プログラミング演習における行き詰まり状況の分類手法

2019SE010 平野翼
指導教員：名倉正剛

1　はじめに

大学学部では，実践力を高めるために情報システム開発等を行う演習科目が開講されている [1]．そのような演習では，一般的に多くの学生に対し指導者は少数であるので，一度の多くの学生が演習に行き詰った場合に対応することが困難な状況が発生する．

2　演習における行き詰まりと対象にする課題

2.1　プログラミング演習での行き詰まり

プログラミング演習では，一般的に指導者が学生に与えた課題に対して，学生は限られた時間内にプログラムを記述し提出する．学生が与えられた課題を完成することが困難な状態を，「行き詰まり」と呼ぶ．プログラミング演習では受講者自身が自律的にコーディングを行うことが求められる．しかしトラブルや理解不足に起因する様々な問題によって，課題を完成させることのできない行き詰まりの状態が発生しうる．コード量変化に着目し長時間に進捗が少ない学生を検出したり [2]，あるいは修正・コンパイル・実行を繰り返す探索的プログラミング行動を行っている学生を検出したりする手法 [3] を利用したりすることで，そのような行き詰まり状態の学生を検出できる．

2.2　本研究で対象にする課題

演習授業では，教員や TA などの指導者は通常は教室内を巡回し，学生の編集画面を確認することでコーディング状況を把握しながら，学生による質問への対応を行う．その際に行き詰まり状態に陥った学生は，指導者に対して質問を行う．

課題の難易度や学生の理解のレベルによっては一度に多くの学生が行き詰まることがある．前述の行き詰まり検出手法は，行き詰っていても質問ができないような学生や，自身が行き詰っていることを認識できないような学生を支援する．しかし行き詰まり状態の多くの学生を検出できても，少数の指導者により対応することは容易ではない．

多数の学生が同じ課題に対して同じような内容で行き詰っている状況も起こり得る．その場合は一斉に指導できる可能性があるが，同じ内容で行き詰っているかどうかは個々の学生の内容を精査しないとわからない．したがって，指導に対応できる指導者の人手が足りないにもかかわらず，結果的に重複した内容の指導を個別に行うことになる場合があり，効率的に指導を実施することが困難である．

3　提案

3.1　概要

本研究では，Java 言語を対象にしたプログラミング演習授業において，同じような状況で行き詰まっている学生の行き詰まり状況をクラスタリングして指導者に提示する手法を提案する．行き詰っているそれぞれの学生が作成しているプログラムには，正解プログラムに対して異なるコード部分が存在する．そこで行き詰っている学生の作成しているプログラムと正解のプログラムで異なるコード部分を，その学生が行き詰っている箇所のコード断片（以降，行き詰まりコード片，と呼ぶ）として抽出する．複数の学生の行き詰まりコード片が一致する場合，それらの学生を同時に指導できる可能性がある．そこで提案手法では，行き詰りコード片をクラスタリングすることによって，行き詰っている学生をクラスタリングし，指導者に提示する．

3.2　分類手順

行き詰まりコード片の分類手順を以下に示す．

(1) 前処理として表記ゆれを統一
(2) 行き詰りコード片を検出
(3) 行き詰りコード片をクラスタリング

提案手法によって行き詰りコード片を分類する際に，行き詰っている学生のコーディングスタイルの相違により，同じ内容に対する行き詰りコード片が異なる文字列になる場合がある．例えば if 文や for ループに対する複合文の開始を示すブロックの括弧（‘{’）を if や for と同じ行に記述するか，それとも次の行に記述するかによって，同じ部分で行き詰っていても改行や空白の挿入箇所の異なる文字列になる．(1) ではこのようなコーディングスタイルに代表される表記ゆれを統一する．

次に，行き詰っている学生のプログラムと正解のプログラムの文字列を比較することで，行き詰りコード片を検出する．(2) では，まず行単位で差分を検出し，正解プログラムと異なる文字列の行を特定する．そして特定した行について，正解プログラムと学生プログラムの該当する行を比較し，それらの行内で異なる文字列を抽出する．この手順で抽出した文字列が行き詰りコード片であり，それぞれのコード片をその文字列を含む行に関連付ける．

そして，抽出した行き詰まりコード片をクラスタリングする．クラスタリングは，行き詰まりコード片が存在するプログラム名や行き詰まりコード片が存在する行の行番号などによって行う．

また，正解プログラムに対して記述できていない文が連続して複数存在する場合，それらをまとめて 1 つの行き

ソースコード 1: 正解プログラム

```
1  public class MaxOf
2  {
3      static int maxOf(int a[])
4      {
5          int max = a[0];
6          for(int i = 1; i <= a.length; i++)
7          {
8              if(a[i] > max)
9              {
10                 max = a[i];
11             }
12         }
13         return max;
14     }
15 }
```

```
[Group #1]
<Correct Answer> (MaxOf.java)
 4:        for(int i = 1; i <= a.length; i++)

<User A> (MaxOf.java)
 4:        for(int i = 1; i < a.length; i++)

[Group #2]
<Correct Answer> (MaxOf.java)
 5:            if(a[i] > max)

<User A> <User B>
 None
```

画面 1: 分類された行き詰まりコード片の表示

詰まりコード片のまとまりとする．このとき，行き詰まりコード片のまとまりに含まれる文を 1 つずつ減らし行き詰まりコード片が一致するまとまりが存在するかを確認し，存在した場合，そのまとまりが記述できていないクラスタを生成する．

その後，行き詰まりコード片をクラスタリグした結果を指導者に提示する．このとき，「その行き詰まりコード片が存在するプログラム名」，「クラスタリングされたユーザ ID」，「行き詰まりコード片が存在する行」を表示する．正解プログラムに対する不足行や追加行といった，片側にしか行が存在せず対応する行が存在しない場合，行が存在しない方は「None」と出力する．

3.3 実施例

リスト 1 に示すプログラムの記述に対して，行き詰っている学生が 2 名（学生 A，B）存在しているとする．学生 A は，4 行目の for ループに記述する条件式が間違えており，5 行目の if 文の記述については学生 A，B ともにできていない．このような状況で提案手法により行き詰まりコード片をクラスタリングした結果を画面 1 に示す．この画面に示すように，正解プログラムとの差分が複数の学生に共通していた場合は，クラスタリングして表示する．

4 評価

提案手法の評価を行うために，実際に行われた java 言語を扱うプログラミング演習において，既存研究 [2] によって取得した情報を用いた．既存研究 [2] で取得できる情報は，「ユーザ ID」「ある特定の行動を行った時刻」「ある特定の行動を行った際のソースコード」である．その中から，一定時間行動を行っていない学生を行き詰まっている学生として，行き詰まっている学生のプログラムの中から同じ時間帯で行き詰まっていた学生 10 人を対象に評価実験を行った．10 人の学生が取り組んでいた課題の内訳を表 1 に示す．また，3.2.1 項でのべたように，プログラム内の変数名については演習課題の出題時に問題文の中で定義されるものとしているので，プログラム間で変数名の差異を取り除くために目視で確認し，差異が存在した場合は手動で差異を取り除いた．

表 1: 対象とした学生が取り組んでいた課題の内訳

学生	記述しているプログラム	行き詰まりコード片
A	Work03.java	4
B	Work03.java	1
C	Work03.java	8
D	Work04.java	2
E	Work02.java	2
F	Work04.java	1
G	Work02.java	2
H	Work02.java	1
I	Work02.java	18
J	Work01.java	8

この 10 人の学生，プログラムに対して実行したところ，行き詰まりコード片の取得，行き詰まりコード片に基づいたクラスタリングの実行が確認できた．しかし，クラスタリングの結果生成されたクラスタの数が，41 個と膨大な数となってしまった．これでは，参照するクラスタが膨大すぎて，指導者がこのクラスタリング結果を参照して指導を行うことは実用的ではない．このことを解消する方法として，連続して検出された行き詰まりコード片を個別に扱わずに，連続する行き詰まりコード片を結合する方法が考えられる．

また，条件式の表記の違いにより，正解は 1 つとは限らないことが考えられ，そうした場合には対応が出来ない．

5 おわりに

プログラミング演習の課題に対して同時に行き詰まっている複数の学生を，学生が行き詰まっている箇所でクラスタリングし，表示することが可能となった．

参考文献

[1] 情報処理推進機構 IT 人材育成本部. IT 人材白書 2017. 独立行政法人情報処理推進機構, 2017.

[2] 井垣宏, 斎藤俊, 井上亮文, 中村亮太, 楠本真二. プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案. 情報処理学会論文誌, Vol. 54, No. 1, pp. 330–339, 2013.

[3] 槇原絵里奈, 井垣宏, 吉田則裕, 藤原賢二, 飯田元. プログラミング演習における探索的プログラミング行動の自動検出手法の提案. コンピュータソフトウェア, Vol. 35, No. 1, pp. 110–116, 2018.