Vol.2019-CE-151 No.1
2019/10/5

学習進度に対応するパズルを利用した プログラミング思考過程の分析

中村陽太[†1]　大場みち子[†1]　山口琢[†2]　伊藤恵[†1]

概要：初等教育でのプログラミング必修化に伴い，プログラミング教育への関心が高まっている．最近では理系大学のみならず，文系大学でもプログラミング教育が実施されている．プログラミング教育における学習者への評価方法はテスト結果などのアウトプットでの評価が主流であり，プログラミング時の過程や思考などは評価していない．プログラミング時の思考過程がわかれば，学習者の考え方の傾向や躓きの要因を把握することができ，適切な指導ができるであろう．先行研究ではパズルを利用してプログラミング思考過程を分析しているが，問題の出題意図が不明確であった．本研究では，授業の学習進度に対応するパズルを利用してその操作過程を記録することで，プログラミング時の思考過程の傾向について分析することを目指す．開発したツールと授業の学習進度に対応するパズルを利用した実験によりプログラミング思考過程を分析した結果を報告する．

キーワード：プログラミング，思考過程，分析，パズル，操作過程の記録

Analysis of Programming Thinking Process Using Puzzles Corresponding to Learning Progress

YOTA NAKAMURA[†1]　MICHIKO OBA[†1]　TAKU YAMAGUCHI[†2]　KEI ITO[†1]

Abstract: With the obligatory programming in elementary education, interest in programming education is increasing. Recently, programming education is being conducted not only at science universities but also at liberal art universities. Evaluation methods for learners in programming education are mainly evaluations based on outputs such as test results, and thoughts processes during programming are not evaluated. If thinking processes during programming are known, it will be possible to understand tendency of learners' thinking and factors of tumbling and will be able to give appropriate guidance. Previous research used puzzles to analyze programming thinking processes, but the intent of the questions was not clear. In this paper, we aim to analyze the tendency of thinking process during programming by recording the operation process using the puzzle corresponding to lesson progress. We report the result of analyzing programming thinking process by experiment using the developed tools and puzzles corresponding to learning progress of the class.

Keywords: Programming, Thinking Process, Analysis, Puzzle, Recording of Operation Process

1. はじめに

初等教育でのプログラミング必修化に伴い，思考力と共にプログラミング教育への関心が高まっている[1][2]. プログラミング教育における学習者への評価方法はテスト結果などのアウトプットでの評価が主流であり，プログラミング時の過程や思考などは評価していない．プログラミング時の思考過程がわかれば，学習者の考え方の傾向や躓きの要因を把握でき，適切な指導ができるであろう．

われわれはパズルに仕立てたプログラミング課題を使って，パズル操作の過程を記録して分析することでプログラミング時の思考/問題解決過程の傾向を検出することを目指している．このアイデアについて発表して議論を深めてきた[3][4][5][15][16]. 先行研究[15][16]ではパズルを利用してプログラミング思考過程を分析しているが，問題の出題意図が不明確であった．本研究では，授業の学習進度に対応するパズルを利用してその操作過程を記録することで，プログラミング時の思考過程の傾向について分析することを目指す．開発したツールと授業の学習進度に対応するパズルを利用した実験によりプログラミング思考過程を分析した結果を報告する．

2. 関連研究

英語のテスト問題では，単語を取捨選択・並べ替えて文章を完成させるものが昔からあり,現代ではコンピュータ・アプリ版もある．このプログラミング版というべきものにParson's Problem がある[8]．また「短冊コード」は，これよりも実際のコーディングに近い．言わば，限定されたコード断片クラスから実際のコード・インスタンスを生成・削除しながらコード全体を完成させる[6].

†1 公立はこだて未来大学
Future University Hakodate
†2 フリー
Independent Researcher

これらの過程を記録・分析する研究も進んでいる．Parson's Problem を解く様子をビデオ録画して人手で分析した研究では，初心者(novices)と熟達者(experts)を対比して問題を解く戦略(problem-solving strategies)を見出した．初心者はインデントを手がかりにしたり，試行錯誤したりするのが見られる．これに対して，熟達者は「解の型(model of solution)」を持っていて，そこへ向けてトップダウンに解く．また，両者に共通に見られるのは，変数の宣言は関数の最初にあるといった構文を手がかりにする戦略であった[9]．

Parson's Problem では，最短手数で正解にたどり着くベスト手順と比較することで熟達度を数値化する研究もある[10]．

このように，パズル操作の過程を測定・分析する研究は，パズル・アプリ，パズル問題，操作の記録・測定，測定データの分析の 4 点それぞれが工夫・研究の対象となる．

また，Parson's Problem に関する上記研究に共通するのは，試行錯誤を初心者の戦略として低く評価[9]したり，データ分析で試行錯誤を無視[10]したりすることである．Parson's Problem に関する上記研究と「短冊コード」との違いは，前者が「まずパズルがあって，これを解く最善の戦略を問題とする」のに対し，後者[10]は「まず思考力の育成・評価を目的とし，それを測る装置としてパズルがあり，パズル問題の設計とセットで取り組む」点であろう．

3. これまでの取り組み

3.1 着想

ゲームやパズルでは，プレイヤーがゲームやパズルを操作する様子を観察すると，そのときのプレイヤーの思考過程を推定できることがある．われわれは，これを作文やプログラミングなどに適用しパズルを適切に設計して，パズルを解く操作を測定・分析することで，プレイヤー，すなわちライターやプログラマーの思考を推定する研究・指導・学習手法を開発している．テクニカル・ライティングの授業や就活指導では，このコンセプトをロジック・ツリー作文に応用して書く手順を測定・分析している[11][12]．また，関連研究と同様に，ランダムに並んだピースをプレイヤーが適切と考える順序に並べ替えて完成させるアプリを，読解・作文，プログラミング，年表，計画，地理に応用している[3]．われわれは，これらをジグソー・パズルになぞらえて，ジグソー・テキストなどと呼んでいる．

3.2 プログラミング・パズル「ジグソー・コード」

ジグソー・コードはランダムに並んだプログラムの行を，プレイヤーが適切と考える順序に並べ替える，プログラム・コードのジグソー・パズルである[13]．行がパズルのピースとなる．Web アプリケーションとして実装されている．

図 1 はパズル問題（JavaScript のプログラム）の例である．s1 等はピースである行の ID で，測定・分析や本稿での説明に用いる．パズルを開始すると，これらピースがランダムに並べ替えられてプレイヤーに提示される．プレイヤーはドラッグ＆ドロップで並べ替えて，適切と考える順序になったところで完成ボタンを押す．

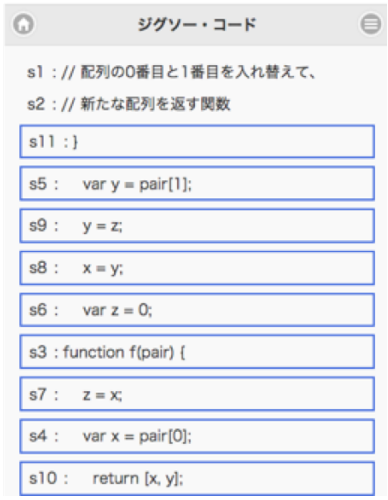

図 1　ジグソー・コードのパズル

ジグソー・コードではプレイヤーの操作を記録している開始，ドラッグ，ドロップ，完成のイベントを時刻と，そのときの全体の順序とともに記録する．さらに，ドラッグとドロップについては，対象のピースと，その前後のピースの ID も併せて記録する(図 2)．

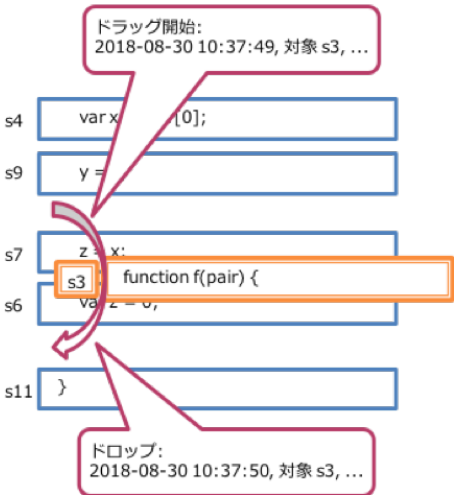

図2　並べ替え操作の測定

3.3 プログラミング・パズル「ジグソー・コード 2」

ジグソー・コード 2 は，ジグソーコードと同様にランダムに並んだプログラムの行を，プレイヤーが適切と考える順序で別のエリアに並べ替える，プログラム・コードのジグソー・パズルである．行がパズルのピースとなる．ジグソー・コードと同様に Web アプリケーションとして実装されている．

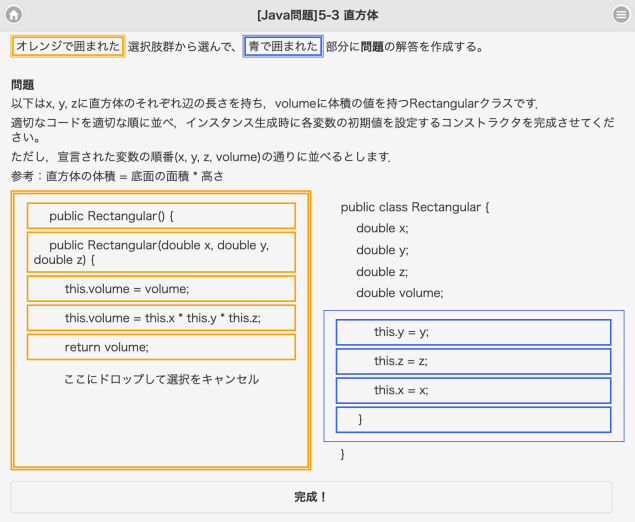

図 3 ジグソー・コード 2 のパズル

図 3 はパズル問題（Java のプログラム）の画面例である．パズルを開始すると，左側に不正解のピースを含む並べ替え対象行，右側に問題の解答欄が表示される．これらピースがそれぞれランダムに並べ替えられてプレイヤーに提示される．プレイヤーは左側のエリアから正解の行をドラッグ＆ドロップで右側のエリアに移動し，右側のエリアにあるピースを並べ替えて，適切と考える順序になったところで完成ボタンを押す．

3.4 思考パターンの検出手法

操作の時間的な経過を分析して操作のパターンを検出するには，分析手法を工夫する必要がある．例えば，時間的に近くで操作対象となるピース間には何かの関係があると考えられ，操作の時間的な共起分析が，パズル操作の分析に有効と考えられる．

n \ n+1	s3	s4	s5	s6	s7	s8	s9	s10	s11
s3	0	3	3	1	2	1	0	1	14
s4	0	3	13	4	1	0	1	1	0
s5	0	3	2	11	0	0	1	7	1
s6	2	1	1	5	11	4	3	1	0
s7	0	0	0	2	2	9	4	0	0
s8	1	0	0	1	1	1	2	1	2
s9	0	1	0	2	6	1	4	0	1
s10	1	6	6	3	1	1	2	4	0
s11	3	6	0	0	0	0	0	10	7

図4　ピース操作の時間的な共起行列

図 4 は図 1 についてドラッグ対象となったピースの時間的な共起行列，すなわち，時間的に連続して操作された回数の頻度行列の，想定例である[3]．縦軸のピースの次に横軸のピースを動かした回数を，その演習全体について集計して表示しており，例えば s3 の次に s11 を動かした回数は演習全体で 14 回であったと読む．この集計では s3→s11 という手順が試行錯誤によって何度現れても，その都度カウントする．この共起行列は，s3→s11，s4→s5，s5→s6，s6→s7，s7→s8 と，それぞれ続けて動かすことが多いことを示している．このような場合，例えば「s3 に対する操作と s11 に対する操作が時間的に共起している」と呼んでいる．

時間的に近くで操作対象となるピース間には何かの関係があると考えられる．図 4 が示す共起関係はパズル問題図 1 について，プレイヤーたちが読み取ったプログラム構造を反映していると考えられる．ランダムに並べられたピース群から，特定のピース間の関係を読み取り(構造理解)，それを反映した順序に並べ替えよう(プログラミングしよう)としたと想定できる．以下，ピースを行と呼ぶ．

ジグソー・コードでは分析ツールとして共起行列の分析の他に最初に操作した行，行の操作回数，完成順序，時系列の操作などの表示機能を用意している．

3.5 従来の実験と課題

先行研究における実験と分析[15][16]では，ジグソー・コードの問題 5 問を実験用に作問し，出題した．しかし，先行研究で出題された実験問題では出題意図が不明確であり，回答者がどの程度基礎知識を理解していたか把握することができないという課題があった．

4. アプローチ

先行研究の課題を解決するため，プログラミング授業の進捗に合わせて作問し，授業範囲の復習として出題する．プログラミング授業の進捗に合わせた作問をすることで，理解が不十分な項目を把握して復習に活かすことができる．ジグソー・コードおよびジグソー・コード 2 を使って実験し，操作を記録・測定し，操作ログと正解/不正解などから思考過程を分析する．

プレイヤーが，あるパズルを解くためにある考え方をすることと，そのパズルで正解/完成することが，関連するとは限らない．それはパズルの内容や難易度や，プレイヤー全体のスキルなどによるだろう．それでも，パズル操作のパターンとパズルの正解/不正解との差異を分析することは，思考パターンを検出する有効な手法の一つではあるであろうと考えた．

5. 実験

5.1 実験の目的

実験の目的は授業の進捗に合わせたジグソー・コード問題を各授業後の復習として回答させ，その操作ログを分析して，プログラミング思考過程を検出することである．

5.2 実験対象者

実験対象者は，公立はこだて未来大学の学部 2 年生対象授業「情報処理演習 I」を履修している学生のうち，情報提供の同意が得られた 48 名（情報システムコース 8 名，情報デザインコース 8 名，知能システムコース 32 名）である．

「情報処理演習 I」は Java 言語を題材とした課題を通して，ソフトウェア開発プロセスにおける基本技能を習得す

ることを目的とした授業である[14].

5.3 実験手順

(1) 概要説明: 初回の実験前に，実験の概要やジグソー・コードの操作方法を説明する．

(2) 実験問題の回答: 用意した実験問題 3 問を，各回の授業後に回答する．

(3) 解答解説の提示: 回答後に実験問題 3 問に対応した解答解説を提示する．

5.4 ジグソー・コードの問題

実験は全 9 回を 8 週に分けて，授業の復習として実施した．実験回ごとの単元概要を表 1 に示す．実験問題は全て Java のプログラムコードとし，「情報処理演習 I」の各授業回に対応した範囲の問題を作問し，出題した．作問時には，並べ替えの対象行を 5 行以上とした．これは少ない行をランダムに並べ替えた際に，正解の並びになる確率を下げるためである．作問ではインデントによる区別やコメントによる順番指定，同一処理の排除などにより順不同の回答をできるだけ減らす工夫をした．これらは正解のパターンが多くなると分析ツールがそのままでは使えないからである．復習として理解に役立てるために，出題内容についての解答解説を作成した．全 27 問のうち，全体の操作の傾向を含む代表的な問題例として以下に 5 問を挙げ，各問の狙いや工夫点を説明する．

表 1 実験回ごとの単元概要

	単元概要
第 1 回	if 文, switch 文
第 2 回	配列，拡張 for 文，メソッド
第 3 回	オブジェクト指向， getter メソッド, setter メソッド
第 4 回	オブジェクト指向， メソッドの引数と返り値
第 5 回	オブジェクト指向， メソッドの引数と返り値
第 6 回	オブジェクト指向， インスタンスのメソッド呼び出し
第 7 回	オブジェクト指向， ArrayList，引数の指定方法
第 8 回	オブジェクト指向，クラスの継承
第 9 回	HashMap，外部ファイルの読み書き

各問題は[Java 問題]n-m “問題名”と表記し，第 n 回の m 番目の問題であり，“問題名”は問題概要を示している．

(1) [Java 問題]1-2 成績判定 1

```
s1  // 評点(score)に基づいて評定(rating)を出力するプログラムを完成させてください.
s2  // 90点以上でS，80点以上90点未満でA，70点以上80点未満でB，60点以上70点未満でC，60点未満でFの順に判定します.
s3  // ただし，評点の初期値は85とし，最高で100とします.
s4  public class Main {
s5      public static void main(String[] args) throws Exception {
s6          int score = 85;
s7          System.out.println("評定: " + judgeRating(score));
s8      }
s9      public static char judgeRating(int score) {
s10         char rating;
s11         if (90 <= score) {
s12             rating = 'S';
s13         } else if (80 <= score && score < 90) {
s14             rating = 'A';
s15         } else if (70 <= score && score < 80) {
s16             rating = 'B';
s17         } else if (60 <= score && score < 70) {
s18             rating = 'C';
s19         } else {
s20             rating = 'F'
s21         }
s22         return rating;
s23     }
s24 }
s25 // コンソールの出力
s26 // > 評定: A
```

本問題には正解が複数通りある．s21 と s23 はどちらも"}"なので，ピースの順序としては s21 と s23 が入れ替わっても良い．

本問題を出題した週の対応範囲では，変数の宣言や代入，if 文などの基本的な構文，関数の構造を扱っていた．そこで本問題では，条件式に適切な処理を対応させる順番を見ることと，return 文を適切に扱えるかを見ることを狙いとした．

(2) [Java 問題]4-1 銀行口座と 2 つの通帳

```
s1  // 通帳2つでお金の預け入れと引き出しを行い，口座残高を表示するプログラムを完成させてください.
s2  // 1つ目のクラス(Main)では通帳2つのインスタンスを生成し，預け入れと引き出し，口座残高の表示を行います.
s3  // 2つ目のクラス(Bankbook)は通帳名(name)と口座残高(money)，口座残高のゲッター，
s4  // お金の預け入れの関数(deposit)と引き出しの関数(withdraw)を持ちます.
s5  // ただし，Bankbookクラスの関数の中ではコンストラクタが一番上の順番とし，他の関数の順番は関数の上のコメントに従うものとします.
s6  public class Main {
s7      public static void main(String[] args) throws Exception {
s8          Bankbook bankbook1 = new Bankbook("通帳1");
s9          Bankbook bankbook2 = new Bankbook("通帳2");
s10         bankbook1.deposit(7_000);
s11         bankbook2.withdraw(12_000);
s12         System.out.println(bankbook1.name + ": 口座残高は，" + bankbook1.getMoney() + "円です");
s13         System.out.println(bankbook2.name + ": 口座残高は，" + bankbook2.getMoney() + "円です");
s14     }
s15 }
s16 public class Bankbook {
s17     String name;
        public static int money = 10_000;
s18     public Bankbook(String name) {
s19         this.name = name;
        }
s20     // 1つ目の関数
        public static int getMoney() {
s21         return money;
        }
s22     // 2つ目の関数
        public void deposit(int money) {
s23         this.money = this.money + money;
        }
s24     // 3つ目の関数
        public void withdraw(int money) {
s25         this.money = this.money - money;
        }
s26 }
s27 // コンソールの出力
s28 // > 通帳1: 口座残高は，5000円です
s29 // > 通帳2: 口座残高は，5000円です
```

本問題を出題した週の対応範囲では，オブジェクト指向を扱っていた．そこで本問題では，クラス内の変数，コンストラクタや関数を完成する順番を見ることを狙いとした．また，対応範囲に含まれている static の性質について理解できるような作問をした．

(3) [Java 問題]5-1 履修登録 1

```
// 入力した授業が存在するかを返すプログラムを完成させてください．
// 1つ目のクラス(Main)では授業のインスタンスを生成し，存在するかによってメッセージを表示します．
// 2つ目のクラス(Lecture)は授業の曜日(weekday)，時限(period)，授業名(name)を持ちます．
// 3つ目のクラス(Registration)は授業の配列(Lectures)と，引数で渡された科目が存在するかを判定する関数(isExistLecture)を持ちます．
public class Main {
    public static void main(String[] args) throws Exception {
        Registration regist = new Registration();
        if (regist.isExistLecture(new Lecture("火", 3, "プログラミング演習"))) {
            System.out.println("この科目は存在します");
        } else {
            System.out.println("この科目は存在しません");
        }
    }
}
public class Lecture {
    String weekday;
    int period;
    String name;
    public Lecture(String weekday, int period, String name) {
        this.weekday = weekday;
        this.period = period;
        this.name = name;
    }
}
public class Registration {
    Lecture Lectures = {new Lecture("火", 3, "プログラミング演習"), new Lecture("木", 1, "サービスデザイン")};
    public Registration() {
    }

    public Boolean isExistLecture(Lecture lec) {
        for (int i = 0; i < lectures.length; i++) {
            Lecture l = lectures[i];
            if (l.weekday.equals(lec.weekday) && l.period == lec.period && l.name.equals(lec.name)) return true;
        }
        return false;
    }
}
// コンソールの出力
// > この科目は存在します
```

本問題を出題した週の対応範囲では，前週に引き続きオブジェクト指向を扱っていた．そこで本問題では，他クラスのインスタンスを使った処理を正しく理解して条件分岐と return 文が配置できているかを見ることや，直前に宣言された変数との順番を見ながら配置できているかを見ることを狙いとした．

(4) [Java 問題]7-1 可変な値の格納

```
// ランダムな整数値を10回生成し，10より小さいものを全て出力するプログラムを完成させてください．
public class Main {
    static ArrayList numbers = new ArrayList();
    public static void main(String[] args) {
        Random random = new Random();
        for (int n = 0; n < 10; n++) {
            int i = random.nextInt(20);
            if (i < 10) numbers.add(new Integer(i));
        }
        for (Integer num: numbers) {
            System.out.print(num.intValue() + " ");
        }
    }
}
// コンソールの出力
// > 2 4 6 8
```

本問題を出題した週の対応範囲では，Integer などのアッパークラスや ArrayList を扱っていた．そこで本問題では，変数をアッパークラスのオブジェクトにし，for 文が複数ある場合に完成させる順序を見ることを狙いとした．

(5) [Java 問題]7-2 要素削除時の順番

```
// 連続した整数値が入ったArrayListから要素を削除し，コンソールの出力と同じ出力をするプログラムを完成させてください．
public class Main {
    static ArrayList numbers = new ArrayList();
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) numbers.add(new Integer(i));
        numbers.remove(7);
        numbers.remove(4);
        numbers.remove(1);
        for (Integer num: numbers) System.out.print(num.intValue() + " ");
    }
}
// コンソールの出力
// > 0 2 3 5 6 8 9
```

本問題を出題した週の対応範囲では，Integer などのアッパークラスや ArrayList を扱っていた．そこで本問題では，ArrayList で要素を削除したときの添字の変化を考慮できているかを見ることを狙いとした．

6. 実験結果

5 章の実験に対する結果を示す．

問題毎の実験結果を以下に示す．表 2 は各問題の正解者と不正解者の数である．

表 2 各問題の正解/不正解者数

問題	1-2	4-1	5-1	7-1	7-2
正解	43	18	11	3	4
不正解	5	22	23	8	7

[Java 問題]1-2,4-1,5-1,7-1,7-2 の各問題回答者全員の最初に動かした行，最後に動かした行，共起行列を図 5〜図 9 に示す．

ID	回数	割合	グラフ
s9	29	60%	############
s10	1	2%	
s11	7	14%	###
s12	2	4%	#
s13	1	2%	
s14	1	2%	
s15	0	0%	
s16	1	2%	
s17	0	0%	
s18	0	0%	
s19	0	0%	
s20	0	0%	
s21	2	4%	#
s22	1	2%	
s23	3	6%	#
合計	48		

ID	回数	割合	グラフ
s9	1	2%	
s10	12	25%	#####
s11	0	0%	
s12	1	2%	
s13	0	0%	
s14	0	0%	
s15	1	2%	
s16	1	2%	
s17	0	0%	
s18	0	0%	
s19	4	8%	##
s20	2	4%	#
s21	6	12%	##
s22	15	31%	######
s23	5	10%	##
合計	48		

n \ n+1	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
s9	6	11	18	2	0	0	1	0	1	0	1	0	1	2	7
s10	3	4	6	5	1	1	0	1	0	1	3	0	4	10	1
s11	1	2	9	23	4	6	1	0	2	0	1	1	1	0	1
s12	0	0	1	2	25	13	1	1	1	0	2	2	0	0	1
s13	1	1	1	2	5	28	3	2	4	0	0	0	0	0	1
s14	0	2	1	2	1	0	25	17	3	1	0	1	0	1	1
s15	0	2	0	1	3	0	2	23	6	2	0	1	0	0	0
s16	0	2	0	4	1	2	2	3	23	12	2	2	0	0	0
s17	2	2	0	3	2	1	3	1	1	19	7	1	1	1	1
s18	3	1	0	0	2	1	0	1	1	0	17	6	0	2	2
s19	0	5	0	0	2	0	0	0	1	0	0	18	6	2	2
s20	2	8	0	2	0	0	0	2	1	1	5	0	4	6	2
s21	2	1	1	2	1	0	2	0	0	0	0	0	2	3	5
s22	0	3	3	0	0	2	0	1	0	0	2	1	3	4	0
s23	2	7	5	0	0	0	1	1	1	0	0	2	1	2	3

図 5　**[Java 問題]1-2** の回答者全員の操作状況

ID	回数	割合	グラフ
s17	5	12%	##
s18	5	12%	##
s19	3	7%	#
s20	21	52%	##########
s21	2	5%	#
s22	3	7%	#
s23	0	0%	
s24	1	2%	
s25	0	0%	
合計	40		

ID	回数	割合	グラフ
s17	10	25%	#####
s18	1	2%	
s19	11	27%	#####
s20	0	0%	
s21	4	10%	##
s22	0	0%	
s23	6	15%	###
s24	5	12%	##
s25	3	7%	#
合計	40		

n \ n+1	s17	s18	s19	s20	s21	s22	s23	s24	s25
s17	7	9	3	1	7	8	8	0	3
s18	6	8	24	4	4	2	5	0	1
s19	11	7	3	1	6	2	4	1	5
s20	6	4	1	6	4	10	0	4	4
s21	6	8	5	1	3	6	6	1	4
s22	4	3	1	4	3	1	7	10	5
s23	4	2	5	0	7	2	5	6	8
s24	3	5	1	0	4	2	4	4	4
s25	4	4	5	1	4	2	6	5	4

図 6　**[Java 問題]4-1** の回答者全員の操作状況

ID	回数	割合	グラフ
s29	18	52%	##########
s30	7	20%	####
s31	1	2%	
s32	2	5%	#
s33	1	2%	
s34	2	5%	#
s35	3	8%	##
合計	34		

ID	回数	割合	グラフ
s29	0	0%	
s30	0	0%	
s31	6	17%	###
s32	8	23%	#####
s33	11	32%	######
s34	8	23%	#####
s35	1	2%	
合計	34		

n \ n+1	s29	s30	s31	s32	s33	s34	s35
s29	4	4	4	8	8	2	2
s30	3	1	12	6	3	4	0
s31	2	3	4	5	5	3	5
s32	2	4	6	12	4	8	2
s33	0	4	2	8	0	4	1
s34	2	4	3	4	2	7	2
s35	1	2	1	1	7	2	1

図 7　**[Java 問題]5-1** の回答者全員の操作状況

ID	回数	割合	グラフ
s5	6	54%	###########
s6	4	36%	#######
s7	0	0%	
s8	1	9%	##
s9	0	0%	
s10	0	0%	
s11	0	0%	
合計	11		

ID	回数	割合	グラフ
s5	1	9%	##
s6	0	0%	
s7	2	18%	####
s8	6	54%	###########
s9	1	9%	##
s10	1	9%	##
s11	0	0%	
合計	11		

n \ n+1	s5	s6	s7	s8	s9	s10	s11
s5	3	3	2	1	1	1	1
s6	2	1	1	1	2	6	0
s7	0	3	2	2	0	1	2
s8	0	0	5	2	1	1	0
s9	1	0	0	2	0	2	2
s10	0	1	1	6	2	1	1
s11	1	1	1	0	2	1	0

図 8　**[Java 問題]7-1** の回答者全員の操作状況

ID	回数	割合	グラフ
s5	5	45%	#########
s6	0	0%	
s7	1	9%	##
s8	2	18%	####
s9	3	27%	#####
合計	11		

ID	回数	割合	グラフ
s5	0	0%	
s6	2	18%	####
s7	6	54%	###########
s8	0	0%	
s9	3	27%	#####
合計	11		

n \ n+1	s5	s6	s7	s8	s9
s5	1	0	1	3	5
s6	2	5	2	0	0
s7	0	2	3	0	0
s8	0	4	3	1	0
s9	2	0	1	2	0

図 9　**[Java 問題]7-2** の回答者全員の操作状況

7. 実験の考察

7.1 問題ごとの考察

6 章の実験結果(図 5～図 9)に対する考察を以下に示す.

(1) [Java 問題]1-2 成績判定 1 の考察

本問題では s13 と s15，s17 で else-if 節が含まれるが，これらの順番は問題文で限定した.

回答者が最初に操作した行で最も多かった行は s9 で, 次に多かった行は s11 であった．s9 は関数の宣言文であり，最初にわかりやすい関数の構造の組み立てから思考をしていると推測できる．s11 は if 文の最初の行であり，s9 と同様に最初にわかりやすい構文の組み立てから思考をしていると推測できる．回答者が最後に操作した行で最も多かった行は s22 で, 次に多かった行は s10 であった. s22 は return

文，s10 は変数の宣言であり，どちらも if 文を先に完成させてから操作したと推測できる．回答者の操作パターンで最も多かったのは s13→s14 であった．s13 は 1 つ目の else-if 節であり，s14 は s13 の else-if 節に対応する変数の代入処理である．条件式に対応する処理を条件式の次に操作する傾向が見られた．この傾向は s11 から s20 にかけても同様の傾向が見られた．また，s9→s11 の操作パターンも多く見られた．s9 は関数の宣言文で，s11 は if 文の最初の行である．最初にわかりやすい関数の宣言から動かし，次に構文に着手する傾向が見られた．

(2) ［Java 問題］4-1 銀行口座と 2 つの通帳の考察

本問題ではコンストラクタと 3 つの関数が含まれるが，これらの順番は問題文で限定した．

回答者が最初に操作した行で最も多かった行は s20 で，次に多かった行は s17 と s18 であった．s20 は 1 つ目の関数の宣言文であり，順番の限定された 3 つの関数の順番を整えたと推測できる．s17 は変数の宣言文であり，変数の宣言が一番上に配置されることを理解していると推測できる．s18 は BankBook クラスのコンストラクタの宣言文であり，最初にわかりやすい構造の組み立てから思考をしていると推測できる．回答者が最後に操作した行で最も多かった行は s19 で，次に多かった行は s17 であった．s19 はコンストラクタ内の処理であり，ほかの 3 つの関数を完成させてから動かしたと推測できる．s17 は変数の宣言であり，関数を完成させてから動かしたと推測できる．回答者の操作パターンで最も多かったのは s18→s19 であった．s18 はコンストラクタの宣言文であり，s19 はコンストラクタ内の処理である．コンストラクタを完成させる際には宣言と処理を連続して操作する傾向が見られた．これに対して他の操作パターンでは s20→s22，s22→s24 という操作パターンが見られた．s20 は 1 つ目の関数の宣言文，s22 は 2 つ目の関数の宣言文，s24 は 3 つ目の関数の宣言文である．関数の宣言文を操作した後に次の関数の宣言文を操作する傾向が見られた．これは該当しうる処理が複数あったため，先に関数の宣言を順番通りに並べたと推測できる．ほかの操作パターンには s19→s17，s17→s18 という傾向が見られた．s17 は変数の宣言文，s18 はコンストラクタの宣言文，s19 はコンストラクタ内の処理である．変数の次にコンストラクタを操作したり，コンストラクタの次に変数を操作したりする傾向が見られた．これはクラスの上部に変数とコンストラクタが配置されることを理解しているためと推測できる．

(3) ［Java 問題］5-1 履修登録 1 の考察

回答者が最初に操作した行で最も多かった行は s29 で，次に多かった行は s30 であった．s29 は関数の宣言文であり，最初にわかりやすい関数の構造の組み立てから思考をしていると推測できる．s30 は for 文の最初の行であり，s29 と同様に最初にわかりやすい構文の組み立てから思考をしていると推測できる．回答者が最後に操作した行で最も多かった行は s33 で，次に多かった行は s32 と s34 であった．s33 は for 文の閉じかっこ，s32 は return 文を返す if 文であり，最後まで for 文に含む処理に迷いが生じていたと推測できる．s34 は return 文であり，for 文の構文を完成させてから移動する傾向があると推測できる．回答者の操作パターンで最も多かったのは s30→s31 であった．s30 は for 文の最初の行，s31 は for 文内の処理の 1 行目であった．for 文の宣言の次に正しい処理を動かす傾向が見られた．

(4) ［Java 問題］7-1 可変な値の格納の考察

回答者が最初に操作した行で最も多かった行は s5 で，次に多かった行は s6 であった．s5 は変数の宣言，s6 は 1 つ目の for 文の最初の行であった．最初にわかりやすい構造や構文の組み立てから思考をしていると推測できる．回答者が最後に操作した行で最も多かった行は s8 で，次に多かった行は s7 であった．どちらも 1 つ目の for 文に含まれる処理で，s8 は if 文，s7 はランダム数を使った変数の宣言である．2 つある for 文のどちらに含むかで迷いが生じたか，for 文の構文を完成させてから操作したと推測できる．回答者の操作パターンで最も多かったのは s6→s10，s10→s8 であった．s6 は 1 つ目の for 文の最初の行，s10 は 2 つ目の for 文の最初の行，s8 は 1 つ目の for 文に含まれる if 文であった．for 文の宣言の次に別の for 文の宣言や if 文を操作するといった，構文を連続して操作する傾向が見られた．次に多かったパターンは s8→s7 であった．s8 は 1 つ目の for 文に含まれる if 文，s7 は 1 つ目の for 文に含まれるランダム数生成の処理であった．同じ for 文に含まれる処理を連続して動かす傾向が見られた．

(5) ［Java 問題］7-2 要素削除時の順番の考察

回答者が最初に操作した行で最も多かった行は s5 で，次に多かった行は s9 であった．s5 は 1 つ目の for 文，s9 は 2 つ目の for 文であり，最初にわかりやすい構文の組み立てから思考をしていると推測できる．回答者が最後に操作した行で最も多かった行は s7 で，次に多かった行は s9 であった．s7 は 4 という数値を削除する処理であり，数値の削除の順番に迷いが生じたと推測できる．s9 は 2 つ目の for 文であり，1 つ目の処理との迷いが生じたか，上から順番に完成させたと推測できる．回答者の操作パターンで最も多かったのは s5→s9 であった．s5 は 1 つ目の for 文，s9 は 2 つ目の for 文であった．2 つの for 文を連続して動かす傾向が見られた．

7.2 プログラミング思考パターンの考察

7.1 節からプログラミング思考パターンを考察する．

・構文を整える思考

最初に動かしたピースから，まず，構文や定型文に着目する傾向が見られた．例えばつぎのような文である．

・if 文，for 文などの構文
・関数の宣言文，変数の宣言文，return 文などの定型文

プログラムの構造に沿った思考の傾向が多く見られた．

例えば，if 文の if 節から順に完成させたり，関数の宣言文の次に for 文を移動したりするなどである．

・処理の内容に沿った思考

ピースの移動プロセスからプログラム構造を考えない並び替えを実施しているケースが散見される．これは勘違い，あるいは知識不足が考えられる．複数の for 文がある問題では，for 文と実行文で組み合わせの誤りが発生している．これは問題を理解していなかったり，問題を理解していても構文内での実行プロセスのイメージができなかったりなど，論理的な思考ができていない傾向があると推測できる．

7.3 問題の狙いに対する考察

・関数やクラスの構造について

関数やクラスを完成させる問題では，return 文や変数の位置を正しく配置できているパターンが多かった．これは内部の処理の内容に関わらず，変数や関数の記法を理解しているためと考えられる．

・構文について

単純でわかりやすい if 文などの構文で構成される問題では，正解率が高い傾向にあった．しかし，授業に対応した範囲の教科書に載っている構文の中でも，授業の演習問題で出題されなかった switch 文や拡張 for 文では正解率が下がったため，対応範囲内でも出題されなかった構文は理解が薄いと推測できる．

・処理の順番について

宣言されていない変数を処理に使っている状態で完成としているパターンが少なくなかった．これは条件式を理解せずに if 文や for 文というだけで移動したか，あるいは確認不足と推測できる．また，要素の削除順によって出力が変わってしまう[Java 問題]7-2 では，並べ替え対象行が一番少なかったのにも関わらず，誤った順番で完成しているパターンが半数を超えた．これは ArrayList における要素の削除で添字が変わる性質を理解していなかったか，あるいは処理の前後で変数がどのように変化するかを考えていなかったと推測できる．

7.4 解答解説についての考察

問題の出題後に解答解説の提示とあわせて実施したアンケートの結果を以下に示す．アンケートは第 4 回以降から実施したため，[Java 問題] 1-2 は対象外である．表 3 は質問「解答を最後まで読みましたか？」の回答数，表 4 は質問「解答は理解に役立ちましたか？」の回答数である．アンケートの結果では，回答者の過半数が「はい」と回答しており，解答解説が内容の理解や復習に役立っているといえる．

表 3 質問「解答を最後まで読みましたか？」の回答数

問題	4-1	5-1	7-1	7-2
はい	34	28	11	11
いいえ	6	6	0	0

表 4 質問「解答は理解に役立ちましたか？」の回答数

問題	4-1	5-1	7-1	7-2
はい	39	30	10	10
いいえ	1	4	1	1

8. おわりに

本研究では，学習進度に対応したプログラムをパズル化したパズルの並べ替え操作から，プログラミング時の思考過程を分析することを目指した．先行研究で開発されたツール「ジグソー・コード」および「ジグソー・コード 2」を利用した実験により，回答者のプログラミング思考過程を分析することで，構文を整える思考，処理内容に沿った思考のパターンの傾向を検出した．問題作成時の狙いに対して，授業の演習問題で出題されなかった構文は理解が薄いことや，例えば ArrayList の要素削除時に添字が変わるといった性質を理解していないと解けない問題では，正解率が低くなることが判明した．

今後は正解者と不正解者の傾向比較や授業における理解度との関連について分析し，プログラミング授業での予習や復習に適用するなど実用化を目指す．実用化に向けて「ジグソー・コード」の操作後に解答解説，分析結果やアドバイスなどを提示するシステムを構築していく．

謝辞 本研究は JSPS 科研費 17K01085 の助成を受けたものである．

参考文献

[1] 日本経済再生本部．未来投資戦略 2018 -「society 5.0」「データ駆動型社会」への変革-
https://www.kantei.go.jp/jp/singi/keizaisaisei/pdf/miraitousi2018_zentai.pdf
[2] 文部科学省．小学校学習指導要領 2017.
http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/05/07/1387017_1_2.pdf
[3] 山口琢，大場みち子，できごと，手順，プログラムや地理の

並べ替え操作の測定と分析，情報教育シンポジウム論文集，2018(25), pp179-184, 2018.
[4] 山口琢，小林龍生，高橋慈子，大場みち子，パズル操作の測定・分析による思考の推定，日本認知科学会第 35 回大会，2018.
[5] 山口琢，大場みち子，並べ替えプログラミングの測定・分析，日本ソフトウェア科学会第 35 回大会, 2018.
[6] 角田博保，久野靖，短冊型問題: プログラミングの技能を評価可能な試験出題形式，夏のプログラミング・シンポジウム 2016「教育・学習」, 2016.
[7] 久野靖，思考力・判断力・表現力を評価する試験問題の作成手順，情報処理学会 情報教育シンポジウム論文集,2018(1),1-8, 2018.
[8] Dale Parsons, Patricia Haden, Parson's programming puzzles: a fun and effective learning tool for first programming courses, 2006.
[9] Geela Fabic, Antonija Mitrovic and Kourosh Neshatian, investigating strategies used by novice and expert users to solve Parsons problems in a mobile Python tutor, The 24th International Conference on Computers in Education (ICCE 2016), 2016.
[10] Amruth N. Kumar, Representing and Evaluating Strategies for Solving Parsons Puzzles, International Conference on Intelligent Tutoring Systems (ITS2019), 2019.
[11] 山口琢，大場みち子，高橋慈子，小林龍生，作文行動を測定・分析するためのマトリックス型テキスト編集モデルの設計，情報処理学会 研究報告コンピュータと教育(CE), 2017-CE-141(2), 2017.
[12] 大場みち子，山口琢，作文行動の記録・分析ツールを用いた就活自己紹介書の作成と分析，情報処理学会 研究報告コンピュータと教育(CE), 2018-CE-147(12), 2018.
[13] 山口琢，伊藤恵，大場みち子，プログラミング・パズルの測定と分析，研究報告ドキュメントコミュニケーション(DC), 2018-DC-111(2), pp1-6, 2018.
[14] 公立はこだて未来大学，学部シラバス（講義要項），https://www.fun.ac.jp/control-panel/wp-content/uploads/2018/04/H30_syllabus.pdf
[15] 川北紘正，大場みち子，山口琢，プログラミング思考過程に基づくプログラミング時の行動分析と傾向，第 81 回全国大会講演論文集, 2019(1), 521-522, 2019.
[16] 大場みち子，山口琢，川北紘正，パズルを利用したプログラミング思考過程の分析，情報教育シンポジウム論文集, 152-159, 2019.