An Analysis of Iterative and Recursive Problem Performance

Madeline Endres
endremad@umich.edu
University of Michigan, CSE

Westley Weimer
weimerw@umich.edu
University of Michigan, CSE

Amir Kamil
akamil@umich.edu
University of Michigan, CSE

ABSTRACT

Iteration and recursion are fundamental programming constructs in introductory computer science. Understanding the relationship between contextual factors, such as problem formulation or student background, that relate to performance on iteration and recursion problems can help inform pedagogy. We present the results of a study of 162 undergraduate participants tasked with comprehending iterative, recursive, and tail-recursive versions of CS1 functions. First, we carry out a task-specific analysis, finding that students perform significantly better on the iterative framings of two problems with non-branching numerical computation and significantly better on the recursive framing of another that involves array classification ($p \leq 0.036$). Second, we investigate differences in the most common student mistakes by program framing. We find that students were more likely to produce wrong answers with incorrect types or structures for recursive and tail-recursive program versions. Finally, we investigated correlations between programming performance and background factors including experience, gender, ethnicity, affluence, and spatial ability. We find that the factors relevant to explaining performance are similar for both iterative and recursive problems. While programming experience is the most significant factor, we find that spatial ability, gender, and ethnicity were more relevant for explaining performance than affluence.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; **CS1**; **Student assessment**.

KEYWORDS

iteration, recursion, undergraduate CS, CS1

ACM Reference Format:
Madeline Endres, Westley Weimer, and Amir Kamil. 2021. An Analysis of Iterative and Recursive Problem Performance. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432391

1 INTRODUCTION

Iteration and recursion are fundamental concepts in computer science and learning them is critical for students in introductory programming courses. The comprehension of iteration and recursion has been studied extensively by the computer science education research community [1, 6, 8, 19, 20, 30, 32, 39]. However, students still struggle with both concepts [4] and instructors still struggle to teach them effectively [6]. Indeed, surveyed CS1 instructors identified recursion and loops as two of the most problematic areas to teach [6]. Furthermore, while many amelioratory teaching techniques for iteration and recursion have been proposed, educational interventions typically benefit from the early identification of struggling students [25]. There also remains debate in the computer science community over which (if any) programming tasks students find easier when represented recursively or iteratively [1, 20].

We present the results of a study with 162 CS undergraduates in which equivalent programming problems were presented iteratively to some students and recursively to others. Our study expands on previous work in several ways. First, our study is larger in scope; related work studying student ability on recursive and iterative problem framings regardless of student preference considers at most two programming tasks while we consider six [1, 20, 30]. Second, we include an analysis of common student errors to give insight into why students may struggle. Finally, we investigate correlations between background features (e.g., experience, gender, affluence, etc.) and student performance [22] on iterative and recursive problems. While student *mistakes* may differ between iterative and recursive framings, we find that the *factors* that predict success remain the same between them. To the best of our knowledge, this is the first work investigating correlations between background features and performance on iteration and recursion.

We focus on three main research questions. First, we investigate if the framing of specific CS1-level functions as recursive or iterative significantly affects student performance. Next, we question if the most common errors differ depending on program framing. Finally, we ask which demographic and background features correlate with student success. The main contributions of this paper are:

- A human study with 162 CS undergraduates investigating performance on iterative and recursive function versions.
- An analysis of how high-level program features (e.g., whether or not a program contains a symmetric array) correlate with student performance on iterative and recursive versions.
- An analysis of differences in student mistakes between iterative and recursive problem versions.
- An analysis of how certain background features (e.g., gender, affluence) correlate with iterative and recursive performance.

2 BACKGROUND: ITERATION & RECURSION

To place this work in context, we summarize prior work on student differences when learning iteration and recursion. In the computer science education literature, there is extensive research into student performance on iteration and recursion, both in tandem and individually. For instance, there have been several studies investigating

SIGCSE '21, March 13–20, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432391

methods for teaching and learning recursive programming (e.g., example-driven learning [39]) or exploring the effects of teaching iteration or recursion first; see McCauley *et al.* for a survey [19].

Prior work has also investigated student preference for iterative or recursive representations, both in general and for specific tasks. For example, Sulov *et al.* found that students programming in C generally choose iterative approaches for all six of their tasks [32]. However, they also found that, in most cases, the success rate of those who chose recursion was higher than those who chose iteration. Similarly, Esteero *et al.* observed a student preference for iteration. In contrast, however, they found that those students who chose iteration generally performed better overall [8]. Such divergent results suggest the presence of a confound (i.e., an overlooked feature beyond preference or choice that explains or predicts variance in student performance) and thus underscore the need for additional investigation into problem-specific preferences as well as for the sort of formal modeling we propose.

Benander *et al.* specifically explored differences in performance on iteration and recursion regardless of student preference [1]. They found that, for one list-based task, student comprehension was significantly higher for a recursive representation. However, for another non list-based task, there was no significant difference. They also found that students who understood the problem were significantly faster to come to their answers with the recursive representation for both problems. However, a replication by McCauley *et al.* found no difference in comprehension on the list-based task, underscoring the need for further investigation [20].

Also investigating performance on iterative and recursive program framings, Sinha *et al.* used the "cognitive fit framework" to explore two tasks as a function of problem representation and programming language (PASCAL or LISP) [30]. Their results varied by programming language: for LISP, performance was higher on recursive represented tasks, while for PASCAL, there was no significant performance difference. To ease applicability to students just starting CS1, we do not consider multiple languages. Instead, we present all of our programming stimuli in C++, an imperative object-oriented language commonly taught in CS1 or CS2 [7].

Finally, we note that all of these studies exploring student performance on iterative and recursive tasks regardless of student preference have included at most two programming tasks. The small scope of these experiments combined with apparently-contradictory results indicate that more investigation is needed. In general, prior work has focused more closely on language design, understanding comprehension, and student paradigm preference. To the best of our knowledge, no previous study has investigated correlations between non-programming or task-based features and a student's performance with iteration and recursion.

3 BACKGROUND: CORRELATIONS

In this paper, we also explore correlations between participants' demographic and background features and their programming success on the recursive and iterative problems. In particular, we consider past programming experience, gender, ethnicity, spatial reasoning ability and socioeconomic status. All of these features are simple to collect and largely institution agnostic. A deeper understanding of the factors influencing success on recursive and iterative problems could aid preemptive identification, and thus assistance, of students likely to struggle with either concept. We briefly overview prior work for each feature and its connection with programming.

Past Programming Experience: With the expansion of computer science in high school curricula, students enter CS1 and CS2 with a wide range of past programming experiences [15]. Unsurprisingly, increased prior experience predicts better performance in introductory CS courses, both empirically and as perceived by enrolled students [33]. The benefits of prior experience for students in introductory computer science can be significant: Wilcox and Lionelle found that CS1 students with prior experience outperformed students without experience by over 6% on exams and 10% on programming quizzes [40]. However, they also found that the prior-experience advantage diminished over time with no significant observable difference at the end of CS2.

Gender: There is a persistent gender representation gap in computer science at both the collegiate and professional levels [5, 13, 21, 36]. For example, only 18% of computing bachelor's degrees in the United States are awarded to women [5, 21], and only 37% of Chinese collegiate computer science students are women [10]. There is also some evidence of the existence of a gender-based performance gap in computer science. For example, a recent ITiCSE working-group survey noted that multiple studies have found that, when compared to male students, "females have lower self-efficacy when enrolled in an introductory programming course", a fact that "can affect [CS] performance and persistence" [18]. Several other studies, however, have failed to find a significant gender-based performance gap after controlling for year-level, academic ability, and other preparatory factors [16, 41]. Regardless of the existence of a computer science performance gap, the relationship between gender and CS success is nuanced, influenced by a multitude of societal and cultural factors [36]. For example, multiple studies have found that males and females have disparate computer experiences, attitudes, and usage patterns from a young age [2, 29]. Furthermore, CS1 students with programming experience are disproportionately male [27, 40]. This has a cumulative effect, as many female students are not only at a disadvantage in individual classes, but also often take key classes later, making it harder to declare and complete a computer science major [27]. We consider self-reported gender as a rough yet easy-to-collect proxy for cultural, social, and environmental factors.

Ethnicity: Prior work has also found significant differences in self-perception and performance in computer science depending on ethnicity [36]. For example, while exploring success factors in 65 novice programmers, Katz *et al.* observed that the 18 African American participants in their study had significantly lower percentages of error-free program compilations and executions [14]. Due to the African American participants' "significantly lower SAT scores", the authors attribute their lower performance to inadequate college preparation [14]. Furthermore, ethnicity and gender can interact in nuanced and potentially predictive ways. For example, in 2015, 32% of Asian Software Engineers in Silicon Valley were female compared with just 12% of White Software Engineers [13]. We consider self-reported ethnicity as a rough yet easy-to-collect proxy for a multitude of cultural, social, and environmental factors.

Affluence: There is also evidence that socioeconomic status may impact computer science success [22] through many factors. For example, researchers have observed socioeconomic-based differences in computing access and usage patterns of college students [22] to children as young as six months old [3]. Socioeconomic status is also often related to disparities in students' academic preparation for college. For instance, Long *et al.* found that 48% of low-income students were prepared for college-level math as compared to 64% of all high school students [17]. Furthermore, Terenzini *et al.* found that first-generation college students, who were disproportionately from low-income families, had "weaker cognitive skills in reading, math, and critical thinking" [34]. College students' incoming math and reading abilities have both been posited as predictors for success in computer science [11, 26].

Spatial Ability: Spatial reasoning is a blanket term for the capacity to understand and reason about spatial relationships. It is a major factor in performance in fields such as mathematics [38], natural sciences [43], and engineering [31]. It also positively correlates with programming performance [23]. Furthermore, Parker *et al.* found spatial reasoning to be a better mediating variable for socioeconomic gaps in computer science than access to computing [22]. On a neurological level, Huang *et al.* used medical imaging techniques to find that similar parts of the brain are recruited to solve spatial problems and data structure problems [12].

4 APPROACH

This section describes our data collection and analysis approaches. We conducted an online study with 162 CS undergraduates. The key aspect of the study is its controlled presentation of the same conceptual problem to different students in different ways (e.g., some students are shown an iterative problem while others see a recursive version of that same problem). We also collect simple background and demographic features (see Section 3). We then describe the structure of the experiment (Section 4.1), the programming stimuli (Section 4.2) and participant recruitment (Section 4.3). Finally, in Section 4.4 we describe our statistical methods.

4.1 Experimental Stimuli and Structure

Our online study consisted of three main parts: a free response portion where participants wrote the output of functions, a multiple choice test assessing spatial reasoning ability, and a demographics questionnaire. The order of the spatial reasoning and programming sections were randomized, while the demographics questionnaire was always given last to avoid biasing participants. Both the spatial reasoning ability and programming ability sections were timed, and the survey was intended to take participants at most 35 minutes.

In the programming portion, participants were shown 6–7 C++ functions where half contained an iterative structure and half contained a recursive structure. Participants were given a maximum time limit of 3 minutes per programming question, and individual participants were not shown the recursive version and the iterative version of the same function. The programming stimuli are described in greater detail in Section 4.2.

To assess spatial skills, participants took the *Paper Folding Test*, a validated test of spatial ability [42]. This test consists of 20 multiple choice questions split into two parts of 10 questions. Each half has a time limit of 3 minutes. An example problem is shown in Figure 1.

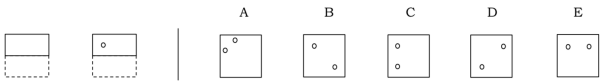

Figure 1: Paper Folding Test example: Participants select the choice on the right which corresponds to an unfolded hole-punched paper on the left. Here, the correct answer is "C".

Table 1: Program Descriptions. R, I, and T indicate if there is a recursive (R), iterative (I), and/or tail-recursive (T) version.

Name	Program Description	R	I	T
Digit	Returns sum of the input's digits	✓	✓	✓
Exp	Exponentiation by squaring	✓		
Fiblike	Variation on Fibonacci sequence	✓	✓	✓
Gcd	Greatest common divisor	✓	✓	
Palindrome	Evaluates if array is a palindrome	✓	✓	
Reverse	Reverses an array	✓	✓	
Triangle	Prints a triangle	✓	✓	

Finally, students were asked a series of demographics questions. These questions included participant gender, ethnicity, and programming experience (including both years programming and institution-specific courses). To assess socioeconomic status, participants also completed a version of the validated *Family Affluence Scale* (FAS III) [35]. This assessment consists of a small number of multiple choice questions about an individual's material belongings and living situation. These multiple choice questions included "Does your family own a car or other motorized vehicle?", "How many computers (including laptops and tablets, not including game consoles and smartphones) does your family own?", and "How many times did you or your family travel out of your home state for holiday/vacation last year?". [1]

4.2 Programming Stimuli

Programming tasks were selected from a corpus of seven functions modeled on common CS1 exam questions at the University of Michigan. Students were shown either an iterative, recursive, or tail-recursive C++ implementation of each function. Students were then asked to type either the output or return value of the function for a specified input. These inputs (and their corresponding outputs) were the same across function implementations. That is, the iterative, recursive, and tail-recursive questions and answers were entirely the same except for the iterative or recursive structure shown in the source code. Students were limited to 3 minutes per question. All functions had recursive implementations; six admitted iterative versions and two admitted tail-recursive versions with the addition of a helper function. Table 1 details each function and its implementations. As an example of our stimuli structure, Figure 2

[1] Copies of the complete survey instrument used in this study are at https://github.com/CelloCorgi/SIGCSE2021-IterativeAndRecursive

```
int func(int n) {
    if (n <= 1) { return n; }
    int a = 0;
    int b = 1;
    for (int i = 1; i < n; ++i) {
        int c = b - 2 * a;
        a = b;
        b = c;
    }
    return b;
}
```

```
int func(int n) {
    if (n <= 1) { return n; }
    else {
        return func(n - 1) - 2 * func(n - 2);
    }
}
```

```
int func_helper(int n, int i, int a, int b) {
    if (i == n) { return b; }
    else {
        return func_helper(n, i + 1, b, b - 2 * a);
    }
}
int func(int n) {
  if (n <= 1) { return n; }
  else {
     return func_helper(n, 1, 0, 1);
  }
}
```

Figure 2: The iterative (top), recursive (middle), and tail-recursive (bottom) versions of the Fiblike function. Participants respond to a prompt such as "Please write the return value of `func(3)` in the box below:".

presents all three versions of Fiblike, a Fibonacci-esque function. Responses were scored using regular expression matching.[2]

4.3 Participant Recruitment

Participants were recruited via email during November of 2019. All participants were current undergraduates at a large public research institution, and all had completed (or were close to completing) at least one semester of CS coursework in C++. Compensation included several options for an Amazon gift card of up to $100.

Out of 5,638 emails sent, 306 students clicked on the study link. The study was completed by 175 students (57.2%). To ensure response quality, responses were removed from the data if either their overall study completion time was less than two standard deviations below the mean, or if they scored worse than, or equivalent to, random on the spatial reasoning test. In the end, 162 responses passed these quality filters. Of these responses, 44 were female, 114 were male, 2 were non-binary / non-conforming, and 2 preferred not to say. Regarding ethnicity, 62 were Asian, 3 were Black / African, 86 were Caucasian, 4 were Hispanic / Latinx, and 7 were other or preferred not to say. Self-reported programming experience ranged from 0 years to 10 years, with half of participants reporting 2 years or less. Compared to the institution's departmental demographics, our responses featured slightly more non-men participants and slightly fewer members of underrepresented racial or ethnic groups.

[2]Scored data at https://github.com/CelloCorgi/SIGCSE2021-IterativeAndRecursive.

Table 2: Percent of student answers correct on individual problems. p-values reported for differences with $p < 0.05$.

Name	R	I	T	p-value
Digit	46.0%	59.7%	46.3%	0.027 (R vs I) 0.036 (T vs I)
Exp	28.1%			
Fiblike	53.7%	48.5%	47.2%	
Gcd	45.3%	57.3%		0.025
Palindrome	86.3%	66.3%		<0.0001
Reverse	35.1%	34.4%		
Triangle	74.6%	71.7%		

4.4 Statistical Methods

We applied several statistical tests to the participant data: the paired t-test for binary data, Spearman correlations, and analyses of variance (ANOVAs). We choose to use Spearman correlations instead of the more common Pearson correlations because Spearman tests for any monotonic correlation, not just linear correlations. We choose to use the ANOVA to look at feature interactions as it is a generalization of the Student's t-test for more than two populations. We use the implementations of ANOVA and Spearman found in the scikit-learn API [24] and SciPy [37].

5 EXPERIMENTAL RESULTS

In this section, we report the results of our experiment. We organize our analysis of the data around three research questions:

- RQ1: Does the framing of specific functions as recursive or iterative significantly affect student performance?
- RQ2: Do the most common student errors differ depending on the framing of a function as recursive or iterative?
- RQ3: What demographics and background features correlate with student success on iterative and/or recursive problems?

5.1 RQ1: Program Framing Effects: Correctness

We analyze the effects of problem framing (iterative, recursive, tail-recursive) on performance. First we give an overview of significant results, and then we explore differentiating program features.

Three out of six problems showed a statistically-significant difference ($p < 0.05$) for student performance between iterative and recursive framings. For two of the programs, Digit and GCD, students performed significantly better on the iterative version. For Palindrome, students performed significantly better on the recursive version. For the two additional tail-recursive implementations, we did not find any significant differences between performance on the tail-recursive and recursive implementations. For Digit, however, there was a statistically-significant difference between student performance on the tail-recursive and iterative versions, with students performing better on the iterative version. Table 2 presents a detailed breakdown of significant results.

Qualitatively, we note several features relevant to the programs with statistically-significant differences. First, Digit and Gcd, the two programs with significantly better student performance on the iterative version, are the only two programs with non-branching numeric computations. By contrast, the other two numerical programs,

Exp and Fiblike, both involve branching computation. Second, Palindrome, the program with statistically better student performance on recursion, is one of two programs involving arrays and the only one involving a symmetric array. These results align with previous work proposing that recursion might be more natural for students completing list manipulation tasks [30].

> Students performed significantly better on the iterative version of 2/6 programs and better on the recursive version of 1/6. Better performance on the iterative versions was associated with non-branching numeric computation.

5.2 RQ2: Program Framing Effects: Errors

We analyze differences in participant mistakes depending on program framing. The results of our analysis can be found in Table 3. In general, however, we find that the program framing affects the kinds of mistakes that students make; for all six problems with both iterative and recursive framings, the most common student mistake was different for each version. For example, for the Palindrome program, for which the correct answer is the boolean "false", the most common wrong answer for the iterative version is the boolean "true", while the most common wrong answer for the recursive version is $(\{6, 2, 4, 3, 10, 4, 2, 6\}, 6)$. Identifying and correcting student mistakes is very relevant in introductory classes [28]; beyond student accuracy, the disparate patterns of common mistakes observed in iterative vs. recursive vs. tail-recursive formulations are important for informing alternate guidance or remedial explanations.

When solving recursive and tail-recursive versions, participants generally struggled more with identifying the output's structure than when solving iterative versions. For instance, when solving the recursive version of Gcd, a function that returns a single integer representing the greatest common divisor of two input parameters, the most common incorrect student answer was the tuple (70, [_]). In fact, 19% of student answers for the recursive framing had this structure. For the iterative version, however, none of the incorrect answers exhibited that tuple structure. Instead, the most common mistakes were plain integers, the same type as the correct answer. We observe that for 4/9 recursive and tail-recursive problem versions, the most common incorrect answer differed structurally from the correct answer compared to just 1/6 iterative versions. Furthermore, the incorrect structural results are often in list or tuple form. While more research is needed, this may indicate that while students are successfully mentally tracing through the recursive process, they are struggling with collapsing the process into a single answer.

> Common student mistakes differed by problem framing. Students were much more likely to produce wrong answers with incorrect types or structures (e.g., tuples vs. single integers) for recursive and tail-recursive program versions.

5.3 RQ3: Significant Background Features

We investigate how demographic features correlate with both overall performance and also performance on iteration and recursion individually. We conduct an ANOVA for three models: scoring above 50% on iteration, scoring above 50% on recursion, and scoring above 50% on the programming test as a whole. We also present Spearman correlations (which do not assume data normality) between each demographic feature and overall programming performance. We consider the following features: experience, completion time, spatial ability, affluence, gender and ethnicity. We break down the categorical features gender and ethnicity using one-hot encoding, and we only present results on sub-features with at least twenty responses. The recursive model does not include scores on the additional tail-recursive versions. Our results are presented in Table 4.

For overall performance, self-reported experience was the only moderate-to-strong positive correlate ($\rho = 0.64$). This correlation is unsurprising; it is intuitive that students with more programming experience will perform better on a programming test. While we observed no other strong correlations, several features exhibited weak correlation ($\rho \geq 0.20$). In particular, spatial reasoning scores, self-identification as male, and self-identification as Asian all weakly correlated positively with overall performance. On the other hand, self-identifying as female correlated negatively with performance. Time to complete the test, affluence, and self-identifying as Caucasian were not significantly correlated with performance.

With respect to our ANOVAs, we find that only five features or sub-features were significantly important and distinct for any of the three models: experience, spatial ability, self-identification as female, self-identification as male, and self-identification as Asian. These are the same features with Spearman correlations greater than 0.20. In general, the same features are important for predicting performance on iteration, recursion, and overall; for all three models, the most important feature is experience followed by self-identifying as female. We also note that spatial ability was only a significant predictor for recursive performance. While spatial ability appears to be important (spatial skills p-values approached, but did not meet, our 0.05 threshold for all three models), a more in depth analysis of spatial reasoning as a driving factor between iteration and recursion differences is left for future work. Regardless, we find that spatial reasoning ability is a more important feature than socioeconomic status when predicting performance.

> Experience, spatial skills, gender, and ethnicity (but not socioeconomic status) were significant features for predicting student performance. In general, the same features are predictive of iterative, recursive, and overall programming.

6 LIMITATIONS

We now discuss generalizability, as well as threats to construct validity. First, we acknowledge that our selected iterative and recursive functions may not perfectly represent general recursive and iterative constructs encountered by CS1 students. We mitigate this, however, by testing with a larger number of programming stimuli than previous work. We also deliberately use a variety of function input formats and structures (e.g., array transformation, number accumulation, and visual printing). Furthermore, while prior work has considered recursion and iteration in multiple programming languages (e.g., [30]), it is not clear how well our results translate to other institutions using different languages or curricula.

Table 3: Three most common mistakes for each problem: "Total" contains the total number of responses. "Wrong Answer *n*" is the *n*th most common wrong answer for that question. Each percent column measures the proportion of all answers that were the corresponding wrong answer. All reported answers had at least 2 instances; blank answers are not reported.

Name	Correct Answer	Version	Total	Wrong Answer 1	%	Wrong Answer 2	%	Wrong Answer 3	%
Digit	9	I	67	131.4 ... †	6.0%	8	4.5%	0	4.5%
		R	37	135	16.2%	131.4 ...	13.5%	8	5.4%
		T	41	131.4 ...	9.8%	1	7.3%	0	4.9%
Exp	16	R	135	4	23.7%	2	9.7%	0	5.2%
Fiblike	-1	I	68	-3	17.6%	1	7.4%	0	7.4%
		R	41	0	13.0%	-2	4.9%	n/a	n/a
		T	36	3, 2, ... †	13.9%	1	11.1%	0	11.1%
Gcd	10	I	89	70	15.7%	30	10.1%	40	4.5%
		R	53	(70, ...) †	18.9%	30	13.2%	0	5.7%
Palindrome	false	I	89	true	14.6%	0	5.6%	n/a	n/a
		R	51	({6,2,4,3,10,4,2,6}, 6)	5.9%	true	4.0%	n/a	n/a
Reverse	9 6 7 3 2 1 4	I	96	2 1 7 3 9 6 4	4.2%	4 1 2 3 7 6 9	3.1%	9 7 2 3 1 6 4	3.1%
		R	57	9 7 2 3 1 6 4	3.5%	n/a †	N/A	n/a	n/a
Triangle	****	I	92	****	3.3%	*	2.2%	n/a	n/a
	***			****		*			
	**			****					
	*			****					
		R	63	****	6.4%	n/a †	N/A	n/a	n/a

† All incorrect answers reported are exact textual matches except for: (1) For all Digit types, "131.4 ..." represents a set of answers that all start with that prefix; (2) In Fiblike T, five students wrote comma-separated lists starting with "3, 2", a pattern not seen in Fiblike R and I; (3) For Gcd R, the most common student error was a tuple with the first component "70". We note that this tuple formation was not observed in iterative responses; (4) and (5) For the Reverse and Triangle recursive versions, there was only one duplicate wrong answer.

Table 4: ANOVAs for three models: participants scoring over 50% for recursive problems, over 50% for iterative problems, and over 50% overall. Each box contains a f-value / p-value pair. Significant results ($p \leq 0.05$) are highlighted in blue. We also include the Spearman correlations between each feature and performance on the programming test overall.

Model	Experience	Time	Spatial Skills	Affluence	Female	Male	Asian	Caucasian
> 50% Recursive	52.2 / <0.01	0.073 / 0.80	3.90 /0.05	1.30 / 0.25	8.44 / <0.01	3.83 / 0.05	3.60 / 0.07	1.66 / 0.19
> 50% Iterative	46.1 / <0.01	0.68 / 0.41	3.69 / 0.06	1.29 / 0.26	4.88 / 0.03	3.48 / 0.06	5.18 / 0.02	1.05 / 0.30
> 50% Overall	57.6 / <0.01	0.83 / 0.36	2.78 / 0.10	1.37 / 0.24	12.59 / <0.01	7.26 / <0.01	5.80 / 0.02	1.98 / 0.16
Spearman	0.64	0.11	0.20	0.10	-0.25	0.21	0.21	-0.14

We also consider two possible construct validity issues. First, as the test was online, some students may have used a programming environment or other external aid to solve the problems. While we explicitly instructed students not to do this, we further mitigate this threat by eliminating participants who completed the survey significantly faster than the mean. Second, our results rely on self-reported data. Previous studies have found that self-reporting can be unreliable (e.g., [9, 12]). We mitigate this threat by using previously-validated assessments, such as the *Paper Folding Test*, when possible.

7 CONCLUSION

We explore undergraduate students' success rates and error patterns on iterative and recursive programming tasks, two fundamental concepts in CS1 and CS2. We present the results of a human study of 162 undergraduate participants comprehending iterative, recursive and tail-recursive versions of a set of CS1 functions. We found that, in a statistically-significant manner ($p \leq 0.05$), **students perform better on iterative versions of two problems with non-branching numerical computation and better on the recursive version of another involving arrays**. We further **find that common student errors vary as the problem framing changes**. Students' mistakes were more likely to be of the wrong type or structure (e.g., a tuple vs. a plain integer) for recursive or tail-recursive framings. We also investigated correlations between programming performance and experience, gender, ethnicity, socioeconomic status, and spatial reasoning ability. We find that **the factors correlated with aggregate performance are similar for both iterative and recursive program framings**. While experience remains the most significant factor, we find that spatial reasoning ability, gender, and ethnicity were more correlated with programming performance than was socioeconomic status.

ACKNOWLEDGEMENTS

We acknowledge the partial support of the NSF (CCF 1763674).

REFERENCES

[1] Alan C. Benander, Barbara A. Benander, and H. Pu. 1996. Recursion vs. Iteration: An Empirical Study of Comprehension. *Journal of Systems and Software* 32, 1 (1996), 73–82.

[2] Zhihui Cai, Xitao Fan, and Jianxia Du. 2017. Gender and attitudes toward technology use: A meta-analysis. *Computers & Education* 105 (2017), 1–13.

[3] Sandra Calvert. 2005. Age, Ethnicity, and Socioeconomic Patterns in Early Computer Use: A National Survey. *American Behavioral Scientist* 48 (01 2005), 590–607. https://doi.org/10.1177/0002764204271508

[4] Jie Chao, David F. Feldon, and James P. Cohoon. 2018. Dynamic Mental Model Construction: A Knowledge in Pieces-Based Explanation for Computing Students' Erratic Performance on Recursion. *Journal of the Learning Sciences* 27, 3 (2018), 431–473. https://doi.org/10.1080/10508406.2017.1392309

[5] ComputerScience.org. 2020. *Women in Computer Science: Getting Involved in STEM.* ComputerScience.org. Retrieved February 20, 2019 from https://www.computerscience.org/resources/women-in-computer-science/

[6] Nell B. Dale. 2006. Most difficult topics in CS1: results of an online survey of educators. *SIGCSE Bulletin* 38, 2 (2006), 49–53.

[7] Stephen Davies, Jennifer A. Polack-Wahl, and Karen Anewalt. 2011. A Snapshot of Current Practices in Teaching the Introductory Programming Sequence. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 625–630. https://doi.org/10.1145/1953163.1953339

[8] Ramy Esteero, Mohammed Khan, Mohamed Mohamed, Larry Yueli Zhang, and Daniel Zingaro. 2018. Recursion or Iteration: Does it Matter What Students Choose?. In *SIGCSE*. ACM, 1011–1016.

[9] Zachary P. Fry, Bryan Landau, and Westley Weimer. 2012. A human study of patch maintainability. In *ISSTA*. ACM, 177–187.

[10] Congbin Guo, Mun C Tsang, and Xiaohao Ding. 2010. Gender disparities in science and engineering in Chinese universities. *Economics of Education Review* 29, 2 (2010), 225–235.

[11] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V. Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. 2018. Predicting Academic Performance: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 175–199. https://doi.org/10.1145/3293881.3295783

[12] Yu Huang, Xinyu Liu, Ryan Krueger, Tyler Santander, Xiaosu Hu, Kevin Leach, and Westley Weimer. 2019. Distilling neural representations of data structure manipulation using fMRI and fNIRS. In *International Conference on Software Engineering*. 396–407. https://doi.org/10.1109/ICSE.2019.00053

[13] June Park John and Martin Carnoy. 2019. The case of computer science education, employment, gender, and race/ethnicity in Silicon Valley, 1980–2015. *Journal of Education and Work* 32, 5 (2019), 421–435. https://doi.org/10.1080/13639080.2019.1679728

[14] Sandra Katz, John Aronis, David Allbritton, Christine Wilson, and Mary Lou Soffa. 2003. Gender and race in predicting achievement in computer science. *IEEE Technology and Society Magazine* 22, 3 (2003), 20–27.

[15] Michael S. Kirkpatrick and Chris Mayfield. 2017. Evaluating an Alternative CS1 for Students with Prior Programming Experience. In *SIGCSE*. ACM, 333–338.

[16] Wilfred WF Lau and Allan HK Yuen. 2009. Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. *British Journal of Educational Technology* 40, 4 (2009), 696–712.

[17] Mark C. Long, Patrice Iatarola, and Dylan Conger. 2009. Explaining Gaps in Readiness for College-Level Math: The Role of High School Courses. *Education Finance and Policy* 4, 1 (2009), 1–33. https://doi.org/10.1162/edfp.2009.4.1.1

[18] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 55–106. https://doi.org/10.1145/3293881.3295779

[19] Renée McCauley, Scott Grissom, Sue Fitzgerald, and Laurie Murphy. 2015. Teaching and learning recursive programming: a review of the research literature. *Computer Science Education* 25, 1 (2015), 37–66.

[20] Renée A. McCauley, Brian Hanks, Sue Fitzgerald, and Laurie Murphy. 2015. Recursion vs. Iteration: An Empirical Study of Comprehension Revisited. In *SIGCSE*. ACM, 350–355.

[21] Akira Miyake, Lauren E Kost-Smith, Noah D Finkelstein, Steven J Pollock, Geoffrey L Cohen, and Tiffany A Ito. 2010. Reducing the gender achievement gap in college science: A classroom study of values affirmation. *Science* 330, 6008 (2010), 1234–1237.

[22] Miranda Parker, Amber Solomon, Brianna Pritchett, David Illingworth, Lauren Margulieux, and Mark Guzdial. 2018. Socioeconomic Status and Computer Science Achievement: Spatial Ability as a Mediating Variable in a Novel Model of Understanding. 97–105. https://doi.org/10.1145/3230977.3230987

[23] Jack Parkinson and Quintin I. Cutts. 2018. Investigating the Relationship Between Spatial Skills and Computer Science. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER 2018, Espoo, Finland, August 13-15, 2018*. ACM, 106–114. https://doi.org/10.1145/3230977.3230990

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[25] Lyndsay Pinkus. 2008. Using early-warning data to improve graduation rates: Closing cracks in the education system. *Washington, DC: Alliance for Excellent Education* (2008).

[26] Chantel S Prat, Tara M Madhyastha, Malayka J Mottarella, and Chu-Hsuan Kuo. 2020. Relating Natural Language Aptitude to Individual Differences in Learning Programming Languages. *Scientific Reports* 10, 1 (2020), 1–10.

[27] Katie Redmond, Sarah Evans, and Mehran Sahami. 2013. A Large-Scale Quantitative Study of Women in Computer Science at Stanford University. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 439–444. https://doi.org/10.1145/2445196.2445326

[28] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[29] Lily Shashaani. 1994. Gender-differences in computer experience and its influence on computer attitudes. *Journal of Educational Computing Research* 11, 4 (1994), 347–367.

[30] Atish P. Sinha and Iris Vessey. 1992. Cognitive Fit: An Empirical Study of Recursion and Iteration. *IEEE Trans. Software Eng.* 18, 5 (1992), 368–379.

[31] Sheryl A. Sorby, Edmund Nevin, Avril Behan, Eileen Mageean, and Sarah Sheridan. 2014. Spatial skills as predictors of success in first-year engineering. In *IEEE Frontiers in Education Conference*. 1–7. https://doi.org/10.1109/FIE.2014.7044005

[32] Vladimir Sulov. 2016. Iteration vs Recursion in Introduction to Programming Classes: An Empirical Study. *Cybernetics and Information Technologies* 16 (12 2016), 63–72. https://doi.org/10.1515/cait-2016-0068

[33] Anya Tafliovich, Jennifer Campbell, and Andrew Petersen. 2013. A student perspective on prior experience in CS1. In *SIGCSE*. ACM, 239–244.

[34] Patrick T Terenzini, Leonard Springer, Patricia M Yaeger, Ernest T Pascarella, and Amaury Nora. 1996. First-generation college students: Characteristics, experiences, and cognitive development. *Research in Higher education* 37, 1 (1996), 1–22.

[35] Torbjørn Torsheim, Franco Cavallo, Kate Ann Levin, Christina Schnohr, Joanna Mazur, Birgit Niclasen, Candace Currie, FAS Development Study Group, et al. 2016. Psychometric validation of the revised family affluence scale: a latent variable approach. *Child Indicators Research* 9, 3 (2016), 771–784.

[36] Roli Varma. 2010. Why so few women enroll in computing? Gender and ethnic differences in students' perception. *Computer Science Education* 20, 4 (2010), 301–316. https://doi.org/10.1080/08993408.2010.527697

[37] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[38] Jonathan Wai, David Lubinski, and Camilla P Benbow. 2009. Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology* 101, 4 (2009), 817.

[39] Susan Wiedenbeck. 1989. Learning Iteration and Recursion from Examples. *International Journal of Man-Machine Studies* 30, 1 (1989), 1–22.

[40] Chris Wilcox and Albert Lionelle. 2018. Quantifying the benefits of prior programming experience in an introductory computer science course. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 80–85.

[41] Brenda Cantwell Wilson. 2002. A Study of Factors Promoting Success in Computer Science Including Gender Differences. *Computer Science Education* 12, 1-2 (2002), 141–164. https://doi.org/10.1076/csed.12.1.141.8211

[42] John Winslow, Ruth B Ekstrom, and Leighton A Price. 1963. *Kit of reference tests for cognitive factors.* Educational Testing Service.

[43] Eun-Mi Yang, Thomas Andre, Thomas J Greenbowe, and Lena Tibell. 2003. Spatial ability and the impact of visualization/animation on learning electrochemistry. *International Journal of Science Education* 25, 3 (2003), 329–349.