Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #47822

Datastorm - Using Data-Driven Challenges to Improve Student Engagement in Computer Science

Dr. Ankunda Kiremire, Louisiana Tech University

Dr. Ankunda Kiremire is a senior lecturer of Computer Science and Cyber Engineering at Louisiana Tech University and serves as the program chair for its Computer Science Department. His research interests include Computer Science Education, Cyber Security, and Data Science.

Dr. Kevin A Cherry, Louisiana Tech University

Dr. Kevin Cherry is a senior lecturer of Computer Science and Cyber Engineering at Louisiana Tech University. His research interests include Computer Science Education, Game Development, and Programming Languages.

Mr. Christian Smith, Louisiana Tech University

Mr. Christian Smith is an instructor of Computer Science and Mathematics at Louisiana Tech University and serves as the Course Coordinator for Calculus 0 (First year Calculus Course). His research interests include Computer Science Education, Mathematics Education, Computational Graph Theory, and Data Science.

Datastorm - Using Data Driven Challenges to Improve Student Engagement in Computer Science

Abstract

Computer Science and other computing-based courses with traditional pedagogy suffer from low student engagement, low student retention, antiquated information delivery, and a dearth of real-time evaluation tools for its instructors. In this paper, the authors present an innovative solution to all of these problems in the form of in-class Datastorm events.

The Datastorm challenges presented in this paper are a series of in-class competitions among teams of undergraduate Computer Science and Cyber Engineering students at Louisiana Tech University. In these challenges, the student teams are tasked with creating and implementing original solutions to problems from the field of data structures. The challenges are designed to test the students in three curricular targets: proficiency in Java coding, algorithm analysis, and data structure application. The collaborative, competitive, and exhilarating nature of the challenges also provides students with the opportunity to develop the soft skills necessary to thrive in this field.

The effectiveness of the in-class Datastorm challenges is analyzed through systematically conducted surveys. The collected data is related to student performance in the aforementioned areas, their engagement with the material and the field, their perception of their own mastery of the subject, their collaborative skill set, and the impact of the Datastorm challenges on their continued retention in our student body. This data is collected from three sections of a sophomore level Computer Science class at our institution containing a total of 95 students. The surveys, which consist of both subjective and objective measures, show that the Datastorm challenges help students grasp content better and help them improve their soft skills within the context of team work.

The authors also present feedback from faculty showing the Datastorm challenges' impact on the quality of information delivery and real-time evaluation options available to Computer Science instructors.

Introduction

Computer Science and computing-based majors at the university level face a variety of challenges.

One significant issue is low student engagement. Many students are unwilling to invest the effort needed to grasp complex concepts and develop the demanding skills required in Computer Science [1, 2]. This is a widespread issue with even international surveys reporting lower engagement rates in Computer Science compared to other majors [3]. While multiple factors contribute to this problem, research suggests that student engagement can be improved by integrating activities that are relevant to students' lives into Computer Science courses [1].

Another critical issue is low student retention. Attrition rates in Computer Science programs range from 9.8% [4] to 28% [5]. This results in both direct losses, as students fail to complete the

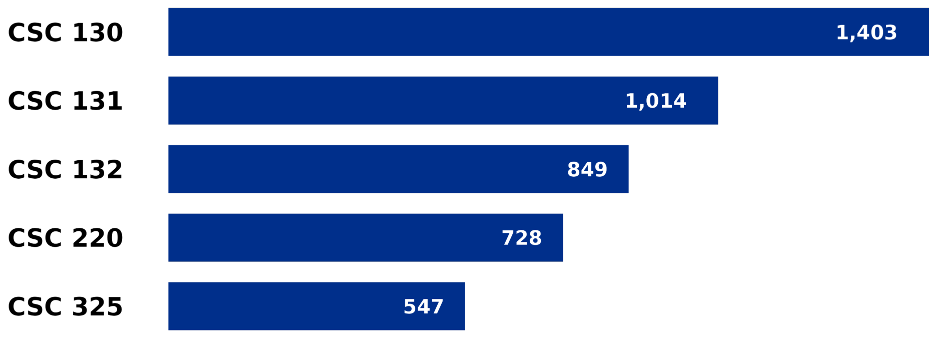


Figure 1: The size of our institution's first 5 Computer Science classes shows a significant drop in numbers as students progress through the curriculum.

major, and indirect losses, as potential students may be discouraged from entering the field due to its perceived difficulty and high withdrawal rates.

Figure 1 highlights a similar trend at Louisiana Tech University. It displays the total number of students who have taken the first five classes in our Computer Science curriculum over the past five years. While calculating exact attrition rates is not a straightforward process, the data reveals a significant decline in student enrollment as they progress through the curriculum.

Another challenge faced by Computer Science and computing-related majors is the outdated approach to presenting material to students. This issue manifests in two primary ways. First, the material is often not culturally relevant, as it fails to connect with the social and cultural interests that motivate students to pursue the field [6, 1]. Second, freshman and sophomore Computer Science courses frequently lack opportunities for collaboration and real-world application. As a result, some students become disillusioned during these formative years, perceiving Computer Science to be a field dominated by solitary coding in isolation.

A fourth issue, faced by instructors, is the limited variety of student evaluation methods used in these majors. While exams and programming assignments are effective for assessing students' understanding of concepts, there is considerable potential to enhance these traditional tools with innovative evaluation methods [7]. The methods presented herein could help bridge the gap between when a concept is taught and when both instructors and students identify the students' level of comprehension. This would enable timely interventions and corrective measures, both inside and outside the classroom.

In previous work, we introduced an innovative solution to address these issues at our institution: Datastorm [8]. Datastorm is an in-class competition where groups of students collaboratively create and implement original solutions to problems in the field of Data Structures within the context of Computer Science.

The feedback from that initial implementation informed the design and launch of three sets of Datastorm challenges, which were distributed to 95 Computer Science and Cyber Engineering students across three sections of a sophomore Data Structures class, taught by three different instructors at our institution. This paper presents the results of that process and offers insights into which aspects of Datastorm can be easily replicated at other institutions and applied to other subfields of Computer Science.

The results indicate that the Datastorm challenges improved and standardized students' understanding of the material, regardless of differences in instructor pedagogy. Additionally, the challenges enhanced students' soft skills through the context of small group competitions. Finally, the Datastorm challenges boosted students' self-confidence in the subject matter.

Methodology

Data Storm Challenge Implementation

The Datastorm challenges are divided into three sets, each testing a different aspect of the curriculum.

Challenge 1: Java Introduction

This challenge evaluates students' understanding of basic Java concepts. Questions focus on writing simple Java code involving loops, if-statements, arrays, and string manipulation. For example, one problem asks students to create a function that takes a minimum and maximum number and returns a string of all even numbers within that range. Teams submit their solutions to a server, which validates the code and unlocks subsequent problems if the validation passes. If the code fails, teams receive an error message and must troubleshoot using provided test code.

Challenge 2: Time Complexity

This challenge focuses on asymptotic time complexity. Students analyze 20 code snippets, including nested loops, recursion, and function calls, and select the correct time complexity from 20 multiple-choice options. Teams submit their answers to a server, which provides feedback on the number of correct responses without identifying specific errors. Each submission incurs an increasing time delay to encourage accuracy and discourage a guess-and-check approach.

Challenge 3: Data Structures (DS)

This challenge tests students' ability to work with basic Abstract Data Types (ADTs), such as arrays, lists, queues, and stacks. For instance, one task asks students to write a function that converts a string into a list of its individual characters. This set includes five questions that are increasing in difficulty and are evaluated similarly to the "Java Introduction" challenge.

Challenge Administration

Students were organized into teams of up to four, with each challenge contributing to their course grade. A passing score for each challenge was predefined and communicated in advance, and teams could earn additional points to improve their grades. Each challenge set was conducted during a single class period, referred to as a "challenge day." Examples of specific problems from each challenge set can be found in Appendices A, B, and C.

In the rest of this paper, we shall refer to these three challenges as Java, Complexity, and Basic ADTs respectively.

Data Collection

Each challenge set was accompanied by a survey or quiz designed to collect pertinent data. These quizzes were administered three times in relation to the challenge day to track student progress. The structure of each quiz mirrored that of the corresponding challenge: coding-based challenges included coding questions, while challenges with multiple-choice questions featured similar formats in the quizzes. Additionally, students were asked to self-evaluate their knowledge on topics related to each challenge. Quiz scores did not contribute to the overall course grade, and responses were scored on a scale from Poor (1) to Excellent (4), encompassing both objective metrics and self-assessments.

The first quiz attempt for all challenges was conducted on the first day of the term to establish a baseline of students' understanding before traditional instruction began. Challenges were then implemented at points deemed appropriate by each instructor based on their pacing. Consequently, the quiz schedules varied across sections. Regardless, the second quiz attempt was administered the day before the corresponding challenge, reflecting the conclusion of traditional instruction. The third attempt occurred during the class period immediately following the challenge to measure students' understanding afterward. In total, nine quizzes were conducted throughout the course. The questions for each quiz are detailed in Appendices A–C, corresponding to the challenge sets.

In addition, a fourth quiz, unrelated to the course content, was administered at both the start and end of the term to evaluate students' perceptions of team dynamics and competitions. The end-of-term version of this quiz included additional questions aimed at assessing students' perceived improvements over the school term. Responses were measured on a scale ranging from Strongly Disagree to Strongly Agree. The questions for these assessments can be found in Appendix D.

Results

This section presents the results of systematically conducted surveys from up to 95 sophomore students. Where applicable, the results are compared across three sections of the same class, referred to as Section 1, Section 2, and Section 3. In other instances, the results from all sections are aggregated and analyzed collectively. The topics covered in the class and addressed by the challenges include Java, Complexity, and Basic ADT's.

The section is divided into four areas of discussion: the impact of the Datastorm challenges on students' understanding of the class material, their ability to work in teams and their perceptions of teamwork, their confidence in their mastery of the course content, and anecdotal feedback from the instructors.

Mastery of Material

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
34%	40%	24%	0%	2%

Table 1: The Datastorm challenges helped me grasp the class material better than I would without it

As shown in Table 1, 74% of students agreed that the Datastorm challenges helped them grasp the class material better than they would have without them, while only 2% disagreed.

Students were also evaluated using objective questions and the results of these evaluations, ranked by topic and improvement resulting from the challenges, are shown in Figure 2 below.

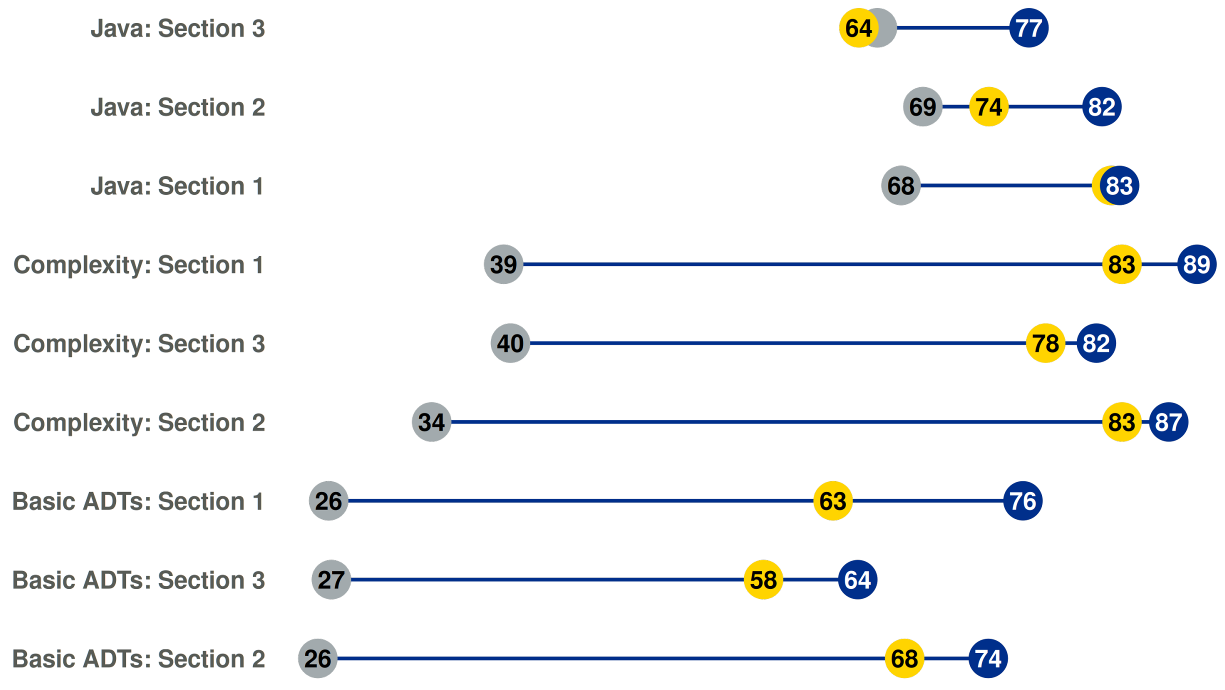


Figure 2: Students' mastery of the concepts was evaluated at three different points, before covering the material, after covering the material, and after participating in the Datastorm challenge related to that material. Generally speaking, there was an improvement in student performance across all sections and all topics.

The figure illustrates consistent improvement in average class scores for all concepts when evaluated **before** and **after** a Datastorm challenge. For example, in the "Java" topic, Section 3 improved from 64% to 77%, Section 2 from 74% to 82%, and Section 1 from 82% to 83%.

This result is noteworthy for several reasons. First, it demonstrates that Datastorm challenges are beneficial even for students who have already grasped some or all of the material in a topic. The challenges provide opportunities to practice and apply a variety of concepts. For "Java" specifically, all students had been exposed to Java syntax at the tail-end of a prerequisite class in which Python was taught.

Second, the results suggest that Datastorm challenges help standardize students' grasp of tested concepts, regardless of their initial exposure to the material. Although all three sections spent two 75-minute class periods reviewing Java syntax (each instructor emphasizing different aspects of the language), the students' initial understanding varied widely (64%, 74%, and 82%). After the Datastorm challenge, the scores converged significantly (77%, 82%, and 83%, respectively). The fact that this improvement occurred within a single class period is particularly remarkable.

A similar trend was observed for the "Complexity" and "Basic ADTs" topics, where students had no prior exposure to the material. The Datastorm challenges were instrumental in helping students master these concepts, as they were specifically designed to evaluate them.

Teaching styles and instructional durations varied significantly across the three sections. For example, the instructor of Section 1 relied heavily on in-class Java programming to drive theoretical discussions, while Section 2's instructor used visualizations to explain the same concepts. In contrast, Section 3's instructor favored traditional theoretical discussions supplemented with pseudocode examples.

Additionally, the time spent on individual topics and the amount of graded versus ungraded content differed among instructors. Even with these variations, the Datastorm challenges effectively bridged the gaps in understanding, leading to comparable results across all sections.

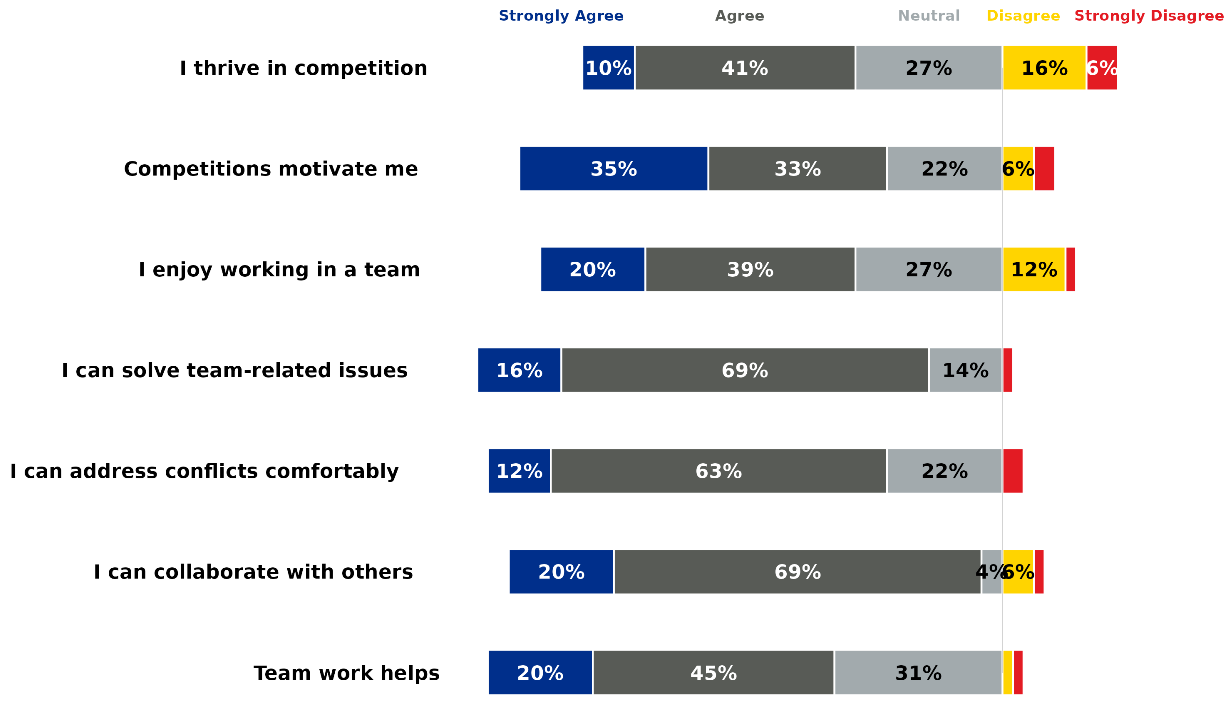
Working in Teams

As part of the second assessment in Quiz 4, students were asked to rate their agreement with the statement: "*I believe that the Datastorm challenges have allowed me to become better at working in teams,*" on a scale from Strongly Agree to Strongly Disagree. The results, presented in Table 2, show the percentage of students in each category.

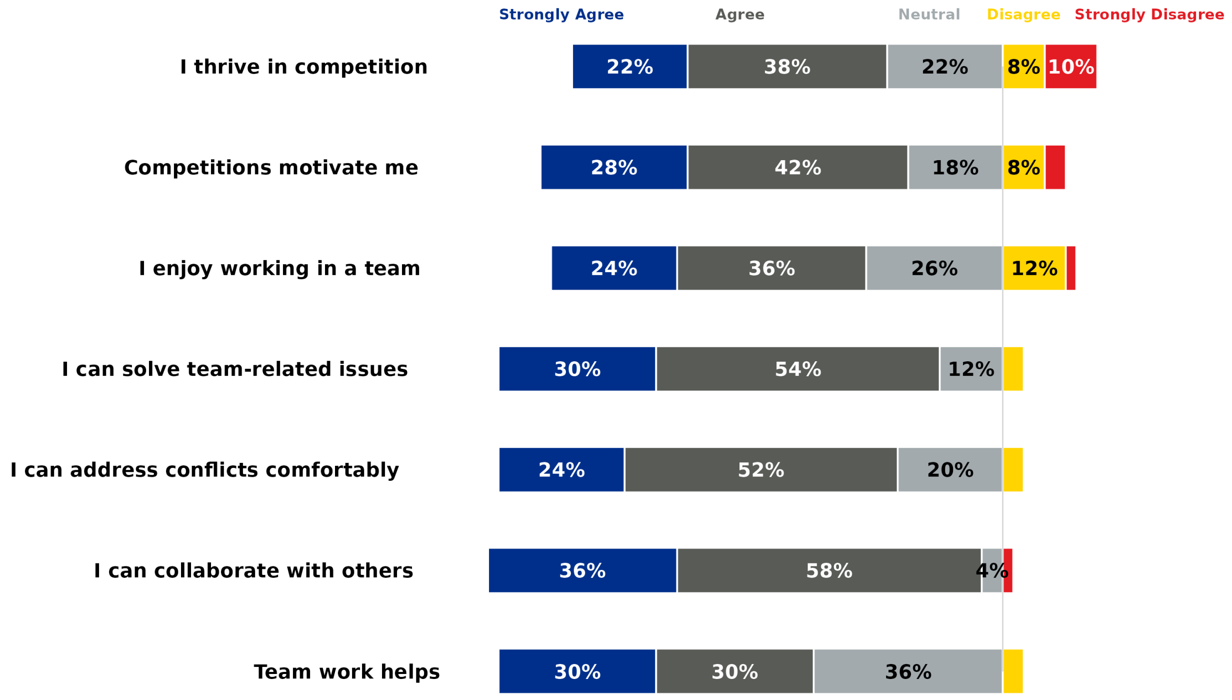
Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
26%	52%	20%	0%	2%

Table 2: Students' Perceived Impact of Challenges on Their Own Team Skills

This data demonstrates that 78% of students agreed or strongly agreed that the implementation of Datastorm challenges was helpful in developing their soft skills in collaborative environments. Further support for this claim is provided by additional data from Quiz 4, where students answered questions related to their confidence in handling conflict, working in groups, and the effects of competitive environments. This supporting data is shown in Figure 3, with the order and titles of the questions corresponding to those in Appendix D.



(a) At the beginning of the school term



(b) At the end of the school term

Figure 3: Self Assessment of team work, competitions, and soft skills.

Several notable conclusions can be drawn from this data. First, the subjective responses confirm that students felt the challenges improved their ability to work in teams. Additionally, the number of students who reported enjoying competition increased as a result of the Datastorm challenges. However, there was no significant change in the students' motivation to participate in competitive environments or to work in teams, as reflected in the responses to the final question.

Questions related to solving issues, addressing conflict, and collaboration revealed that students experienced increased confidence in the soft skills essential for success in team-based work. Notably, the number of students expressing a lack of confidence in solving team-related issues decreased, with many shifting toward a more positive outlook. A similar trend was observed for confidence in collaboration with others. Furthermore, there was a significant increase in students' confidence in addressing conflict effectively and comfortably.

These findings demonstrate that the implementation of the Datastorm challenges improved students' self-confidence in their soft skills, even though their desire to work in teams or competitive settings remained largely unchanged. This dichotomy suggests that while students may not have developed a stronger preference for collaboration or competition, their ability to perform effectively in these contexts has improved significantly through the development of their soft skills.

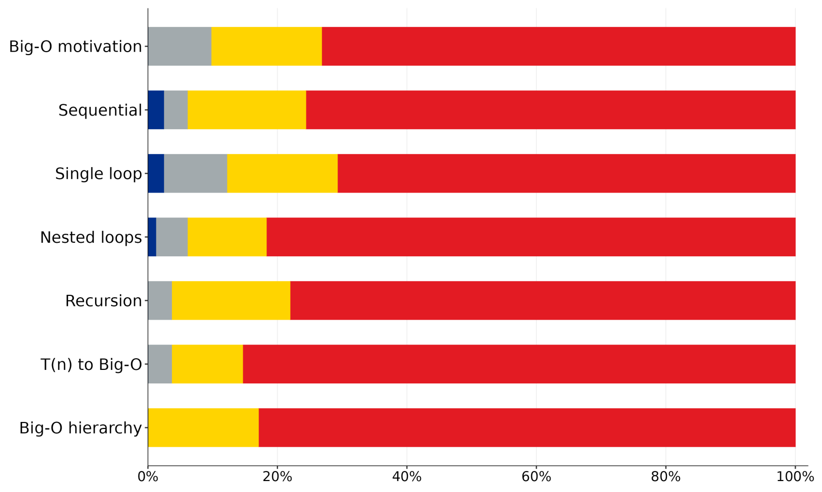
Student Perception of Content Mastery

The Datastorm challenges also influenced how well students believed they had grasped the class material. As part of three quizzes administered at different points during the school term, students were asked to evaluate their knowledge of concepts in the three major class topics. Each topic was assessed with five to seven questions, and students self-rated their mastery as Poor, Fair, Good, or Excellent. The results of the quiz on Complexity are shown in Figure 4.

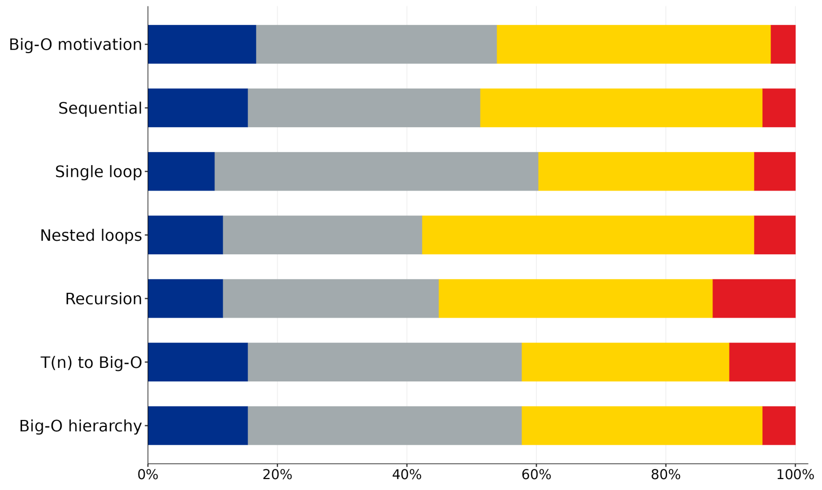
For instance, prior to covering the material on "Complexity", about 12% of students rated their understanding of single loops as Excellent or Good. This increased to 61% after the material was taught in class and further rose to 78% following the Datastorm challenge on Complexity.

Similar patterns were observed in the quizzes on "Java" and "Basic ADTs", with consistent improvements across all topics. The results of these quizzes are provided in Appendix E for reference.

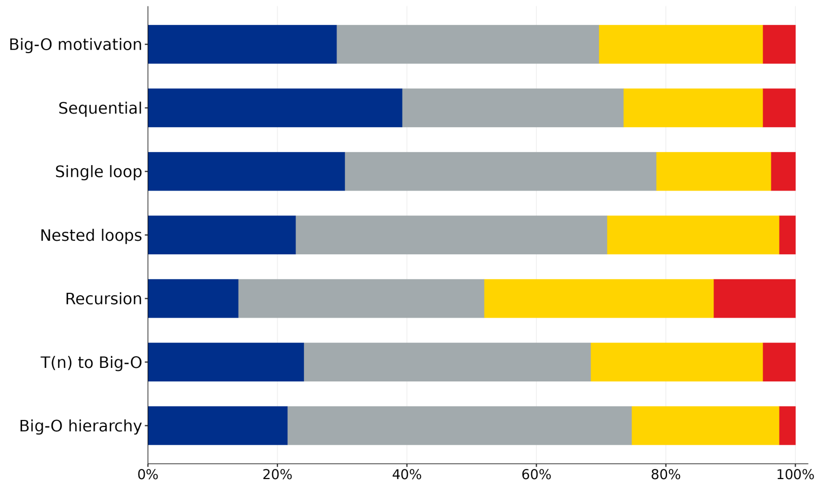
These findings highlight another advantage of the Datastorm challenges: they help students develop an accurate and healthy understanding of their grasp of key concepts within a subject.



(a) Before material was covered in class



(b) After material was covered in class



(c) After the appropriate Datastorm challenge

Figure 4: There was improvement in the students' perception of their content mastery in the Complexity topic across the school term. The students' evaluated their mastery of the indicated subtopics as either excellent, good, fair, or poor.

Faculty Feedback

The instructors involved in this study were pleasantly surprised by the significant quantitative improvement in students' grasp of the material achieved through the incorporation of Datastorm challenges into their teaching. While some were initially skeptical about dedicating a class period to each challenge during the school term, the data presented earlier in this section validated their decision.

Student feedback indicated that the challenges felt like engaging Computer Science puzzles that were both enjoyable and intellectually stimulating. Instructors also observed how teams celebrated their successes together after overcoming challenge problems, often after multiple failed attempts. This camaraderie and shared sense of achievement seemed to enhance the vigor with which teams approached subsequent challenges.

However, the instructors emphasized the importance of carefully assembling balanced teams to ensure equitable participation and collaboration. Teams should be structured to promote shared effort and idea exchange rather than relying on a single member to complete the challenges. Addressing disparities within or between teams is crucial to minimizing frustrations and preventing a loss of confidence that can arise in competitive environments.

To increase the scope of the challenges, it was suggested to implement a longer, cross-section competition that could incorporate more complex problems and real-world datasets. They believe such an initiative would foster even greater student engagement and provide opportunities to apply their skills to practical scenarios. However, they acknowledged that this approach would require additional coordination and administrative effort.

Discussion

In this section, the authors provide some insights into the feasibility of recreating the successes in this paper across other institutions and other subfields.

The current implementation of in-class challenges imposes an initial administrative cost on instructors, primarily during setup and the first challenge. This phase requires ensuring that students configure their laptops correctly to interact with the online challenge evaluation system. However, this cost decreases significantly after the first challenge, as students become familiar with the technical requirements and can focus more on the theoretical concepts being tested.

For larger class sizes, having Teaching Assistants (TAs) is recommended to help students troubleshoot setup issues. The TAs could also provide support to those with knowledge gaps in the challenge topics consistent with the instructor's pedagogical preferences. In this study, the instructors did not have TAs and managed class sizes ranging from 20 to 44 students.

Another important factor to consider is challenge design, specifically how the sub-tasks within each challenge are structured. Challenges can exist on a spectrum between fully linear and fully parallel.

A linear challenge consists of sequential sub-tasks, where each task must be completed before the next can begin. In contrast, a parallel challenge allows all sub-tasks to be tackled simultaneously in any order.

To encourage collaboration within groups, we found that fully linear challenges often lead to dominance by one or two of the most skilled students, limiting engagement from the rest of the team. In this study, the challenges used were either fully parallel or a hybrid, incorporating small linear sections within an overall parallel structure.

The design of the challenges also influences the recommended maximum group size. To promote collaboration, instructors suggest a maximum group size of four, with three being ideal. As mentioned earlier, larger groups often lead to dominance by stronger students, especially when challenges lack multiple sub-tasks that allow all members to contribute meaningfully over an extended period.

Smaller groups also facilitate better communication among team members. To maintain this balance, instructors with larger classes are encouraged to increase the number of teams rather than expand team sizes. For context, the instructors in this study have historically worked with between three and twelve teams per section.

Louisiana Tech University has successfully implemented a similar challenge system in the field of Cyber Security and is in the process of developing similar systems in Programming Languages as well as Introductory Python Programming. Expanding this approach to other subfields only requires the development of a bank of challenge questions and tasks that are interconnected and designed to be tackled collaboratively by student teams.

Future Work

We have identified several paths for improvement that we would like to explore. One area of focus is the scalability of our Datastorm challenges. Specifically, we aim to evaluate the feasibility of larger-scale challenges involving more student teams and extended durations that go beyond the typical classroom period and size. We plan to pilot this approach and assess its impact on learning outcomes. Our expectation is that this will further motivate students while fostering deeper learning and improved retention.

Another area for enhancement is the incorporation of real-world datasets into the Datastorm challenges. For example, we could use sales or operational data sourced from local companies. By solving problems using actual datasets from sources that the students are intimately familiar with and passionate about, we anticipate that students will be more engaged and motivated in the problem-solving process. We suspect this will lead to increased student enthusiasm and participation.

Additionally, while our assessments were not partitioned by demographic, we believe the inclusive competition and teamwork fostered by our challenges have the potential to enhance engagement among women and minorities. We anecdotally noticed more communication among teams that embraced the various perspectives from all team members. To further support this goal, we plan to collaborate with minority-owned businesses to source culturally relevant datasets for future challenges. Additional research and assessments will be required to evaluate whether these efforts lead to greater diversity and inclusion within the student population and the broader field.

References

[1] B. Hoffman, R. Morelli, and J. Rosato, "Student engagement is key to broadening participation in cs," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1123–1129. [Online]. Available: https://doi.org/10.1145/3287324.3287438

[2] D. Mahatmya, B. Lohman, J. Matjasko, and A. Farb, *Engagement Across Developmental Periods*, 01 2012, pp. 45–63.

[3] M. Butler, J. Sinclair, M. Morgan, and S. Kalvala, "Comparing international indicators of student engagement for computer science," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2843043.2843065

[4] K. Flinders, "Computer science undergraduates most likely to drop out," August 2019, [Online; posted 1-August-2019]. [Online]. Available: https://www.computerweekly.com/news/252467745/Computer-science-undergraduates-most-likely-to-drop-out

[5] E. O'Brien, "Why are computer science drop-out rates so high?" February 2016, [Online; posted 21-February-2016]. [Online]. Available: http://trinitynews.ie/2016/02/why-are-computer-science-drop-out-rates-so-high/

[6] N. (IUSE:CUE), "Student-centered computing: Teaching computer science using culturally authentic practices," July 2022, [Online; retrieved 22-July-2022]. [Online]. Available: https://aaas-arise.org/2022/07/05/student-centered-computing-teaching-computer-science-using-culturally-authentic-practices/

[7] D. Hamilton, J. McKechnie, E. Edgerton, and C. Wilson, "Immersive virtual reality as a pedagogical tool in education: a systematic literature review of quantitative learning outcomes and experimental design," *Journal of Computers in Education*, vol. 8, 07 2020.

[8] A. Kiremire and K. A. Cherry, "Board 52: Work in progress: Datastorm: Using data-driven competition to improve student engagement in computer science," in *2024 ASEE Annual Conference & Exposition*, no. 10.18260/1-2–47049. Portland, Oregon: ASEE Conferences, June 2024, https://peer.asee.org/47049.

Appendix A

Sample Question Challenge 1:

Code Number 6:
In the JavaIntro_Challenges class (in JavaIntro_Challenges.java), create a public static function called 'c6' that does the following: The function will return a String generated by following these directions:

- Create an int array of a specified length (given as a parameter)
- Fill it with random integers in the range of 0 (inclusive) to 74 (inclusive)
- The random number generator must use a specified number as the seed, given as the second parameter
- Now sort all numbers in the array (you can use a built in Java methods for this – look it up)
- Create a String and add the sorted sequence of numbers to the String (with a space between each number)
- The last number may not contain spaces after it
- The first number may not contain spaces before it
- Return the String

Use this for the function signature/header:

```
public static String c6(int size, int seed)
```

Quiz Questions for Challenge 1:

Please rate how confident you are in the following areas related to the Java programming language. (Poor, Fair, Good, Excellent)

1. Creating, compiling, and executing code
2. Recognizing and creating syntactically correct code
3. The creation and use of basic arrays and strings
4. The design and use of classes and objects
5. The design and use of for and while loops
6. The design and use of functions
7. The design and use of conditional statements
8. Which of these are syntactically correct if they were in a file called Main.java.
   a.)
   ```
   public class Program {
       private void main(String[] args) {
           System.out.println(foo(5));
       }

       private static int foo(int x) {
           return x + 3;
       }
   }
   ```

   b.)

```java
public class Main {
    public static void main(String[] args) {
        System.println(foo(5));
    }

    public int foo(int x) {
        return x + 3;
    }
}
```

c.)

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(foo(5));
    }

    private static int foo(int x) {
        return x + 3;
    }
}
```

d.)

```java
private class Main {
    private void main(String[] args) {
        System.out.println(foo(5));
    }

    public static int foo(float x) {
        return x + 3;
    }
}
```

9. Write a complete function called getMax that takes two integers and returns the larger value of the two. Assume that your function will be called in a different function in the following manner. Do not worry about class declarations for this question. Example:

   ```java
   int x = getMax(32, 13);
   ```

10. Write a complete function called fillArray that meets the following criteria
    - It takes an integer as its only argument.
    - It creates and fills an array with all integers from 0 up to that integer (inclusively).
    - It returns the resulting array to the calling statement.

    Assume that your function will be called in a different function in the following manner. Do not worry about class declarations for this question.

    ```java
    int[] array = fillArray(10);  // array
            // returned would have {0,1,2,...,10}
    ```

    Match the terms below to the lines of code on which those terms are referenced. If a term is referenced on multiple lines, make sure to mention all the relevant lines.

11. Class signature
12. Declaration of Instance variables
13. Initialization of instance variables
14. Typecasting
15. Instantiation

16. Method call
17. Class constructor

```java
class Fraction {
    int num;
    int den;

    public Fraction(int num, int den) {
        this.num = num;
        this.den = den;
    }

    public float getReal() {
        return this.num / (float)this.den;
    }
}

public class Main {
    public static void main(String[] args) {
        Fraction fract = new Fraction(1, 2);
        System.out.println(fract.getReal());
    }
}
```

Appendix B

Sample Questions from Challenge 2:

Determine the time complexity of the following functions in terms of Big-O notation.

1.
```java
for(int i = 1; i < n; i += 3) {
    for(int j = 3; j < 10; j++) {
        for(int k = 5; k < n; k += n) {
            ...
        }
    }
}
```

2.
```java
// Assume both the names and the ranks array are of equal size.
// Let n be the size of the array.
void e(String[] names, String[] ranks) {
    String[] namesAndRanks = new String[names.length];
    for(int i = 0; i < names.length; i++) {
        namesAndRanks[i] = names[i] + " " + ranks[i];
    }
    String combined = "";
    for(int i = 0; i < namesAndRanks.length; i++) {
        combined += namesAndRanks[i] + ", ";
    }
    System.out.println(combined);
}
```

Quiz Questions for Challenge 2:

Please rate how confident you are in the following areas related to the Java programming language. (Poor, Fair, Good, Excellent)

1. Understanding the relevance/need for Big-O notation as a way of comparing algorithms
2. Identifying the time analysis of a snippet of code that doesn't contain any repetition
3. Identifying the time analysis of a snippet of code that contains a single loop
4. Identifying the time analysis of a snippet of code that contains nested loops
5. Identifying the time analysis of a snippet of code that contains recursion
6. Translating a time analysis(T(n)) into Big-O notation
7. Comparing different big-O categories i.e. which is better/worse

   What is the time of complexity in Big-O Notation of the code snippets below?

8.
```
// code snippet with constant run time.
public int getSquare(int n)
{
        return n * n;
}
```

9.
```
// code snippet with single loop
int[] array = getValues();
int n = array.length;
int i = 5;

while (counter < n - 7) {
    System.out.println(array[i]);
    i *= 2;
}
```

10.
```
//code snipped with nested loop
String[] names = getNames();
int n = names.length;
for (int i = 0; i < n; i++) {
    int j = 3;
    while (j < n * n) {
        j += 2;
    }
}
```

11. Order the following time complexities (T(n)) in order of the fastest execution.

    $n^2, 3n^1, n * \log(n), 3^n, 10, \log(4n)$

Appendix C

Sample Question Challenge 3:

Code number 3:
Create a public static function called "c3" that does the following: The function will receive an array of chars (called "letters") and an array of ints (called "numbers") (in that order). You will need to create a string with one letter and

one number from the same index in the array and concatenate those with a colon between (refer to the example below). Each string will then be pushed into a Stack of strings and that stack will need to be returned. Assume the given arrays are not null but may be empty. If empty, you should return an empty Stack. Also assume the letters and numbers arrays will always be the same length.

Example:

```
/* call */
Ds_Challenges.c3(['a','b','c'],[1,2,3]);
/* return */
```

top
$c : 3$
$b : 2$
$a : 1$
bottom

Quiz Questions for Challenge 3:

Please rate how confident you are in the following areas related to the Java programming language. (Poor, Fair, Good, Excellent)

1. Explaining how a List works behind the scenes
2. Explaining how a stack works
3. Explaining how a queue works
4. Programmatically adding and removing elements from a stack
5. Programmatically adding and removing elements from a queue
5. Assuming you have access to a Stack ADT with the functions “push”, “pop”, “isEmpty”, write a function of code called “fillStack” that will
   – Receives an array of integers as its only argument
   – Creates a stack and puts all elements in the array into the stack in order
   – Returns the resulting stack
6. Assuming you have access to a Queue ADT with the functions “enqueue”, “dequeue”, “isEmpty”, write a function of code called “toString” that will.
   – Receives a queue of characters as its only argument
   – Creates a string and puts all the characters in the queue into the string in order
   – Returns the resulting string
7. Assuming you have access to a List ADT with the functions “size”, and “getValue(int index)”, write a function of code called “getSum” that will
   – Receive a List of floating point values as its only argument
   – Calculates the sum of all the values in the List
   – Returns the resulting sum

Appendix D

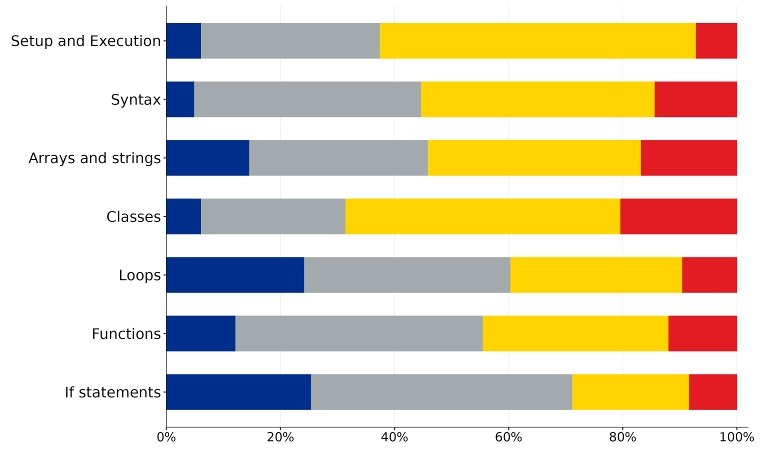
Quiz 4 Questions: Team Dynamics, Competitions, and Improvement Survey Questions

1. I feel that I thrive in competitive environments.
2. Competitions motivate me to perform better.
3. I enjoy working as part of a team.
4. I feel confident in finding solutions to team-related issues.
5. I am comfortable addressing conflicts or disagreements within a team.
6. I am confident in my ability to collaborate effectively with others.
7. I believe that working in teams helps me achieve better outcomes.
8. I believe that the data storm challenges have allowed me to become better at working in teams.
9. I believe that the data storm challenges helped me to grasp the class material better than I would have without it.

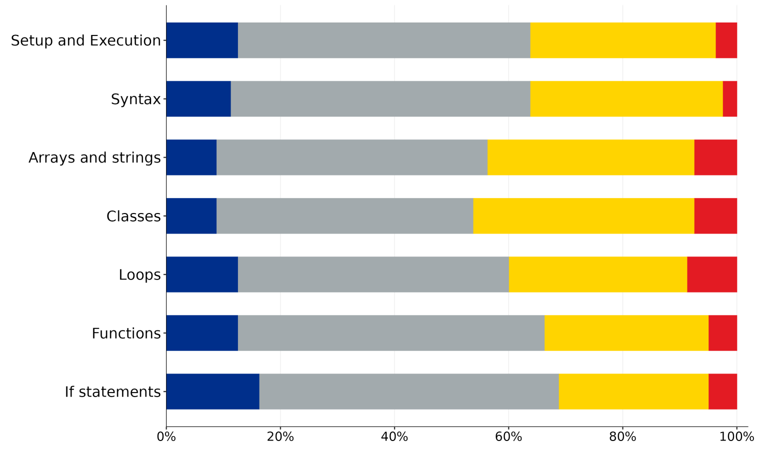
Note: Questions 8 and 9 were asked on the second assessment of the survey at the end of the term.

Appendix E

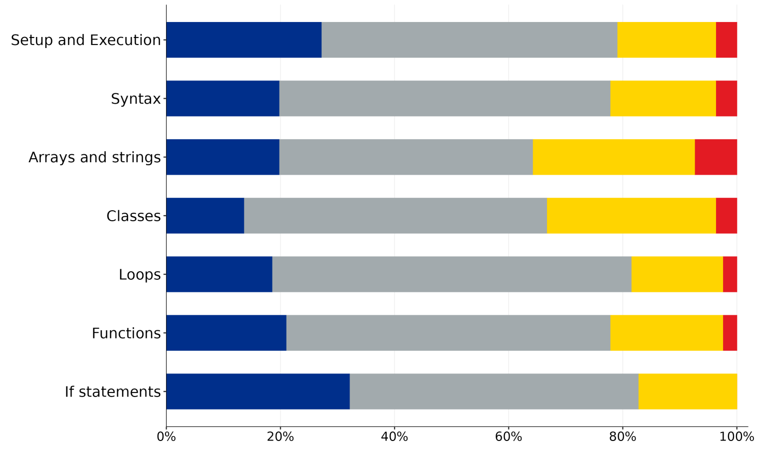
Results showing the change in student perception of their own mastery of the concepts in different topics across the School term in both Java (Figure 5) and Basic ADTs (Figure 6).



(a) Before material was covered in class

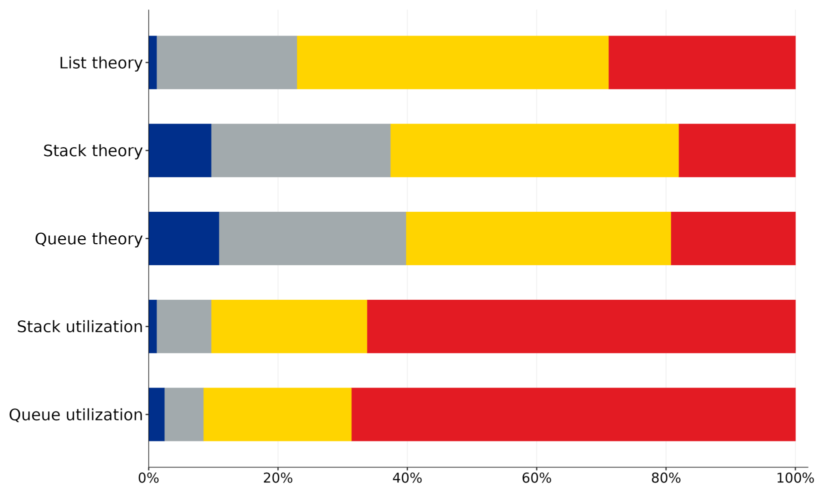


(b) After material was covered in class

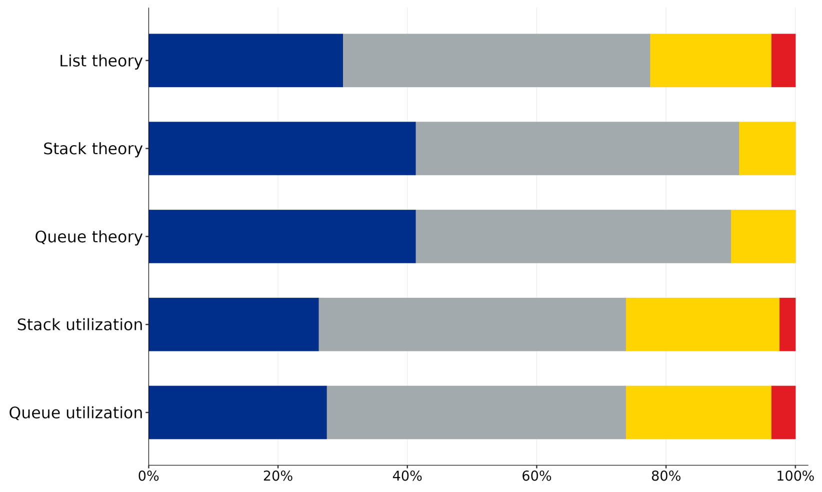


(c) After the appropriate Datastorm challenge

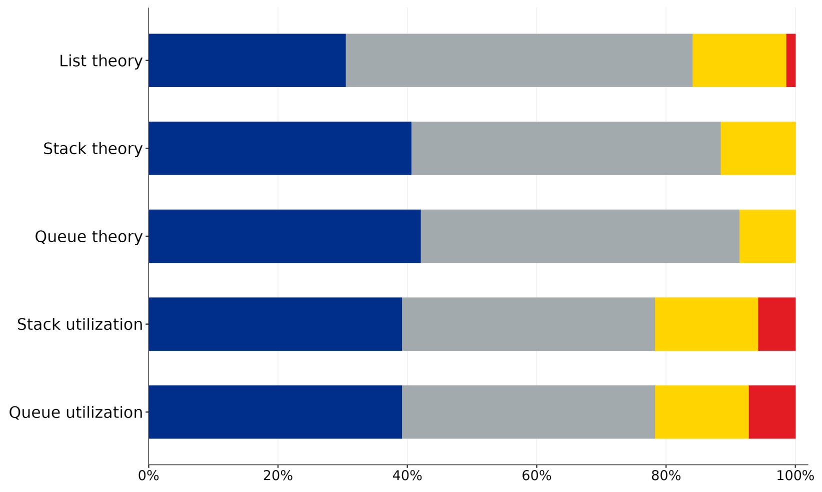
Figure 5: Students' evaluated their grasp of Java concepts as either excellent, good, fair, or poor.



(a) Before material was covered in class



(b) After material was covered in class



(c) After the appropriate Datastorm challenge

Figure 6: Students' evaluated their grasp of Basic ADT concepts as either excellent, good, fair, or poor.