穴埋め問題を用いたプログラミング教育支援ツール pgtracer の運用実験

柳田崚[†1]　太田康介[†1]　大月美佳[†1]　掛下哲郎[†1]

我々は，学生のプログラム理解度やプログラミング能力を定量的に評価する教育支援ツール pgtracer を開発している．本ツールは Moodle のプラグインとして開発されており，プログラムやトレース表に対する穴埋め問題を学生に出題する．学生が解答すると，各問題の採点結果や学習時間などの学習ログ，学生が穴に入力した答案，時刻，正誤などの解答ログを個別に収集できる．本論文では pgtracer が半年間のプログラミング教育を受けた初学者の教育に有用かどうか評価するために運用実験を行った．運用実験の実施により得られたログを元に，学生の解答過程の分析，問題の難易度の定量的評価を行う．学生の解答過程の分析では，個人の成績に応じていくつかのパターンがあることがわかった．また，問題の難易度を定量的に評価するため，穴の種類ごとの所要時間を集計し，比較することで穴の種類に応じた難易度の違いを定量的に明らかにした．これを通じて，穴の重みを系統的に決定できる．

Operation Test of Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions

RYO YANAGITA[†1]　KOSUKE OHTA[†1]
MIKA OHTSUKI[†1]　TETSURO KAKESHITA[†1]

We are developing a programing education support tool pgtracer in order to evaluate program's understanding level and programming ability of students. pgtracer is developed as a Moodle plug-in, and make questions that some parts are masked in a pair of a program and a trace table. When a student answers a problem, pgtracer can collect study log (evaluation score and study time) and answer log (student's answer, answer process and correct answer). In this paper, we carried out an operation test of pgtracer in order to evaluate the usefulness of the tool for beginners after a half year programming education. We analyzed the collected log data. The answer process has typical patterns depending on the evaluation score. We clarified the difficulty of a mask for each mask type by aggregating and comparing the answer log. The result is useful to assign weight for each mask in a systematic manner.

1. はじめに

プログラミング教育は理工系の大学や高専において重要性が高いが，学生の学力低下に関する懸念や，プログラミング実習時に教員や TA 等だけでは十分な指導が行えない等の課題がある．学生は，授業や教科書を元に新しい知識を学習するが，復習する際には自身の能力にあった学習をすべきだろう．そこで我々は学生のプログラム理解度やプログラミング能力を定量的に評価するために，穴埋め問題を用いたプログラミング教育支援ツール pgtracer の開発・運用を行っている[1]．本研究では，穴埋め問題を出題し，学生個人やクラス全体の理解度や不得意箇所を把握することで，能力に応じた問題を出題する PDCA サイクルを繰り返し，学生の不得意箇所に対する重点的な教育を行う方針を採用する．そのためには，学生の理解度や不得意箇所が把握でき，それを元に適切な難易度の問題を作成し出題できる環境が必要である．

本ツールの問題形式はプログラムとトレース表を用いた穴埋め問題である．穴埋め問題を用いたプログラミング課題は，プログラムとトレース表が一貫した状態になるようにする作業を学生に課す．プログラムやトレース表に自由に穴を空けることにより，以下に列挙するように，穴を設定した要素に応じて学生の理解度を把握できる．

- トレース表の変数値に穴抜きすることで，変数値の変化を正しく理解していることを確認する．
- トレース表のステップ番号や変数名を穴抜きすることで，文の実行順序や対応する変数を正しく理解していることを確認する．
- プログラムの変数名，演算子，予約語等に穴を設定することで，初歩のプログラム作成能力を確認する．
- 式，文，複合文，ルーチン等に穴抜きすることで，より高度なプログラム作成能力を確認できる．

pgtracer を用いると，文単体を理解していれば解けるような易しい問題から，制御構造を正しく理解し実行順序が記述できるかを確認する問題，予約語やルーチン呼び出し等の知識が要求される問題，プログラムの記述能力やプログラム全体の理解が必要な問題まで様々な難易度の問題を作成できる．このように，穴抜きのパターンを変えた様々な問題を作成できることがプログラムとトレース表，そして穴埋め形式を用いた際に有用な要素であり，様々な問題を解かせることで，学生の理解度や不得意な部分を把握し，

†1 佐賀大学
Saga University

適切な難易度の問題を出題できる．

本論文では，プログラミングに関する半年間の講義・演習を受講した情報系学部 2 年の学生を対象として運用実験を行う．運用実験では，pgtracer を用いて，21 個のプログラム毎に4つの難易度を用意した計84個の問題を学生に解いてもらった．その過程で，学生毎の各問題の採点結果や学習時間，また学生が穴に入力した答案，時刻，正誤などのログを収集し，それを元に学生の理解度やプログラムの理解過程を分析・評価する．また，その分析・評価結果から適切な難易度の問題を作成し出題するために，穴埋め問題の難易度を定量的に評価する．

2. 関連研究

本ツールと同じようにプログラミング教育を行う上で，穴埋め問題を用いた研究はいくつか行われている．柏原らは，プログラム空欄補充問題を作成する際に「どこに」空欄を設定するかに焦点を置き，プログラムに内在する処理の流れの要所を PDG という手法により見つけ自動的にプログラムに空欄を設定し，その空欄の妥当性を評価している[2]．しかし，空欄設定における自由度は低く，様々な種類の問題を作成する点では不向きである．

また，菅沼らは学生の理解度と問題の難易度を動的に評価する練習問題自動生成システム AEGIS を開発している[3]．AEGIS では，学生の理解度を評価する際に，その学生が問題に回答する度に，学生の現在の理解度と問題の難易度,そして回答結果の得点を元に学生の理解度を算出する．しかし，これは学生に対して適切な難易度の問題を出題するための指標として用いており，具体的な理解傾向や不得意箇所を特定するといった目的には向いていない．

pgtracer は，プログラムのみでなくトレース表に対して自由に穴埋めを設定することができるため作成できる問題の幅が広いほか，学生が問題を回答する度に，採点結果や学習時間，また学生が穴に入力した答案，時刻，正誤等の多様なログを収集するため，学生の能力を分析するのに適している．その他，穴埋め問題を出題する研究のほとんどがプログラムに対して穴抜きを行うものであり，本ツールのようにトレース表に対しても穴抜きを行うシステムは知られていない．

3. プログラミング教育支援ツール pgtracer

3.1 概要

pgtracer は大学等で広く普及している e-Learning システム Moodle[4]のプラグインモジュールとして動作する Web アプリケーションであり，授業中の演習，低学力の学生を対象とした補習，学生の自習を支援するためのシステムである． 本ツールで使用する問題は，プログラムとトレース表に対する穴埋め問題であり，プログラムとトレース表本体の他に，穴埋め箇所を定義したプログラム用マスクとトレース表用マスクを作成し，これらを組み合わせて問題を構成するアプローチを採用する．プログラムやトレース表本体と穴抜きのマスク情報を分離することで，プログラムやトレース表の再利用，マスクの編集を容易に行うことができるほか,同じプログラムでもトレース表やプログラム用,トレース表用マスクを変更することで様々な難易度に設定でき，一つの問題について易しい問題から難しい問題まで用意することで学生は自分の学力に合った問題を選択・解答できる．

答案は問題の穴部分を選択することで自由に入力でき，学生が入力した答案，時刻，正誤などは穴のフォーカスが外れた際にログとして収集され，教員は収集したログを分析することで個々の学生や全体の理解度や不得意箇所を把握し，適切な難易度の問題を作成するなど，教育改善に役立てることができる．また，ログから学生が解答した順序や，それぞれの穴を解くのに要した時間を求めることもでき,学生のプログラムの理解過程を分析する際にも役立つ.

3.2 問題生成機能

教員は，pgtracer の問題作成機能で問題の作成を行う．本ツールで使用する問題はプログラム，トレース表，プログラム用マスク,トレース表マスクの情報を保持する XML ファイルの組み合わせで構成される．このうち，プログラムおよびトレース表の XML ファイルは pgtracer が自動生成する．一方，マスクを定義する XML ファイルは，プログラムやトレース表を表示した状態で編集する機能を提供している[5]．これにより，XML の知識がない教員でも，容易に問題を作成できる．

3.3 問題の出題機能

問題の登録・編集機能で登録された問題は，MySQL テーブルの情報をもとにテーマ，タイトル，難易度毎に一覧表示される．学生はテーマ，タイトル，難易度から自分の学力に合った,もしくは教員に指示された問題を選択する．それぞれの問題は学生が一度も受験したことのない問題ならば「未受験」と表示され，受験したことのある問題ならば「最高得点/満点」の形式で表示される．これらの文字列は解答画面へのリンクとなっており，解答したい問題のリンクをクリックすれば何度でも繰り返し受験できる．解答画面は選択された問題に登録された 4 種類の XML ファイルをもとに穴埋め問題を作成・表示する[6].

3.4 自動採点機能

学生が穴埋めを行った際の正誤判定は，プログラムないしトレース表の穴の XML ファイルから対応するノードの値を取得し，穴に入力された文字列と比較することによって行う．また、c=a*b が答えの穴に c=b*a と記述した場合など，学生の答えは意味的には間違っていないが正解の答えとは一致しないため不正解となってしまう．そのため，本ツールでは，学生が入力したプログラムをコンパイル・実行し，その実行結果と正解の実行結果の一致判定に基づ

いて採点を行う方法も実装している．

本ツールでは，問題は自習モードと試験モードに分類されている．自習モードの場合は，穴埋めを行った直後に正誤を判定し，穴の色で正誤の表現を行う．正解は黄緑で，不正解は赤で表す．不正解の穴にマウスカーソルを合わせると，吹き出しを用いて正解を表示する（図 1）．

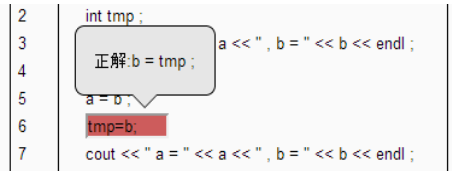

図 1 自動採点及び正解の表示

学生が解答を終え終了ボタンを押すと自習モードと試験モードともに採点結果画面（図 2）へ移動する．

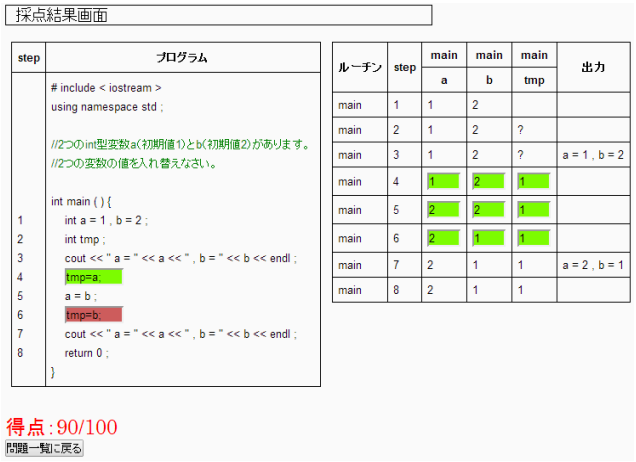

図 2 採点結果画面

採点結果画面では，最終的な解答画面の状態を元に採点を行う．合計点を出す際は正解した穴の重みの合計を求め，問題全体の重みの合計で割り，問題の配点に乗じて求める．重みは，その問題における穴の重要度であり，マスク作成時に設定される．

3.5 学習履歴及び解答履歴の収集機能

本ツールでは，学生が穴を埋める際や解答を終える際に，ログの収集を行う．学生が穴を埋めた際には，対象の穴の XPath 式，学生の入力文字列，正解文字列，正誤，入力終了時刻といった解答ログが解答履歴テーブル（表 1）に保存される．また，学生が問題の解答を終えた際には，ユーザ ID，問題 ID，得点，解答開始時間，解答終了時間といった学習ログが学習履歴テーブル（表 2）に保存される．

4. 運用実験

運用実験では，本学科で実施している 1 年間のプログラミング教育のうち，後半の授業を受けている学生 62 名を対象とし，自習モードと試験モードで計 21 項目を出題し，2 週間の期限を設けて取り組んでもらった．出題する問題 1

表 1 解答履歴テーブル(pgtracer_answer_log)

名前	データ型	説明
id（主キー）	int	解答履歴 ID
study_id	int	学習履歴 ID
blank_path	varchar	穴の XPath 式
answer	varchar	答案
correct_answer	varchar	正解の文字列
end_time	int	解答時刻

表 2 学習履歴テーブル(pgtracer_study_log)

名前	データ型	説明
id（主キー）	int	学習履歴 ID
user_id	int	ユーザ ID
question_id	int	問題 ID
point	int	採点結果
start_time	int	学習開始時刻
finish_time	int	学習終了時刻

項目は，4 つの難易度に分けられている．それぞれ，元となるプログラムは同じものであり，プログラムやトレース表に対するマスクの位置や大きさを調整することで難易度を定めている．難易度の設定は，問題を作成する側が難易度を主観的に定めているものである．今回の運用実験では，収集したログを元に難易度を定量的に評価することも目的としている．これは，学生の理解度に応じて，適切な難易度の問題を出題する上で重要な要素となる．また，自習モードと試験モードの双方を解いてもらうことで，両モードにおける学生の解答プロセスの分析とその比較を行う．

4.1 学生の利用状況

今回の運用実験における学生の利用状況を把握するため，学生がどれだけの時間 pgtracer を利用し問題を解いたのか，また問題毎に何人の学生に解答してもらったのか集計を行った．

利用時間を分単位で区切り，利用時間毎の学生数を集計したところ図 3 のようになった．60~240 分利用している学生が集中しており，多く問題を解いている学生に関しては 300 分以上 pgtracer で学習を行っている学生も見られた．また，問題毎に解答者数を集計した（表 3）．

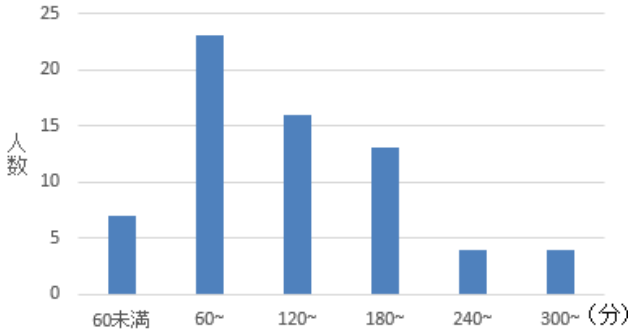

図 3 利用時間毎の学生数

表 3 の結果では，試験モードと自習モードの各問題の解答数は十分な数得られている．試験モードに関しては，4 つの難易度全てで解答が得られているため，難易度毎の分析も行うことができる．その他にも，問題毎の分析や，学

表 3　問題毎の回答学生数

	問題番号・タイトル		難易度			
			1	2	3	4
自習モード	02	配列	45	45	45	62
	06	条件分岐(1)	42	32	25	58
	09	繰り返し(1)	20	19	18	58
	11	入出力	19	19	17	58
	13	条件分岐(2)	17	18	16	57
	14	switch 文(1)	17	17	16	57
	16	多重ループ	17	16	16	57
	18	while 文(1)	17	15	16	57
	23	実数型	15	15	15	56
	25	実数型配列(1)	15	14	15	56
試験モード	04	整数型	48	46	46	64
	08	繰り返し(2)	46	45	45	64
	10	while 文(2)	46	45	45	64
	12	条件分岐(3)	50	48	47	64
	14	switch 文(2)	46	45	45	64
	15	while 文(3)	46	45	45	64
	17	条件分岐(4)	45	45	45	64
	21	代入文	45	45	46	64
	24	入れ子 if 文	49	47	47	63
	29	実数型配列(2)	45	45	45	64
	32	ユークリッド互除法	45	45	45	62

生の解答過程にも着目して分析を行っていく．

4.2　問題毎の学生の理解状況

問題毎の学生の理解状況を分析するために，学習ログから問題毎の平均点を求めることで，問題毎の出来・不出来を把握する（表 4）．学生は，同じ問題について何度も解答を行っていることも考えられるため，平均点を求める際には，学生が最初に問題を解いた際の得点を用いることで集計を行った．これは，学生が同じ問題について何度も解答を行い，理解を深める前の素の能力を評価するためである．

表 4 を見ると，自習モードに比べて試験モードの問題の平均得点が低いことが分かる．自習モードでは，解答時に学生が穴を埋めた際に，該当する穴の正誤と正解を表示するため，学生が誤った解答を入力した際に，正しい答えに修正を行ったからであることが解答ログより分かった．また，表 4 の試験モードの問題を見ることで，学生がどの問題を苦手としているか，また得意としているかを把握できる．今回の運用実験で最も出来が悪かった問題は，問題 34 難易度 4 のユークリッド互除法（表 3 参照）のプログラムを用いた問題であり，学生は while 文の条件式や文内での変数の扱い方がわかっていないようだった．一方で，出来が良かった問題は，問題 10，問題 12，問題 21 の 3 つであった．その中で，問題 10 は while 文を用いた問題であり出来が悪かった問題 32 と単元で一致する．そこで，問題の構造を見たところ，問題 10 はトレース表の要素を中心に穴抜きが行われているのに対して，問題 32 はプログラムの要素を中心に穴抜きが行われていた．よって，学生は while 文について実行順序や変数値の変化は正しく理解しているが，そこからプログラムを書くことができないと判断できる．

表 4　問題毎の平均得点

	問題番号	難易度				
		1	2	3	4	平均
自習モード	問題 02	86.29	98.56	99.12	79.31	90.82
	問題 06	99.00	98.95	100.00	93.53	97.87
	問題 09	94.74	99.47	100.00	92.64	96.71
	問題 11	100.00	100.00	100.00	92.95	98.24
	問題 13	98.59	99.71	100.00	94.12	98.11
	問題 14	100.00	90.88	100.00	89.82	95.18
	問題 16	97.82	98.40	93.75	94.46	96.11
	問題 18	100.00	100.00	95.07	100.00	98.77
	問題 23	99.60	100.00	90.00	92.73	95.58
	問題 25	100.0	100.0	100.0	97.1	99.28
	平均得点	97.60	98.60	97.80	96.67	96.33
試験モード	問題 04	90.88	93.91	97.39	72.81	88.75
	問題 08	88.24	94.00	88.20	66.06	84.13
	問題 10	93.78	90.02	100.00	76.30	90.03
	問題 12	92.50	92.60	96.43	78.97	90.13
	問題 14	73.35	86.98	88.11	76.17	81.15
	問題 15	93.24	92.18	89.78	78.23	88.36
	問題 17	92.78	89.64	88.78	81.70	88.23
	問題 21	95.73	93.04	87.76	88.19	91.18
	問題 24	87.35	85.43	87.49	70.54	82.70
	問題 29	87.00	94.22	85.53	70.03	84.20
	問題 32	82.78	85.98	88.22	60.00	79.25
	平均得点	88.88	90.73	90.70	74.45	86.2

以上のように，各問題を単元ごとに作成すれば，学生が何を理解し何を理解していないか，表や問題の構成から読み取ることも可能である．

上で示した表 4 では，初回時の回答結果から得点を集計したが，学生によっては，良い点数がとれるまで，繰り返し同じ問題を解いているケースもある．そこで，問題毎に各学生の初回時の得点と最終時の得点を比較し，どの程度違いがでるか分析を行った．

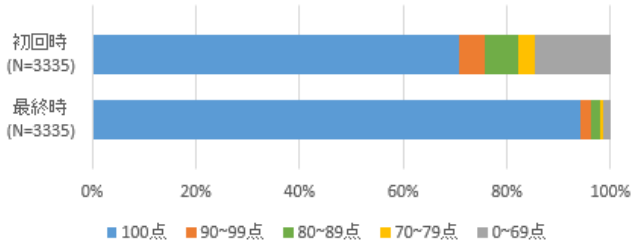

図 4　問題毎の各学生の初回時と最終時の得点分布

図 4 を見ると，100 点をとった学生が初回時には 7 割程度だったが，最終的には 9 割以上に上昇している．また，初回時 70 点を下回っていた学生が，最終的には減少している．今回の運用実験において，学生が意欲的に pgtracer を利用し，自主学習に取り組んだことが確認できる．

4.3　問題の難易度と解答時間の関係性

問題毎に学生らが解答に要した時間の平均値を出すことで，難易度と解答時間の関係性について分析する．問題の解答所要時間は，学習ログの解答開始時間及び解答終了時間によって求める．

表 5　問題毎の解答所要時間

	問題番号	難易度			
		1	2	3	4
自習モード	問題 02	221.89	165.76	227.44	303.80
	問題 06	61.10	46.62	95.72	103.66
	問題 09	134.95	51.80	60.53	159.06
	問題 11	113.94	139.67	123.06	208.69
	問題 13	98.28	130.11	93.38	205.21
	問題 14	64.95	69.33	103.29	297.83
	問題 16	273.76	140.69	86.94	227.38
	問題 18	112.80	111.07	111.31	117.54
	問題 23	112.17	109.00	73.88	270.94
	問題 25	245.19	139.71	150.80	180.90
	平均時間	143.90	110.38	112.64	207.50
試験モード	問題 04	68.18	34.68	32.51	108.25
	問題 08	46.61	36.87	49.13	99.66
	問題 10	108.75	163.07	26.52	46.56
	問題 12	77.02	78.42	82.98	108.94
	問題 14	104.74	146.38	120.13	147.13
	問題 15	127.46	79.82	126.17	176.59
	問題 17	83.29	72.59	136.32	218.96
	問題 21	101.16	63.32	46.36	84.52
	問題 24	84.47	140.03	149.76	218.98
	問題 29	97.50	89.96	133.75	164.56
	問題 32	71.00	72.47	103.78	87.45
	平均時間	88.20	88.87	91.58	132.87

表 5 を見ると，まず試験モードに比べて自習モードの方が解答に要した時間が長い．これは自習モードでは学生が間違えると正解が表示され学生は正しい答えを入力することができるが、試験モードではある程度考えてわからなければ解答を終え，正解を見るためだと思われる。自習モードでは以上のようなことが考えられるため，ここでは試験モードに含まれる問題について考える．

難易度毎の所要時間の平均を見ると，難易度 1~3 については大きな差はないが，難易度 4 になると 132.87 秒と難易度 1~3 に比べておよそ 40 秒の時間を要している．同時に表 4 について見ると，難易度毎の平均得点は難易度 1~3 に関しては大きな差はなく，難易度 4 に関してのみ 74.45 と平均得点が低かった．また，試験モード全体で平均得点が 90 点以上である問題の平均解答所要時間は 78.38 秒であり，80 点台の問題は 105.55 秒，80 点未満の問題は 126.29 秒であった．

以上のように，各問題の平均得点には考える時間（解答時間）が影響している．よって，問題の難易度を上げる場合には，より長い時間学生に考えさせるような問題を作れば良い．問題を 4 つの難易度に分けるのは，プログラムやトレース表に対して行う穴抜きの位置や大きさである．よって，この穴抜きの何の要素が解答所要時間に影響を与えているのか導くことができれば，問題の難易度を定量的に評価することも可能である．問題の難易度の定量的評価については 6 節で議論する．

5.　学生の解答過程の分析

pgtracer では，収集したログを元に，学生の解答過程を見ることができる．学生一人ひとりの解答過程を把握することで, 学生個人に対して適切な指導を行うことができる．実際に，問題 15（図 5）を解いた学生の解答過程について分析を行う．

解答過程を分析する際には，穴ごとの解答時間と解答順の観点から分析する．分析方法としては，解答履歴に基づいてガントチャートを作成し，それを分析する．ガントチャートでは横軸を穴の解答時間，縦軸を穴の解答順として作成することで，プログラミング問題の解答過程が可視化できる．また，より具体的な解答過程を分析するために，今回，対象となる問題（図 5）に対して，穴の種類ごとに色分けし，学生が問題を解く中で，どこに着目しているのかを把握できるようにした．プログラムに対する穴をオレンジ，トレース表に対する穴については，ステップ番号を赤，変数値を青，出力値を緑で表す．

step	プログラム

	# include < iostream >
	using namespace std ;
	
	// キーボードから整数値が何度も入力されます。入力された値が7の倍数ではなければ
	//「7の倍数ではありません」と表示し、次の入力を待ちなさい。入力された値が7の
	// 倍数であれば「7の倍数です」と表示してプログラムを終了しなさい。
	
	int main ( ) {
	1　int number ;
	2　①
	3　②
	3.1　cout << " 7の倍数ではありません " << endl ;
	3.2　③
	}
	4　cout << " 7の倍数です " << endl ;
	5　return 0 ;
	}

ルーチン	step	main number	出力
main	1	?	
main	2	1	
main	3	1	
main	④	⑤	⑥
main	3.2	3	
main	3	3	
main	⑦	⑧	⑨
main	3.2	5	
main	3	5	
main	⑩	⑪	⑫
main	3.2	7	
main	3	7	
main	⑬	⑭	⑮
main	5	7	

図　5 試験モード・問題 15（レベル 4）

図 6-8 は，それぞれ，問題 15・レベル 4 を解いた学生に見られた代表的な解答過程である．

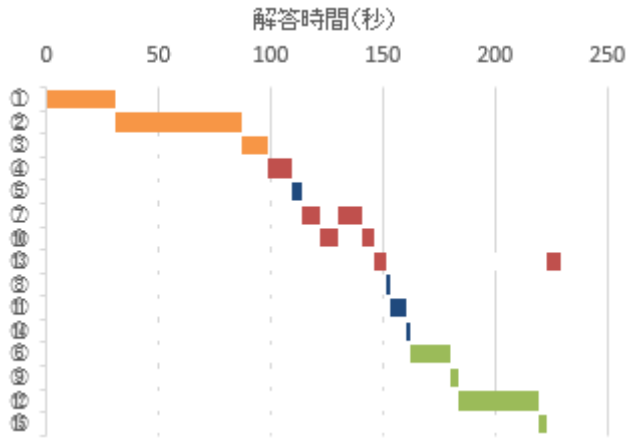

図 6 代表例 1（53 点）

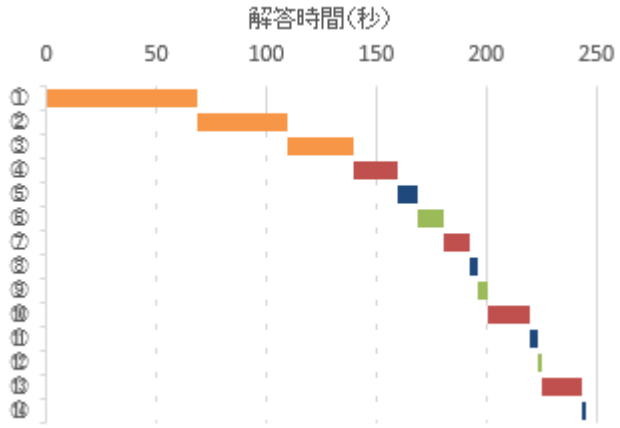

図 7 代表例 2（73 点）

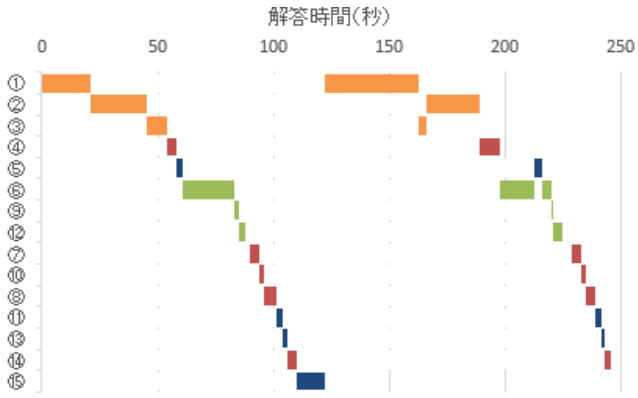

図 8 代表例 3（100 点）

これら 3 つの解答過程に共通しているのは，3 人ともまずプログラムの穴に対して解答し，プログラムを完成させた後にトレース表の解答に移っている点である．また，最初に解答を行った穴の所要時間を見ると，比較的長い時間を要している．この 3 つのガントチャートに限らず，この特徴はほとんどの学生に対して見られた．これについて考えられるのは，学生が解答を初めてから，まず問題全体を見通しているということである．

次に，各ガントチャートについて見ていくと，トレース表に対する解答過程に違いが現れていることが分かる．図 6 および図 8 の学生は，解答順で色の変化を見ると，一つの色についてまとまって解答を行っている．これは，この学生がトレース表を縦に解いている，つまりステップ番号や変数等の要素毎に考えて解答していることを表している．また，図 7 の学生について見てみると，解答順の色の変化は，赤，青，緑のペアが何度も現れている．これは，トレース表を横の要素で考え穴を埋めている，つまりステップ毎に考えているということを表している．

また，図 8 の解答過程は他の 2 つと比べると大きな違いが見られる．図 8 では，一度穴に対する解答を終えた後に，もう一度全ての穴を見ていることが分かる．これは，この学生が一度解いた後に見直しを行っていると考えられる．図 8 では，最初の穴から順番に選択して見直しを行っているが，中には，最後に解答を行った後，カーソルを移動せずに見直しを行い，既に入力した穴を書き換える場合も考えられる．そのため，見直しを行っているガントチャートには，最後の穴に対する所要時間が長いものや，穴に対する解答を一旦終えた後で，解答を書き換えているものも含まれる．

6. 問題の難易度の定量的評価

pgtracer では，学生が自身の理解度に応じた難易度の問題を解くことで，効率的に学習できる環境が必要である．現状で行っているような主観的な難易度の設定では，学生が実際に感じる難易度とのギャップが生じ，作成した問題が意図した難易度になるか明確でない．そこで，今回収集した学生の解答ログから，問題の難易度を解答所要時間によって定量的に評価する方法を提案する．

pgtracer で出題する問題において，難易度に関係する要素としては，主にプログラムやトレース表に対して行う穴抜きの位置や種類が考えられる．問題に利用される元プログラムの内容も重要だが，マスクの位置や大きさによっても難易度は変化するだろう．そこで，我々は pgtracer で設定できるマスクの種類を以下に示した表 6 の大・中・小の 3 階層の区分で分類することで，種類毎に学生がどれだけの時間を要して解答を行ったか分析を行い，解答所要時間の長さによって穴の難易度を評価する．

大区分において，プログラムとトレース表という分類を行う．中区分において，トレース表の変数値については，直前の変数値と比較して値が変化していた場合には変更あり，そうでない場合には変更なしとしている．また，ステップ番号については，繰り返しや条件分岐など，次のステップ番号が連続していない場合には不連続，ステップ番号が連続している場合は連続としている．

以上の分類に基づき，収集した解答ログから穴の種類毎に解答所要時間の平均（以下の「解答所要時間」は全て平均とする）と標準偏差を求めた（表 6）．また，成績が良い学生と悪い学生で解答所要時間の違いを比較するために，

表 6 穴の種類毎の解答に要した時間の平均

大区分	中区分	小区分	全体(N=2024)			上位層（N=1442）（90 点以上）		下位層（N=306）（70 点未満）		有意差
			解答所要時間（秒）	標準偏差	有意差	解答所要時間（秒）	標準偏差	解答所要時間（秒）	標準偏差	
トレース表	変数値	変更なし	5.34	11.7	○	4.66	10.6	9.65	18.08	○
		変更あり	9.14	28.7		8.57	28.1	9.71	25.10	×
	ステップ番号	連続	5.04	9.88	○	4.36	7.74	8.87	16.98	○
		不連続	7.41	10.0		6.87	9.53	12.29	12.78	○
	変数名		11.55	77.7		12.92	84.3	2.50	2.33	×
プログラム	トークン単体		9.75	27.0	※	9.21	28.7	14.07	21.31	○
	トークン列		19.45	49.7		18.34	43.3	16.42	29.52	×
	文全体		22.24	50.6		20.96	30.8	26.50	89.23	×
問題全体	最初の穴		37.40	67.8		35.93	72.0	41.06	48.41	
	最後の穴		9.92	13.1		9.40	13.2	13.28	12.85	

得点が 90 点以上であった上位層と 70 点未満であった下位層に分けた場合の集計も行っている．ただし，対象となる解答ログは，試験モードの問題に対して学生が初めに取り組んだ際のものとする．また，学生が 1 つの答案の中で同じ穴に対して何度も入力を行っていた場合には，その穴についての理解に要した時間は，その穴に入力に要した時間の合計だと考え 1 つにまとめる．

6.1 穴の種類毎の解答所要時間における有意差の検定

表 6 で示された穴の種類同士の解答所要時間の平均値を比較する場合には，比較する 2 つの解答所要時間に有意な差があるか示すために t 検定を行う．例えば，中区分の変数値において小区分の変更なしと変更ありとの間に有意差があるか t 検定を行った場合の検定結果を表 7 に示す．

表 7　t 検定結果値

	変数値（変更あり）	変数値（変更なし）
平均	9.145	5.335
分散	828.5	137.3
観測数	3549	4049
仮説平均との差異	0	
自由度	4569	
t	7.368	
P(T<=t) 片側	1.023E-13	
t 境界値 片側	1.645	
P(T<=t) 両側	2.046E-13	
t 境界値 両側	1.960	

表 7 から，t 値が t 境界値 両側よりも大きいことが確認できるため，帰無仮説が棄却されこの 2 つの集団間には有意な差があるといえる．また，P(T<=t) 両側の値が非常に小さな値であることから，いわゆる 1%水準で有意差があるといえる．以上の方法で表 6 の「全体」欄で他の組み合わせについても同様の検定を行い，有意差があるものは○，そうでないものは×で示している．なお，プログラムを構成するトークン単体，トークン列，文全体の解答所要時間については，トークン単体の解答所要時間と，他の 2 種類の解答所要時間には有意差があるが，トークン列と文全体の解答所要時間には有意差があるとは言えなかった．

6.2 穴の種類毎の比較

解答所要時間について大区分で比較すると，トレース表に対する穴よりもプログラムに対する穴の解答所要時間の方が長くかかっている．このことから，プログラムの理解よりもプログラミングの難易度が高いことが分かる．

トレース表に関しては，変数値とステップ番号の解答所要時間には違いは見られず，最も時間を要しているのは変数名となっている．

トレース表に設定した小区分に関しては，変数値の変更の有無及び，ステップ番号の連続と不連続の間で解答所要時間に有意差がある．変数値の変更の有無による差は，変更なしの場合には値の同一性を認識するステップのみ必要なのに対して，変更ありの場合には，これに加えて変更後の値を認識する必要があるからだと考えられる．ステップ番号の連続と不連続についても同様のことが言え，ステップ番号が不連続の場合には条件分岐やループ判定等を考えるステップが余分に必要となる．このように，解答者に求められる要素が多くなれば解答所要時間は長くなるため，有意差がある場合には難易度の違いがあると考えられる．

一方，プログラムに関しては，穴の大きさに応じて解答所要時間が長くなる．プログラムのトークン単体の穴とトークン列及び文の穴への解答に要する時間にはおよそ 2 倍の差がある．これは，学習者に与えられる情報量が穴の大きさによって変化するため，穴が大きくなるにつれて，学習者に求められる知識や能力が増えるためだと考えられる．

標準偏差については，プログラムに関する穴の種類に対して大きな数値が確認でき，データにばらつきがあることが分かる．これは，学生個人の能力によって，解答に要した時間が変化し差が生じるためだと考えられる．標準偏差の高い穴は，学生の能力の差を評価する上で効果的だと考

えられる．

6.3 上位層と下位層との比較

90 点以上の答案の解答ログについて集計した上位層のデータと 70 点未満の答案の解答ログについて集計したデータを比較することで，成績が良い場合と悪い場合での各穴の種類について解答所要時間がどのように変化するのか分析する．表 6 の右端の有意差欄には，こうして求めた t 検定の結果をまとめている．

上位層と下位層で有意な差が見られたのは，変数値の変更なし，ステップ番号の連続，不連続，トークン単体である．これらの穴の種類を見ると，下位層は上位層に比べて解答所要時間を多く要している．成績が悪い場合には解答所要時間が長くなる，つまり成績の悪い学生は，成績の良い学生に比べて解答所要時間が長くなる傾向があるということになる． 具体的に，変数値の変更なしについて見てみると，上位層では変更有りと変更なしで明確な差が出ていたのに対して，下位層では変更なしの解答所要時間は変更ありの場合と大差ない程まで長くなっている．6.2 節でも述べたように，変数値の変更ありは変更なしに比べて変化した値を認識する必要が有るため解答所要時間は長くなるはずである．しかし，下位層においては変更ありの場合と同程度の時間を要している．これは，上位層の学生は変数の値が同じかどうか判断できるが，下位層の学生は同じかどうかの判断をすることができず，変更ありの場合と同じ 2 ステップを行っているためだと考えている．

6.4 問題を構成する穴の種類と難易度

表 4 に示す試験モードの全ての問題に対して，難易度レベル毎に問題を構成する穴の種類を集計して比較を行った．表 8 では，その中の 1 つ（問題 15）を示す．

表 8 問題 15 における難易度レベル間の比較

問題 15	難易度			
	1	2	3	4
変数値（変更なし）	10		6	6
変数値（変更あり）	2		2	2
ステップ番号（連続）		10		
ステップ番号（不連続）		4	4	4
変数名		1		
トークン単体			5	
トークン列			1	
文全体				3
平均得点	93.24	92.18	89.78	78.23

表 8 を見ると，平均得点が高い難易度 1，難易度 2 については表 6 で解答所要時間が少なかった変数値やステップ番号に対する穴を中心に構成されている．一方，平均得点が低い難易度 3, 4 は解答所要時間を多く要するプログラムに対する穴が増えている．また，難易度 3 と 4 においても平均得点に差があることから，6.2 節で有意差を示したトークン単体と文には難易度の違いがあることが分かる．この傾向は他の問題でも見られ，設定された穴の種類には，問題の平均得点, 難易度との関連があることが確認できた．

現在は穴の重みをおおむね 1 としているため，どのような穴の種類であっても，誤った解答をした場合の減点はほぼ同じである．しかし，今回，穴の種類による難易度の違いを明らかにすることができたため，それに応じて穴の重みを定めることで，穴に対して適切な配点を与えることができる．また，問題に含む穴の種類によって問題の難易度を定量的に定めることが可能となるため，今後，pgtracer による教育効果を更に高めるために活用したいと考えている．

7. まとめと今後の課題

本稿では，運用実験により収集した学生の学習ログや解答ログ，アンケートを元に分析を行った．学生個人の解答過程の分析では，学生の解答方法にいつくかのパターンがあることが分かった．解答ログからガントチャートを作成することで，学生が問題全体を把握するまでにどれだけの時間を要したのか，どのような順序で解答したか，最後にどれだけ見直しを行っているか等を具体的に把握できる．また，穴の種類に応じた難易度の違いを，穴の種類ごとの平均所要時間から定量的に評価した．

今後は，今回行った分析の他にも，様々な視点から学生の理解度や理解傾向，解答過程を分析する予定である．また，今回の運用実験によって，問題の難易度を定量的に評価できるようになったため，新たに問題を作成し，学生に対して適切な難易度の問題を出題できる環境を整えたい．

本運用実験や教員・学生による pgtracer のレビューを通じて多数のコメントを収集したので，現在，pgtracer の機能改善を行っている．また，運用実験の中で学生を対象としたアンケート調査も行ったため，収集したログと併せて分析を行っている．これらのトピックについては，稿を改めて報告する予定である．

参考文献

[1] 掛下哲郎，大月美佳 : 穴埋め問題を用いたプログラミング教育支援ツールの全体構想，平成 25 年度電気関係学会九州支部連合大会 11-2p-01

[2] 柏原昭博，久米井邦貴，梅野浩司，豊田順一 : プログラム空欄補充問題の作成とその評価，人工知能学会論文誌，16 巻 4 号 pp384-391(2001)

[3] 菅沼明，峰恒憲，正代隆義 : 学生の理解度と問題の難易度を動的に評価する練習問題自動生成システム AEGIS，情報処理学会研究報告. 2003(11), 25-32, 2003-01-31

[4] Moodle.org，https://moodle.org/

[5] 柳田崚，太田康介，掛下哲郎，大月美佳 : 穴埋め問題を用いたプログラミング教育支援ツール pgtracer における教員用機能の実装，情報処理学会 コンピュータと教育研究会，2014 年 3 月

[6] 太田康介，柳田崚，掛下哲郎，大月美佳 : 穴埋め問題を用いたプログラミング教育支援ツールの概要と学生用機能の実装，情報処理学会　コンピュータと教育研究会，2014 年 3 月