Computer Science Education: Synthesis of the Curricula of Data Structures and Algorithms

Analysis of Algorithmic Bias and its Interaction with Society

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Mark Chitre

Fall 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Benjamin Laugelli, Department of Engineering and Society

Rosanne Vrugtman, Department of Computer Science

Introduction

The field of computer science has many different branches that have been studied for decades. The social impact of computer science has been unparalleled thus far as a result of how fast we are moving and the amount of information that becomes available so quickly in the world nowadays. Computer systems involve mainly coding, a way of interacting with the hardware of the computer to complete an intended task (Gordon, 2019). Once the computer is given a set of instructions in the form of a programming language, it is then able to interpret the characters given and do what the user wills of it. A set of instructions can be called an algorithm; this term is used heavily in computer science because it encapsulates the general idea of a process that the computer follows (Brogan, 2016). Since the field of computer science has such an impact on daily life, education about the field is very important as well as the social implications of new technologies.

As computer science becomes more impactful, the issue of how to output known information into education is important to address. There can be ways of teaching that are advantageous and other ways that are detrimental to a student's learning. The technical portion of this paper focuses on how courses in computer science (one about Data Structures, one about Algorithms) can interact to create a proper learning experience for a student in college to prepare them for interviews and the workforce. This addresses the issue of how to optimize computer science education while accounting for the current rate at which the field is moving. In addition to looking at computer science education, the social implications of algorithms and applications have to be considered as well. Students should be educated on algorithmic bias and its impact on society and vice versa. The STS portion of this paper discusses how algorithms and technologies impact social justice issues as a result of bias. The determination of whether technology

influences society or the other way around will also be explored to further this analysis. Possible consequences of leaving this problem unaddressed encompass a lot of different fields. This includes students not getting the right education; with a rehaul of the curriculum this would benefit them greatly. It is important to make sure that students can build off of previously learned information, showing them how certain building blocks are essential to CS. Leaving future algorithmic bias unaddressed has huge implications- non-inclusion of users, possible lawsuits, and more unintended consequences.

The idea that the Technical Topic and the STS Topic converge on is how information output in the field of computer science affects people and their perception of the impact of computer science. There are very different ways in which computer science education and algorithmic bias intersect, but the problem that this proposal addresses is how to optimize the delivery of information about computer science in a way that is inclusive and all-encompassing.

Technical Problem

Computer science (CS) education has been approached differently for decades and can vary greatly from institution to institution. Most of the innovation in this discipline has flourished over the last 30 years since the advent of the World Wide Web, as the world has been able to share information and knowledge at a very efficient rate (Amiri, 2019). As a result of the extensive use of computer science, the field has created many high level jobs and most of them require an ample amount of education/a degree as a bare minimum. The study of CS education has produced varied results; the order in which students take classes or the specific classes themselves can affect the quality of their education and ultimately prepare them well or poorly for future interviews and jobs (Vegas, 2020). The idea of synthesizing two CS classes could be beneficial to students based on their topics.

At the University of Virginia, there are several required classes that CS Majors have to take, each with its own purpose. Most classes are part of the sequence in which CS students start off with introductory programming then move on to doing specific tasks with the skills that they mastered as a result of the previous classes. The second class in the sequence after the introductory course is Software Development Methods (CS 2110) which teaches a combination of various subtopics in the field. CS 2110 covers Object-Oriented Programming, Data Structures, abstraction, and modern software engineering practices. The class is mostly based in a programming language called Java and while there are many goals, the focus on learning how to do different things in Java is emphasized. Another course that is required toward the end of the sequence of required courses is Algorithms (CS 4102). In this course, students delve into common CS algorithms and how to apply them to specific problems and scenarios to optimize code runtime.

The current CS curriculum contains themes of CS 2110 in CS 4102 and vice versa, but the courses are mostly distinct because of the sheer amount of information taught to students as a result of these courses. In CS 2110, a lot of the algorithms that are taught in CS 4102 are utilized without actually implementing the algorithms from scratch- students end up using built-in libraries to run the algorithms. On the other hand, in CS 4102, students have to use data structures and objects in order to write code for some specific algorithms. There are major overlapping themes between the two classes that can be synthesized to provide a more valuable experience to students.

Students would benefit from these courses being combined in terms of application of these ideas to CS work. While applying to CS jobs, it is very common for prospective hires to have to show a combination of skills that require a mastery of topics covered in CS 2110 and CS

4102. Interviews for these jobs consist of coding assessments, design questions and other general questions (Gardner, 2013). Combining these two classes would be very beneficial to students with interviews and also in the workforce because they would learn how data structures and algorithms interact.

There are several ways that the ideas in these courses can be synthesized to optimize student learning. In CS 2110, there is a strong emphasis on learning the language Java, and not specific data structures over different languages. A proposed change to this would be to have a heavy focus on data structures and then teach students what those data structures can be used for-such as algorithms. While the course is being taught with the focus on data structures, for the students to get practice with the newly learned material, they can implement certain algorithms using the developed data structures to satisfy both learning objectives at once. Additionally, some advanced data structures already contain algorithms within them as properties, so analyzing and breaking down those structures and making students rewrite those algorithms could be useful.

There are several scholarly articles that assert that data structures being taught with algorithms benefits CS education compared to the traditional approach. By analyzing the results of the findings of these articles, the proposition that these two CS courses can be combined can be made with evidence. The frameworks used by education researchers support the idea of combining the general ideas of data structures and algorithms and having them be taught together for the sake of understanding and application to the field of CS (SynergisticIT, 2020).

To propose future improvements to the course, there is a great deal of information that is needed. Starting with current information about the two courses, analysis can be done to see what ideas specifically are being overlooked, or what ideas between the two courses would be

optimized by being combined. This involves the syllabi for the courses, specific course materials, assignments, homework, and projects. Also to correctly make this proposition, evidence of workplaces and interviews using the content that is found in both courses need to be utilized. By addressing scholarly articles with CS recruiting and general workplace ideas, evidence that supports synthesizing these courses can be added to the proposition.

STS Problem

Ever since the start of the information age in the mid-20th century, technology has taken the world by storm and impacted millions and millions of people. With the advent of the internet and the World Wide Web, there are countless new technologies that have been made to improve human experience. From research on drones to social media, technology has been used every which way and continues to shape the way that humans, society, political entities, and nature interact. One idea that has been at the forefront of technical discussion for a while has been the idea of the ethics of algorithms and other automated programs. There have been many instances where biases have been unknowingly built into applications and algorithms process them in a way that is detrimental to a group of users (Alake, 2020). The groups that design these algorithms hold control of what exactly goes into their algorithms, so accounting for certain biases needs to be completed before a product is put on the market. For example, there is a significant discrepancy between females and males in accuracy of face recognition software, and even between darker skin tones and lighter skin tones (Najibi, 2020).

In this project, I propose that algorithms and product development push the social agenda to an extent in addition to social justice issues that are being highlighted at the time of advent. Additionally, the technologies that have been created as a result of new discoveries have added fuel to the fire of current social issues. The idea of society having an impact on technology and

its design has been presented through several frameworks and arguments. This proposition mainly echoes the sentiments of social constructivism, which says that society shapes the technology around it; most technology is a product of what society needs (Kline, 1999). Others argue that there is a balance between social constructivism and technological determinism (the argument that technology shapes society and new inventions drive social agenda), which is also displayed by the idea of technological momentum (Hughes, 2009). After considering all of these frameworks, I will use the Technological Determinism framework to support the argument that algorithms and their applications have brought forward issues in society rather than the other way around.

To support my argument, I will use a plethora of scholarly articles and research about specific instances in which algorithms have pushed social change. Additionally, I will also consider the opposite, to confirm that this is not the other way around. The main focus of this proposition will be based around facial recognition algorithms, health-care decision making technology, and other image analysis algorithms among other examples. I will also analyze the breakdown of users of each of these algorithms in an attempt to decipher any trends in the training and testing of these algorithms. After completing this data collection, it is important to take a look at the direct social impacts of these algorithms, if any. Utilizing scholarly articles to gauge if there have been significant social impacts as a result of these technologies will add to the understanding of this argument. Accounting for the possibility of the result being the opposite of what was stated in the argument, the data collection will involve looking for counterexamples as well.

Conclusion

The technical report will offer the synthesis of two classes in the University of Virginia computer science program that will address the issues of having those classes separate, while adding features that will optimize the real world applications of this learning through combining the classes. This will contribute to the curriculum by offering a new alternative to the current classes that have been scheduled for students. The STS report will look to explain the impact and interaction of algorithmic bias and society, and how technology and society have been shaping each other since the advent of the computer.

The output of the technical report will address the issues brought up in the problem frame about computer science education by the design of a new class including the synthesis of ideas from two perspectives. The results of the STS paper will display how exactly algorithms and new technologies have impacts in society and they will also show how humans interact with technologies to alter them and bring about positive change.

These two projects work in conjunction with each other because they both relate to how computer science and its auxiliary fields need a great deal of improvement. The idea of algorithms is discussed in both the STS topic and the Technical Topic; with the concepts in algorithms being taught in a different way, the development of algorithms can be enhanced thus reducing bias as well. With these two topics being looked at in conjunction, the improvement of CS can happen efficiently.

References

Alake, R. (2020, April 28). *Algorithm bias in artificial intelligence needs to be discussed (and addressed)*. Medium. Retrieved November 1, 2021, from https://towardsdatascience.com/algorithm-bias-in-artificial-intelligence-needs-to-be-discussed-and-addressed-8d369d675a70.

Gordon, A. (2019, July 17). *Computer Science vs. Computer Programming: What's the difference?* Medium. Retrieved November 1, 2021, from https://medium.com/@adamjgordon24/computer-science-vs-computer-programming-whats-the-difference-5e3764be9532.

How data structures and algorithms are important for computer science graduates - SYNERGISTICIT. Synergistic It. (2021, April 12). Retrieved November 1, 2021, from https://www.synergisticit.com/how-data-structures-and-algorithms-are-important-for-computer-science-graduates/.

Hughes, T. P. (2009). *Technology and society: Building our sociotechnical future*. MIT PRESS.

Klein, H. K., & Kleinman, D. L. (2002). The Social Construction of Technology: Structural Considerations. *Science, Technology, & Human Values*, *27*(1), 28–52. http://www.jstor.org/stable/690274

Matlin, C. (2016, February 2). *Defining algorithms-a conversational explainer*. Slate Magazine. Retrieved November 1, 2021, from https://slate.com/technology/2016/02/whats-the-deal-with-algorithms.html.

Najibi, A. (2020, October 26). *Racial discrimination in face recognition technology*. Science in the News. Retrieved November 1, 2021, from

https://sitn.hms.harvard.edu/flash/2020/racial-discrimination-in-face-recognition-technology/.

NBCUniversal News Group. (2019, March 12). *The World Wide Web is 30 years old - and its inventor has a warning for us*. NBCNews.com. Retrieved November 1, 2021, from https://www.nbcnews.com/tech/tech-news/world-wide-web-30-its-inventor-has-warning-us-n982156.

Utah State University Digitalcommons@usu. (n.d.). Retrieved November 1, 2021, from https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=3024&context=etd.

Vegas, E., & Fowler, B. (2020, August 4). *What do we know about the expansion of K-12 Computer Science Education?* Brookings. Retrieved November 1, 2021, from https://www.brookings.edu/research/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/.