アルゴリズムとデータ構造に関する コードデータのアノテーションとその分析

香取 浩紀[1]　山本 恒輔[1]　佐藤 安理紗 ジエンジエラ[1]　矢谷 浩司[1]

概要：アルゴリズムとデータ構造の学習は, 情報教育の 1 つとして欠かせないものである. これまでに, アルゴリズムとデータ構造に関して体系立てて説明した書籍や教材, 研究などは存在する. アルゴリズムとデータ構造の知識がある人がソースコードレベルで重要な行とその理由について述べることはできるが, 複数人への調査を経て, 一般に高等教育などで教えられるアルゴリズムとデータ構造に関してアノテーションされたものは少ない. そこで, 本研究では, アルゴリズムとデータ構造の問題と解答ソースコードを 30 問用意し, 各問題において, そのソースコードの中で重要な行を選び, その理由を書いてもらうというアンケート調査をアルゴリズムとデータ構造の知識およびプログラミング経験のある 11 人に対して実施した. さらに, 著者ら 2 人で, 14 個のラベルを作成し, 回答で得られた全ての理由データに対してラベリングを行った. 本稿ではそのデータ制作過程を述べるとともに, プログラミング穴埋め問題を作成する際のデータとして使用するなどのコードデータの活用法について述べる.

Annotations and Analysis of Code Data Regarding Algorithms and Data Structures

HIROKI KATORI[1]　KOSUKE YAMAMOTO[1]　ARISSA J. SATO[1]　KOJI YATANI[1]

1. はじめに

2025 年度の大学入学共通テストより新教科「情報」が国立大学では原則として受験必須となることをはじめ, 近年日本においては, 情報教育の重要性がますます高まっている. 情報系科目においては, 情報理論や情報セキュリティ, プログラミング, アルゴリズムとデータ構造など様々な分野が存在し, アルゴリズムとデータ構造は, 効率的で正しいプログラムを記述する際に重要である. アルゴリズムの学習においては, 知識の習得はもちろんのこと, 知識をプログラムとして実装する能力を得る必要がある.

これまでのアルゴリズムとデータ構造に関する研究では, アルゴリズムのデータ構造の学習におけるツールや手法について調べた研究が挙げられる [2,4,7,10]. また, アルゴリズムとデータ構造を体系立ててソースコードとともに説明した書籍も存在する [13,17]. 一方で, アルゴリズムとデータ構造のソースコードのレベルにおいて, 定量的にどこが重要であるのか, 定性的になぜそこが重要なのかを包括的に調査した研究は少ない.

本研究では, アルゴリズムとデータ構造の問題 30 問とその解答ソースコードを用意し, アルゴリズムとデータ構造に関して十分な知識と実装力を有する 11 人に対してアンケート調査を行うことで, ソースコードの行レベルで具体的にどこが重要なのか定量的に明らかにした. また, アンケート回答者には, 重要である行の理由も記述してもらい, 適切に作成したラベルを各理由データに対して付与することで, アルゴリズムとデータ構造に関するアノテーション付きのコードデータを得ることができた.

上記の調査実験の結果では, 問題によって重要な行が特定の行に集中したり, 複数の行に分散していた. また, 理由データと合わせると, 回答者は適切な理由を持って重要な行を選んでいることがわかり, データとしての十分な信頼性が担保されていることがわかった. 理由のラベリングでは, 合計 14 個のラベルを作成し, 全体に占める各ラベルの割合についても適度に分散していた. さらに, 本研究の結

[1] 東京大学 Interactive Intelligent Systems Laboratory

果をもとにした応用の可能性についても検討した.

2. 関連研究

2.1 アルゴリズムとデータ構造の学習

高等教育レベルでのアルゴリズムとデータ構造の教育においては, アルゴリズム教育とプログラミング教育を別々で行うか否かで分けることができる [14]. 別々で教える場合は, 川口 [15] のアルゴリズムの処理手順全体のイメージを用いて教える手法などがある. 一方で, 本研究ではアルゴリズムとデータ構造のコードデータの収集を行うので, アルゴリズムとプログラミングを同時に教える方法に着目すべきである. この教え方に関連する研究では, 浅野ら [16] のアルゴリズムをスワップなどの基本操作の集合として捉えてアルゴリズムの問題を考えるように教えるという研究や, 兼宗ら [12] による, 入門用アルゴリズムの難易度や理解度に関する調査研究などがある.

また, アルゴリズムとデータ構造の学習においては, 様々な手法やツールが開発・研究されてきた. Rebanal ら [7] は, アルゴリズムに関する質問に回答するインタラクティブなシステムを開発することを目指し, ユースケースとして, クイックソートに関する学生の質問に対する回答を自動生成する Xalgo というシステムを作成した. Choi ら [2] は, アルゴリズムとデータ構造における subgoal 学習 (細かい部分に区切って目標を設定する [1]) において, Learnersourcing(学習者が自らも学習活動によるメリットを享受しながら, 活動における教材の構築や改善に貢献するというもの [10]) を用いた. Halim [4] は, 様々なアルゴリズムを可視化する VisuAlgo という Web アプリケーションを開発した.

このように, アルゴリズムとデータ構造における学習ツールや教育方法, またアルゴリズムとデータ構造を体系的にまとめた教科書などの書籍は多くある. その一方で, ソースコード上で具体的にどこがなぜ重要かを明らかにした研究は少ない.

2.2 プログラミング教育における穴埋め問題に関する研究

本節では, プログラミング穴埋め問題のシステムの先行研究や機能例の紹介, および穴の位置を自動で選択する手法に関する先行研究について述べる. プログラミング穴埋め問題とは, 完成されたソースコードの中から文やトークン列, トークンなどを 1 つ以上選択して穴とし, 学習者はその穴を開けた部分に当てはまるものを埋めるというものである. プログラミング穴埋め問題の学習効果としては, プログラミングの知識定着の促進 [3] や, 問題文の指示を把握してソースコードを読むことを通じソースコードの挙動を想像しその動作を理解する力の向上への寄与 [18] といったことが挙げられている.

これまで, プログラミング穴埋め問題システムに関した研究は複数なされてきた. Shinkai ら [8] の教員が作成したプログラミング穴埋め問題を学生に共有し, 学習者の理解度などの情報を提示するシステムを検討した. Hara ら [5] は, C 言語のプログラミングにおいて, 学習履歴をもとに学習者個々の苦手な箇所を分析し, ソースコード上で穴を開ける箇所を自動選択する穴埋め問題システムである PROVIT を開発した. 掛下ら [11] は, C++言語のプログラミングにおいて, ステップごとの変数の値の変化のトレース表も学習者に埋めさせることで, よりソースコードの動作の理解を促進する穴埋め問題システム pgtracer を開発した.

また, プログラミング穴埋め問題の難易度の調整やプログラムの動作理解能力の促進のために, ソースコードのどの箇所を穴とするかを決定する手法は大切である. ルールベースの手法として, 柏原ら [18] の Program Dependence Graph(PDG) を用いた処理の要所を発見して, そこを穴とする手法がある. 他にも, 自動で穴を決定して穴埋め問題を生成するシステムとして, Java 言語の予約語に注目した Funabiki ら [3] の JPLAS や, Python での解答の一意性と解答可能性が保証される条件を考察した Hnin ら [6] の PYPLAS などが挙げられる. Terada ら [9] は, 穴埋め問題に使用するソースコードと穴の位置を自動選択する手法として, オンラインジャッジシステム上の問題と解答のソースコードを学習データとした CRF 付き Bi-LSTM を用いた手法を提案した.

3. データ収集

3.1 アルゴリズムとデータ構造の問題と解答ソースコードの準備

プログラミングコンテスト向けのアルゴリズムとデータ構造が体系的にまとめられた解説書 2 冊 [13,17] と 2021 年度の本学工学部電子情報工学科のアルゴリズムの講義内容に基づいて取り扱うアルゴリズムとデータ構造を決定した. 次に, 問題の選定においては, AtCoder [*1]及び AOJ[*2]にある問題を引用し, 一部は著者が自作した. 採用した問題の多くは, 問われているアルゴリズムとデータ構造を実装することに主眼を置いた問題である. また, それらの問題のうち一部は, 複数のアルゴリズムとデータ構造の知識を組み合わせて解くものや, どの知識を使用するかについて考察を要するものである. このようにして, 幅広い分野からアルゴリズムとデータ構造の問題を 30 問用意し, 各問題について 1 つの Python の解答ソースコードを用意した. 解答ソースコードは, 著者が書いたものである. 表 1 において, 30 問の問題の番号, 解答ソースコードの行数, 問題の出

[*1] 競技プログラミングサイトの 1 つであり, 様々な難易度に富んだ数千問以上の問題がある. オンラインジャッジシステムを備えている. (`https://atcoder.jp/`)

[*2] AIZU ONLINE JUDGE. 多数のプログラミング問題があるオンラインジャッジシステム. アルゴリズムとデータ構造の基本的な練習問題が多数存在する. (`https://judge.u-aizu.ac.jp/onlinejudge/index.jsp`)

典, 使用されているアルゴリズムとデータ構造をまとめて記載する. なお, 30 問の問題を選定した後, 問題番号はランダムに割り当てた.

表 1: 問題と解答ソースコードのリスト.

番号	行数*	問題の出典	内容
1	21	AtCoder	尺取り法
2	35	AOJ	一点加算 Range Sum Query セグメント木
3	14	AOJ	計数ソート
4	46	AOJ	二分探索木 (挿入, 中間・先行順巡回)
5	24	AOJ	深さ優先探索 (再帰での実装, 行きがけ・帰りがけ順)
6	10	AOJ	動的計画法 (0・1 ナップサック問題)
7	23	AOJ	二分探索
8	29	AOJ	リングバッファを用いたキューの実装
9	17	AtCoder	二項係数, 繰り返し自乗法 (剰余)
10	11	AOJ	動的計画法 (レーベンシュタイン距離)
11	22	自作	クイックソート
12	35	AOJ	一点更新 Range Minimum Query セグメント木
13	35	AOJ	クラスカル法
14	22	自作	マージソート
15	18	AOJ	ベルマンフォード法
16	12	自作	挿入ソート
17	27	AOJ	ダイクストラ法
18	21	自作	シェルソート
19	25	AOJ	ローリングハッシュを用いたラビン・カープ法
20	22	自作	シェーカーソート
21	27	AOJ	スタックの実装
22	4	AtCoder	最小公倍数
23	23	AtCoder	エラトステネスの篩, 累積和
24	27	AOJ	Kahn のトポロジカルソート
25	22	AOJ	幅優先探索
26	20	AOJ	ワーシャルフロイド法
27	29	AOJ	ハッシュによる探索
28	31	AOJ	SPFA
29	15	AOJ	動的計画法 (0・1 ナップサック問題, 重さが大きい)
30	23	AOJ	優先度付きキューを用いたダイクストラ法

*解答ソースコードの行数 (空行は除く)

3.2 調査実験の質問内容

調査実験参加者に, 表 1 の問題と解答ソースコードを共有した. 参加者のタスクは, 各問題ごとに, 重要だと考える相異なる 3 つの行を順位をつけて選択し, 各行について重要だと考える理由を 30 文字以上で記述することである. 加えて, 2 位, 3 位の行については, それぞれなぜ 1 位, 1 位・2 位ではないのかについても記述することを求めた. また, 理由データの信頼性を担保するために, 各行の理由の記述では 30 字以上の字数制限を設けた. 上記のタスクについて, 1 人の参加者につき, 8 日以内に, 30 問全てについて, Google Form 上で回答してもらい, 1 人あたり 7500 円分の Amazon ギフトカードを謝礼として支払った.

3.3 調査実験の参加者

参加者は東京大学の学生から募集した. 信頼性のあるデータを得るために, 参加者の条件として, プログラミング経験とアルゴリズムとデータ構造の知識を有することを設定した. そこで, アルゴリズムとデータ構造に関連する講義の取得単位数や Python でのプログラミング経験, 競技プログラミングやプログラミングのアルバイト・インターンシップ経験などを問う事前調査に回答してもらった. 事前調査の内容をもとに, 調査実験参加者として適切だと判断した 11 人 (表 2) が調査実験に参加した. なお, 表 2 中の参加者の属性やスキルなどの情報は 2022 年 12 月時点のものである.

4. 収集したデータの結果と分析

4.1 収集したデータ

表 1 及び表 2 に示した 11 人, 30 問分のデータを集めることができ, データの欠損などはなかった. 図 1 は, 表 1 の問題のうち, 問題 1, 問題 4, 問題 20 の解答ソースコードと実験参加者が重要だと選択した行を可視化した図である.

4.2 重要である理由のラベリング

調査実験で, 参加者が回答した重要である理由データの分析において, データを分類するためのラベルを作成した.

ラベルの作成および全理由データに対してラベリングする作業については, 著者ら 2 人で相談しながら行なった. 2 人で理由データを読みながら, ラベルの候補を検討し, 各ラベルの内容や分類基準などを作成した. その上で, 2 人の協議のもとラベルの統合や削除などを行い, 最終的に, 表 3 に示す 14 個のラベルを決定した. この際, 複数のラベルが当てはまる理由データがあった場合は, 理由の文意からどのラベルが最も当てはまるかをもとにラベルを決定し, ラベル番号 8 のよう他のラベルと重複する場合の優先順位の決定やラベルの具体的な内容や判断基準を明確化した. 30 問全てについて 2 人で別々にラベリングし, ラベルが不一致であったデータに対しては, 2 人で協議の上で一致できる部分についてはラベルを一致させた. 990 個の理由データに対してラベルを対応させ, そのうち 2 人の間で一致したラベルをつけられたものは 956 個であった (一致率 96.6%). なお, 1 つの理由データにつき 1 人がつけるラベルの数は 1

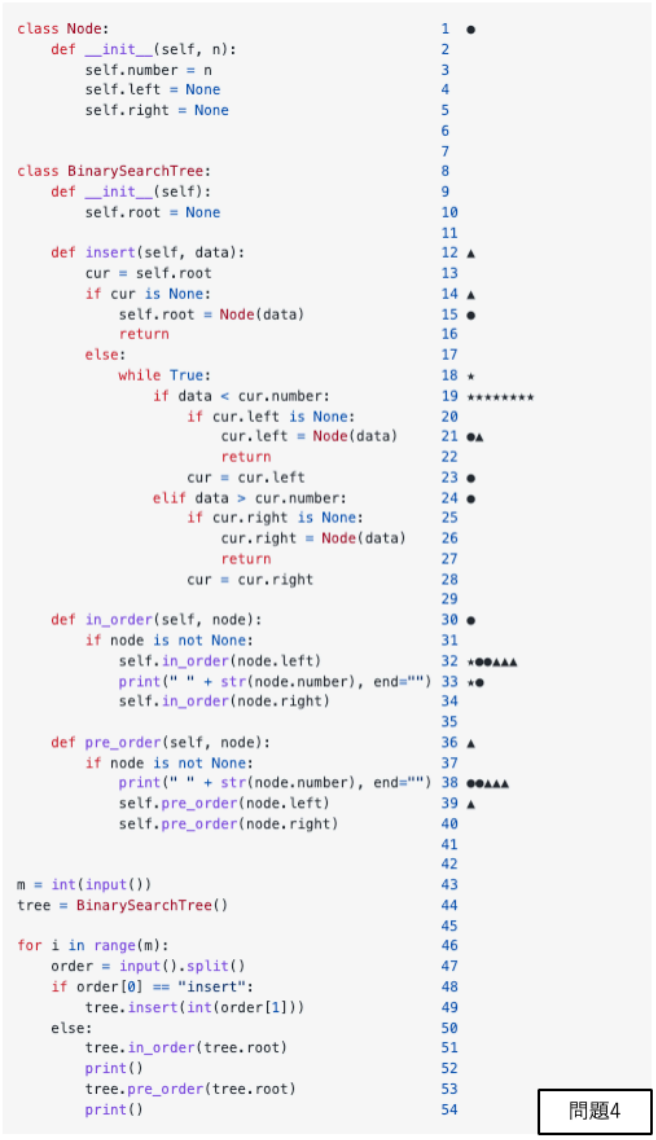

図 1: 問題 1(尺取り法), 問題 4(二分探索木), 問題 20(シェーカーソート) の解答ソースコードと, 実験参加者の回答結果. コードの右横の数字が行番号で, 行番号の隣の「黒い星」, 「黒い丸」, 「黒い三角」については, それぞれ 1 位, 2 位, 3 位を表しており, 1 記号につき 1 人選択したことを表している.

つと限定した.

次に, ラベルを貼り付けた理由データの集計を行なった. 全ての問題の全ての理由データ (990 個) に対して, 2 人分のラベルが貼ってあるため, 合計で 1980 個のデータが存在する. 以降では, 各理由データに対して, 2 人分のラベルがあるとして扱うものとする. 集計結果を図 2 に示す.

5. 考察

5.1 重要とマークされた行の分布

各問題ごとにどの行がどの程度重要なのかに関する定量的なデータを得ることができた. 図 1 に示すように, 問題によって少数の行に集中するあるいは多数の行に分散するなどの傾向が異なることがわかった. 例えば, 動的計画法の問題では, DP テーブルを更新するという 1 行に集中した. これは, 場合分けの処理も含めて三項演算子でワンライナーで記述したからであると考えられる. 逆に, 4 行以上と複数の行に分散した問題は, 問題 4 や問題 20 のように, 同様の処理をする行が複数出てくる問題に多く見られた. これは, 実験参加者がどちらの行を選択するかで分かれたことが原因であると考えられる.

5.2 ラベリングデータ

図 2 より, 順位によってラベルの構成比が変動することがわかる. ラベル番号 1, 3, 6, 7 は, 1 位ではあまり見られず, 2 位, 3 位の方が割合が大きいラベルである. このようなことから, 文法事項や初期・終端での処理, コーナーケースなどは最も重要だとは考えられていないということがわかる. 一方で, ラベル 2 は 2 位, 3 位に対して, 多くの割合を占めている. これは調査実験で, 「重要」という非常に曖昧な表現で質問をしたことが原因であると考えている. こ

表 2: 調査実験の参加者のリスト.

番号	属性*1	性別	単位数*2	その他関連するスキル・経験など
1	4 年生	男性	4	AtCoder*3 水色・Python を使用
2	3 年生	男性	2	AtCoder 黄色
3	3 年生	男性	2	AtCoder 青色
4	3 年生	男性	2	エンジニアのアルバイトで半年 Python を使用
5	4 年生	男性	2	AtCoder 青色 (最高)・Python を使用 アルバイトで 3 ヶ月 Python を使用
6	4 年生	男性	2	アルゴリズムとデータ構造の講義の TA 経験
7	4 年生	女性	2	AtCoder 緑色・研究で Python を使用
8	3 年生	男性	2	短期インターンシップで Python を使用
9	4 年生	男性	2	AtCoder 黄色 (最高) アルゴリズムとデータ構造の講義の TA 経験
10	3 年生	女性	2	
11	4 年生	女性	2	AtCoder 緑色・Python を使用

*1 全員が東京大学工学部の学部生である.
*2 アルゴリズムとデータ構造に関連する講義で取得した単位数.
*3 AtCoder では競技プログラミングコンテスト (同時間内で, 1 問以上のプログラミング問題がコンテスト参加者に出題され, より正確にかつより速く問題の要求に解答する競技.) が定期的に開催されており, コンテストでの成績によって個人のレートが決まる. レートは数値が高い方が実力があることを意味し, 色はレートの数値によって決まり, この表で出てくる色では, レートが高い方から, 黄色, 青色, 水色, 緑色の順である.

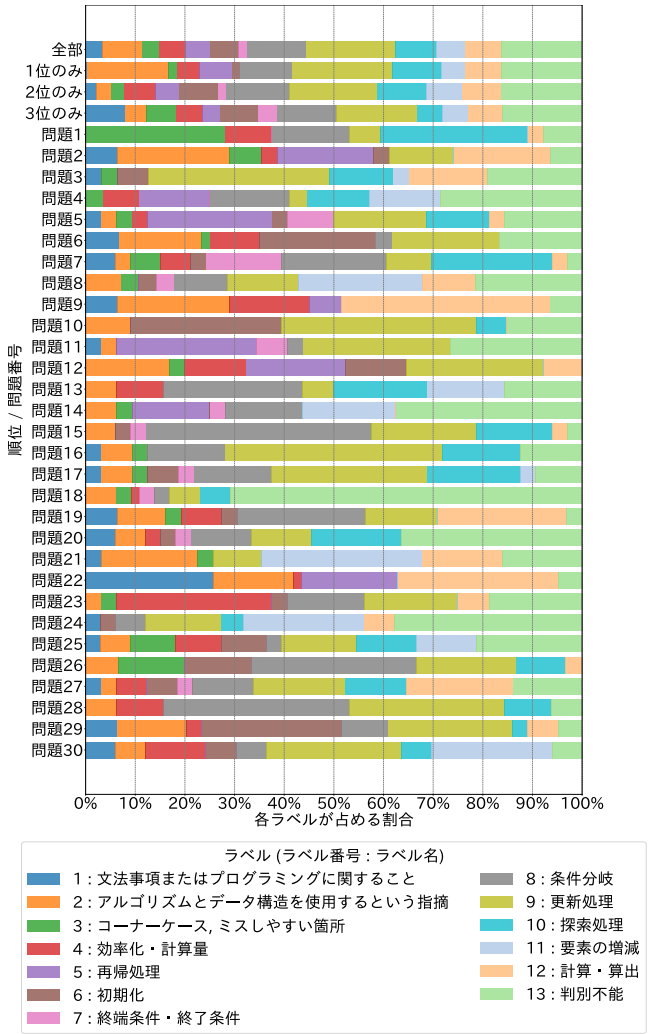

図 2: 各ラベルが全順位 (1 位, 2 位, 3 位), 順位ごと, 各問題ごとでどの程度の割合を占めるかを示した累積棒グラフ (ラベル 14 については分析・考察に用いるのは不適切であるので除外している, 1980 個中 78 個がラベル 14 であった.). 一番上のグラフは, 全問題・全順位合わせて集計したもの, 上から 2 番目から 4 番目のグラフは, 全問題を各順位ごとに集計したもの, 上から 5 番目以下のグラフは, 全順位合わせて各問題ごとに集計したものである.

の点では, 例えば, 「ワーシャルフロイド法で解けることに気付くことがこの問題のコアだから. (問題 26, 参加者 3, 1 位)」のように, アルゴリズムとデータ構造自体のソースコードではなく, あくまで問題を解くという視点において, その問題で使用されているアルゴリズムとデータ構造を呼び出している行を選択するという理由が多数見られた. 特に, 参加者 3 と参加者 7 はその傾向が顕著であった.

また, 問題ごとにラベル割合が大きく異なることがわかり, 作成したラベルが問題のアルゴリズムとデータ構造の特徴を反映していると考えることができる. ラベルごとに着目すると, 次のような洞察が得られた.

ラベル番号 1「文法事項またはプログラミングに関すること」というラベルは, 問題 22 で多いが, これは問題 22 のソースコードが 4 行と短く, 関数定義のような実験参加者があまり重要ではないと考えている行も選ばざるをえなかったためである. ラベル番号 2「アルゴリズムとデータ構造を使用するという指摘」というラベルは, 問題文の条件から考察して使用するアルゴリズムを検討する問題 (問題 8, 9, 21, 22, 23) において多く付けられている傾向があった. 他にもセグメント木の問題 (問題 2, 12) でも多く付けられていたが, これはセグメント木自体があまり馴染みがなく, 他の問題のアルゴリズムとデータ構造より難易度が高いものとして実験参加者が捉えていたためではないかと考えている. ラベル番号 3「コーナーケース, ミスしやすい」というラベルについては 問題 1 が他の問題より多いが, これは問題 1 の仕様として, 0 を含む場合と含まない場合とでコードを書き分けていることが影響していると考えられる. ラベル番号 4「効率化・計算量」のラベルは, 調査実験の問題において, ナイーブな実装をするよりアルゴリズムとデータ構造を使用することにより計算量で有利に

表 3: 作成したラベルとその説明のリスト.

ラベル番号	ラベル名	ラベルの説明
1	文法事項またはプログラミングに関すること	Python 特有の書き方, 実装の工夫, 標準入力など.
2	アルゴリズムとデータ構造を使用するという指摘	関数呼び出しなど. アルゴリズムとデータ構造そのものへの具体的な言及がないが, このアルゴリズムとデータ構造を使用すると宣言しているもの.
3	コーナーケース, ミスしやすい箇所	コーナケース, 間違いやすい, ミスをしやすい, 忘れやすいなどの記述がされている.
4	効率化・計算量	効率的にや高速に, 計算量の削減などということが明示的に書かれている.
5	再帰処理	「再帰的に」などと再帰に関して明示的に指摘されている.
6	初期化	初期値の設定や初期条件, 配列などの準備など, 初期や初めにすることへの言及がある.
7	終端条件・終了条件	無限ループや探索打ち切りなど, 終端・終了に関する言及がある.
8	条件分岐	条件分岐や, 値を確認する, 判定するなど. ラベル 3 と重複する場合は, ラベル 3 を優先し, ラベル 7 と重複する場合は, ラベル 7 を優先する.
9	更新処理	値を更新する, 上書きする, 記録する, swap 操作, index を決めるなど.
10	探索処理	走査する, 探索する, ループを回すといったことが読み取れる.
11	要素の増減	Queue や Stack, 配列などで要素を入れる, 要素を除くなど.
12	計算・算出	計算する, 算出する, 答えを求めるなど. ラベル 9 との違いは, ラベル 9 では前状態があるが, ラベル 12 では 1 回きりの求値であること.
13	判別不能	ラベル 14 には該当しないが, その他のラベルにも割り当てられない場合.
14	読解不可能, 理由として不十分	論理が破綻している, 理由になっていない, 書かれている理由に順位を決定する以上の情報が乏しい.

なっているにも関わらず, 問題ごとにラベル番号 4 の占める割合が異なる結果となった. 特に問題 23 では多くなっているが, 理由を読む限りでは, この問題が累積和とエラトステネスの篩という 2 つのアルゴリズムを組み合わせて, 問題の条件を満たす高速化を実現していることから, 計算量に焦点を当てている人が多いからだと考えられる. ラベル番号 5「再帰処理」については, 再帰を用いたソースコードの問題で登場しており, ラベリング結果としては順当である. しかし, ラベル番号 5 についてもラベル番号 4 と同様, 問題によってラベル番号 5 が全体に対して占める割合が異なっており, 問題ごとに再帰以外にも着目すべき重要な箇所の多寡があると考えられる. ラベル番号 6「初期化」は, 問題 6, 10, 29 で多かった. いずれも動的計画法の問題で, DP テーブルを最初何で埋めるかや, 初期状態をどうするかに焦点を当てた理由が多数見られた. ラベル番号 8「条件分岐」は, 問題 15, 16, 28 で多いが, いずれもコストが負の閉路を検知可能な実装の解答ソースコードであり, 負閉路の検知を指摘したデータが多かったためであった. ラベル番号 9「更新処理」については, 割合の大小の差はありながら, ほとんどの問題で見られた. 唯一問題 9 では見られなかったが, 二項係数や繰り返し自乗法など計算に関連する箇所が多かったためであると考えられる.

以上より, それぞれのアルゴリズムとデータ構造において, どのような処理や特徴が重要であるとみなされているかについてのデータを得ることができた. これは, それぞれのアルゴリズムとデータ構造を学ぶ際, 具体的な行とともに, どのような処理や特徴に注目すべきかに関する示唆を与えるものである.

5.3 制約

本研究での制約として, 調査対象が学生に限定されていることや 1 行を選択する必要があることなどがある. 実際に, 図 1 の問題 4 の二分探索木でデータの挿入を左右それぞれで分けているが, 左右で同じようなコードを書いており, 順位付けするのに困惑したという声があった. また, ラベリングにおいては, 1980 個のデータの中で, ラベル 13「判別不能」がついたデータが 308 個あり, ラベル 13 のついたデータ数が多いことも制約の 1 つである.

6. 今後の展望

6.1 プログラミング穴埋め問題作成に向けたデータセットの構築

2.2 節で述べたように, これまでのプログラミング穴埋め問題の研究では, 専らプログラミング初学者を対象としており, プログラミングの問題も FizzBuzz 問題のようなはじめてプログラミングを学習するのに適した問題を対象としていることが多かった. 一方で, アルゴリズムとデータ構造を問題として使用した穴埋め問題の研究は少ないが, 定期試験などでアルゴリズムとデータ構造の穴埋め問題が出題されたり, 基本情報技術者試験 [*3]や応用情報技術者試

[*3] 情報処理推進機構 (IPA) が実施している国家試験の 1 つ.(https://www.jitec.ipa.go.jp/1_11seido/fe.html)

験 *4では, アルゴリズムとデータ構造の知識を問うものではないにしろ, 穴埋め問題が出題されている.

そこで, 調査実験で得られたデータをもとにアルゴリズムとデータ構造の穴埋め問題を作成することを今後の展望の 1 つとして挙げられる. 調査実験で使用した問題と解答ソースコードに限定した場合, 各問題, 各行での重要度が定量化されており, その情報をもとに穴にする場所を選ぶといったことができる. 上述のことを利用して, 例として, 図 3 に示した問題 20 の解答ソースコードでプログラミング穴埋め問題を作成した.

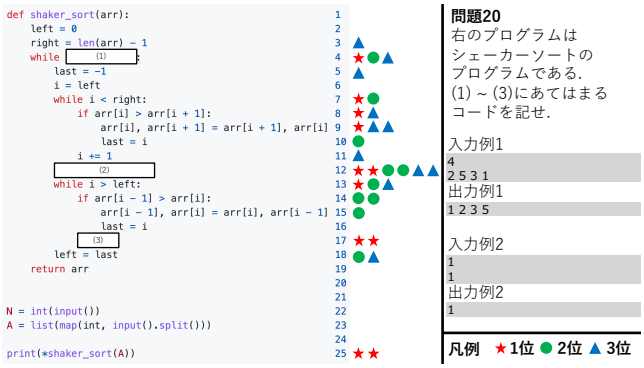

図 3: 問題 20(シェーカーソート) の解答ソースコードを使用した穴埋め問題. 穴の選択については, 4.1 節の図 1 の問題 20 の回答結果をもとにしている.

また, 4.2 節で作成したラベルとラベリング結果をもとに, ラベルを絞って穴埋め問題を作成するなどして, テーマやトピックを絞った出題も考えられる. 別のソースコードを利用するにせよ, 同じアルゴリズムとデータ構造を主眼とする問題であれば, 調査実験で明らかになった, 各アルゴリズムとデータ構造で重要と考えられている部分の知見を生かすことができる可能性がある.

上述した通りの手法で, 実際に学生の学習課題としてやってもらうユーザ実験を通して, 今まで研究されてこなかったアルゴリズムとデータ構造という分野の穴埋め問題において, より高い学習効果を得るためにどの穴を選択するかや, そもそもアルゴリズムとデータ構造の学習において穴埋め問題を利用した学習がどの程度有効であるかなどを明らかにすることを目指したい.

6.2 エキスパートのデータと初学者のデータの比較

本研究と同様の調査を, プログラミングの学習経験はあるがアルゴリズムとデータ構造に関する知識を持たない人にもしてもらうことで, アルゴリズムとデータ構造のエキスパートと初学者の重要だと考える行の違いを明らかにすることができると考えられる. その際, 調査実験では実験参加者が明確な理由なく行を選択することを防止するために, 実験参加者に重要だと考える理由を記述してもらったが, 時間的コストがかかるため, より大規模に調査を行おうと考えた場合, 理由記述は適さない. そこで, 参加者には重要な行を選んだ上で, その理由を記述する代わりに, 4.2 節において作成したラベルからラベルを選んでもらうことで, 省力化することができると考えている. この点については, ラベルの選択だけでデータとしての妥当性があるかないかをどのように判別するのかを十分に検討する必要がある. エキスパートと初学者のデータを比較することの応用として, 下記の事柄を考えている.

- エキスパートと初学者の選んだ行及びラベルの違いから, 初学者の躓くポイントがわかり, 指導者・学習者への示唆となる.
- 学習者に実際にラベルも含めて重要な行を選んでもらい, エキスパートのデータと比較することで, その学習者がその問題のアルゴリズムとデータ構造を理解しているかを判定という手法を提案する. 精度の問題などはあるが, 短時間で判定が可能な新しい学習者の評価手法となる可能性がある.

*4 情報処理推進機構 (IPA) が実施している国家試験の 1 つ.(https://www.jitec.ipa.go.jp/1_11seido/ap.html)

7. 結論

本研究では, Python のプログラミング経験とアルゴリズムとデータ構造の知識がある人たちに対して, アルゴリズムとデータ構造に関する問題と解答ソースコードにおいて, どの行がなぜ重要であるのかというアンケート調査を実施した. その結果, ソースコードレベルにおいてどの行が重要であるかというデータと, その理由のデータを得ることができた. 加えて, 著者ら 2 名でラベルを作成して, 理由データに対してラベリングすることで, ラベリングされたコードデータを得ることができた.

謝辞

本研究において様々な助言をくださり, 建設的な議論をしていただいた東京大学 Interactive Intelligent Systems Laboratory の皆様, 実験に参加してくださった皆様に厚く御礼申し上げます.

参考文献

[1] Richard Catrambone and Keith J Holyoak. Learning subgoals and methods for solving probability problems. *Memory & Cognition*, Vol. 18, No. 6, pp. 593–603, 1990.

[2] Kabdo Choi, Hyungyu Shin, Meng Xia, and Juho Kim. Algosolve: Supporting subgoal learning in algorithmic problem-solving with learnersourced microtasks. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[3] Nobuo Funabiki, Yousuke Korenaga, Toru Nakanishi, and Kan Watanabe. An extension of fill-in-the-blank problem function in java programming learning assistant system. In *2013 IEEE Region 10 Humanitarian Technology Conference*, pp. 85–90. IEEE, 2013.

[4] Steven Halim. Visualgo–visualising data structures and algorithms through animation. *Olympiads in informatics*, Vol. 9, pp. 243–245, 2015.

[5] Kohei Hara, Yu Yan, Hiroto Nakano, Hiroki Chino, Takenobu Kazuma, and Aiguo He. A fill-in-the-blank problem generator for c program beginner. *IEICE Technical Report; IEICE Tech. Rep.*, Vol. 115, No. 127, pp. 37–42, 2015.

[6] Hsu Wai Hnin and Khin Khin Zaw. Element fill-in-blank problems in python programming learning assistant system. In *2020 International Conference on Advanced Information Technologies (ICAIT)*, pp. 88–93. IEEE, 2020.

[7] Juan Rebanal, Jordan Combitsis, Yuqi Tang, and Xiang 'Anthony' Chen. Xalgo: A design probe of explaining algorithms' internal states via question-answering. In *26th International Conference on Intelligent User Interfaces*, IUI '21, p. 329–339, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Junko Shinkai, Yoshikazu Hayase, and Isao Miyaji. A study of generation and utilization of fill-in-the-blank questions for programming education on moodle. *IEICE Technical Report; IEICE Tech. Rep.*, Vol. 110, No. 263, pp. 7–10, 2010.

[9] Kenta Terada and Yutaka Watanobe. Automatic generation of fill-in-the-blank programming problems. In *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 187–193. IEEE, 2019.

[10] Sarah Weir, Juho Kim, Krzysztof Z. Gajos, and Robert C. Miller. Learnersourcing subgoal labels for how-to videos. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work amp; Social Computing*, CSCW '15, p. 405–416, New York, NY, USA, 2015. Association for Computing Machinery.

[11] 掛下哲郎, 柳田崚, 太田康介ほか. 穴埋め問題を用いたプログラミング教育支援ツール pgtracer の開発と評価. 情報処理学会論文誌教育とコンピュータ (TCE), Vol. 2, No. 2, pp. 20–36, 2016.

[12] 兼宗進, 島袋舞子, 岸本浩輝, 漆原宏丞, 本多佑希, 岸本有生ほか. アルゴリズム入門教育に適したソートアルゴリズムの検討. 研究報告コンピュータと教育 (CE), Vol. 2022, No. 33, pp. 1–11, 2022.

[13] 秋葉拓哉, 岩田陽一, 北川宜稔. プログラミングコンテストチャレンジブック: 問題解決のアルゴリズム活用力とコーディングテクニックを鍛える. マイナビ出版, 2012.

[14] 新開純子, 宮地功. プログラミング学習支援システムを用いた入門教育の実践. 日本教育工学会論文誌, Vol. 33, No. Suppl., pp. 5–8, 2009.

[15] 川口順功, 情報学部. イメージ化したアルゴリズムの教育への応用. 大学 ICT 推進協議会 2012 年度年次大会論文集, 2012.

[16] 浅野考平, 森戸隆文ほか. アルゴリズム教育再考. 研究報告コンピュータと教育 (CE), Vol. 2014, No. 1, pp. 1–4, 2014.

[17] 渡部有隆. プログラミングコンテスト攻略のためのアルゴリズムとデータ構造. マイナビ出版, 2015.

[18] 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, Vol. 16, No. 4, pp. 384–391, 2001.