レーベンシュタイン距離と実装行数を用いた学生のプログラミング行動把握手法の検討

筒井　善規[1]　井垣　宏[1]

概要：プログラミング演習において，学生一人一人が課題をどのように解いているかを知ることは，授業の質向上という観点において非常に重要である．本研究では，既存研究でも用いられているレーベンシュタイン距離と実装行数を利用して，学生のコード編集履歴を可視化し，5 種類のプログラミング行動が特定できる可能性があることを確認した．

1. はじめに

プログラミングの教育を目的としたプログラミング演習は一般に，少数の教員とアシスタントが数十名〜100 名程度の学生を指導するといった形式で実施されており，学生らは与えられた課題を締め切りまでに実装することが求められる [2]．昨今ではコロナ禍の影響もあり，リモートでのプログラミング演習なども実施されている．そのため，各学生がどのように課題を解いているかを教員やアシスタントが知ることはますます難しくなりつつある．

そこで本研究では，リモート環境で実施されたプログラミング演習において収集された学生のコード編集履歴を利用し，実装時のプログラミング行動を特定する手法を検討する．既存研究で対象となっている行き詰まり状況 [1] だけでなく，様々な状況の具体的な事例がグラフにどのように影響するかを分析する．

2. プログラミング演習環境と学生のプログラミング行動

2.1 プログラミング演習環境

著者らの大学では 2 年生後期に Java を対象としたプログラミング演習を実施している．演習では Visual Studio Code，Amazon Corretto11, Windows 用の bash ターミナルを学生らのノート PC にインストールさせている．この環境では学生らのサポートやトラブル時の手戻りを目的として，学生が何の入力も行わずに 5 秒経過すると自動的に対象ファイルのスナップショットが保存される仕組みをエディタの機能を利用して導入している．このスナップショット，すなわちファイル編集履歴は学生らに収集内容と使途を説明したうえで，課題提出時に回収している．

2.2 学生のプログラミング行動の分析

プログラミング演習における学生のプログラミング行動については，様々な分析が行われている．その中でもコードの編集履歴を対象として，藤原ら [1] はレーベンシュタイン距離を用いた行き詰まり特定手法を提案している．レーベンシュタイン距離とは，ある文字列を別の文字列に変換するのに必要となる最小の手順数であり，編集距離とも呼ばれる．藤原らは受講生らの課題ごとの任意時刻におけるスナップショットと最終提出ファイル間のレーベンシュタイン距離を算出してグラフを作成し，その推移から行き詰まり状況と行き詰まり箇所を特定する手法を提案している．

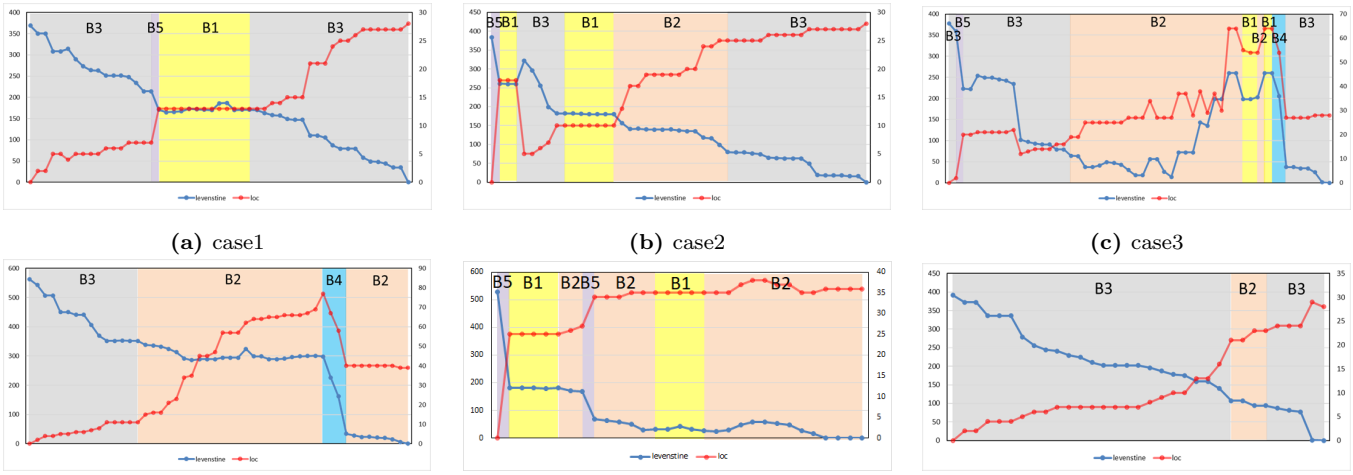
本研究では，レーベンシュタイン距離を用いた藤原らの手法を参考に，著者らの所属する大学におけるプログラミング演習における学生らの多様なプログラミング行動を分析する手法を検討する．

3. レーベンシュタイン距離と実装行数を用いたグラフ化

2.1 節で述べたプログラミング演習で得られたファイル編集履歴から，学生の課題提出時のコードと各スナップショットの間のレーベンシュタイン距離及びスナップショットごとの実装行数をグラフ化し，学生らのプログラミング行動の分析を行う．なお，本稿では 2020 年度後期に実施した Java 演習の課題の中から，switch 文に対する課題 1 つを選び，課題を完成させた学生を対象としてグラフ化を行う．

図 1 に本稿で対象とする課題を完了したある 6 名の学生

[1] 大阪工業大学 大学院情報科学研究科
Graduate School of Information Science, Osaka Institute of Technology 1-79-1 Kitayama, Hirakata City, Osaka, 573-0196 Japan



(a) case1 **(b)** case2 **(c)** case3

(d) case4 **(e)** case5 **(f)** case6

図 1: 6 名の学生を対象としたレーベンシュタイン距離と実装行数の推移

のレーベンシュタイン距離（青線）と実装行数（赤線）の時系列での推移をグラフ化したものを示す．ここで，横軸はエディタによって保存されたスナップショットを時系列順に並べたときの保存時刻を表しており，縦軸は左側がレーベンシュタイン距離，右側がスナップショットごとの実装行数（改行のみの行やコメントのみの行は除く）を表している．なお，レーベンシュタイン距離算出時はコメント，改行コード，空白は除いているが，正規化は実施していない．また，今回グラフ化した学生は提出時に全員仕様通りに実装を完了していることを確認している．

4. グラフとプログラミング行動の関係

今回作成したグラフ及びスナップショットから，以下の 6 種類のグラフとプログラミング行動の関係を識別できた．

B1. 試行錯誤

行数及びレーベンシュタイン距離両方が一定期間変化がない場合，既存行を微修正しながら試行錯誤していることが確認できた．レーベンシュタイン距離が増えるほどの変更ではなく，既存行を微修正しているだけであるため，レーベンシュタイン距離と実装行数ともに変化していない．

B2. 間違った実装の継続

このプログラミング行動では，case4 や case5 のように最終的なコードが間違っていた場合，case2 や case6 のようにレーベンシュタイン距離が停滞または減少しており，実装行数のみが増えている場合，case3 のように実装行数，レーベンシュタイン距離両方が増加傾向にある場合がある．また，case3 はより大幅に間違い続けている状況が確認された．

B3. 順調な開発

レーベンシュタイン距離と実装行数の推移から，レーベンシュタイン距離が減少傾向にあり，実装行数が増加傾向にあるときは，順調に開発が進んでいることが確認できた．

B4. 間違いの気づきに伴う編集

B3 と行動としては類似しているが，case3，case4 のように，B1 や B2 のあとにレーベンシュタイン距離と実装行数両方が大幅に減少しているときは，間違いに気づいて大幅な修正を行っていたことが確認できている．

B5.Copy&Paste

実装行数が大幅に増えている場合は Copy&Paste が行われていることが多い．今回のグラフ化では Copy&Paste にあわせてレーベンシュタイン距離も大きく減少していた．スナップショットを確認したところ，学生が自分の過去のコードを Copy&Paste した場合と自分以外のコードを Copy&Paste した場合の両方の行動が確認できた．

5. おわりに

今回，レーベンシュタイン距離と実装行数から学生のファイル編集履歴をグラフ化し，5 つのプログラミング行動を特定した．今後は分析する課題，人数を増やし，プログラミングパターンの抽出や授業の改善につなげていく．

謝辞 本研究の一部は JSPS 科研費 17K00500 の助成を受けた．

参考文献

[1] 藤原賢二，上村恭平，井垣宏，吉田則裕，伏田享平，玉田春昭，楠本真二，飯田元：スナップショットを用いたプログラミング演習における行き詰まり箇所の特定，コンピュータソフトウェア，Vol. 35, No. 1, pp. 1_3–1_13 (2018).

[2] 槇原絵里奈，藤原賢二，井垣宏，吉田則裕，飯田元ほか：初学者向けプログラミング演習のための探索的プログラミング支援環境 Pockets の提案，情報処理学会論文誌，Vol. 57, No. 1, pp. 236–247 (2016).