Exploring Teachers' Perspectives on Navigating Recursion Pedagogies

Jude Nzemeke
Department of Computer Science
City, University of London
jude.nzemeke@city.ac.uk

Marjahan Begum
Department of Computer Science
City, University of London
Marjahan.begum@city.ac.uk

Jo Wood
Department of Computer Science
City, University of London
j.d.wood@city.ac.uk

Abstract

Recursion is a fundamental and powerful concept in algorithm design and programming. While invaluable for solving complex problems such as tree traversal and permutation generation, recursion presents challenges for students who often struggle with comprehension, tracing recursive calls, and devising efficient solutions. This study investigates teachers' pedagogical and instructional strategies for teaching recursion, as well as effective assessment techniques. It explores the order in which programming concepts, such as iteration, selection, sequencing, recursion, and object-oriented programming (OOP) are taught in relation to how well students understand the concepts. It highlights the significance of the instructional sequence of these concepts, and reveals that, contrary to the advocated early teaching approach by some researchers – for example, teaching recursion first before iteration – recursion is mainly introduced last to students and is perceived by most of the surveyed teachers as the most challenging concept for students to learn. Teachers' perceptions of the difficulty in teaching these concepts were also explored. Programming Assignments and Coding Challenges are found to be the most popular and effective assessment methods for recursion. The study advocates for an integrated teaching approach that combines tangible objects (e.g., boxes and envelopes) and visual aids (diagrams and animations) to enhance student engagement and understanding during recursion instruction. This multi-sensory approach caters to diverse learning styles and preferences among students, offering a strategy for addressing the challenges associated with teaching recursion.

1. Introduction

The importance of studying recursion spans various domains, particularly in computer science and mathematics. Recursion serves as a powerful technique for addressing problems characterized by repetitive and self-similar structures, forming the basis for the development of intricate algorithms in computer science. By embracing recursive principles, these algorithms offer efficient solutions to challenges that might otherwise be daunting to tackle. Recursion's influence is particularly pronounced in the domain of problem decomposition. While the benefits of recursion are well acknowledged, this study investigates its teaching aspects from the viewpoint of teachers. Exploring how teachers navigate teaching recursion and implement effective assessment methods is crucial for enhancing teaching strategies, addressing challenges, and optimizing instructional sequences.

2. Highlighting Research Questions

1. Identifying the sequence in which programming concepts (iteration, selection, sequencing, recursion, and OOP) are taught, and observing possible relationships with how well students understand these concepts.
2. What are the current instructional approaches and assessment methods that teachers find successful in delivering recursion?
3. How does the frequency and consistency of incorporating movement-based activities, tangible elements, and visual aids in teaching recursion affect the perceived effectiveness of these teaching approaches?

3. Theoretical Foundations of Key Issues

Many students have difficulty understanding recursion and they often use incorrect mental models when evaluating recursive functions (Segal, 1995; Haberman and Averbuch, 2002; Sanders et al. 2006). Novice programmers also face challenges in learning recursion as they have few real-world analogies to formulate a mental model, unlike iteration (Benander et al., 1996). Most learners do not naturally think recursively (Anderson et al., 1988) and learning recursion poses difficulties due to its unconventional thinking process, especially for students lacking exposure to backward reasoning which involves working from a goal state back to an initial state. Students' previous problem-solving experiences mainly relied on forward reasoning, necessitating a paradigm shift in thinking when encountering recursion (Ginat, 2005).

3.1. Base Case

Learners often struggle with recursive functions, especially understanding the significance of the base case (McCauley et al., 2015). Misconceptions also arise in treating mathematical variables as programming variables, leading to errors, emphasized by McCauley et al. (2015) and compounded by context dependency and processing strategies (Segal, 1995). Hamouda et al. (2017) studied student misconceptions about the base case in recursion, drawing on insights from Sanders and Scholtz (2012). They linked difficulties in flow comprehension to base case misconceptions (Scholtz & Sanders, 2010). Close and Dicheva (1997) associated programming language choice with base case misconceptions, aligning with LOGO studies and Kurland and Pea's (1985) findings on language confusion. Segal (1995) dealt with categorization of "base-case as a stopping condition". Haberman and Averbuch (2002) identified challenges in identifying base cases, crucial for recursive algorithm functionality, as emphasized by them. Inadequate base cases may lead to non-terminating processes and computational inefficiencies, especially with substantial input data.

3.2. Recursion vs Iteration

In a study comparing comprehension of recursion and iteration, Benander et al. (1996) found a statistically significant advantage for recursion. Benander, et al. (2000) found that in small code segments involving linked lists, programmers might find locating bugs in recursive code, particularly in copying tasks, to be easier. Mirolo (2012) contradicted the notion that novice students find iteration easier than recursion, attributing difficulty to task characteristics rather than programming paradigm. Endres et al. (2021) observed superior performance in iterative-framed problems involving non-branching numerical computation. McCracken (1987) cautioned against deeming recursive programming "hopelessly difficult", emphasizing the importance of task matching. Sinha and Vessey (1992) linked construct choice to cognitive fit, advocating task and problem representation considerations.

The debate over whether to teach recursion or iteration first in computer science education involves conflicting perspectives on foundational concepts and ease of understanding. Guzdial in a conversation at the ITISCE 2023 conference and in Guzdial (2018) while referring to studies by Kessler and Anderson (1986) and Wiedenbeck (1989), suggested teaching iteration first due to its easier grasp and broader practical application. Turbak et al. (1999) found introducing recursion before iteration more effective, contrary to traditional methods, challenging the ongoing discourse on optimal sequencing in computer science education. Maiorana et al. (2021) concluded that students can grasp both recursion and iteration simultaneously, supporting the early introduction of recursion to enhance algorithm understanding in the curriculum.

3.3. Pedagogical Approach

To enhance students' understanding of recursion across computer science domains, Velázquez-Iturbide (2000) proposes a progressive teaching method introducing recursion through formal grammars, functional programming, and imperative programming. Syslo and Kwiatkowska (2014) recommend presenting recursion as a "real-life topic" to make it more accessible and relatable, especially for beginners. Explaining recursion to novice programmers can be challenging, and approaches like inductive definitions, Runtime Stack Simulation, Process Tracing, Mathematical induction, Russian Dolls, and the recursion tree (Dann et al., 2001; Haynes, 1995) help address this complexity. Wu et al. (1998) emphasize the importance of conceptual models for teaching recursion to novice programmers, cautioning about adapting or designing concrete models without conveying internal mechanism details. Gunion et al. (2009) challenge concerns about 'middle school' students learning recursion, demonstrating that hands-on activities effectively increase engagement and facilitate learning. Enhancing the learning experience for students can be significantly facilitated by employing tangible materials instead of abstract concepts (Akbaşlı and Yeşilce, 2018). The use of animations, such as in tools like Alice, during recursion introduction has shown promise in enhancing student comprehension, although further research is needed to establish its long-term impact (Dann et al., 2001).

3.4. Learning Styles or Not

Understanding individual learning styles, especially in programming concepts like recursion (Wu et al., 1998), is vital for effective teaching. Dunn and Dunn advocate tailoring teaching methods to enhance students' attainment, behaviour, and attitudes based on their research (Dunn, 1984; Dunn et al., 2009). However, teaching in a style different from students may increase cognitive load, hindering learning (Sweller, 1988). Recognizing diverse learning styles, such as visual learning, can reduce cognitive load, improving information assimilation and retention (Jawed et al., 2019). Aligning teaching strategies with varied learning styles is crucial in programming education (Bargar and Hoover, 1984).

Kavale and Forness (1990) defended their meta-analysis against Dunn's (1989) critique, asserting the ineffectiveness of modality testing and teaching. Pashler et al. (2008) questioned the experimental basis and commercial motives of learning styles, echoed by Reynolds (1997) and Willingham (2005), who cited a lack of scientific evidence. Tarver and Dawson (1978), and Dembo and Howard (2007) opposed modality preference theory, citing empirical limitations and potential harm. Arbuthnott and Krätzig (2015) highlighted the inefficacy of tailoring teaching to sensory learning styles. Teachers are advised to focus on content-driven modality choices and universal methods (Kavale and Forness, 1990; Tarver and Dawson, 1978; Willingham, 2005). Various methods to measure modality preferences exist, but caution is needed due to limitations (Willingham, 2005). Instead of catering to individual differences, teachers should employ diverse modalities for variety, attention, and memory strategies, benefiting all students (Tarver and Dawson, 1978; Kavale and Forness, 1987; Willingham, 2005).

In line with Coffield et al. (2004) and other researchers who argue that learning styles are not fixed traits, but rather flexible preferences influenced by context and tasks, our own teaching experiences support this perspective. Through working with diverse groups of learners, we have observed how individuals' preferences for learning can vary depending on the subject matter and the learning environment. Embracing this viewpoint, we too believe that teachers should prioritize flexibility in their teaching methods, employing a range of strategies that accommodate the dynamic nature of learning preferences.

With this principle in mind, our focus will be on exploring the teaching methods utilized when teaching recursion, particularly looking at teachers' perceptions of the impact of these methods, including the use of visualization, auditory, reading/writing, and kinaesthetic techniques in teaching. This approach offers a pragmatic means of addressing the research questions and shifts the focus toward identifying effective practices that can benefit a wider array of students in the teaching and learning of recursion, rather than attempting to tailor instruction to each individual student's unique learning style.

4. Methodology

Creswell (2009) highlighted the importance of philosophical worldview – "a basic set of beliefs that guide action" in research design framework. These beliefs can be used by researchers to decide if they should make use of qualitative, quantitative, or mixed methods approach. The design framework can be illustrated further as seen in figure 1 below:

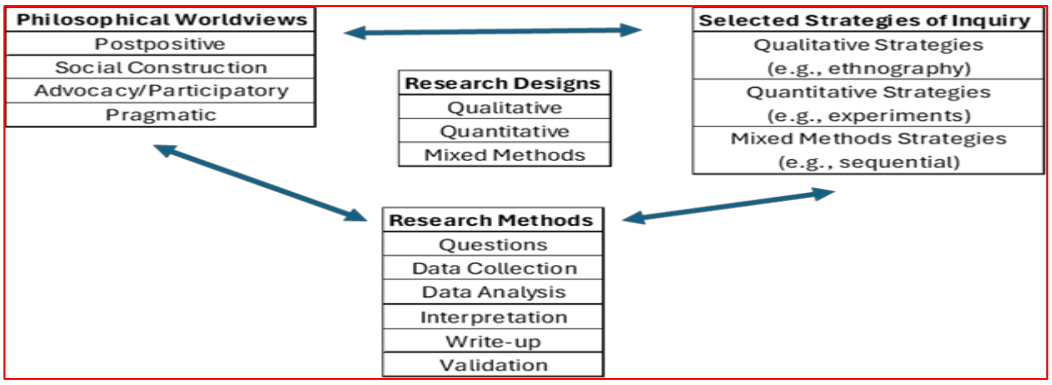

Figure 1: Design Framework [Creswell, 2009].

We apply a mixed-methods approach to investigating current teaching practice. Born from the paradigm wars, it combines qualitative and quantitative approaches, offering a comprehensive view of complex topics (Johnson and Onwuegbuzie, 2004; Terrell, 2012; Poth and Munce, 2020). Utilizing both methods enhances understanding and explores multifaceted problems from various perspectives (Poth & Munce, 2020). Rooted in pragmatism, mixed methods emphasizes practical outcomes and provides diverse design choices for researchers (Shorten & Smith, 2017; Terrell, 2012). This approach, applicable across disciplines, proves valuable in answering intricate research questions (Terrell, 2012). The point of integration is one of the primary design dimensions for mixed method research. It is defined as "any point in a study where two or more research components are mixed or connected in some way" (Schoonenboom and Johnson, 2017). Getting the process of data integration from qualitative and quantitative components of the study right is key to have more insight of the data collected in mixed methods, and this can take place during the analysis phase of the study.

Quantitative data were gathered through a comprehensive online survey featuring 24 questions, with 21 focused on quantitative information. The survey covered non-sensitive demographic data, programming language used by teachers, challenges in teaching programming concepts and pedagogical approaches. It inquired into instructional methods, assessment approaches, and effective techniques in teaching recursion, providing a thorough overview of quantitative aspects in programming education.

Complementing the quantitative findings, qualitative insights were obtained through open-ended questions, enabling in-depth participant responses unconstrained by predetermined choices (Hyman and Sierra, 2016). To optimize completion rates, the survey incorporated a three-box limit for open-text responses, guided by Qualtrics online experts, acknowledging that exceeding this limit could reduce completion rates due to increased cognitive effort required for responses.

5. Ethical Considerations

In adhering to ethical standards outlined by the British Educational Research Association (BERA, 2018) and City, University of London, this educational research prioritized informed consent. Ethical approval from City, University of London's ethics committee was secured for this research.

6. Sampling and Recruitment

Examination boards in England, tasked with developing detailed subject specifications that outline the curriculum framework – including what students are expected to learn, understand, and achieve by the end of the course – focus on recursion exclusively in post-secondary education (for students aged 16 to 18). Therefore, it was expected that primarily, teachers who teach recursion at this level and above, would take part in the study. However, due to the relatively low enrolment of computer science students in England, in post-secondary school (OFQUAL, 2023), the pool of teachers specializing in teaching recursion is anticipated to be limited. From our experience and from interaction with teachers, we know that this challenge arises because some teachers may opt not to cover recursion at primary and secondary school (attended by students less than 16 years old), potentially due to time constraints in delivering the curriculum. To address the challenge of recruiting teachers for the study, who teach recursion, the Digital SchoolHouse Ingenuity Day 1 conference event was strategically targeted, a gathering primarily attended mainly by computer science teachers.

Initially, 36 computer science teachers began the survey, comprising 61% male, 30% female, with 5% opting not to disclose their gender, and 2% identifying as non-binary. Regarding ethnicity, 64% identified as White British, 17% as other white backgrounds, and 6% each for Black and Asian backgrounds, with an additional 5% identifying as other ethnic backgrounds, while 2% chose not to disclose. One of the questions in the survey was designed to screen participants for eligibility – targeting only teachers who have taught or currently teach recursion. The question simply asked, "Have you taught or are you currently teaching recursion as part of your curriculum?" The survey ended for teachers who responded "No" to this question, indicating that they lacked the experience of teaching this topic. The eligibility screening, verifying experience in teaching recursion, narrowed the final sample to 14 teachers. Among them, seven teachers had 15 or more years of teaching experience, three teachers had 11–15 years, and 6 had 6 –10 years, with only one having less than one year teaching experience. All teach recursion to post-secondary students (16 to 18-year-olds), with 3 also teaching at

Foundation and Undergraduate levels (18+ years old). These criteria ensured a focused and valid study with contributions from mostly experienced teachers across institutions, effectively addressing research objectives while acknowledging generalizability limitations.

7. Data Analysis and Discussion

7.1. RQ1: Identifying the sequence in which programming concepts (iteration, selection, sequencing, recursion, and OOP) are taught, and observing possible relationships with how well students understand these concepts.

7.1.1. Order of Teaching Concepts:

In investigating the teaching sequence of programming concepts, we explored the order in which these concepts are typically introduced. The instructional sequence can significantly influence students' comprehension and retention, providing insights into teachers' approaches. Of particular interest is the positioning of recursion relative to other concepts, indicating its foundational or advanced nature. Teachers were asked to rank the order in which they taught the different concepts. Analyzing mean rankings, with lower numbers indicating earlier introduction, reveals a consistent progression: sequence (1.64), selection (1.93), iteration (2.50), OOP (4.43), and recursion (4.50). This order aligns with a pedagogical strategy that introduces simpler concepts as building blocks before tackling more complex and abstract ideas. It appears to be a strategy that supports argument made by Kessler and Anderson (1986) and the subsequent study conducted by Susan Wiedenbeck (1989) that teaching iteration before recursion is more beneficial, as iteration is easier to grasp and has wider practical application.

7.1.2. How Challenging are these Concepts to Students:

Teachers' insights into students' struggles with specific concepts further inform the analysis. Teachers were asked to rank the order in which students understood the different concepts from easy to understand, to very hard to understand. Recursion topped the list as the most challenging concept for students (with a mean value of 3.93), followed closely by OOP (with a mean value of 3.79). Sequencing was perceived as the least challenging (with a mean value of 2.29). Examining standard deviation and variance highlighted the variability in ratings, with recursion exhibiting the highest values (0.74 and 0.55) and iteration the lowest (0.81 and 0.66), indicating the range of opinions among respondents.

7.1.3. How Difficult are these concepts to teach:

Investigating the difficulty levels teachers encounter when teaching the programming concepts, our findings highlight a significant variation in perceived challenges. For OOP, teachers reported a mean difficulty of 3.71, with a median difficulty of 4, indicating it is one of the most challenging topics to teach. Recursion followed closely with a mean difficulty of 3.64 and a similar median (of 4). Both concepts showed a wide range of challenge levels, from somewhat challenging to very challenging. Sequencing, iteration, and selection were considered less challenging, with mean difficulties of 2.07, 2.21, and 2.00, respectively, and medians at 2. These topics were generally seen as easier to teach, with their difficulty ranging from non-challenging to moderately challenging.

7.1.4. Findings for RQ1

Due to the impact on the statistical power of a smaller sample size, it was deemed that data collected might not have enough power to detect a significant correlation using nonparametric measures for example Spearman Rank Correlation. Bujang and Baharum (2016) proposed a minimum of 29 samples (or subjects) to detect a reasonably high correlation (specifically, a correlation coefficient of 0.5) with a good balance of error tolerance and study power. They noted other studies "(Bujang et al., 2009; Bujang et al., 2015)" that suggest samples larger than 300 can yield statistical results highly representative of the true population values. This is based on the idea that larger samples tend to provide more precise estimates of population parameters, thereby improving the generalizability of the findings. This made it apparent that the best way to analyse the data will be to look at it from a practical or observational perspective.

The violin plots below (Figure 2: Teacher Order and Students Understanding Order) are used to illustrate both the spread and the median of the orders in which the programming concepts are taught and students understanding, with one side of the violin for teaching orders and the other for indicating order in which student understanding concepts. The following observations were made:

The concept of sequence typically appears early in learning, supported by students finding it easier to understand, which aligns with it likely being an introductory concept. Selection is also introduced early on, yet students find its difficulty level consistent, irrespective of its teaching order. Iteration tends to be taught mid-way through the curriculum and is well understood by students, indicating its placement is appropriate for their learning curve. OOP is often reserved for the latter part of educational programs, which is reflected in its broader and more challenging understanding distribution, highlighting its complexity. Recursion stands out with a distinct pattern where both teaching and understanding orders are skewed to the higher end, indicating it is both taught late and considered difficult to understand.

Overall, the data reflects a structure in teaching programming that rises from simpler to more complex concepts, aligning well with student comprehension levels.

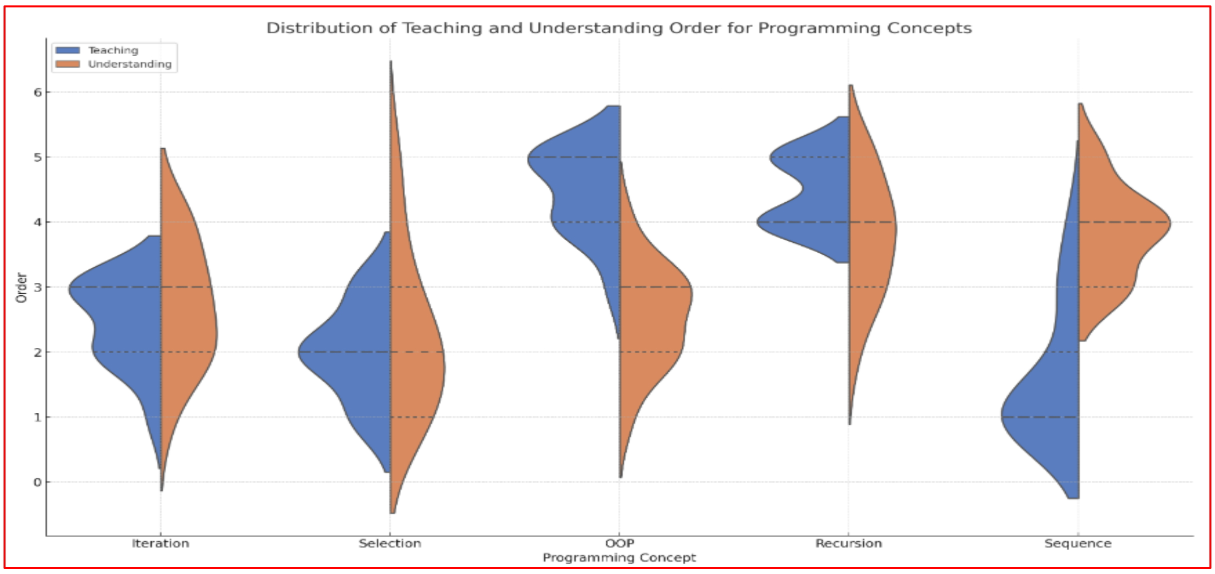

Figure 2: Teacher Order and Students Understanding Order

Note for Figure 2. The shape and width of the violins provide an immediate visual indication of the distribution's spread and density. A wider section of a violin plot indicates a higher frequency of data points (i.e., more teachers reported similar orders), whereas a narrower section indicates fewer data points. The horizontal lines within each violin represent the median order. This is crucial for exploring the most common teaching order, and the understanding order for each concept. The degree of symmetry between the teaching and understanding sides of each violin gives an immediate visual cue about alignment. High symmetry suggests that the understanding order closely matches the teaching order, whereas asymmetry suggests discrepancies.

7.2. RQ2 What are the current instructional approaches and assessment methods that teachers find successful in delivering recursion?

Teachers' approaches to teaching recursion offer insights into adapting methods for diverse learners. The choice of instructional approach significantly influences students' understanding and engagement with recursion concepts in programming education. The approaches used in the survey are defined as follows:

Inquiry-based learning is an approach where students learn through questioning and investigation. When teaching recursion, this approach might involve encouraging students to explore recursion concepts by asking questions, conducting research, and experimenting. For instance, students might investigate different recursive algorithms and their applications.

Direct instruction involves presenting information to students in a structured and systematic way. When teaching recursion, this approach may include clear explanations of recursion concepts, step-by-step examples, and guided practice. For instance, the teacher might systematically introduce recursive functions and then provide exercises to reinforce the learning.

Project-based learning is an approach where students learn by working on projects. When teaching recursion, this approach may involve assigning projects that require students to apply recursive concepts practically. For example, students could be asked to create a recursive artwork generator or a recursive maze-solving program.

Problem-based learning is an approach where students learn by solving problems. In the context of recursion, this approach might involve presenting students with real-world problems that can be solved using recursive techniques.

Collaborative learning is an instructional approach where students work together. When teaching recursion, this approach might involve students collaborating on recursive coding projects or solving recursion-related problems as a team. For example, students may work together to create a recursive function in a programming language.

Differentiated instruction [or Adapting Teaching] acknowledges the diverse needs of students. If a teacher selects this approach when teaching recursion, it means they are adapting their instruction to cater to individual students' learning styles and abilities. For instance, a teacher might provide additional resources or assignments to support struggling students while challenging advanced learners with more complex recursion problems.

Blended learning combines online and in-person instruction. In the context of teaching recursion, this approach could involve using online resources and platforms to complement in-person lessons. For example, students might watch online tutorials on recursion algorithms and then apply what they've learned during in-person coding sessions.

Independent learning is an approach where students research topics on their own. In the context of recursion, this approach may involve assigning self-directed projects or providing resources for students to explore recursion independently. For example, students could be given a list of recursion-related books and websites to explore as part of their learning process.

Common approaches include problem-based and inquiry-based learning, with problem-based and project-based learning considered the most effective. Collaborative learning, although less prevalent, still proves effective. However, further exploration is needed regarding differentiated instruction and blended learning. See Figure 3 for graph on instructional approaches and their effectiveness.

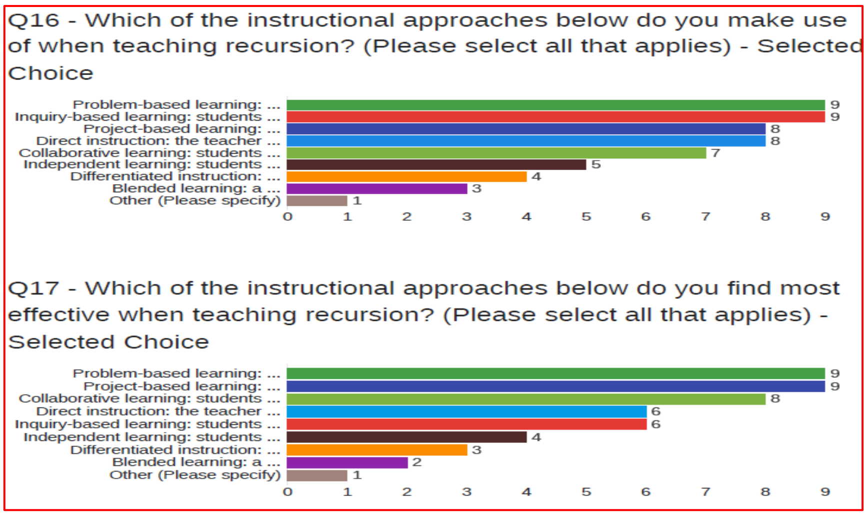

Figure 3: Instructional Approaches for Teaching Recursion

Evaluating assessment methods used in teaching recursion is crucial for assessing their effectiveness in measuring student comprehension. The options include Written Exams/Quizzes, Programming Assignments/Coding Challenges, Verbal Question and Answer sessions in lessons, Pair Review/Pair Programming, and Pupil Demonstration/Presentation. The diversity of assessment methods recognizes varied student learning styles and abilities, influencing how teachers adapt teaching strategies. This information helps identify assessment methods that promote a deeper understanding of recursion concepts.

Assessment methods vary, with programming assignments being the most popular and effective, followed by verbal Q&A and written exams. Pair review/pair programming and pupil demonstration/presentation are less common. Our results indicate that most teachers find programming assignments the most effective assessment method for recursion, followed by written exams/quizzes, verbal Q&A, and pupil demonstration/presentation. Pair review/pair programming is perceived as the least effective method. No respondents indicated the use of alternative assessment methods. See Figure 4 and Figure 5 for graph on assessing students.

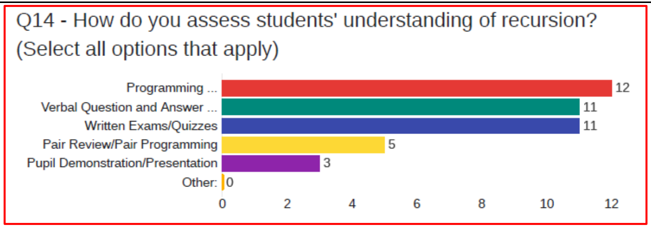

Figure 4: Assessing students' understanding of Recursion.

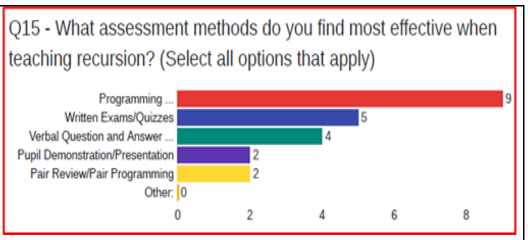

Figure 5: Effectiveness of Assessment Methods

What are teachers' perceptions of the challenges faced by pupils when learning recursion?

Further inquiry was made to investigate the challenges students commonly encounter while learning recursion, with the goal of informing targeted teaching strategies. This inquiry is crucial for understanding specific pain points in student learning and facilitating the creation of effective teaching materials. Different challenges may emerge at varying educational levels or with specific programming languages, emphasizing the need for customized instruction. Responses from teachers highlighted prevalent challenges, including students struggling to comprehend the purpose of recursion and lacking foundational knowledge of iteration – which is sometimes mistaken for recursion and vice versa. To address this, teachers should ensure students have a solid grasp of fundamental concepts before introducing recursion, indicating the importance of a well-structured curriculum.

Another noteworthy challenge is students relying solely on data tracing to understand recursion, calling for encouragement to look into the underlying principles for a deeper conceptual understanding. Confusion between recursion and other programming concepts requires clear differentiation and practical examples to alleviate misunderstandings. Understanding the context and rationale behind recursive code is identified as a challenge, suggesting the importance of real-world examples and practical applications to enhance comprehension.

Teachers also expressed a lack of in-depth resources on how to teach recursion as a challenge, emphasizing the need for comprehensive materials catering to diverse learning styles and experience levels. This highlights the importance of resource development to support teachers in delivering effective instruction on recursion.

The survey sought advice from teachers on teaching recursion effectively to those new to the subject. Recommendations included thorough preparation, confidence, and simplicity in tasks to avoid overload. Peer support, understanding individual student needs, and fostering a student-centric approach were highlighted.

Practical aspects, such as extensive practice, providing examples, and using teaching tools like real-world examples and visual aids, were emphasized for diverse learning styles. The belief in spending more time on the topic highlighted the importance of patience and a comprehensive exploration of recursion for better understanding. Ensuring a strong foundation by understanding basics before tackling complex topics was advised. Teachers suggested addressing potential difficulties students may face with recursion by adapting teaching methods and maintaining focus and conciseness in delivery.

7.3. How does the frequency and consistency of incorporating movement-based activities, tangible elements, and visual aids in teaching recursion affect the perceived effectiveness of these teaching approaches?

The research explored teaching approaches for recursion, focusing on methods and the effectiveness of the use of hands-on activities, tangible materials, and visual aids.

Six out of 14 teachers occasionally use hands-on or movement-based activities for recursion, with 3 using them frequently. The perceived effectiveness varies, with 7 teachers rating them moderately effective, 4 very effective, and 3 slightly effective. Interestingly, no extreme opinions were expressed, indicating varied perceptions among teachers. While not universally adopted, hands-on activities are generally perceived as beneficial by those who incorporate them; and can enhance student engagement and understanding of abstract concepts like recursion.

Most teachers (9 out of 14) seldom or occasionally incorporate tangible elements in teaching recursion. The effectiveness varies, with 5 finding it very or extremely effective and 4 considering it slightly important. Further investigation is suggested to understand why some find this approach effective despite infrequent use. Understanding the specific tangible elements and materials used could offer valuable insights into effective teaching strategies for recursion.

The study emphasizes the use of visual aids, such as diagrams and animations, in teaching recursion. A significant majority (10 out of 14) frequently or always use visual aids, finding them highly effective in conveying recursion concepts. Only two teachers found them slightly effective. Visual aids have gained widespread acceptance, indicating their effectiveness in teaching recursion.

Two teachers who mainly teach post-secondary students in different schools, strongly advocated for the use of PRIMM (Predict, Run, Investigate, Modify and Make) and differentiation approaches when teaching recursion and other programming concepts in general. In a follow-up conversation with one of the teachers after the survey, they claimed to have observed marked improvements in students' outcomes, particularly in learning about recursive algorithms, since implementing PRIMM approach at their school "for over three years now". In their study, Sentance et al. (2019) suggested that PRIMM offers an efficient method for teaching programming, enhancing comprehension, and boosting confidence in students. They emphasized that teachers found PRIMM's structured lessons beneficial, offering clarity in both lesson planning and delivery and it allowed teachers to tailor tasks to individual student needs. Additionally, they recommended PRIMM's suitability for teacher training and various stages of programming education.

8. Further Discussion

The study emphasizes recursion as the most challenging programming concept for students, from the teacher's perspective, shedding light on practical difficulties in the classroom. This recognition prompts teachers to allocate additional time and resources, enhancing instructional strategy effectiveness. Contrary to literature advocating for early or pre-iteration teaching of recursion, as highlighted in the Theoretical Foundation section above, our study reveals that among the programming concepts examined, recursion is, in fact, introduced to students last. The findings regarding assessment methods and pedagogical approaches offer practical insights for teachers and researchers alike. Programming assignments and coding challenges emerge as the preferred and most effective assessment tools for recursion, providing a clear direction for teachers when designing students' evaluations, in their lesson planning. Additionally, the endorsement of problem-based learning and inquiry-based learning, alongside the nuanced impact of collaborative learning, offers valuable guidance for teachers seeking effective instructional strategies in the teaching and learning of recursion. While employing tangible objects for example boxes and envelopes to symbolize values returned by functions in recursive calls has proven highly effective in our teaching experience, our research emphasizes that, from the teachers' perspective, visual aids (diagrams and animations), are widely embraced and very effective. We contend that the use of tangible objects, despite being a hands-on approach, also offers a form of visualization for learners. This highlights their crucial role in bolstering student engagement and understanding during recursion instruction. The study advocates for an integrated teaching approach that combines for example, tangible objects and visual aids, fostering a multi-sensory learning experience that caters to diverse learning styles and preferences among students. This integrated approach holds the potential to significantly enhance the effectiveness of recursion instruction and contribute to improved learning outcomes.

9. Conclusion

In conclusion, the relationship between the order of teaching programming concepts and students' understanding suggests a general alignment with educational theory: simpler concepts are introduced first, leading to a smoother learning curve for students. However, certain concepts like OOP and recursion present challenges that are recognized by both the teaching order and students' understanding. This could point to areas where additional teaching aids, practice, or alternative instructional strategies might be beneficial. The study brings to light the complexities inherent in teaching and learning recursion. Ongoing monitoring of students' understanding relative to the teaching order is crucial. Adjustments to this teaching sequence, if necessary, should be data-driven and responsive to the observed learning outcomes. Furthermore, problem-based learning emerged as the most effective method for teaching recursion, with programming assignments being the most popular and effective assessment approach. Practical insights into effective assessment methods and teaching approaches empower teachers to refine their techniques. We advocate for an integrated teaching approach that incorporates tangible objects and visual aids, offering a strategy that may enhance student engagement and comprehension. Together, these findings provide teachers with a framework to address the challenges associated with teaching recursion, fostering an environment conducive to improved learning experiences and outcomes.

10. Limitations

While we are confident in the credibility and meaningfulness of the data collected from the teachers surveyed, it is essential to acknowledge several limitations inherent in our study. Firstly, the sample size presents a challenge. While a larger sample size would enhance the statistical power and generalizability of the study, the inclusion criteria ensured that all participants had relevant experience, which is crucial for the study's focus on teaching recursion. Additionally, the distribution of teaching experience among the final sample, with a majority having 15 or more years of experience, adds credibility to the insights gathered. Furthermore, while our study provides valuable insights into current teaching approaches used in the delivery of recursion, it is not without its constraints. We cannot definitively establish causality between teaching sequence of concepts and students' learning outcomes, as correlation does not imply causation. As such, our study does not offer comprehensive explanations for the observed variables. Despite these limitations, we believe that our findings contribute valuable insights into the field of Computer Science education.

11. Future Work

Drawing from the findings of this research, future investigations could explore the impact of instructional sequence on the perceived and actual difficulty of programming concepts. Research could scrutinize how altering the sequence of these concepts influences teacher perceptions, student understanding, and overall learning outcomes especially regarding recursion. Future research could also aim to address these limitations by employing larger sample sizes, utilizing experimental designs to establish causality, and explore the underlying mechanisms driving teaching practices. Furthermore, specific studies should be conducted, focusing on the role and effectiveness of visual aids, alongside the integration of tangible elements and physical materials in teaching recursion. These focused inquiries promise to provide practical insights for teachers, exploring the potential benefits of these approaches in enhancing both student comprehension and engagement.

12. References

1. Akbaşlı, Sait & Yeşilce, İlknur. (2018). Use of Tangible Materials and Computer in Mathematics Teaching: Opinions of School Principals. Eurasia Journal of Mathematics, Science and Technology Education. 14. 2523-2532. 10.29333/ejmste/90087.
2. Anderson, J. R., Pirolli, P., & Farrel, R. (1988). Learning to program recursive functions. In M. T. Chi, R. Glaser, & M. J. Farr (Eds.), The nature of expertise (pp. 153–183). Hillsdale: Erlbaum.
3. Arbuthnott, K. D., & Krätzig, G. P. (2015). Effective Teaching: Sensory Learning Styles versus General Memory Processes. Comprehensive Psychology, 4. https://doi.org/10.2466/06.IT.4.2.

4. Bargar, R. R., & Hoover, R. L. (1984). Psychological Type and the Matching of Cognitive Styles. Theory Into Practice, 23(1), 56–63. http://www.jstor.org/stable/1476739.
5. Benander, A. C., Benander, B. A., & Pu, H. (1996). Recursion vs. iteration: An empirical study of comprehension. Journal of Systems and Software, 32, 73–82.
6. Benander, A.C., Benander, B.A., & Sang, J. (2000). An empirical analysis of debugging performance - differences between iterative and recursive constructs. J. Syst. Softw., 54, 17-28.
7. BERA. (2018). Ethical guidelines for educational research (4th ed.). https://www.bera.ac.uk/publication/ethical-guidelines-for-educational-research-2018-online.
8. Bujang, Mohamad Adam & Baharum, Nurakmal. (2016). Sample Size Guideline for Correlation Analysis. World Journal of Social Science Research. 3. 37. 10.22158/wjssr.v3n1p37.
9. Claudio Mirolo. 2012. Is iteration really easier to learn than recursion for CS1 students? In Proceedings of the ninth annual international conference on International computing education research (ICER '12). Association for Computing Machinery, New York, NY, USA, 99–104. https://doi.org/10.1145/2361276.2361296j
10. Claudius M. Kessler & John R. Anderson (1986) Learning Flow of Control: Recursive and Iterative Procedures, Human–Computer Interaction, 2:2, 135-166, DOI: 10.1207/s15327051hci0202_2
11. Coffield, F. & Moseley, D. & Hall, Elaine & Ecclestone, K. (2004). Learning Styles and Pedagogy In Post-16 Learning: A Systematic And Critical Review. Book Learning styles and pedagogy in post-16 learning: a systematic and critical review.
12. Creswell, J. W. (2009). Research Design: Qualitative, Quantitative, and Mixed Methods Approaches (3rd ed.). Sage Publications, Inc.
13. Dembo, M. H., & Howard, K. (2007). Advice about the use of learning styles: A major myth in education. Journal of College Reading and Learning, 37(2), 101-109. https://doi.org/10.1080/10790195.2007.10850174.
14. Dicheva, Darina & Close, Sean. (1997). Misconceptions in recursion: diagnostic teaching.
15. Dunn, R. (1984). Learning Style: State of the Science. Theory Into Practice, 23(1), 10–19. http://www.jstor.org/stable/1476733
16. Dunn, R. (1990). Bias over Substance: A Critical Analysis of Kavale and Forness' Report on Modality-Based Instruction. Exceptional Children, 56(4), 352-356. https://doi.org/10.1177/001440299005600409
17. Dunn, R., Honigsfeld, A., Doolan, L. S., Bostrom, L., Russo, K., Schiering, M. S., Suh, B., & Tenedero, H. (2009). Impact of Learning-Style Instructional Strategies on Students' Achievement and Attitudes: Perceptions of Educators in Diverse Institutions. The Clearing House, 82(3), 135–140. http://www.jstor.org/stable/30181095
18. Endres, M., Weimer, W., & Kamil, A. (2021). An Analysis of Iterative and Recursive Problem Performance. Proceedings of the 52nd ACM Technical Symposium on Computer Science Education.
19. F. Turbak, C. Royden, J. Stephan, and J. Herbst, Teaching Recursion Before Loops In CS1, Journal of Computing in Small Colleges, Volume 14, Number 4, pp 86-101, May 1999.
20. Ginat, D. (2005). The suitable way is backwards, but they work forward. Journal of Computers in Mathematics and Science Teaching, 24, 73–88. Norfolk, VA: AACE.
21. Gunion, Katherine & Milford, Todd & Stege, Ulrike. (2009). The Paradigm Recursion: Is It More Accessible When Introduced in Middle School?. The Journal of Problem Solving. 2. 10.7771/1932-6246.1063.
22. Guzdial, M. (2018) Exploring the question of teaching recursion or iterative control structures first, Computing Ed Research - Guzdial's Take. Available at: https://computinged.wordpress.com/2018/03/09/exploring-the-question-of-teaching-recursion-or-iterative-control-structures-first/ (Accessed: 22 October 2023).
23. Haberman, B., & Averbuch, H. (2002, June 24–28). The case of base cases: Why are they so difficult to recognize? Student difficulties with recursion. In Proceedings of the 7th conference on innovation and technology in computer science education. Aarhus.
24. Hamouda, S., Edwards, S., Elmongui, H., Ernst, J., & Shaffer, C. (2017). A basic recursion concept inventory. Computer Science Education, 27(2), 121–148.
25. Hyman, Michael & Sierra, Jeremy. (2016). Open- versus close-ended survey questions. NMSU Business Outlook. 14.

26. Ian Sanders, Vashti Galpin, and Tina Götschi. 2006. Mental models of recursion revisited. SIGCSE Bull. 38, 3 (September 2006), 138–142. https://doi.org/10.1145/1140123.1140162
27. J Terrell, Steven. (2012). Mixed-Methods Research Methodologies. Qualitative Report. 17. 254-265. 10.46743/2160-3715/2012.1819.
28. Jawed S, Amin HU, Malik AS and Faye I. (2019). Classification of Visual and Non-visual Learners Using Electroencephalographic Alpha and Gamma Activities. Front. Behav. Neurosci. 13:86.
29. Johnson, R. B. & Onwuegbuzie, A. J. (2004). Mixed-methods research: a research paradigm whose time has come. Educational Researcher, 33(7), 14-26.
30. Kavale, K. A. and Forness, S. R. (1987). Substance over style: Assessing the efficacy of modality testing and teaching. Exceptional Children, 54(3), 228–239.
31. Kavale, K. A., & Forness, S. R. (1990). Substance over Style: A Rejoinder to Dunn's Animadversions. Exceptional Children, 56(4), 357-361. https://doi.org/10.1177/001440299005600410
32. Kurland, D. M., & Pea, R. D. (1985). Children's Mental Models of Recursive Logo Programs. Journal of Educational Computing Research, 1(2), 235-243. https://doi.org/10.2190/JV9Y-5PD0-MX22-9J4Y.
33. Maiorana, F., Csizmadia, A., Richards, G., Riedesel, C. (2021). Recursion Versus Iteration: A Comparative Approach for Algorithm Comprehension. In: Auer, M.E., Centea, D. (eds) Visions and Concepts for Education 4.0. ICBL 2020. Advances in Intelligent Systems and Computing, vol 1314. Springer, Cham. https://doi.org/10.1007/978-3-030-67209-6_27
34. McCracken, D. D. (1987, January). Ruminations on computer science curricula, viewpoint column. Communications of the ACM, 30, 3–5.
35. OFQUAL (2023). Official Statistics, Provisional Entries for GCSE, AS and A Level: Summer 2023 Exam Series. Gov.UK. Retrieved December 24, 2023, from www.gov.uk/government/statistics/provisional-entries-for-gcse-as-and-a-level-summer-2023-exam-series/provisional-entries-for-gcse-as-and-a-level-summer-2023-exam-series
36. Pashler, H., McDaniel, M., Rohrer, D. and Bjork, R., 2008. Learning styles: Concepts and evidence. Psychological science in the public interest, 9(3), pp.105-119.
37. Poth, C., & Munce, S. E. P. (2020). Commentary—Preparing today's researchers for a yet unknown tomorrow: Promising practices for a synergistic and sustainable mentoring approach to mixed methods research learning. International Journal of Multiple Research Approaches, 12(1), 56-64. doi:10.29034/ijmra.v12n1commentary
38. Renée McCauley, Scott Grissom, Sue Fitzgerald & Laurie Murphy (2015) Teaching and learning recursive programming: a review of the research literature, Computer Science Education, 25:1, 37-66, DOI: 10.1080/08993408.2015.1033205.
39. Reynolds, M. (1997). Learning Styles: A Critique. Management Learning, 28(2), 115-133. https://doi.org/10.1177/1350507697282002.
40. S. M. Haynes. 1995. Explaining recursion to the unsophisticated. SIGCSE Bull. 27, 3 (Sept. 1995), 3–6. https://doi.org/10.1145/209849.209850.
41. Sanders, I., & Scholtz, T. (2012). First year students' understanding of the flow of control in recursive algorithms. African Journal of Research in Mathematics, Science and Technology Education, 16(3), 348–362
42. Scholtz, T. L., & Sanders, I. (2010). Mental models of recursion: Investigating students 'understanding of recursion. In Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2010)(pp. 103–107). Bilkent, Ankara, Turkey.
43. Schoonenboom J, Johnson R. B. (2017). How to Construct a Mixed Methods Research Design. Kolner Z Soz Sozpsychol. 2017;69(Suppl 2):107-131. doi: 10.1007/s11577-017-0454-1. Epub 2017 Jul 5. PMID: 28989188; PMCID: PMC5602001.
44. Segal, J. (1995). Empirical studies of functional programming learners evaluating recursive functions. Instructional Science, 22, 385–411. https://doi.org/10.1007/BF00891962
45. Shorten A., & Smith J. (2017). Mixed methods research: Expanding the evidence base. Evid Based Nurs, 20, 74–5. http://dx.doi.org/10.1136/eb-2017-102699
46. Sinha, A., and Vessey, I., (1992) Cognitive Fit: An Empirical Study of Recursion and Iteration, IEEE Trans. Software Eng. 18, 368-379.

47. Sue Sentance, Jane Waite, and Maria Kallia. 2019. Teachers' Experiences of using PRIMM to Teach Programming in School. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19). Association for Computing Machinery, New York, NY, USA, 476–482. https://doi.org/10.1145/3287324.3287477
48. Susan Wiedenbeck, Learning iteration and recursion from examples, International Journal of Man-Machine Studies, Volume 30, Issue 1, 1989, Pages 1-22, ISSN 0020-7373, https://doi.org/10.1016/S0020-7373(89)80018-5.
49. Sweller, J. (1988), Cognitive Load During Problem Solving: Effects on Learning. Cognitive Science, 12: 257-285. https://doi.org/10.1207/s15516709cog1202_4.
50. Syslo, Maciej & Kwiatkowska, Anna. (2014). Introducing Students to Recursion: A Multi-facet and Multi-tool Approach. 124-137. 10.1007/978-3-319-09958-3_12.
51. Tarver, S. G., & Dawson, M. M. (1978). Modality preference and the teaching of reading: A review. Journal of Learning Disabilities, 11(1), 17-29. https://doi.org/10.1177/002221947801100104
52. Velázquez-Iturbide, J. Ángel. (2000). Recursion in gradual steps (is recursion really that difficult?). ACM SIGCSE Bulletin. 32. 310-314. 10.1145/331795.331876.
53. Wanda Dann, Stephen Cooper, and Randy Pausch. 2001. Using visualization to teach novices recursion. SIGCSE Bull. 33, 3 (Sept. 2001), 109–112. https://doi.org/10.1145/507758.377507.
54. Willingham D. T. (2005) Do visual, auditory, and kinesthetic learners need visual, auditory, and kinesthetic instruction? American Educator, 29, 31–35.
55. Wu, Cheng-Chih & Dale, Nell & Bethel, Lowell. (1998). Conceptual models and cognitive learning styles in teaching recursion. ACM Sigcse Bulletin. 30. 292-296. 10.1145/273133.274315.