CSE 373 Blog Post
by Shoaib Laghari

Hello! I'm not really sure how to start this assignment but this is my reflection / blog post for CSE 373. I feel like I'm definitely not going to meet the word count for this assignment, but that's not why I'm writing this introductory paragraph before my introductory question. No, definitely not. I just wanted to summarize that overall, CSE 373 has been a great experience. It's definitely a unique course, with its emphasis on analysis and understanding instead of perfect code. I think I got to develop a good understanding of lots of topics that I didn't know before, and I also got to brush up on Java fundamentals and coding challenges that I hadn't seen since CSE 143. I think a valuable aspect of this class was the group work and collaboration, both during lectures and after class, and I appreciated that TAs were nearly constantly available for help. I'm going to try to keep this blog post as honest as possible, and I hope that that takes me somewhere. I'm also going to stop re-reading for grammar, so stick with me if correct grammar is what you were looking for. Thanks in advance for reading.

1. How has your understanding of CS changed?

To see how my understanding of CS has changed through this course, we can compare it with my understanding of computer science when I was taking CSE 143X my first quarter at the UW. Although I lost all of my code for 143 assignments, I remember that Stuart Reges was doing this demo in class of recursive programming where he solved the "N-Queen problem," which tries to answer the question: how can you place N queens on an N-by-N chess board such that no queen is attacking another? My thought process back then was pretty straightforward. "Here's a CS problem, now what's the answer they're looking for? Surely there's one way that everyone is doing this…"

One thing that I learned from this course that I didn't see in CSE 143 was that there are *multiple* ways to solve almost any complex problem. There are definitely popular routes, that seem the most "efficient" or "clever," but even those have their downsides, too. In the shortest paths project, for example, we saw that the real-world solution to the shortest paths problem didn't even need a graph algorithm, it used dynamic programming. This makes me think that the N-Queen problem doesn't need to use recursive programming either. I wonder if we could use DP to avoid recursively checking options that we know won't work, e.g., queens on the same row/column/diagonal as another queen. Though this seems like a recursive problem, DP may have a faster solution for larger chess boards.

Next, I learned that there are many ways to analyze or compare a program. Not only did I learn about asymptotic (runtime and worst case) analysis through this course, but I also learned how to use affordance analysis to catch details that might make clear why we should prefer one

algorithm over another. In the priority queue project, for example, we developed increasingly efficient implementations, but how effective were they for content moderation? Through affordance analysis, we saw that our entire approach to the problem disafforded the moderation of low-priority content in large scale environments. This meant that once we began to see millions of comments, we might take so much time taking down high-toxicity comments that we never resolve any of the low-toxicity (but still important to remove) comments. To solve this, we had to reconsider a different algorithm: separate, simultaneous priority queues. Similarly, for the N-Queen problem, we should think about what the goal of the code is before we decide to use recursive programming, DP, or another solution. If we just want to display solutions one-by-one, recursive programming isn't so bad, but if we want to store them, we might prefer DP.

Lastly, through this course, I learned how to critique and connect my work to the rest of the world. In the autocomplete project, for example, we explored the social impact of predictive text algorithms, how they can affirm harmful biases, be prone to cyberterrorism, and so on. Before, I would think that coding assignments were one thing, and the real world was another. Little did I see that there was much overlap between the two once you began to apply yourself. Thinking about the N-Queen problem, for example, the problem can be spun to think about how any set of abstract actors with a range of attacking power can coexist with other actors in a defined space. By breaking the problem down into this form, we can apply our previous solutions (recursive, DP, anything else) to problems in economics, geopolitics, and much more. I'm thinking economics and game theory because a lot of problems like the N-Queen problem exist there.

2. What part of the course has changed you the most?

Professionally, I am ecstatic about the fact that I can use hash tables, invariants, and asymptotic analysis in my future coding interviews. I feel like these are three concepts that I wasn't exposed to in CSE 143, and I wouldn't have been at all if I didn't take this course, and I can already tell how useful they're going to be.

Personally, the most meaningful part of this course was working on project implementations together in teams. This is pretty unorthodox because it isn't the most efficient way to "get work done," but it does a great job at showing students how to share and integrate ideas and move towards solutions in a team setting. Here, I'm most proud of directly addressing and resolving collaboration issues that my team had in the second project, priority queues. One of my teammates in that project had (and still has) a very outgoing, talkative personality, while the other was more reserved and waited to speak. The talkative partner (out of complete good intention, wanting to help the team find the best solution) would always give their opinion and my other teammate felt that they weren't being heard. They brought this up to me after class, frustrated, and I am proud to say that we talked about it as a team during our next group meeting and the first teammate had been much more understanding for the rest of the project.

3. What's next?

I think I'm going to stick with my Economics major. I've been thinking more about this, stressed that I won't meet the technical requirements for the data science and data engineering jobs that I know I'm going to apply to, but I think I would regret not pursuing my passions and chasing my dreams. My dreams, to me, are clear and simple. I want to do work in computational economics, combining computer and data science tools to solve economic problems, and I want to work on problems such as sustainability (through smart cities) and international trade (through blockchain technologies), to give two examples. It's what I'm passionate about (economics) and it's what I'm good at (computer science). I feel like I would lose a part of my identity if I gave up either of the two, and unfortunately a double major isn't possible for me.

To make up for the fact though, I'm going to use this course as a springboard for my data science career. I'm going to take CSE 414 next quarter, CSE 412/416 sometime in the future hopefully, and I'm going to use the resources provided in this course (LeetCode, Kaggle) to get extra practice when looking for internships. I also got some machine learning textbooks that I might try to read this summer.

Overall, I'm really grateful for my experience in this course. It was great getting to meet my TAs and classmates, and this course has advanced my understanding of computer science in so many ways. I feel much more comfortable entering the career that I'm going in to, and I know it wouldn't have been possible if I didn't have this class a resource. Thank you for offering and supporting non-major classes, thank you for all of your feedback throughout the quarter, and thank you for reading this blog post. I hope you have an awesome Spring break and even better Summer!

Sincerely,
Shoaib