Vol.2021-CE-160 No.4
2021/6/5

バグ修正とコードリーディング -アルゴリズム学習に適しているのはどちらか-

倉持 雄樹[1,a)]　坂本 一憲[1,b)]　鷲崎 弘宜[1,c)]　深澤 良彰[1,d)]

概要：アルゴリズム学習において，アルゴリズムのコードを読むことが必要と考えられている．しかし，ただ読むだけでは明確な終了条件が存在せず，理解不足のまま学習を終えてしまったり，学習のモチベーションの維持が難しくなったりするという問題がある．そこで，本論文ではリーディングの代わりにバグを含むコードを読んでそれを修正する手法を提案する．評価のためにシステムを作成し，40 人の被験者を募集し評価実験を行った．その結果，バグ修正問題を解いた後の確認問題の正答数はリーディングのみの場合と比べて p 値は 0.340 となり有意な差は見られなかったが，ハノイの塔を題材として扱った場合は学習効果が高まることが示唆された．

1. 導入

アルゴリズムの理解にあたって，コードリーディングは重要な学習方法の一つである [1]．そのため，コードリーディングのスキルを鍛える方法がいくつか提案されてきた．ビーコンをコードに埋め込んでコードリーディングを助けるツール [2] や，学生のコードトレーシングの流れを可視化するツール [3] などがその例である．

しかし，アルゴリズムの理解を踏まえて実際に実装をすることへの支援は少ない．アルゴリズム自体を理解できていても，その実装がなされたコードを理解できていなければ，自分でコードを書くことにはつながらない．

近年，ミューテーション解析と呼ばれるテスト技法をプログラミング教育に活用する研究事例が増えている．例えば，亀井らは，ミュータント生成を利用して学習の支援をするツールを作成した [4]．これは，ソフトウェアテストの分野に存在するミューテーション解析 [5] と呼ばれる手法を教育に応用した事例である．亀井らは，二項演算子に関するバグを埋め込み，それに関連する問題を初学者に提示する実験を行った．その結果，演習問題を解く学習法を行ったグループよりもバグを埋め込んだ学習法を行ったグループの方が学習効果が向上したことが確認された．また，バグ埋め込みの学習法は初学者の学習意欲の向上にも与することがアンケート結果から判明した．

ミューテーション解析は，他にもプログラミング教育に応用されている．例えば，ミューテーションツールを改良して作られたクイズツール [6] や，ミューテーションを学生に行わせつつテストについて学ぶツール [7] などが存在している．

本論文でもミューテーション解析技術に着目して，同技術を用いてコードリーディングのスキルを鍛える手法を提案する．学習時に読むコードをミュータントとし，内包するバグの修正を行わせる．これにより，ただ漠然とアルゴリズムの実装コードを読むよりもより注意深くコードの全体を読むようになるのではないか，という仮説を立てた．そこで，この仮説を検証するために bug-fix-learning-system(BFLS) と名付けたシステムを作成した．

本論文における研究課題は以下の 4 つである．

RQ1　バグ修正問題によってアルゴリズムの理解度は向上するか

RQ2　バグ修正問題によって学習時間が伸びたかどうか

RQ3　バグ修正問題について主観評価でどう思うか

RQ4　何を基準にアルゴリズムの学習を終了するか

本論文の貢献は以下のとおりである．

- アルゴリズムの学習方法としてバグ修正問題を提案したこと．
- 上記手法を検証するためのシステム BFLS を開発したこと．
- BFLS を使って被験者実験を実施して，リーディングと比べてバグ修正問題後の確認問題の正答数の向上の可能性を確認したこと．

[1] 早稲田大学
Waseda University
[a)] ykuramochi@akane.waseda.jp
[b)] exkazuu@gmail.com
[c)] washizaki@waseda.jp
[d)] fukazawa@waseda.jp

2. 関連研究

R.smith らは，VizQuiz というアクティブコードラーニングとプログラムの視覚化を結びつける選択肢式のクイズツールを作成した [6]．VizQuiz は mutpy [8] を改良して作られたもので，ミュータントを自動生成する機能を有している．教員側は初期化コードと実行すべきプログラム，問いたい概念を選択することで，VizQuiz がミュータントとヒントを生成する．このミュータントは問いたい概念の部分にバグが仕込まれており，ヒントはその概念に対応するものとなっている．学生側は VizQuiz によって初期状態と実行すべきプログラム，および 4 択の終了状態が提示され，正しい終了状態を選択するよう求められる．2 学期にわたる実験の結果，VizQuiz を用いることによって，学生の計算的思考能力と決定的な思考能力の改善が報告された．他にミュータントをミューテーション解析以外の用途に用いた例として，Code Defenders [7] が挙げられる．ソフトウェアテストに関する教育にゲーミフィケーションを導入したもので，コードにバグを埋め込んでミュータントを手動で生成する攻撃側と，そのミュータントが生き残らないようなテストスイートを用意する守備側に分かれて戦う，というものである．このシステムを学期を通じて授業で扱った結果，学生によく受け入れられた上でソフトウェアテストの練習につながり，学生のパフォーマンスが向上した．また，別の年の研究では，Code Defenders を実施することで，自動化されたツールで作成されたものよりもより強力なテストスイートとミュータントが作成されることが明らかとなった [9]．これらの研究は，ミュータントを教育に応用しているという点で本研究と共通している．

B.Harrington らは，コードトレーシングとコードライティングの達成度の差に関する研究を行った [10]．学習時にはコードライティングの前提としてコードトレーシングが求められるという階層構造が存在すると一般に言われている．学習がある程度進んだ後の成績はより深い学習概念を理解する能力に左右されるため，コードトレーシングとコードライティングの差が小さくなることが予想された．実際に学習がある程度進んだ学生について調査をすると，その差は小さくなっていることが明らかとなった．ただし，どちらか一方が不得意な学生はコンピュータサイエンスのコース全体のパフォーマンスが低下していることも同時に示唆された．また，A.Kumar はコードトレーシングとコードライティングの関係性について実験を行った [11]．コードトレーシングによる，主に言語構文に関するコードライティングの改善は以前に指摘されていたが [12]，プログラムの意味論に関する改善は不明確であった．そこで，4 学期にわたって学生にコードトレースの前後に対照群を設けつつコードを書かせる実験を行った．その結果，実験群ではプログラムの意味論に関するコード記述能力の向上が観察されたが，対照群では観察されなかった．このことから，プログラムの意味論に関してもコードトレーシングを行うことによってコードライティングの能力が向上することが確認された．本論文ではある程度学習が進んだ学生を対象としていることや，コードトレーシングでコードライティングの能力が向上することなどを踏まえ，コードトレーシングに主眼を置いている．

3. コードリーディングとバグ修正によるアルゴリズム学習

3.1 コードリーディングによるアルゴリズム学習とその課題

従来のアルゴリズムの学習において，アルゴリズムの解説とそのアルゴリズムが実装されたコードを読んでアルゴリズムを理解して，実際にコードを記述することで理解を定着させる活動が一般的である．

しかし，解説及びコードを読むだけではアルゴリズムを十分に理解できない可能性が危惧される．また，コードを読むだけでは終了条件が存在しないことも課題である．これによって 2 つの問題が発生する．1 つ目の問題として，学習者はどこまでやれば学習を終えていいかがわからず，理解が完全でないまま学習を終えてしまう可能性がある．また，2 つ目の問題として，ゴールがないことによって，学習をする意欲が維持しにくいことも挙げられる．

3.2 バグ修正問題によるアルゴリズム学習の提案

コードリーディングを用いた学習の問題点を解決する手法として，バグ修正問題を提案する．バグ修正問題では，図 1 に示すように，アルゴリズム学習時にコードリーディングを行う代わりに，バグ入りのコードを読み，そのバグを修正する作業を行う．図 1 でいう解説は自然言語で記述されたアルゴリズムに関する解説をまとめた文書であり，バグはその解説で説明されている仕様を満たしていないソースコード中の誤りを指す．また，教員が作成するバグ入りコードは，教員が書いた正解コードの一部を改変して仕様を満たさないようにすることで作成されるものである．ここで，教員側が用意するバグはアルゴリズムの根幹に関わる部分に関連させることが重要である．具体的なフローとしては，教員がアルゴリズムの解説とバグ入りのコードをシステムに登録し，システムはこの解説とコードを学習者に提示する．学習者は解説を読みつつバグ入りのコードを読み，解説の仕様に照らし合わせながらバグを修正してシステムに提示する．システムはそのバグ修正の正誤について提示して，学習者はそこでアルゴリズム学習を終了するか再びバグ修正を続けるかを選択することができる．この作業では学習終了の目安が「バグ修正が完了したとき」とはっきりするため，先ほどあげた課題を解決することができる．具体的には，1 つ目の問題に対しては，含まれてい

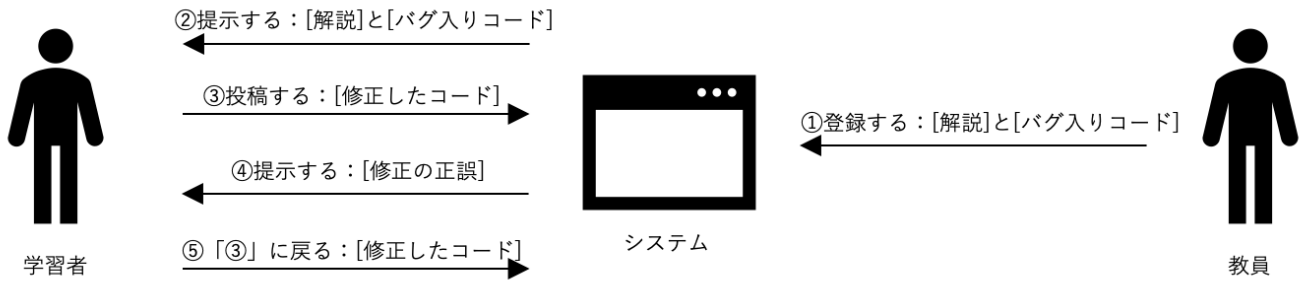

図 1 バグ修正問題によるアルゴリズム学習

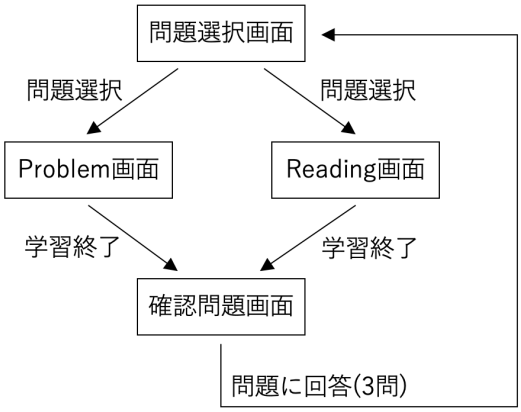

図 2 画面遷移図

るバグを全て修正できたら一定以上の理解度を達成できている，と判断することが可能となる．また，達成したい理解度はバグの混入のさせ方で調整をすることができる．2 つ目の問題に対しては，終了条件が明確になっており，バグを修正するという目標があるため学習する意欲を維持しやすくなっているといえる．

4. bug-fix-learning-system の提案

本論文では，バグ修正問題がアルゴリズム学習において有効かどうかを検証するためのシステム BFLS の提案をする．

4.1 概要

BFLS は，バグ修正問題に取り組んだ後に確認問題を行った場合と，単純なプログラムリーディングのみに取り組んだ後に確認問題を行った場合との比較を行うことを目的としている．そのために，バグ修正の正誤判定や時間計測機能，および確認問題の正誤判定を行う．

4.2 実装

BFLS は以下の 4 つの画面で構成されており，その画面遷移は図 2 のように行われる．

- 問題選択画面
- Problem 画面
- Reading 画面
- 確認問題画面

問題選択画面から，Problem 画面および Reading 画面へ進むことができる．BFLS にアクセスした時点で A 群と B 群に分けられている．A 群は 1 つ目のアルゴリズムに Problem，2 つ目のアルゴリズムに Reading が割り当てられ，B 群はその逆が割り当てられるようになっている．Problem 画面では，バグ修正問題を実施する．この画面に表示されるコードにはバグが含まれており，被験者はこのバグを BFLS 上で直接修正する．バグ修正が済むと，その修正が正しいかどうか判定をすることができる．学習が終了したら，そのまま確認問題画面へ進むことができる．Reading 画面では，コードリーディングを実施する．この画面に表示されているコードにはバグは含まれておらず，純粋にコードを読む学習を行うことができる．Problem 画面と同様，学習が終了したらそのまま確認問題画面へ進むことができる．確認問題画面では，アルゴリズムに関する 3 つの確認問題を実施する．1 つ目はアルゴリズムに関する説明文として正しいものを 4 択の中から選択する問題で，2 つ目はフローチャートで示されたアルゴリズムのステップについて空欄となっているステップを答える問題，3 つ目はトレース問題としてコードに特定の入力をした場合の変数や配列の値を答えさせる問題である．BFLS は問題選択画面を除いた 3 つの画面における所要時間を計測しており，自動的にデータベースに保存する仕組みとなっている．

5. 実験

5.1 実験環境

早稲田大学の 40 名を被験者とした．その際，大学で C 言語に関する講義の単位を 1 以上取得済みで，簡単な C 言語プログラムの読み書きができることを条件とした．これは，基本的なプログラミングの構文への理解を前提として，純粋にアルゴリズムの学習に主眼を置くために設定した．そして，比較実験を行うためこの 40 名を 2 つの群に

ランダムに分類した．A 群には 20 名，B 群には 20 名を割り当てた．実験で扱ったアルゴリズムはエラトステネスの篩とハノイの塔の 2 つである．A 群はエラトステネスの篩をリーディングで学習し，ハノイの塔をバグ修正問題で学習した．また，B 群はその逆で，エラトステネスの篩をバグ修正問題で学習し，ハノイの塔をリーディングで学習した．学習終了後は確認問題を行った．この中で，各学習時間および確認問題の正答数を測定した．また，実験終了後にはアンケートを実施した．

5.2 実験結果

本節では，正答数，学習時間，アンケート結果について述べる．

5.2.1 正答数

正答数は，確認問題において正答した問題の数である．確認問題は各アルゴリズムについて 3 問ずつ用意したため，アルゴリズムごとの正答数の最大は 3，最小は 0 である．また，各群について，正答数を基に平均正答数を算出したものを表 2 にまとめると同時に，箱ひげ図を図 3 に示す．平均正答数は，各群の確認問題における正答数の平均をアルゴリズム別にとったものである．例えば，A 群の平均正答率（バグ修正問題）は，A 群の被験者がバグ修正問題として学習したハノイの塔に関する確認問題の正答数の平均をとったものを表している．

この結果について検証するため，ウィルコクソンの符号順位検定を用いる [13]．これは，2 つのデータ間における中央値の差を検定する手法の 1 つで，得られた 2 つのデータ間に対応があるときに用いられるものである．この手法では母集団の分布は仮定しない．本論文で比較する 2 群は，A 群と B 群をまとめた同一被験者におけるバグ修正問題後の確認問題の正答数とリーディング後の確認問題の正答数とする．そのため，対応があり，正答数の分布が不明であるから，この手法が適しているといえる．なお，教育分野における有意水準は 0.05 が一般的に用いられているため，本論文でもこの数字を用いる．

A 群と B 群をまとめた 40 名について，バグ修正問題とリーディングを比較する形でウィルコクソンの符号順位検定を行なった結果，p 値は 0.340 となった．この値は 0.05 よりも十分に小さいとは言えず，バグ修正問題とリーディングの正答数には有意差は見られなかった．また，扱ったアルゴリズムごとに分けて検定すると，エラトステネスの篩では p 値は 0.310，ハノイの塔では p 値は 0.0725 となった．さらに，A 群と B 群と分けて検定すると，A 群では p 値は 0.613，B 群では 0.0332 となった．

5.2.2 学習時間

学習時間は，アルゴリズムの学習を行なった時間である．具体的には，バグ修正問題であれば，解説資料を読みながらバグ入りコードの修正を行なった時間，リーディングであれば解説資料を読みながらバグのないコードを読んだ時間である．平均学習時間は，各群の各アルゴリズムにおける学習時間の平均をとったものである．この平均学習時間を算出した結果を表 3 に示す．また，学習時間に関する箱ひげ図を図 4 に示す．

検証には正答数と同様にバグ修正問題とリーディングを比較する形でウィルコクソンの符号順位検定を用いる．その結果，p 値は 0.0000000138 となった．この値は 0.05 よりも十分に小さいため，バグ修正問題とリーディングの学習時間には有意差が見られたといえる．

5.2.3 アンケート結果

実験終了後に実施したアンケートについて述べる．アンケートは以下の 4 つで構成されている．

Q1 今回の学習方法に対して意欲的に取り組めたか？

表 1 確認問題の平均正答数（群）

群	バグ修正問題	リーディング	p 値
A 群	2.50	2.65	0.613
B 群	2.50	2.05	0.0332
A+B 群	2.50	2.35	0.340

表 2 確認問題の平均正答数（アルゴリズム）

アルゴリズム	バグ修正問題	リーディング	p 値
エラトステネスの篩	2.50	2.65	0.310
ハノイの塔	2.50	2.05	0.0725

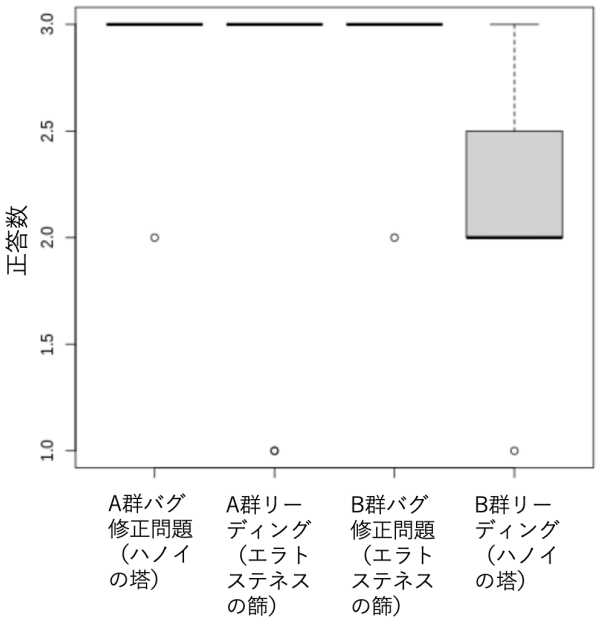

図 3 正答数の箱ひげ図

表 3 アルゴリズムの平均学習時間（秒）

群	バグ修正問題	リーディング	p 値
A 群	1,184	337	0.000009537
B 群	804	409	0.001209
A+B 群	994	373	0.0000000138

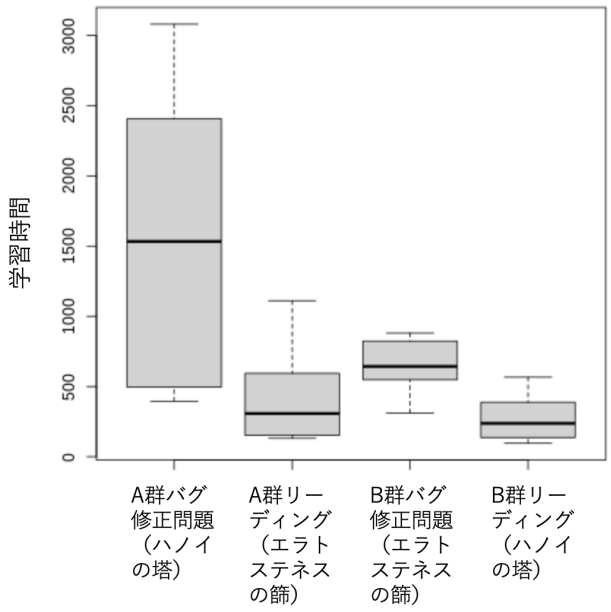

図 4 学習時間の箱ひげ図

表 4 アンケート結果

学習方法	Q1 平均値	Q2 平均値	Q3 平均値
バグ修正問題	4.15	3.63	3.45
リーディング	3.90	3.30	3.13

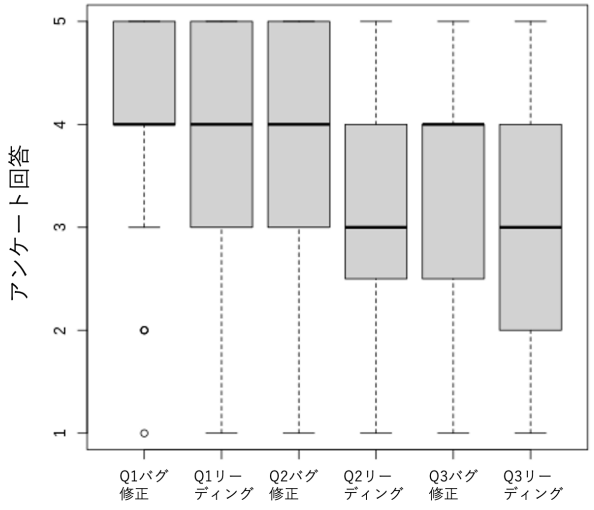

図 5 アンケート結果の箱ひげ図

Q2 今回の学習方法をアルゴリズム理解の方法として今後も使いたいか？

Q3 今回の学習方法を他人に勧めたいか？

Q4 学習パートを終えたきっかけは何だったか？

以上の Q1 から Q4 について，バグ修正問題の場合とリーディングの場合の 2 つの場合についてそれぞれ被験者から回答を得た．なお，Q1 から Q3 は 1 から 5 の五段階評価で，1 が最も当てはまらず，5 が最も当てはまる，となっている．また，Q4 は自由記述とした．Q1 から Q3 の結果について，箱ひげ図で表したものが図 5 である．また，Q1 から Q3 の結果の平均値について，表 4 に示す．

Q1 から Q3 の結果について，正答率・学習時間と同様ウィルコクソンの符号順位検定を行なったところ，Q1 では p 値が 0.111，Q2 では p 値が 0.173，Q3 では p 値が 0.168 となった．平均値としてはバグ修正問題の方が各質問で高い点数を得ていたが，検定を行なった結果としては有意差は見られなかった．

また，Q4 に関しては，得られた回答のうち一部を以下にまとめる．

バグ修正問題 「正しくバグ修正が完了したから」（24 人）
バグ修正問題 「答えがわからなかったため」（6 人）
リーディング 「内容を理解したと感じたため」（25 人）
リーディング 「一通り目を通したため」（4 人）

6. 実験結果の考察

研究課題に関連付けて得られた結果について考察する．

RQ1 バグ修正問題によってアルゴリズムの理解度は向上するか

バグ修正問題後の確認問題とリーディング後の確認問題の正答数を比較した結果，有意差が存在することは確認できなかった．ただし，ハノイの塔に限って見れば平均値はバグ修正問題の方が正答数が良い結果になっており，p 値も 0.0725 と比較的 0.05 に近い値となっていた．多重検定とはなるが，A 群と B 群それぞれについて検定を行ったところ，A 群では p 値は 0.613，B 群では 0.0332 と，B 群のみ有意差が見られた．このことから，扱う問題によってバグ修正問題の効果に差が存在している可能性が示唆された．また，エラトステネスの篩に関しては，どちらの学習方法でも確認問題が満点となっていた人が多かったため，天井効果が発生して正しい結果が得られなかった可能性がある．また，別の見方としてはハノイの塔のリーディング時の正答数が著しく低いとみなすこともできる．これは，コードの分量が少ないために一読しただけでコードを理解した気になっているだけで，実際には理解があまり深まっていなかったことを示唆している可能性がある．

RQ2 バグ修正問題によって学習時間が伸びたかどうか

バグ修正問題の学習時間とリーディングの学習時間を比較したところ，バグ修正問題の方が有意に学習時間が長くなっていることが確認された．表 3 で平均学習時間がバグ修正問題の方が高いことを踏まえると，両群においてバグ修正問題の方がリーディングに比べて有意に学習時間が長いといえる．これによって，自律的な学習時間が伸びており，意欲を向上できた可能性がある．

RQ3 バグ修正問題について主観評価でどう思うか

学習方法に対して意欲的に取り組めたかどうかを尋ねた質問の結果では，表 4 にあるように平均値ではバグ修正問題の方が高評価を得ているが，検定の結果としては有意差

は見られなかった．同様に，アルゴリズム理解の方法として今後も使いたいかどうか，および学習方法を他人に進めいたいかどうかでも平均値はバグ修正問題の方が高評価だったが，有意差はないという結果となった．平均値はバグ修正問題の方が高かったが有意差が見られなかったため，誤差の可能性がある．

RQ4　何を基準にアルゴリズムの学習を終了するか

バグ修正問題においては，正しくバグ修正が完了したタイミングで学習を終了する被験者が最も多かった．よって，バグ修正問題ではバグの修正完了という明確な終了条件を持つことができたと考えられる．また，バグ修正問題では，答え，すなわちバグの箇所がわからなかったという回答もあったが，これは被験者がバグ修正に不慣れであったためと考えられる．リーディングにおいては，内容を理解したと感じたためという回答が最も多かった．ところが，実際に正答数をみるとリーディングの方がアルゴリズムの理解度は低い場合もあったため，あくまで主観評価による理解度であることに注意が必要である．次いで多かったのが一通り目を通したためという理由だが，これも理解度とは無関係であり，リーディングで適切な終了条件を設定するのがいかに難しいかを物語っている．

6.1　妥当性の脅威

被験者は早稲田大学内で広く募集する形をとったため，被験者ごとにプログラミングへの習熟度が異なっている可能性がある．習熟度だけでなく，バグ修正そのものへの経験や，扱ったアルゴリズムへの事前の理解度も異なる可能性がある．これは内的妥当性への脅威である．今後はこれらについてもアンケートで尋ねるようにしたい．

アルゴリズム学習から確認問題までの流れは連続して行われた．そのため，確認問題への慣れが生じた可能性がある．これも内的妥当性への脅威である．今後，扱うアルゴリズムの順番を変更するなどして影響の有無を確認したい．

また，外的妥当性の脅威として，本実験が本学の生徒に限って募集された点が挙げられる．今後，他の集団に対しても実験を行うことで，バグ修正問題の教育効果を検証したい．

7.　結論と展望

アルゴリズム学習において，アルゴリズムのコードを読むことが求められてきた．具体的には，アルゴリズムの解説資料とともに実装コードを読み，両者を照らし合わせながら理解を深めるといった方法である．しかし，ただ読むだけでは明確な終了条件が存在しない．これでは，完全な理解に至らないまま学習を終えてしまうほか，モチベーションの維持が難しくなることが考えられる．

そこで本論文では，バグ修正問題を解く手法を提案した．この手法は，アルゴリズム学習の際にコードリーディングをする代わりにバグ修正問題を解くというものである．これであれば，全てのバグ修正が終わったのであればある程度の理解度を担保することができる．加えて，終了条件が明確となることでゴールが存在し，学習意欲を維持しやすくなる．したがって，このバグ修正問題とリーディングでの学習の比較を行うシステム BFLS を実装した．そして，早稲田大学で広く被験者を募集する形で実験を行った．

結果として，バグ修正問題を解いたときの確認問題の正答数はリーディングの時よりも有意に高い場合が存在することが確認された．よって，バグ修正問題を解いたときのアルゴリズムの理解度は向上する可能性があることが示唆された．同様に，学習時間についてもバグ修正問題の方が有意に長くなっており，自律的な学習時間が伸びたと捉えることができ，意欲を向上できた可能性がある．一方で，主観評価での感想については有意差を見出すことができなかったが，平均値としてはバグ修正の方が良い結果となっていた．

展望としては，バグ修正の正答判定の手法の改善が挙げられる．現在は文字列としての比較を行い，バグ修正後のコードと正解コードの一致を判定しているため，別解が存在する場合を考慮していない．そのため，実際にコードをコンパイルしてテストケースを実行し，別解でも正解かどうかを判定する必要がある．他には，アルゴリズムの選定や確認問題の難易度調整が挙げられる．今回扱ったエラトステネスの篩とハノイの塔では，エラトステネスの篩の確認問題の方が簡単だった可能性がある．確認問題の難易度を調整するかアルゴリズムそのものを変更することで正答数の差が縮小するものと思われる．また，バグ修正問題を解くことができなかった人への対応も課題である．バグ修正問題を解くことができなかった場合，正しく理解できていない状況が続いてしまっていると言える．こういった人の学習効果を高めるために，修正場所のみの提示をするといった方法を検討する必要があるだろう．別の話題として，正答率が向上した要因はバグ修正そのものの効果によるものなのか学習時間が伸びたためなのかを確かめる必要がある点が挙げられる．バグ修正問題と同程度の時間をリーディングにもかけたらどうなるかを実験する必要がある．最後に，どういったバグをどういった場所に設定すれば学習の効率が上がるかを検討する必要がある．バグが小さく些細なものでは簡単に修正が終わってしまうが，あまりに大きな広範囲にわたるバグでは学びにつながらない可能性もある．アルゴリズムの本質的な部分にバグを埋め込むのが最適だと思われるが，どういった部分がアルゴリズムの本質かを判断する必要がある．この点に関しても今後の展望としたい．

参考文献

[1] Harrington, B. and Cheng, N.: Tracing vs. writing code: beyond the learning hierarchy, *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 423–428 (2018).
[2] Leppan, R., Cilliers, C. and Taljaard, M.: Supporting CS1 with a program beacon recognition tool, *Proceedings of the 2007 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pp. 66–75 (2007).
[3] Chou, C.-Y. and Sun, P.-F.: An educational tool for visualizing students' program tracing processes, *Computer Applications in Engineering Education*, Vol. 21, No. 3, pp. 432–438 (2013).
[4] 亀井亮汰，吉塚大浩，　篠埜功，古宮誠一：写経型学習の欠点を補う摂動を用いた理解度確認問題生成手法―二項演算子の事例に基づく有効性評価，コンピュータソフトウェア，　Vol. 38, No. 1, pp. 1_111–1_139 (2021).
[5] Jia, Y. and Harman, M.: An analysis and survey of the development of mutation testing, *IEEE transactions on software engineering*, Vol. 37, No. 5, pp. 649–678 (2010).
[6] Smith, R., Tang, T., Warren, J. and Rixner, S.: Autogenerating visual exercises for learning program semantics, *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 360–366 (2019).
[7] Fraser, G., Gambi, A., Kreis, M. and Rojas, J. M.: Gamifying a software testing course with code defenders, *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 571–577 (2019).
[8] Derezinska, A. and Halas, K.: Experimental evaluation of mutation testing approaches to python programs, *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, IEEE, pp. 156–164 (2014).
[9] Rojas, J. M., White, T. D., Clegg, B. S. and Fraser, G.: Code defenders: crowdsourcing effective tests and subtle mutants with a mutation testing game, *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, IEEE, pp. 677–688 (2017).
[10] Harrington, B. and Cheng, N.: Tracing vs. writing code: beyond the learning hierarchy, *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 423–428 (2018).
[11] Kumar, A. N.: Solving code-tracing problems and its effect on code-writing skills pertaining to program semantics, *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 314–319 (2015).
[12] Kumar, A. N.: A study of the influence of code-tracing problems on code-writing skills, *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pp. 183–188 (2013).
[13] 池田郁男：統計検定を理解せずに使っている人のために II，化学と生物，　Vol. 51, No. 6, pp. 408–417 (2013).