What is an Algorithms Course? Survey Results of Introductory Undergraduate Algorithms Courses in the U.S.

Michael Luu
University of California, Irvine
Irvine, CA, USA
luum6@uci.edu

Matthew Ferland*
University of Southern California
Los Angeles, CA, USA
mferland@usc.edu

Varun Nagaraj Rao*
Princeton University
Princeton, NJ, USA
varunrao@princeton.edu

Arushi Arora
Randy Huynh
arushia2@uci.edu
randylh@uci.edu
University of California, Irvine
Irvine, CA, USA

Frederick Reiber
Boston Universtiy
Boston, MA, USA
freddyr@bu.edu

Jennifer Wong-Ma
Michael Shindler
jwongma@uci.edu
mikes@uci.edu
University of California, Irvine
Irvine, CA, USA

ABSTRACT

Algorithms courses are a core part of many CS programs, but have received little focus in computing education, lacking statistical data about how they are generally taught. To remedy this, we present the results of the first large-scale comprehensive survey of undergraduate introductory algorithms courses at four-year institutions in the United States. Questions in the survey targeted instructor information, course concepts, the ways students are evaluated, challenges instructors encountered, and instructor envisioned improvements. We received 87 responses from 34 different states, across a wide variety of 4-year institutions. The results indicate that algorithms courses vary dramatically in most surveyed areas.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**.

KEYWORDS

algorithms education, survey, curricula

ACM Reference Format:
Michael Luu, Matthew Ferland, Varun Nagaraj Rao, Arushi Arora, Randy Huynh, Frederick Reiber, Jennifer Wong-Ma, and Michael Shindler. 2023. What is an Algorithms Course? Survey Results of Introductory Undergraduate Algorithms Courses in the U.S.. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569820

1 INTRODUCTION

The study of the design and analysis of algorithms is fundamental to undergraduate computer science (CS) education [8]. In addition to their wide applicability, algorithmic skills are foundational to improving student programming ability [15].

Educators play an important role in establishing curriculum standards and ensuring graduates are competent. The ACM Computer Science Curricula 2013 [8] makes such an attempt for *undergraduate CS education.* It provides "realistic and adoptable recommendations" for 18 knowledge areas, including "Algorithms and Complexity." This curricula was formed through a combination of survey data and and several rounds of community feedback. However, this lacks detailed (public) data about the content and structure of courses.

It is important to understand the current content and structure of courses for several reasons. First, it can help establish curriculum standards like those published by the ACM [1, 2, 8]. Second, the complexity of transferring course credits between institutions can be reduced when content is aligned and prerequisites are well-calibrated [3]. And third, it can help inform future curriculum revisions and education research, such as the prioritization of topics for inclusion in a concept inventory (CI), like one created for data structures [10]. For more about Computer Science CIs, see [14].

We set out to answer two research questions - *What is the current content and structure of introductory algorithms courses?* and *What associations and trends emerge from the analysis of the content and structure of algorithms courses?* To answer these questions, we have conducted the first large-scale comprehensive survey of introductory algorithms courses at institutions in the U.S. that offer a four-year undergraduate degree in CS or a closely related field (e.g. software design, information technology). We sent the survey to 411 faculty from across all 50 states who have recently taught such a course. We received 87 responses. The rest of the paper is structured as follows - we discuss data collection and survey design (Section 3), we present quantitative results of our survey (Section 4) and we analyze our results to obtain additional data-driven insights (Section 5). Overall we see major divergences in course structure and content

2 RELATED WORK

Prior work on algorithms education has primarily focused on identifying student misconceptions. Farghally et al. [4] study misconceptions across a list of algorithm analysis topics, including asymptotic

*Equal Contribution

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569820

analysis and recursive analysis, among others. Shindler et al. [12] perform a replication study of [19] that targeted misconceptions about dynamic programming. Özdener [9]'s study identifies students' misconceptions about time-efficiency of algorithms, while Velázquez-Iturbide [16]'s work addresses misconceptions about optimization problems and their corresponding algorithms.

Other research related to algorithms concerned effective teaching and evaluation strategies [5, 11, 18] and means to incorporate responsible-computing content into the course structure [6].

More relevant to our work is the study conducted by Hertz [7]. Their survey was focused on CS 1 and CS 2 course topics, whereas our survey concerned all algorithms course topics. Similar to our own work, they found significant divergence between courses.

3 RESEARCH METHODS

3.1 Data Collection

The first step of our data collection process entailed creating a (non-comprehensive) list of academic institutions to include in the study. We only included universities in the United States that had a 4-year computer science (or related) program. We also ensured that each of the 50 states was represented, at least to some extent.

Once our list of institutions had been constructed, we began examining the catalog descriptions of undergraduate courses at each institution in order to identify a suitable course. If the course title indicated the class was introductory and solely an algorithms course (e.g., "Design of Algorithms," "Analysis of Algorithms," "Fundamental Algorithms," etc.), we included it. Otherwise, if there was no such course, we would include another course with "algorithm(s)" in the title so long as there was evidence that at least one topic from the ACM's "Algorithmic Strategies" was included. This condition was most often applied to "Data Structures and Algorithms" courses, which were often a terminal algorithmic course.

After creating lists of undergraduate algorithms courses, for each course, we compiled a list of up to three instructors who had taught it within the past 2 years. To find the instructors for each course, we applied a variety of methods. The most common way was through looking at publicly viewable course schedules which list instructors' contact information, though other sources were employed, such as course websites. After finalizing the list of emails, we sent an invitation to participate in the survey to all instructors in that email list. Each survey respondent was offered a $25 Amazon gift card for completing the survey.

3.2 Survey Design

When designing the survey, our goal was to gain insight into both the course structure and the topics typically taught. The survey contained two main parts. The first, consisting of three sections, focused on the organization of the course. These sections ask about general course details, such as grading breakdowns, dedicated class section times, programming assignments, written assignments, and in-class assessments (exams and quizzes).

One of our motivations behind these questions was to determine emphasis on the theoretical components versus applied components of the courses. As such, we asked questions about both the content and number of assignments.

In the second part, based on the core topics from the ACM Computer Science Curricula 2013 [8], we asked where the topic is first taught (i.e., prerequisite course, the algorithms course in question, other elective or required courses, nowhere in the curriculum, unsure, or other). Multiple core topics under Basic Analysis were combined into the topic "Asymptotic Analysis" to simplify and shorten the survey length. Similarly, we included an abridged version of the Proof Techniques section from the Discrete Structures knowledge area to identify how proofs were integrated into the course. The "Basic Automata, Computability and Complexity" and "Advanced Data Structures, Algorithms, and Analysis" sections were preceded by a question on whether the instructor taught any topics that could fall under either section and if not, the section was skipped.

Finally, we asked three open ended questions, on issues encountered, desired course changes, and any other comments.

3.3 Threats to Validity

The most significant threat to the validity of our survey is in the selection process of courses and institutions. In the first wave, we selected the initial universities ourselves. As such, the first institutions we reached out to were more often well-known universities. Our second and third waves addressed this problem by attempting to be almost fully comprehensive in previously uncovered states along with a few other states. However, it should be noted that this means we do not have a perfectly random sample. Secondly, while "properly titled" introductory courses would always be surveyed, programs that didn't have such a course were only surveyed if they included a topic from "Algorithmic Strategies." This criterion, however, may cause some confirmation-bias, since this pre-supposes that algorithms courses need to cover these strategies.

Another threat comes from the six adjunct respondents. While adjuncts likely know the course they are teaching very well, it is possible that they are less familiar with the general curriculum of the institution as a whole. A survey by the TIAA Institute indicates around 23% of adjuncts have jobs outside of academia, and 26% teach at multiple institutions [17]. As such, this could lead to certain information being omitted or misrepresented.

Finally, there are threats related to doing surveys. All of the data is given voluntarily by those who responded, which can bias the sample. There can be errors when filling out surveys. For ours particularly, we allowed the final two sections to be skipped if the instructor indicated they did not teach any topic in the section. It could be that in these cases, the topics are taught in different locations at these universities. There is also the issue of sample size; we only received responses from 87 instructors.

4 RESULTS

Our initial list had 615 higher-education institutions. Of these, 495 had a 4-year computer-science adjacent program, and 373 of those had an applicable algorithms course. From this, we were able to send out emails to 411 instructors from 302 institutions whose contact information we were able to find. We were able to gather responses from 87 instructors across 79 institutions in 34 states. 57 of these responses were from doctoral universities with 36 being from an R1 university, 12 from an R2 university, and 9 from a doctoral/professional university as defined by the Basic Carnegie

Classification [13]. 14 of these responses were from master's degree-granting institutions, and 16 were from baccalaureate colleges. These rates mirror the ACM 2013 curricula survey [8]. 70 of these institutions used a semester system, seven used a quarter system, and two classes had a term length of 12-13 weeks. As answered by respondents, of the distinct courses, 70 were upper-division courses, 11 were lower-division, and 2 were indicated as both (by different respondents teaching the same course).

4.1 Instructor Background

We found that over 90% of instructors were professors, comprising 31 assistant professors, 25 associate professors, and 24 full professors. 6 respondents were adjunct faculty, and one was a graduate student teacher. We also asked about their involvement in teaching in comparison to research. We found that only 11% of instructors considered themselves more focused on research, with 54% being more teaching-oriented. Additionally, we categorized the respondents' research areas into theoretical and non-theoretical[1]. Of the 82 that did any amount of research, we classified 39 of them as doing theoretic research.

4.2 Course Structure

4.2.1 General Course Details. Participants were asked to describe the grading criteria of their courses. We provided categories for participation, programming assignments, in-class assessments (exams, quizzes, etc), and written homework assignments[2] (Figure 1). Among these categories, participation and programming assignments typically accounted for the lowest portion of the total course grade, with nearly 29% of respondents weighting programming assignments at 0%, and over 55% of respondents weighting participation at 0%. In-class assessments, in contrast, were usually the highest portion of the course grade, with over 26% of instructors noting that they weighted it at over 60% of the total grade.

For the "other" category, responses varied. There were presentations, student-made class notes, digital quizzes, reading assignments (with a flipped-classroom), take-home exams, and some logistical complications due to different grading systems.

4.2.2 Evaluation of Student Mastery. Among the 81 instructors who gave non-programming homework assignments, the average number of assignments was 8.1, with many indicating that they required proof writing, algorithm design, complexity analysis, and algorithm tracing. Similarly, for the 85 instructors who gave in-class assessments, an average of around 4.3 were given, again with the same five areas as the homework.

4.2.3 Programming Assignments. Of the 66 respondents (76%) with programming assignments, they assigned on average 5.2 throughout the academic term. There was significant variation among which languages were accepted - Java (89%), Python (47%), C++ (36%), and C (13%). 13% accepted any language. 91% of instructors indicated their assignments including having students implement algorithms, 82% had assignments where they design their own, and 21% had assignments that involved plotting the algorithms with data (eg: the time elapsed at various input sizes).

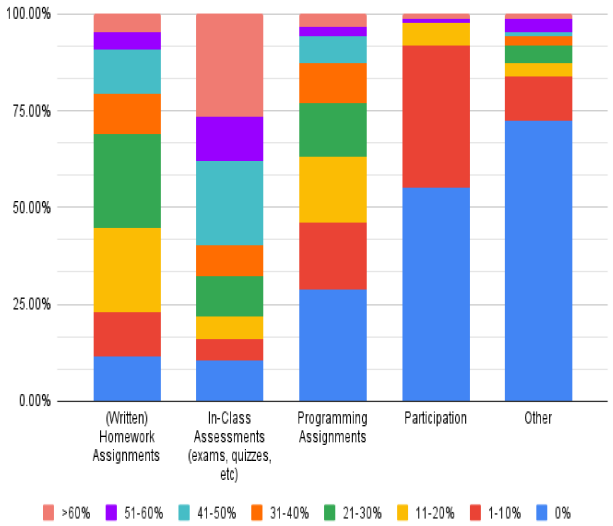


Figure 1: Grading criteria for course categories

4.3 Course Topics

In figure 2, we present graphs of all of the results from this part of the survey. We use the written part of this section to note what we believe are the highlights.

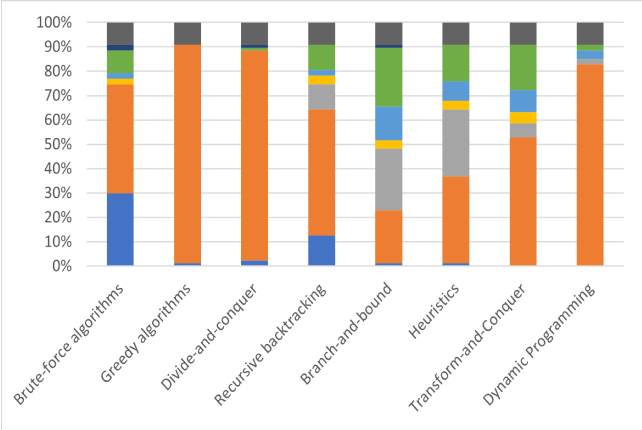
4.3.1 Algorithmic Strategies. This first part of the Course Topics section deals with topics concerning algorithm design techniques. Of the 79 responses[3], we observe that nearly all of the responses cover divide-and-conquer and greedy algorithms with one and four responses respectively stating the topic is covered in a different course. The same is true for dynamic programming, with only seven responses stating they do not cover it. The other topics were taught in between 24% and 58% of classes.

4.3.2 Fundamental Data Structures and Algorithms (DSA). Multiple topics related to data structures, like binary search, numerical algorithms, binary trees, hash tables, and sorting algorithms, are taught in a prerequisite course for 45 to 60% of responses. Moreover, graph topics, such as graph representations as well as breadth- and depth-first search, are taught in over 50% and over 60% of the courses respectively. Notably, shortest-path and minimum-spanning tree algorithms were taught by over 80% of the respondents.

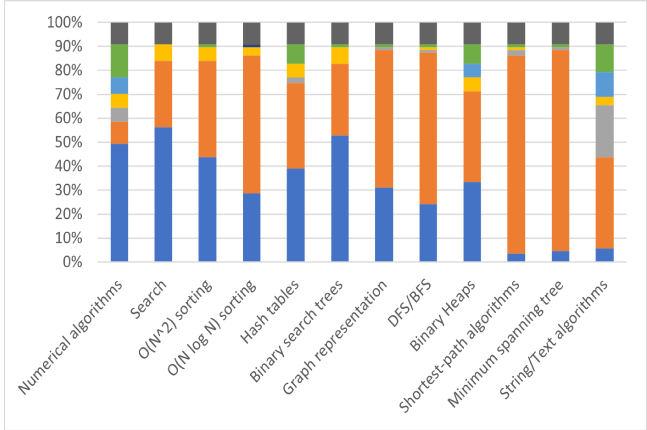
4.3.3 Proof Techniques. We split the proof techniques section into two main categories: basic logic (ie: notions of implication, equivalence, converse, inverse), and mathematical proof techniques. Proof techniques were taught in algorithms by only 11 instructors (13%), and only 1 taught elementary logic. Both topics were covered in previous courses by 47 respondents (54%), with 2 indicating only proofs were taught in a perquisite, and 7 (8%) indicating that logic was

[1]We considered theoretical to be any research involving algorithms, data structures, theory of computation, complexity, or other heavily mathematically-oriented fields.
[2]One respondent did not use a weighted grading system. They instead had specific requirements for each grade. They gave >60% to homework, quizzes, and participation

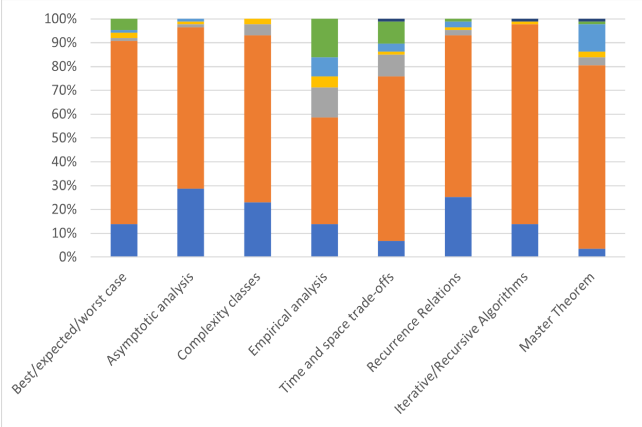
[3]Due to an early error in the survey, we did not receive responses from 8 instructors for this section and the Fundamental Data Structures and Algorithms section.



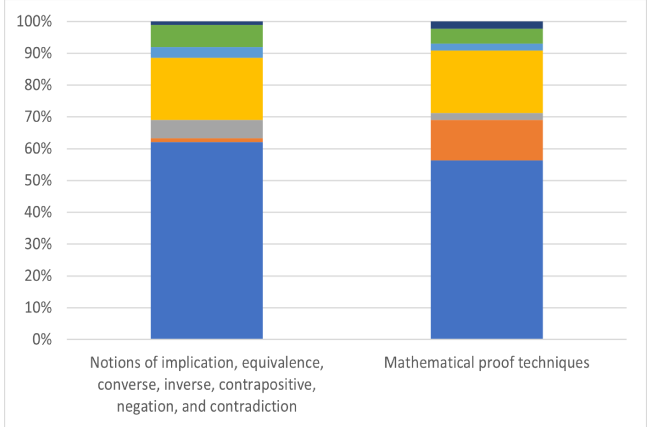
(a) Algorithmic Strategies



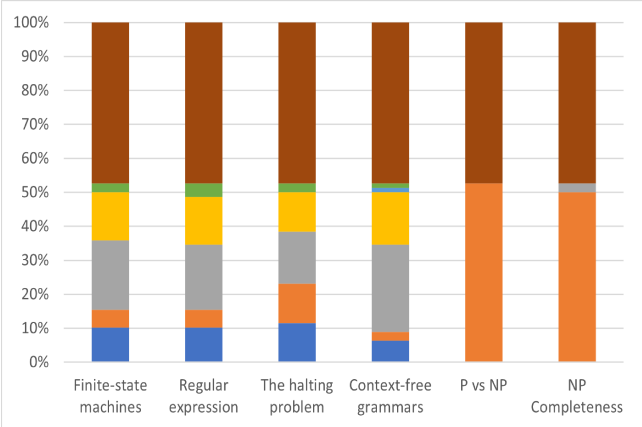
(b) Fundamental Data Structures and Algorithms



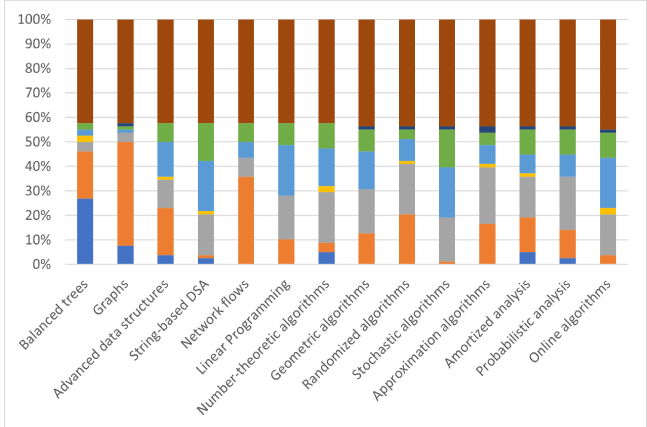
(c) Basic Analysis



(d) Proof Techniques



(e) Basic Automata Computability and Complexity



(f) Advanced Data Structures Algorithms and Analysis

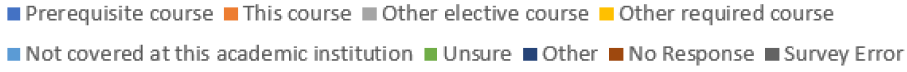


Figure 2: Distribution of where Sub-Topics are taught as defined by ACM Computer Science Curricula 2013 [8]

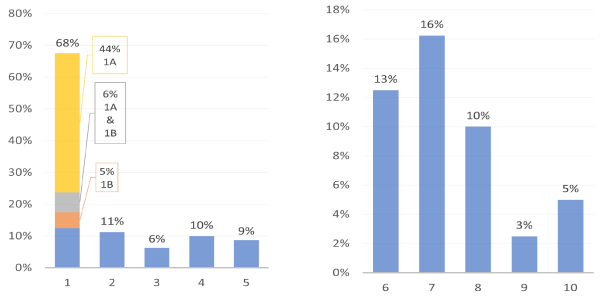


Figure 3: Breakdown of themes of 80 responses for Q1 (left)[4] and 46 responses for Q2 (right)

previously taught but not proofs. Most other instructors indicated that it was taught in a required (but non-prerequisite) course.

4.3.4 Basic Analysis. The basic analysis section of our survey contained several categories, such as asymptotic analysis and recurrence relations. The majority of respondents (over 50% or higher for each topic) indicated all eight topics as being covered in their own algorithms courses; around 10%-20% of respondents noted that these basic analysis topics were covered in a prerequisite course.

4.3.5 Basic Automata, Computability, and Complexity. Automata and complexity theory was skipped by 52% of instructors surveyed. However, the other 48% cover at least P vs NP, 45% cover NP-completeness, and 10% also cover the halting problem. Only 2% to 4% of instructors covered other complexity and theory of computation topics, but those topics appeared elsewhere in the curriculum in an elective, prerequisite, or required course.

4.3.6 Advanced Data Structures, Algorithms, and Analysis. Topics in advanced DSA and analysis were skipped by 47% of instructors. Out of the remaining 53% of instructors, graph theoretical algorithms such as topological sort and finding strongly connected components were common with 73% of instructors who answered this section covering it (38% overall). Network flow is also fairly common, covered by 64% of this section's respondents (34% overall). Balanced tree-related algorithms were usually covered in prerequisite courses. Instructors who answered this section indicate that the remaining topics from this section are either typically covered by an elective course or not covered in the curriculum at all.

4.4 Free Response Questions

In the final section of the survey, we asked respondents three optional free-response questions about difficulties encountered in teaching their course (Q1), changes they would like to implement assuming they had full control over their course (Q2), and any other final thoughts they believed would be important for us to know. We found commonalities among the responses and categorized them into common themes with Q1 and Q2 each having 5 different themes (with 2 sub-themes for Q1), listed in Table 1. Percentage breakdowns of these themes are presented in Figure 3.

[4]The stacked bar graph represents responses showing Theme 1 and subthemes 1A and 1B (responses may have exhibited either or both as shown) with the remaining blue section showcasing the percentage of responses that did not fall under 1A or 1B, often due to plainly stating that prerequisite knowledge was lacking.

Theme	Description
Q1	
1	Difficulties regarding students' insufficient background, particularly in these areas: A) Mathematical and proof-writing skills B) Programming and implementation skills
2	Difficulties in creating problems for in-class assessments and homework assignments, especially those with answers that cannot be easily found online
3	Difficulties with illustrating relevance and real-world applications for algorithms
4	Difficulties with a particular algorithms topic that students struggled with
5	Difficulties in having students adequately apply algorithm design techniques to new problems
Q2	
6	More programming assignments and focus on implementation of algorithms
7	Cover less topics, move topics to other courses (existing or new), or replace topics with other topics
8	Ensure prerequisite courses cover necessary knowledge
9	More emphasis on proofs and mathematical aspects
10	Add lab/recitation section to the course for more chances to practice problems

Table 1: Themes in instructors' difficulties with teaching algorithms (Q1) and in changes instructors' would like to implement, assuming full control over course (Q2)

5 ANALYSIS

5.1 Implications of Instructors' Background

As previously noted, 39 respondents reported performing research in theoretical computer science or mathematical fields while 47 were involved in other kinds of research or none at all. For the 37 of those 39 theory researchers who gave written homework assignments, 33 (89%) indicated that they require students to write proofs, all 37 required algorithm runtime analysis, 35 (95%) required designing algorithms, and 22 (59%) required tracing algorithms. For the 44 instructors involved in other fields of research who gave homework, 28 (64%) required proofs, 39 (89%) required algorithm runtime analysis, 37 (84%) required algorithm design, and 35 (82%) required tracing algorithms.

Notably, those involved in theoretical research required proofs more often and tracing algorithms less often than those not involved in theoretical research, suggesting some difference in emphasis on the mathematics of algorithms versus the implementation. This is also reflected in the in-class assessments, sporting similar trends to the data regarding assignments. Although, it is significant to note that only 61% and 49% of theory and non-theory researching instructors respectively expect students to write proofs in assessments. Furthermore, the grading criteria for programming assignments, written assignments, and exams did not significantly differ between instructors involved in theoretical and non-theoretical research.

5.2 Themes from Free Response Questions

When teaching algorithms, by far the most common issue instructors reported facing was insufficient background on prerequisite concepts, with 54 of the 80 (68%) responses to Q1 exhibiting this (Theme 1). 40 out of those 54 (74%) made references to students' weak foundation in mathematics and proof-writing. Only nine responses (17%) expressed concern with a lack of programming skills to properly implement algorithms (the final 10 responses (19%) not elaborating on the prior knowledge students were lacking).

Another concern raised was the assessment of students' algorithm design skills (Themes 4 and 7). Instructors frequently commented on the difficulty in creating assessments and assignments for students to apply their knowledge in designing algorithms. A key difficulty is to come up with novel problems, unique from those discussed online. Another is that some students are unable to apply algorithm design beyond problems encountered in class.

Respondents also noted the abstract nature inherent in learning these topics as a struggle. Themes 5 and 6 illustrate this as instructors note students' struggles in recognizing the importance of algorithms and their relevance with real-world applications. Some instructors listed particular algorithmic concepts that students had trouble with as well, most commonly dynamic programming, greedy algorithms, and NP-completeness.

Of the 46 responses we collected for Q2, we witness two categories of themes: those expressing desires to add to the course in some way, like adding more material, emphasizing particular topics, or providing more opportunities to apply what they learned (Themes 6, 9, 10) and those which seek to address the difficulty of the course by reducing the amount of material covered or ensuring prerequisite courses cover the concepts needed (Themes 7 and 8).

5.3 Associations Among Survey Variables and Institution Data

We looked to see if there were any associations between the survey variables and additional data we collected on the institutions we surveyed[5]. We also used our previous categorization of instructors' research in algorithms or non-algorithms adjacent research as a variable. We obtained these associations by calculating the Cramér's V measure for each pair of variables[6].

We found that there are some moderate associations between variables from the same sections of the survey. We first see this with the variables from the "Evaluation of Student Mastery" sections where question types on the homework and exams were correlated with one another. Likewise, for the Fundamental DSA topic section, there are moderate associations between quadratic sorting algorithms, hash tables, binary trees, and heaps, having Cramér's V values in the range 0.18 and 0.55. On the other hand, graph representations, depth- and breadth-first search, shortest-path algorithms, and minimum spanning tree algorithms, the most common topics from the DSA section, have Cramér's V measures less than 0.13 with the other topics (and associations between 0.13 and 0.78 amongst the topics). Given these observations, it is possible that the "typical" DSA topics concern tree traversal and graph algorithms. Furthermore, $O(n \log n)$ sorting algorithms are the only DSA topic that has any moderate associations with any Algorithmic Strategic topics, most notably having an Cramér's V score of 0.34 with divide-and-conquer, suggesting $O(n \log n)$ sorting algorithms may be covered to serve as examples for algorithmic paradigms. For the advanced algorithms topics, they have moderate intra-associations and weak inter-associations, possibly implying these topics are rarely covered together. Balanced trees, graph-theoretic algorithms, and network flow, which show up in a higher percent of algorithms/prerequisite courses, have less association with the other advanced topics.

We did not find any high associations between institution data and instructor's research area and our survey variables, suggesting that algorithms course structure and topics are not particularly dependent institution type or professor research background.

[5]This includes the institution's Basic Carnegie Classification, its research activity, and whether or not the institution was public/private and religious/nonsectarian.

[6]We note that these associations values do not indicate the kinds of relationships the variables have and whether variables are positively or negatively correlated with one another in one manner. We provide interpretations of these associations with the intention of yielding additional potential avenues of consideration and future study.

6 FUTURE WORK

A more focused survey could expand upon the themes we derived. Other research could look for and apply solutions to these issues. Finally, conducting surveys in other areas can further increase our understanding of the current state of CS education.

7 CONCLUSION

In this study, we found that while instructors often use the term "algorithms" to refer to a specific course, the actual course being referenced has a large variation between academic institutions: it can be oriented around mathematics, programming, or somewhere in between. Even within these divisions, instructors often select very different subsets of topics. In addition, instructors regularly feel students are unprepared for their courses and have a variety of new directions they wish to take it - whether to focus on certain course topics or to make the class more or less mathematical.

As educators and researchers, we need to take into account the diversity in how algorithms courses are taught. Future studies should keep this in mind, and either make efforts to be applicable to a variety of courses or be tailored to a specific version of the course. When interacting with new transfer students or graduate students, we should be careful not to make assumptions about the topics covered in prior algorithms courses. Finally, as educators, we should make use of this heterogeneity to pull elements from other courses into our own to continue to improve and refine.

ACKNOWLEDGEMENTS

We are extremely grateful to the 87 respondents who filled out the survey. We are grateful to Michael Dillencourt for helpful comments with early drafts of the survey. We would also like to thank an anonymous reviewer from a different paper that one author published at CCSC-Southwest 2022, which helped to highlight the need for this work. Varun Nagaraj Rao was supported in part by NSF awards #1916153, #1956435, and #1943584. Matthew Ferland was supported in part by Simons Investigator Award for fundamental and curiosity-driven research and NSF grant CCF-1815254. We also thank UCI's Academic Senate Council on Research, Computing and Libraries (CORCL) for the research fund that enabled us to provide gift cards as incentives for participants.

REFERENCES

[1] ACM. 2005. Computing Curricula 2005. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf

[2] ACM. 2020. Computing Curricula 2020. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

[3] Philip E Agre. 1999. Information technology in higher education: The "global academic village" and intellectual standardization. *On the Horizon* 7, 5 (1999), 8–11.

[4] Mohammed F Farghally, Kyu Han Koh, Jeremy V Ernst, and Clifford A Shaffer. 2017. Towards a concept inventory for algorithm analysis topics. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Seattle, WA, USA, 207–212.

[5] Mohammed Fawzi Seddik Farghally. 2016. *Visualizing algorithm analysis topics*. Ph. D. Dissertation. Virginia Tech.

[6] Kathi Fisler, Sorelle Friedler, Kevin Lin, and Suresh Venkatasubramanian. 2022. Approaches for Weaving Responsible Computing into Data Structures and Algorithms Courses. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*. ACM, Providence, RI, USA, 1049–1050.

[7] Matthew Hertz. 2010. What do "CS1" and "CS2" mean? Investigating differences in the early courses. In *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, Milwaukee, WI, USA, 199–203.

[8] The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. 2013. Computer Science Curricula 2013. https://dl.acm.org/doi/pdf/10.1145/2534860

[9] Nesrin Özdener. 2008. A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. *Computers & Education* 51, 3 (2008), 1094–1102.

[10] Leo Porter, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Kevin C. Webb, Cynthia Lee, and Michael Clancy. 2019. BDSI: A Validated Concept Inventory for Basic Data Structures. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 111–119. https://doi.org/10.1145/3291279.3339404

[11] Michael Shindler, Matt Ferland, Aaron Cote, and Olivera Grujic. 2020. Experience report: preemptive final exams for computer science theory classes. *Journal of Computing Sciences in Colleges* 35, 10 (2020), 9–14.

[12] Michael Shindler, Natalia Pinpin, Mia Markovic, Frederick Reiber, Jee Hoon Kim, Giles Pierre Nunez Carlos, Mine Dogucu, Mark Hong, Michael Luu, Brian Anderson, Aaron Cote, Matthew Ferland, Palak Jain, Tyler LaBonte, Leena Mathur, Ryan Moreno, and Ryan Sakuma. 2022. Student misconceptions of dynamic programming: a replication study. *Computer Science Education* 32, 3 (2022), 288–312.

[13] Lee S Shulman. 2001. The Carnegie classification of institutions of higher education.

[14] Cynthia Taylor, Michael Clancy, Kevin C Webb, Daniel Zingaro, Cynthia Lee, and Leo Porter. 2020. The practical details of building a CS concept inventory. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, Portland, OR, USA, 372–378.

[15] Pinar Mihci Türker and Ferhat Kadir Pala. 2020. The effect of algorithm education on students' computer programming self-efficacy perceptions and computational thinking skills. *International Journal of Computer Science Education in Schools* 3, 3 (2020), 19–32.

[16] J Ángel Velázquez-Iturbide. 2019. Students' Misconceptions of Optimization Algorithms. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Aberdeen, UK, 464–470.

[17] Paul Yakoboski. 2018. Adjunct faculty: Who they are and what is their experience.

[18] Jeffrey Young and Eric Walkingshaw. 2018. A domain analysis of data structure and algorithm explanations in the wild. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, Baltimore, MD, USA, 870–875.

[19] Shamama Zehra, Aishwarya Ramanathan, Larry Yueli Zhang, and Daniel Zingaro. 2018. Student misconceptions of dynamic programming. In *Proceedings of the 49th ACM technical symposium on Computer Science Education*. ACM, Baltimore, MD, USA, 556–561.