and allow user-specified commands to perform various operations on the tree.

Note: this assignment requires you to use a modified JAR file supporting the EarthquakeUSGS Tweet class. A link to the JAR will be provided on this assignment's page on Moodle.

Tasks

Copy your package from program 2.0 and **download** the new program2_1_Skeleton driver from Moodle

Add the new Bridges JAR file to the build path for this project. A link to the JAR will be provided on this assignment's page on Moodle.

Documentation for Bridges classes can be found at the following link: http://bridgesuncc.github.io/doc/java-api/current/

BridgesBST –

Modify your implementation from Program 2.0
When you instantiate your Tree, it will be parameterized to hold <Double, EarthquakeUSGS>; the *key* is the magnitude of the Earthquake, and the *value* is the EarthquakeUSGS object containing all attributes of each quake. These objects will be imported through Bridges. See the skeleton code for further details.

- Add methods to find and highlight the **min** and **max** nodes
- Add a method to **reset** all nodes in the tree to a particular color and/or opacity
- Add a method to **setLabels**, traversing the entire tree and setting each node's label to show location and date of the earthquake. *Note that the value at each node contains EarthquakeUSGS objects, and each has getLocation and getDate methods.*
- Add a method to **findLocation**, traversing the entire tree and highlighting all nodes whose locations contain a user-specified string location. *For example, if a user enters 'Alaska,' then all nodes with 'Alaska' in the location string (ignore case) should be highlighted and all other nodes should be ignored. You can use opacity to make the highlighted nodes prominent.*
- Add a method to **highlightRange**. Allow the user to specify the min and max magnitudes then highlight all nodes falling within that range of magnitudes.

Driver –

Download the skeleton driver from Moodle and familiarize yourself with any code it contains. The comments will provide a structure for you to follow.

- Initialize the Bridges object with this assignment number, your username, and your API key. (See the Bridges template on Moodle for details)
- Observe the method calls building a List of Earthquake Tweets from Bridges, write a small loop to test the results if you like
- Create an instance of your Tree, and parameterize it to store EarthquakeUSGS objects (the same objects stored in the List from Bridges). Use the Earthquake's magnitude as the Tree's search key. You should also familiarize yourself with the methods each EarthquakeUSGS object contains
- Read every EarthquakeUSGS object from the List into your Tree, then use *Bridges.setDataStructure* to point Bridges to your data structure
- Create a menu in the console allowing users to specify operations and arguments. Each option should correspond to one of the new Tree methods you've added, and should visualize the tree after performing its operation.

Deliverables –

Your program should generate a number of visualizations (one for each user-specified command). Each successive visualization will appear as a sub-assignment on the Bridges website.

Scoring Rubric

Driver: 40 points

- Up to 5 points for appropriate documentation and comments
- Up to 15 points for reading the USGS Earthquake Tweets into your BST
- Up to 20 points for creating a simple menu allowing users to specify operations and arguments

BridgesBST: 45 points

- Up to 5 points for appropriate documentation and comments
- Up to 5 points for implementing the **min** and **max** methods
- Up to 5 points for implementing the **reset** method
- Up to 10 points for implementing the **setLabel** method
- Up to 10 points for implementing the **findLocation** method
- Up to 10 points for implementing the **highlightRange** method

Visualization: 15 points

- Up to 15 points for visualizing the Tree after each operation. *Min, max, reset, setLabels, findLocation,* and *highlightRange* should all generate new visualizations.

Total points available: 100

This will be graded as a programming assignment

Late programs will lose 10% of the available points per day.

Teaching Review

Instructional Performance Appraisal Form

Evaluator Angelina Tzacheva and James Frazier Instructor David Burlinson

Date 03/29/16 Time 11:00 am – 12:15 pm Class ITCS-2214-002 No. Students Present 26

INSTRUCTIONS

- The evaluator is to assess the instructor's performance with respect to the major functions of teaching listed below
- The evaluator must add pertinent comments at the end of each major function for which an assessment of "Sometimes" or "Rarely" is given.
- The instructor is provided an opportunity to react to the evaluator's ratings and comments.
- The evaluator and the instructor must discuss the results of the appraisal and any recommended actions pertinent to it.
- The instructor and the evaluator must sign the form in the assigned spaces.
- The form must be filed in the instructor's personnel folder.

Check the appropriate boxes in categories 1 – 5 on this form.

1. Classroom Management

	Often	Some.	Rarely	N/A	Yes	No
1. Instructor makes good use of available time for teaching and keeps students on task.					x	
2. Instructor stops inappropriate behavior promptly and consistently, yet maintains the dignity of the student.				x		
3. Instructor's expectations are clearly explained when giving assignments and other directives.	x					

Comments: Class started on time. Most students present at start of class. Had some difficulty in starting topic and maintaining class engagement. In a few minutes, lecture flow was better. Classroom environment well managed.

2. Instructional Monitoring

	Often	Some.	Rarely	N/A	Yes	No
1. Instructor poses questions clearly.	x					
2. Instructor uses student responses to adjust teaching as necessary.	x					

Comments: Instructor addressed student questions properly. Suggestion: do more 'practical' applications before discussing the underlying theory, in this way questions are more effective. Example: assign a simple problem / exercise to engage the class. Walk around and gauge the level of understanding. Use this insight to ask questions pertaining to your observation of the students.

3. Instructional Presentation

	Often	Some.	Rarely	N/A	Yes	No
1. Instructor links instructional activities to prior learning.		x				
2. Students appear to comprehend what instructor is saying.	x					
3. Instructor provides relevant examples and demonstrations to illustrate concepts.	x					
4. Instructor asks appropriate levels of questions that students handle with a high rate of success.	x					
5. Instructor conducts the lesson or instructional activity at an appropriate pace, slowing presentations when necessary for student understanding but avoiding unnecessary slowdowns.	x					
6. Instructor summarizes key points.	x					
7. Instructor presents content in a logical manner using smooth transitions from one topic to another.	x					
8. Instructor encourages creativity and critical thinking in problem-solving.		x				

Comments: Instructor engaged students through asking questions. Students participated, and answered questions. Used Power Point presentation. Mentioned real-life examples, and applications to motivate students. Used white board to illustrate tree concepts. Engaged students through in-class activity, and working with peers. Some students were distracted. Suggestion: observe board visibility from student's point of view (laptop blocked part of board).

Instructional Feedback

	Often	Some.	Rarely	N/A	Yes	No
1. Instructor provides sustaining feedback after an incorrect response by probing, rephrasing the question, giving a clue, or allowing more time.	x					
2. Instructor treats all students in a fair and equitable manner.	x					

Comments: Instructor handled students' questions well. Instructor was able to handle incorrect responses properly. Provided feedback to in-class activity. Suggestion: for better response - do repeat student's question for others.

5. Post-Observation Interview with Students

	Often	Some.	Rarely	N/A	Yes	No
1. Instructor has established a set of procedures that govern the handling of routine administrative matters.					100%	
2. Instructor returns graded material in a timely manner.	69%	27%	4%			
3. Instructor regularly provides useful feedback on out-of-class work.	62%	35%	3%			
4. Instructor understands overall concepts.	100%					
5. Instructor creates learning activities that make subject matter understandable.	50%	42%	8%			
6. Class assignments are reasonable and encourage learning.	58%	35%	7%			
7. Out-of-class assignments are clearly set forth.	65%	31%	4%			
8. Students know what is expected of them.	69%	23%	4%	4%		
9. The instructor uses strategies that encourage critical thinking and problem solving.	69%	31%				
10. Instructor uses in-class activities during the semester.	62%	38%				
11. Test questions are appropriate for the material covered in class.	81%	15%	4%			
12. Instructor is readily available to students outside of class.	88%	8%	4%			
14. The class environment encourages students to ask questions.	77%	19%	4%			

Overall Performance (check one):

Superior ☐ Above Average ☒ Average ☐ Below Average ☐ Unsatisfactory ☐

Evaluator's Summary Comments: Lecture well prepared. Instructor engaged students through questions and in-class activity. Classroom environment conducive to learning. Student Comments: "Awesome job going over midterm." "Good on email." "Appreciate extra office hours." "He encourages students." Complained about the use of Bridges, felt the TA should be better trained in it in order to help students.

Instructor's Reactions to Evaluation:

Evaluator's Signature 4/27/16 Date

Instructor's Signature 4/27/16 Date

Instructor's signature indicates only that the written evaluation has been discussed.

Appendix

Contact Information

Email: dburlins@uncc.edu

Phone: (843) 465-6283

Lab: VisCenter lab, Woodward, UNC Charlotte