Joe Lambrecht

CISC 480- Senior Capstone

5/18/2022

Final Paper

Senior Portfolio

Over the past four years at St. Thomas, I have received more than just an education in computer science and data analytics. The University of St. Thomas takes a great deal of pride in its liberal arts education and its mission to raise scholars who “think critically, act wisely, and work skillfully, all for the common good”. In my time as St. Thomas, I have found many opportunities in my classes to advance the mission of the university while developing my own skills as a student and a programmer. In the past year alone, I’ve had classes that directly addressed issues such as diversity in education and academia, the role of religion in a Catholic university, and how sustainability could be implemented individually and systemically. But the problem that most interests me was one I encountered regularly in my work as a tutor.

I have been a statistics and computer science tutor for the past 3 years. During that time, I’ve had students of all skill levels stop into my tutoring hours to get help on introductory course material. What I heard, time and time again, was that students felt totally inexperienced when it came to programming. “I’ve never had to do anything like this before”, “I don’t understand what this even does”, and “When will I ever use something like this?” are all phrases that I have committed to memory. Introductory Statistics is a required course at the University, so these students came from all sorts of majors, but this sentiment was universal. To me, this shows a huge problem in how students are educated about computer science and technology. My peers did not have any exposure to the concept of programming and computer science before college! This drastically reduces their likelihood to pursue a career that involves programming, widening the diversity gap in computer science. It also discourages them from pursuing a higher level of technological literacy. If algorithms, artificial intelligence, and smart devices are all just black boxes and you are unable or unwilling to learn how and why they operate, it becomes much more difficult to think critically when algorithms choose the media you see, more difficult to act wisely when artificial intelligence suggests course of action for you, and more difficult to work skillfully when smart devices continue to replace jobs, one industry at a time. I strongly believe that a basic understanding of programming, statistics, and computer science is a mandatory element of a well-rounded education, and it is a serious problem that most students are not gaining this at St. Thomas.

I have encountered this problem through my time tutoring at St. Thomas, and I’ve made a few observations across the years. First, it seems that students express roughly the same level of inexperience and discomfort with programming, regardless of observed ability. I’ve had students come in with every question correctly answered, but with no confidence and little understanding of the principles behind their work. Equally, students have come in with every answer incorrect and no idea why it doesn’t work the way they expect it to. I acknowledge that this is a volunteer sample and definitely biased, but it is a common enough issue that I believe it deserves addressing anyways. My second observation is that, when given an opportunity to truly understand the material, nearly every student finds so much joy in seeing their code run successfully and *knowing why*. I’ve encountered the same enjoyment of programming in my friends and family when they solve a problem in that realm, which confirms for me that, given the chance to learn, programming is universal. Anyone can program and enjoy it!

The projects in my portfolio were all chosen for their ability to introduce people to programming. I have personally used each of these projects to explain an aspect of computer science to friends, family, and acquaintances who expressed some interest in what I do in college. I chose to include the Two Way Directed Graph project because it illustrates complex computer science principles and the function of algorithms at a very high level. It poses an easy question for anyone to understand: Can you get to work going only uphill, then only downhill? They can visualize it and answer the question, then they have to think about how they found the answer. Through this framework, I can explain brute force approaches, and then work toward efficiency and runtime to explain more complex methods. I use the pipelined architecture less to demonstrate what programmers do, and more to show how programming influences everyone’s daily life. By showing a simple simulation of how a computer works, I have explained to my tutoring students why passing by reference matters, how arrays actually work in memory, and why it is important to understand these concepts early on. I’ve also used it as an example of what you can do with a degree in computer science when talking to high school classes. While this example is rather complicated to actually explain, it sparks curiosity and, in many cases, a desire to learn more about computer science. In turn, this creates students who are eager to explore the field, and thus feel more prepared to learn and enjoy in their classes.

Of all my projects, I use the ‘Deal or No Deal’ spreadsheet the most. Everyone likes games, and the hit game show is popular enough to be recognizable and interesting to a wide audience. I’ve shared this spreadsheet with high school students, peers in other majors, and family members, and every time the conversation is very similar. They play through the game, and inevitably ask some question about how it works. I explain how Excel formulas work to calculate things, then show the equation used to calculate the offers. Then I demonstrate how the buttons work to update the different cells, before finally touching on the true programming happing in Visual Basic macros behind the scenes. More than once, people have offered suggestions on features or improvements, and I work with them to implement the change. They get to see in real time how their idea is thought out, turned into code, and then reflected in the playable product, all in a medium they are already familiar with. I truly love to use this project as a demonstration of computer science because it showcases the creative nature of coding hidden behind the rules and syntax. After four years of difficult classes and projects, I know that with patience and understanding, programming produces anything you can imagine, and I love seeing the glimmer of that realization on someone else’s face as they see their suggestion come to life.

There are plenty more projects that can be used to help expose people to the world of computer science, but I’m happy to highlight these three as some of the most impactful I’ve used. They are wonderful tools for introducing people to programming, computer science, and statistics in a low-stress and low-stakes environment, and I personally have had great success in using them to give otherwise unfamiliar people some experience in what it means to be a computer scientist. By introducing people to programming casually and making them feel comfortable with the concepts behind coding, I believe they are more willing to seek out opportunities to learn more, and certainly to feel more confident in asking questions and seeking deeper understanding of how and why things work.

After graduation, I plan to continue my outreach efforts by giving back to my high school, which just started offering its first computer science course. By sharing projects such as these with high schoolers who are seeing code for the first time, I can show what is possible down the road and begin a conversation with students where they can ask questions and learn more about what they are being taught in the classroom. After completing my own education in computer science, I wish that I had someone to answer those questions for me when I was that age, so I hope that through my efforts, the next generation of programmers feels even more proficient and confident throughout their schooling.

For some, having the opportunity to ask questions and explore programming in a casual environment may spark an interest that grows into a career. But not everyone who sees a fancy spreadsheet or a school project will decide to pursue computer science. For most people, my efforts at best result in a bit more willingness to learn about programming in the future. I consider that to be a very positive outcome- algorithms alone control media, advertising, navigation, and so many other aspects of life that it is certainly in everyone’s best interest to learn more about how they work. In a world unapologetically dependent on technology, I firmly believe that understanding computer science is one of the best ways to think critically, act wisely, and work skillfully all for the common good.