SAGE Project Proposal Spring 2019

Parson's Puzzle Library

Data and Loop Puzzles

Calvin Goah

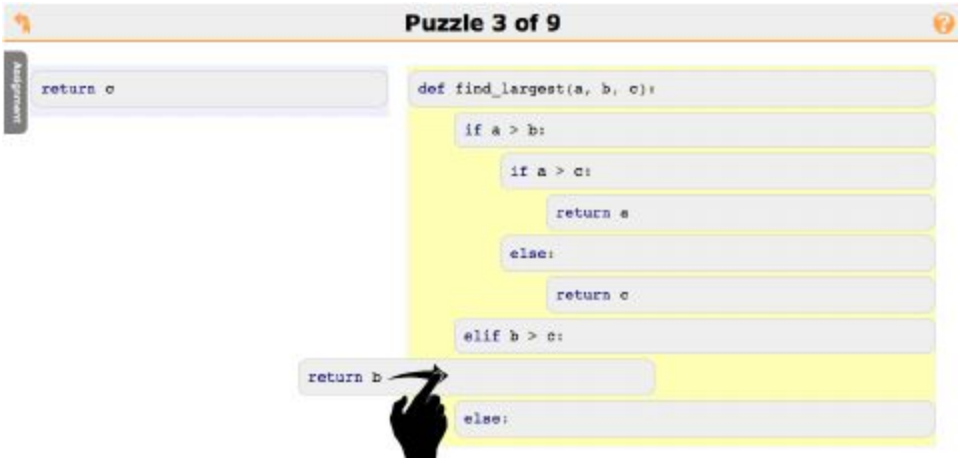
Table of Contents

Abstract

In advancement of SAGE's mission of improving computational thinking amongst 6-8th grade students, the Parson's Puzzle System serves as a key component in facilitating many core computational thinking concepts, such as code sequencing, looping, data storage, etc., but in order for this system to work optimally it requires a library of interactive games based on those concepts. The current Parson's Puzzle Library is incomplete and requires extensive development. For this semester, I plan on primarily focusing on developing various puzzles for the Data and Loop concepts, and, if time permits, for a few of the other concepts (Sequence, Events, Parallelism, Conditionals, and Operators).

Introduction

Parson's programming puzzles are a family of code construction assignments where lines of code are given, and the task is to form the solution by sorting and possibly selecting the correct code lines [1].



The objective for this semester will be to work on the Gameful Direct Instruction Epic on the SAGE backlog, with a special focus on the Parson's Puzzle Library.

The current library consists of the following computational thinking concepts: Sequences, Loops, Events, Parallelism, Conditionals, Operators, and Data. The goal for each concept is to develop several Parson's Programming Puzzles that instructors will be able to use to properly teach that concept. The general approach for this semester will be to focus on the data and looping concepts by developing several puzzles for each concept. After development, I hope to conduct some field work in hopes of getting feedback from current teachers and instructors that might be using this system. This will help in fine-tuning each puzzle and ensuring that the proper approach to concept conveyance is taken. Time permitting, I might create puzzles for as many of the remaining concepts as possible.

Assuming these puzzles are validated through fieldwork, it would then be imperative to collaborate with other members of the Parson's Puzzle team to ensure that automatic assessments based on student progress are given a little bit more attention. This is very important because many instructors do not have the time or resources to provide substantive feedback to each individual student in their class, so ensuring that each puzzle has a set of assessment tools at its disposal is very important [1].

Related Work

Two-Dimensional Parson's Puzzles: The Concept, Tools, and First Observations
Petri Ihantola and Ville Karavirta

Ihantola and Karavirta survey the current landscape of approaches towards Parson's Puzzle construction then present a new method for Parson's Puzzle inspired by the Python programming language know as a Two-Dimensional Parson's Puzzle. Essentially this new method uses the vertical ordering of lines to denote ordering as in traditional Parson's puzzles; whereas, the horizontal dimension is used to define code blocks based on indentation, as in Python. Lastly, they discuss proper methods of designing puzzles, of which the most important design principles are avoiding ambiguity and utilizing distractors.

Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses
Dale Parsons and Patricia Haden

Parsons and Haden describe the motivations for Parson's Programming Puzzles. They touch on best practices for design and state that since each puzzle solution is a complete sample of well-written code, use of their tool exposes students to good programming practice. They also describe a web-based authoring tool used to build the puzzles. In their introduction, they describe two main problems associated with drill-based exercise. For one, they are boring and lack engagement. For two, it is difficult to remove a single syntactic unit from the logical context in which it occurs. To address these issues their proposed tool attempts to maximize engagement, constrain the logic by providing good code structure, permit common errors allowing students to make direct comparisons between what they commonly do wrong with what the correct syntactic construct looks like, model good code, and provide immediate feedback. The puzzle described is in a drag-and-drop style in which a student is given a selection of code fragments and the student is tasked with dragging choices into the indicated answer locations. Lastly, they discuss built-in distractors that are meant to provide nuance and subtleties in each puzzle for a stronger understanding.

A Mobile Learning Application for Parsons Problems with Automatic Feedback
Karavirta et al.

Karavirta et al. present a mobile application tool called MobileParsons that facilitates the learning of programming for Parson's programming puzzles. Their argument for this approach is that Parson's problems are attempted in small chunks and as a result are suited to a mobile application. They present issues with automatic feedback associated with *trial-and-error* behavior in the context of frequent feedback requests. Strategies that are currently applied in attempting to solve these issues include limiting the number of submissions allowed and/or limiting the details of feedback. Karavirta et al. propose feedback for partial

solutions, such that a student is given continuous feedback for every possible state he or she gets in. To do this, they implement a feedback algorithm know as the longest common subsequences (LCS). This algorithm searches for the longest common subsequence shared by the model and the student solution that has the greatest number of consecutive code fragments in the student's solution. Another implementation Karavirta et al. devised that is intended to improve automatic feedback for *trial-and-error* behavior was to implement a functionality that recognized aimless exploration resulting in loops (revisiting previous states of the solution along the solution path). Lastly, they propose a solution to gaming the system for constant feedback by instilling penalties for the frequency with which students request feedback; thus, encouraging exploration as opposed to hacking.

How Do Students Solve Parsons Programming Problems? -- Execution-based vs. lione based feedback
Helminen et al.

Helminen et al. discuss a field study in which a group of students was split in half in order to determine the most efficient means by which students learn. Specifically, the paper presents a study on how the type of automatic feedback in Parsons problems affects how students solve them. It was discovered that the type of feedback had an effect on how students constructed their programs and how quickly they were able to complete them. With feedback based on execution as opposed to the visible arrangement of code, the programs were more frequently executable when feedback was requested and, overall, feedback was requested less frequently. The two primary contributions to the existing knowledge of automatic assessment and feedback on Parsons programming problems presented in this paper include an extension of js-parsons with execution-based feedback where tests are run on learner's code and then checked fr expected results, and the second is a comparison between this model and the existing line-based feedback.

Proposal

The current SAGE framework for the Parson's Puzzle does not have any actual puzzles in the puzzle library. As it stands, there are currently seven concepts whose puzzle needs substantiation. For this semester I hope to maintain the following guidelines in developing and field-testing puzzles for the Data and Loop computational thinking concepts. The process for accomplishing this goal will resemble something like the following: Learning Objective Development, Puzzle Corpus Development, Coding & Testing, Fieldwork, Refinement Procedures.

Learning Objective Development: During this process, I will be looking at the key components of the concepts of interest. This is where I will determine all the necessary ideas that the final puzzles will have to touch on. It will also be the time during which I will research ways in which typical Parson's Puzzles are shuffled and presented to students. Additionally, this is where design principles will come into play as I try to limit ambiguity in the examples I come up with and find neat/nuanced distractors. Additionally, I hope to use this phase to come up with a road map as to how I want the questions in each puzzle to progress in terms of difficulty.

Puzzle Corpus Development: During this phase is when I will actually be creating coding examples and testing them on my computer to make sure that they work. I intend to at least five puzzles each of three difficulty levels: Easy, Medium, Hard. Once these puzzles are created for the Data and Loop concepts, I hope to then disseminate them amongst current SAGE researchers to receive input on further improvement.

Coding & Testing: During this step is where I will be involved in creating the actual Parson's Puzzle framework. As it stands, the two-dimensional Parson's Programming Puzzle seems to be the most intriguing method.

Fieldwork: Once a few of the puzzles are created and working I hope to acquire feedback from those that will be using it (i.e. teachers and students in grades 6-8). The hope is that this is an ongoing process that blends into the Coding & Testing phase.

Refinement Procedures: Based on Fieldwork feedback, the codebase will be refined.

Expected Timeline

Week 1 - 2: **Learning Objective Development**
Week 3: **Puzzle Corpus Development**
Week 4 - End of Semester: **Coding & Testing**
Week 6 - End of Semester: **Fieldwork & Refinement Procedures**

Future Work

As I am primarily focusing on the Data and Loop concepts for this semester, I most likely won't have time for all the other concepts (Sequence, Events, Parallelism, Conditionals, and Operators). As a result, these will most likely need to be developed in future semesters. I'm hoping that the work I do this semester will be a blueprint for future SAGE researchers in creating new and nuanced puzzles. An additional area of research includes UI design. The interphase that is to be developed this semester may be very bland and rudimentary, so individuals with UI experience might want to work on enhancing this feature in the future. Lastly, more research may need to be conducted on how best to utilize distractors in each puzzle as well as different architectures other than the two-dimensional, python influenced implementation used this semester.

References

1. Petri Ihantola and Ville Karavirta. *Two-Dimensional Parson's Puzzles: The Concept, Tools, and First Observations*
2. Dale Parsons and Patricia Haden. *Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses*
3. Karavirta et al. *A Mobile Learning Application for Parsons Problems with Automatic Feedback*
4. Helminen et al. *How Do Students Solve Parsons Programming Problems? -- Execution-based vs. lione based feedback*