**Abstract**

Parsons problem has been used as a convenient tool in introductory programming courses to equip students for writing codes. However, intermediate students may also have problems in directly writing codes when taking some higher-level courses, therefore, Parsons problem is also needed in these teaching activities.

Parsons problem is a convenient tool used in introductory programming courses since it can reduce students’ cognitive load when writing codes directly. But in Data Structures and Algorithm, the traditional Parsons problem is not so applicable because of the change in course focus.

This project attempts to build an application of Parsons problem in website form for assisting students in learning Data Structures and Algorithms. This project expands on traditional Parsons problem by introducing more flexible question types including dividing into multiple steps, algorithm comparison and etc. Additionally, the project applies more ways to handle difficulty levels.

The results of this project show that it can provide students with help in familiarizing themselves with codes involved in this course. And experiment will be done in the future to verify the effectiveness.

Parsons problem has been used as a convenient tool in introductory programming courses to equip students for writing codes. However, intermediate students may also have problems in directly writing codes when taking some higher-level courses, therefore, Parsons problem is also needed in these teaching activities.

This project attempts to build an application of Parsons problem in website form for assisting students in learning one of higher-level programming courses – Data Structures and Algorithms. Based on the analysis of concrete problems in this course, more flexible question types are expanded in this project, including splitting into multiple steps, algorithm comparison and algorithm analysis. In addition, more ways to handle difficulty level is applied, and switching difficulty level is allowed.

The result shows that this project can provide some help for students in familiarize themselves with basic codes of data structures and algorithms, algorithm time complexity analysis, comparing similar codes.

* Start with a clear and concise statement of the problem you are trying to solve.
* Explain the motivation behind your project and why it is important.
* Describe the methods you used to solve the problem.
* Highlight the key results and findings of your project.
* Conclude with a summary of the implications of your work and future directions.

This project aims to develop a website application of Parsons problem to assist students in learning higher-level programming courses such as Data Structures and Algorithms. The project expands on the traditional Parsons problem by introducing more flexible question types, including splitting into multiple steps, algorithm comparison and algorithm analysis. Additionally, the project applies more ways to handle difficulty levels and allows switching between difficulty levels. The project is based on the analysis of concrete problems in the Data Structures and Algorithms course.

The results of the project show that it can provide students with help in familiarizing themselves with basic codes of data structures and algorithms, algorithm time complexity analysis, and comparing similar codes. The project is motivated by the need to equip intermediate students with the necessary skills to write codes when taking higher-level courses. The project has implications for the development of more effective teaching tools for programming courses.

Parsons problem is a type of programming question that allows students to drag and drop prepared blocks of code to build unique predefined solutions. This type of question has been popularly applied in introductory programming courses (CS1). However, the application scope of Parsons problem has never been expanded to higher-level courses such as Data Structures and Algorithms (CS2). Although students taking this course should be able to write some codes, the abstractness and complexity of the course make it challenging for students to write codes directly. Therefore, it is essential to introduce Parsons problem as a preparation stage for students to equip themselves for writing codes in the future.

This project aims to build a website to apply Parsons problem in CS2. The website will implement the common functions of traditional Parsons problem, including inputting problems and solutions in Python, generating Parsons problems, solving Parsons problems, and giving feedback. Additionally, the website will make some changes to adapt to the specific exercises in CS2, including introducing more different types of questions and more ways to handle difficulty levels.

This project explored the possible application of Parsons problem in Data Structures and Algorithms. The new forms of Parsons problem that were developed in this project can support some situations not involved in introductory programming, including different property codes, the coexistence of lines and blocks, and the existence of multiple solutions. These new forms can also handle the difficulty level in different ways.

The project’s main contribution is to provide a new technical tool that can help students learn data structures and algorithms more effectively. The tool can not only improve students’ engagement and reduce their cognitive load, as the traditional Parsons problem does, but also help students consolidate knowledge by comparing similar concepts.

In the future, experiments will be conducted to compare the effectiveness of this tool with traditional learning methods. Additionally, questionnaires will be sent to students and lecturers to collect their opinions on improving the system.

**Introduction**

Parsons problem is a type of programming question to let students drag and drop to reorder the mixing up prepared blocks of codes to build the unique predefined solutions. This type of question has been applied in introductory programming courses (CS1) popularly. However, the application scope of Parsons problem has never been expanded to higher-level course such as Data Structures and Algorithms (CS2). Since students taking this course should be able to write some codes, the application of Parsons problem in CS2 should focus less on specific code writing and more on high level issues. Thus it is of the essence to expand traditional Parsons problem and make it more applicable in CS2.

This project aims to build a website to apply Parsons problem in CS2. In this project, not only the common functions of traditional Parsons problem should be implemented (including inputting problems and solutions in Python, generating Parsons problems, solving Parsons problems and giving feedbacks), but also some changes should be made to adapt the specific exercise in CS2 (more different types of questions and more ways to handle difficulty levels).

**Problem Analysis**

The traditional Parsons problem has shown its advantages in exercises of CS1. However, since there are plenty of differences between the exercise in CS1 and CS2, it is not so proper to apply Parsons problem directly in CS2 without tailored improvement. To identify the limitation of traditional Parsons problem and illustrate the new ideas, four kinds of concrete questions from exercises in CS2 are analyzed.

**Object-oriented programming:** it is a programming model based on objects, which includes attributes and methods. This model is not involved in CS1, but it is the foundation to implement different data structures. As a consequence of involving this programming model, the uniqueness of correct answer is broken since the methods in the classes can changing their positions without affecting the correctness of the results. To handle this problem, the project should only check the availability of methods and the contents of methods but ignore the positions of the methods.

**Algorithm analysis:** as a core part of CS2, it is to use the running time to evaluate whether a data structure or an algorithm is efficient or not. This concept is first introduced in CS2, which means that it is never mentioned in CS1. Non-linear data structures and recursion make it not straightforward to observe the results directly. To help students to learn algorithm analysis, comments is introduced as options for choosing proper big O classes to describe codes.

**Recursion:** it is solving a problem by solving a subproblem with the same structures as the original problem. Unlike in CS1, the recursion method requires students with deeper understanding in the whole picture of entire methods besides concrete lines in the methods. For this reason, the traditional Parsons problem cannot reduce the difficulty as usual through providing code reading. To give students some ideas, a recursion question can be divided into several steps to let students build from base case and subproblem to the original recursion codes.

**Comparison:** there are some similar codes in CS2, for example the same data structure with different implementation or different algorithm for solving the same problems. Although it is ok to use the traditional Parsons problem individually for each one, it would be more worthwhile to have ways to help students compare similar codes and consolidate the difference and similarities between these codes. There are two types of questions to have comparison – combine codes from two pools into one pool and split two solutions from one code pool.

**Problem Analysis**

There are plenty of differences between the exercise in CS1 and CS2, it is not so proper to apply Parsons problem directly in CS2 without tailored improvement. To illustrate the difference in detail, four kinds of concrete questions from exercises in CS2 are analyzed.

**Object-oriented programming:** this model is not involved in CS1, but it is the foundation to implement different data structures. Because of involving this mode, the uniqueness of correct answer is broken since the order of methods in a class can be changed without affecting correctness. To handle this problem, the project should only check the availability and the contents of methods but ignore the positions of the methods.

**Algorithm analysis:** this concept is first introduced in CS2 not mentioned in CS1. To help students to learn algorithm analysis, comments is introduced as options for choosing proper big O classes to describe codes.

**Recursion:** the recursion method requires students with deeper understanding in the whole picture of entire methods besides concrete lines like CS1. For this reason, the traditional Parsons problem cannot reduce the difficulty as usual through providing code reading. To give students some ideas, a recursion question can be divided into several steps to let students build from subproblem to the original recursion codes.

**Comparison:** there are some similar codes in CS2, for example the same data structure with different implementation or different algorithm for solving the same problems. Although it is ok to use the traditional Parsons problem individually for each one, it would be more worthwhile to have ways to help students compare similar codes and consolidate the difference and similarities between these codes. There are two types of questions to have comparison – combine codes from two pools into one pool and split two solutions from one code pool.

**Balance the degree of difficulty**

One of the key issues of Parsons problem is to find a balance point in the degree of difficulty, which is more accessible than writing codes from scratch but still valuable to practice. To fulfill this demand, three different methods (selecting difficulty level, pre-scaffold, and context) are introduced in the following paragraph.

**Switch Difficulty Levels:** due to the increase in the complexity of the codes in CS2, it is necessary to provide further help by giving the same questions with more hints.

**Pre-scaffold:** it is to give students some hints by giving structures, steps or subgoals.

**Context:** it is to give some ordered codes as information and let students reorder parts of original codes.

**Holistic Design**

To sum up all descriptions of Parsons problem mentioned before, Parsons problems in this project can be declared as following question types:

**Traditional:** themost basicParsonsproblem – only support rearranging jumbled predefined codes and getting feedback about the reordered answers. This type of Parsons problem is used in CS1.

**Context:** some ordered codes are provided as hints. This type of question can be used to handle the difficulty.

**Multiple Step:** this typeof Parsons problem divides the original codes into several steps to let students solve complex problems by solving each small part. It can be used in recursion problem or be used as pre-scaffold

**Algorithm Comparison:** this type of question is used to compare similar algorithms. In this type of question, students split two solutions from one group of jumbled codes.

**Algorithm Analysis:** this type of question can introduce the time complexity of algorithm analysis by using big O class comments.

The additional characteristics of Parsons problem in this project can be summarized from the following perspectives – allowing some special-marked code fragments (context or comment), allowing the coexistence of lines and blocks (groups of multiple lines) and allowing multiple solutions. Based on these characteristics, Parsons problem in this project should also provide the function to customize solutions (like marking special code fragments or grouping fragments as a block). Besides, this project should allow checking in different situations including switching the orders of blocks or comment existing.

**Holistic Design**

**Question type summarization:**

* **Traditional:** themost basicParsonsproblem – only support rearranging jumbled predefined codes and getting feedback about the reordered answers. This type of Parsons problem is used in CS1.
* **Context:** some ordered codes are provided as hints. This type of question can be used to handle the difficulty.
* **Multiple Step:** this typeof Parsons problem divides the original codes into several steps to let students solve complex problems by solving each small part. It can be used in recursion problem or be used as pre-scaffold
* **Algorithm Comparison:** this type of question is used to compare similar algorithms. In this type of question, students split two solutions from one group of jumbled codes.
* **Algorithm Analysis:** this type of question can introduce the time complexity of algorithm analysis by using big O class comments.

**Result**

Figure 1 shows the question page of traditional Parsons problem. Students need to drag the jumbled codes from left code pool, drop the codes in the right code pool in order, and push arrow buttons to set proper indent of codes. After finishing, the codes are marked as different background colors to give students some feedbacks.

Figure 2 shows the question page of Parsons problem with context. Compared to the traditional Parsons problem, this form provides partially ordered code and placeholders in the initialization phase. This format allows students to get more hints by reading the code in context, thus reducing the difficulty of the problem.

Figure 3 shows the question page of Multiple Step Parsons problem. Compared to the traditional Parsons problem, this form supports breaking a complex problem into many smaller problems. It gives students hints by labeling the number of steps of different codes.

Figure 4 shows the question page of Algorithm Comparison. Compared to the traditional Parsons problem, students need to split two algorithms mixed in left code pool to middle and right code pool separately. This form is worthwhile for students to distinguish similar algorithms and prevent students from using them mostly because of blurry memory.

It is worthwhile when students have learned more than one algorithm and begin to use them mostly because of blurry memory

In conclusion, this project explored the possible application of Parsons problem in Data Structures and Algorithms. In this project, new forms of Parsons problem are explored to fit the concrete exercise in Data Structures and Algorithms. These new forms support some situations not involved in introductory programming including different property codes, the coexistence of lines and blocks and the existence of multiple solutions. In addition, these new forms can also handle the difficulty level in different ways. In the future, experiments on the comparison of old learning ways with new technical tool will be done to verify the effectiveness of helping students to learn data structures and algorithms.

In conclusion, this project explored the possible application of Parsons problem in Data Structures and Algorithms. The new forms of Parsons problem that were developed in this project can support some situations not involved in introductory programming, including different property codes, the coexistence of lines and blocks, and the existence of multiple solutions. These new forms can also handle the difficulty level in different ways.

The project’s main contribution is to provide a new technical tool that can help students learn data structures and algorithms more effectively. It can not only improve students’ engagement and reducing students’ cognitive load as the traditional Parsons problem, but also can help students to consolidate knowledge though comparing similar concepts.

In the future, experiments will be conducted to compare the effectiveness of this tool with traditional learning methods. In addition, questionnaires will be sent to students and lecturers to collect their opinions on improving system.

The variation of Parsons problem can not only take the advantages of the existing Parsons problem (like improving students’ engagement and reducing students’ cognitive load [2]), but also have some new benefits. To be more specific, the new design can handle the specific concepts only existing in CS2. In addition, it can help students to build excellent habits to solve problems. Last but not least, it can help students to consolidate what they have learned through many different ways.