# Student Misconceptions of Dynamic Programming: A Replication Study

Michael Shindler[a] and Natalia Pinpin[a] and Mia Markovic[a] and Frederick Reiber[a] and Jee Hoon Kim[a] and Giles Pierre Nunez Carlos[a] and Mine Dogucu[a] and Mark Hong[a] and Michael Luu[a] and Brian Anderson[a] and Aaron Cote[b] and Matthew Ferland[b] and Palak Jain[a] and Tyler LaBonte[b] and Leena Mathur[b] and Ryan Moreno[c] and Ryan Sakuma[a]

[a]University of California, Irvine; [b]University of Southern California; [c]University of Wisconsin-Madison

**ARTICLE HISTORY**
Compiled May 11, 2022

**ABSTRACT**

**Background and Context**: In this study, we replicated and expanded the work of Zehra et al. (2018), which studied how well students learn dynamic programming, a notoriously difficult topic for students in a core algorithms class (Enström and Kann, 2017). Their study interviewed a number of students at one university in a single term. We expanded on these results by recruiting a larger sample size of students, over several terms, in both large public and private universities as well as liberal arts colleges.

**Objective**: Our aim was to investigate whether the results of Zehra et al. (2018) generalized to other universities and also to larger groups of students.

**Method**: Our methodology was similar to theirs: we interviewed students who completed the Divide-and-Conquer and Dynamic Programming portions of their undergraduate algorithms class, asking them to solve problems that required one of those techniques without telling them which they needed. We observed the students' problem solving process to glean insight into how students tackle these problems.

**Findings**: We found that students generally struggle in three main ways, those being "technique selection," "recurrence building," and "inefficient implementations." We then explored these themes and specific misconceptions qualitatively. We observed that the misconceptions found by Zehra et al. (2018) generalized to the larger sample of students.

**Implications** Our findings demonstrate areas in which students struggle, paving way for better algorithms education by means of identifying areas of common weakness to draw the focus of instructors.

**KEYWORDS**
Replication study; dynamic programming; algorithms education

## 1. Introduction

In this study, we replicated the work of (Zehra et al., 2018), as there has not been extensive research in algorithms education, such as student attainment in complexity analysis, divide-and-conquer, and, most notably for this paper, dynamic programming

CONTACT Michael Shindler Email: mikes@uci.edu

(DP). In Zehra et al., they investigated student misconceptions about dynamic programming (Zehra et al., 2018). We performed this replication study because DP is particularly difficult for students to learn (Enström and Kann, 2017), and while many works have studied students' thought processes in solving algorithms problems, few have examined DP misconceptions specifically. We believe Zehra et al.'s unique focus on DP misconceptions is a valuable research direction which merits large-scale replication. We believe DP is important because it is one of the few algorithm design techniques that can take problems with an exponentially large problem space and produce polynomial time algorithms (Goodrich and Tamassia, 2014). The technique, and the algorithms produced by it, is used in a variety of fields outside of computer science (Erickson, 2019). For these reasons, it is a core topic in undergraduate algorithms courses.

In this paper, we present our findings in replicating and extending Zehra et al. (2018). In the original study, a small research team conducted 14 think-aloud interviews at a North American research university with students who had recently learned DP. Each interview was administered by two researchers, with one researcher guiding the interview and the other taking notes. Participants were asked to solve a series of algorithm design problems, including both DP and non-DP questions. Each interview was then individually analyzed for DP misconceptions by the two researchers. Following this analysis, both researchers discussed the found misconceptions to reach a consensus. The researchers grouped these misconceptions into broader themes.

The original study had a low sample size of 14 students from only one university. We have expanded the study to include 65 students from 15 universities with varying demographics. Due to the COVID-19 pandemic, we conducted the interviews via remote conferencing.

We hypothesized that our findings would validate the previously identified student misconceptions in DP. Additionally, by interviewing more students across a variety of universities, we expected to discover new misconceptions and come to a better understanding of how widespread each misconception is. By expanding our sample, we sought to not only provide a more representative study, but also find out how accurately students identify DP problems, investigate what mistakes students make when implementing DP, and detect patterns between these misconceptions.

Our results can be summarized as follows. First, students struggled with properly identifying when DP is an appropriate technique, and often chose to use a greedy heuristic instead. Second, when students did select to use DP, they most often struggled with building the recurrence relationship. Finally, we explored if there was any connection between misconceptions and basic demographic information, like the university attended or the number of years the student had taken university-level computer science courses.

In Section 2, we give a review of the relevant literature pertaining to DP and concept inventories. Section 3 contains the full interview process as well as slight differences between our process and that of Zehra et al. (2018). Section 4 explains how our study was preregistered with the journal and describes any changes between the preregistration approved and final methods. Section 5 details the results of the paper and provides qualitative analysis of the data. Section 6 groups the found misconceptions into themes and provides examples of student mistakes. In Section 7, we compare our results with the original study, and in Section 8 we discuss improvements for future studies.

## 2. Literature Review

One of the primary motivations of this study was to provide a rigorous foundation for the study of algorithms education. Prior to the publication of Zehra et al. (2018), few studies had investigated students' misconceptions about dynamic programming.

In Danielsiek et al. (2012), researchers utilized both multiple choice assessments and think-aloud interviews to identify misconceptions about algorithms and data structures. Part of these interviews required students to complete partial pseudocode for two DP problems. The researchers found that almost all students struggled with the problems presented, and they exhibited a wide variety of DP misconceptions, such as conflating DP with divide-and-conquer and misinterpreting memoization. The following academic year, a subset of the same authors performed a similar study, Paul and Vahrenhold (2013), to validate the results of Danielsiek et al. (2012) by re-administering the same two DP problems on a larger sample of students. They examined if the conceptual difficulty of the DP problems influenced the identified DP misconceptions, namely if students were able to guess correct answers based on deduction even if they did not have an actual understanding of DP. The results confirmed that the problems are in fact sufficient for detecting DP misconceptions and can cross-validate the misconceptions found in each. They also constructed a multiple choice test and a fill-in-the-blank test to detect when students are likely to conflate divide-and-conquer and DP. By assessing the tests, they discovered that the multiple choice test is useful for gauging passive knowledge on the topic while the fill-in-the-blank test evaluates students' active knowledge and can be used to verify the results of the first test.

Additional papers have investigated the difficulty of DP, but not in the lens of student misconceptions. Enström (2013) examined the effectiveness of methods for teaching DP in algorithms courses, including clicker questions, labs, and visualization tools. Through self-efficacy reports and course surveys, students expressed that the most challenging aspects of DP were finding an "evaluation order" for a recurrence relation and designing a complete DP algorithm on their own. In a similar method of redesigning teaching activities, approaches, and objectives, Enström and Kann (2017) looked at the difficulties of teaching DP and NP-completeness through a multi-year study in an advanced CS theory course, finding successful results from adding programming assignments, clicker questions, and pattern-oriented instruction among other things.

Other works have developed the notion of a concept inventory, a course-level bank of meticulously crafted content questions, from which student answers can identify common misconceptions. A key part of creating concept inventories is data collection. By leveraging misconceptions of students taking the same course, we can formulate questions that properly target these issues in student learning (Taylor et al., 2020). There has been preliminary work in building concept inventories for topics related to DP such as algorithm analysis (Farghally et al., 2017) and recursion (Hamouda et al., 2017).

The development of concept inventories for basic data structures has received significantly more research than concept inventories for algorithms. Taylor et al. (2020) provides an outline on how to develop a concept inventory, citing the previous work of Porter et al. (2018) about basic data structures as a practical example. One motivation for identifying DP misconceptions is building a necessary foundation for an algorithms concept inventory. Other studies have found misconceptions in and developed concept inventory questions for specific data structures and elementary algorithms, such as binary search trees and hash tables (Karpierz and Wolfman, 2014). However, aside

from Danielsiek et al. (2012), little work has been done to develop concept inventory questions related to DP and other related algorithms topics.

## 3. Method

In this section, we detail our process for conducting the study. This includes the process of gathering volunteers, conducting the interview, and handling the data. Most importantly, we also discuss how our version of the study differs from the original Zehra et al. (2018) paper.

### *3.1. Participants*

The process of gathering participants spanned from March through August of 2021. Our target demographic for participants was students currently enrolled in an undergraduate algorithms course that covered DP and had completed the DP unit. We began our recruitment process by researching which US universities teach undergraduate algorithms courses. By searching through course catalogs, course descriptions, and syllabuses, we determined which algorithms courses cover DP. We contacted professors of those courses via email to introduce our study and ask them to forward an announcement to their students to recruit participants.

This announcement described the basic logistics of the interview, informed students that they would be compensated with a $20 Amazon gift card, and provided a link to a Qualtrics form where they could sign up to participate. In this pre-interview survey, students provided information about their current algorithms course to ensure they fit into our target demographic, their availability in order to schedule an interview, and additional course-related data to be used in our study analysis. We then corresponded with students via email to confirm a date and time for their interview. Neither the announcement nor the pre-interview survey mentioned that our study was focusing on DP, only mentioning the interview was for an algorithms study.

### *3.2. Conducting the Interview*

All interviews took place virtually on Zoom, with two interviewers from our research group and one interviewee. For each interview, the interviewers were unaffiliated with the university the interviewee was from to preserve anonymity and avoid bias. Each interview lasted at most 1 hour and 15 minutes.

We began each interview with an icebreaker phase where interviewers spent approximately five minutes introducing themselves to the student and chatting with them about anything unrelated to the study. We wanted students to feel comfortable talking with us, in the hopes that they would be more comfortable sharing their solutions aloud in the think-aloud interview. Next, we began the introductory phase in which we addressed anonymity in the study, explaining that the student's professor would not be informed of their participation in the study, and established the logistics of the interview.

With few exceptions, each student was given a sequence of three questions to work through, out of a set of five possible questions. These five questions were taken from Zehra et al. (2018), with slight modifications (as detailed in Section 3.5) for clarity. Questions 1-3 are DP problems; questions 4 and 5 do not require DP in the solution.

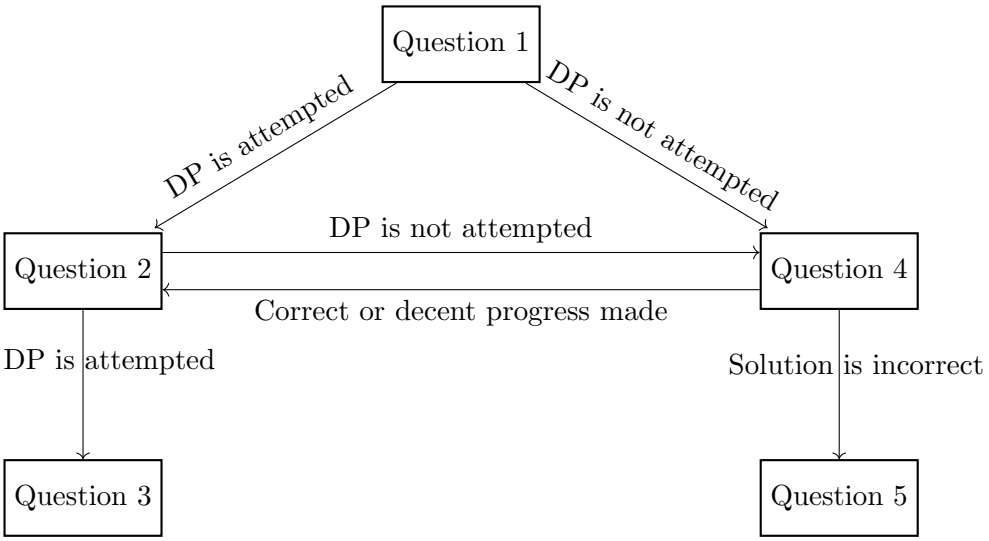


**Figure 1.** Flowchart for which 3 questions were asked during the interview

Refer to Appendix A for the full questions.

As illustrated in Figure 1, the sequence of questions was determined based on the student's progress on each question. Every student started with Q1. If the student correctly identified DP as the programming paradigm and produced a somewhat correct solution, they moved on to the next DP problem (i.e. from Q1 to Q2 or from Q2 to Q3). If not, they moved on to Q4. They continued to Q2 if they made decent progress on or solved Q4, or Q5 otherwise. Q4 and Q5 were meant to be used as morale boosters, with Q5 being easier than Q4. This way, students would walk away from the interview feeling confident that they solved at least one problem correctly. There were occasional variations on this sequence due to timing constraints or rare interviewer errors that did not affect the integrity of the study.

The students were given the choice to work through the questions on a collaborative text document from codeshare.io or to connect a tablet to the Zoom meeting and work on their choice of whiteboard application. In both scenarios, the student shared their screen so that the interviewers could view their work. As the student worked through each problem, interviewers turned off their Zoom video and audio to avoid influencing the student. Interviewers only intervened to answer clarifying questions about the problem or ask the student to elaborate on their thought process if they fell silent for a long period of time. Interviewers would not, however, provide hints if the student got stuck.

Each interview ended with a reflection phase in which the interviewers asked open-ended questions to clarify the student's thought process or glean additional information about their understanding of DP. Students were emailed their compensation at the conclusion of the interview.

### *3.3. Tagging Data*

To preserve anonymity, each interview was transcribed by a researcher who was not affiliated with the university of the interviewee. The transcriber, while viewing a recording of the interview, created a verbatim transcript from the auto-generated Zoom

transcript with identifying details anonymized, which includes names of the interviewee, interviewers, and other individuals, course numbers, pronouns, school names and other information. The transcript also included images of the interviewee's work for each problem, as well as additional notes and images to provide context to visual events or unclear language and references made during the interview.

Once transcripts were anonymized, they were assigned to groups of three researchers. We ensured that no group tagged more than two transcripts together, and we balanced the number of times each researcher worked together to the greatest extent possible. In most cases, the tagging group included one researcher who had conducted the interview. For a particular transcript, one of the three taggers was assigned the role of verifier, checking that no identifying details were left in the transcript before passing it on to the other group members for tagging. Any transcripts found to have errors were fixed and then reassigned to a new group.

Initially, all taggers tagged the interview transcript individually. As the taggers read through the transcripts, they noted any misconceptions they found, including where in the transcript this misconception occurred, for which question number, and whether the student overcame the misconception. Taggers were not given a list of misconceptions or guidance on what to look for to determine if something was a misconception or not; they were expected to analyze the transcript independently in order to determine errors and misconceptions. This was done in order to determine if we could get similar results to Zehra et al. (2018) independently. To this end, we also ensured that the majority of researchers doing the tagging had not read the original study. Generally, taggers tagged misconceptions only if they felt that the student spent a significant amount of time towards using that misconception in a solution, which would vary from transcript to transcript. Taggers were advised not to tag multiple transcripts in one sitting and to take breaks in between tagging in order to discourage rater drift.

### *3.4. Determining Misconceptions*

After individual tagging was completed, the taggers for that transcript held a meeting to discuss the misconceptions they found for each question and where. If there were any differences between taggers, a discussion was held to determine if the misconception was valid and not a misunderstanding on the part of the tagger, or if it was significant enough to be noted. In addition, for each question, the taggers jointly graded the student's progress on a scale of 0 to 1 using an itemized grading rubric. This meeting was also held with a referee, a researcher who did not transcribe the interview. The referee provided their outside opinion if there were any major discrepancies for which the taggers could not come to a consensus on their own.

Once all the transcripts were tagged, the entire research group came together to review the misconceptions, pruning a few that were not DP misconceptions. Misconception codes were created by clustering similar misconceptions and defining distinct clusters. After all the codes were created, they were grouped into themes.

### *3.5. Differences From Original Study*

While we tried to stay true to the original study, we made some changes to the original methodology. Due to the COVID-19 pandemic, we were unable to conduct in-person interviews, so the interviews were conducted over Zoom. This allowed us to interview

a more diverse group of students, as we were able to interview students from a variety of schools without worrying about location and travel. Another difference is that once the interview started, we used a screen recording to capture the student's work, and we gave them the choice of writing on a tablet or using codeshare.io, since we could not ensure that each student had a tablet to use. This is in contrast to the original study in which each student used a smart pen.

There were also several changes made to the interview process. For the DP problems, Zehra et al. (2018) asked each student 3 questions: what the appropriate algorithmic design technique was, the subproblems and the recurrence that relates to this problem, and the pseudocode of a bottom-up DP solution, as these questions mirrored the "DP recipe" that was taught in their sample university's algorithms course. This was easier for the original study to standardize since it was hosted at one university. However, not every university follows the same "DP recipe," so we decided to forgo asking these specific questions. Instead, we avoided any explicit mention of DP, only informing students that this was an algorithmic study and asking them to provide their solutions. We also changed the order of the original questions, swapping Q2 (Consecutive 1's) with Q3 (Maximum Pixel Sum), which we also renamed to Minimum Pixel Sum due to a typo in the original paper. We swapped the order because we felt that students would be able to make more progress on Minimum Pixel Sum than Consecutive 1's and wanted to present that question earlier. Furthermore, we made the Minimum Pixel Sum prompt more specific so that each step could only move right or down. We did so as there is a natural greedy algorithm. Finally, we also included example inputs and outputs with each question for additional clarity.

Finally, a major difference between our study and the original is that the interviewers in the original study directed struggling students to a DP solution if they were unable to identify DP after a significant portion of time, but we decided against doing this. Because we had significantly more interviewers than the original study, we felt it would not be possible to standardize any hint giving. Therefore, we opted to give clarifications, but no hints. When we talked to the original authors, they also suggested that we give no hints, as this would influence student decisions and bias our data. Finally, we did not guarantee that students were given at least two DP problems, as they were in the original study. We did this to guarantee that every student walked away from the interview having completed at least one problem successfully.

We also made several changes to the original study's data analysis step. One such change was our decision to exclude the confidence scores altogether. We came to this decision because papers such as Enström and Kann (2017) call into question the confidence score and related self-efficacy scores as a valid metric for student understanding. When tagging the transcripts, since the original study only had two interviewers, both interviewers tagged all of their own interviews. In contrast, we had a subset of three researchers tag each interview, and we anonymized the interviews to reduce bias. We followed a similar technique for grading progress on a 0 to 1 scale. However, we chose a 0 to mean that the student did not attempt a DP solution on Q1-3, and a lack of understanding of the question for Q4 and Q5, whereas the original study had 0 mean a lack of understanding of the question for Q1-5.

## 4. Preregistration

For this special issue journal, our study was preregistered, meaning that it was accepted into the journal prior to when the study was conducted. To be selected for the journal,

we submitted a stage 1 manuscript that outlined our plan for the method and analysis. A panel of reviewers provided feedback on our proposal, allowing us to make necessary changes to the method before the collection of any data for the study.

Since preregistration with the journal, we have made some minor edits to our method. Our preregistered interview procedure did not have any major changes, with the only minor adjustment of moving the icebreaker section before asking for consent to record instead of after the recording began. Thus, this allowed the interviewees to feel more at ease to talk without the worry of being immediately recorded. In the original method, we did not know how we would provide compensation for interviewees, as we did not have enough funding to compensate each participant at the time of preregistration. We thought of offering extra credit for participation or creating a raffle for gift cards using the funds we did have. However, we were able to procure funding so we could provide each participant with a $20 Amazon gift card for participation.

We had a few more inconsistencies between our preregistration and the actual tagging and analysis procedures. Originally, we wanted a group of taggers who were unassociated with the interview to perform the tagging and transcription, after which all data would be anonymized and coded by a unique ID. However, we felt like this would be a poor use of time to have each interview transcribed multiple times. We also wanted to make sure there would be no implicit bias from the taggers if they could hear and see the interviewee while tagging. We instead chose a designated transcriber per interview, who was not an original interviewer and could provide an anonymized transcript with all the relevant work from the interviewee. This means that we could use the original interviewers as taggers, and as such we attempted to get at least one of the interviewers from a certain interview to tag it. We also originally stated that taggers would cluster the misconception codes into themes before meeting up with other taggers, yet we decided to form codes and themes after most of the interviews were finished tagging. Reviewers of the original proposal suggested that we should not use the misconception codes from Zehra et al. (2018), as that would interfere with the replication and may lead to confirmation bias in attempting to achieve similar results. To this extent, we also encouraged new researchers to not read the results of the original paper so that they could develop an unbiased set of misconceptions. This let us get a better idea of how many unique misconceptions there were without feeling forced into a pre-created set of misconceptions, which resulted in waiting to form the themes until we knew which codes we would use and how they may fit together.

When gathering student participants, we collected information on how many years participants have taken computer science courses at the university level. This was not stated in our original proposal and was an exploratory analysis, but we felt the information gathered through our analysis was worth reporting. Lastly, our proposal stated that we would provide an inter-rater reliability coefficient to examine the consistency between taggers. We ended up not including this data as we felt there were a lot of variables to take into account that could have yielded different results. For example, when tagging, taggers may have chosen different starting or ending segments for the same misconception, or they may have correctly identified the same segment but came to slightly different conclusions about what the participant was trying to suggest. Therefore, it was difficult to create an inter-rater reliability coefficient that encapsulated all of these small disparities, and we decided to forgo it, as it was also not relevant to the work of replicating Zehra et al. (2018).

| T# | M# | Misconception | # of Students |
|---|---|---|---|
| T1 | M1 | Student used a Brute Force Method. | 14 |
| | M2 | Student used a Greedy Algorithm. | 31 |
| | M3 | Student used Divide-and-Conquer. | 2 |
| | M4 | Student incorrectly used DP for a problem without overlapping subproblems. | 6 |
| T2 | M5 | Student failed to define correct base case(s). | 17 |
| | M6 | Student failed to identify the overlapping subproblems. | 13 |
| | M7 | Student failed to combine overlapping subproblems to generate optimal solution. | 5 |
| T3 | M8 | Student did not use memoization or had repeated function calls. | 11 |
| | M9 | Student stored information that is not used later during the algorithm. | 13 |
| | M10 | Student did an extra unnecessary process. | 1 |
| T4 | M11 | Student did not use a standard undergraduate programming paradigm. | 5 |
| | M12 | Student used the Even-Odd approach on Q1. | 3 |
| T5 | M13 | Student had an incomplete understanding of the definition of DP. | 2 |

**Table 1.** Student Misconceptions organized into Themes

## 5. Results

For this section, we will detail the main quantitative results of our study. Each section answers one of the following research questions:

(1) Do students select the correct algorithm technique when presented with a DP problem?
(2) After properly identifying a problem as DP, what misconceptions about dynamic programming do students have and how common are they?
(3) Are certain DP misconceptions correlated with university, years in computer science, and expected grade in algorithms course?

We will also provide exact numerical results of the misconceptions encountered during our 65 interviews. It is important to note that our analysis is based on the total number of misconception occurrences, not misconceptions by a particular student. In other words, if a student demonstrated the same misconception in two different questions, both misconceptions were counted. However, because the original work uses the number of students for generating statistics, we will also provide that data, which can be seen in Table 1 along with how the misconceptions were divided into themes.

### *5.1. Choosing the Correct Programming Paradigm*

The first challenge in solving a DP problem is properly identifying it as such. Out of the 196 questions asked, 140 of the questions were Q1-3, which are optimally solved with

a DP approach. Out of these 140 questions, students attempted a non-DP solution 71 times. These students used a greedy technique (M2) most frequently, with 35 (56%) instances of this misconception. The second most frequent was a brute force (M1) approach, with 17 (27%) occurrences. Five (8%) times, students attempted to solve the problem in a manner that did not use a standard technique taught in an undergraduate algorithms course (M11). We also had three (5%) instances of the "Even-Odd" approach for Q1 (M12). In this answer, students simply created two possible solutions, one with all the even index coins and one with all the odd index coins. They then returned the set with the highest sum. Divide-and-conquer (M3) appeared for two (3%) occurrences. It is also important to note that, although not a misconception, students used a feasible non-DP approach (M0) 9 times, such as a graph algorithm for Q2. These numbers are not factored into the percentages as they present a valid approach to solving the problem. Refer to Table 2 for the complete results.

From this data, we can conclude that students most often conflate DP with problems that can be solved using a greedy heuristic. Intuitively, this makes sense as it is often difficult to tell when a greedy approach will not find the optimal solution. Intuition also backs a brute force method as the second most common non-DP approach since a brute force method is always guaranteed to find a correct solution. Use of both of these approaches imply that students struggled at identifying when a given problem has overlapping subproblems. It is also interesting to look at which questions students made certain misconceptions on. For example, in Q1, 64% of students who implemented the wrong paradigm used a greedy approach, while this number is 71% for Q2, and 0% for Q3. This may imply that students are more likely to conflate a greedy heuristic with problems of path nature versus a more standard constrained subset optimization problem. As for Q3, the 0% greedy percentage makes sense, as there is no natural way to define a greedy heuristic. This also explains why Q3 has a much higher percentage of brute force (80%), as students were unable to find a more efficient approach.

We also tracked which of these misconceptions were overcome during the interview process. Of these 62 instances of using an incorrect, non-DP approach, students overcame their misconception 13 (21%) times, ultimately choosing to use DP instead of the incorrect programming paradigm. Most commonly, students who used a brute force method overcame their misconception five out of 17 (29%) times. One out of five (20%) decided to use DP after initially not using a standard undergraduate programming paradigm. Six out of 35 (17%) times students overcame their attempt at using a greedy solution. For students who implemented a greedy solution, used a divide-and-conquer approach, or used the Even-Odd approach to Q1, this implies that students are not properly identifying counterexamples where their algorithm may fail. Finally, the higher percentage of students who changed away from a brute force technique suggests that students are more successful at finding more efficient solutions than finding counterexamples to incorrect solutions. For the full results, see Table 2.

### *5.2. Implementation of DP*

Out of the 140 DP questions asked, students decided on a DP solution but had misconceptions about its implementation 66 times. Students most commonly did not use the correct base case(s) (M5) or did not determine the proper overlapping subproblem (M6), each of which occurred 17 (26%) times. At a close second, there were 15 (22%) instances of students memoizing values that were not useful in future calcu-

| Q# | Misconceptions | M0 | M1 | M2 | M3 | M11 | M12 |
|---|---|---|---|---|---|---|---|
| **Q1** | **# Occurrences** | 0 | 5 | 25 | 2 | 4 | 3 |
| | **# Overcame** | 0 | 0 | 5 | 1 | 0 | 0 |
| **Q2** | **# Occurrences** | 8 | 4 | 10 | - | - | - |
| | **# Overcame** | 1 | 3 | 1 | - | - | - |
| **Q3** | **# Occurrences** | 1 | 8 | - | - | 1 | - |
| | **# Overcame** | 1 | 2 | - | - | 1 | - |
| **Total** | **# Occurrences** | 9 | 17 | 35 | 2 | 5 | 3 |
| | **# Overcame** | 2 | 5 | 6 | 1 | 1 | 0 |

**Table 2.** Misconceptions about selecting the incorrect programming paradigm for each DP question

| Q# | Misconceptions | M5 | M6 | M7 | M8 | M9 | M10 |
|---|---|---|---|---|---|---|---|
| **Q1** | **# Occurrences** | 9 | 9 | 2 | 9 | 7 | 1 |
| | **# Overcame** | 2 | 5 | 1 | 2 | 1 | 1 |
| **Q2** | **# Occurrences** | 7 | 4 | 3 | 2 | 1 | - |
| | **# Overcame** | 0 | 0 | 0 | 0 | 1 | - |
| **Q3** | **# Occurrences** | 1 | 4 | - | - | 7 | - |
| | **# Overcame** | 0 | 1 | - | - | 2 | - |
| **Total** | **# Occurrences** | 17 | 17 | 5 | 11 | 15 | 1 |
| | **# Overcame** | 2 | 6 | 1 | 2 | 4 | 1 |

**Table 3.** Misconceptions about the implementation of DP for each DP question

lations (M9). In 11 (17%) occurrences, students did not use memoization properly, unnecessarily repeating function calls (M8). There were five (8%) instances of students struggling to combine their subproblems to create the optimal solution at each step (M7). Full results can be seen in Table 3.

Students overcame 16 of the above instances of misconceptions. They most often overcame misconceptions about their incorrect subproblem, with six out of 17 (35%) eventually determining the proper subproblem. Four out of 15 (27%) times, students overcame their memoization of unnecessary values, and two out of 11 (18%) times, students overcame their misconception of not utilizing memoization. One out of five (20%) occurrences of incorrectly combining subproblems was overcome. Only two of the 17 (12%) of the instances of base case misconceptions were overcome.

Considering that the two most common misconceptions were problems with the base case and issues with overlapping subproblems, we see that students' biggest hurdle to fully solving a DP question is developing the recurrence relation. This seems to corroborate our earlier conclusion that students have trouble identifying overlapping subproblems. Beyond this, students also had significant issues properly applying memoization, with some students storing too much data and others not enough. This implies that students may not fully understand the structure of the recursive calls of their algorithm, as they fail to properly identify which function calls will be repeated. Finally, we see that students generally did quite well when it came to determining the optimal solution at a given step. Thus, the hardest part for students is identifying the overlapping subproblems. Once the subproblems have been correctly identified, students are generally able to combine them to find the optimal solution.

### *5.3. Correlation With Other Data*

In the pre-interview survey, we gathered additional data about the students including the university they were attending and the number of years they had been taking university-level computer science courses. Here, we examine the correlation between these statistics and the misconceptions students exhibited.

#### *5.3.1. University*

Our sample consists of students from 15 different universities across the US. See Table 4 for the breakdown of universities. A comparison of misconceptions exhibited by students of each university can be seen in Figure 2.

#### *5.3.2. Years Taking University-Level Computer Science Courses*

We opted to ask how many years students have been taking computer science courses at the university level instead of their year at university because some students may not have been taking computer science courses for their entire time there (e.g., if they transferred into a computer science major) or took university level courses elsewhere (e.g., at a community college or AP courses). We had six students with one year of computer science courses, 43 with two years, 13 with three years, two with four years, and one with five or more years in our sample. See Figure 3 for the analysis.

Students who had been taking CS courses for only one year struggled the most with choosing the correct programming paradigm for DP questions as well as defining the correct base cases. They also struggled with incorrect use of a greedy approach, which all students had difficulty with, particularly those with three years of CS courses. Students with two years of CS courses represented the majority of the sample and had the most trouble with identifying the overlapping subproblems and determining what information to store in memoization. In addition to using a greedy approach in place of DP, students with three years of CS courses had the highest proportion of students who did not utilize memoization and who attempted the Even-Odd method for Q1. We cannot draw any meaningful conclusions about students who have taken CS courses for four or more years because only three students represented this category and thus have left this data out of the figure.

### *5.4. Student Progress on Questions*

Based on a grading rubric per question, taggers scored the students' progress on each question they were given on a scale of 0 to 1. With these scores, we can look at the students' progress on, or correct conceptions about, these questions.

Zehra et al. (2018) examined the progress students made in Q1 (Coin Row) versus Q3 (Consecutive 1's). They speculated that, although the questions have "similar recursive decompositions," the "abstractness" of Consecutive 1's makes it more difficult to solve than Coin Row, which is a more "concrete" question. Our results support that hypothesis. Of the 26 students who received both questions, the average score on Coin Row was 0.78 while the average score on Consecutive 1's was 0.58. Furthermore, all of the students who received Consecutive 1's also received Minimum Pixel Sum, which is another "concrete" DP question, and their average score was 0.78.

We also considered whether the order that the questions were presented affected the students' performances on the DP questions. Everyone received Q1 first, and the only students who saw Q3 were those who received questions in the order Q1, Q2, Q3.

| University | # of Students |
|---|---|
| A | 1 |
| B | 6 |
| C | 1 |
| D | 3 |
| E | 1 |
| F | 7 |
| G | 1 |
| H | 3 |
| I | 21 |
| J | 3 |
| K | 4 |
| L | 2 |
| M | 1 |
| N | 5 |
| O | 6 |

**Table 4.** Codes for universities along with the number of students interviewed from the institution

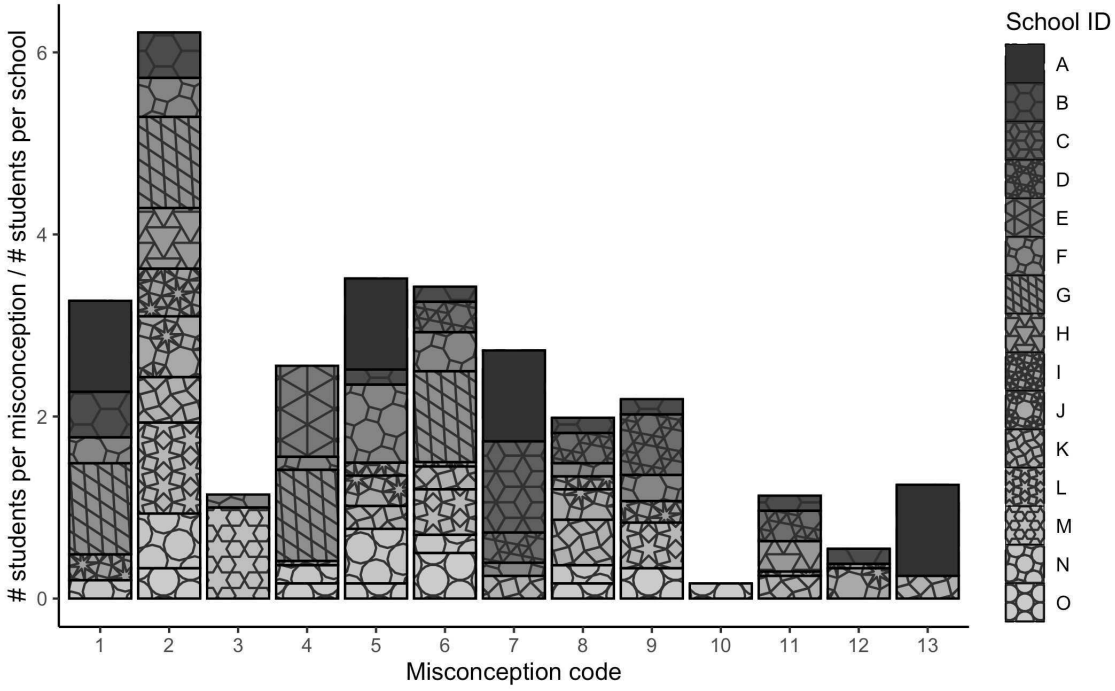


**Figure 2.** Normalized number of students for each misconception per University

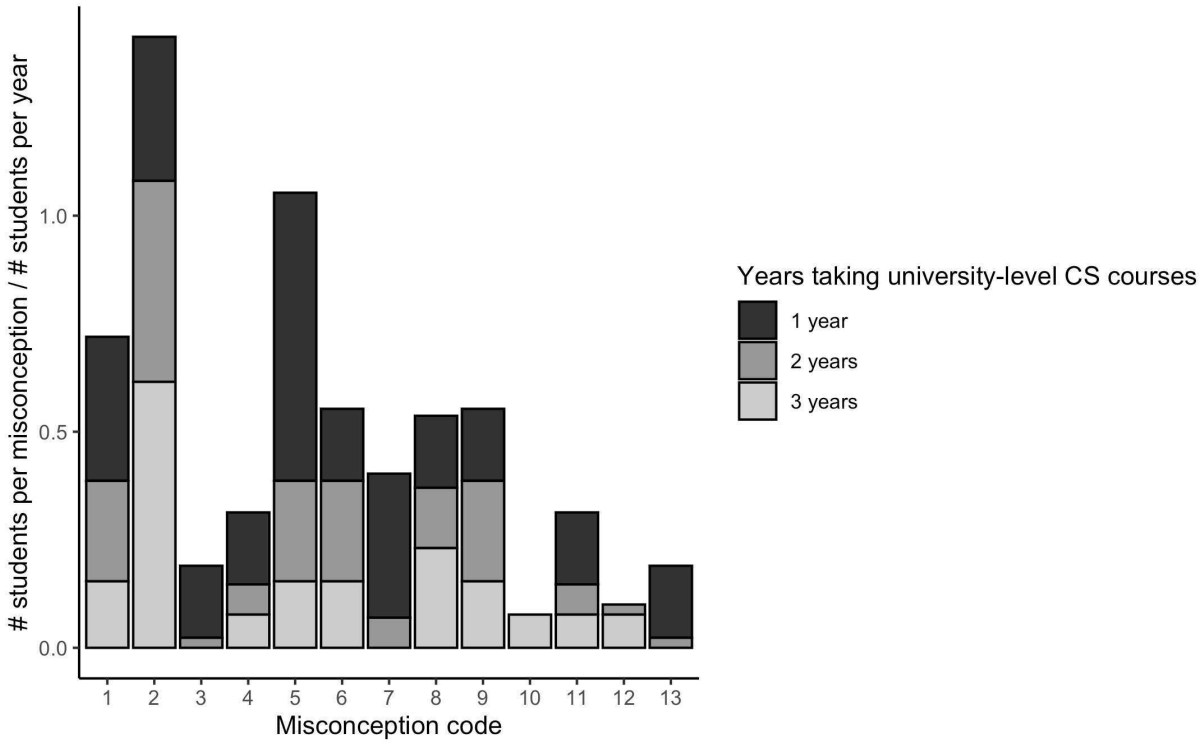


**Figure 3.** Normalized number of students for each misconception per year of CS courses

| Earlier Performance | Average performance on Q4/Q5 |
|---|---|
| Did well on Q1 | 0.87 |
| Struggled on Q1 | 0.62 |
| Did well on Q2 | 0.91 |
| Struggled on Q2 | 0.71 |

**Table 5.** Average score on Q4/Q5 correlated with performance on DP questions

Thus, we can only consider the effect of the order of the questions as it pertains to student progress on Q2.

In both Zehra et al. (2018) and our study, students who struggled on Q1 were then given Q4. If they produced a sufficient answer for Q4, they received a DP question next: Consecutive 1's for the original and Minimum Pixel Sum for ours. Zehra et al. (2018) observed that these students typically did well on the Consecutive 1's, despite struggling on the first DP question. In our results, the 34 students who got Q2 immediately after Q1 scored a 0.66 on average for Q2; by comparison, the 11 students who had the sequence Q1, Q4, Q2 scored a 0.36 on average for Q2. In contrast to Zehra et al. (2018), our data suggests that working on a non-DP question (Q4) before a second DP question (Q2) did not assist the student in performing better on the second DP question. However, since we presented the questions in a different order than the original study, we cannot examine the effects of solving Q4 before Consecutive 1's, the exact scenario considered in Zehra et al. (2018), and further research would be necessary to draw conclusions.

We also compared the grades students received on the DP questions (Q1 and Q2) versus the non-DP questions (Q4 and Q5) (Table 5). Based on our grading rubric, we

defined a student who "did well" on a DP question as having received a grade greater than 0.5 and a student who "struggled" on a DP question as having received a grade less than or equal to 0.5. The five students who did well on Q1 and received Q4 or Q5 had an average of 0.87 on Q4 and Q5 combined. The 31 students who struggled on Q1 and received Q4 or Q5 had an average of 0.62 on Q4 and Q5 combined. For the six students who did well on Q2 and received Q4 or Q5, the average grade was 0.91 for Q4 and Q5 combined. For the 14 students who struggled with Q2 and received Q4 or Q5, the average grade was 0.71 for Q4 and Q5 combined. Q3 is not included in this analysis because none of the students presented with Q3 were presented Q4 or Q5. We can see that the students who struggled with the DP questions scored significantly higher on the non-DP questions, even if they scored lower overall than the students who did well on the DP questions. This indicates that the students who struggled with DP tend to struggle less with designing and implementing algorithms for programming paradigms other than DP. It also indicates that students who perform well on DP questions are more likely to perform better on non-DP questions.

## 6. Discussion of Themes

We will now discuss the five major themes we found during our analysis of student misconceptions. We grouped our misconceptions into themes for two reasons. The first was to validate the results of Zehra et al. (2018). The second was to help provide a general area in which students tend to fail, as many of our observed misconceptions fall under broader misconceptions about DP. We also provided quotes from interviews to help demonstrate students' misconceptions. For a breakdown on how misconceptions were grouped, see Table 1.

### *6.1. Theme 1: Student did not identify when a DP solution is appropriate.*

Some students struggled with determining if a problem required a DP solution, resulting in an attempted solution with the incorrect programming paradigm. For the DP problems Q1-3, students attempted brute force (M1), greedy (M2), and divide-and-conquer (M3) solutions. A greedy approach was predominant in Q1 and Q2 where the solution finds a maximum or minimum value, whereas a brute force approach often occurred in Q3 where the solution requires a count of all valid strings. Divide-and-conquer was an uncommon choice but appeared in solutions for Q1.

In the majority of these cases, students thought an incorrect programming paradigm was the optimal method, sometimes even mentioning DP before discarding it. For example, one student immediately decided on a greedy approach for Q1:

> "...the goal is to pick up the maximum value coins, so I think about greedy algorithms right away, such that you never pick up two adjacent coins. Yeah, this totally seems like intuitively like a greedy algorithm."

In some other cases, students attempted DP but struggled with the implementation, causing their solution to fall under a different programming paradigm. In these interviews, students' intentions were to create a DP solution, but they inadvertently implemented a different paradigm. These students often recognized that their solution was incorrect but could not figure out how to fix it. One student tried to write an iterative DP algorithm for Q1 and, after struggling, tried to formulate the recurrence

relation by writing a recursive algorithm:

> "Well, [the recursive call] just returns the largest element in the new array. So... So... A perceived greedy approach...[laughs]...for this problem. Just because I don't know how to do it any other way."

Another misconception students exhibited was attempting to use DP for a problem without overlapping subproblems (M4). For Q4, an optimal solution uses either a linear traversal of the days or divide-and-conquer. A brute force algorithm also produces a feasible $\mathcal{O}(n^2)$ solution. However, DP does not further optimize the solution because the subproblems (the pairs of minimum and maximum days) do not overlap. One student thought about utilizing memoization to avoid recalculating subproblems, but failed to determine how the stored subproblems could be reused:

> "I think there is a fact in here somewhere that I did not acknowledge that would have allowed to like cut down which combinations I could have tried by half or more...I was thinking, maybe dynamic will be good, because I want to try every combination and probably store it in a backtrack array..."

Another student defaulted to DP when they observed that the solution of Q4 finds a maximum value:

> "...[Q4] kind of reminded me of [Q1] a little bit, when I first read it. Um, and it just seemed like the maximize earnings again so that's why I'd kind of gone with, uh, a recurrence relation."

### *6.2. Theme 2: Student did not recognize the correct recurrence.*

The second theme we identified was that students commonly struggled with the recurrence relation. We split this up into three misconceptions, the first of which was students failing to identify the overlapping subproblems (M6). In this case, students were unable to determine which subproblem to use for the recurrence relation. For example, in Q1, students knew that they could only pick every other coin, but struggled to recognize that they would need to pick the max of the current coin and the subproblem of the rest of the row starting from two coins away, versus the subproblem of the row starting from the next coin. One student was able to recognize the need for subproblems to solve coin row, but fell into this pitfall:

> "I'm not able to see the repeating subproblems. I'm not sure why I'm not able to see the repeating subproblems in this case, and that's not good."

Another misconception was that students failed to define the correct base case(s) (M5). This misconception came in a variety of forms. Some students did not define any base cases or forgot to include some of the base cases (e.g., only including the base case of taking one coin for Q1, but not including the base case with two coins for which they should take the larger coin). Another common base case error occurred in Q2, where some students would assume we would always start in a specific location, such as starting in the top left of the array.

Lastly, some students failed to properly combine the overlapping subproblems to generate the optimal solution (M7). The predominant example of this was in Q2, where students would incorrectly assume the solution would always return from the bottom corner of the array, or they would make a mistake in taking the minimum while generating their solution.

### *6.3. Theme 3: Student did not use the most efficient implementation of DP.*

The third theme covered students' inability to produce the most efficient DP solution for a problem. This theme describes students who were able to recognize the correct subproblems and recurrence but created an inefficient solution due to improper use of memoization or unnecessary additional actions.

The most prominent misconception within this theme arose when students did not properly use memoization in their solutions (M8). For instance, many students produced a correct recursive solution for a problem but failed to recognize the need for memoization. Many students who showcased this misconception did not mention DP or memoization, suggesting that they may not have considered whether to use memoization. However, there were some occurrences when students explicitly opted to not perform any memoization, thinking it would not improve their recursive solution. One student, who correctly implemented a DP solution for Q2, showcased this misconception in Q1:

> "So [Q2 has] two ways, one that goes right first and goes down and another one to which is pointless to go down and then go right. So in both ways, we would just do the calculation for this point two times, so it's just like redundant. We don't need to recalculate if you arrive here, its different...uh...its different ways. But for, uh, the coin row problem, I think even now we don't need to recover any point."

This misconception was also displayed in students who used memoization to store values from solved subproblems and yet did not refer back to those stored values in subsequent function calls, leading to a solution that recomputed the memoized values.

A number of students who made use of memoization in their DP solution displayed another misconception (M9), in which their memoization itself was not efficient and stored data that was not useful for the problem. This was often seen in the use of arrays that had an unnecessary dimension, such as a 2D array for Q1 or a 3D array for Q2, with the additional dimension to the array storing repeated values. We also had two students mention 2D arrays being frequently used in their class in reference to their incorporation of a 2D array in their solution for Q1.

Another misconception that caused inefficiency was performing unnecessary actions they thought would be necessary for DP (M10). This occurred only once, when a student sought to make use of a combination of a hash table and a priority queue to memoize their function calls (before eventually opting for a 1D array).

### *6.4. Theme 4: Student did not use a standardized programming paradigm.*

Theme 4 was reserved for students using irregular programming paradigms to solve the problems. In M11, students did not use any standard undergraduate programming paradigm. This misconception was almost exclusively related to Q1, for which students would try to find a pattern of what coins they should and should not take. The most common pattern for Q1 was an Even-Odd approach (M12), where students would return the maximum value of either all of the even or all of the odd-index coins. The other instance we had of this theme was in Q3, where a student briefly considered a non-deterministic finite automaton based solution.

### *6.5. Theme 5: Student had an incomplete understanding of the definition of DP.*

Theme five describes students who only showed a partial understanding of DP. This theme has one misconception code (M13) with the same name as the theme. Only two participants displayed theme five. The first student stated that recursion and dynamic programming were two explicitly different concepts. The second student stated that dynamic programming is exclusively used to solve optimization problems.

## 7. Comparison to Original Study

Zehra et al. (2018) found seven misconceptions which they grouped into three themes. Below, we discuss our findings in relation to the original study. All of the included percentages were calculated with the entire sample of students, not per theme. Note that we did not refer to the misconceptions from the original study when generating our own misconceptions to ensure our process was independent of the original. In this section, we will refer to the original study's misconceptions (M1, M2, M3, M4, M5, M6, M7) as O1, O2, O3, O4, O5, O6, O7 respectively, for clarity.

### *7.1. Theme 1: Subproblem Identification*

The first theme of Zehra et al. (2018) is Subproblem Identification, which consisted of two misconceptions. We encompassed this theme in our T2 and T3. The first misconception in the original study is "Student struggles to write iterative code to solve the problem"[1] (O1). This misconception is captured in M8 and M9, related to the absence or misuse of memoization. Zehra et al. (2018) reports that six students were "unable to justify their array's dimensions," especially students who chose a 2D array because of previous DP examples they had seen. We observed this same behavior in M9. O1 is also evident in our sample when students wrote a recursive solution but did not apply memoization to it correctly or at all, resulting in the inability to move from recursive to iterative code or recognize that a recursive solution would require some kind of memoization. 50% of students displayed this misconception in Zehra et al. (2018), whereas 34% of our sample had M8 or M9.

The second misconception in their first theme is "Student struggles to compose subproblems or define what is meant by 'subproblems' " (O2). This misconception is analogous to our M6, the failure to identify overlapping subproblems. In the original study, O2 was tied for the most common misconception, appearing with 57% of the students. By contrast, M6 only appeared in 20% of students in our study.

We observed discrepancies in the rate of students displaying misconceptions M6, M8, and M9 versus the corresponding misconceptions in Zehra et al. (2018). While the exact reason for this cannot be determined, it should be noted that we did not redirect students who used a different programming paradigm to DP for Q1-3. Thus, intuitively, one would expect to encounter less misconceptions about the implementation of a DP solution. In fact, 15 out of 65 (23%) of our sample received only one DP question because their sequence of questions was Q1, Q4, Q5. These students struggled with both Q1 and Q4 and would very likely have contributed to the frequency of M6, M8, and M9 if directed to use DP for any of Q1-3.

[1]Based on the wording in Zehra et al. (2018), we assumed all final dynamic programming solutions were to be written iteratively.

### *7.2. Theme 2: Solution Technique*

Zehra et al. (2018) defined their next theme as Solution Technique, containing three misconceptions. This theme is equivalent to our T1 and T4, which occurred when students failed to recognize that DP is the optimal programming paradigm. In both the original and our studies, a greedy approach (O5, M2) was the prevailing misconception, with 57% of the students in Zehra et al. (2018) and 48% of the students in our study falling under this category. Both studies also shared a misconception representing the incorrect use of divide-and-conquer (O4, M3). However, our rate of this misconception is not in line with the rate in the original study, with 29% of the original study's students, but only 3% of ours corresponding to this category. Zehra et al. (2018) found one other misconception: "Student uses sets for the DP problem" (O3). They elaborated on this misconception, explaining that it is "characterized by use of set terminology in expressing [the student's] solution." As an example, they mentioned that a couple of students "attempted to use finite state machines," which occurred only once in our sample. We classified this misconception as not using a standard undergraduate programming paradigm (M11). Similarly, we observed multiple students using an "Even-Odd" approach on Q1 (M12), which could potentially fall under O3 due to students generating two sets: even-index coins and odd-index coins. Finally, we discovered two completely novel misconceptions that the original study did not identify: students attempting a brute force approach (M1) and students attempting to use DP on Q4 (M4). It is surprising that these were not identified in the original study, especially in the case of M1 which occurred for 22% of students in our sample.

### *7.3. Theme 3: Defining a Recurrence*

The final theme from Zehra et al. (2018) is Defining a Recurrence, with two misconceptions under this theme. This is comparable to our T2, when students could not recognize the correct recurrence. The original study's first misconception was "Student struggles to write a recurrence" (O6). In our results, this misconception is represented jointly by M6, the failure to identify the subproblems, and M7, the failure to properly combine the subproblems into the optimal solution. M6 or M7 occurred for 28% of the students in our study while O6 occurred in 29% of the students in the original study. We also incorporated incorrect base case(s) (M5) into T2 because of its relevance to the recurrence relation. This was very common for us, with 26% of students demonstrating this misconception, but not mentioned in the original study. The final misconception in the original study is "Student fails to use previous experience with a recurrence to solve an analogous problem" (O7). Due to the diverse sample of students from several universities, we did not have a standardized way to assess students' previous experience and thus could not detect this misconception.

### *7.4. Discussion of comparison*

Overall, we have validated the themes and most of the misconceptions from Zehra et al. (2018). Because of the qualitative nature of this study, slight variations in the interpretations of themes and misconceptions occurred and were expected; regardless, the misconceptions at their core reveal similar patterns of student mistakes in DP. Some discrepancies existed in the rate at which students exhibited certain misconceptions. We observed lower percentages of students exhibiting M2, M3, M6, M7, M8, and

M9 compared to the analogous misconceptions in the original study (O1, O2, O4, O6). These differences may be due to the fact that our methods were slightly different, but the original study likely also didn't have a large enough and diverse enough sample size to accurately represent the population. The most significant addition to the original study was the novel occurrence of M1, M4, M5, and M13. M1 and M5 are particularly notable because of their frequency. The expanded demographic may have shed light on these additional misconceptions, as the variability in which professors present concepts may make their students more or less likely to have a given misconception.

## 8. Improvements for Future Similar Studies

Throughout the study, we gained insight into the process of running studies involving interviews and misconception identification. We included this section of suggestions for improvement in the hope that other groups aiming to discover misconceptions in computer science will avoid these issues.

First, it would have been better to determine which schools we wanted to reach out to and determine appropriate times to reach out to them at the beginning of the study. We incrementally asked schools for their involvement, gathering a list of schools, algorithm courses, and professors throughout the study. This incremental approach led to slow responses, taking much more time than if we had already prepared a list of schools and courses to reach out to. We also would get responses at varying points during their academic term, leading to some students having just learned DP, while others were already further along in their term, or already at finals. Furthermore, we were overly reliant on public information (catalogs and course descriptions) in order to determine which algorithms courses were considered part of our demographic. This caused us to unknowingly have students who had learned DP in a previous term sign up for interviews, causing these interviews to get thrown out. This reliance on public information also meant that we couldn't accurately determine how much detail each course went into for DP, which may have influenced the misconceptions students demonstrated. These are the reasons why we were unable to reach our original goal of recruiting 100 participants and instead got 71, after having thrown out 10 interviews.

Second, our interview process's reflection phase did not have a standard set of questions to ask interviewees. It was our belief that interviewers would come up with their own questions that would delve into why an interviewee reached their particular solutions. In practice, interviewers asked a wide range of questions that were not always aimed at why a DP misconception was made; therefore, we could not always accurately pinpoint the reason behind these misconceptions. In future studies, there should be a standardized set of questions that interviewers ask as it will more uniformly give insight into the students' thought processes.

Third, we could have had a better pipeline of interviews to transcripts. Rewatching the interviews to create anonymized transcripts took significant time. In retrospect, starting the transcribing process earlier could have saved us from spending several weeks solely transcribing interviews. This would also have allowed us to spend more time on the tagging portion of the study and prevented us from throwing out six more interviews we did not have time to transcribe.

## 9. Future Work

We believe we have validated the major themes from Zehra et al. (2018). As such, the future directions suggested by those authors would also build on our study. They identified three next steps to further understanding student misconceptions of DP. The first is to make an exclusively DP study; this would follow the same procedure but replace Q4 and Q5 with successively easier DP questions rather than questions from other programming paradigms. A second avenue of future work is to investigate how prior knowledge influences DP skill and to determine if misconceptions can be attributed to a misunderstanding of other algorithmic concepts. The third suggested step is a replication of these two studies with different DP problems. This would help determine if the specific problems chosen in Zehra et al. (2018) (and reused here) led to some of the difficulties observed.

Additional future work would expand on the foundation built by these two studies and lead towards a strong concept inventory for DP by following the pathways described by Taylor et al. (2020). In the case of DP, current research on misconceptions largely relies on the open-ended questions phase. Even with a large sample of students across a variety of universities and a large research team like ours, the open-ended questions phase gains information slowly. One improvement could be converting the prompts to multiple choice questions. In order to do this, future work could hold study sessions for groups of students who are taking algorithms courses. As outlined in Taylor et al. (2020), the students would be asked to answer both open-ended questions and multiple choice questions, with the latter asking them to explain their answers. The study sessions would serve as a source of data for the researchers, and the students would also be rewarded for their participation by having their misconceptions addressed in real time.

Lastly, these methods could be used to study and create concept inventories for other paradigms within algorithm design and analysis. Typical undergraduate algorithms courses focusing on algorithm design cover the divide-and-conquer paradigm as well as greedy. Similar work can also be done on more advanced topics like network flow or computational complexity, as to our knowledge, none of these topics have significant work on concept inventories.

## 10. Conclusion

In this paper, we have replicated and expanded on the work of Zehra et al. (2018). To do so, we conducted similar think aloud interviews on a larger population across multiple institutions. Analysis of these interviews has led to five themes of difficulties in solving DP problems, encompassing 13 specific misconceptions. The two most common themes found are that students have difficulty identifying DP problems as such (T1) and students do not recognize the correct recurrence relation (T2). For overall misconceptions, we found that the most common misconception students had was the use of a greedy heuristic in place of DP. We then compared these misconceptions to the misconceptions found in the original study. In doing so, we validated the misconceptions found in Zehra et al. (2018) while also finding new areas of difficulty for undergraduates solving DP problems, the two most common being that students attempted to use a brute force solution (M1) and students failed to properly define a base case (M5). These new misconceptions, when combined with the previously identified misconceptions, provide a fuller picture of the areas student struggle with when

solving DP problems.

To support these findings, a more comprehensive body of work detailing the several misconceptions found in students' lines of thinking using DP can be further developed, which would aid algorithms educators in resolving common misconceptions their students hold. We hope this spurs more research in DP education as well as in education of other programming paradigms, culminating in the creation of a concept inventory for algorithms.

## 11. Acknowledgements

We would like to thank Leo Porter, who first introduced Michael to the notions of concept inventories and to research about them in computer science. Professor Porter also helped our initial understanding of the area by providing references and introductions to other researchers doing work in the field. When we decided to build on the work of Zehra et al. (2018), we were introduced to Shamama Zehra, Aishwarya Ramanathan, Larry Yueli Zhang, and Daniel Zingaro, who we would also like to thank; their help in understanding the methodology behind their paper was extremely valuable. We also thank Jan Vahrenhold for helpful discussions. We thank the students who participated in this study and their professors who helped us with recruitment.

We also thank UCI's Academic Senate Council on Research, Computing and Libraries (CORCL) for the research fund that enabled us to provide gift cards as incentives for participants.

## References

Danielsiek, H., Paul, W., and Vahrenhold, J. (2012). Detecting and understanding students' misconceptions related to algorithms and data structures. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, page 21–26, New York, NY, USA. Association for Computing Machinery.

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

Enström, E. and Kann, V. (2017). Iteratively intervening with the "most difficult" topics of an algorithms and complexity course. *ACM Trans. Comput. Educ.*, 17(1).

Erickson, J. (2019). *Algorithms*. Independently published.

Farghally, M. F., Koh, K. H., Ernst, J. V., and Shaffer, C. A. (2017). Towards a concept inventory for algorithm analysis topics. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, page 207–212, New York, NY, USA. Association for Computing Machinery.

Goodrich, M. T. and Tamassia, R. (2014). *Algorithm Design and Applications*. Wiley Publishing.

Hamouda, S., Edwards, S. H., Elmongui, H. G., Ernst, J. V., and Shaffer, C. A. (2017). A basic recursion concept inventory. *Computer Science Education*, 27(2):121–148.

Karpierz, K. and Wolfman, S. A. (2014). Misconceptions and concept inventory questions for binary search trees and hash tables. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 109–114, New York, NY, USA. Association for Computing Machinery.

Paul, W. and Vahrenhold, J. (2013). Hunting high and low: Instruments to detect misconceptions related to algorithms and data structures. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, page 29–34, New York, NY, USA. Association for Computing Machinery.

Porter, L., Zingaro, D., Lee, C., Taylor, C., Webb, K. C., and Clancy, M. (2018). Developing course-level learning goals for basic data structures in cs2. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 858–863, New York, NY, USA. Association for Computing Machinery.

Taylor, C., Clancy, M., Webb, K. C., Zingaro, D., Lee, C., and Porter, L. (2020). The practical details of building a cs concept inventory. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 372–378, New York, NY, USA. Association for Computing Machinery.

Zehra, S., Ramanathan, A., Zhang, L. Y., and Zingaro, D. (2018). Student misconceptions of dynamic programming. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 556–561, New York, NY, USA. Association for Computing Machinery.

## Appendix A. Interview Questions

In this section, we provide the interview questions exactly as used in the interviews.

### *Question 1, Coin Row:*

There is a row of n coins whose values are positive integers c1, c2, . . . , cn . The goal is to pick up the maximum-value coins such that you never pick up two adjacent coins. Your solution should be feasible for input of size n = 1000.

Example:
—
Input: [7, 15, 12]
Solution: 7 and 12 = 19

Input: [10, 4, 4, 10, 4, 4, 10]
Solution: 10 and 10 and 10 = 30

### *Question 2, Minimum Pixel Sum:*

Given an n by m 2D image where each pixel can be represented by a value that is 0 or 1, (where n is the number of rows and m is the number of columns) find the minimum sum of pixel values of a connected path of pixels. The path must begin somewhere in the first row of the image and end somewhere in the last row of the image, and each move can only go down or to the right.
$[1 <= n <= 1000][1 <= m <= 1000]$
Your solution should be feasible for input of size n times m = 100000.

Example
—
Input:
$[[0, 1]$,
$[1, 0]$,
$[1, 0]]$
Solution: Sum = 1. (0,0) -> (0,1) -> (1,1) -> (2,1)

Input 2:
$[[1, 0, 1, 0]$,
$[0, 0, 0, 0]$,
$[0, 1, 0, 1]]$
Solution: Sum = 0. (0,1) -> (1,1) -> (1,2) -> (2,2)

***Question 3, Consecutive 1's:***

Design an algorithm to find the number of binary strings of length N that do not contain two consecutive 1's. Your solution should be feasible for input of size N = 1000.

Example
—
Input: N = 3
Solution: 5
Explanation of Solution: [000] [001] [010] [100] [101] all work, but [110] [011] [111] do not

***Question 4, Stock Market:***

Given a sequence of historical daily prices for a stock on n consecutive days, find the days on which someone should have bought the stock and then sold the stock so as to maximize earnings. On each day, you can do nothing, or buy the stock if you don't own it, or sell the stock if you do own it.

Note: (Can only buy and sell once, and the "buy" day must be before the "sell" day)

Your solution should be feasible for input of size n = 1000.

Example
—
Input: [2, 5, 7, 1]
Solution: Buy on day 1, sell on day 3
—
Input: [5, 2, 4, 8, 5, 6, 7]
Solution: Buy on day 2, sell on day 4

***Question 5, Peak-Finding:***

Given an array of n numbers that we know ascends up to some maximum value and then descends until the end of the array, find the maximum element of the array. Your solution should be feasible for input of size n = 1000.

Example
—
Input: [1, 3, 15, 17, 2]
Solution: 17