Important and Difficult Topics in CS2: An Expert Consensus via Delphi Study

Lea Wittie, Bucknell University

Lea Wittie is an Associate Professor in the department of Computer Science in the Engineering College at Bucknell University.

Anastasia Kurdia, Tulane University

Anastasia Kurdia is a Senior Professor of Practice of Computer Science at Tulane University. She received her undergraduate degree in Applied Mathematics from Belarusian State University, and Ph.D. from University of Texas at Dallas.

Prof. Meriel Huggard, Trinity College Dublin

Dr. Meriel Huggard has been a tenured faculty member in Trinity College Dublin, Ireland since 2000. During 2015/16 she was a visiting associate professor in Electrical and Computer Engineering at Bucknell University, PA. She teaches courses in computer

Khai-Nguyen Nguyen, Bucknell University

A senior student at Bucknell University

Important and Difficult Topics in CS2: An Expert Consensus via Delphi Study

Lea Wittie
Department of Computer Science
Bucknell University
Lewisburg, PA 17837

Anastasia Kurdia
Department of Computer Science
Tulane University
New Orleans, LA 70118

Meriel Huggard
School of Computer Science and Statistics
Trinity College Dublin
Dublin 2, Ireland

Nguyen Nguyen
Bucknell University
Lewisburg, PA 17837

Abstract

Almost every computer science program contains two semester-long introductory courses, usually named Computer Science 1 (CS1) and Computer Science 2 (CS2). They have been a mandatory element of the ACM Computing Curriculum for nearly fifty years and are likely to remain so for many years to come. While there seems to be a broad agreement on the key elements of CS1, the curriculum for CS2 can vary significantly between institutions. What material should in fact be included in CS2? Ideally, an educator would want to cover the topics that students need to master in order to successfully learn other topics further in the course and in the subsequent courses in the curriculum (important topics). They would also want to concentrate on topics that students are likely to struggle with and take a longer time to understand (difficult topics).

This paper details the process and results of a multi-year international study that examined the topics of difficulty and importance in CS2 using Delphi method (an iterative process for reaching consensus among a group of subject experts that allows the participants to reconsider their opinions based on the anonymized responses of the other experts in the group provided at preceding iterations). We present four topic sets aiming to inform the choice of topics for designing a CS2 course or exam. The first set contains the topics selected based on importance and indicates what topics should be included in a CS2 course or its textbook. The second and third sets of topics are based on both importance and difficulty and as such would be a guide for the creation of exams and concept inventories. The fourth set contains the topics that none of the faculty experts marked as either important or difficult, making them candidates for delegation to a different course, exclusion, or leaving them for self-study. We also provide a comparison with other published topic sets for CS2.

Introduction

A concept inventory (CI) is a research-based multiple-choice test that seeks to measure a student's knowledge of a set of concepts while also capturing conceptions and misconceptions they may have about the topic under consideration. Each multiple-choice question includes one correct answer and a set of expertly designed incorrect answers that would result from one or more misconceptions. Compared a to straightforward multiple-choice test, where the most useful information lies in how many students answer each question correctly, a concept inventory provides an additional level of insight into the students' (mis)understanding of the material. From the instructor's standpoint, it can be used to assess the impact of changing instruction methods on student understanding, to identify topics that need a different instructional approach, or as a way of comparing instructional methods[1], to identify appropriate teaching and learning activities, or to evaluate overall learning and instructional effects[2]. For the student, it can guide future learning by not only confirming the areas they mastered but also highlighting the topics that they haven't fully grasped at a finer level compared to a standard test. (We note that a CI is not an appropriate instrument for assessing teaching performance or for determining grades[1]). CIs for many subjects have already been developed[3,4,5,6,1,7,8,9,10,11,12,13,14,15,16]. One of the earliest concept inventories in computer science is that of Almstrum et al.[1]

Developing a concept inventory involves identifying a set of topics that the CI should cover, creating questions, conducting field tests, carrying out validity checks, and reliability testing. The very first step (finding the set of topics for the concept inventory that would be useful for many instructors of one course) becomes a difficult task when there is a great variety in opinions on what should constitute the content of such a course. In fact, there is a significant variance in the curriculum of the second-semester introductory computer science course (commonly referred to as CS2) between institutions. Which topics should be included in concept inventories for CS2? Which topics should the instructors and developers of instructional materials for CS2 focus on? The goal of this paper is to help address these questions. We present details of the process and results of a multi-year international study that examined the topics of difficulty and importance in CS2. As is common in previous CI work, this study uses the *Delphi method* to identify these topics. The Delphi method is a systematic iterative process of arriving at a common opinion or decision by a diverse group of subject experts[17]. At each iteration (round), the experts are provided with a questionnaire and an aggregate anonymized group response at the preceding round that might affect the experts' responses at a current round[18,17,19,20]. After several rounds, the group's response closely reflects the consensus opinion of the group. By giving each expert's opinion equal weight and through the use of anonymous feedback, each expert is only influenced by the opinions and arguments put forward by other experts, and not by their reputation in the field[16]. Using the Delphi method also eliminates the natural bias of individual experts based on their experience and field of expertise[21].

The contributions of this paper are twofold. We present four topic sets that characterize the topics that should (or should not be) included in the learning materials, exam, or concept inventory of a CS2 course. The first set of topics is based on importance and suggests what topics should be included in a CS2 course or its textbook. The second set and the third sets of topics are based on both importance and difficulty and would be targeted in the creation of exams and concept inventories. The fourth set of topics contains the ones that none of our experts marked as either

important or difficult. Such topics become candidates for relocation to a different course, exclusion, or self-study. We also provide a comparison with another published topic set for CS2[8,22].

The rest of the paper is organized as follows. In Section we introduce the methods of identifying the topics for the concept inventory. Section presents the data we obtained. In Section we compare these results with the relevant published topic set. Section concludes the paper.

Methods

Recruiting and selecting experts The experts were recruited in person at international computing/engineering education conferences (SIGCSE, ASEE, FIE, EDUCON), via dedicated lightning talk at SIGCSE'18[23], and via email outreach. Approximately 500 email invitations were sent to CS2 textbook authors and computer science departments on all continents. Among those recipients, we intentionally included multiple school sizes, and both private and public institutions as well as 4-year and community colleges.

The group of 34 experts initially signed up to be a part of our study. We then asked them to take a quick survey on the focus of the CS2 course that they were involved in (focus area meant that at least 60% of the course was spent on that area). We had identified 3 versions of CS2: one which taught students to use existing data structures from libraries (*Application-focused* CS2), one that taught students to implement their own data structures (list/stack/tree/queue), and also to use them (*ADT-focused* CS2), and a third option, which focused on object-oriented programming (*OOP-focused* CS2). Nineteen CS2 experts indicated they were involved in the ADT-focused CS2 and seventeen of those chose to participate further in this study.

Our efforts to recruit participants with a wide variety of backgrounds were partially successful. All seventeen participants were current or recent instructors of a CS2 course. Six of the participants also conducted research on CS2; one of them additionally had authored CS2 textbooks. Three of the participants were female and fourteen were male. Seven taught in 4-year universities with a large graduate program and ten taught in predominantly undergraduate 4-year universities. One was teaching in Asia and sixteen were teaching in North America. We achieved a good balance of undergraduate- (10) vs graduate- (7) focused schools but no participants from 2-year institutions. According to the Taulbee survey[24], in 2018, when the experts were recruited, 20.8% of computer science professors were female so our 17.6% female participation is very close to being representative. Although we attempted to heavily recruit from each continent, we only succeeded in recruiting one participant from outside North America.

In phase 1, we asked our experts to create an unordered list of the 10-15 topics they felt were most important and difficult for a CS2 course. Although the overall focus of this study was data structures and implementation/usage, the topics in this phase came from the entire range of topics our experts covered. Seventeen experts completed phase 1. The researchers each separately and then as a group removed duplicate topics and arranged the remaining topics into 16 categories containing 119 topics overall.

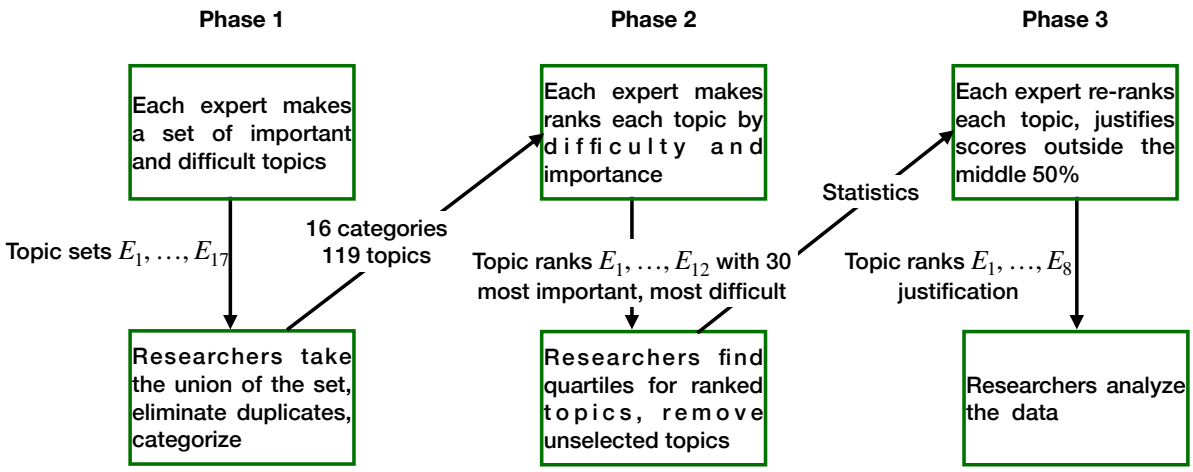

Figure 1: **Phases of the Delphi method in this study.**

Identifying topics In phase 2, we asked our experts to select and order their 30 most important and 30 most difficult topics. Twelve experts completed phase 2. From their ordered responses, the researchers found quartile data indicating which topics were in each quartile for most important and also for most difficult. The 10 topics that no one indicated in their top 30 for importance and 7 topics no one indicated for difficulty were dropped from those overall respective lists.

In phase 3, we sent out the remaining sets of important and difficult topics broken into quartiles. We asked our experts to reconsider their ranking given the quartile information and to justify any topic they kept that fell into the 3rd or 4th quartile. Eight experts completed phase 3. We stopped the Delphi rounds at this point and began our final data analysis.

The phases of the Delphi method are illustrated in Figure 1. There were intended to be 4 phases in the study. Due to the global pandemic, our participants clearly no longer had time to give to our study. We, therefore, made the decision to stop after phase 3. Goldman[19] found that in practice, the top-ranked results from phase 2 of a Delphi study accurately predicted the top-ranked topics in later phases so the loss of phase 4 is unlikely to have had much effect on our top results.

Results

The results of this Delphi study are rankings on the 119 topics for importance and for difficulty. From these rankings, we obtained sets of the most important topics (Set 1), the topics that are both difficult and important (Sets 2 and 3), and the leftover topics (Set 4). We chose Sets 1 and 2

from the topics that at least 50% of our experts agreed should be included in the top 30 topics by importance and/or by difficulty. We did not take agreement into account with Set 3.

Set 1: Important topics A set of 25 topics important to at least 50% of our experts can be seen in Table 1. Each topic is shown with its cumulative importance rank which is a measure of *how* important the topic is. Higher rank implies that the topic is more important to master. These are the topics that our study suggests should be included in an ADT-focused CS2 course or textbook. Of these topics, only binary search had 100% agreement in its importance. This shows that there is a considerable breadth of opinion on the subject of what topics are important (agreeing with the results by Hertz[25] who also showed a lack of broad agreement on what should be included in a CS2 course).

Set 2: Important and difficult topics Table 2 shows the set of the 11 important topics from Set 1 that at least 50% of our experts also agreed were difficult. Each topic is shown with its cumulative difficulty rank which is a measure of *how* difficult the topic is. A higher rank indicates that the topic is more difficult to learn. This set of difficult and important topics is useful for the creation of both exams and concept inventories. Difficulty also correlates with the time needed to cover the topic.

Set 3: Top ten topics for importance and difficulty Figure 2 shows the 10 highest ranked topics as measured by Manhattan distance to the top cumulative importance and difficulty ranking. This set does not exclude topics with low agreement.

Set 4: Leftover topics We also obtained a set of topics that are either not important, not difficult, or belong elsewhere. After the experts chose the 30 topics that were important and the 30 topics that were difficult, there were leftover topics that no one marked as either important or difficult for a CS2 course (Table 3). In short, these cold either be the topics that belong in a different course, topics that can be left for self-study, or topics that are not important or difficult relative to other topics in the ADT-focused CS2 course. Note that the topic *Linked implementation* here refers to ADTs other than the Linked List. Presumably, once students can implement a linked list, it is not difficult for them to implement a linked stack or queue.

Table 1: **(Set 1)** Cumulative importance rank of topics with at least 50% agreement in the final phase

Topic	rank	% agreement
How to choose data structures (by complexity; what's a reasonable default)	168	87.5
LinkedList and its implementation	167	87.5
Binary search trees (simple, not self-balancing)	128	87.5
Difference between array-based and link-based implementations	126	75
Stack	124	87.5
Hash tables	118	87.5
Maps	117	75
Binary search	114	100
Recursion	112	75
Queue	108	75
Array processing: resizing, insertions and removals of items	106	62.5
Abstraction	100	50
Algorithmic complexity (can include runtime or space or cache usage etc)	98	50
Recursion (as applied to usage in processing data structures)	96	75
Abstract Data Types (specifically: List, Map, Set)	95	50
Big-Oh notation	93	75
Merge sort (another divide & conquer)	87	75
Binary trees	77	50
Heaps and heapsort	70	62.5
Comparison-based sorting	66	75
Debugging	62	50
Structural recursion on trees	61	62.5
Runtime analysis	43	50
Deep vs shallow copy	40	62.5
Trees and recursive traversals	34	50

Table 2: **(Set 2)** Cumulative difficulty rank of topics with at least 50% agreement in both importance and difficulty in the final phase

Category	Topic	rank	% difficulty agreement
Recursion	Recursion	158	75
Tree	Structural recursion on trees	144	75
Recursion	Recursion (as applied to usage in processing data structures)	105	62.5
Abstraction	Abstraction	104	75
List	LinkedList and its implementation	101	75
Analysis and Big O	Big-Oh notation	89	62.5
Heap	Heaps and heapsort	65	62.5
Memory	Deep vs shallow copy	65	62.5
Array	Array processing: resizing, insertions and removals of items	50	62.5
Tree	Trees and recursive traversals	44	50
Map and Hash Table	Hash tables	39	50

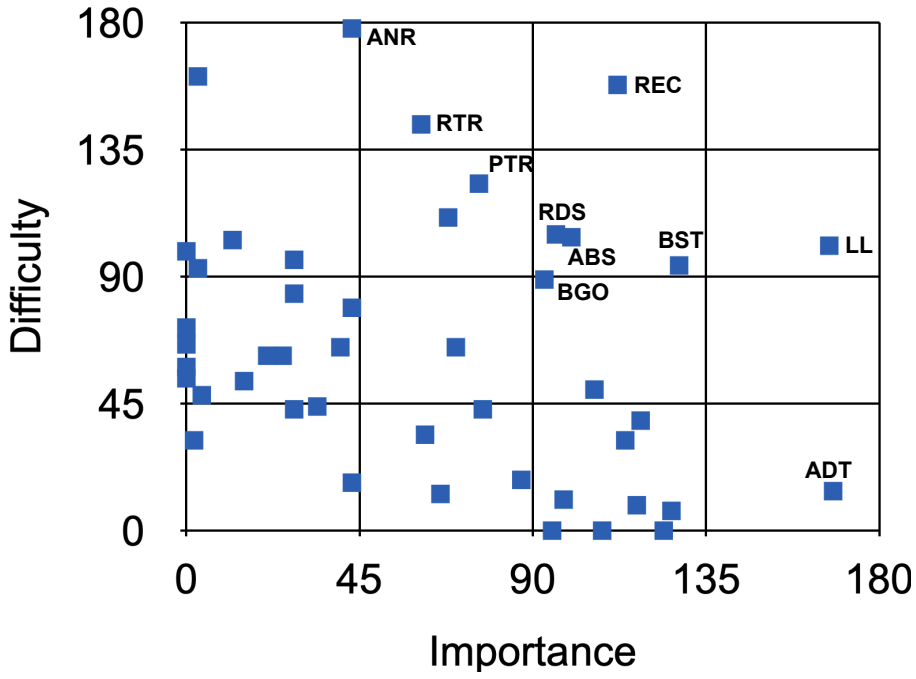

Figure 2: **(Set 3)** Topics ranked by Manhattan distance from 0,0 for cumulative importance and difficulty. The highest top ten ranked topics are labeled. The other (unlabeled) topics are shown for comparison.
REC Recursion
LL LinkedList and its implementation
BST Binary search trees (simple, not self-balancing)
ANR Analysis of recursive algorithms
RTR Structural recursion on trees
ABS Abstraction
RDS Recursion (as applied to usage in processing data structures)
PTR Dealing with pointers in linked structures
BGO Big-Oh notation
ADT How to choose data structures (by complexity; what's a reasonable default)

Table 3: **(Set 4)** Topics no one marked as either important or difficult in the final phase

Category	Topic
Abstraction and ADT	Abstraction (interfaces, how to write comments without implementation details)
Array	Matrix
Memory	How array-based data structures grow through reallocation
Graph	Creating a linked directed graph implementation
Graph	Graph traversals
List	Usage of simple data structure (list/sets)
List	Linked implementations
Set	Set implementation
Tree	Forest
Sorting	Sorting algorithms (sequential, plus parallel merge sort)
Software engineering	Organizing all collection ADTs in a single inheritance hierarchy
Software engineering	GUIs/Java AWT
Software engineering	C++ operator overloading
Software engineering	Invariants
Software engineering	Implementing a list iterator for multiple list implementations
Software engineering	MVC design pattern
Software engineering	Testing
Software engineering	Teamwork
Other	Intro to security concerns

Discussion

Core topics In comparison, the sets of top-ranked topics for both importance and difficulty with (Set 2) and without (Set 3) taking agreement into account have 6 topics in common (see Table 4 in the Appendix). The set where agreement is irrelevant includes 4 other topics that were highly ranked but less than half of our experts chose to list. The set with 50% agreement includes 5 other topics that were slightly lower ranked than those seen in the agreement irrelevant group but at least half of our experts agreed that they merited to be included in the top importance and difficulty lists. These results suggest that the topics of *Abstraction, Big-Oh notation, Linked list and its implementation, Recursion, Structural recursion on trees, and Recursion in processing data structures* should all be included in the core of an ADT-focused CS2 course (be given a large share of the time spent and should appear on exams) and in CS2 concept inventories.

Broader context A notable related work is that of Clancy, Lee, Porter, Taylor, Webb, and Zingaro[8,22] (for clarity of exposition we use the name of the alphabetically first author and refer to this work as the *Clancy Project*). Analyzing course syllabi for various CS2 courses, they identified 6 categories (*topics*[8]) taught in CS2: sorting, recursion, basic data structures, advanced data structures, object-oriented programming, and algorithm analysis. They then assembled a panel of 8 experts and generated a set of 49 topics (*subtopics*[8]) based on personal expertise and analysis of course descriptions, syllabi, exams, and other course materials from their own and expert's courses. They then asked the experts to quantify "How critical is the following category for student success in your CS2 class?" and "What is the importance that students know the

content of the following category as a prerequisite for courses that follow your CS2 class?" using a 7-point Likert scale, where 1 was "not at all critical" and "not at all important" and 7 was "absolutely essential" and "very important". The resulting assessment is summarized in Table 5 in the Appendix.

Besides the direct methodological difference (the Clancy Project used an expert panel and this study used the Delphi method to generate topics), we note a number of finer-grained differences. First, in addition to topic importance, our study also considers the question of topic difficulty. Second, our study specifically targets the experts in an *ADT-focused* version of CS2 while Clancy Project experts are involved in different versions of CS2. Third, Clancy Project's focus is on categories, while ours is on topics. Fourth, the categories in the Clancy Project are evaluated by their absolute importance (very important/not at all important) and our topics are evaluated by their relative importance and difficulty with respect to other topics of the course. Fifth, Clancy Project considers the importance of topic categories for the courses following CS2 in the curriculum, this study concentrates on the CS2 course itself. Finally, topic identification in the Clancy Project is rooted in the actual existing course materials, i.e. the categories and topics emerged from *what is already being taught* in CS2 by the experts. In this study, the task of naming the important and difficult topics was given to the experts and is likely to reflect experts' belief in *what should be taught*.

Comparing and aligning our findings with those of the Clancy Project, a number of interesting observations emerge.

Lack of complete agreement amongst different studies There is no uniform agreement of what should constitute the content of a CS2 class. Increasing the number of experts broadens the number of topics suggested for inclusion in CS2 (in the Clancy Project, the materials of 8 experts elicited 49 topics, in this study, 17 experts generated 116 unique topics).

Substantial overlap between study results There is a substantial overlap between the categories of important and difficult topics (Set 2) and the important categories in the Clancy Project: 9 of 11 Set 2 topics belong to a category from Table 5.

Topics present in our results absent in the Clancy Project Two Set 2 topics, *Abstraction* and *Deep and shallow copy*, represent respective categories Abstraction and Memory that were not explicitly identified as critical in the Clancy Project (or other previous work), but were found both important and difficult in this study. Mastering these topics is currently a byproduct of CS2 studies; clearly, many experts believe that those should instead be some of the central themes.

Topics present in the Clancy Project absent in our results Conversely, none of the topics from Sorting and OOP categories identified in the Clancy Project are included in Set 2. We interpret exclusion of OOP topics to be a result of our expert selection process (we intentionally included experts on an ADT-focused CS2). The exclusion of Sorting-related topics can be explained by the relative nature of Set 2: it is likely that other topics are seen as more important and more difficult. Notably, sequential sorting algorithms and parallel merge sort are included in our Set 4, leftover topics.

Set 2 specifies particularly important and difficult topics from Basic Data Structures (Array, List)

and Advanced Data Structures (Tree, Map and Hash Table, Heap). Standard algorithmic and data structures topics (Search, Stacks, Queues) are absent from Set 2. Our experts believed they were important (they were included in Set 1) but their relative difficulty is apparently lower than that of other topics in Set 2.

Conclusion

There is considerable variation among the CS2 community as to what the contents of such a course should be or how difficult those contents are to learn. There are however, a few categories in common between both this study and previous work; they include *Analysis and Big-O, Recursion, Basic Data Structures (Arrays, Lists), and Advanced Data Structures (Tree, Heap, Hash Table)*. Considering difficulty allowed us to identify specific important topics in these categories that are hard for students to master and where further work in concept inventory and content development is likely to be the most impactful. The expert consensus also illuminates the topics that have not previously been explicitly named important in a CS2 course (*Abstraction, Deep vs. shallow copy, How to choose data structures*). These also represent the topics where more content and assessment development work is needed.

Acknowledgements

We thank the 17 experts that participated in our Delphi study.

References

[1] Vicki L Almstrum, Peter B Henderson, Valerie Harvey, Cinda Heeren, William Marion, Charles Riedesel, Leen-Kiat Soh, and Allison Elliott Tew. Concept inventories in computer science for the topic discrete mathematics. In *ACM SIGCSE Bulletin*, volume 38, pages 132–145. ACM, 2006.

[2] Reed-Rhoads, T., and Imbrie, P. K. Concept inventories in Engineering Education. In *National Research Council Promising Practices in Undergraduate STEM Education Workshop 2*, Washington, DC., 13 – 14 October 2008.

[3] Joan Krone, Joseph E Hollingsworth, Murali Sitaraman, and Jason O Hallstrom. A reasoning concept inventory for computer science. 2010.

[4] Geoffrey L Herman. *The development of a digital logic concept inventory*. PhD thesis, University of Illinois at Urbana-Champaign, 2011.

[5] Holger Danielsiek, Wolfgang Paul, and Jan Vahrenhold. Detecting and Understanding Students' Misconceptions Related to Algorithms and Data Structures. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, pages 21–26, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1098-7. doi: 10.1145/2157136.2157148. URL `http://doi.acm.org/10.1145/2157136.2157148`.

[6] Kuba Karpierz and Steven A. Wolfman. Misconceptions and Concept Inventory Questions for Binary Search Trees and Hash Tables. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 109–114, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2605-6. doi: 10.1145/2538862.2538902. URL `http://doi.acm.org/10.1145/2538862.2538902`.

[7] Leo Porter, Saturnino Garcia, Hung-Wei Tseng, and Daniel Zingaro. Evaluating student understanding of core concepts in computer architecture. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 279–284. ACM, 2013.

[8] Leo Porter, Daniel Zingaro, Cynthia Lee, Cynthia Taylor, Kevin C Webb, and Michael Clancy. Developing course-level learning goals for basic data structures in cs2. In *Proceedings of the 49th ACM technical symposium on Computer Science Education*, pages 858–863, 2018.

[9] Leo Porter, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Kevin C. Webb, Cynthia Lee, and Michael Clancy. Bdsi: A validated concept inventory for basic data structures. In *Proceedings ACM ICER*, 2019.

[10] C. Taylor, D. Zingaro, L. Porter, K.C. Webb, C.B. Lee, and M. Clancy. Computer science concept inventories: past and future. *Computer Science Education*, 24(4):253–276, 2014. doi: 10.1080/08993408.2014.970779. URL `http://dx.doi.org/10.1080/08993408.2014.970779`.

[11] Allison Elliott Tew. *Assessing fundamental introductory computing concept knowledge in a language independent manner*. PhD thesis, Georgia Institute of Technology, 2010.

[12] Allison Elliott Tew and Mark Guzdial. The FCS1: a language independent assessment of CS1 knowledge. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 111–116. ACM, 2011.

[13] Kevin C Webb and Cynthia Taylor. Developing a pre- and post-course concept inventory to gauge operating systems learning. In *Proc. 45th ACM SIGCSE*, pages 103–108. ACM, 2014.

[14] Ricardo Caceffo, Steve Wolfman, Kellogg S. Booth, and Rodolfo Azevedo. Developing a Computer Science Concept Inventory for Introductory Programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 364–369, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3685-7. doi: 10.1145/2839509.2844559. URL `http://doi.acm.org/10.1145/2839509.2844559`.

[15] GL Herman, MC Loui, and C Zilles. Administering a digital logic concept inventory at multiple institutions. In

Proceedings of the 2011 American Society for Engineering Education annual conference and exposition, pages 26–29, 2011.

[16] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey L. Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. Setting the Scope of Concept Inventories for Introductory Computing Subjects. *Trans. Comput. Educ.*, 10(2):5:1–5:29, June 2010. ISSN 1946-6226. doi: 10.1145/1789934.1789935. URL http://doi.acm.org/10.1145/1789934.1789935.

[17] Norman Dalkey and Olaf Helmer. An experimental application of the Delphi method to the use of experts. *Management science*, 9(3):458–467, 1963.

[18] Mark J. Clayton. Delphi: a technique to harness expert opinion for critical decision [U+2010] making tasks in education. *Educational Psychology*, 17(4):373–386, 1997. doi: 10.1080/0144341970170401.

[19] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. Identifying Important and Difficult Concepts in Introductory Computing Courses Using a Delphi Process. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 256–260, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-799-5. doi: 10.1145/1352135.1352226. URL http://doi.acm.org/10.1145/1352135.1352226.

[20] Juri Pill. The Delphi method: Substance, context, a critique and an annotated bibliography. *Socio-Economic Planning Sciences*, 5(1):57–71, 1971. URL http://EconPapers.repec.org/RePEc:eee:soceps:v:5:y:1971:i:1:p:57-71.

[21] Mary A Nelson, Monica R Geist, Ronald L Miller, Ruth A Streveler, and Barbara M Olds. How to create a concept inventory: The thermal and transport concept inventory. In *Annual Conf. of American Edu. Research Association*, 2007.

[22] Cynthia Taylor, Michael Clancy, Kevin C. Webb, Daniel Zingaro, Cynthia Lee, and Leo Porter. The practical details of building a cs concept inventory. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 372–378, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367936. doi: 10.1145/3328778.3366903. URL https://doi.org/10.1145/3328778.3366903.

[23] Lea Wittie, Anastasia Kurdia, and Meriel Huggard. Recruiting experts: Toward a concept inventory for computer science 2 (abstract only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 1104, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351034. doi: 10.1145/3159450.3162210. URL https://doi.org/10.1145/3159450.3162210.

[24] S. Zweben and B. Bizot. Undergrad enrollment continues upward; doctoral degree production declines but doctoral enrollment rises. *CRA Taulbee Survey*, pages 1–81, 2018. URL https://cra.org/wp-content/uploads/2019/05/2018$_{T}aulbee_{S}urvey.pdf$.

[25] Matthew Hertz. What Do "CS1" and "CS2" Mean?: Investigating Differences in the Early Courses. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pages 199–203, 2010. ISBN 978-1-4503-0006-3.

Appendix

Table 4: Comparison of Sets 2 and 3; topics that were both difficult and important both with and without agreement.

Topic	50% agreement	agreement irrelevant
Abstraction	X	X
Big-Oh notation	X	X
LinkedList and its implementation	X	X
Recursion	X	X
Structural recursion on trees	X	X
Recursion (as applied to usage in processing data structures)	X	X
Binary search trees (simple, not self-balancing)		X
Analysis of recursive algorithms		X
Dealing with pointers in linked structures		X
How to choose data structures (by complexity; what's a reasonable default)		X
Heaps and heapsort	X	
Deep vs shallow copy	X	
Array processing:	X	
Trees and recursive traversals	X	
Hash tables	X	

Table 5: Average importance of each component for student success in CS2 and as a prerequisite for later courses[8], on a scale from 1 (not at all important) to 7 (very important).

Category	CS2 Success	As Prereq.
Object Oriented Prog.	6.4	5.9
Basic Data Structures	6.4	5.7
Recursion	6.3	5.9
Sorting	5.3	3.5
Algorithm Analysis	5.1	3.6
Advanced Data Structures	3.9	2.3