プログラミング学習での利用に向けた プログラミングコンテストの問題の調査

田畑 彰洋[1,a)]　松下 誠[1,b)]　肥後 芳樹[1,c)]

概要：IT 人材の不足や教育現場の拡大といった背景から，プログラミング学習への注目が高まっている．学習の一環で問題演習を扱うにあたり，多数の問題を入手するための検討候補としてプログラミングコンテストが考えられるが，それらの問題がどう利用できるかを示した例は見られない．そこで本研究では，プログラミングコンテストの問題利用を検討する学習者・教育者への支援を見据え，「学習に向けた利用」という観点からその実態把握を行う．大学のプログラミング教育に着目し得られた学習単元に即してタグを策定し，それらをプログラミングコンテストの問題に付与することで生じる対応関係を分析した．結果，難易度ごとの問題の傾向や単元ごとの問題入手可能性が明らかとなったため，学習者・教育者に向けた問題利用に関する情報の提示を行った．

1. はじめに

近年の急速な IT 化に伴い，プログラミング学習・プログラミング教育への注目が高まっている．背景の 1 つに IT 人材の不足が挙げられ，経済産業省の予測によれば，その不足数は 2030 年時点で約 79 万人にのぼるとされている [8]．人材の育成が求められる中で，企業の新入社員研修におけるプログラミング教育の導入などが見られる [7]．また，AI の発展や IoT 技術の進展といった社会の動きとともに情報活用能力の習得が重要視され，新学習指導要領では全国の小学校・中学校・高校における段階的なプログラミング学習の導入が決定された [14]．大学教育においてもプログラミングを含む「新たな一般情報教育のカリキュラム標準」が策定される [12] など，プログラミング教育の拡大に向けた動きがうかがえる．そうした流れに伴い個人でのプログラミング学習の需要も高まり，プログラミングスクールや学習サイトが広がりを見せている [2]．

プログラミング学習方法の 1 つとして，「事前に用意された問題に対しその解答となるプログラムを実際に作成する」問題演習が挙げられる．演習の効果を得るためには多数の問題をこなしたい．より多くの問題を解きたい学習者や，より多くの問題を学生に提供したい教育者にとって，大量に揃う場所から一度に問題を入手できれば効果的であろう．そのような問題の入手先候補として，プログラミングコンテストを検討する者も存在すると考えられる．しかし現状として，プログラミングコンテスト上の多数の問題が，プログラミング学習においてどのように利用可能であるかを示した例は見られない．

そこで本研究では，そうした学習者・教育者への支援を見据え，プログラミング学習に向けた利用という観点からプログラミングコンテストの問題の特徴を明らかにすることを目指す．そのアプローチとして，タグの付与による問題の分類整理に注目した．これは，大量にあるものの実態を把握する際に用いられる手法の 1 つであり，Web 上のコンテンツに多数のユーザがタグを付けていくソーシャルタギング [3,4,6] はその一例である．

タグと問題に生じる多対多の関係について，その全体像を掴むにあたり（1）タグから問題への対応，（2）問題からタグへの対応，の 2 点に着目する．ここでは付与するタグの内容として，配列や再帰といった，詳細な学習項目としての学習単元に着目した．これは，学習状況に基づく判断に役立つと考えられるからである．例えば学習単元の全体像の把握により，自身の未学習単元が何であるかの認識が可能となる．その単元に関するタグが付与された問題に特化して学習すれば，不足している能力を効果的に補える．また，ある単元 X の学習を検討している際，X に分類される問題の数に着目することで，「X の学習にプログラミングコンテストの問題が適しているか」といった傾向を掴みやすくなると考えられる．

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Toyonaka 565-0871, Japan
[a)] a-tabata@ist.osaka-u.ac.jp
[b)] matusita@ist.osaka-u.ac.jp
[c)] higo@ist.osaka-u.ac.jp

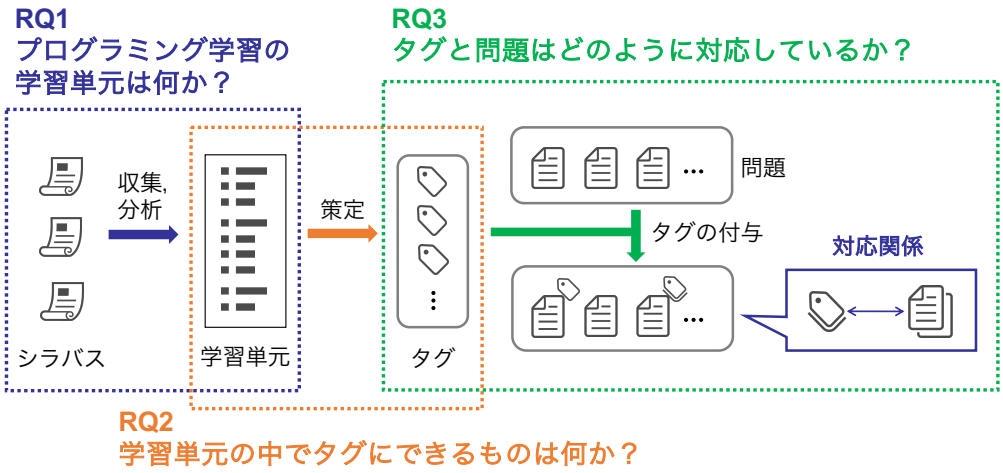


図 1: 各 RQ への回答の流れ

以降 2 節では，問題分類の手法について詳細を述べる．分類結果の考察は 3 節で行い，4 節では妥当性への脅威に関して論じる．最後に 5 節で本論文をまとめ，今後の課題について触れる．

2. タグを用いたプログラミングコンテストの問題分類

本研究における目的達成のため，学習単元に即したタグを付与してプログラミングコンテストの問題を分類するアプローチをとる．その過程で「何を学習単元とするか」及び「学習単元のうち，何をタグとするか」を明示する必要が生じることから，次に示す 3 つの Research Question（RQ）を設定し，順に回答する形で研究を進めた．

RQ1　プログラミング学習における学習単元は何か？
RQ2　学習単元の中でタグにできるものは何か？
RQ3　タグとプログラミングコンテストの問題はどのように対応しているか？

各 RQ への回答の流れを**図 1** に示す．

以降，RQ1〜RQ3 それぞれに対する詳細なアプローチと回答について述べる．

2.1 RQ1: プログラミング学習における学習単元は何か？

2.1.1 対象

プログラミング学習が行われている場として，国立大学の情報系学科[*1] に着目した．これは，教育の条件が比較的近しく，具体的な教育内容がシラバスで公開されているためである．そこで RQ1 への回答に先立ち，情報系学科のリスト化を行った．情報処理学会が 2016 年度に行った情報教育の調査研究 [11] において，「調査 A：情報専門学科」の対象とされた学科をもととし，各大学・学科の HP を適宜確認することで最新の情報を取得した．

[*1] 私立大の学部・学科は国立大に比べて数が多く，その構成が複雑であることから，今回の調査対象に含めていない．

2.1.2 アプローチ

複数のプログラミング教育現場で実際に扱われている教育内容を，3 つのステップを用いて調査・分析することで回答する．

STEP1: シラバスデータの収集

予めリスト化した各情報系学科について，2022 年度に開講されるプログラミング教育を調査し，そのシラバスから教育内容に関するデータを収集する．基本的には各学部・学科の HP に記載されたカリキュラムマップを参照して判断し，加えて「プログラミング」等の講義名・キーワード検索や開講学科での絞り込みにより，調査漏れがないかを確認した．ただし，プログラミング教育の対象はプログラミングの根幹をなす概念・知識や特定のプログラミング言語の技法を扱う講義とし，「手段」としてプログラミングを扱うもの（PBL やグループワーク等），基礎知識が習得済みであることを前提とするもの（システムプログラミングや Web プログラミング等），コンピュータ操作に慣れることが主目的のもの（コンピュータリテラシ・入門系の講義等）は対象外としている．

各シラバスにおける収集項目と収集内容を**表 1** に示す．大学によってシラバスの記載項目及び表記形式に差異があるため，授業内容に大きく関連すると考えられる主要項目に絞り，形式の統一化を行った．

上記に従い，59 の学科から 316 件のデータを収集した．

STEP2: シラバスデータの分析

収集データの特徴を把握するため，トピックモデルの代

表 1: シラバスデータの収集項目と収集内容

収集項目	収集内容
講義名	講義名をそのまま収集.
授業の目的	授業の概要や目的・ねらいに関する記述全般を収集.
到達目標	目標として独立した項目が設けられている場合に収集.
授業計画	計画の概要，及び各回の具体的な実施内容を収集.
キーワード	シラバス中に記載がある場合のみ収集.

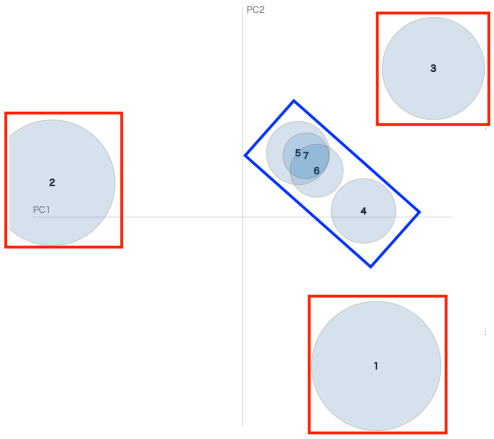


図 2: トピック数 7 の場合におけるトピックの分布傾向
（赤枠は 3 大トピックを，青枠は小トピック群を示す）

表的な手法である LDA（Latent Dirichlet Allocation） [1] を適用して分析を行う．収集したシラバスデータにトピックモデルを適用することで，関連する教育項目群がトピックという形で得られると期待できる．

分析対象は，表 1 のうち授業計画，及びキーワードとした．授業の目的といった他の項目に比べ，シラバス間での記述形式や記述量の差異が少なかったためである．

分析に先立ち，MeCab[*2] と辞書 [9,10] を用いた単語の分割，表記ゆれ統一などの単語の正規化，SlothLib [13] の単語一覧[*3]や調査対象から自明な単語（講義，プログラミング等）を用いたストップワードの設定といった前処理を施した．これは分析精度の向上や処理効率の上昇を目的としている．

Python ライブラリ scikit-learn [*4] 及び pyLDAvis [*5] を利用し，トピック数を様々に変更しながら LDA の実行と可視化を行った．まずトピック数 2〜10 で分布を観察したところ，トピック数 4 以上の場合において，大まかに「3 つの大トピックとその他の小トピック群の計 4 種に分かれる」という特徴が共通して見られた．例として，トピック数 7 の場合の可視化結果を**図 2** に示す．結果を観察すると，3 大トピックはサイズが大きく，互いに類似度が低い傾向がある一方で，小トピック群はサイズが小さく，互いに類似度が比較的高い傾向にあることが分かった．

トピック数 11〜30 における実行結果も確認したところ，明確に 5 つ以上の塊に分割されているといえるトピックの分布は見られなかった．よってトピック数を増やした場合も，3 大トピックと小トピック群の 4 つに分けられる，という全体の傾向は変わらないと判断した．

[*2] https://taku910.github.io/mecab/
[*3] http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt
[*4] https://github.com/scikit-learn/scikit-learn
[*5] https://github.com/bmabey/pyLDAvis

STEP3: 主要単元の抽出

STEP2 で得られたシラバスデータの特徴に基づき，カテゴリを策定して主要な単元を得る．3 大トピックには収集したシラバスデータの多数が属することから，これらに含まれる単語は，様々な講義で共通して扱われる教育項目とみなせる．またトピック間の類似度が低いことから，それぞれのトピックが指す教育内容は独立したものであるとみなす．本研究では，3 大トピックをもとにカテゴリを策定するとともに，各トピックを特徴づける単語（教育項目）を抽出して得られた結果を，主要な学習単元として扱うこととした．

具体的には，以下に示す手順に従う．

(1) トピック数 4〜9 における LDA の実行結果を，それぞれ pyLDAvis で可視化する．
(2) 6 つの実行結果それぞれにおいて，3 大トピックを選択した際に表示される上位 30 の単語を記録する[*6]．
(3) 3 大トピックのそれぞれにおいて，4 回以上出現した単語をまとめたカテゴリを定める．
(4) 同義の単語をまとめ，包含関係にある単語を整理し，主要単元群とする．

2.1.3 回答

RQ1 の回答として，2.1.2 節のアプローチにより得られた学習単元の一覧を**表 2** に示す．なお，包含関係にあると判断した単元はインデントで表現している．

2.2 RQ2: 学習単元の中でタグにできるものは何か？

2.2.1 アプローチ

RQ1 で得られた学習単元（表 2）からタグの設計方針に

表 2: 学習単元の一覧

(a) カテゴリ 1	(b) カテゴリ 2	(c) カテゴリ 3
アルゴリズム	制御構造	オブジェクト指向
再帰	分岐・条件	オブジェクト
動的計画法	if	インスタンス
分割統治法	繰り返し・ループ	クラス
探索	for	メソッド
二分探索木	while	コンストラクタ
データ構造	型・データ型	インターフェース
キュー	文字列・文字	継承
スタック	構造体	抽象化
リスト	ポインタ	カプセル化
ハッシュ	配列	ポリモーフィズム
ヒープ	入出力	モジュール
木構造	ファイル	GUI
ソート・整列	標準	スレッド
クイックソート	演算・演算子	例外処理
ヒープソート	変数	イベント
マージソート	関数・引数	パッケージ
グラフ	メモリ	API
文字列照合	ライブラリ	

[*6] パラメータ relevance の値を 0.6 とした [5].

適合するものを選別して回答する．次に示す 2 つのステップを用い，問題に付与するタグの全体像を明示する．

STEP1: タグの設計方針の策定

分類結果が自明なタグはノイズになり得ることが予想されるため，「方針 1: 全ての問題に付与されることが明らかなタグは設けない」「方針 2: 全ての問題に付与されないことが明らかなタグは設けない」の 2 つを，またタグ間に包含関係（依存関係）が生じると分類結果の把握が困難になり得るため，「方針 3: タグ全体をフラットな構造にする（階層構造を持たない）」をタグの設計方針として定めた．

STEP2: 単元の選別

STEP1 の方針に従った単元の選別により，タグを策定する．その際，各単元の学習目的やプログラミングコンテストの仕様を踏まえる．

まず方針 1 に従い，演算・変数・入出力といったカテゴリ 2（**表 2b**）の一部単元を策定対象外とした．プログラミングコンテストの問題は必ず入出力形式が指定されており，入力をもとに演算・加工した結果を出力して解答することが前提となっているためである．

また，プログラミングコンテストの問題は手続き型言語を含む様々な言語で解答可能であることから，オブジェクト指向言語にまつわる単元であるカテゴリ 3（**表 2c**）の全体を除外した（方針 2）．

さらに方針 3 に則り，階層構造を崩して抽象度を高める方向で統一を行った．ただし，表 2 における第 1 レベル[*7]まで抽象化すると分類の意義が薄れると判断した単元については，第 2 レベルを策定対象にした．例えば表 2a で「アルゴリズム」の配下にある単元はそのまま残している．その上で，タグ全体で見たときにタグ間の包含関係が生じないよう，著者による判断で整理を行った．

2.2.2 回答

RQ2 の回答として，2.2.1 節のアプローチにより策定したタグの一覧を**表 3** に示す．各カテゴリは表 2 のものを踏襲している．

2.3 RQ3: タグとプログラミングコンテストの問題はどのように対応しているか？

2.3.1 対象

本研究ではプログラミングコンテストの中でも AtCoder Beginner Contest（ABC）に着目し，各コンテストの A・B・C 問題を分類の対象とした．ユーザ数が多く問題量が豊富であることや，各問題に公式解説が付属していることが主な理由である．また，予め問題が難易度という尺度で分けられているため，難易度に応じた分類結果の比較等も可能となる．

表 3: 策定したタグの一覧

(a) カテゴリ 1

再帰	動的計画法
分割統治法	探索
データ構造	ソート
グラフ	文字列照合

(b) カテゴリ 2

条件分岐	ループ
構造体	ポインタ
配列	関数
メモリ	

[*7] 第 1 レベルはインデントなし，第 2 レベルはインデント 1 つ，第 3 レベルはインデント 2 つで区別している．

2.3.2 アプローチ

RQ2 で得られたタグ（表 3）の付与を通した問題の分類により回答する．以下の 2 ステップを用いて付与を実現し，タグと問題に生じた多対多の関係を観察する．

STEP1: 参照情報の収集

タグ付与の判断に必要となる，問題の特徴を表す情報を収集する．各問題の問題文・解説文・解答コードを収集対象とし，さらに解答コードは「C 言語で書かれ，かつ問題に正解したもの全て」に限定した．それぞれの情報はスクレイピングにより入手可能である[*8]．

同じ基準でデータを扱うため，2022 年 12 月 18 日時点で解説文の公開形式が HTML で統一されていた問題全てを対象とし，当時の最新 108 コンテスト[*9] の A〜C 問題（計 324 問）について情報を収集した．

STEP2: タグの付与

表 3 の各タグを，その学習に適していると判断された問題に対して付与する．付与の方針は**表 4**，**5**，**6** に示すとおりである．ここで，表 4 におけるカテゴリ 1 の条件「タグ X に関連する単元の利用が問題として意図されている」は，キーワードがヒットした場合のみ，著者による問題文・解説文の内容確認を通して判定される．例えば「文字列照合」のキーワードがヒットした際に内容を確認した結果，演算結果をただ文字列に変換して出力するだけの問題と判明すれば，タグを付与しない．

各解答コードの参照にあたり，Python ライブラリ Clang Python Bindings [*10] により作成した AST（抽象構文木）を走査し，得られたノードをリスト化した．その際，グローバル宣言部や未使用でない関数の内部といった「解答コードの実行に必要な部分」のみを参照した．さらにリスト化されたノードを探索し，各マッチ条件に適合するものが 1 つでもあれば，そのマッチ条件を満たすとして扱った．ただし関数タグに関しては，main 関数以外に使用する関数があった場合にマッチ条件を満たすものとした．

2.3.3 回答

2.3.2 節のアプローチに従い，AtCoder Beginner Contest の A・B・C 問題 108 問ずつ，計 324 問に対しタグ付けを

[*8] 解答コードのページは，AtCoder Problems（`https://kenkoooo.com/atcoder/`）で公開されている提出情報ファイルを基にアクセスできる．

[*9] それ以前のコンテストでは，解説文が PDF 形式のみで公開されていた．

[*10] `https://github.com/llvm-mirror/clang/tree/master/bindings/python`

実施した．RQ3 の回答として，まず各タグが付与された問題数と全体に占める割合を**表 7** に示す．割合を合計すると 100%を超えるのは，1 つの問題にタグが複数付与されている場合があるためである．

続いて問題からタグへの対応関係として，各難易度の問題に付与されたタグの種類別分布を**図 3** 及び**図 4** に示す．またタグから問題への対応として，各タグが付与された問題数の難易度別分布を**図 5** に示す．

表 4: タグの付与方針

カテゴリ	タグの付与条件	参照情報
カテゴリ 1	「キーワードが問題文・解説文に含まれる」かつ「タグ X に関連する単元の利用が問題として意図されている」	問題文・解説文
カテゴリ 2	「タグ Y のマッチ条件を満たす解答コードが全体の過半数」	解答コード

表 5: 【カテゴリ 1】タグごとに設定したキーワード

タグ名	キーワード
再帰	再帰
動的計画法	動的計画，DP
分割統治法	分割統治
探索	探索
データ構造	データ構造，キュー，スタック，リスト，ハッシュ，ヒープ，木構造
ソート	ソート，整列，並べ替え，並び替え
グラフ	グラフ
文字列照合	文字列

表 6: 【カテゴリ 2】タグごとに設定したマッチ条件

タグ名	マッチ条件
条件分岐	if 文，switch-case 文のいずれかが使用されている
ループ	for 文，while 文，do-while 文のいずれかが使用されている
構造体	構造体が宣言されている
ポインタ	ポインタが宣言されている
配列	配列が宣言されている（文字列定数の宣言は除く）
関数	main 関数以外にソースファイル上で定義された関数があり，それがプログラム実行時に呼び出される
メモリ	malloc, calloc, realloc 関数のいずれかが呼び出されている

表 7: 各タグが付与された問題数と全体に占める割合

(a) カテゴリ 1

タグ名	問題数	割合
文字列照合	86	26.5%
探索	49	15.1%
ソート	32	9.9%
データ構造	19	5.9%
動的計画法	9	2.8%
グラフ	8	2.5%
再帰	4	1.2%
分割統治法	0	0.0%

(b) カテゴリ 2

タグ名	問題数	割合
条件分岐	260	80.2%
ループ	226	69.8%
配列	203	62.7%
関数	28	8.6%
構造体	13	4.0%
ポインタ	4	1.2%
メモリ	2	0.6%

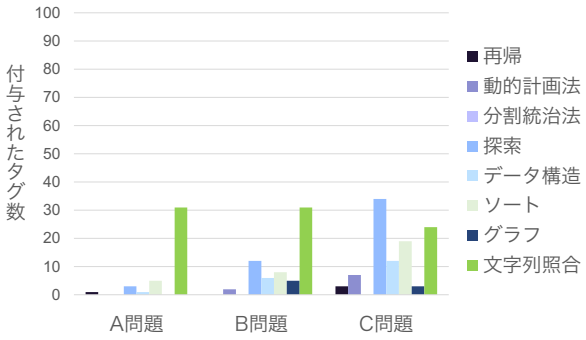


図 3: 【カテゴリ 1】問題に付与されたタグの分布

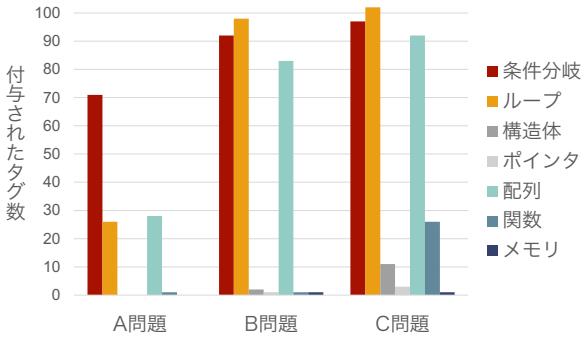


図 4: 【カテゴリ 2】問題に付与されたタグの分布

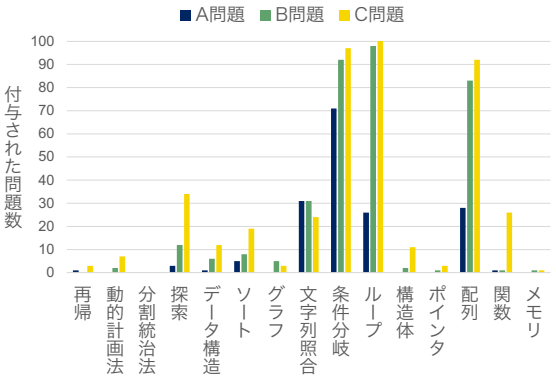


図 5: 各タグが付与された問題数の難易度別分布

3. 考察

2 節の各 RQ で得られた結果について考察を行い，学習者・教育者に向けた問題利用に関する情報をまとめる．

3.1 学習単元のカテゴリの解釈

RQ1 で得られた学習単元（表 2）の各カテゴリについて，その構成要素に着目すると，カテゴリ 1 はアルゴリズムに関する単元，カテゴリ 2 は手続き型言語の基礎に関する単元，そしてカテゴリ 3 はオブジェクト指向に関する単元との解釈が可能である．これは単元のカテゴリのもととなっ

た 3 大トピック（図 2）の解釈でもある[*11]．

また，分析で使用した LDA は「1 つの文書が複数のトピックを有する」としたモデルである．よって，分析対象の各シラバスが複数カテゴリにまたがった単元を有することもあり得る．実際のシラバスを確認したところ，カテゴリ 1・2 に属する項目を同じ授業で扱うケースが見られた．すなわちこれらの 2 カテゴリは，必ずしも完全に分割されているわけではなく，互いに関係していることがうかがえる．一般的にアルゴリズムの内容はプログラミング基礎の内容を包含しているため，カテゴリ間の関係は学習の順序にも繋がると考えられるが，LDA による分析のみでは，学習における順序関係を把握できなかった．したがって本研究では，RQ1 により得られたカテゴリをもとにその後の議論を進めている．

ただし，最終的に RQ3 でタグが付与された状態であれば，順序関係を考慮した問題の選定は可能である．例えば，単にループを扱う問題にはループタグのみが付与される一方，ループ構文を利用してソートを扱う問題には，ループとソートのタグがいずれも付与される．この差異により，習熟度に応じて問題を選定できる．

3.2 タグと問題の対応

RQ3 で得られたタグの付与結果から各種傾向・特徴を掴み，それらが生じた理由を考察する．以下では，表 3 のうちカテゴリ 1 のタグを「アルゴリズムに関するタグ」，カテゴリ 2 のタグを「基礎事項に関するタグ」として扱う．

タグの付与数に見られる難易度ごとの問題の特徴

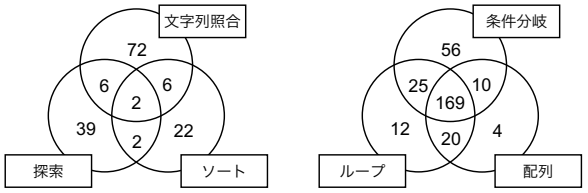
各難易度の問題に付与されたタグの数や種類に着目し，その特徴を得る．図 3 より C 問題にはカテゴリ 1 のタグが多く付与された一方，A・B 問題への付与数は全体として少なく，探索やソートで大きな違いが見られる．また図 4 より，B・C 問題にはカテゴリ 2 のタグが多く付与され，条件分岐・ループ・配列の 3 つが顕著である．それに比して A 問題への付与数は少なく，条件分岐のみが突出して多いという違いが見られた．

上記のことから，A 問題には基礎事項の利用と習得を目的とした問題が多く，B・C 問題には複雑な数式やアルゴリズムを実現する手段として基礎事項を用いるものが多いといえる．特に C 問題はアルゴリズムに関する問題を多数含む．これらの傾向は，各問題で要求される処理が難易度の上昇とともに複雑化していることから生じていると考えられる．

付与された問題の難易度に見られるタグの特徴

各タグが付与された問題の難易度分布（図 5）に着目すると，タグを大きく 4 種類に分けることができる．その詳細を**表 8** に示す．(2) のようなタグが生じたのは，B・C 問題では A 問題に比べ複雑な処理が必要とされるためであると考えられる．例えば，B 問題以降では座標計算や数式・不等式の処理技術やアルゴリズムの実現を求める問題が増えている．それらの問題を解くにあたってループや配列といった技術が必要とされ，この傾向が生じたと解釈できる．

[*11] LDA におけるトピックの解釈は，通常人間が行う．



(a) カテゴリ 1　　(b) カテゴリ 2

図 6: 頻出 3 タグのみに着目した問題数の分布

また (3) のタグが生じた要因として，C 問題ではより複雑な解答が要求されるようになったことが挙げられる．例えば，解答プログラムの規模が大きくなり，複数の関数へ処理を分割したり，構造体を導入しデータを適切に管理したりしないとうまく解けないような問題が増加している．

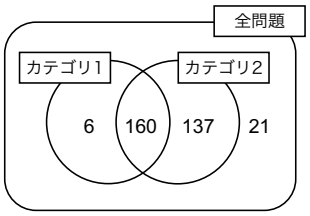
頻出タグの付与関係に見られるカテゴリの特徴

表 7 より，カテゴリ単位でタグの付与傾向に差が生じていることが読み取れる．カテゴリ 2 は分布の二極化という点で特徴的であり，条件分岐・ループ・配列の上位 3 タグが 6 割以上の問題に付与されたのに対し，他のタグが付いた問題は全体の 1 割を切っている．一方で，カテゴリ 1 のタグはカテゴリ 2 に比べると，タグ間の付与数に大きなばらつきは見られず，かつ平均的な付与数はカテゴリ 2 より少ない．

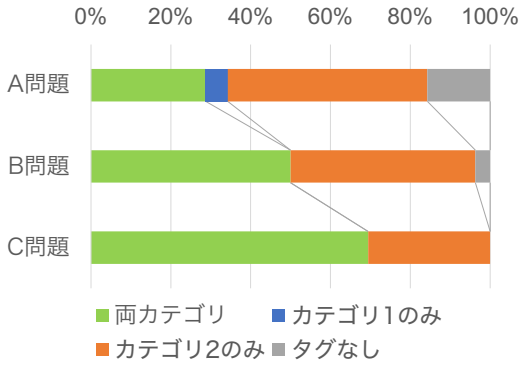
付与傾向の差が生じた原因として，同カテゴリ中のタグ間に生じる付与関係の違いが考えられる．以下，各カテゴリにおいて付与された問題数が多い上位 3 タグを「頻出 3 タグ」と呼ぶ．2 カテゴリの頻出 3 タグについて，それぞれの付与有無に着目した問題数の分布を**図 6** に示す．円内部の数字は対応するタグが付与された問題数を表し，複数のタグが同時に付与された場合は円の共通部分に記載している．

表 8: 付与された問題の難易度に応じたタグの種別

タグの種別	特徴
(1) A〜C 問題が同程度の割合 例：文字列照合・条件分岐	・難易度によらず，一般的に求められる基本技術 ・学習難度**低**，初学者向き
(2) B・C 問題の割合が高い 例：ループ・配列	・ある程度複雑な処理を行う際に必須となる技法 ・学習難度**中〜高**，ある程度の経験者なら利用可
(3) C 問題のみ割合が高い 例：探索・ソート・構造体・関数	・アルゴリズムの使用や複雑な実装を求められたときに必要となる技術 ・学習難度**高**，初学者には不向き
(4) その他	（付与数が少ないもの）



(a) 問題全体の分布



(b) 難易度別の付与パターンの割合

図 7: タグの付与有無に着目した問題数の分布

カテゴリ 1 のタグは，互いに独立して付与される傾向が見られる．すなわち，同じ問題で複数のアルゴリズムに関するトピックが設定されることはほとんどなく，カテゴリ全体として見たとき，タグの付与数が少なくなりやすい．また，ある 1 種のタグが大半の問題に付くといったことは見られない．

一方カテゴリ 2 は，頻出 3 タグの共通部分に多数の問題がある．カテゴリ全体としてタグの付与数が多くなりやすいものの，その実態として，付与された問題自体が重複していることが分かる．特に 3 種のタグがいずれも付与された問題は 169 問にのぼり，これはカテゴリ 2 のタグが付与された問題の総数（297 問）の半分を上回る．169 問のうちおよそ半数の 83 問が C 問題，次いで 75 問が B 問題，A 問題は 11 問に留まったことから，難易度が高い問題ほど，1 つの問題で複数の基礎事項を組み合わせて使うことが求められるといえる．ただし条件分岐タグの独立部分は 56 問と他の 2 タグに比べて多く，うち 44 問が A 問題であった．難易度の低い問題では基礎事項を単体で用いるケースが多いと考えられる．

カテゴリ間の付与関係に見られる問題の特徴

各カテゴリのタグが 1 つ以上付与されたか否かに着目して問題数の分布を調査し，その特徴を知る．全体の分布，及び難易度ごとの付与パターンの割合を**図 7** に示す．全体の大多数を占めるのは 2 パターンの問題であることが読み取れ，その性質は大きく以下のように分けられる．

(1) カテゴリ 2 のみが付与された問題: 基礎事項に関する技術のみで解答可能な問題.

(2) 両カテゴリが付与された問題: 基礎事項に関する技術を組み合わせ，アルゴリズムを用いて解答する問題.

(1) は A 問題に多く見られ，難易度の上昇とともに減少傾向にある．対して (2) は難易度の上昇とともに増加し，C 問題で最も多く見られる．単純な基礎事項のみを学習したい場合は (1) の問題に，アルゴリズムを含む比較的複雑な問題を学習したい場合は (2) に注目すれば，目的の問題を入手しやすくなると考えられる．

また，カテゴリ 1 のタグが付いた問題に共起するカテゴリ 2 のタグに着目すると，各アルゴリズムの実現において重視される基礎事項が何かを把握できる．カテゴリ 1 の頻出 3 タグについて，共起しやすいカテゴリ 2 のタグを**表 9** に示す．いずれも共起する 3 つのタグは同様だが，文字列照合・ソートは配列と最も関連しているのに対し，探索は条件分岐・ループと一番関連が深いという点で性質が異なる．何らかのアルゴリズムを実装するにあたり，これらの技法が要求されることは当然であるとは考えられるが，実際にそのような傾向を確認できた．

3.3 問題利用に際する情報

3.2 節での考察を踏まえ，AtCoder Beginner Contest の A〜C 問題をプログラミング学習に利用する際に有用となり得る情報をまとめる．まず，問題全体を (1) 難易度と (2) 内容で分類し対応関係を考察していたため，問題の利用ケースは大きく (1) と (2) の 2 軸で捉えられる．さらに (2) は，RQ への回答の過程で 2 つのカテゴリに分けられた（カテゴリごとでタグの付与方法も異なる）．これらを加味し，計 3 つの観点に分けて情報を提示する．

なお以降では，各単元の難易度は対応する A〜C 問題の比率に依存し，対応する問題数が多いほど単元の学習適性も高いとみなしている．また A 問題は初学者に適しており，C 問題は経験者向き，B 問題はそのおよそ中間として扱っている．

難易度別の利用

難易度でドメインを区切って問題を利用する際の指針となる情報を，**表 10** に示す．

アルゴリズムに関する単元の利用

アルゴリズムに関する単元に着目して問題を利用する際，指針となる情報を**表 11** に示す．

また，関連する情報として以下が挙げられる．

- 文字列照合に関して実際に問題を確認すると，文字列の完全一致等の単純な照合は A 問題に，複雑なパター

表 9: 頻出 3 タグと共起しやすいカテゴリ 2 のタグ

文字列照合 （86 問）	探索 （49 問）	ソート （32 問）
配列 (87.2%)	条件分岐 (89.8%)	配列 (96.9%)
条件分岐 (80.2%)	ループ (89.8%)	ループ (90.6%)
ループ (74.4%)	配列 (77.6%)	条件分岐 (87.5%)

ンマッチはB・C問題にみられ，習熟度に応じた選択が可能である．

- 問題をまとめて入手した際，多くの問題は各単元1つだけに対応している．1つのアルゴリズムに特化して学びたい場合には適している一方，複数を一括で学べるような問題はあまり含まれない．
- ほとんどの問題で，付随して配列・条件分岐・ループといった基礎事項の単元も学習可能である．

基礎事項に関する単元の利用

基礎事項に関する単元に着目して問題を利用する際，指針となる情報を**表 12** に示す．また，関連情報として以下が挙げられる．

- 条件分岐・ループ・配列の3単元に対応する問題は難易度を問わず存在する．状況や目的に応じて難易度を選択することで学習の効果を高められる．
- 基礎事項に関する単元のみをまとめて学習したい場合，A問題へ重点的に取り組むとよい．
- 問題をまとめて入手した際，半数以上の問題で一度に条件分岐・ループ・配列を学ぶことができる．
- 1つの基礎事項のみに特化して学びたい場合，条件分岐を除き，あまり多くの問題は手に入らない．

表 10: 難易度別の問題利用に際する情報

種別（難易度）	問題と解答の特徴
A 問題（低）	・基礎事項の習得を目的としたものが多い ・それぞれの基礎事項を単体で用いて解答
B 問題（中）	・ある程度複雑な処理が求められる ・複数の基礎事項を組み合わせて解答
C 問題（高）	・アルゴリズムに関する問題が大きく増加 ・B 問題よりも多くの基礎事項を組み合わせて解答

表 11: アルゴリズムに関する単元の利用に際する情報

単元	問題と解答の特徴
文字列照合	・どの難易度でも比較的多く問題が手に入る ・習熟度に応じた選択が可能で，学習に適する
探索 データ構造 ソート	・A・B 問題にはあまり見られず，C 問題で比較的多く入手可能 ・単元としての難度が高い ・初学者には不適，経験者なら適する
再帰 動的計画法 分割統治法 グラフ	・対応する問題数が非常に少ない ・学習にはあまり適さない

表 12: 基礎事項に関する単元の利用に際する情報

単元	問題と解答の特徴
条件分岐	・難易度を問わず大量の問題が手に入り，学習に適する
ループ 配列	・A 問題ではある程度，B・C 問題では大量に入手可能 ・条件分岐同様，習熟度に応じた難易度選択が可能
関数 構造体	・難易度の低い問題がほとんど手に入らない ・C 問題では一定数手に入るため，難度が高い ・初学者には不向き，経験者の学習には適する
ポインタ メモリ	・対応する問題数が非常に少ない ・学習にはあまり適さない

4. 妥当性への脅威

4.1 内的妥当性

シラバスデータの収集・分析方法

RQ1 におけるシラバスデータの収集では，各大学の Web システムやシラバス記法が統一されていないため，すべての作業を手動で行った．カリキュラムマップによるプログラミング教育の把握やシラバスの項目収集に関して，判断が主観的である可能性がある．

また，収集データの特徴把握にはトピックモデルとして LDA を適用した．教師なし学習モデルによるアプローチでは，事前処理の精度や可視化方法の選択によって結果が大きく左右されることを否定できない．

さらに本研究では，LDA により得られたカテゴリをもとに単元の一覧（表 2）を提示し，その後の議論を進めている．これはカテゴリ内・カテゴリ間における個々の単元に関して，学習の順序関係を加味せず並列に扱っている．単語間の依存関係を明示可能な分析手法の適用により，順序を加味した形式（入れ子構造やグラフなど）で単元を列挙できれば，より実態を反映した回答が可能になると考えられる．RQ1 の回答の質が RQ2・RQ3 にも関わるという点でも，異なる分析手法の検討が必要である．

タグの策定・付与方法

RQ2 でのタグ策定について，各工程が主観的である可能性がある．本研究ではカテゴリ 3 の単元（表 2c）を全て除外したが，オブジェクト指向に関連するかの判断が曖昧なものが一部含まれる（スレッド・例外処理 等）．また，階層構造のフラット化は著者の主観に基づいており，定量的な指標がなく再現性に欠ける．

RQ3 におけるタグ付与方法（表 4）では，カテゴリ 1 に属する単元が問題として意図されていても，その明確な記述が無い場合にタグを付与できていない．すなわち，実態よりもタグの付与数が少ない可能性がある．また，内容確認を著者 1 人で行っており，主観性の混在を否定できない．複数人でタグ付与を行うなどの改善が必要である．さらに，タグ付与の情報源として AtCoder 公式の解説文を採用しているため，解説が提供されていないコンテストや他サイトへの適用は困難である．分析方法の汎用性が低く，AtCoder 以外のプログラミングコンテストサイトを対象とする場合，本研究と同様の手順は適用できない場合がある．

その他，2 カテゴリ間でタグの付与方法が完全に異なるため，付与状況の単純比較が妥当か検討する必要もある．

4.2 外的妥当性

シラバスデータの収集対象

RQ1 について，シラバスの収集対象は国立大学の情報系学科に限られている．私立大学も含めると収集結果の傾向

が異なる可能性があり，対象範囲を拡大したデータ収集が求められる．

タグの付与対象

RQ3 におけるタグの付与対象は, AtCoder Beginner Contest のうち A〜C 問題，計 324 問のみであり，最終的に得られた結果の妥当性に欠ける可能性がある．問題数や難易度の範囲を拡大してタグ付けを行った場合，結果の傾向が異なる可能性を否定できない．

有用性に関する評価

本研究では，RQ への回答を通して得られた分類結果に関して，その有用性を評価できていない．学習者・教育者への支援という観点からは，以下のような手法で評価が可能と考えられる．

- 学習者・教育者から見た具体的なユースケースを聴取し，そのストーリーに沿った適用実験を行う．
- 本研究で実施したタグ付けをもとに，特定の単元を含む具体的な問題を特定するシステムを提案する．

5. おわりに

本研究では，プログラミング学習に向けた利用という観点からプログラミングコンテストの実態を把握すべく，学習単元に即したタグの付与により問題を分類整理し，生じる対応関係を分析した．

その結果，問題の難易度に応じて付与されるタグの傾向は異なり，低難易度の問題は単純な基礎事項のみで解答可能なことや，難易度の上昇とともに問題が複雑化し，アルゴリズムに関する単元が多く含まれることなどが明らかとなった．また，各タグが付与された問題の難易度傾向にも差異があり，難易度によらず問題を入手可能な初学者向けの単元，高難易度の問題からのみ入手できる経験者向けの単元，及び対応する問題数が少なく学習に適さない単元などの存在が明示された．そうしたデータをもととし，問題利用に際する情報を複数の観点からまとめて提示した．

本研究を通して得られた知見の中には，AtCoder の問題構成を理解している人にとって既知のものも存在すると思われる．しかし，そのような傾向が実際にあるかどうかを確認した，という点では意義がある．また，複数大学の情報系学科のシラバスデータから 3 大トピックと小トピック群を抽出し，学習単元として整理した点も貢献の 1 つである．

今後の課題としては，次の 3 点が挙げられる．

分類対象の拡大

AtCoder 以外のコンテストの問題も分類すれば，要求されるアルゴリズムの違いや，ある基礎事項に特化した問題の量などに着目した比較が可能となる．

オブジェクト指向言語に関する調査

本研究ではタグの策定対象外としたオブジェクト指向言語について，プログラミングコンテストの問題との関連性を調査することも課題の 1 つである．クラスや継承など特有の技法を用いた方が解答しやすい問題があるのか，実際にオブジェクト指向言語で解答している参加者は，それらの技法を活かして解答しているのか，といった点にも注目したい．

実用的なシステムの構築

本研究で用いたタグをもととした，学習者・教育者を支援するシステムの構築も有用であると考える．入力された問題に適した学習単元を出力するシステムや，指定された単元の学習に適した問題を複数提案するシステムを開発できれば，個人のニーズに合わせた利用が可能となる．

また，分類済の問題をデータベースとして提供すれば，問題の入手性向上に加え，新たな分析への発展等も期待できる．

謝辞 本研究は JSPS 科研費（JP20H04166, JP21K18302, JP21K11829, JP21H04877, JP22H03567, JP22K11985）の助成を得て行われた．

参考文献

[1] Blei, D. M., Ng, A. Y. and Jordan, M. I.: Latent Dirichlet Allocation, *Journal of Machine Learning Research*, Vol. 3, pp. 993–1022 (2003).

[2] GMO メディア, 船井総合研究所：2022 年プログラミング教育市場規模調査, `https://www.gmo.media/archives/4324/` (Accessed on 2023/07/20).

[3] Golder, S. A. and Huberman, B. A.: Usage patterns of collaborative tagging systems, *Journal of Information Science*, Vol. 32, No. 2, pp. 198–208 (2006).

[4] Marlow, C., Naaman, M., Boyd, D. and Davis, M.: HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, to Read, *Proceedings of the Seventeenth Conference on Hypertext and Hypermedia*, HYPERTEXT '06, Association for Computing Machinery, pp. 31–40 (2006).

[5] Sievert, C. and Shirley, K. E.: LDAvis: A method for visualizing and interpreting topics, *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, Association for Computational Linguistics, pp. 63–70 (2014).

[6] Voss, J.: Tagging, Folksonomy & Co - Renaissance of Manual Indexing? (2007).

[7] 塩崎正人，敦賀誠一， 中川宙，古谷芳康，吉田典弘：企業の新入社員研修におけるプログラミング教育の実践例，情報処理学会研究報告， Vol. 2020-CE-153, No. 8, pp. 1–9 (2020).

[8] 経済産業省 情報技術利用促進課：IT 人材の育成，`https://www.meti.go.jp/policy/it_policy/jinzai/index.html` (Accessed on 2023/07/20).

[9] 佐藤敏紀，橋本泰一， 奥村学：単語分かち書き用辞書生成システム NEologd の運用 —文書分類を例にして—，言語処理学会研究報告， Vol. 2016-NL-229, No. 15 (2016).

[10] 佐藤敏紀，橋本泰一， 奥村学：単語分かち書き辞書 mecab-ipadic-NEologd の実装と情報検索における効果的な使用方法の検討，言語処理学会第 23 回年次大会発表論文集，No. NLP2017-B6-1 (2017).

[11] 情報処理学会：超スマート社会における情報教育の在り方に関する調査研究，文部科学省先導的大学改革推進委託事業 (2017).

[12] 情報処理学会：カリキュラム標準一般情報処理

教　育（GE）　，https://www.ipsj.or.jp/annai/committee/education/j07/ed_j17-GE.html　(Accessed on 2023/07/20).
[13] 大島裕明, 中村聡史, 田中克己: SlothLib：Web サーチ研究のためのプログラミングライブラリ，日本データベース学会 Letters, Vol. 6, No. 1, pp. 113–116 (2007).
[14] 文部科学省：平成 29・30・31 年改訂学習指導要領（本文、解説），https://www.mext.go.jp/a_menu/shotou/new-cs/1384661.htm (Accessed on 2023/07/20).