Evaluation of the Quality of Exercises for the Data Structures' eTextbook and Find the Difficult Topics Using Item Response Theory and Logged Data Analysis

Ahmed Abd Elrahman, Taysir Hassan A Soliman, Mohammed F. Farghally and Ahmed I. Taloba

Information System Department, Faculty of Computers and Information, Assiut University, Egypt

ahmedabdo@aun.edu.eg, taysirhs@aun.edu.eg, mfseddik@aun.edu.eg, Taloba@aun.edu.eg

Abstract

The growing dependence on eTextbooks and Massive Open Online Courses (MOOCs) has led to an increase in the amount of students' learning data. By carefully analyzing this data, educators can identify difficult exercises, and evaluate the quality of the exercises when teaching a particular topic. In this study, an analysis of log data from the use of a semester of the OpenDSA eTextbook was offered to identify the most difficult data structure course exercises and to evaluate the quality of the course exercises. Our study is based on analyzing students' responses to the course exercises. To identify the difficult exercises, we applied two different approaches, the first of which involved analyzing student responses to exercises using item response theory (IRT) analysis and a latent trait model (LTM) technique, and the second involved determining which exercises were more difficult based on how students interacted with them. We computed different measures for every exercise such that difficulty level, trial and error, and hint ratio. We generated an item characteristics curve, item information curve, and test information function for each exercise. To evaluate the quality of the exercises, we applied the IRT analysis to the students' responses to the exercises and, we computed the difficulty and discrimination index for each exercise. We classified whether the exercise is good or poor based on these two measures. Our findings showed that the exercises that related to algorithm analysis topics represented most of the difficult exercises that students struggle with, and there existing six exercises out of 56 exercises are classified as poor exercises which could be rejected or improved. Some of these poor exercises do not differentiate between students with different abilities; the others give preference to low-ability students to answer these exercises over high-ability students.

Keywords: interactive learning, Item response theory, eTextbooks, Item characteristics curves, Test Information Functions, Item Information Curve, data structures and algorithms.

1. Introduction

The rising usage of interactive online course materials at all levels of education, including online eTextbooks, Massive Open Online Courses (MOOCs), and practice platforms like Khan Academy and Code Academy has spread especially with the spread of
COVID-19 worldwide [1, 2, 3 and 4]. In order to reduce its spread in educational institutions, most educational institutions have resorted to teaching their courses via online platforms.

During the teaching of any online course, the interaction between students and educators takes place very little. It could be challenging for educators to know parts that students suffer from, as well as to assess the quality of exercises due to the lack of interaction with students. So knowing which topics students struggle with and attempting to improve or develop new methods to present these topics may be an essential step in enhancing the educational process and boosting the quality of the educational process. When students struggle with some issues in a course and no one strives to treat and simplify these topics. It is possible that they will drop out of the course and they will not finish studying the course resulting in failure in the educational process [2]. Knowing what topics students find difficult helps instructors to better allocate course resources.

Based on the interactions that students made in the OpenDSA eTextbook system [3, 5], we present techniques for automatically determining the most difficult topics for students. The topics students struggle with the most can be detected by experienced instructors, but this may frequently takes a long time and effort, and it does not give a real-time analysis that an intelligent tutoring system (ITS) would benefit from [6]. Automated measures can be beneficial for a set of reasons such that Instructors must devote a significant amount of time and effort to identifying key topics and they can assist in the discovery, confirmation, and quantification of relationships, as well as bring new insights that even experienced instructors may miss [6].

The topic of our study is a data structure and algorithms course (CS2). There are two objectives of our study the first one is identifying difficult course exercises, to achieve it, we applied two different approaches, the first one is IRT theory and an LTM technique for analyzing student responses to exercises. LTM assumes that specific traits or characteristics can predict test performance [7]. IRT provides a model-based association between the responses of the item and the characteristics of the test [8]. The second approach involved analyzing how students interacted with exercises to see which ones were more challenging. We looked at how often students guessed, and how often they used hints. Based on the finding, we found that the topics related to algorithm analysis are considered the most difficult topics which students struggle with.

The second objective is the evaluation of the quality and efficiency of exercises in the CS2 course. To achieve it, we also applied IRT. We computed the item difficulty and item discrimination for each exercise to evaluate their quality and to classify whether the exercise is good or poor. Based on our finding obtained, we found that six out of 56 exercises were classified as poor exercises that could be improved or rejected, some of these exercises give preference to low-ability students to answer to it over high-level students, and the other exercises don’t differentiate among students with different abilities. These poor exercises are related to Binary Tree Traversals, Binary Search Tree, Selection Sort Analysis, Array-Based List, Sorting Terminology, and Algorithm Analysis topics.
OpenDSA was used as the main eTextbook to teach CS2 course in a large public research institution [1]. In OpenDSA, a module represents a single topic or portion of a typical lecture, such as a single sorting algorithm and it is considered the most elementary functional unit for OpenDSA materials [2]. Each module in the OpenDSA content is focused on a different topic, such as selection sort or insertion sort. Each module is a full instructional unit and often includes AVs, interactive assessment exercises with automated feedback, and text that is of textbook quality. Chapters may be created out of modules, much like in conventional paper books. OpenDSA content is browser and device independent due to it built using of HTML5 and JavaScript. The JavaScript Algorithm Visualization (JSAV) module [9] was used to build AVs.

There is a range of various exercises in each module. One of these exercises requires the student to manipulate a data structure in order to show the effects of an algorithm on it. These are known as "Proficiency Exercises" (PE). PE exercises were developed and utilized for the first time in the TRAKLA2 system [10]. The other type of exercise is the Simple questions, which include various types of system questions such as true/false, multiple-choice, and short-answer questions. OpenDSA made utilized the exercise framework from Khan Academy (KA) [11] to save and present Simple questions.

There are various studies that applied the OpenDSA eTextbook in teaching the data structures and algorithm course, such that, in [2] To predict student performance based on their interactions with the eTextbook, a predictive model was created. In [3] a study is conducted to Explore students learning behavior.

This paper is organized as follows. In Section2, previous work is reviewed. Section 3 presents the experimental analysis of exercises with their results. Section 4 discusses the evaluation of exercise quality. Section 5 presents the Conclusions and Future Work.

2. Related Work

In [12], the responses of 372 students who registered in one first-year undergraduate course were utilized to evaluate the quality of 100 MCQs written by an instructor that was used in an undergraduate midterm and final exam. In order to compute item difficulty, discrimination, and chance properties they applied Classical test theory and IRT analysis models. The two-Parameter logistic (2PL) model consistently had the best fit to the data, they discovered. According to the analyses, higher education institutions need to guarantee that MCQs are evaluated before student grading decisions are made.

In an introductory programming course, IRT was applied to assess students' coding ability [13]. They developed a 1PL Rasch model using the coding scores of the students. Their findings revealed that students with prior knowledge performed statistically much better than students with no prior knowledge.
In order to analyze the questions for the midterm exam for an introductory computer science course, the authors of [14] utilized IRT. The purpose of this study was to study questions’ item characteristic curves in order to enhance the assessment for future semesters.

The authors applied IRT for problem selection and recommendation in ITS. To automatically select problems, the authors created a model using a combination of collaborative filtering and IRT. [15].

We are aware of attempts to identify challenging subjects in CS2 courses, as most of them has focused on introductory courses [6, 16, 17 and 18]. A questionnaire is created by [19] and sent to computer science educators to ask them to mention which topics they consider critical to learn, and which topics they believe are difficult to learn and teach. The finding of this questionnaire showed that the most challenging topics to learn related to Pointers, recursion, and object-oriented such that polymorphism and parameter passing. On the other hand Recursion, pointers, error handling, algorithms, and polymorphism were the most difficult topics to teach. Many of these topics were taught in the CS2 course.

3. Experimental Analysis

Students make many interactions during their dealing with the eTextbook, every student interaction represents a log, and all student logs are stored in the OpenDSA system. OpenDSA contains different types of interactions. Interactions are divided into two types the first one is only interactions with the eTextbook itself, such as loading a page, reloading a page, clicking on a link to go somewhere else, or viewing slideshows. The second type is the interactions with all types of eTextbook exercises such that attempts to answer any exercise, submit an answer, or request a hint. This study focused more on the second type. In [2] a more description of interactions and exercise types.

The amount of questions in each exercise varies. Any exercise in any module may be interacted with by any student at any time throughout the study period. Each question has a Submit button on it. If the student answers the question correctly, he clicks on it; however, if he doesn’t, his attempt will only be counted for this one question. Every time a student interacts with a question, he has the opportunity to answer it whether correctly or incorrectly. In this case, a grade is given to him. It is possible for the student to attempt only, in this situation, no points will be awarded. In any case, he may request a hint.

In this work, during the fall of 2020, we analyzed data of students who were enrolled in a CS2 course at a large public research institution. There are about 303,800 logs that represent the interactions of students with the eTextbook. These logs contain the name and description of the action, the time of the interactions, and which module the student executed the interactions on it.

As for the interactions of students with the exercises there exist about 200,000 logs. every log consists of the features mentioned in Table I.
Table I. The features for exercises interactions

| Feature Name     | Description                                                                 |
|------------------|-----------------------------------------------------------------------------|
| time_done        | Time in which a student interacted with a question.                         |
| time_taken       | Total seconds in which a student finished interacting with a question.      |
| count_hints      | Total count of hints that the student requested when interacting with a question. |
| count_attempts   | Total counts of attempts for student attempts to a question.                |
| hint_used        | Was the student using a hint in a question (yes, no)?                      |
| request_type     | The type of request to a question (attempt or hint).                        |
| correct          | "1" when a student answers a question correctly, "0" when a student answers a question wrong or requests for an attempt to a question. |

In [6], different measures were applied in order to determine the difficult topics in a CS3 course. These measures are correct attempt ratio (r), difficulty level (dl), students' hint usage (hr), and incorrect answer ratio (it). We computed these measures for every exercise. In the next subsections, we will talk about them.

3.1 Analysis of correct answer ratios

Our purpose is to give value to each OpenDSA exercise in terms of “relative difficulty”. Our aim is to find which exercises average-ability students find comparatively difficult. From this, we intend to learn which themes are the most difficult for students. As a result, maybe lead us to refocus our instructional efforts. In the OpenDSA, students can answer an exercise as many times as they want until they get it correct [6]. This will lead to that most students will gain full credit on almost all exercises. Among the vulnerabilities, as is typical with online courseware that most students exploit is that some exercises can be "gamed" [20]. In OpenDSA means that in order for students to get a question instance which easy to solve they reload the current page repeatedly. Due to the previous reasons, we have not counted the number of students who completed an exercise correctly. Instead, we employed other definitions for difficulty.

To measure the exercise difficulty, we looked at the ratio of correct to incorrect answers in OpenDSA exercises, such that the correct attempt ratio for difficult exercises should be lower. We utilized the fraction r to evaluate student performance [6].

$$r = \frac{\text{#of correct attempts}}{\text{#of total attempts}}$$  \hspace{1cm} (1)

We calculate the difficulty level (dl) for each exercise, such that

$$dl = \frac{1 - \frac{\sum_{i=1}^{n} r(i)}{n}}{n}$$  \hspace{1cm} (2)

The number of students is referred to as n, and the ratio of correct attempts is referred to as r.
In [21], similar measures was utilized to rate the difficulty of exercises, the authors utilized “the number of attempts it takes a student to figure out the right answer once making their initial mistake” as a metric of how difficult a logic exercises are. To determine the workout difficulty for an ITS, history of attempts conjunction with IRT was also applied in [22].

We categorized the exercises into categories based on their dl. The scores on the dl ranged from 0 to 0.85. The exercises in the fourth quartile (dl >0.34) most of them focused on algorithm analysis concepts (6 of a total of 11), one selection sort multiple choice question, one recursion programming exercise, and one binary tree practice question. Exercises of the third quartile (0.21 ≤ dl ≤ 0.34) covered mainly (7 of a total of 27) binary Tree Analysis. Six of these exercises covered Linked List concepts. Four of these exercises covered Sorting Analysis and three of these exercise covered the introduction chapter. The mechanics of algorithms or data structures were the focus of the exercises in the second quartile (0.12 ≤ dl ≤ 0.21) which covered (20 out of 24). The remaining four were exercises that covered lists, queues, and introduction chapter. The first quartile (dl < 0.21) covered all exercises that related to algorithms or data structures mechanics. The following two figures show the exercises with the highest and the lowest difficulty level values.

![Fig. 1. Exercise which has the highest difficulty level value [1].](image1)

![Fig. 2. Exercise which has the lowest difficulty level value [1].](image2)

We can see from the Figures 1 and 2 that the exercise in Fig. 1 belongs to the algorithm analysis module, whereas the exercise in Fig. 2 belongs to the binary search module. The number of questions in each exercise varies.
The outcomes from previous results lead us to the conclusion that most of the difficult exercises belong to the fourth quartile that has the largest difficulty level values and this quartile contains most of the exercises that belong to the algorithm analysis concept. And the other quartiles that have less difficulty level value contain exercises related to the mechanism of algorithms and data structures, these results showed that students had no difficulty completing tasks related to the behavior and mechanics of algorithms and data structures. They appear to be struggling with the algorithms and analysis concepts.

### 3.2 Using Hints and Guessing

Our "incorrect attempts" measure does not distinguish between utilizing a hint and giving an incorrect answer. As a result, we took a closer look at the various types of incorrect submissions for each exercise. We looked at how many hints were used in OpenDSA exercises and a trial-and-error technique was utilized to "guess" the answers. It is expected that more difficult exercises will show a higher hint rate and/or trial and error. To complete the exercise, the student must obtain a specific number correct (usually five) [6]. When a student submits an incorrect response, a point has been deducted from their credit toward this requirement. To avoid guessing, Students can also make use of one or more hints to aid in understanding the answer to the question. The attempt in this situation is not assessed.

The hint ratio ($hr$) was calculated for each exercise to analyze exercises based on students' hint usage.

$$hr = \frac{\text{# of hints used}}{\text{# of total attempts + # of hints used}}$$  \hspace{1cm} (3)

We divided the number of hints used into (the number of total attempts and the number of hints used) because it is possible that the number of hints is greater than the number of attempts. For instance, it is possible when a student solves a specific exercise, he may have used two hints, but he has only attempted once, in this case, the $hr$ value will become $>1$.

We computed the incorrect ratio ($ir$) for each exercise in order to analyze the exercises based on the rate of trial-and-error ($ir$).

$$ir = \frac{\text{# of incorrect answers}}{\text{# of total attempts}}$$  \hspace{1cm} (4)

**Table II** contains eleven exercises with high percentages of hints or incorrect answers ratios. They related to topics covering Algorithm Analysis, Queues Analysis, Linked List and runtime analysis of bubble sort, and insertion sort. We observed that most exercises that had a low incorrect answer and a low hint ratio belong to Binary Trees arrays, introduction chapter, object-oriented programming, and lists. The reason for this is that most students are familiar with these concepts from previous courses. When used as a measure of
exercise difficulty, a high rate of hint use is used, Algorithm analysis, Linked List Analysis, Queues Analysis, and Sorting Analysis exercises looked to be more "difficult" than others. The reason for this is that students may be not familiar with these concepts or they deal with these concepts for the first time in the CS2 course.

Table II: ir and hr for difficult exercises

| Exercise                  | hr  | ir  | Topic                          |
|---------------------------|-----|-----|--------------------------------|
| ListOverheadp             | 0.66| 0.62| ListOverheadAnalysis           |
| LqueueDequeuePROp         | 0.40| 0.61| Linked Dequeue Analysis        |
| BubsortPROp               | 0.37| 0.33| Bubble Sort Analysis           |
| AqueueEnqueuePROp         | 0.35| 0.73| Array-Based Queue Enqueue      |
| GrowthRatesPROp           | 0.24| 0.94| Growth Rates                   |
| LlistRemovePROp           | 0.34| 0.68| Doubly Linked Lists            |
| InnsortPROp               | 0.33| 0.38| Insertion Sort Analysis        |
| AqueueDequeuePROp         | 0.27| 0.56| Array-Based Queue Dequeue      |
| AnlsIntroMCQtmcmplxp      | 0.21| 0.68| Algorithm Analysis             |
| LLMCQchngcrsrp            | 0.20| 0.30| Linear structure Analysis      |
| ComparingAlgorithmsSumm   | 0.19| 0.82| Comparing Algorithms           |

3.4 IRT analysis

IRT [23] examines test behavior at the item level and offers feedback on the relative difficulty of various questions. Many IRT models have been developed on the assumption that each response has a value of 0 or 1.

We dichotomize the answers in order to perform IRT analysis. For $r \geq 0.75$, we gave 1 point and for $r < 0.75$, we gave 0 point. Each chapter was analyzed independently. In order to build a two-Parameter logistic model (2PL) for our study, we utilized R software specifically (ltm package) and built the 2PL model for our study.

The equation for the two-parameter logistic model (2PL) is given in equation below: -

$$p(\theta) = \frac{1}{1 + e^{-a(\theta-b)}}$$

Where: $e$ is the constant 2.718, $(b)$ is the difficulty parameter, $(a)$ is the discrimination parameter and $\theta$ is an ability level [23]. The logistic equation when graphed produce plots
that is called item characteristic curves. We will discuss the two parameter \((a \text{ and } b)\) with brief details in the next sections.

For each OpenDSA exercise, we generated the Item Characteristic Curves (ICC), Item Information Curves (IIC) and Test Information Curves (TIF). Each curve's x-axis depicts the students' ability from -4 to 4. \(x = 0\) denotes average ability. Given a student's ability, ICC shows the likelihood of a score of 1 and it allows us to characterize qualitatively whether these exercises are efficient or not. The IIC demonstrates how much information each exercise may inform us regarding the ability of students. The main purpose of the TIF is to determine the reliability of the overall exam at differentiating students' different ability. Students with above-average ability would be better distinguished by harder items. Easy items, on the other hand, would better differentiate students with below-average ability. On an ICC graph, the likelihood of receiving a score of 1 for students with average ability may be shown. Difficult items will obtain a \(P_i (0) < 0.5\).

Inequalities between students with the below-average ability and students with average and above-average ability \((0 \geq 0)\) are shown in the curves for the easier exercises. As shown in Fig. 3. This means that these exercises are reasonably effective at differentiating between those who learned in the course versus those who did not. The TIF graph determines the test' overall performance.

Fig. 5 depicts the results for three of the more difficult exercise. A student with average ability has a less than 0.5 likelihood of receiving a score of 1. we may infer that these exercises differentiate students with average ability from students with above-average ability, but they do not differentiate very low level students from average students. Low abilities are students with low performance who need better performance.

**Introduction and Abstract Data Types exercises**: As illustrated in Fig. 3, almost all curves represent easy items because the likelihood of answering questions correctly for low-ability examinees is high and approximately reaches 1 for high-ability examinees. So these exercises were previously familiar to students, since these were covered in prerequisite courses. They helped us in providing information about students with below-average abilities \((x<0)\).

![Item Characteristic Curves](image)

**Fig. 3.** Abstract Data Types ICC.
**List Interface & Array based Lists exercises:** As illustrated in Fig. 4, all curves represent easy items because the likelihood of answering questions correctly for low-ability is high and approximately reaches 1 for high-ability examinees. All Exercises in this chapter are easy and all students familiar with these exercises. Through these exercises, we were able to learn more about students with below-average abilities ($x < 0$).

![Item Characteristic Curves](image)

**Fig. 4.** List Interface & Array based Lists ICC.

**Algorithm analysis chapter exercises:** As illustrated in Fig. 5 and Fig. 6, almost all curves represent difficult items because the likelihood of answering questions correctly for most of ability-scale is low and increase only when reached to high-ability levels. So Most students struggled with the exercises related to this chapter. As a result of these exercises, it aided us in giving information about students with above-average abilities.

![Item Characteristic Curves](image)

**Fig. 5.** Algorithm Analysis ICC.

![Item Information Curves](image)

**Fig. 6.** Algorithm Analysis IIC.
**Stacks exercises:** As illustrated in Fig. 7, Most of the Exercises in this chapter seem to be easy and most of the students are familiar with them. As a result, these exercises helped us in providing information about students who have below-average abilities (x < 0).

![Item Characteristic Curves](image1)

**Fig. 7.** Stack ICC.

**Recursion exercises:** For students, just three exercises looked to be difficult. These involved Forward flow tracing exercises, Recursion programming exercise: Subset-sum and Recursion programming exercise: Pascal triangle. As illustrated in Fig. 8, because of TIF curve maximum at ability < 0, it means these exercises helped us in providing information about students who have below-average abilities.

![Test Information Function](image2)

**Fig. 8.** Recursion TIF.

**Sorting exercises:** The chapter on sorting features the most exercises, all of which are of varying difficulty levels. More advanced sorting algorithms (Bubble sort, Insertion sort, and selection sort) appeared to give more information on students with above-average abilities(x > 0). As illustrated in Fig. 9, because at ability > 0, the TIF curve reaches maximum, it means these exercises appeared to give a decent range of exercises from simple to difficult, as well as helpful information for discriminating between students of various ability levels.

![Sorting Information Function](image3)
Fig. 9. Sorting TIF.

Queues exercises: As illustrated in Fig. 10, most of exercises seem to be easy and it’s clear that most of students don’t struggle with these exercises. Only one exercise seems to be difficult for students. this exercise is related to Array-Based Queue (Enqueue). we can see from Fig. 11, ability < 0, the TIF curve reaches maximum, it refers that these exercises helped us in providing information about students who have below-average abilities.

Fig. 10. Queues Item Characteristic Curves.

Fig. 11. Queues TIF.

Linked List exercises: The Linked List exercises have different difficulty levels. Exercises covering Linked List Remove, List element detection, and Linear Structure seemed to reveal more information about students with above-average ability (x > 0). In general, we can see from Fig. 12, at ability = 0 the TIF curve reaches maximum. So it appears that these exercises offer a fair variety of exercises ranging from simple to difficult, as well as useful information for distinguishing between students of various abilities.
Binary trees exercises: The Binary trees chapter has the most exercises. For students, as illustrated in Fig. 14, only two exercises seemed to be difficult. Those two exercises covering Preorder Traversal and Inorder Traversal. In general, as illustrated in Fig. 13 we mention that at ability < 0, the TIF curve reaches its maximum so these exercises helped us in providing information about students who have below-average abilities.
We can conclude that by using the measures ir, hr, dl, and IRT analysis that the exercises belonging to the algorithm analysis module is the most difficult exercises in the CS2 course followed by some exercises in other modules such that Binary Tree and Linked List. These modules may need some attention. While students seem to be familiar with the exercises related to behavior and mechanics of algorithms and data structures.

3.4 Pearson’s Correlation Coefficient

To find out the relationship and impact of the measurements (ir and hr) on determining the performance of the students during the CS2 course study. We calculated the relationship between the student’s grade in the final exam and hr, the student’s grade and the ir, and the relationship between hr and ir by using Pearson’s Product Moment Correlation. In order to achieve this; we computed the hr and ir for each student in all exercises.

The measurement of association, relationship, or correlation between two variables in order to determine whether they are positively or negatively related, or not related at all, is known as correlation. Two variables are related if the changes in one variable affect or influence the changes in the other variable [24]. The Pearson’s Correlation Coefficient (r) is defined as:

\[
r = \frac{\sum (AB) - n\bar{A}\bar{B}}{(n-1)\sigma_A\sigma_B}
\]

Where, \( r\) = Pearson’s Correlation coefficient, \( n\) = #of values, \( \sum AB\) Sum of the products of A and B, \( \bar{A} \) and \( \bar{B} \) are the means of A and B variables, \( \sigma_A\) is A variable standard deviation, and \( \sigma_B\) is the standard deviation of B variable. Table III shows Correlation values.

| Correlation          | r-value |
|----------------------|---------|
| Correlation(hr, ir)  | -0.245  |
| Correlation(ir, etest)| -0.78   |
| Correlation(hr, etest)| 0.26    |

According to the results shown in Table III, hr and ir are negatively correlated, as indicated by a negative correlation value of -0.245. In other words, a student with a high ir score is likely to have a low hr score, and vice versa. etest and ir are negatively correlated, as indicated by a negative correlation value of -0.78. In other words, a student with a high ir score is likely to have a low etest score, and vice versa. hr and etest are positively correlated, according to a positive correlation coefficient of 0.26. In other words, a student with a high hr score is likely to have a high etest score, and vice versa.
4. Evaluation of exercises quality

Such as we applied IRT analysis in determining the most difficult exercise, we applied it also in the Evaluation of exercises quality. We also applied (2 PL) model and the test item analysis is based upon item discrimination (a) and item difficulty (b). In [25] the (2PL) model was utilized to examine the quality of test items. We classified each exercise as poor or good exercise based upon its item discrimination and difficulty value. The next section describe the two terms.

4.1. (a) Parameter: Item discrimination

One of the characteristics of a good test item is that it will be answered correctly by high-ability students more often than lower-ability. The (a) parameter reflects how effectively an item can differentiate between examinees of various abilities. A high discrimination level means the item can tell the difference between individuals who have high and low abilities. [25]. While most test items will have a positive value, some items have negative discrimination. In such items, the probability of correct response decreases because the ability level increases from low to high. This tells that something is wrong with the item and it's a warning that the item needs some attention. For many of the item response theory, the worth of the discrimination index value is positive [23].

4.2. (b) Parameter: Item difficulty

The point where the S-shaped curve has the steepest slope is the (b) parameter, which denotes an item's difficulty. The greater an examinee's ability level must be to successfully answer an item, the harder the item is. Items with high b values are challenging, meaning that low-ability test takers are unlikely to successfully respond. Values of b larger than 1 indicate a challenging item. Easy items are those with low b values below -1 [25].

4.2. Results of the quality of the exercise

As we said earlier, we built a (2PL) model to assess the quality of the exercises; we dichotomize the answers in order to perform IRT analysis. For r ≥ 0.75, we gave 1 point and for r < 0.75, we gave 0 point, and we computed difficulty and discrimination index for each exercise and we used the two measures to classify the exercise as good or poor.

The difficulty index and discrimination index for each exercise are shown in Table V.

As shown in Table V, there exist some exercises that have negative discrimination or Very Low discrimination level this means that these exercises are poor and need to be improved. There exist some exercises that have low discrimination levels, these exercises seem to be good but they may need to improve. The other exercises seemed to be good exercises.

Table IV shows ranges of values used to describe an item’s discrimination level.
### Table IV. Levels for item discrimination parameter value [23]

| Verbal label | Range of values |
|--------------|-----------------|
| none         | <0              |
| Very low     | 0.01-0.34       |
| Low          | 0.35-0.64       |
| moderate     | 0.65-1.34       |
| High         | 1.35-1.69       |
| Very High    | >1.70           |

### Table V. Difficulty index and discrimination index for each CS2 course exercises

| Exercise          | Topic                                | Difficulty index | Difficulty Level | Discrimination index | Discrimination level | Interpretation                                      |
|-------------------|--------------------------------------|------------------|------------------|----------------------|----------------------|-----------------------------------------------------|
| IntroMCQgoalsp    | Data structures and algorithms       | -2.85            | Easy             | 0.5563               | Low                  | good exercise but possibly need to be improved      |
| CMdatatypetypeMCQintp | Abstract Data Types                  | -0.99            | Easy             | 0.8706               | moderate             | Good exercise                                      |
| ADTMCQsimpleType  | Abstract Data Types                  | -0.13            | Easy             | 0.5618               | Low                  | good exercise but possibly need to be improvement  |
| Data Structures&AlgorithmsS umm | Data structures and algorithms | -1.96            | Easy             | 0.3777               | Low                  | good exercise but possibly need to be improvement  |
| AlistInsertPROp   | Array-Based List (Insertion)         | -3.10            | Easy             | 0.6691               | moderate             | Good exercise                                      |
| AlistRemovePROp   | Array-Based List (Remove)            | 6.72             | Hard             | -0.4715              | None                 | poor exercise, need to be improved                 |
| ALCMQdelarb       | Array-based List (Practice questions)| -1.19            | Easy             | 0.9108               | moderate             | Good exercise                                      |
| ProblemAlgorithmProgramSumm | Algorithm Analysis(Problems,Algorithms) | -2.40            | Easy             | 0.6047               | Low                  | good exercise but possibly need to be improvement  |
| CompareGrowthp    | Comparing Growth Rates               | 2.55             | Hard             | 0.5975               | Low                  | good exercise but possibly need to be improvement  |
| GrowthRatesPROp   | Growth Rates                         | 9.63             | Hard             | 0.3385               | Low                  | good exercise but possibly need to be improvement  |
| AnsIntroMCQtnmcmplxp | Algorithm Analysis                  | 2.36             | Hard             | 1.0223               | moderate             | Good exercise                                      |
| AlgAnsMCQsqlsrchp | Algorithm Analysis                   | 2.26             | Hard             | 0.7628               | moderate             | Good exercise                                      |
| binarySearchPROp | Programming runtime(binary search)   | 8.02             | Hard             | -0.3379              | None                 | poor exercise, need to be improved                 |
| AnalProgramMCQthetap | Analyzing Code                       | 1.18             | Medium           | 1.2680               | High                 | Very good exercise                                 |
| Comparing Algorithms Summ | Comparing Algorithms | 1.88 | Medium | 0.67 | moderate | Good exercise |
|---------------------------|----------------------|------|--------|------|----------|----------------|
| AstackPushPROp           | Array-Based Stack (Push) | 0.037 | Medium | 0.6  | Low      | good exercise but possibly need to be improvement |
| AstackPopPROp            | Array-Based Stack (pop) | -2.21 | Easy   | 0.54 | Low      | good exercise but possibly need to be improvement |
| LstackPushPROp           | List-based Stack (Push) | -2.72 | Easy   | 0.48 | Low      | good exercise but possibly need to be improvement |
| LstackPopPROp            | List-based Stack (pop) | -4.60 | Easy   | 0.6  | Low      | good exercise but possibly need to be improvement |
| RectFIBcountp            | Recursion(Tracing practice) | -1.68 | Easy   | 0.64 | Low      | good exercise but possibly need to be improvement |
| FinderrortracingSumm     | Recursion(Tracing practice) | -1.05 | Easy   | 0.78 | moderate | Good exercise |
| RectMCQwritecommfixp     | Recursion(Tracing practice) | 0.53  | Medium | 0.40 | Low      | good exercise but possibly need to be improvement |
| ForwardFolowTracingSumm  | Recursion(Tracing practice) | 0.66  | Medium | 0.38 | Low      | good exercise but possibly need to be improvement |
| BackwardFolowTracingsum  | Recursion(Tracing practice) | 0.82  | Medium | 0.98 | moderate | Good exercise |
| RectFIBmysteryadp        | Recursion(Writing practice) | -0.60 | Easy   | 0.35 | Low      | good exercise but possibly need to be improvement |
| Harder QuestionsSumm     | Recursion(Tracing practice) | -1.59 | Easy   | 1.0465 | High | Very good exercise |
| SortIntroMCQ3bp          | Sorting Terminology and Notation | 1.11  | Medium | 0.5976 | low | good exercise but possibly need to be improvement |
| CompareTF&MCQ5p          | Sorting Terminology and Notation(comparing Records) | -2.24 | Easy   | 0.1614 | Very Low | poor exercise, need to be improved |
| InssortPROp              | Insertion Sort Analysis (proficiency) | -1.03 | Easy   | 0.7523 | moderate | Good exercise |
| InssortMCQcostap         | Insertion Sort Analysis | 0.21  | Medium | 0.6887 | moderate | Good exercise |
| BubsortPROp              | Bubble Sort | -1.41 | Easy   | 0.7045 | moderate | Good exercise |
| BubsortMCQcQuestinisp    | Bubble Sort Analysis | 0.40  | Medium | 1.4966 | High | Very Good exercise |
| SelSortPROp              | Selection Sort Analysis (Proficiency) | -34.98 | Easy   | 0.0496 | Very Low | poor exercise, need to be improved |
| SelsortMCQworstp         | Selection Sort Analysis | 1.72  | Medium | 0.7976 | moderate | Good exercise |
| Operation                         | Type                   | Complexity | Difficulty | Note                                      |
|----------------------------------|------------------------|------------|------------|-------------------------------------------|
| AqueueEnqueue                    | Array-Based Queue Enqueue | 2.16       | Medium     | 0.5925 Low | good exercise but possibly need to be improvement |
| AqueueDequeue                    | Array-Based Queue Dequeue | -1.26      | Easy       | 0.4780 Low | good exercise but possibly need to be improvement |
| LqueueEnqueue                    | Linked Queues Analysis  | -3.18      | Easy       | 0.5965 Low | good exercise but possibly need to be improvement |
| LqueueDequeue                    | Linked Dequeue Analysis | 0.048      | Easy       | 0.4709 Low | good exercise but possibly need to be improvement |
| StackQMCQqoprts                  | stack and queue Analysis | -0.66      | Easy       | 1.3778 High | Very good exercise |
| LlistInsertPRO                   | Linked List Insert      | -1.38      | Easy       | 0.4113 Low | good exercise but possibly need to be improvement |
| LlistRemovePRO                   | Doubly Linked Lists     | 0.84       | Medium     | 0.7978 moderate | Good exercise |
| ListOverheadp                    | List Overhead Analysis  | 0.27       | Medium     | 0.6133 Low | good exercise but possibly need to be improvement |
| ComparasionOfListEx              | Comparison of List Analysis | 0.75     | Medium     | 1.1492 moderate | Good exercise |
| ListMCQdelcurrp                  | List Element Implementations | -0.46      | Easy       | 1.1338 moderate | Good exercise |
| Linear Structure Summ            | Linear structure Analysis | -0.33      | Easy       | 1.1509 moderate | Good exercise |
| LLMCQchngcrsrp                   | Linear structure Analysis | -0.62      | Easy       | 1.0325 moderate | Good exercise |
| DInternMinFIBp                   | Binary Trees Analysis   | 2.61       | Hard       | 2.0348 Very High | Very Good exercise |
| Treerordering problems           | Binary Trees Traversal  | 0.68       | Medium     | 1.1507 moderate | Good exercise |
| btTravPreorderPRO                | Binary Tree Preorder Traversal | -3.57      | Easy       | 0.3508 Low | good exercise but possibly need to be improvement |
| btTravPostorderPRO               | Binary Tree PostOrder Traversals | -1.16      | Easy       | 0.5998 Low | good exercise but possibly need to be improvement |
| btTravlnorderPRO                 | Binary Tree Inorder Traversals | -0.07      | Easy       | 0.7067 moderate | Good exercise |
| BSTSummaryQuestions              | Binary Tree Traversals  | 2.20       | Hard       | -0.0303 None | poor exercise, need to be improved |
| BSTsearchPRO                     | Binary Search Trees (Search) | -2.17      | Easy       | 0.5877 Low | good exercise but possibly need to be improvement |
| BSTInsertPRO                     | Binary Search Trees (Insert) | -2.15      | Easy       | 0.7287 moderate | Good exercise |
| BSTremovePRO                     | Binary Search Trees (Remove) | -0.20      | Easy       | 0.3297 Very Low | poor exercise, need to be improved |
In the summary of previous results, according to the ranges in Table IV, there are three exercises that have a negative discrimination value, and these questions have a high difficulty value, this means that students with high levels have an advantage in answering these questions over students with low levels, but negative discrimination means that these exercises give preference to the student with a low level to solve these exercises, so we classified these exercises as poor exercises, these exercises maybe need to be improved or maybe needed to rephrase. The reason for the improvement is that these exercises give preference to the student with a low level to solve this exercise about the student with a high level, and There exist Three exercises that have a very low discrimination level and easy difficulty level, we classified these three exercises on the basis that they are poor exercises because these three exercises differentiate between students who have below-average abilities and, do not differentiate between different abilities levels students, so these exercises also maybe need to be improved or maybe needed to rephrase. The poor exercises related to Binary Tree Traversals, Binary Search Tree, Selection Sort Analysis, Array-Based List, Sorting Terminology, and Algorithm Analysis topics. There are 25 out of 56 exercises that have low discrimination levels they seem to be good exercises but possibly need to improve. The remaining exercises are good and they can separate students with high and low ability abilities. The next figures show these poor exercises.

**Fig.15.** AlistRemovePROp exercise [1].

**Fig.16.** BinarySearchPRO exercise [1].
Fig. 17. BSTSummaryQuestions exercise [1].

Fig. 18. BSTremovePRO exercise [1].

Fig. 19. SelSortPRO exercise [1].

Fig. 20. CompareTF&MCQ exercise [1].
Fig. 15 belongs to the Array-Based List, Fig. 16 for Algorithm Analysis, Fig. 17 for Binary Tree Traversals, Fig. 18 for Binary Search Tree, Fig. 19 for Selection Sort Analysis and Fig. 20 for Sorting Terminology. There are various numbers of questions in each exercise.

5. Conclusions and future work

With the spread of COVID-19 worldwide, online eTextbooks have become more prevalent, when teaching a particular course via eTextbook; it has become necessary to identify the difficult exercises and to evaluate the quality for course exercises. This may allow instructors and instructional material creators to concentrate their efforts on the most difficult topics. Our study focuses on CS2 course which was studied through eTextbook in a large public research institution during fall 2020. Our study is based on an analysis of students' responses to the CS2 course exercises.

Our study has more than one objective; the first is to identify the difficult exercises in the CS2 course. To achieve it, we analyzed every exercise in each module, and we applied two approaches the first one is IRT and LTM to analyze the interactions of students with exercises. While the second one is analyzing how students interact with exercises to know which of them is more difficult than the other. We built a 2PL model for our analysis, and the ICC, IIC, and TIF curves were generated. Our findings showed that students seem to be familiar with the exercises related to the behavior and the mechanics of algorithms and data structures. But on the other hand, they seem to not be familiar with the exercises related to algorithms analysis concepts and they have difficulty in dealing with these types of exercises.

The second goal of our study is to evaluate the quality of the exercises in the CS2 course. To achieve it; we applied IRT analysis and built a 2PL model, and we computed the difficulty and discrimination index for every exercise. We classified each exercise as poor or good based on these two metrics. The results showed that three workouts out of 56 had negative discrimination values and were deemed to be poor exercises. When solving these exercises, the student with a low ability is given precedence over the student with high ability. And there are three exercises that have low discrimination levels, and these exercises don’t differentiate between students with different abilities. These poor exercises maybe need to be improved or rephrased. The poor exercises covered topics including Binary Tree Traversals, Binary Search Tree, Selection Sort Analysis, Array-Based List, Sorting Terminology, and Algorithm Analysis topics. Finally, we applied Pearson’s Correlation to find out the effect of hr and ir on the student’s final exam degree, we found that hr and ir are negatively correlated to each other, and there exists a positive correlation between hr and student’s final exam grade, and there exists a negative correlation between ir and student’s final exam grade.

There are many interactions that each student makes with the OpenDSA eTextbook, a summary of interactions was explained in [2], so In the future. We will try to find the best sequence of the interactions so that the student gets the best benefit from the eTextbook.
References

[1] https://opendsa-server.cs.vt.edu/ODSA/Books/CS2/html/index.html.

[2] Elrahman, Ahmed Abd, et al. "A Predictive Model for Student Performance in Classrooms Using Student Interactions With an eTextbook." arXiv preprint arXiv: 2203.03713 (2022).

[3] E. Fouh, D. A. Breakiron, S. Hamouda, M. Farghally, and C. A. Shaffer. Exploring students learning behavior with an interactive eTextbook in computer science courses. Computers in Human Behavior, pages 478–485, December 2014.

[4] Baturay, Meltem Huri. "An overview of the world of MOOCs." Procedia-Social and Behavioral Sciences 174 (2015): 427-433.

[5] E. Fouh, V. Karavirta, D. A. Breakiron, S. Hamouda, S. Hall, T. L. Naps, and C. A. Shaffer. Design and architecture of an interactive eTextbook–The OpenDSA system. Science of Computer Programming, 88:22–40, 2014.

[6] Fouh, Eric, et al. "Investigating Difficult Topics in a Data Structures Course Using Item Response Theory and Logged Data Analysis." International Educational Data Mining Society (2016).

[7] R. K. Hambleton and L. L. Cook. Latent trait models and their use in the analysis of educational test data. J. of Educational Measurement, 14(2):75–96, 1977.

[8] F. Drasgow and C. L. Hulin. Item response theory, Handbook of industrial and organizational psychology, 1:577–636, 1990. 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE ’15, pages 51–56, 2015.

[9] KARAVIRTA, Ville; SHAFFER, Clifford A. JSAV: theJavaScript algorithm visualization library, In: Proceedings of the 18th ACM conference on Innovation and technology in computer science education , (2013) 159-164.

[10] L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, and P. Silvasti. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. Informatics in Education, 3(2):267–288, September 2004.

[11] (http://github.com/Khan/khan-exercises).

[12] Brown, Gavin TL, and Hasan HA Abdulnabi. "Evaluating the quality of higher education instructor-constructed multiple-choice tests: Impact on student grades." Frontiers in Education. Vol. 2. Frontiers, 2017.

[13] Berges, Marc, and Peter Hubwieser. "Evaluation of source code with item response theory." Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education. 2015.
[14] L. A. Sudol and C. Studer. Analyzing test items: Using item response theory to validate assessments. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE ’10, pages436–440, 2010.

[15] P. Jarušek and R. Pel’ anek. Analysis of a simple model of problem solving times. In S. Cerri, W. Clancey, G. Papadourakis, and K. Panourgias, editors, Intelligent Tutoring Systems, volume 7315 of LNCS, pages 379–388. Springer, 2012.

[16] N. Dale. Content and emphasis in CS1. SIGCSE Bulletin, 37(4):69–73, Dec. 2005.

[17] N. B. Dale. Most difficult topics in CS1: Results of an online survey of educators. SIGCSE Bulletin, 38(2):49–53, June 2006.

[18] K. Goldman, P. Gross, C. Heeren, G. L. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Setting the scope of concept inventories for introductory computing subjects. Transactions on Computing Education, 10(2):5:1–5:29, June 2010.

[19] P. Brusilovsky, J. Grady, M. Spring, and C.-H. Lee. What should be visualized: Faculty perception of priority topics for program visualization? SIGCSE Bulletin, 38(2), June 2006.

[20] R. Baker, A. Corbett, and K. Koedinger. Detecting student misuse of intelligent tutoring systems. In Proceedings of the 7th International Conference on Intelligent Tutoring Systems, pages 531–540, 2004.

[21] D. Barker-Plummer, R. Cox, and R. Dale. Student translations of natural language into logic: The grade grider corpus release 1.0. In Proceedings of the 4th international conference on educational data mining, 2011, pages 51–60.

[22] G. Ravi and S. Sosnovsky. Exercise difficulty calibration based on student log mining. In Proceedings of DAILE: Workshop on Data Analysis and Interpretation for Learning Environments, 2013.

[23] Baker, Frank B. The basics of item response theory. For full text: http://ericae.net/irt/baker, 2001.

[24] Obilor, Esezi Isaac, and Eric Chikweru Amadi. "Test for significance of Pearson’s correlation coefficient." International Journal of Innovative Mathematics, Statistics & Energy Policies 6.1 (2018): 11-23.

[25] Adedoyn, O. O., and T. Mokobi. "Using IRT psychometric analysis in examining the quality of junior certificate mathematics multiple choice examination test items." International Journal of Asian Social Science 3.4 (2013): 992-1011.
