Computer Science Portfolio Reflections

Charles Bailey

04/25/2015

CMPT 390

I believe the following artifact satisfies the Computer Science Learning Goal of Developing Effective Problem Solving Skills:

Minimum Spanning Tree – the python files are DisjointSet.py and mst.py

The following program is an implementation of Prims and Kruskals algorithms for Minimum Spanning Trees on Undirected Graphs. This was the only assignment from CMPT 306 that involved programming, as other programming assignments in the class were labs or included with the Midterm and Final tests. Writing this program involved the Computer Science learning goal of developing effective problem solving skills. The disjoint Set program was already written for us and the students were responsible for implementing the functions that implemented Prims and Kruskals.

This program involved having to pick an appropriate data structure to represent an adjacency List, the visited edges, the minimum spanning tree, and the priority queue. The adjacency list is a dictionary data structure that held the adjacent edges for the current edge. The purpose of having the adjacent edges was to add the adjacent edges to the priority queue. The visited edges is a list that keeps track of which edges have already been checked by the algorithm. The minimum spanning tree was a list that contained the edges that are included with the minimum spanning tree of the undirected graph. The priority queue was a list that can be used with the heapq functionality of python and can give a priority to items based on numerical value. One of the many mistakes I made when first using the priority queue was not having the right value listed first on the edge. I tried having a node as the first object in a tuple, but when I switched it to the cost of the edge, the priority queue started implementing the algorithm correctly. This program also fits the problem solving learning goal because I often had to look up information on how the algorithms work, and then write the code to implement the various steps of the program. While implementing Prims was easier as much of the code was written for the function already, Kruskals was harder to implement. I did not need to use much of the code that was already written, but I had to find out how Kruskals worked, and then think of a pseudo code approach to how to solve it. I also had to implement two functions from the DisjointSet program and figure out where they would be used. I often had some compilation issues as the find function needed the root of the edge, but the current function had the roots as strings. The find function would only accept integers, and therefore the root needed to be converted into an integer value. I learned from looking online that python had an ord function that could convert a string value into a Unicode value. Although the ord function worked for the roots, the priority queue still was not returning the values in the minimum spanning tree. I had to subtract 96 from the Unicode values while implementing the find function, and then use the union function to add the edge to the minimum spanning tree. My final issue was indentation with python and where to put certain code. Once the code was put in the right place, both kruskals and prims algorithms were working. The assignment taught me about what data structures are appropriate to use in python, how to implement an algorithm, as well as looking to outside sources for how to perform certain actions in python such as converting a string character to an integer.

I believe the following Artifact satisfies the Computer Science Learning Goals of becoming a more effective problem solver, as well as understanding how to implement object oriented software design:

Queens – Specifically the Queens.java and TestQueueEnsureCapacity.java files.

One of the earliest programs I had to write for a Computer Science Class was a program to determine where to place N number of queens on a chessboard so that none of them could capture each other. This was a program created in CMPT 202, where I was just getting introduced to data structures and applicable problem solving. While this program may not be as impressive as many of the other programs I have done for Upper Division Computer Science Courses, It was one of the first steps of learning how to implement an algorithm and testing it with unit tests. The algorithm needed a few methods to perform basic functions like clearing out a board, displaying the board in a command window, setting a Queen on the board, and removing a queen from the board. The hardest part of writing this code was being inexperienced with problem solving and understanding Java, since I had only taken the introductory computer science course. I had to get help to understand the importance of having methods to perform certain functions in a program, and how to apply the theoretical algorithm into an actual program.

Once the methods that did basic functions were written, I had to implement an algorithm to determine if another queen could capture a queen. To do this, I had to have a for loop check and see if there were any other queens in it. Once the method worked for columns, I had to think about how to check diagonally. This involved two for loops check right and left of the current queen chess piece. The entire method was written as a private Boolean method to return true or false. I also had to implement a method for placing a queen and use the method for determining if a queen was under attack. Once this method was completed, the main method was created to display the board and show a person if there was a solution for a board of size N. In addition to learning how to problem solve and implementing an algorithm, the program also had to pass a unit test. I had some experience with unit tests in the introductory course, but did not have a good grasp on them. As I wrote the N-Queens problem, I had to periodically run the unit tests to see if my program was working. I was glad when some of the tests were passing, and frustrated when other tests were still failing. Working with unit tests taught me that you can’t expect your program to function perfectly the first time you write it, and that it is alright to fail a few unit tests as you write your solution one step at a time. It is easy to create something and say it works when it does what you want it to, but sometimes having a unit test program can challenge you to rethink how you are solving the problem at hand. Once all the unit tests were being passed by my solution, I felt proud of my accomplishment because I had solved a challenging (at the time) problem and it passed the unit tests. When looking back on the program, I learned about persevering through implementing a tough algorithm, how to temporarily use other ways of getting the methods to work, and how you don’t need to pass all of the unit tests at once to make progress.

I believe the following artifact demonstrates the Computer Science learning goals of developing software project experience working alone and in a group setting, as well as attaining a system level understanding of a computer:

Chatroom Homework 7 – All of the java files

For our final project in Computer Networks, we had to create a chat room server, and interface to use. This was the culmination of the other theoretical and programming assignments we had done in that class. Since our server had to connect to a network, we learned about how our server interacted with a network and an operating system. When a chat room server has set up, we would always specify the port to use and connect to. The chat room also had a specific protocol that the entire class chose and we had to implement the protocol. Our class had five basic command to use, which were login in, checking other users on the server, closing out of the connection, and sending either a private or public message. This project introduced the concept of parsing, which is to separate words with spaces and have a specific word perform a specific function. While I learned about how to implement a program to connect to a network and allow a user to perform specific functions, the most important aspect of this project actually taught me about how to work in a group setting and how to create a project with other people.

My partner for this assignment was Tony Garcia. We both agreed to work together and write the chat room server, as we were allowed to work alone or with another person. While many programming assignments are written alone and many classes have labs where students work together, this was one of the few programs I wrote with another person. Tony was busy in the month before the project was due, so I had to create the server file and work on it by myself for a while. By the time Tony and I met up, I had the server created and could use all the commands needed by the protocol, as well as have the server give feedback for certain actions. Despite this, I was using putty to test out the server, and needed help creating a user interface with buttons. Tony introduced me to the development environment of Eclipse, and I fell in love with it. I had tried using Dr. Java with all the other previous assignments and this slowed down the results of debugging many of my programs. Eclipse had the advantage of being able to change one variable name and applying it to the whole program.

Tony and I had to look up many ways to create an interface, and we both took turns writing code and thinking through the algorithms of how the screen could interact with the server. We both had to be patient and give up most of our Saturday and Sunday before the project was due to make the screen work. Sacrificing time to get the project done is something anyone can learn in programming by themselves, but coordinating when to meet and discussing ideas are important qualities to have as I transition to the work environment. Working in a team of two taught me about compromising as there were many times where Tony and I had different ideas about how to implement the chat screen. When we often combined our ideas, we found great success and were happy with how we had an interface and how the buttons implemented server commands. We were both happy when we turned in the project and could simulate a chat room and run the protocol. While working together on a project can relieve the stress off of one person, it can help to have multiple perspectives on how to implement a program and have a discussion to create more efficient software.

I believe the following artifact demonstrates the computer science learning goal of Understanding the concepts and techniques of object oriented design:

Search Algorithms- FindPath1.py, and FindPath2.py

In Artificial Intelligence, one of our first major homework assignments was to implement 4 different search strategies for finding the quickest path through a maze. The four methods of searching through a maze were depth first, breadth first, uniform cost, and astar searches. This was one of the very first times where I programmed in python, as I had primarily written many programming assignments in Java or C language. I had some difficulty adjusting to a new language, as there were python specific issues I had to overcome. I searched online and asked Greg Gagne for help on many python issues. The most important challenge for me was differentiating the different search algorithms and figuring out how each one worked on the theoretical level. The breadth first algorithm worked by starting at a root node and then traversing the neighboring nodes in a level first before it moved to another level to search. The depth first algorithm revolved around starting at a root and then traversing down a branch before backtracking to another branch. While they are two different algorithms, I was surprised how in python there was only one character difference between switching from a queue to a stack. The depth first search had the line node = frontier.pop(), while the breadth first search algorithm had the line node = frontier.pop(0). If I had been programming in Java or C, I would have written different logic to implement the algorithms, and was surprised by how python simplifies using many data structures, and many less lines of code I had to write. This led me to write many programs in python whenever I could because of how simple python is and how one character changed the implementation of an algorithm.

The final two algorithms were a little harder to implement, as the uniform cost search needed to use a heuristic to find the shortest path between two nodes. The astar algorithm was similar to uniform cost search, but needed the cost of traversing a path in its calculations. I thought that I would have to write many lines of code to differentiate the two algorithms, but it turned out that one line of code was the difference between how to implement the algorithms. In the uniform cost search, I pushed the heuristic onto a priority queue, while the astar algorithm involved having the heuristic and cost pushed onto a priority queue. Once the algorithms were implemented correctly, we tested them by having each algorithm go through 6 mazes and seeing what the cost was to find the shortest path through. The results pointed to breadth first and astar strategies generally having the lowest costs, although the astar maze had astar and uniform cost search have the lowest costs. Writing the algorithms for this program gave me an opportunity to write an object oriented program and test four different algorithms to see which one had the most efficiency in finding the shortest path through a maze. While we had no unit tests, I learned that there are many ways of testing a program such as with text files or seeing how the costs of algorithms compare. The assignment also showed me that many programs can have methods achieve the same results, but that some algorithms are more efficient for certain tasks than others.

I believe the following artifact demonstrates the computer science learning objective of attaining a system level understanding of the computer:

What is Computer Architecture? – An essay for Computer Architecture

One of the more surprising assignments I have done in a computer science class was writing an essay describing the subject of what Computer Architecture is. This assignment gave me one of the few opportunities in the computer science curriculum to develop my writing and communication skills. Scott Overholser, the professor of the class, told us how important it was to be able to communicate technical details of your job to people who don’t have the same background knowledge that you do, and that being able to write technical reports is important in any kind of software development career. I had thought that we would have to implement a hardware project and write code to make the hardware perform a specific function, since many of the labs involved running a program on the Stelaris Launchpad. The essay I wrote was broken into several paragraphs discussing and reflecting on what I had learned throughout the whole course. While I have had final projects and exams to demonstrate my knowledge of previous computer science courses, I never had to write an essay about my experiences in what I had learned. This was one of the few times where I have actually had to reflect on what the class was about. With many of the finals I have take here, I often go in for two hours, complete a set of problems, and then I move on with only programs and written assignments as reminders of what I did or learned in a class. This essay gave me a chance to demonstrate a big picture view of Computer Architecture and prove that I can communicate technical ideas to people without technical backgrounds.

Computer Architecture taught me about how hardware operates a program as well as how various systems work together. In the essay, I talked about how various labs introduced components of the hardware like the processor, and how they used registers and stacks to perform various functions. After each lab, we would write a lab report where we would report our observations and reflect on what was actually happening as opposed to just answering a few questions. When I wrote the essay, I looked back through the textbook, lab reports, homework assignments, and many other sources to remember what I had learned over this semester. The essay made me realize that Computer Architecture is perhaps the only class where students get to interact with hardware and learn about how software interacts with the various components inside a computer. The essay also reflects on how having an understanding of the hardware of any computer can make a programmer write better software, as anyone can program but could write code that does not address hardware limitations or capabilities. The essay also described assembly language, which is how the computer interprets programs written in languages like C, Java, or Python. The labs often have us look at different views to see what’s happening when we step through various lines of code, and the instructions we see are often in assembly language. Writing essays is something that almost never happens in most computer science classes, and as important as it is to be able to write programs and know many theoretical aspects of software and hardware design and implementation, effective communications skills are also valuable in the real world, as many people I will work with won’t understand or care about the technical aspects of software that I am working on.