Why do Students Face Difficulties in Theoretical Computer Science? A Case Study

ARIS FILOS-RATSIKAS, University of Liverpool, United Kingdom

We present a case study which took place at the University of Liverpool, in the context of a postgraduate module in computer science. The goal is to identify the main reasons leading to students' difficulties with modules of a theoretical nature, which require them to prove the validity of statements via proofs and mathematical arguments. Not surprisingly, we find that the students that face the most challenges are those that do not have the appropriate background in similar subjects, and attribute that in part to the lack of emphasis on these topics during their previous degrees, resulting in the heterogeneity of the student cohort. We also find that an abundance of examples, exercises and practice opportunities is quite instrumental for the students in overcoming those difficulties.

CCS Concepts: • **Applied computing → Education**; • **Social and professional topics → Computing education**; **Computer science education**; *Computational thinking*.

Additional Key Words and Phrases: theoretical computer science, learning proofs and concepts, higher education, case study

ACM Reference Format:
Aris Filos-Ratsikas. 2021. Why do Students Face Difficulties in Theoretical Computer Science? A Case Study. *ACM Trans. Comput. Educ.* 0, 0, Article X (2021), 27 pages. https://doi.org/0

1 INTRODUCTION

When thinking of a degree in computer science, one might often envision a specification in software engineering, with an emphasis on coding computer programs and applications. However, in most computer science courses[1], modules in *theoretical computer science (TCS)* are as instrumental as any of the programming and software development modules [20]. TCS deals with the fundamentals of computation which underpin any specific implementations, e.g., algorithms, complexity, data structures, logic, and arguments for proving correctness and efficiency of solutions. In contrast with their more applied counterparts, these modules are much more "mathematical" in nature and often abstract away from the notion of a "computer" as we know it. As a result, students often find them particularly challenging, struggle to find the motivation to study them and fail to see their usefulness in the broader context of their education.

Given the above, the natural question to ask as educators is "what can we do to make these modules more approachable to the students and easier to interact with?". Before attempting to answer this question however, we should first ask ourselves another, more fundamental question: "What is that makes these modules particularly challenging for the students?". Is it simply their

[1]In the UK, the term "course" is roughly equivalent to a "programme of study", whereas the term "module" is equivalent to a "course", e.g., "Introduction to Programming"; we will be using these terms from now on throughout the text.

Author's address: Aris Filos-Ratsikas, Aris.Filos-Ratsikas@liverpool.ac.uk, University of Liverpool, Ashton Building, Ashton Street, Liverpool, United Kingdom, L69 3BX.

1946-6226/2021/0-ARTX $15.00
https://doi.org/0

nature? Is it the often "old-fashioned" way they are being delivered? Is it related to the students' background and approach to learning? Only if we understand the driving factors behind the students' inability to perform as well in these modules as they do in others, we can then move on to consider which measures could help alleviate the problem.

In this paper, we present a case study aimed at identifying precisely the answers to the aforementioned questions. The study has been carried out at the University of Liverpool, in the context of a postgraduate module titled "Advanced Algorithmic Techniques" for which the author was the sole coordinator. This module is of a highly theoretical nature and is offered in the postgraduate course in "Advanced Computer Science", aimed at students who have studied computer science as their primary undergraduate degree. The main motivation for this survey is the fact that compared to most other modules in the same course, the students' marks where notably lower for this module on average, and the goal is to find why this is the case, and moreover how these marks are actually correlated with the students' learning of the module material. In particular, the case study aims to find answers to questions of the following type:

- What are the main reasons for the challenges faced by students in theoretical modules?
- How does the students' performance in such modules correlate with their background?
- Is there something in the content and the delivery of such modules that the students find *inherently* particularly hard to grasp?

1.1 Background and Related Work

The challenges that students typically face when it comes to mathematical concepts have been well-documented in the literature (e.g., see [15, 21–24] and references therein). While some of these works refer to such difficulties in engineering sciences [15, 23], seemingly little work has been devoted exclusively to computer science. Most of the related work has in fact been concerned with proposing new and innovative ways of teaching theoretical computer science modules to students (e.g., see [6–10, 13, 14, 19], rather than dedicate at identifying the reasons for why these innovations are needed, and why TCS is in general a more intricate topic for computer science students' learning.

One of the main questions of this case study highlighted above is the correlation between the students' backgrounds and their propensity to perform well in TCS modules. In the field of engineering, Mustoe [23] poses a similar question with regard to mathematics and identifies the diversity of the student intake as a potential source of concern when teaching the relevant modules. His study refers to undergraduate modules and therefore his proposed solutions are not directly applicable to our case, but the main principles are still quite similar. Interestingly, it is his account of the state of UK engineering departments in 2002 that is particularly relevant to the diversity of students' backgrounds at the postgraduate level. As he explains, mathematics tend to have a different role in different institutions across the country, with some putting the necessary emphasis on it, and some seeing is as a "necessary evil", teaching it as a peripheral rather than a core subject, and often offering "discounted" versions in terms of rigour and formalism. Given the above, different students admitted to the same postgraduate course could have a seemingly similar background in mathematics (as judged from their transcripts and the curricula of their undergraduate courses) but in reality an immensely different expertise in dealing with the core elements of the subject. In a course like the Advanced Computer Science course at the University of Liverpool, where the student body is even more diverse (with students having previously studied in different universities all around the world), the problem is greatly amplified.

In a similar topic, Varsavsky [26] finds that in Australia, students with little mathematics background at the secondary education level perform notably differently from those that have studied mathematics at an advanced level. Similar observations have been made for UK Universities by Armstrong and Croft [1] and Kitchen [18]. These findings demonstrate that “making up” for a lack of mathematical background is challenging even given three of fours years of study, let alone for a single-year postgraduate degree.

Another central question of the case study is whether there are some “inherent” difficulties in learning about theoretical concepts. By “inherent” here, we of course do not mean some innate inability of the students, but rather difficulties that stem from the nature of the material itself, and which exhibit common patterns in the students’ understanding. One such difficulty is very-well discussed by Enström [8], namely the difference between *proof* and *evidence*, which students often find hard to distinguish. In the classic studies of Balacheff [2] and Fischbein [11], it was found that students did not perceive a formal, mathematical argument (a mathematical proof) as conclusive and convincing proof for the validity of a statement, and needed further evidence. Later on Chazan [5] found that students find it hard to decide whether “evidence is proof” or whether “proof is evidence”, in the subject of Geometry. As a potential reason for these misconceptions, Enström [8] identifies the fact that the students are not certain about the objective of the module: is it for them to learn how to “solve” a particular problem using proofs and arguments as a language to convey their ideas, or is it to learn how to develop these arguments in the first place? While the former is the essence of mathematics as a research area, the latter should certainly be the objective from a pedagogical standpoint. The distinction between proof and evidence was also evidently an issue for the Advanced Algorithmic Techniques module, and has been included as part of the case study.

In a related topic, Zerr and Zerr [27] found that students are more likely to classify a correct proof as correct, rather than identifying mistakes in an incorrect one; this probably attests to the fact that they are being trained to understand the course material as presented, but are not sufficiently trained to produce novel proofs using the underlying techniques. This was also documented in the Advanced Algorithmic Techniques module and is reflected in the results of the case study. Hazzan [17] attempts to understand the mental process of students when dealing with abstract, complicated concepts, like those often present in theoretical computer science modules. He argues that through the process of “abstraction reduction” [16], the students often “map” the abstract concepts to a lower level of abstraction that correlates better with their experiences. While Hazzan [17] argues that this process does not necessarily result in misconceptions or mistakes, it is certainly conceivable that it does in many cases.

2 CASE STUDY SETUP

In the section, we explain the setup of the case study, including the details of the module and the course of study and the details of the data acquisition process.

2.1 Details about the course and the module

The Advanced Computer Science course is a postgraduate 1-year course offered to students with a background in computer science, in particular students that have obtained a degree in computer science for their undergraduate studies. The course involves a mix of applied and theoretical modules; Advanced Algorithmic Techniques is a first semester module and therefore one of the first theoretical modules that the students encounter in the course. The course admits students from all over the world, with some students having previously studied at the University of Liverpool or some other UK institution, some students having studied at universities in other European countries, and a large number of students having studied at universities in Asia. This is important to note, as the potential heterogeneity of the students’ background could be quite relevant for this investigation.

2.1.1 Theoretical Modules. By a "theoretical module", we roughly mean a module that is aimed at teaching the students how to methodically and formally *prove statements* given a set of *axioms*, quite reminiscent of the standard approach in mathematics; in that sense, one can also interpret such modules as "mathematics of computer science". For a module on algorithms for example, the goal could very well be to prove a property of a given algorithm (e.g., that it terminates within a certain number of steps, that it always provides a correct outcome, etc). To provide such proofs, the students will have to employ *proof techniques* and *approaches* such as "proof by induction", "proof by contradiction", "worst-case analysis", "adversarial instance generation", as well as others (see also [8]).

To provide a concrete example, imagine that we have a sequence of n numbers stored in some data structure (e.g., an *array*). The goal is to design an algorithm which inputs this array of numbers and outputs an array in which the numbers are *sorted* in non-decreasing order, from the smallest to the largest. This task is called *sorting*, and is one of the most basic algorithmic problems routinely taught to students in a first-year module on algorithms. Ultimately, we would like the algorithm to be *correct* (i.e., always produce a sorted array) and *fast* (i.e., to take only a limited number of steps as a function of the size of the input n).

Here is where the situation becomes trickier than it initially possibly seems. First, how can we convince ourselves that the algorithm is correct? One way to go would be to code it in some programming language, test it on a very large number of possible inputs (arrays of unsorted numbers) and check that the output of the algorithm is a sorted array. But even then, this is not a proof that the algorithm is *always* correct - it is only evidence. In order to formally prove the correctness of the algorithm, we would have to employ a proof technique like those mentioned above. Which technique to apply and how to use it exactly is something that the students are meant to learn in the process of such a module.

Similarly, how can we make sure that an algorithm is fast? We could again code it and check how fast it runs on a large set of examples, but this is not a formal guarantee. In a theoretical algorithms module, the students would have to learn to *bound* the *worst-case running time complexity* of algorithms, again using an arsenal of proof techniques and mathematical arguments. One potential issue that will become particularly relevant later, is that although pretty much all students on the Advanced Computer Science course have studied algorithms as an undergraduate module, the extent to which they are familiar with this type of theoretical analyses (as opposed to e.g., learning to code different algorithms) is uncertain. For more examples of the type of concepts that one would encounter in an algorithms design module, see [8].

2.1.2 The contents of Advanced Algorithmic Techniques. The module, although coined as a module on "Advanced Algorithms" as the title clearly suggests, contains a significant portion of basic algorithmic techniques, namely algorithms for searching and sorting, asymptotic notation, basic graph algorithms, greedy algorithms, flow networks and dynamic programming, which would normally be encountered in an undergraduate module on algorithms. The latter part of the module contains more advanced topics, namely NP-completeness, linear programming, approximation algorithms, randomised algorithms, and online algorithms and competitive analysis.

It is worth mentioning that while many of these topics are typically encountered only in postgraduate modules, the topic of NP-completeness is often taught at an undergraduate level in most institutions. Additionally, the topic of linear programming is covered in another module offered in the course on Advanced Computer Science, namely "Optimisation". For this reason, these two topics are covered to a limited extent in Advanced Algorithmic Techniques, serving mostly as a recap or offering only the required tools for the remainder of the module.

2.1.3 Advanced Algorithmic Techniques in the context of the course. The module is usually elected by 30 to 40 students and is a first semester module. It is considered to be one of the most challenging ones in the course; over the last two years that was led by the current coordinator, the average mark was around 57 (out of 100), whereas the course average is around 65. The failure rate is also notably higher than most modules, at around 14% (for the main exam) and 4% (for the resit exam). Despite these numbers, the student feedback has been overwhelmingly positive, with most students pointing to the nature of the module as the main challenge, rather than the way it is delivered. The student survey and student interviews refer to the latest iteration of the module, for the academic year 2020/21. For this year in particular, besides the relatively low average mark, there was also a visible polarisation in the students' marks: many students received marks close to the bar for passing the module (50), whereas several other students received high marks (between 75 and 93), with very few students in the "expected" or "normal" range of 60-75.

2.2 Data Acquisition

To collect data for the case study, we used the following three methods:

(1) We designed a *survey* which was shared with the students that participated in the module. For this, we used the integrated "Quiz" option offered via Canvas[2]

(2) We performed a limited number of *interviews* with students, asking a series of questions related to their backgrounds, their experience with the module and their experience with the course in general.

(3) We performed an *interview* with the coordinator of another module, offered to multiple postgraduate courses, including the course in Advanced Computer Science. This module is also primarily of a theoretical nature, but the students' performance, as measured by their overall marks, was better compared to Advanced Algorithmic Techniques.

2.2.1 Student Survey. The student survey consisted of 12 questions - 11 multiple choice questions (MCQs) and 1 "open text" question. The questions were aimed at identifying the answers to the following issues:

- How difficult was the module compared to others in the same course?
- What is the level of study in the University of Liverpool compared to their previous institutions?
- What was the main reason for finding the module challenging?
- What is the background of the students with relation to similar modules, and theoretically- or mathematically-oriented material in general?
- Their engagement with the module and the effort they put into studying.
- Their level of confidence with what they have learned in the module.

Out of the 11 MCQs, there was also a question of a more "technical" nature, prompting the students to identify the difference between "evidence" and "proof" (see the related discussion in Section 1.1) - this is something that ideally they should have learned "organically" after going through the module, as this sort of questions underpin the learning of all the concepts of the curriculum. The open-text question was simply present to allow students to provide any general additional feedback.

The student survey can be found in Appendix A.

[2]https://www.instructure.com/en-gb/canvas.

2.2.2 Student Interviews. The student interviews consisted of 25 questions, aiming to address very similar issues to those identified in Section 2.2.1 above, related to the students' background, their engagement with the module, the parts that they found most challenging, what they took away from the module etc. The main difference is that the interview allowed enough time for the students to elaborate on their answers, and for the interviewer to offer clarifications and further information.

The questions asked in the interviews can be found in Appendix B.

2.2.3 Module Coordinator Interview. Besides the information elicited by the students, we also discussed these questions with the coordinator of a related module at the University of Liverpool. This module is also primarily of a theoretical nature and also concerns the design and analysis of algorithms, albeit at a more basic level, as it is targeted towards students that do not necessarily have a computer science background and in particular do not have an undergraduate degree in computer science. That being said, the module is also offered to students in the course on Advanced Computer Science as an elective module. The module also has a more practical component, as the students are asked to code some of the algorithms that they are being taught in a programming language.

The questions asked in the interview can be found in Appendix C.

2.2.4 Ethics of Data Acquisition. The data acquisition process adhered to all the ethics requirements and guidelines from the University of Liverpool and the standard of research involving data from human participants. In particular, for the student survey, we included a text in the beginning of the survey, explicitly stating the purposes of the survey, how the collected data will be used at the same time ensuring the anonymity of the participants and explaining that individual answers will be deleted after this report has been written. See Appendix D for the precise text.

For the student interviews, the permission of the students to be interviewed was obtained as written consent. In the consent form, the purposes of the interview and the data collection were clearly explained. Additionally, students were given the option to opt out of the case study at any point and have their data not used in the report. It was explicitly stated that the interview recordings would be deleted after being transcribed. A very similar consent form was also signed by the module coordinator that was interviewed as part of the study. See Appendix D for both consent forms.

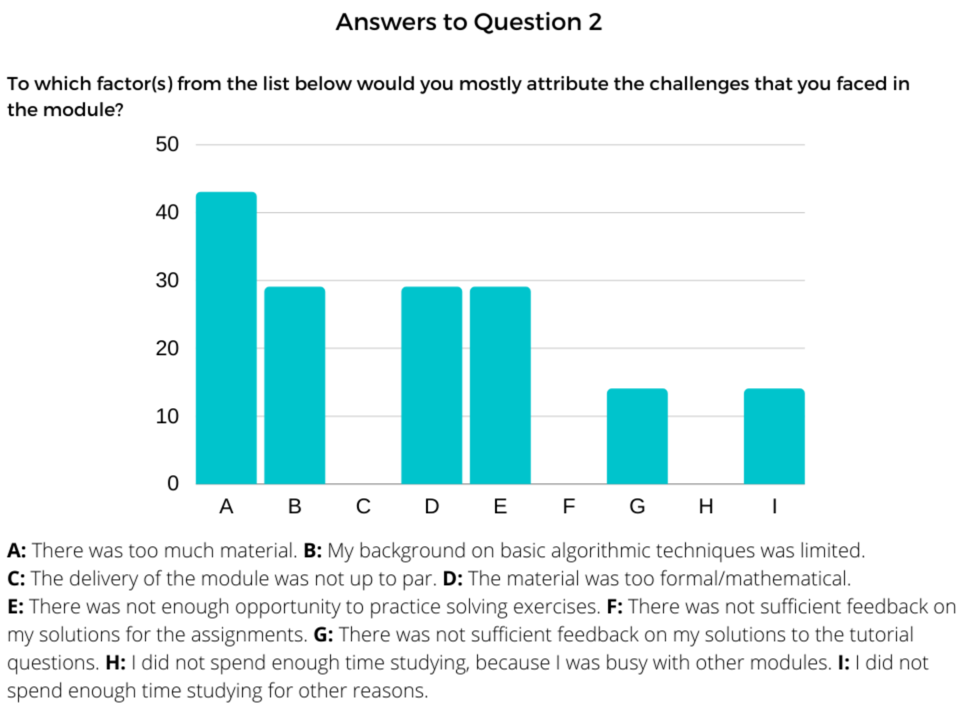
3 RESULTS AND EVALUATION

In this section, we present the results of the case study, together with an interpretation with regard to the main questions asked in the context of the study. In Section 4 we provide a more high-level overview of our findings.

3.1 Results of the Student Survey

We start with the results of the student survey; we present only the most interesting charts in this section, and defer the reader to Appendix E for all the remaining charts. Still, we discuss all the results here. The survey was completed by 20% of the students that participated in the module for the academic year 2020/21.

Question 1 of the survey asks students to compare the Advanced Algorithmic Techniques module to other modules in the same course, in terms of their difficulty. Not surprisingly all of the students rank the module in the 50% of most difficult modules, and 29% actually consider it to be the most difficult amongst all. This clearly correlates with the low mark averages but interestingly, even


Answers to Question 2

To which factor(s) from the list below would you mostly attribute the challenges that you faced in the module?

A: There was too much material. **B:** My background on basic algorithmic techniques was limited. **C:** The delivery of the module was not up to par. **D:** The material was too formal/mathematical. **E:** There was not enough opportunity to practice solving exercises. **F:** There was not sufficient feedback on my solutions for the assignments. **G:** There was not sufficient feedback on my solutions to the tutorial questions. **H:** I did not spend enough time studying, because I was busy with other modules. **I:** I did not spend enough time studying for other reasons.

Fig. 1. The answers of students to Question 2.

students that achieved high marks seem to classify the module as difficult, assuming of course that such students are present in the sample.

Question 2 regarded the factors that led to the challenges that students' faced, from their perspective. The answers to the question are shown in Figure 1. From the chart, we can see that the main issue that the students' identified was the amount of material. Recall from Section 2.1.2 that the curriculum includes a mixture of topics on basic and advanced algorithms. The rationale behind this choice is that while all students have been taught modules on algorithms in the past, a large percentage of them did not prove to be proficient enough to directly study the advanced topics in previous years. This is in fact reinforced by the answers to Questions 3 and 4 of the survey - a large number of students admitted to not having a good enough understanding of the basic concepts and in fact students unanimously thought that having them as part of the curriculum was a good idea. The answers to Question 3 in fact show an interesting pattern (see Figure 2 below): Around half of the students (43%) claimed to be fairly proficient in algorithmic topics and another 43% did not feel as confident. While it is not possible to directly correlate these numbers with the final marks of the students (due to anonymity), it certainly seems to fit the pattern of polarisation that was mentioned in Section 2.1.3. In Question 5, the answers of the students were divided across the board, but there were not any students that found the advanced material of the module easy to follow; this demonstrates that there is also some inherent difficulty stemming from the material itself, that does not seem to correlate with students' backgrounds or inclination towards theory.

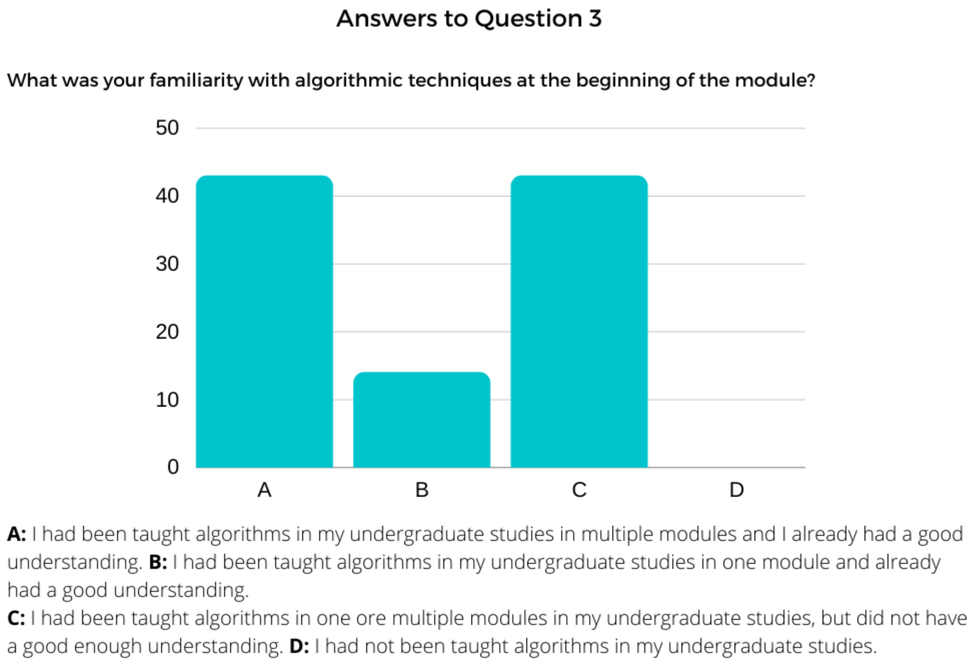


Fig. 2. The answers of students to Question 3.

Question 6 regarded the experience of the students with the module, in terms of the delivery of the module, the level of detail that was provided and the amount of practice on a class level and on an individual level. Perhaps the most important take-away from the students' answers is that a very large proportion of them (57%) found that the amount of practice exercises was insufficient for them to be able to replicate the techniques that they learned in class and apply them to other problems. An interesting point along that axis is offered by the following remark that a student has provided to Question 12 (the free text question):

> *That being said, I struggled a lot with linear programming and ILP throughout both the course and the exam. I had not done LP/ILP substantially before, outside of perhaps one or two examples during my UG studies. I feel that with more relevant examples to the assignments and exam at hand I would have learned how to deal with them in a more pragmatic manner, but the reference material did not provide such examples. Of course I'm not asking for you to just hand the students the answer to what would be on the exam and give them a 'job well done', but at the same time I do not believe the resources at the moment are adequate enough to fully teach students how to deal with LP/ILP.*

Referencing back to Section 2.1.2, linear programming is studied in detail in a separate module of the same course. That being said, the students were not necessarily aware of this overlap when they were electing their first-semester modules (i.e., that more dedicated learning of linear programming would be very useful in Advanced Algorithmic Techniques), and in fact those two modules are offered in the same term, and the student would have to study them in parallel, which is more

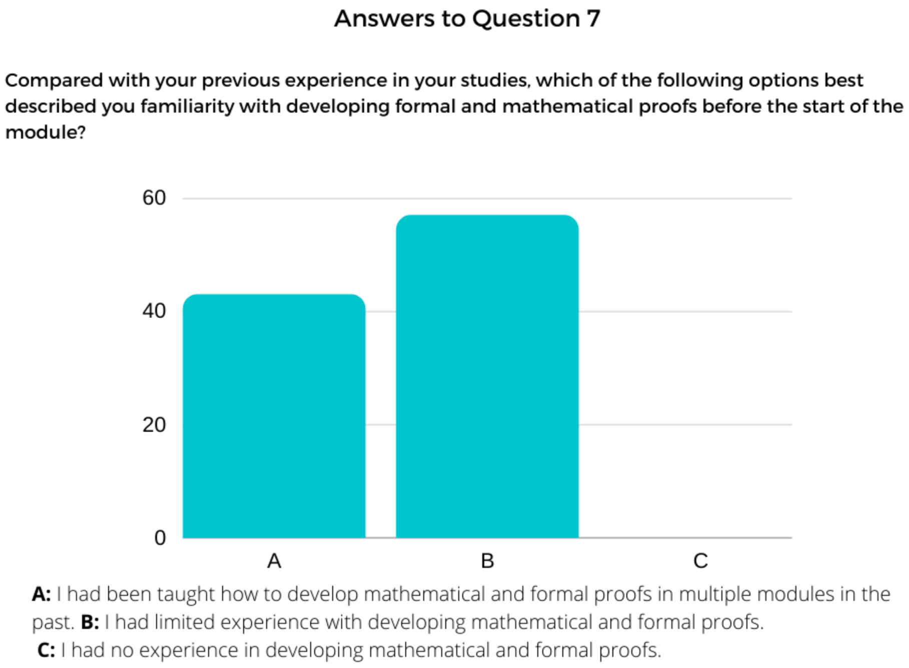


Fig. 3. The answers of students to Question 7.

of a challenge. The answers to Question 6 seem to echo the observations of Zerr and Zerr [27] mentioned in Section 1.1.

Questions 7 and 8 regard the familiarity of students with formal arguments and mathematical proofs before and after having concluded the module. The answers to Question 7 are presented in Figure 3. Most students claimed to only have had limited experience with developing mathematical and formal proofs, although in Question 3 they all have claimed to have been taught algorithms in the undergraduate courses. This could be explained by Mustoe's [23] discussion about the heterogeneity emphasis on mathematics (and generally theoretical subjects) in different courses on the same subject (e.g., computer science). The most interesting take-away from Question 8 is that the overwhelming majority of students claimed to now be confident about their analytical and mathematical skills, after having concluded the module.

However, as the answers to Question 9 demonstrate, this is not necessarily the case, see Figure 4. In Question 9, deemed "the only technical question of the survey", the students were given four statements and they had to decide whether these are "proof" or "evidence" - recall the related discussion from Section 1.1. While all of the students correctly identified the first statement as proof, all of them also incorrectly identified the second one as evidence. Additionally, statements 3 and 4 should be rather straightforwardly identified as evidence, but several students incorrectly identified them as proof.

Questions 10 and 11 asked the students whether they think that the quality of study and the difficulty of modules is higher at the University of Liverpool compared to their previous university

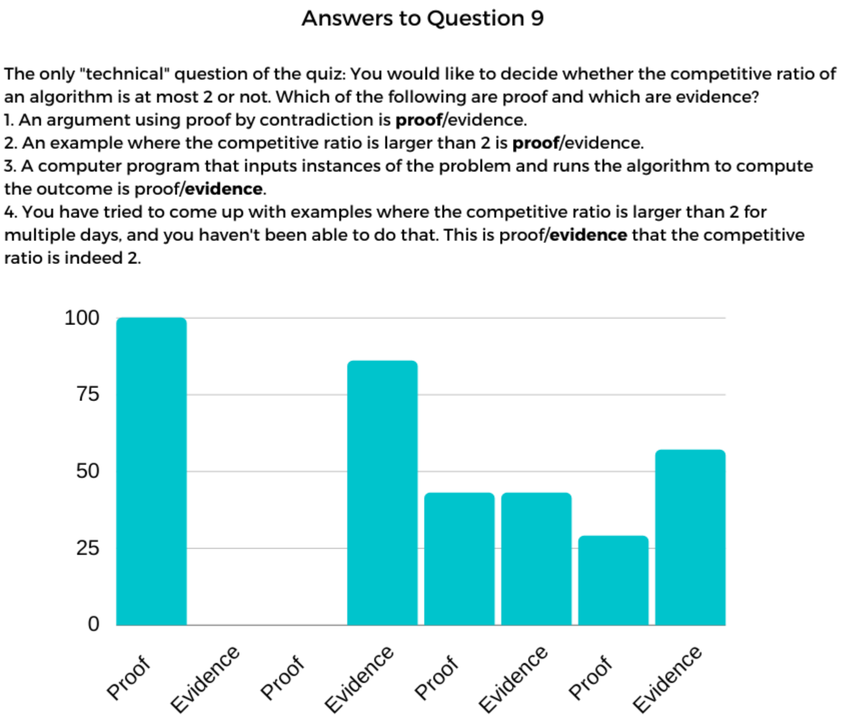


Fig. 4. The answers of students to Question 9. The correct answer is highlighted in black. Some percentages do not add up to 100% because the students were also given the option to not answer - these are not shown for ease of exposition.

of study. The students almost unanimously answered “yes” to both questions. This indicates that perhaps in general studying at the University of Liverpool is a “step up” for several of these students and that facing difficulties, in theoretical modules or otherwise, is to be expected to some degree.

3.2 Results of the Student Interviews

Further to the case study presented in the previous section, we also conducted two interviews with students that participated in the module for the academic year 2020/21. The details of the interviewees are as follows.

- Student 1 studied computer science at the University of Liverpool as an undergraduate degree. Their final mark was in the top 20% of the module.
- Student 2 studied computer science in a top 5%-ranked university in India as an undergraduate degree. Their final mark was in the bottom 20% for the module.

3.2.1 The interview of Student 1. We first summarise the most important answers provided by Student 1. The student was already familiar with theoretically-oriented modules, as the University of Liverpool offers several of those as part of the undergraduate course in computer science, and they were fairly satisfied by the emphasis that was put on such modules in their undergraduate studies (Q7, Q9). A rather interesting point was made by the student with regard to the main challenges encountered in theoretical modules (Q12): the student explained that the main challenge is that it is not often clear how the material correlates with practical applications; on the one hand, this places them in a more abstract level that the students have difficulty grasping and on the other hand, students find it hard to motivate themselves to spend the effort to learn these concepts as presented.

Regarding the advanced parts of the module (Q17), the student claimed limited familiarity with some of them, but others they claimed to have not been taught before. Interestingly, the latter set included NP-completeness, which is in fact taught at the undergraduate course in computer science at the University of Liverpool, albeit in limited scope. This demonstrates clearly that concepts that are only briefly covered might not "stick" with the students, to the extent that they do not even remember learning them in the past. At the same time, the limited exposition of NP-completeness (see Section 2.1.2) in the Advanced Algorithmic Techniques module was very much sufficient for the student to understand the concept fairly well (Q18).

A rather telling answer was provided by the student to (Q20), about the process that went into deciding with algorithmic technique to use when attempting to solve an exercise. The student correctly pointed out that problems usually exhibit patterns which "call for" the respective techniques, and credited his previous experience with algorithmic modules as the reason why they can now identify those patterns.

Finally, in their answers to (Q21) and (Q22), the student clearly indicated that they spent a significant time trying to solve practice exercises, even additional ones over these provided as part of the module material. They also credited their success in the module to this extra practice. This is particularly relevant to the previous discussion about "not enough practice opportunities" which was highlighted by the answers to Question 6 of the student survey in Section 3.1.

3.2.2 The interview of Student 2. We now move on to summarise the most important answers of Student 2. The student ranked their previous university as "definitely lower" compared to the University of Liverpool (Q5) and their workload at the University of Liverpool as "absolutely higher" (Q6). When asked about their experience with theoretical concepts like proofs and formal arguments (Q9), the student described it as "limited", pointing out that in their previous studies they did not have many dedicated modules in TCS. Another very important point was made by the student in their answer to (Q10): the student had worked in the industry as a software engineer for three years before joining the postgraduate course at the University of Liverpool. This gap is potentially significant, as the learning of this type of concepts is placed further in the past, and the student in fact was very much focused on an "application-driven" mindset in his work immediately before joining the course.

In (Q16), the student was asked to compare their experience with the parts of the module that regard basic algorithm design, in particular to compare it to the way they had been taught the same material in their undergraduate studies. The student pointed out that it was different, as this time it was more brief but also more precise. They also said that while they did not find it too hard to follow this part of the module, some of the more difficult examples definitely posed some challenges that they had never faced before. In (Q17) about the advanced parts of the module, the student commented that they had been taught concepts like NP-completeness and linear programming in their previous degree, but they admitted to lacking the deep understanding of the material that the Advanced Algorithmic Techniques module is aiming to achieve. In (Q19), the student also explained that developing proofs of correctness and running time of algorithms was much more challenging that e.g., coding an algorithm in a programming language.

Some interesting information can be extracted from the answers of the student to (Q20) and (Q21), regarding their thought process when approaching the assignments and generally trying to solve a given problem. The main challenge here for students is that there is a rather large inventory of techniques to choose from, and it is hard for them to pinpoint which one would be more appropriate for the problem at hand. The particular interviewee in fact had raised this as their main challenge as informal feedback during the course of the module; they reiterated this again as an answer to this question, but credited the learning during the module as contributing to their improvement in that regard. Finally in (Q22), the student mentioned that they would have liked to have spent more time working on exercises, either as part of the module tutorials or as extra material in the context of the module. Again, this strongly relates to the answers to Question 6 of the student survey, as explained in Section 3.1.

3.3 Results of the Module Coordinator Interview

We conclude our section about the results of the study with the main points of emphasis from the interview with the module coordinator of a related module on algorithms, offered in many of the postgraduate courses at the University of Liverpool.

The student cohort for this module is, as the module coordinator put it, "an interesting mixture" (Q2); it consists of students enrolled in the "Big Data" postgraduate course who have to follow it as a compulsory module, and it also serves as an elective module for the Advanced Computer Science course, as well as a conversion postgraduate course, for students that do not have computer science background. The coordinator quickly pointed out that this heterogeneity of students' background is the main challenge that they were faced with (Q11). This was discussed earlier with regard to Advanced Algorithmic Techniques, and referencing the work of Mustoe [23], but here it is perhaps even more pronounced, as the diversity of students' background is in a sense more institutionalised.

When talking about the level of theoretical formalism presented in the module and the perception of students when it comes to these topics (Q4 and Q5), the coordinator explained that in their module, complete proofs are rarely presented - it is usually proof ideas or sketches. Still, they believe that the students find it quite difficult to understand the theoretical parts of the module and they attribute these difficulties to the students' lack of experience with mathematics. In fact, to test the students' familiarity with basic mathematical concepts, the coordinator performed a "pre-module quiz", including several tasks in basic mathematics that the students should be able to complete before they engage in the module (Q8). There, they detected the following pattern: students on the Advanced Computer Science course and the conversion course performed better than those in the Big Data course. This is explained by the fact that the students in the first two courses usually come from a more technical/engineering background, whereas students in Big Data come from more diverse backgrounds. An additional observation is that the students of the first two groups elected this module, whereas the students in the Big Data course had to follow it as part of their course, which could imply that they exhibited different levels of motivation for studying it.

Related to this is what the coordinator expected from the students to produce in the module assignments (Q7). These were a combination of programming and theory, but in terms of the theory, the students were never asked to develop a complete proof. The coordinator pointed out that for this part, the students' solutions were quite far from what would be desired, and they were marked very generously in the end. It seemed to them that in the quest of assembling a proof via smaller components using formal arguments, the students were still far from reaching the goal, even after the completion of the module. Quite interestingly, the coordinator observed that the students were often "telling a story" about their thought process, what they tried to do and did not work, rather than actually try to compose a proof using basic building blocks; this might be indicative of how they are used to approaching a solution in a different discipline, and that they are not fully aware of the respective approach in theoretical computer science.

Perhaps the most interesting of all was the discussion around (Q8), which regarded any activities designed to help with understanding the more theoretical parts of the module. The coordinator upfront explained that they “didn’t have a good answer”, but also pointed out that one major objective for them is to provide the students some motivation for studying these concepts, trying to relate it to some applications, possibly some problems that they might encounter in their work later on etc. They also looked back at their experience as a student (Brookfield [4] calls this the “autobiographical” lens, as part of his four-lenses approach to critical reflection in teaching); as a student (albeit in a different country and a markedly different educational system), they also recall that it was quite possible to go through similar modules without actually learning much of the necessary theory, and the students that ended up being more proficient in those topics were the ones that were mostly interested in them. In their response to (Q13) (the open-ended question), they again brought up the motivation as a driving factor; as they explained, we, working in theoretical computer science, find this topic almost artistically beautiful, but it is rather dangerous to think that way when it comes to teaching. We cannot expect students to have the same viewpoint as us, and we have to seek for the motivation through their perspective (this is often referred to as the “students’ eyes” lens in Brookfield’s reflection system [4]).

Coming to the critical question about students’ marks (Q9), which was one of the main motivations for this case study: why are the marks in e.g., this module notably higher than those in Advanced Algorithmic Techniques? The coordinator explained that while the marks in their module are relatively high, this is primarily because of the nature of the exam: the students are mostly being tested on basic understanding of concepts and algorithms that they have been taught, and much less on using these ideas to produce new solutions and develop proofs. Even if a student performs very badly at the latter type of questions, they can still achieve a favourable mark from the ones of the former type. If one were to ask how satisfactory the achieved level of knowledge was for the average student (from the perspective of the coordinator), the coordinator believed that compared to e.g., a programming module, the average student’s learning in their algorithms module is still on a more superficial level.

4 CONCLUSIONS FROM THE CASE STUDY

What have we learned from the case study? The main take-aways seem to be centred along three axes: the students’ diverse backgrounds, the opportunities for practice and the students’ motivation for learning in theoretically-oriented modules.

4.1 The students’ background

From the results of the students’ survey, as well as the interviews conducted with the students and the module coordinator, clear patterns seem to emerge. Students that have had experience in TCS modules in the past, or even some extended experience with mathematics seem to perform much better. This was evident in the algorithms module taught by the module coordinator that we interviewed, which was by design targeted to a very diverse student cohort, but it was also very much present in the data collected for the Advanced Algorithmic Techniques module. Interestingly, students that studied at the University of Liverpool, or other universities that put emphasis on theory as part of the their undergraduate studies seemed to navigate through the rather demanding postgraduate module with relative ease. On the other hand, students from universities where theory was only peripherally taught seemed to encounter more difficulties. We refer the reader to Section 1.1 and the related discussion around Mustoe’s [23] observations about the level of mathematics taught in different UK institutions.

This brings forward an interesting observation: it is not enough to look at a student’s degree qualification to decide whether they are suitable for a postgraduate course or a specific module. We really have to look deeper in what they have learned in their courses and the way that the respective modules were delivered. From the data that we collected, it was evident that all students participating in the module had been taught algorithms in the past, and based on that information, they were all eligible for learning about advanced topics on algorithms in the module. However, this seems to be true only under the assumption that they were taught these subjects in the same manner as they would have been in the University of Liverpool, but this is clearly not the case. Therefore, these students should have additional opportunities to learn about these topics "the right way", meaning the way that would allow them to cope with the difficulties introduced by their postgraduate modules, and for this, reforms would need to take place at the course level, rather than on the module level. All of the students found the recap on basic algorithms offered as part of Advanced Algorithmic Techniques very useful (see Section 2.1.2 and the answers to Question 4 in Appendix E), but clearly for some students this is insufficient to compensate for a lack of experience in approaching these topics more theoretically. As the module coordinator that we interviewed expertly put it:

> *A years-long exposure seems to work, not surprisingly, but there does not seem to be any substitute for that.*

4.2 Opportunities for practice

A common theme in the answers of students, those that performed better and those that performed worse, is that they would like to have had more opportunities to practice what they have learned as part of the module. It seems that one of the tasks that the students find most challenging is to try to apply a technique or an approach that they have learned applied to one problem, to solve a *different* problem. More specifically, students are mostly able to follow a proof as it is being developed, but they are less likely to be able to use the same approach to prove a similar but different statement. This was corroborated by the experiences of the module coordinator that we interviewed, and has also been observed more generally as discussed in Section 1.1 and the survey of Zerr and Zerr [27].

In our experience, learning to do that comes through excessive repetition. For example, to get a good sense of how to design a reduction for proving an NP-hardness result, one has to normally work out a large set of such reductions, to obtain the right intuition of how to approach it (see also [8]). Showing the students a limited number of examples and asking them to replicate them is unlikely to work. In the author's experience in working at non-UK institutions, students would go through weekly 3-hour long tutorials of exercise-solving sessions in order to obtain this type of experience. Within the restricted time frame of UK postgraduate courses however, this is not easily implementable. The most realistic alternative would be for students to spend time practicing at home - this was done by Student 1 as explained in their interview, but not Student 2 (see Section 3.2). The coordinator could provide feedback in terms of model solutions, but it is hard to replicate the immediate feedback that the students would get in a tutorial environment.

This is actually inherently one of the main challenges in studying TCS as opposed to studying e.g., computer programming: if a piece of code does not compile, or does not produce the correct outcome, the student can find a lot of information on how to debug it online, and can easily test it on their code. When trying to develop a proof for a statement however, such information is not readily available, and it is very hard for the student to know if they are doing the right thing or not, certainly without an instructor to help them. As it was evident by the interviews, those students with a stronger background in TCS-related topics are more likely to engage in this process.

4.3 The students' motivation

As it turns out, one of the most important issues with teaching theoretical modules is that of the students' motivation; why should they learn these topics and how will they be useful to them in the future? This is often surprisingly overlooked by the lecturers of those modules and was in fact also possibly overlooked at the beginning of this case study, but it certainly emerged as a central consideration. Especially for a computer science degree (undergraduate or postgraduate), the students might go in with the expectation that they will learn how to develop software, including several state-of-the-art approaches in software engineering and programming. This is true to a large extent, but as we mentioned in the introduction, the underpinning theoretical principles are also quite important for the students' all-around education.

A major challenge is that in reality, it is possible to become a software engineer, even perhaps a successful one, without being proficient in the theoretical principles of computer science. The students are probably, to some level, aware of this, and might be predisposed against putting the effort into learning in TCS modules. As educators, we need to find examples of application areas where the TCS principles will be useful and convince the students that in the job market, a candidate with a more global understanding of computer science is much more desirable. As we discussed in Section 3.3, in order to do that we will have to perhaps abandon our idealised view of TCS as "a beautiful area" and try to see what the students, who will not necessarily move on to purse careers in TCS, could salvage from this in the future. In this quest however, we have to be very mindful that we should not compromise the integrity of TCS and offer "compromised" versions that are devoid of its fundamental principles, as evidently happens in several institutions.

As the module coordinator pointed out, there is often an inclination, either in school or at university, to classify students as "a maths person" or not. We should do our best move away from any such mindset and find ways to motivate the students to learn about theory or mathematics the same way they do about programming or software development. Anyone can become "a maths person", given the right motivation and teaching.

4.4 Other Conclusions

When it comes to mathematical modules, what often comes under scrutiny is the "traditional" way in which they are delivered and new and innovative ways of engaging the students are sought [7–10, 13, 14]. While these are certainly valuable, the results of the case study did not reveal any issues with the way the module was delivered, which pretty much followed a variant of the standard "chalkboard teaching" approach. In fact, studies have found that this approach is still probably best-suited for teaching mathematics and related subjects [3, 12, 25]. As the student survey and the student interviews revealed, the students were quite satisfied with the structure and the means of delivery, and the challenges lied elsewhere; primarily the lack of sufficient practice as highlighted earlier. From this we could extract that sometimes "overcorrecting" with interventions in the learning style, simply because it seems to be aligned with popular approaches may be ill-advised.

Another interesting conclusion is that students do not wish to see "easier" modules, just for the sake of obtaining better marks. This is instructive to university directors and decision-makers in higher education, who are often seemingly willing to compromise the integrity of study in order to achieve certain desirable "averages", as an evidence of good student performance. Students still want to obtain the best education possible, and we should try to elevate their abilities towards the desired level, rather than lower the desired level towards their current abilities. The students of Advanced Algorithmic Techniques, though achieving their lowest marks on average across their degree marks on this module, unanimously appreciated the challenges that it imposed and what they learned in the end. It would be easy to make our modules easier - the real challenge is to make our students better.

REFERENCES

[1] PK Armstrong and AC Croft. 1999. Identifying the learning needs in mathematics of entrants to undergraduate engineering programmes in an English university. *European Journal of Engineering Education* 24, 1 (1999), 59–71.

[2] Nicolas Balacheff. 1988. A study of students' proving processes at the junior high school level. In *Second UCSMP international conference on mathematics education*. NCTM.

[3] Anneli Billman, Ansie Harding, and Johann Engelbrecht. 2018. Does the chalkboard still hold its own against modern technology in teaching mathematics? A case study. *International Journal of Mathematical Education in Science and Technology* 49, 6 (2018), 809–823.

[4] Stephen Brookfield. 1998. Critically reflective practice. *Journal of Continuing Education in the Health Professions* 18, 4 (1998), 197–205.

[5] Daniel Chazan. 1993. High school geometry students' justification for their views of empirical evidence and mathematical proof. *Educational studies in mathematics* 24, 4 (1993), 359–387.

[6] Pierluigi Crescenzi, Emma Enström, and Viggo Kann. 2013. From theory to practice: NP-completeness for every CS student. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 16–21.

[7] Emma Enström. 2013. Dynamic programming-structure, difficulties and teaching. In *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1857–1863.

[8] Emma Enström. 2014. *On difficult topics in theoretical computer science education*. Ph. D. Dissertation. KTH Royal Institute of Technology.

[9] Emma Enström and Viggo Kann. 2017. Iteratively intervening with the "most difficult" topics of an algorithms and complexity course. *ACM Transactions on Computing Education (TOCE)* 17, 1 (2017), 1–38.

[10] Emma Enström, Gunnar Kreitz, Fredrik Niemelä, Pehr Söderman, and Viggo Kann. 2011. Five years with kattis—using an automated assessment system in teaching. In *2011 Frontiers in Education Conference (FIE)*. IEEE, T3J–1.

[11] Efraim Fischbein. 1982. Intuition and proof. *For the learning of mathematics* 3, 2 (1982), 9–24.

[12] Christian Greiffenhagen. 2014. The materiality of mathematics: Presenting mathematics at the blackboard. *The British journal of sociology* 65, 3 (2014), 502–528.

[13] Hashim Habiballa and Tibor Kmet'. 2004. Theoretical branches in teaching computer science. *International Journal of Mathematical Education in Science and Technology* 35, 6 (2004), 829–841.

[14] Wilhelmiina Hamalainen. 2004. Problem-based learning of theoretical computer science. In *34th Annual Frontiers in Education, 2004. FIE 2004.* IEEE, S1H–1.

[15] Diane Harris, Laura Black, Paul Hernandez-Martinez, Birgit Pepin, Julian Williams, and with the TransMaths Team. 2015. Mathematics and its value for engineering students: what are the implications for teaching? *International Journal of Mathematical Education in Science and Technology* 46, 3 (2015), 321–336.

[16] Orit Hazzan. 1999. Reducing abstraction level when learning abstract algebra concepts. *Educational Studies in Mathematics* 40, 1 (1999), 71–90.

[17] Orit Hazzan. 2003. How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science. *Computer Science Education* 13, 2 (2003), 95–122.

[18] Ann Kitchen. 1999. The changing profile of entrants to mathematics at A level and to mathematical subjects in higher education. *British Educational Research Journal* 25, 1 (1999), 57–74.

[19] Maria Knobelsdorf, Christoph Kreitz, and Sebastian Böhne. 2014. Teaching theoretical computer science using a cognitive apprenticeship approach. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 67–72.

[20] John Konvalina, Stanley A Wileman, and Larry J Stephens. 1983. Math proficiency: A key to success for computer science students. *Commun. ACM* 26, 5 (1983), 377–382.

[21] Johan Lithner. 2011. University mathematics students' learning difficulties. *Education Inquiry* 2, 2 (2011), 289–303.

[22] AT Morgan. 1990. A study of the difficulties experienced with mathematics by engineering students in higher education. *International Journal of Mathematical Education in Science and Technology* 21, 6 (1990), 975–988.

[23] Leslie Mustoe. 2002. The mathematics background of undergraduate engineers. *International Journal of Electrical Engineering Education* 39, 3 (2002), 192–200.

[24] Yvette Solomon. 2007. Not belonging? What makes a functional learner identity in undergraduate mathematics? *Studies in Higher Education* 32, 1 (2007), 79–96.

[25] Attila Szabo and Nigel Hastings. 2000. Using IT in the undergraduate classroom: should we replace the blackboard with PowerPoint? *Computers & education* 35, 3 (2000), 175–187.

[26] Cristina Varsavsky. 2010. Chances of success in and engagement with mathematics for students who enter university with a weak mathematics background. *International Journal of Mathematical Education in Science and Technology* 41, 8 (2010), 1037–1049.

[27] Jessica M Zerr and Ryan J Zerr. 2011. Learning from their mistakes: Using students' incorrect proofs as a pedagogical tool. *Primus* 21, 6 (2011), 530–544.

A STUDENT SURVEY QUESTIONS

Question 1. Compared to other modules in the same course (MSc programme), how difficult did you find COMP523 - Advanced Algorithmic Techniques overall?

Please choose one option from below.

(1) Less difficult than 50% of the modules.
(2) More difficult than 50% of the modules.
(3) More difficult that 25% of the modules.
(4) More difficult than 10% of the modules.
(5) It was the single most difficult module.

Question 2. To which factor(s) from the list below would you mostly attribute the challenges that you faced in the module?

Please select as many answers as you think are appropriate.

(1) There was too much material.
(2) My background on basic algorithmic techniques was limited.
(3) The delivery of the module was not up to par.
(4) The material was too formal/mathematical.
(5) There was not enough opportunity to practice solving exercises.
(6) There was not sufficient feedback on my solutions for the assignments.
(7) There was not sufficient feedback on my solutions to the tutorial questions.
(8) I did not spend enough time studying, because I was busy with other modules.
(9) I did not spend enough time studying for other reasons.

Question 3. What was your familiarity with algorithmic techniques at the beginning of the module?

Please select the most appropriate option from the ones below.

(1) I had been taught algorithms in my undergraduate studies in multiple modules and I already had a good understanding.
(2) I had been taught algorithms in my undergraduate studies in one module and already had a good understanding.
(3) I had been taught algorithms in one ore multiple modules in my undergraduate studies, but did not have a good enough understanding.
(4) I had not been taught algorithms in my undergraduate studies.

Question 4. The first part of the module concerned basic algorithmic techniques, such as sorting, searching, graph algorithms, greedy algorithms and maximum flow algorithms. Which of the following options best describes your experience with this part of the module?

(1) I already was familiar with these topics and I would rather not have seen them in a module about Advanced Algorithms.
(2) I already was familiar with these topics, but I needed a refreshing, so it was good that they were part of the curriculum.
(3) I did not have much or any experience with these topics, so it was quite important that these were part of the curriculum.

Question 5. The latter part of the module concerned advanced algorithmic techniques, such as approximation algorithms, NP-completeness, linear programming and Online Algorithms. Which of the following options best describes your experience with this part of the module?

(1) I was already familiar with some of these topics and they were easy to follow.
(2) I was already familiar with some of these topics but still there were parts that were hard to follow.
(3) I was not familiar with most of these topics, but I could still follow.
(4) I was not familiar with most of those topics and they were quite hard to follow.

Question 6. Regarding the most advanced topics of the module, which of the following options best describes your experience? You may choose as many options as you think are appropriate.

(1) The exposition was appropriate, and I could follow them by following the lectures and studying.
(2) The material was too advanced and it was hard to follow.
(3) The material was not exposed in enough detail, which made it hard to follow.
(4) I would have rather spent more time on these topics rather than the more basic ones, to have a chance to understand them better.
(5) There were not enough practice exercises to allow me to truly familiarise myself with these topics.
(6) I found questions about these topics in the exam very challenging.

Question 7. Compared with your previous experience in your studies, which of the following options best described you familiarity with developing formal and mathematical proofs before the start of the module?

(1) I had been taught how to develop mathematical and formal proofs in multiple modules in the past.
(2) I had limited experience with developing mathematical and formal proofs.
(3) I had no experience in developing mathematical and formal proofs.

Question 8. After the completion of the module, which of the next options best describes your familiarity with developing mathematical and formal proofs?

(1) I now feel confident in my ability to develop mathematical and formal proofs.
(2) I have learned several techniques for developing mathematical and formal proofs, but I am not entirely confident that I can use them.
(3) I have not learned much more on developing mathematical and formal proofs compared to where I started.
(4) I am still finding it very hard to understand the principles of developing mathematical and formal proofs.

Question 9. The only "technical" question of the survey: You would like to decide whether the competitive ratio of an algorithm is at most 2 or not. Which of the following are proof and which are evidence?

(1) An argument using proof by contradiction.
(2) An example where the competitive ratio is larger than 2.
(3) A computer program that inputs instances of the problem and runs the algorithm to compute the outcome.
(4) You have tried to come up with examples where the competitive ratio is larger than 2 for multiple days, and you haven't been able to do that. Is this evidence or proof that the competitive ratio is indeed 2.

Question 10. According to you experience, would you say that the quality of study at the University of Liverpool is higher than that of the University where you completed your undergraduate studies?

Question 11. According to you experience, would you say that the difficulty of the modules at the University of Liverpool is higher than that of the University where you completed your undergraduate studies?

Question 12. Is there anything else that you would like to add?
If your answers contains and information that could potentially reveal information about your identity, this information will be altered carefully to prevent this from happening in the final report, or will not be used for th report at all.

B STUDENT INTERVIEW QUESTIONS

Q1. Did you have a chance to read the consent form? Do you have any questions regarding how the information from this interview will be used?

Q2. What subject did you study for your undergraduate degree? (computer science, informatics, engineering etc).

Q3. Which university did you do your undergraduate degree at?

Q4. How would you rank this university compared to other universities in the same country? Is it top 1%, top 5%, top 25%, top 50% or in the bottom 50%?

Q5. Compared to the University of Liverpool, would you rank your former university higher, lower, or about the same?

Q6. Compared to your previous university, would you say that the level of difficulty or workload in the University of Liverpool was higher, lower or about the same?

Q7. Did you study modules that were theoretically/mathematically oriented in your previous university? How many roughly? Could you provide some examples?

Q8. Compared to your previous university, would you say that in the University of Liverpool there was more/less/about the same emphasis on theory?

Q9. Before entering the MSc course at the University of Liverpool, would you say that you had familiarity with concepts and techniques related to mathematical proofs and formal arguments? Would you say that was sufficient or limited?

Q10. Was this the first MSc course that you did? Did you start this directly after your undergraduate studies or was there a gap?

Q11. How many theoretically oriented modules did you follow at the University of Liverpool. Which ones?

Q12. Besides COMP523 – Advanced Algorithmic Techniques, what did you find particularly challenging about those modules?

Q13. Did you find COMP523 more/less/about the same challenging than those other modules?

Q14. (Depending on the answer to Q13), what made COMP523 more/less challenging compared to those modules?

Q15. The first part of the COMP523 curriculum contained material that would normally be covered in an undergraduate algorithms module. Were you already familiar with those topics or not?

Q16. If you were already familiar with those topics, did you find it relatively easy or relatively hard to follow the material regarding them? Would you attribute that to having learned about them differently in your undergraduate studies? What was different?

Q17. Regarding the more advanced parts of the module (approximations algorithms, NP-completeness, linear programming, online algorithms). Which of those concepts were you familiar with from your undergraduate studies or from elsewhere? Were you also following COMP557 – Optimisation?

Q18. COMP523 covered topics like linear programming and NP-completeness briefly because they are assumed to be already known. If you were already familiar with those concepts, did you think that the length and detail of the exposition in COMP523 was appropriate? If you were not sufficiently familiar with those topics, would you instead be interested in learning about them in separate modules as part of the same programme?

Q19. In COMP523 you were asked to develop algorithms and proofs of correctness and efficiency. What did you find most challenging about this task? How would you compare it to e.g., being asked to code an algorithm in some programming language and test it on a set of inputs?

Q20. When attempting to solve tutorial exercises or the assignments for the module, what was your process? How did you decide which technique to use over another? Did you find that challenging?

Q21. When you submitted your solutions, how confident were you that they were correct? If they ended up being incorrect, were you already aware of the errors you had made or at least suspected them? More generally, when looking at a proof, how confident are you that you will be able to detect if there are arguments missing or if arguments are incorrect?

Q22. How much time did you spend trying to solve exercises on your own (e.g., the tutorial exercises)? Would you have spent more time on this if there were more opportunities for feedback? Generally, do you think that more exercise-solving sessions would help?

Q23. How confident are you that you can use the skills you acquired in the COMP523 module in the future, e.g., your work?

Q24. Having the experience of having been through the module, if you were to repeat it, what would you have done differently?

Q25. Is there anything else that you would like to add?

C MODULE COORDINATOR INTERVIEW QUESTIONS

Q1. Did you have a chance to read the consent form and do you have any questions about how the information will be used?

Q2. Which module(s) are you teaching at the University of Liverpool? Is it an MSc or undergraduate module? What was the background of students in your module?

Q3. How many times have you taught this module?

Q4. Can you talk a little bit about the contents of the module, especially the parts that are more theoretical / mathematical?

Q5. When explaining theoretical concepts, arguments and proofs, did you find that students found them particularly challenging?

Q6. Did you find that students find it more difficult to develop a proof for a statement, rather than write a piece of code, e.g., for an algorithm?

Q7. Where the assignments for your module theoretically-oriented, coding, or a combination of both? Did you detect any patterns in the mistakes made by the students?

Q8. Did you design any activities to help the students understand the theoretical parts of the module? Is there something that you found particularly useful in that regard?

Q9. How did the students do overall, in your module, compared to other modules?

Q10. Did you see any patterns correlating the background of students with their performance (e.g., undergraduate vs MSc students, if you have taught modules of a similar nature to both)?

Q11. What would you say was the biggest issue / challenge with your module? What are you planning to do to correct it, or what have you done to correct it already, from one year to the next?

Q12. Do you think that changes in the programme level are needed to address some of the issues, in particular with regard to the challenges faced by students in theory-oriented modules?

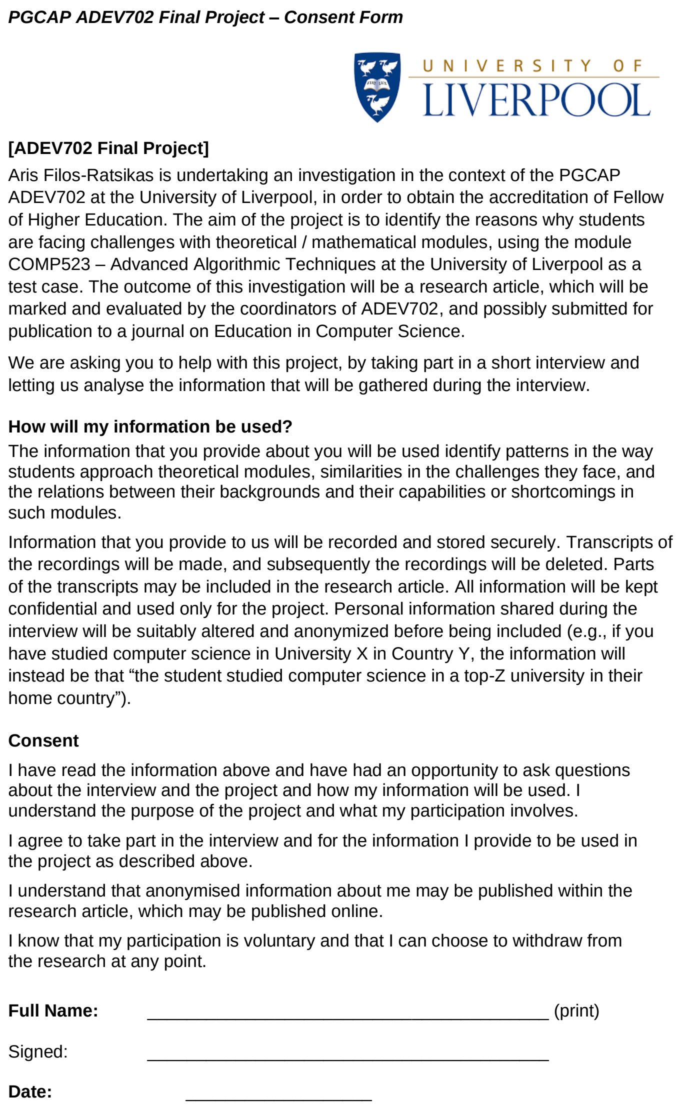
Q13. Is there anything else that you would like to add?

D ETHICS OF DATA ACQUISITION

As we mentioned in Section 2.2.4, the data acquisition process adhered to all the ethics requirements and guidelines from the University of Liverpool and the standard of research involving data from human participants. In particular, for the student survey, we included the following text at the beginning of the survey:

> This is a survey related to COMP523 - Advanced Algorithmic Techniques, targeted to students which participated in the module during the 2020/21 academic year. This survey is being carried out in the context of the Postgraduate Development Academic Practice (PGCAP) (see here: https://www.liverpool.ac.uk/eddev/supporting-teaching/pgcap/ (Links to an external site.)), module ADEV702. The data extracted from this survey will be used for statistical analysis for the Written Submission assignment of the module, which will be marked by the coordinators of the module and possibly will be submitted for publication in a journal in Education in Computer Science.
> The survey is entirely anonymous. It does not contain any questions that might reveal the identity of the participants. Aggregate statistics will be gathered from the results of the survey, and then all individual answers will be deleted.

For the student interviews, the students signed the consent form that is shown in Figure 5. For the module coordinator interview, the coordinator signed the consent form shown in Figure 6.



PGCAP ADEV702 Final Project – Consent Form

UNIVERSITY OF LIVERPOOL

[ADEV702 Final Project]

Aris Filos-Ratsikas is undertaking an investigation in the context of the PGCAP ADEV702 at the University of Liverpool, in order to obtain the accreditation of Fellow of Higher Education. The aim of the project is to identify the reasons why students are facing challenges with theoretical / mathematical modules, using the module COMP523 – Advanced Algorithmic Techniques at the University of Liverpool as a test case. The outcome of this investigation will be a research article, which will be marked and evaluated by the coordinators of ADEV702, and possibly submitted for publication to a journal on Education in Computer Science.

We are asking you to help with this project, by taking part in a short interview and letting us analyse the information that will be gathered during the interview.

How will my information be used?

The information that you provide about you will be used identify patterns in the way students approach theoretical modules, similarities in the challenges they face, and the relations between their backgrounds and their capabilities or shortcomings in such modules.

Information that you provide to us will be recorded and stored securely. Transcripts of the recordings will be made, and subsequently the recordings will be deleted. Parts of the transcripts may be included in the research article. All information will be kept confidential and used only for the project. Personal information shared during the interview will be suitably altered and anonymized before being included (e.g., if you have studied computer science in University X in Country Y, the information will instead be that “the student studied computer science in a top-Z university in their home country”).

Consent

I have read the information above and have had an opportunity to ask questions about the interview and the project and how my information will be used. I understand the purpose of the project and what my participation involves.

I agree to take part in the interview and for the information I provide to be used in the project as described above.

I understand that anonymised information about me may be published within the research article, which may be published online.

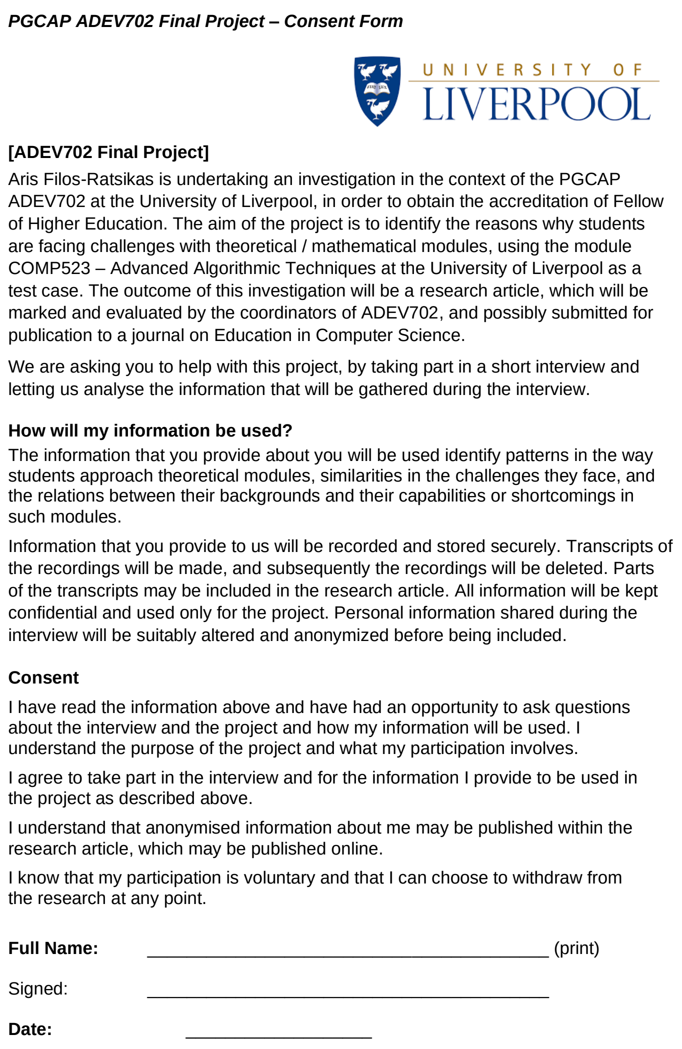
I know that my participation is voluntary and that I can choose to withdraw from the research at any point.

Full Name: ____ (print)

Signed: ____

Date: ____________________

Fig. 5. The student consent form.



PGCAP ADEV702 Final Project – Consent Form

UNIVERSITY OF LIVERPOOL

[ADEV702 Final Project]

Aris Filos-Ratsikas is undertaking an investigation in the context of the PGCAP ADEV702 at the University of Liverpool, in order to obtain the accreditation of Fellow of Higher Education. The aim of the project is to identify the reasons why students are facing challenges with theoretical / mathematical modules, using the module COMP523 – Advanced Algorithmic Techniques at the University of Liverpool as a test case. The outcome of this investigation will be a research article, which will be marked and evaluated by the coordinators of ADEV702, and possibly submitted for publication to a journal on Education in Computer Science.

We are asking you to help with this project, by taking part in a short interview and letting us analyse the information that will be gathered during the interview.

How will my information be used?

The information that you provide about you will be used identify patterns in the way students approach theoretical modules, similarities in the challenges they face, and the relations between their backgrounds and their capabilities or shortcomings in such modules.

Information that you provide to us will be recorded and stored securely. Transcripts of the recordings will be made, and subsequently the recordings will be deleted. Parts of the transcripts may be included in the research article. All information will be kept confidential and used only for the project. Personal information shared during the interview will be suitably altered and anonymized before being included.

Consent

I have read the information above and have had an opportunity to ask questions about the interview and the project and how my information will be used. I understand the purpose of the project and what my participation involves.

I agree to take part in the interview and for the information I provide to be used in the project as described above.

I understand that anonymised information about me may be published within the research article, which may be published online.

I know that my participation is voluntary and that I can choose to withdraw from the research at any point.

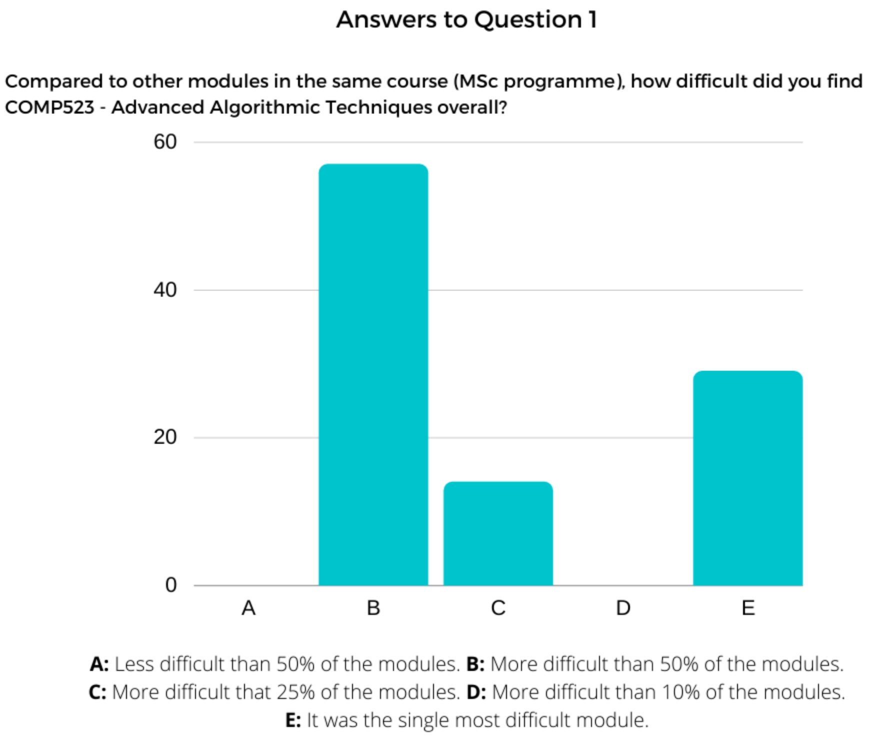
Full Name: ____ (print)

Signed: ____

Date: ____________________

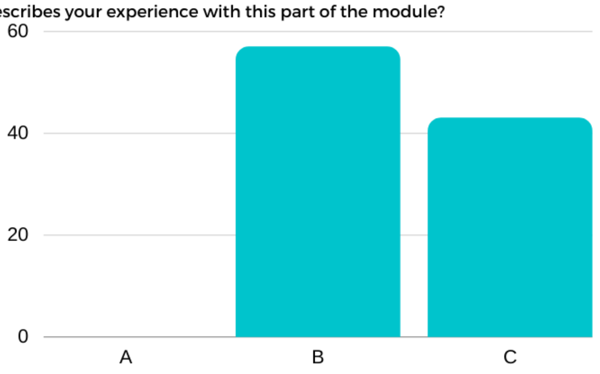
Fig. 6. The module coordinator consent form.

E STUDENT SURVEY RESULTS



Answers to Question 4

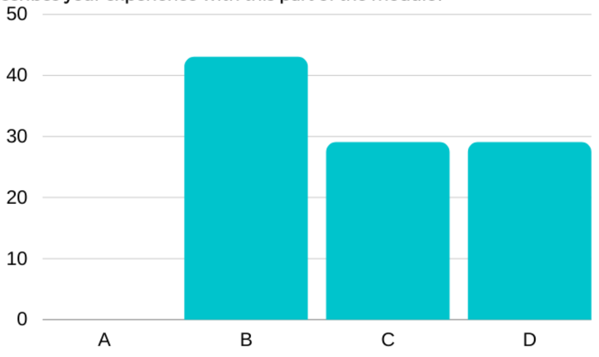
The first part of the module concerned basic algorithmic techniques, such as sorting, searching, graph algorithms, greedy algorithms and maximum flow algorithms. Which of the following options best describes your experience with this part of the module?



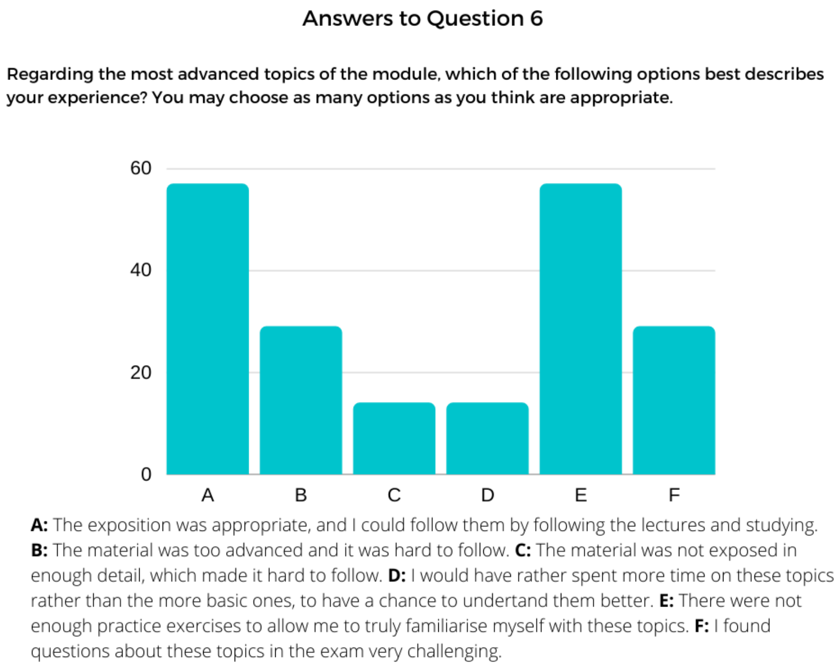
A: I already was familiar with these topics and I would rather not have seen them in a module about Advanced Algorithms. **B:** I already was familiar with these topics, but I needed a refreshing, so it was good that they were part of the curriculum. **C:** I did not have much or any experience with these topics, so it was quite important that these were part of the curriculum.

Answers to Question 5

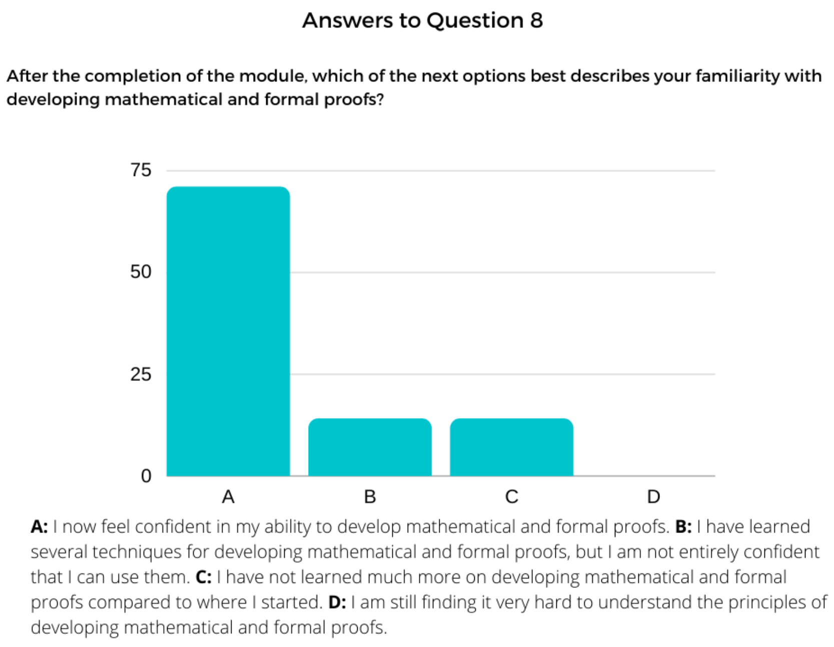
The latter part of the module concerned advanced algorithmic techniques, such as approximation algorithms, NP-completeness, Linear Programming and Online Algorithms. Which of the following options best describes your experience with this part of the module?



A: I was already familiar with some of these topics and they were easy to follow. **B:** I was already familiar with some of these topics but still there were parts that were hard to follow. **C:** I was not familiar with most of these topics, but I could still follow. **D:** I was not familiar with most of those topics and they were quite hard to follow.



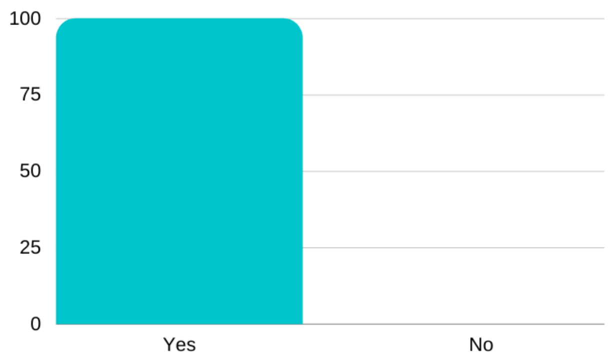
A: The exposition was appropriate, and I could follow them by following the lectures and studying. **B:** The material was too advanced and it was hard to follow. **C:** The material was not exposed in enough detail, which made it hard to follow. **D:** I would have rather spent more time on these topics rather than the more basic ones, to have a chance to undertand them better. **E:** There were not enough practice exercises to allow me to truly familiarise myself with these topics. **F:** I found questions about these topics in the exam very challenging.



A: I now feel confident in my ability to develop mathematical and formal proofs. **B:** I have learned several techniques for developing mathematical and formal proofs, but I am not entirely confident that I can use them. **C:** I have not learned much more on developing mathematical and formal proofs compared to where I started. **D:** I am still finding it very hard to understand the principles of developing mathematical and formal proofs.

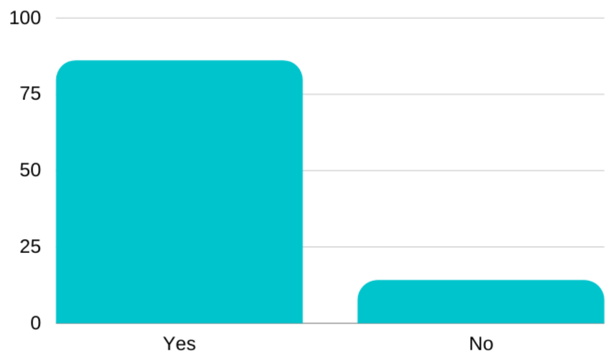
Answers to Question 10

According to you experience, would you say that the quality of study at the University of Liverpool is higher than that of the University where you completed your undergraduate studies (select ``yes'' if you studied at the University of Liverpool for your undergraduate studies).



Answers to Question 11

According to you experience, would you say that the difficulty of the modules at the University of Liverpool is higher than that of the University where you completed your undergraduate studies?


Yes
No