Project Description for REU Site PRiMAL: A Cyclic Process of PRogramming, Math, and ALgorithms to Guide Discovery and Computational Thinking

A Overview

We propose a novel cyclic process (as depicted in Figure 1) by which a student starts in one of three computational domains: programming, math, or algorithms; and cycles through them in that order, so as to make each pass enhance the next, until a certain discovery is made. The value of such an approach is two-fold. First, it integrates different elements of computational thinking that are essential for computer scientists and problem solvers in general. Second, it mimics to a great extent a natural process of learning, where people move from concrete observations (output of a **PR**ogram), to conceptualization and hypotheses (**M**ath), to active experimentation based on hypotheses and what has been learned (**AL**gorithms); new algorithms in turn can yield better programs and close the cycle. [1] Such view has been around in educational psychology since 1984 when popularized by Kolb as *experimental learning* [14]. Indeed, while the research topics are theoretical in nature (examples appear in Section B), the research process itself can still rely heavily on a component of experimentation, where students will design and try different algorithms, modify program parameters, make observations, and formulate a hypothesis that can later be proved mathematically.

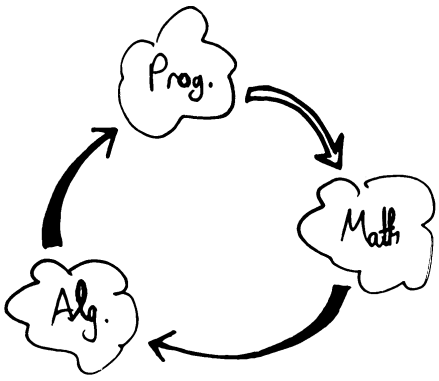

Figure 1: The cyclic process of PRiMAL that embodies observation, conceptualization, and experimentation.

In a computational field, the transition from Math, to Algorithms, to Programming may or may not be trivial, but it is definitely a natural one. It's the transition from Programming to Math (indicated by a white arrow in Figure 1) where the magic happens. We believe that this will be a moment when students put themselves in an active learner mode rather than simply being receivers of information (thus enabling themselves to become independent thinkers). After observing the output of their programs, possibly by running them with different parameters each time, students can draw conclusions and *conjecture a hypothesis*. This will jump-start their mathematical exploration now that they know what to look for. It will also convey that programming is not just about writing code, and cultivate a skill for *abstraction* and *generalization* that is much needed in computational thinking and problem solving.

The following scenario will illustrate and motivate how, through computational thinking, the cyclic process of programming, math, and algorithms can guide the student on a typical journey of discovery. Assume that we place a number $p \geq 3$ of points on a two-dimensional $n \times n$ grid, where $n > 1$. Given n, how large must p be to guarantee that three of the points make a right triangle (see Figure 3)? Observe that, with no guidance whatsoever, it is not clear how one would even approach this problem. But what if the student is encouraged to write a program to experiment with small values of p and n?

Level 1: Some students will be comfortable to immediately start writing code and modifying it as they go. Others might think about an algorithm first to determine whether some right triangle exists. Regardless, the code/algorithm need not be efficient, so one could simply try all possible placement of p points on an $n \times n$ grid and check every set of three points, which is feasible when p and n are small. By observing the output of the program for several small values of p and n, a student will be able to make a conjecture about p as function of n. The student will now embark on a mathematical adventure

[1] The acronym PRiMAL is chosen for the REU name as an indication of something that is essential, fundamental, primitive, and powerful.

knowing precisely what statement to prove, and will be guided through this phase by exploring various proof techniques. Once the existence of right triangles is established, the student might be asked to quickly find such a triangle when n is large; for instance, if the points keep moving on the grid. We are back to algorithms. Any attempt must now be accompanied by algorithmic analysis to claim the efficiency. Guidance can also be provided here by considering different types of data structures.

Level 2: One could go even further to ask for the minimum number of right triangles we can guarantee. The student can modify the algorithm to count all triangles for small values of n and every placement of the points. By observing the output of the program again, the student will conjecture a minimum and move to another mathematical exploration, thus cycling twice through the PRiMAL process.

In general, the research problems will be attractive (and fun) to students with a decent background in programming who have genuine interest in math, algorithms, and computation, and who would seek, under the proper guidance, the sub-field of theoretical computer science as a graduate or career choice; this is especially true for women and underrepresented minorities in STEM who shy away from the idea due to other overwhelming trends in computing.

A.1 Objectives of the REU

Based by on the above discussion, we define the objectives of the REU as follows:

1. Involve eight participants per summer (10 weeks) in the PRiMAL research.
2. Provide training in tools and technologies that are common for this area of research, such as C/C++, Python, mathematical writing and proofs, algorithmic analysis, online resources such as `oeis.org`, `wolframalpha.com`, and `desmos.com`, and the use of pseudocode and LaTeX (e.g. `overleaf.com`) for documentation. The training will also include a research and work ethics component, as discussed in detail in Section E and Broader Impacts.
3. Inspire participants to pursue algorithms, math, and theoretical CS as a graduate and/or career choice.
4. Target participation of women and minority groups, such as non-binary, as well as students from institutions with limited resources for research opportunities (e.g. no PhD programs).

A.2 Targeted student participants

Our targeted participants are women and minority students who love CS and math, most of whom will be from non-doctoral granting institutions. We will recruit at least 80% of the participants from institutions other than CUNY; therefore, the budget will support one or two students from CUNY colleges per year, with the rest being selected from external institutions. More CUNY students may be included through other internal funding mechanisms. In supporting objective (4) above, we aim for at least 50% participation from women and other underrepresented groups (e.g. non-binary), and from institutions with limited resources in providing research similar to ours, such as undergraduate teaching colleges.

A.3 Intellectual focus

The national STEM research scene is now overwhelmingly dominated by machine learning and data driven tools. Despite finding practical use in many disciplines, these trends do not offer much insight to a student who has not yet understood the fundamentals of research in computing, and who will often find today's technologies mysterious and misleading. Our REU seeks to renew the interest in algorithmic and mathematical research to build the skills in computational thinking and problem solving, which will eventually prepare and empower students to engage in any endeavor they may find interesting and fulfilling. To that end, our research problems will **i)** be original, **ii)** contain interesting mathematical notions, and **iii)** exhibit the potential for designing efficient algorithms. In addition, mathematics and algorithms offer versatile tools that are useful not only for future computer scientists, but also for professionals in many computational fields such as biology and physics. Our assessments will quantify the extent to which this REU will serve to inspire students to consider the learned research topics, and theoretical CS in general, as a career path, and pursue such discipline at the graduate level.

A.4 Organizational structure (team structure)

The REU will be held at Hunter College of CUNY with the support of the department of computer science. The team will consist of the PI, other mentors (COA documents and standard NSF letters of collaboration have been added to this proposal), two paid PhD students (in CS/math), and two paid highly motivated and qualified undergraduate students. The selection of the four students is described in Section E.2. The other mentors will include:

- Amotz Bar-Noy (Brooklyn College, CUNY), research focus in algorithms and graph theory. Amotz has been a collaborator of the PI on the yearly CUNY Math Challenge for undergraduates.
- Mayank Goswami (Queens College, CUNY), research focus in algorithms and data structures. Mayank supervised undergraduate students in his lab on several occasions, and he shares the PI's vision on this REU.
- Adam Sheffer (Baruch College, CUNY), research focus in discrete mathematics. Adam is passionate about undergraduate research in math/CS and is a longtime director of multiple REUs. Adam and the PI collaborated on NSF's Polymath Jr 2024.

All mentors are in CUNY and geographically close to Hunter College. In addition to mentoring, the team will work on new research problems that will heavily explore the PRiMAL cyclic process highlighted earlier. While the PhD students will play an active role in preparing research material and supervising participants (see section below), the presence of the undergraduate students will tangentially create opportunities to solidify and enrich the knowledge of exceptional undergraduate students, help them engage in peer tutoring, and provide them with a satisfying internship-like experience. This, perhaps non-typical, team structure of PI/mentors, graduate students, and undergraduate students is part of the PI's vision to foster a collaborative, fruitful, and intimate research environment. It does not downplay the essential role of the PI (and mentors) in directing and managing the research. It has been successfully implemented and tested during the summer of 2024 (see Sections B.5 and G).

A.5 Timetable (and responsibilities)

In the Fall semester, the PI will start working on the project descriptions, preparing the application material, making recruiting visits to nearby colleges and universities, and contacting friends, colleagues, and conference chairs/organizers for outreach. The PI will also solicit contributions from the three other mentors mentioned above to provide additional ideas that match the cyclic research framework (`-->PR-->M-->AL-->`) of PRiMAL. These mentors will also supervise students during the REU, where one mentor could supervise multiple participants (see Section B.5 and Footnote 2, page 6 for this paradigm). When possible, Fall preparations will take into consideration any learned experience from the previous year; evaluation and reporting for the previous year is also done in parallel during Fall.

The REU website will be updated at the beginning of the Spring semester when applications are considered and processed on a rolling basis, until finalized by the end of March. The two PhD students will help the PI finalize the research problems and prepare teaching material and warm-up exercises during the Spring semester, and will actively take part in guiding the participants once the REU starts. Their responsibilities include (see also Mentoring Plan):

- Write up documentation explaining the interplay of programming, math, and algorithms, and laying out potential research questions with suggested approaches (see Data Management Plan).
- Extend the REU problems to encompass settings, questions, and tools related to their own work, thus adding enrichment to both the participants' research and their own, while contributing additional perspectives and applications.
- Make sure solutions (albeit not necessarily known) and/or partial solutions are reasonably within reach for the duration of the REU.
- Help identify at least two levels of depth in pursuing questions and solutions, such as Level 1 and Level 2 described in the Overview section (participants can start with the easier levels).
- Provide additional teaching of preliminary material and warm-up exercises, especially during the first week, and hold weekly office hours (thus possibly help with assessment and evaluation), inspire participants to pursue graduate school, and provide a role model (during REU).

- Possibly, each PhD student will also closely work with four of the REU participants (alongside the PI and mentors).

The logistics of travel and housing will be finalized with participants in April. In April and May, the PI will also finalize the assignment of projects to participants and the schedule of talks and instructional sessions. The summer research program will start in June and will run for ten weeks (approximately 40 hours per week). During this time, the two undergraduate students will join for a tutoring/internship-like experience to further support a positive research environment among undergraduates (for instance, they may help out with coding). Evaluation and assessment of the ongoing activities take place during this time as well. When the summer REU is completed, participants will present their work. A summary of the timetable is outlined in Table 1.

Months	Activities
Sept. - Dec.	Preparation of research material, recruiting, previous year's feedback, annual evaluation and reporting.
Jan. - May	Application process, logistics, PhD students joining the team, finalizing and assignment of research problems, general scheduling of talks and tasks.
Jun. - Aug.	Duration of REU, evaluation/assessment conducted in parallel, undergraduate students join the team, participants present their work.

Table 1: Timetable for REU.

A.6 Departmental and institutional commitment

Hunter College has a strong culture of commitment to undergraduate research. The Undergraduate Research Initiative provides Hunter College students with the opportunity to work on research alongside faculty. The computer science department actively seeks to recruit women and minority through the CUNY 2X initiative, and supports undergraduate research through *supervised research* courses. In addition, both the department of computer science and Hunter College have prior experience in hosting REUs, and have committed physical space for the participants of this REU, while providing labs (including Linux, Mac, and Windows computers), Linux accounts for all the participants, and a one course release per semester for the PI for the entire duration of the REU. Therefore, both Hunter College and the department of computer science will welcome this REU on campus. For more information on departmental and institutional support, refer to the Facilities, Equipment and Other resources document within the proposal (a standard NSF letter of collaboration from the department chair is also provided).

The PI has long been an active member of the MIT-Hunter College outreach that supports on an annual basis the participation of six undergraduate students from various STEM disciplines in the MIT Workshop on Quantitative Methods. The PI has an extensive experience mentoring undergraduate students by frequently offering the supervised research courses in CS. The PI conducted a successful REU through NSF's Polymath Jr project in the summer of 2024 (see Sections B.5 and G). Finally, the three mentors listed in Section A.4 share the PI's vision and have committed their support.

B Nature of Student Activities

All participants will attend an orientation session on the first day of the REU to learn about the overall goals of the program and the nature/merit of the cyclic process of PRiMAL (`-->PR-->M-->AL-->`), meet the team (PI, mentors, PhD students, and undergraduate students), sign the necessary waivers and medical forms, get their accounts and building access cards, and tour the research space/labs provided by Hunter College and the computer science department. Following this, each participant will be working on activities pertaining to their assigned research project (after an initial week of teaching and warm-up exercises). In addition, there will be several group and social activities.

B.1 Group and social activities

The REU will include a number of social activities to foster a comfortable sense of community, belonging, and friendliness. These will be distributed over the entire REU period of ten weeks. Activities may include: Games to break the ice; daily teaching sessions during the first week to provide a reasonable background on the REU subjects; informal meetings to chat about research issues; sessions

on how to read CS/math papers; talks by visitors such as Chaim Goodman-Strauss from MoMath (a MoMath colleague of the PI who helped discover a new tiling shape), Adam Sheffer (who is a mentor on this REU), or other researchers such as Piotr Indyk and Dina Katabi from MIT (who are colleagues and personal friends of the PI); presentations on how to apply to graduate school, how to prepare NSF fellowship applications, and how to deliver effective talks; trips to Museums such as MOMA and MoMath (where the PI is a volunteer and integrator), and other New York specific activities such as walking on the high line; occasional group dinners and movie nights; and a final presentation event. Finally, the PI will make use of proximity to promote cohort interaction through social events and professional development talks/activities of other REUs in CUNY (for instance, joint lunches, dorm activities, CV writing workshops, etc...) to enrich the experience of the REU and provide a holistic academic perspective (see letter from N. Greenbaum, who also submitted an REU proposal).

B.2 Research activities, mentoring, and student progress

Each participant will receive a document describing in detail the research problem, the corresponding questions (grouped by level of depth), possible approaches to follow, warm-up exercises, and specific instructions to guide the PRiMAL cyclic process. Initially, The PI and mentors will meet with each participant on almost a daily basis to discuss the research progress, address issues that come up, and advise on how to proceed.The frequency of these meetings will then decrease gradually over time to allow the student some room for independence. However, a fixed longer weekly meeting will be scheduled for constant follow-up. In addition, each PhD student will closely supervise four participants, and will hold weekly office hours for everyone who wants to join (see Section A.5).

The PhD students (and to a lesser extent the undergraduate students) will provide continuous support and guidance on topics related to the research problem, the coding (C/C++ and/or Python), algorithmic issues, and the use of online resource when necessary, such as `oeis`, `wolframalpha`, `desmos`, and `overleaf`. These online resources provide the participants with useful information and a means of documentation. For instance, `oeis` (Online Encyclopedia of Integer Sequences) will help identify patterns, and possibly relate the research problem in a non-trivial way to other existing problems. `Wolframalpha` is a symbolic mathematical engine that can help establish certain mathematical identities and jump-start an effort for constructing proofs. `Desmos` is very useful when it comes to visualizing data and plotting functions. Finally, `overleaf` will be the REU's main LaTeX platform where participants share their findings and progress with the team. All participants will be trained to use LaTeX to document their work, including actual code, pseudocode, algorithmic analysis, and mathematical derivations/proofs (as described in the Data Management Plan document). LaTeX is well-equipped to handle all this data. Most importantly, all participants will be asked to maintain a LaTeX blog of their progress (updated at least weekly). Such blog may include failed attempts, successful attempts, and what has been learned. In addition, it may include challenges and questions addressed to the team, and a list of tasks grouped into stages such as past, current, and future. This will be effective in helping the participants manage their time, as well as communicating their progress (this approach has been successfully tried by the PI for the Polymath Jr REU in the summer of 2024).

The team will also foster a sense of community, collaboration, and collegiality by occasionally organizing sessions in which participants are encouraged to exchange ideas and discuss research problems among themselves (in the presence of the entire team). This arrangement will also provide the participants with a broader research exposure. Additional activities are in Section E.1.

B.3 PRiMAL leads to participant's independence

The ultimate goal of the REU is to provide the participants with a well-rounded experience in computing and problem solving using the thoroughly developed framework of PRiMAL. However, a truly rewarding experience is one where the participant can eventually achieve some level of independence in experimenting, formulating, and finding solutions. We believe that the PRiMAL process itself is crucial in that aspect, as it offers a general tool to develop independence naturally. For instance, and as illustrated in the Overview section, a participant can initially experiment with code to discover answers instead of trying to seek those answers mathematically. Once the facts are known, the math can follow. Therefore, we eliminate the need for providing a mathematical hint that may ruin the experience of an independent discovery. In parallel, careful guidance (by the PI, mentors, and the PhD students) will be provided to monitor the level of comfort for each participant, and minimize

interference as long as there is reasonable progress. We will strive to walk the thin line of balancing between hints, filling knowledge gaps, and actual reveals. In many cases, perfect solutions are not even known to us, and that by itself is a good setting to foster a level of independence and avoid any unintentional over-nurturing. Formative assessment will be conducted for participant's independence as part of the REU evaluation (last paragraph of Section F.2).

B.4 Examples of research projects

The following examples provide a general idea about the type of research problems involved and how they fit into the PRiMAL framework. Each problem is original, contains interesting mathematical notions, and offers the potential to develop algorithms. The problems are also playful, which embodies an element of exploration, possibly requiring little background, leading students to making significant progress. Nevertheless, each problem has different levels of engagements, starting from the simple exploration to a full immersion in research. This could inspire participants to continue working on the research beyond the duration of the REU. [2]

While we do not describe in detail how a participant might cycle through PRiMAL (possibly making multiple passes), we highlight for each problem five aspects: **i)** which of the three domains of the cycle (`PR, M, AL`) makes an appropriate start, **ii)** how the program/code can inform the math, **iii)** how the math can influence the algorithms, **iv)** which online resources can help, **v)** how can the problem be advanced further, and **vi)** a typical form of a result that students can establish (not known yet). Despite being original, the problems are not disconnected from existing work, and have deep roots in the literature as described in Section B.4.6 and Table 2, and can definitely lead to publishable results in computer science and mathematics conferences/journals (see Section B.5).

B.4.1 Infinite Skolem sequences

A Skolem sequence for $k \in \mathbb{N}$ is a sequence of $2k$ integers such that every $n \in \{1, 2, \ldots, k\}$ appears in exactly two positions at a distance n [25]. An example for $k = 4$ is $1, 1, 3, 4, 2, 3, 2, 4$. Skolem sequences exist for $k \equiv 0, 1 \pmod 4$. On the other hand, an infinite Skolem sequence, defined as $\{s_i : i \in \mathbb{N}\}$ where every $n \in \mathbb{N}$ appears in exactly two positions at distance n, can be obtained as follows: Having placed two copies of $1, 2, \ldots, n$, one idea is to place the first copy of $n+1$ in the first available gap, and the second at the appropriate distance. This gives a lexicographic infinite Skolem sequence, where n first appears before m whenever $n < m$: $1, 1, 2, 3, 2, 4, 3, 5, 6, 4, 7, 8, 5, 9, 6, \ldots$. For any given n, find a_n, the position of the first occurrence of n.

One could start with a program to generate $s_1, s_2, \ldots, s_k$ in an array for a large enough k and observe when n first shows up. Since we don't know how large k must be, an initial challenge is to work without this knowledge, or using structures that can be extended at run time. BUt there is a need for a better algorithm. By observing a_n for different values of n, one may be able to conjecture lower and upper bounds on a_n. The program can also help the student observe how the sequence $a_1, a_2, a_3, \ldots$ evolves, perhaps by looking at $a_n - a_{n-1}$. Once patterns emerge, `oeis` may be used to investigate whether anything is known about such patterns. Eventually, the student can potentially develop a better algorithm to compute a_n, possibly one that avoids arrays altogether.

The problem can be extended by requiring, for each integer n, k copies at distance n. For instance, if we follow the above strategy, the infinite sequence for $k = 3$ will begin with: $1, 1, 1, 2, \underline{3?}, 2, \_, 2, \_, \_, \_, \ldots$ But placing 3 will be blocked by the third occurrence of 2! To maintain the lexicographic order, integer 4 can be placed instead, and 3 has to wait. Which integers will be blocked, and will they ever find their place in the sequence?

Theorem (not yet established)**.** *The set of infinite k-copy Skolem sequences have a minimum under lexicographic order (this is equivalent to showing that the lexicographic construction above works).*

What are algorithms for constructing a k-copy (infinite) Skolem sequence, and how is a_n affected? What if we relax the rules to have $k \geq 3$ copies, but only require that some two consecutive copies must be at distance n? Each choice of the two copies changes a_n and defines a separate problem.

Theorem. *The position of the first n, denoted by a_n, in the infinite k-copy Skolem sequence constructed by Algorithm ... is given by $a_n = \ldots$.*

[2] Our mentoring model deviates from the model of placing each participant in a different lab, so the relation between mentors and participants is not necessarily one-to-one. The same is true for problems: a problem can recur as a research theme in subsequent years, and multiple participant can work on different aspects of the same problem (see Section B.5).

B.4.2 Three points make a right triangle

Given $p \geq 3$ points on a two-dimensional $n \times n$ grid where $n > 1$, how large must p be to guarantee that three of the points make a right triangle? Figure 3 shows an example with $n = 5$, $p = 8$, and no right triangles.

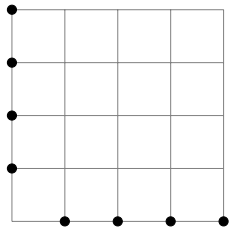

Figure 2: A grid of size 5 with 8 points and no right triangles.

The reader is referred to the Overview section for a detailed discussion on how this problem is exemplary for the PRiMAL cyclic process (`-->PR-->M-->AL-->`), with strong research components in programming, math, and algorithms. One could further add an interesting `wolframalpha` component to explore the use of matrices: The grid may be encoded as the adjacency matrix of a bipartite graph, where a point in row i and column j becomes the edge (i, j).

The number of right triangles that align with the x-axis and the y-axis can now be computed using matrix operations, because such triangles become analogous to paths of length 3 in the bipartite graph. This also draws the student's attention to the fact that some right triangles may not be aligned with the axes. What is the minimum number of right triangles?

Theorem. *Given a $m \times n$ grid and p points, the minimum number of right triangles of the form $(i, j), (i, \neq j), (\neq i, j)$ is ....*

Theorem. *Upon adding or deleting a point, the number of triangles of the form stated in the above theorem can be re-computed in ... time (what's the best we can do?)*

The problem can be nicely generalized to $n \times m$ grids, higher dimensions (e.g. 3D), and other shapes; for instance, one could be looking for rectangles instead of right triangles.

B.4.3 Blindsort

Blindsort is a new esoteric sorting algorithm (unpublished) invented by the PI. With Blindsort, an array $a[1 \ldots n]$ of elements cannot be read, copied, or explicitly modified. The only available operation is a blind `swap(i,j)`, which swaps the two elements $a[i]$ and $a[j]$. If a swap operation ever places an element in its correct position, then that element becomes *frozen*. When all elements are frozen, the array is sorted. Sort the array in the fastest way! The closest idea to Blindsort is a very different algorithm called Gorosort that appeared in Google Code Jam 2011 (an archive can be found in [1]).

The obvious starting point for this problem is the algorithm, either deterministic or randomized (for instance, one could repeatedly choose two random elements to swap). Programming follows, which could then provide a way to empirically compute the running time in terms of the number of swaps on some (random) input. With `desmos`, one can visualize data points, overlay different functions, and immediately observe the effect of parameters on those functions. For instance, one could plot cn, $cn \log n$, cn^2, etc..., and observe changes using a sliding bar for parameter c. This helps to conjecture a complexity for the running time before resorting to a rigorous time analysis. For randomized algorithms, the analysis of the expected running time may be harder, so one could either seek lower and upper bounds instead or, if ambitious, calculate it anyway.

Theorem. *There exists a randomized algorithm for Blindsort that runs in cn^2 expected time, where $c = \ldots$.*

Since frozen elements can be ignored, Blindsort shifts the focus from permutations, as the mathematical object of sorting, to derangements, which provides students with a different fresh setting that is typically not studied. Advanced questions can consider a lower bound on the time of the best deterministic blind sorting (thus establishing a lower bound analogous to that of comparison-based algorithms), the worst derangement for any given fixed algorithm, and special cases such as the effect of a bound $k \ll n$ on the number of distinct elements.

Theorem. *Every deterministic algorithm for Blindsort must run in ... time.*

Theorem. *In Blindsort, the worst-case input for Algorithm ... requires exactly ... swaps.*

B.4.4 Additional problems

Some other original, exemplary problems that exhibit the PRiMAL framework are "Other Towers of Hanoi" (recursion, optimization, and inductive proofs) [19], "Living on a Random Torus" (percolation and probability theory) [16], "Musical Scales" [17] (number theory and combinatorial words), and "The Intimidated Chickens" (counting and combinatorics) [15] (inspired by [6]). See also [18].

B.4.5 Notes on the significance of the problems (and their general research areas)

Skolem sequences are used in coding theory to build Steiner systems and block designs [24]. They are also useful for labeling graphs [24] and can be related to Fibonacci numbers, variants of the game of Nim [8, 29], and irrational spectrums [13]. The problem of making right triangles was inspired by an open problem of three points on a square grid, called the no-three-in-a-line problem [3]. It can also be related to other computational geometry problems and a class of 3SUM problems [10, 11]. Esoteric sorting algorithms are a perfect vehicle to engage students in interesting and often non-trivial algorithmic analysis (especially time complexity); examples can be found in [7, 12, 23]. Blindsort is a new such algorithm invented by the PI. Algorithms for Blindsort can help understand optimization of blind experimental setups, where outcomes can't be determined unless one actually performs an experiment, e.g. in Biology. Variations on the Tower of Hanoi have long been considered in both the educational and the research community; see [19] for some ideas and references to literature. The problem of living on a random torus studies properties of islands and pools, which is closely related to percolation and the formation of giant clusters in physics [28], and has applications related to connectivity in random graphs. Musical scales given by points $(k, ak \bmod n)$ for $k \in \{0, 1, \ldots, n-1\}$, where peaks represent the sharp notes, exhibit Myhill's property of diatonic scales, which in turn results in the "circle of fifth" and other musical properties [5]. These scales are also connected to Christoffel words [4] and other combinatorial patterns. The intimidated chickens problem was inspired by a question in the 2017 Raytheon Mathcounts National Competition [6], and can be related to interesting binary patterns and many combinatorial problems in the literature [27, 26], which will lead to a deeper understanding of its combinatorial nature and, hence, new discoveries. Table 2 provides a summary.

	Start	Object of study
Infinite Skolem sequences	`PR`	sequences, proofs, irrational spectrums, coding theory
Three points/right triangles	`AL, PR`	algorithms and data structures, proofs, bipartite graphs, computational geometry and 3SUM problems
Blindsort	`AL`	derangements, algorithmic analysis, lower bounds
Other towers of Hanoi	`M, AL`	recursion, optimality, induction proofs
Living on a random torus	`PR`	graph search and connected components, probabilistic analysis, limit properties, percolation
Musical scales	`PR`	number theory, musical chords, patterns, Christoffel words
The intimidated chickens	`PR, M`	combinatorics, recurrences, enumeration algorithms

Table 2: Summary of the example research problems and their connection to literature.

B.5 Proof of concept (Polymath Jr 2024 and other undergraduate research)

The PI pioneered the above problems and presented some of them in May 2024 as part of the prestigious Math Encounters series sponsored by MoMath and the Simons Foundation [22, 20]. Additionally, he made notable advancements in these problems with undergraduate participants in the summer of 2024, while assessing the following questions:

- **Q1)** Does the PRiMAL paradigm work?
- **Q2)** Can undergraduate participants work on the research with minimal background?
- **Q3)** Is overleaf effective in tracking progress and as a follow-up tool?
- **Q4)** Does the role PhD students and undergraduate students on the team make sense?
- **Q5)** Do the problems connect to literature and lead to publications?

Throughout this process, the PI conducted an 8-week summer REU, using the problem of *infinite Skolem sequences*, and the full team structure suggested in this proposal, with undergraduate participants who do not necessarily have access to research in their institutions:

- *Infinite Skolem sequences*, an REU under the umbrella of the NSF funded Polymath Jr 2024 [21], with undergraduate participants (eight of them) Benjamin Gildea (UC Berkley), Jack Rosenthal (Princeton), Aahan Chatterjee (University of Waterloo), Sambhu Ganesan (Lynbrook High School), Luca Viscito (Bennington College), Valentio Iverson (University of Waterloo), Yusheen Wang (Bryn Mawr College), and David Phillips (University of Pittsburgh); PhD students Paul Cesaretti (CUNY) and Saman Farhat (CUNY); and undergraduate student Ryan Vaz (Hunter College). (Results from this REU are covered in Section G.)

The PI also started two additional undergraduate research projects to study other problems.

- *Three points make a right triangle*, with undergraduate participants John Lee (Hunter College) and Salma Abu Said (American University of Beirut).
- *Blindsort*, with undergraduate participant Christopher He (Hunter College, now at Google Inc.).

Note: None of the three above problems has been exhausted, and they all still offer many more questions to address in research. In fact, starting them only opened the door for us to realize their actual potential; for instance, as shown below, participants were able to formulate questions that were not even part of the initial problem setting. Nevertheless, by no means will we stop inventing new problems (see also Footnote 2, page 6). Here is the PI's assessment of questions **Q1**-**Q5**:

Q1. The PRiMAL cycle (`-->PR-->M-->AL-->`) was essential in making discoveries. For instance, knowing that $\phi_k = (k-1)(1+\sqrt{1+4/(k-1)})/2$ is an important quantity in a k-copy Skolem sequence where the last two copies of n are at distance n, Sambhu observed by writing a program that the position of the first n is given by $a_n = \lfloor[(k-1)n-(k-2)]\phi_k/(k-1)\rfloor$, which was then proved mathematically in a collective group effort by Sambhu, David, Saman (PhD student), and the PI. Valentio observed through code that a 3-copy Skolem sequence constructed lexicographically exists, which later inspired the mathematical work by Benjamin. Salma wrote a program to compute the minimum number of triangles in an $m \times n$ grid with $m+n-1$ points, by exhaustive enumeration of all possible placements of points. The code was too slow, so she figured out an algorithm to speed it up. Upon rewriting the program, more evidence was gathered for larger values of m and n, which led to a proof. Code by Christopher suggested one deterministic algorithm had $n^2/6$ expected running time under a random derangement. This was then proved rigorously using probabilistic analysis.

Q2. After an initial week of introductory learning material and warm-up exercises, the undergraduate participants were able to address the research question and even formulate their own. Luca explored a 3-copy Skolem sequence that grows in both directions. Sambhu explored a k-copy Skolem sequence where the distance between copies of n is some function $f(n)$, e.g. $f(n) = 2^{n-1}$. Benjamin invented a completely new way of constructing k-copy Skolem sequences that led to a proof that such sequences actually exist and that they are uncountable. Jack (with help from PhD students Saman and Paul) worked on a question about an online algorithm (with a buffer) for placing incoming integers in a Skolem sequence while minimizing gaps. Aahan decided to construct inexact k-copy Skolem sequences with provably bounded errors. Yusheen identified a connection to Beatty sequences. Salma explored the 3SUM and the 3SUM-indexing problems to work on an algorithmic lower bound question related to updating the number of triangles upon adding/deleting points. John explored an algorithm to find a triangle in $O(1)$ time if points were moving. Christopher formulated a Blindsort version with an oracle, which led to considering a new class of algorithms.

Q3. Overleaf proved to be very effective in three ways: 1) participants were able to quickly communicate their work in almost finished form to the PI, who in turn provided timely guidance and feedback within the document, 2) the practice of writing down their thoughts actually helped the participants make progress and share their work with others (everyone had access to overleaf) who sometimes were able to pick up on the work and contribute, and 3) such documentation is now proving to be crucial for writing up papers as a culmination of the research done in the summer.

Q4. Through weekly office hours, the PhD students were able to guide the participants in interesting directions that intersect with their own research (see anecdotes in Q1 and Q2 about Sambhu and Jack). The two PhD students, Saman Farhat and Paul Cesaretti, formulated new research questions based on work they are involved with (see also Mentoring Plan). For instance, a new direction

about the algorithmic Skolem placement of incoming integers using offline and online strategies (with or without buffers) was the sole contribution of Saman and Paul. The undergraduate participants, especially Jack Rosenthal, made important advancements to this idea. In addition, the undergraduate student, Ryan Vaz, a double major in CS and math, was a tremendous help for those who were more mathematically inclined and had no extensive programming experience.

Q5. The summer research resulted in publishable work for all three projects, and participants were able to identify and reveal important literature, such as the connection of Beatty sequences to infinite Skolem sequences. The PI and the participants are now preparing multiple papers: A paper on infinite Skolem sequences for the journal of Discrete Mathematics (Elsevier), and another for the journal of Integer Sequences and/or the Fibonacci Quarterly, and possibly two for the Joint Mathematics Meeting (JMM) 2025 and COCOA 2025; a paper on the placement of points on a grid and the formation of triangles for the MoMath conference MOVES 2025, and another journal version for Discrete Mathematics; and a paper on Blindsort for the journal of Theoretical Computer Science (TCS), and for the biyearly conference Fun With Algorithms 2026 upon future developments.

C The Research Environment

Hunter College and the department of computer science provide a rich environment for undergraduate research (see Section A.6). The PI is a researcher who specializes in discrete mathematics and algorithms, and teaches those topics at Hunter College and the Graduate Center of CUNY. He also has experience in mentoring undergraduate students in research, often leading to posters in the Hunter Undergraduate STEM Research Conference. The PI oversees, through his affiliation with MIT's CBMM, the MIT-Hunter College outreach where he recruits yearly six undergraduate students in all STEM disciplines at Hunter College, especially from minorities in RISE, MARC, IMSD, McNair, or BP ENDURE programs, to participate in the MIT winter Workshop on Quantitative Methods. On average, the PI manages to place one of these students in a summer research internship at MIT, and at least half of them in some other university summer research programs. All of those who finish their research internships go to graduate school. The PI is also a member of the CUNY CoSSMO center, with space carefully designed to support undergraduate research. In addition, the PI frequently runs supervised research courses for undergraduates in the computer science department.

The PI enthusiastically supports minority students' access to research. For instance, last year, he actively helped a non-binary student who is passionate about mathematics, tahda (Lai) queer, to successfully find an REU matching their interest (NYC Discrete Math). In addition, he helped Vladislav Vostrikov, whose family is struggling to achieve legal status in the USA, to get accepted into MIT's summer research program MSRP, conditional on funding availability, and single-handedly secured $4,000 in funding for him from the CS department and the VP of student affairs at Hunter.

The PI will strive to foster a healthy relationship among the participants and the team members. Therefore, the team members will be selected based on not only their knowledge and ability to guide the participants, but also the character and diversity they bring to the program (the diversity of the participants themselves is discussed in Sections A.2 and D). Moreover, the PI will maintain an ongoing contact with the participants after the program ends to track their progress and provide advice on career choices and graduate school admissions. The PI will also accompany the participants to conferences when work originating from the REU research is accepted for publication.

D Student Recruitment and Selection

Recruitment. We will deploy several promotional strategies. We will create brochures and links to the REU website (once it's up and running), speak with faculty, send early emails to our colleagues, and make presentations to undergraduate students at colleges and universities that have programs in computer science, mathematics, or related computational fields, both in the geographical neighborhood and by zoom. We also plan on posting information about the REU in several places, including our website, The Computing Research Association (CRA), NSF's list of REU sites, social media outlets, and venues that support undergraduate minorities such as SCANAS and the Field of Dreams conference. We will ask organizers of other REUs (e.g. MIT's Workshop on Quantitative Methods, Polymath Jr, and others) to announce ours. In addition, we will make use of NSF's ETAP system. We will also rely on personal communication with longtime REU professors to identify students

who love CS and math but are isolated from the information, and would probably stay where they are if not appropriately reached.

Selection. In order to satisfy NSF's requirement for REU sites, the PI will select highly qualified applicants while keeping the demographic targets of objective (4) in mind. We will carefully look for promising applicants whose CVs are non-polished due to their less-than-ideal lifestyle (e.g. working to support their studies). We seek to recruit at least 50% of our participants from underrepresented groups and institutions with limited resources, and at least 80% from institutions other than CUNY.

In order to guarantee a strong cohort of REU participants, we will rely on personal communication with REU directors and conference organizers, and seek recommendation letters for students who have unusual potential but limited opportunities. When looking at applications, we will focus on letters, personal statements, and any additional items related to the applicant's background, and give less priority to transcripts.

E Student/Mentor Development and Expectations of Behavior

E.1 Student professional development

As described in the project overview, the PRiMAL approach will contribute to the professional development of the students by integrating different elements of computational thinking that are essential for computer scientists and problem solvers in general. This is done by leveraging the cycle (`-->PR-->M-->AL-->`) which, at its core, re-enforces a natural learning mechanism to explore theoretical subjects while allowing experimentation.

In addition to the intellectual value gained from using specific programming languages, dealing with mathematical notions, writing derivations and proofs, and developing and analyzing algorithms, students will be trained to make use of relevant online resources such as `oeis.org`, `wolframalpha.com`, and `desmos.com`, as well as document their work using pseudocode and LaTeX (e.g. `overleaf.com`). As part of their documentation, students will be asked to maintain a blog describing their progress. The very nature of the cyclic research approach will produce multiple iterations of code (and algorithms), which can serve as an assessment tool, as well as a way to monitor progress while pacing the research activities. Students will also be mentored to present their findings and make effective presentations. Towards the end of the REU, students will be required to provide results of their summer research in a form of a conference paper (using what they have learned about LaTeX). Section B.1 lists a number of group events/presentations that contribute further to the professional development of the participants, including effective communication, graduate school applications, and fellowships.

Finally, the three technical aspects of PRiMAL (`PR, M, AL`) will be explored in a novel way to convey responsible and ethical conduct of research. Many educational initiatives in STEM strive to include an ethical component in research, making one aware of the danger of seeking quick results and mishandling data (whether intentionally or not). The PRiMAL approach easily lends itself to this disposition by creating a systematic, heightened such awareness, leading students to perform self-reflections: Do different versions of code produce the same result? Does the program faithfully implement the algorithmic specification? Are the computational results and algorithms in accordance with the math? These questions incite students to be responsible, inspire more confidence in the research, strongly validate the outcomes, and are intrinsically verifiable given the nature of PRiMAL (for further discussion, see formative evaluation in Section F.2 and Broader Impacts in Section H.3).

Note: The professional development of the PhD students on the team (both as researchers and mentors) is detailed in the Mentoring Plan.

E.2 Research mentors selection

To ensure a successful and inspiring REU, the team will consist of (see also Sections A.4 and A.5):

- The PI, an expert in the field and the mentor;
- Other mentors (e.g. Amotz Bar-Noy, Mayank Goswami, and Adam Sheffer) whose research is also in algorithms, mathematics, and theoretical CS;
- two PhD students in computer science, mathematics, or related computational field who **i)** have substantial research experience and academic credentials to handle technical problems in CS/Math, and **ii)** have previously taught a course in computer science or mathematics; and

- two highly motivated and qualified undergraduate students who have excelled in at least **i)** three courses in programming (including algorithms) and **ii)** a course in discrete mathematics, and **iii)** who have some experience in tutoring at the math learning center at Hunter College.

Additional colleagues of the PI can be selected based on their academic profile and their ability to design interesting research problems supporting PRiMAL. Moreover, the team members, including mentors and students, will be selected based on not only their knowledge and ability to guide the participants, but also the character and diversity they bring to the program (the diversity of the participants themselves is discussed in Sections A.2 and D).

E.3 Qualification of mentors

The PI is a researcher specializing in algorithms, mathematics, and theoretical computer science. He is also very passionate about teaching programming, algorithms, and discrete mathematics, which he has done for over two decades. He has research publications in those fields, both technical and educational. He frequently receives personal letters from students who have finished his class, expressing gratitude and highlighting how his teaching has made a difference in their lives (those letters are available upon request from NSF). His passion to teach about mathematics led him to volunteer at the MoMath museum (MoMath integrator), where he actively helped kids and their parents understand the mathematics behind the exhibits. He also delivered at the MoMath lessons for middle school kids, as well as general research talks. He helped conceive and create the bioinformatics concentration in five STEM departments at Hunter College, and he continues to guide undergraduate students through the MIT-Hunter College outreach program on computational methods, which encourages and supports minorities. He offers supervised research courses in computer science for undergraduate students on a regular basis. Section C has more detail on involving undergraduates. The PI ran a summer REU in 2024 (Sections B.5 and G). Hence, the PI is highly qualified to mentor the participants, and effectively create an excellent team of helpers to achieve the REU goals: Amotz Bar-Noy, Mayank Goswami, and Adam Sheffer share similar foundation, experience, and passion for undergraduate/minority research; and Adam is a longtime director of REUs. Section E.2 explains the selection of student mentors.

E.4 Expectation of behavior

We will closely follow Hunter College and CUNY guidelines for ethics, employee sexual misconduct prevention and response, and workplace violence prevention. All participants will receive training in these categories through Hunter College to ensure a safe, respectful, and harassment-free environment.

F Project Evaluation and Reporting

We describe summative assessment for the four specific REU objectives, as well as general formative assessment of the entire program, and finally the reporting and evaluation of the REU results.

F.1 Summative assessment of REU objectives (listed in Section A.1)

We will follow the general guidelines of evaluation described in the CISE REU Toolkit, using constructs and metrics, as illustrated in Table 3.

F.2 Formative assessment of PRiMAL (sometimes using PRiMAL itself)

Formative assessment of objective 2 (the main objective) will evaluate the effectiveness of the PRiMAL approach as well as provide the necessary feedback for improving the REU in subsequent years. Such assessment will take advantage of PRiMAL's unique framework (`-->PR-->M-->AL-->`), which places it as a *logic model* [9] where each stage of progress feeds into the next, hence making evaluation an inherent part of the learning process itself. Logic models may have bidirectional links and feedback loops, so such effects come naturally given the PRiMAL cycle. We will develop strategies that closely follow the methods highlighted in [9] and the CISE REU Toolkit, but to give a concrete idea, consider the following two scenarios (refer to the description of problems in Section B.4).

Right triangles (scenario 1): Given a set of points, a student looks for right triangles using exhaustive search → output reveals there are always some right triangles → student conjectures and proves the existence of at least one right triangle → an efficient algorithm to find a right triangle is now obtained knowing there must be one → faster program allows to examine more examples and larger instances, leading to more mathematical discoveries.

Obj.	Construct	Metric(s) and approach
1	Research exposure, activities, and knowledge	Require participants to maintain a blog on `overleaf` updated at least weekly and shared with PI; content of blog: code, mathematical writing, and algorithms (pseudocode and analysis) to assess level of research exposure, activities, achievement, and progress; periodic surveys/interviews/meetings to measure the PI-assessed, as well as a self-reported, level of comfort with research; our target: all 8 participants successfully engage in research.
2	Computing knowledge	Blog reveals improvement in understanding concepts, and efficacy of maneuvering the (`-->PR-->M-->AL-->`) cycle of PRiMAL; mid-program surveys to gauge the self-reported usefulness of online resources such as `oeis`, `wolframalpha`, and `desmos`; periodic surveys to monitor the perceived insight, effectiveness, and ethical component gained from transitions: `PR-->M`, `M-->AL`, and `AL-->PR`; final presentations/possible paper writing to assess the overall success of REU; our target: all 8 participants appreciate PRiMAL and produce non-trivial results, and most of the results make it to publications; **formative assessment** to evaluate the premise of PRiMAL.
3	Computing commitment and attitude	Use email (and/or slack) and longitudinal surveys to track enrollment in graduate programs; pre- and post-surveys to measure changes in self-reported attitude towards research; our target: 50% to pursue theoretical CS (or related field).
4	Demographics	Measure self-reported demographics of participants; our target: at least 50% women and minority and at least 80% from institutions with limited resources (e.g. no PhD programs).

Table 3: Summative assessment of the four REU objectives of Section A.1.

Infinite Skolem sequences (scenario 2): A student generates a 3-copy Skolem sequence using inefficient code → output reveals a pattern for a_n → student conjectures and proves an expression for a_n → an efficient generation algorithm based on a_n is then obtained → faster program allows to examine more properties.

The above two PRiMAL cycles (`PR-->M-->AL-->PR-->...`) provide a template for formative assessment. For each student who successfully moves through such a cycle, we will collect the programming constructs, proof techniques, algorithmic paradigms, and data structures that made this happen. For those who don't, we will identify what went wrong. Those lessons learned will help us improve our techniques and design new and better research problems. In addition, to assess the premise and validity of the approach itself, i.e. that the PRiMAL cycle is indeed a vital "ingredient" for productive research and successful discoveries, we will adopt the following strategy: Every student who is working on a given problem will be randomly assigned **only one** part of **another problem**, either `PR`, or `M`, or `AL`, thus dropping the inherent guidance within the PRiMAL cycle. For example, "`PR`: write a program to output all right triangles", "`M`: find the number of right triangles", "`AL`: design an efficient algorithm to find a right triangle", and so on... The degree of success of these "random assignments", when attempted "in isolation", will serve as the control to measure the effectiveness of PRiMAL.

As discussed earlier, students will be asked to explicitly address the following questions in their research blog: **i)** Do different versions of the code produce the same result? **ii)** Does the program faithfully implement the algorithmic specification? **iii)** Are the computational results and algorithms in accordance with the math? While this activity helps inspire confidence in the research, it also cultivates a sense of ethical awareness (this is described in more detail as a broader impact in Section H.3). We will keep track of the student answers to these questions at different stages of their research experience to assess their self-reported awareness of ethical practices and habits. However, of particular importance are **i)** *How* a student figures out the answers to these questions, and **ii)** *What* the student does upon finding a No answer for any given question. Therefore, we will also instruct the students to elaborate on the *how* and the *what* to assess other aspects of PRiMAL.

The *How*: This will allow us to give feedback about whether the student is correctly handling all

three aspects of the research (`PR, M, AL`). It will also provide us with concrete examples of the extent to which **PRiMAL is self-checking**. This will help us improve our designs and research questions.

The *What*: This will give us an indication of whether the student was able to correct a mistake (either a bug in the program, or a logical flaw in the math, or an incorrect algorithm). This self-administered corrective step should re-enforce the learning and give a sense of achievement and satisfaction, as students will be independently steering their work. We will collect such instances to measure the extent to which **PRiMAL enables student independence**.

F.3 Reporting

Reporting of the REU results, including evaluation and publications, will be done in the Fall of each year as described in the timetable of Section A.5. In addition, the work will be disseminated through journal/conference publications when possible (we list some journals/conferences in the Data Management Plan). Finally, the website of the REU will be maintained on a regular basis, which will include project descriptions, results, pedagogy (see Broader Impacts below), and a list of publications.

G Results from Prior NSF Support (by the PI but in CUNY)

In the summer of 2024, the PI was a mentor on DMS-2341670, entitled "The Polymath Jr Program", funded by NSF for $257,499 from 04/01/2024 to 03/31/2027 (Baruch College). It is an online summer program that provides hundreds of college students the opportunity to participate in mathematical research. The PI ran his CS/math project on infinite Skolem sequences (Section B.4.1) as an 8-week REU in Polymath Jr 2024 (see also Section B.5) [21]. Since the REU concluded around the time of the writing of this proposal, the following results from the REU may not be accessible in published form, but some drafts can be found in [2]. The PI and participants are now preparing multiple papers for the journals of Discrete Mathematics, Integer Sequences, Fibonacci Quarterly (pending a result on infinite Skolem sequences and Fibonacci words), and conferences JMM and COCOA 2025.

- A k-copy Skolem sequence exists for every k; The set of all k-copy Skolem sequences is uncountable; The set of k-copy Skolem sequences has a minimum in the lexicographic order (preliminary proof) (B. Gildea).
- There exists (explicit construction) a k-copy Skolem sequence where the distance between every two consecutive n's is b^{n-1}, for $b \geq 1$ (S. Ganesan); There exists (explicit construction) a k-copy Skolem sequence where the distance between every two consecutive copies of n deviates from n by $O(k)$ (A. Chatterjee and Y. Wang); There exist two 3-copy Skolem sequences, each covering a subset of the integers, such that the two subsets partition $\mathbb{N}$ (empirical) (L. Viscito).
- In a k-copy Skolem sequence where only the last two copies of n need to be at distance n, $a_{n,i} = \lfloor[(k-1)n-(k-1-i)]\phi_k/(k-1)\rfloor$ for $i \leq k-1$, where $a_{n,i}$ is the position of the ith copy of n and ϕ_k is a known irrational number (S. Ganesan and D. Phillips); In a k-copy Skolem seqauence where only the first two copies of n need to be at distance n, $a_{n,i} = n + (k-1)(2^{\lfloor \log_2 n \rfloor} - 1)$ (J. Rosenthal).
- Given that integers in S fall in $\{1, 2, \ldots, n\}$, there exists an online algorithm that places two copies of each in a Skolem sequence with $O(\sqrt{|S|})$ gaps, and zero gaps if a buffer of size $O(n^2)$ is used; Given the set $S = \{1, 2, \ldots, n\}$, a greedy offline algorithm that places two copies of each integer in a Skolem sequence starting with the smallest achieves $O(n/\phi)$ gaps, where ϕ is the Golden Ratio; Given the set $S = \{1, 2, \ldots, n\}$, a greedy offline algorithm that places two copies of each integer in a Skolem sequence starting with the largest achieves $\approx 0.08n$ gaps (empirical) (J. Rosenthal).
- The k-copy Skolem sequence obtained by placing the next integer in the first gap that works leaves position 5 unfilled (and this will be the only such gap) (empirical); The k-copy Skolem sequence that drops an integer if it cannot place it in the first available gap satisfies $1/x+1/(1+x)+\ldots+1/(k-1+x) = 1/(1-r)$, where $x = \lim_{n\to\infty} a_{n,1}/n$ and r is the dropping rate (V. Iverson).

H Broader Impacts

H.1 Broader impacts on underrepresented/minority career track

The program will contribute to the national talent pool of students by renewing interest in computing and mathematical research. It will also broaden participation in theoretical CS by recruiting women and students from underrepresented minorities and from institutions with limited research opportunities. Hunter College and CUNY have a strong, established track record in recruiting underrepresented students. According to Hunter College, the undergraduate student demographics in

2017 were as follows: 65% women, 23.8% Hispanic, 32.8% Asian or Pacific Islander, 12.1% black, and 31.1% white. Hunter College provides higher education to a largely disadvantaged population that includes women and underrepresented minorities, immigrants or the children of immigrants, and first generation college students. The CUNY Graduate Center's student population is roughly 57% women and 36% in underrepresented groups. The PI has always sought to promote computer science and mathematics in STEM programs that encourage underrepresented minorities. Between 2016 and 2019 he was a volunteer at the museum of mathematics MoMATH, where he helped students of all ages and their parents understand the mathematics and the computational aspects behind the exhibits.

H.2 Broader impacts on pedagogy

The research outcomes will not only impact computer science in an academic sense by virtue of exploring new problems, but the PRiMAL approach (with an *experimental learning* paradigm) will also offer a general investigative tool that is viable in many computational disciplines such as biology and physics. In addition, PRiMAL emphasizes a much-needed cultural perception in CS that programming is not just about writing code, but rather a tool for exploration and discovery. From an educational point of view, the technical framework itself encourages independent thinking and can serve as an assessment tool. For instance, multiple iterations of the code obtained through the cyclic process (`-->PR-->M-->AL-->`) will provide good indicators for evaluating the intellectual progress one makes. In addition, by observing the work of a student in three separate but related domains (`PR, M, AL`), we will be able to identify specific knowledge gaps and attend to them. For instance, a student who is comfortable writing code but lacks the ability to express their ideas algorithmically (whether verbally or in pseudocode) can be identified. The same is true for someone who has hard time capturing the mathematical notions after observing examples of program output. This latter ability of *generalization* is important in the learning process, as it embodies the act of *conjecturing a hypothesis*, which is central to scientific research. Finally, the cycle (`-->PR-->M-->AL-->`) closely mimics logical models of evaluation, where each component acts as an input to the next and, therefore, PRiMAL can be easily integrated as a tool in assessment procedures (as described in Section F.2).

We will share on our REU website not only the specific projects, but also our experience with the PRiMAL approach as an evolving pedagogy on its own. When appropriate, we will publish our methodologies in educational conferences and journals so that other educators and researchers can apply them in different contexts. Note: Another pedagogical broader impact is related to training graduates in this unique framework (see Mentoring Plan), as well as providing undergraduates on the team with an exciting internship-like experience.

H.3 Broader impacts on research/work ethics

Many educational initiatives in STEM strive to include an ethical component in research, making one aware of the danger of seeking quick results and mishandling data (whether intentionally or not). The PRiMAL approach easily lends itself to this disposition by creating a systematic, heightened such awareness. Table 4 lists some ethical concerns and their PRiMAL counterpart.

Ethical concern	**PRiMAL counterpart**
data is inadequate	Program/alg. can guarantee all data is generated
Proving a particular statement while there is a better one	Empirical data from program output consistent with mathematical claim
Favoring convenient implementation over efficiency	Program faithfully implements algorithmic specs
Heuristic fails in overlooked cases	Algorithmic analysis backed up by math
Result is a computational coincidence/artifact	Different versions of code produce same result

Table 4: How PRiMAL can promote ethical practices/habits and confidence in research.

REU participants will be asked to explicitly perform the following self-reflective tasks and report them in their blogs: **i)** Do different versions of the code produce the same result? **ii)** Does the program faithfully implement the algorithmic specification? **iii)** Are the computational results and algorithms in accordance with the math? These tasks are intrinsically verifiable given the nature of the PRiMAL approach, and will be assessed as part of the evaluation plan; the handling of these tasks by the students will also provide some input on how much independence they acquired in research (refer to Section F.2).

References

[1] Google Code Jam 2011. Gorosort. https://zibada.guru/gcj/2011qr/problems/#D.

[2] Polymath Jr 2024. Infinite skolem sequences, participants documents and drafts on overleaf (the blogs). https://www.overleaf.com/read/wspyvxmyvspf#1d0309.

[3] Michael A Adena, Derek A Holton, and Patrick A Kelly. Some thoughts oh the no-three-in-line problem. In *Combinatorial Mathematics: Proceedings of the Second Australian Conference*, pages 6–17. Springer, 2006.

[4] Jean Berstel. *Combinatorics on words: Christoffel words and repetitions in words*, volume 27. American Mathematical Soc., 2009.

[5] John Clough and Gerald Myerson. Variety and multiplicity in diatonic systems. *Journal of music theory*, 29(2):249–270, 1985.

[6] 2017 Raytheon Mathcount National Competition, 2017.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms second edition.(2001) problem 7-3. *Google Scholar Google Scholar Digital Library Digital Library*, 2001.

[8] Harold Scott Macdonald Coxeter. *The golden section, phyllotaxis, and Wythoff's game*. Scripta Mathematica, 1953.

[9] J Frechtling and L Sharp. User-friendly handbook for project evaluation. *NSF02-057. NSF, Arlington, VA. Retrieved May*, 10:2005, 2002.

[10] Anka Gajentaan and Mark H Overmars. On a class of o (n2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[11] Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. Data structures meet cryptography: 3sum with preprocessing. In *Proceedings of the 52nd annual ACM SIGACT symposium on theory of computing*, pages 294–307, 2020.

[12] Hermann Gruber, Markus Holzer, and Oliver Ruepp. Sorting the slow way: an analysis of perversely awful randomized sorting algorithms. In *International Conference on Fun with Algorithms*, pages 183–197. Springer, 2007.

[13] Ron Knott. The golden string of 0s and 1s. https://r-knott.surrey.ac.uk/Fibonacci/fibrab.html.

[14] David A. Kolb. Experience as the source of learning and development. *Upper Sadle River: Prentice Hall*, 1984.

[15] Saad Mneimneh. The intimidated chickens. https://www.cs.hunter.cuny.edu/~saad/courses/dm/chickens/.

[16] Saad Mneimneh. Living on a random torus. https://www.cs.hunter.cuny.edu/~saad/research/torus.pdf.

[17] Saad Mneimneh. Musical scales. https://www.cs.hunter.cuny.edu/~saad/other.php.

[18] Saad Mneimneh. Primal. https://www.cs.hunter.cuny.edu/~saad/PRiMAL/.

[19] Saad Mneimneh. Simple variations on the tower of hanoi: A study of recurrences and proofs by induction. *Teaching Mathematics and Computer Science*, 17(2):131–158, 2019.

[20] Saad Mneimneh and Momath. Math encounters. https://www.youtube.com/watch?v=ND6tTjrO72w.

[21] Saad Mneimneh and Adam Sheffer. Infinite skolem sequences, polymath jr. 2024 reu. https://www.cs.hunter.cuny.edu/~saad/Polymath_Jr/.

[22] MoMath and Simons Foundation. Math encounters. https://momath.org/math-encounters/previous-math-encounters-presentations/.

[23] Timothy J Rolfe. Perverse and foolish oft i strayed. *ACM SIGCSE Bulletin*, 40(2):52–55, 2008.

[24] Nabil Shalaby. *Skolem sequences: generalizations and applications*. PhD thesis, McMaster University, 1991.

[25] Th. Skolem. On certain distributions of integers in pairs with given differences. *Mathematica Scandinavica*, pages 57–68, 1957.

[26] N. J. A. Sloane. Entry a006498 in the on-line encyclopedia of integer sequences, 2023.

[27] N. J. A. Sloane. Quarter-squares, entry a002620 in the on-line encyclopedia of integer sequences, 2023.

[28] Dietrich Stauffer and Ammon Aharony. *Introduction to percolation theory*. CRC press, 2018.

[29] Willem A. Wythoff. A modification of the game of nim. *Nieuw Arch. Wisk*, 7(2):199–202, 1907.