特別研究報告

題目

大学におけるプログラミング教育のシラバスに着目した プログラミングコンテストの問題へのタグ付け

指導教員
肥後 芳樹 教授

報告者
田畑 彰洋

令和 5 年 2 月 7 日

大阪大学 基礎工学部 情報科学科

令和 4 年度 特別研究報告

大学におけるプログラミング教育のシラバスに着目した
プログラミングコンテストの問題へのタグ付け

田畑 彰洋

内容梗概

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．その背景として，深刻な IT 人材の不足，全国の教育機関におけるプログラミング教育の拡大などが挙げられる．またそうした流れを受け，個人でのプログラミング学習の場も広がりを見せている．

プログラミング学習方法の一つに，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．演習の効果を得るべく多数の問題を扱いたい学習者や教育者にとって，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．そのような入手先の候補として，プログラミングコンテストを検討する利用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．

そこで本報告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実態を把握することを目指す．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．また，問題の利用者にとっての有用性を考慮し，タグの内容は学習単元に即したものとする．

問題の分類に先立って学習単元を明示するにあたり，大学の情報系学科におけるプログラミング教育に着目した．その教育内容が記載されたシラバスのデータを収集・分析した結果，類似する単元群が得られた．続いて，学習単元に即して策定したタグをプログラミングコンテストの問題に対して付与し，問題からタグへの対応関係，及びタグから問題への対応関係を分析した．その結果，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方針の提示を行った．

主な用語

シラバス
プログラミングコンテスト

目 次

1 まえがき **4**

2 背景 **6**

2.1 シラバス 6
2.2 オンラインジャッジシステム 9
2.3 プログラミングコンテスト 9
2.3.1 AtCoder 9

3 タグを用いたプログラミングコンテストの問題分類 **11**

3.1 RQ1：プログラミング学習における学習単元は何か？ 12
3.1.1 対象 12
3.1.2 アプローチ 13
3.1.3 回答 18
3.2 RQ2：学習単元の中でタグにできるものは何か？ 19
3.2.1 アプローチ 19
3.2.2 回答 20
3.3 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？ 20
3.3.1 分類対象 20
3.3.2 アプローチ 20
3.3.3 回答 24

4 考察 **31**

4.1 学習単元の各カテゴリの解釈 31
4.2 タグの策定方針 31
4.3 タグと問題の対応 32
4.4 問題を利用する際の方針 36
4.5 妥当性への脅威 38
4.5.1 内的妥当性 38
4.5.2 外的妥当性 39

5 まとめ **40**

謝辞 **41**

A 付録：情報系学科リスト **42**

参考文献 44

1　まえがき

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．その背景の一つに IT 人材の不足が挙げられ，経済産業省の予測シナリオの 1 つによれば，2030 年には約 79 万人の IT 人材が不足すると推定されている [9]．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 [8]．また，AI の発展や IoT 技術の進展といった社会の動きを受けて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 [16]．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プログラミング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一つとして，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学習者・教育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．

そこで本報告では，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一つであり，その一例として，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] がある．また，付与するタグの内容が学習単元に即したものであれば，問題の利用者にとって有用であろう．例えば，学習状況に応じた問題の利用判断に役立つと考えられる．ここで学習単元とは，配列や再帰といった詳細な学習項目のことを指すものとする．問題の分類に先立って明らかにすべきこととして，学習単元は何か，及び付与するタグは何か，の 2 点が挙げられる．本報告では，以下に示す 3 つの Research Question（RQ）を設定し，順に回答することで目的の達成を図ることとした．

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？

以降，2 章では本報告の背景として，シラバス，オンラインジャッジシステム，プログラミングコンテストサイトについて説明する．続いて 3 章で RQ の設定背景，及び各 RQ へのアプローチと回答について述べたあと，4 章では RQ の回答についての考察を行う．最後に 5 章で本論文をまとめ，今後の課題について議論する．

2　背景

本章では，本報告の背景としてシラバス，オンラインジャッジシステム，プログラミングコンテストについて説明する．

2.1　シラバス

シラバスとは，大学で開講される授業や講義の大まかな内容や授業目標を示した資料のことであり，この記載内容に基づいて，授業や講義が実施される．学生はシラバスを利用することによって自分の必要とする講義や教科書の検索を行うことが可能となる．

日本の大学では，2008 年に施行された大学設置基準法の改正において “成績評価基準等の明示等（第 25 条の 2 関係）” の項目が明記され [15]，以降シラバスの作成は事実上義務化されている．また，具体的な記載事項の例についても文部科学省による言及があり [17]，表 1 に示すとおりである．

シラバスの執筆は基本的に各講義の担当教員により行われる．また，公開・配布の形式は厳密に定められてはいないが，現在は各大学の Web システム上で一般公開する形態が多く見られる．Web システムには検索機能が備わっており，科目区分・曜日・時限といった開講条件，及び講義名・教員名・キーワードといった単語条件による絞り込みができる．この検索機能を利用して，内部の学生に限らず，誰でもシラバスにアクセスすることが可能となっている．

大阪大学におけるシラバスへのアクセス例を挙げる．まず外部公開シラバス[1] にアクセスすると，図 1 のような検索画面が表示され，赤枠 1 で示した部分で検索条件を指定することが可能となる．この例では，青枠で示した 3 条件（開講年度，開講学部，開講科目名）を指

表 1: シラバスの記載事項の例 [17]

授業名
担当教員名
講義目的
各回ごとの授業内容
成績評価方法・基準
準備学習等についての具体的な指示
教科書・参考文献
履修条件

[1] https://koan.osaka-u.ac.jp/syllabus_ex/campus

定している．続いて，赤枠 2 の [検索] ボタンを押下すると，図 2 のように検索結果の講義一覧が出力される．シラバスを閲覧したい講義があれば，赤枠 3 で示される [和文] ボタンを押下することで図 3 のような画面に遷移し，シラバスを参照することができる．

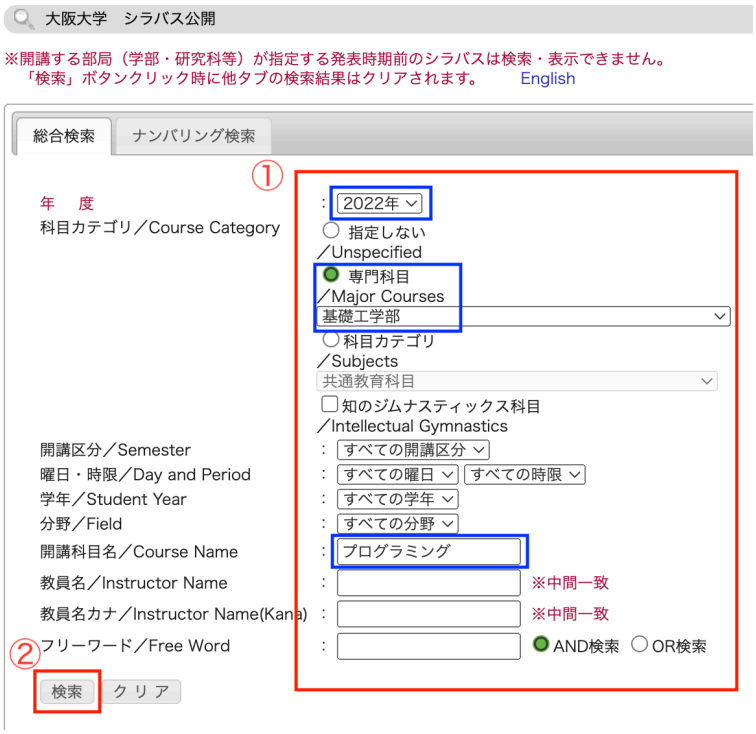


図 1: 大阪大学のシラバス検索画面．青枠は検索条件の入力例を表す

検索結果ページ：　<< 前へ　1　次へ >>

No.	開講所属	開講期	開講区分	曜日・時限	時間割コード	開講科目名	担当	参　照
1	基礎工学部	春学期	春～夏学期	月3,木3	090467	プログラミングA	若宮　直紀 ③	和文 英文
2	基礎工学部	春学期	春～夏学期	月3,木3	099025	プログラミングA	楠本　真二	和文 英文
3	基礎工学部	秋学期	秋～冬学期	月4,木3	090468	プログラミングB	村田　正幸	和文 英文
4	基礎工学部	秋学期	秋～冬学期	月4,木3	099026	プログラミングB	伊野　文彦	和文 英文
5	基礎工学部	春学期	春～夏学期	水1,金2	090469	プログラミングC	長谷川　亨	和文 英文
6	基礎工学部	秋学期	秋～冬学期	月3,水2	090470	プログラミングD	肥後　芳樹	和文 英文
7	基礎工学部	秋学期	秋～冬学期	火2	090543	化学工学プログラミング	内田　幸明	和文 英文

1件目 から **7件目**の検索結果を表示しています（全部で 7件あります）

図 2: シラバス検索結果画面の例

<<最終更新日：2022年01月15日>>

English

基本情報

時間割コード／Course Code	090467
開講区分(開講学期)／Semester	春～夏学期
曜日・時間／Day and Period	月3,木3
開講科目名／Course Name (Japanese)	プログラミングA
教室／Room	基/B102,情教2(豊研,情教3(豊研
開講科目名(英)／Course Name	Programming A
ナンバリング／Course Numbering Code	09CSSS2M300,09MASC2M300
必修・選択／Required/Optional	必修（情報科学科）
単位数／Credits	3.0
年次／Student Year	1,2,3,4年
担当教員／Instructor	若宮　直紀,矢内　直人
メディア授業科目／Course of Media Class	非該当

※メディア授業科目について
授業回数の半数以上を、多様なメディアを高度に利用して教室等以外の場所で行う授業を「メディア授業科目」としています。
学部学生が「メディア授業科目」を卒業要件に算入できるのは60単位が上限です。
なお、非該当の場合であっても、メディアを利用した授業を実施する場合があります。

授業担当教員一覧

詳細情報

授業サブタイトル／Course Subtitle	
開講言語／Language of the Course	日本語
授業形態／Type of Class	講義科目
授業の目的と概要／Course Objective	プログラミングの入門の講義と実習とからなる授業である。プログラムとはどういうものか、どのように書くのかを理解すること。プログラミングとはどのような作業なのか、どのようにエディットして、コンパイルして、実行させるのかなど、計算機の操作法を理解すること。そして、基本的な種々のプログラミング上の概念を理解し、それをプログラム作成演習で実際に使えるようになること、などを目的とする。演習では、サイバーメディアセンターのパソコンを用いて、手続き型計算機言語で基本的なプログラムを作成する能力を養う。計算機言語は、C言語を用いる。

図 3: 講義 “プログラミング A” のシラバス参照画面（一部抜粋）

2.2 オンラインジャッジシステム

オンラインジャッジシステムとは，ユーザから提出されたソースコードをコンパイル・実行した上で自動評価し，その結果をユーザに通知するサービスのことである．具体的には，事前に用意された検証データや検証プログラムを用いることで，コードの正しさやメモリ・実行時間の効率等を評価している．また，一般的には Web サイト上でプログラムの提出，検証を行うことができる．

オンラインジャッジシステムの一例として，国内で代表的な AIZU ONLINE JUDGE [2] が挙げられる．

2.3 プログラミングコンテスト

プログラミングコンテストとは，与えられた要件を満たすプログラムを記述する正確さや速さを競うコンテストのことである．複数の参加者が，定められた制限時間内に同一の問題を解き，オンライン上で解答のソースコードを提出する．提出されたソースコードはオンラインジャッジシステムによって採点され，コンテスト終了後，正解問題数や解答時間に応じて各参加者の順位が決定し，レーティングが変動する．

プログラミングコンテストの例として，国内では AtCoder[3]が，海外では Codeforces[4]や Topcoder[5] が挙げられる．

本報告では AtCoder を対象とするため，以下では AtCoder について説明する．

2.3.1 AtCoder

AtCoder はプログラミングコンテストを行うオンラインサービスであり，その規模は国内最大といわれる [7]．AtCoder の基本情報については下記のとおりである．

コンテストの分類と問題の構成　AtCoder で定期的に行われるコンテストは，主に以下の 3 種に分けられる．

1. AtCoder Beginner Contest（ABC）：初心者，初級者向けコンテスト
2. AtCoder Regular Contest（ARC）：中級者，上級者向けコンテスト
3. AtCoder Grand Contest（AGC）：上級者向けコンテスト

[2] https://judge.u-aizu.ac.jp/onlinejudge/
[3] https://atcoder.jp
[4] https://codeforces.com
[5] https://www.topcoder.com

各コンテストは難易度の異なる A 問題，B 問題，. . . といった複数の問題で構成されており[6]，それぞれ配点が異なる．

なお終了したコンテストについても，問題の閲覧や解答の提出が可能である．よってユーザはいつでもリソースにアクセスし，過去の問題で学習を行うことが可能となっている．

コンテストの流れ　AtCoder におけるコンテストには開始時間と終了時間が設定されており，以下の流れに従いコンテストが開催される．

1. **（開始前）**コンテストの参加希望者は参加登録をしておく．
2. **（開始時間）**全ての問題が公開され，参加者は問題文の閲覧と解答の提出が可能となる．
3. **（開始後）**参加者は次に示す流れに従い，終了時間まで問題に解答していく．ただし問題を解く順番は指定されていない．
   (a) 選択した問題を解き，作成した解答のソースコードをオンラインで提出する．なお，使用するプログラミング言語は自由に選択することができる．
   (b) オンラインジャッジシステムによる正誤判定が行われ，結果が参加者に通知される．
   - 正解した場合：問題の配点に応じた点数が獲得される．
   - 不正解だった場合：問題ごとに記録されている誤答の数が加算される．引き続き同じ問題に取り組んで解答を再提出したい場合は (a) に戻る．

   (c) 別の問題を選択し，(a) に戻る．
4. **（終了時間）**この時点で獲得済みの点数で順位が決まり，レーティングが変動する．ただし得点が同じ場合は下記に示すペナルティが加味され，“最終解答時間にペナルティを加算した時間” が短い方が順位が上となる．

$$(\text{ペナルティ}) = (\text{正答した問題に提出した誤答の数}) \times 5\ [\text{分}]$$

ユーザ数　AtCoder では，アクティブユーザを “2 年以内にレーティングが変動するコンテストに参加したユーザ” と定義している．2023 年 1 月 25 日現在，アクティブユーザは 105,064 人となっている[7]．

[6] 2023 年 1 月 25 日現在，基本的に ABC クラスは A 問題〜G 問題と Ex 問題，ARC クラス及び AGC クラスは A 問題〜F 問題で構成されている．どのクラスにおいても A 問題の難易度が一番低く，B 問題，C 問題，. . . と難易度が上がっていく．

[7] アクティブユーザ数は AtCoder サイト内の “ランキング” ページ（`https://atcoder.jp/ranking`）より確認可能である．

3　タグを用いたプログラミングコンテストの問題分類

本報告では，プログラミングコンテストの問題利用を検討するプログラミング学習者・教育者への支援を見据え，学習に向けた利用という視点から多数ある問題の実態を明らかにすることを目指す．

大量にある問題の傾向を把握する方法の一つに，タグ付けによる分類整理が挙げられる．Web 上のコンテンツに多数のユーザがタグを付与していくソーシャルタギング [3, 4, 6] は，その一例である．本報告では，タグと問題の間に生じる多対多の関係について，タグから問題，及び，問題からタグ，の 2 つの観点で対応を見ることで全体像を掴むことを考える．このとき，付与するタグの内容はプログラミング学習に関するものが望ましいため，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を把握できれば，自身の未学習単元が何であるかの認識が可能となる．それが明らかになれば，不足している能力を効果的に補うために，未学習単元であるタグが付与された問題に特化して学習する，ことが考えられる．また，ある単元 X の学習を検討している際，単元 X に分類される問題の数に着目して，単元 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．

目的達成のためのアプローチは，**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**という形になるが，問題の分類に先立って以下を明示する必要が生じる．

1. 本報告では何を学習単元とするか．
2. 本報告では何をタグとするか．

そこで全体の見通しを良くするため，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ることとした．

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 4 に示す．まず RQ1 については，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調査・分析することで回答し，学習単元の対象を明示する．プログラミング教育の現場としては，大学の情報系学科に着目した．続いて RQ2 については，タグの設計方針を定め，それに適合する学習単元を選択することで回答し，問題に付与するタグの全体像を明示する．最後に RQ3 については，プログラミングコンテストの問題

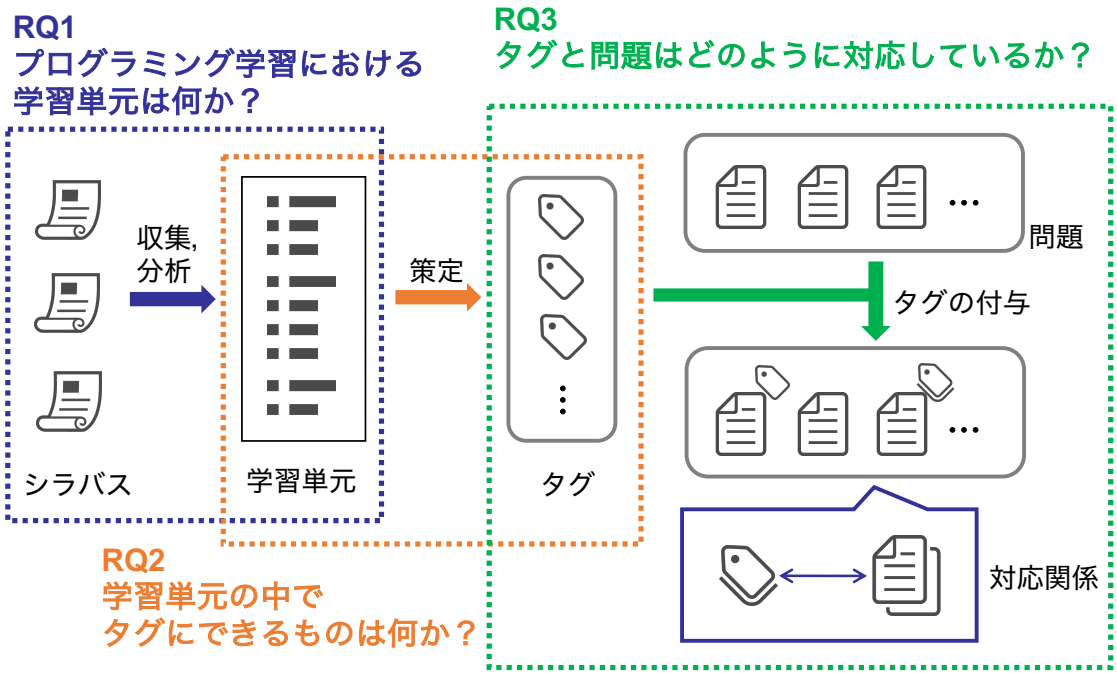


図 4: 各 RQ への回答の流れ

へタグを付与し分類することで回答し，タグと問題の対応関係を明らかにする．ここで分類の対象は AtCoder Beginner Contest（ABC）の問題とした．

以下では，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．

3.1 RQ1：プログラミング学習における学習単元は何か？

RQ1 へは，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調査・分析することで回答し，プログラミング学習における学習単元を明示する．

3.1.1 対象

本報告ではプログラミング教育の場として，大学の情報系学科に着目した．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．ただし私立大の学部・学科は国立大に比べて数が多く，構成が複雑であることから，対象を国立大学の情報系学科に限定することとした．

また，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 [13] 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．これは，一部学科において改組・廃止が行われていたためである．なお，作成した情報系学科リスト

は付録 A に示している．

3.1.2 アプローチ

RQ1 への回答アプローチは 3 ステップに分かれる．各ステップの詳細について以下で述べる．

STEP1：シラバスデータの収集　大学の情報系学科におけるプログラミング教育のシラバスを逐次参照し，教育内容に関するデータを収集する．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をプログラミング教育の対象とし，以下に該当するものは対象外とした．

- 問題解決・目的達成のため “手段” としてプログラミングを扱う講義.（実験，PBL，グループワーク，アプリ開発など）
- すでに基礎知識を習得済みであることを前提とし，高度な開発を目的とする講義.（Web プログラミング，システムプログラミングなど）
- コンピュータ操作に慣れることのみを目的とした，入門系の講義.

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，シラバスを参照した．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等を参照して判断した．それに加え，“プログラミング” 等で講義名検索やキーワード検索を行う・開講学科で絞り込む等で調査漏れがないか確認した．

各シラバスにおける収集項目と収集内容を表 2 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，形式を統一化して収集した．

上記に従い，59 の情報系学科から，316 件のデータを収集した．

表 2: 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する.
授業の目的	授業の概要やその目的・ねらいに関する記述全般を収集する.
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する.
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する.
キーワード	シラバス中に記載がある場合のみ収集する.

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的な手法である LDA（Latent Dirichlet Allocation） [1] を適用して分析を行う．トピックモデルとは自然言語処理モデルの 1 つで， 大量の文書データからトピックを抽出する手法の総称である．トピックは複数の単語の共起性によって得られる潜在的な意味のカテゴリであり，構成単語をもとにその解釈を行う．トピックモデルを用いることにより，文書全体にどのようなトピックが含まれているかに着目して特徴把握が可能となることに加え，各文書がどのようなトピックから構成されているかについても推定が可能となる．よって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．

分析対象とするテキストデータは，STEP1 で収集した各シラバスの項目のうち，**授業計画**および**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いためである．授業計画には各回における授業内容が箇条書きあるいは表形式で記載されており，キーワードには講義の主題に関する単語が羅列されている．

まず，分析精度の向上や処理効率の上昇を目的とし，以下の手順で前処理を施した．

1. **単語の分割**：形態素解析により名詞のみを抽出する．形態素解析ツールには MeCab[8] を用い，参照辞書には，単語分かち書き用辞書生成システム NEologd [10] の提供する mecab-ipadic-NEologd [9] [11] を用いた．

2. **単語の正規化**：以下のルールに従い，表記の異なる同義の単語をできるだけ統一化する．

   (a) 大文字・小文字：小文字で統一．

   (b) 半角・全角：英数字は半角で，カタカナは全角で統一．

   (c) 表記ゆれ：Unicode 正規化により統一．正規化形式には NFKC を用いた．

   (d) 数値の置き換え：0 で統一．

3. **ストップワードの設定**：教育内容の特徴把握においてノイズとなりうる単語を，処理対象外とする．方式は辞書による設定を採用し，以下の単語をストップワードとした．

   - SlothLib [14] が提供する日本語のストップワード一覧[10] に記載された単語．
   - 講義における一般的な単語．（課題，演習，試験，予定，計画 など）
   - 収集対象の特徴から明らかな単語．（プログラミング など）

[8] https://taku910.github.io/mecab/

[9] https://github.com/neologd/mecab-ipadic-neologd

[10] http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt

- その他，数値や人名など．

続いて Python ライブラリ scikit-learn [11] のパッケージを使用し，トピック数を様々に変更しながら LDA による分析を実行した[12]．また，実行結果は Python ライブラリ pyLDAvis [13] を用いて可視化した．トピック数を 7 とした場合の実行結果を例に，pyLDAvis による可視化の様子を図 5 に示す．

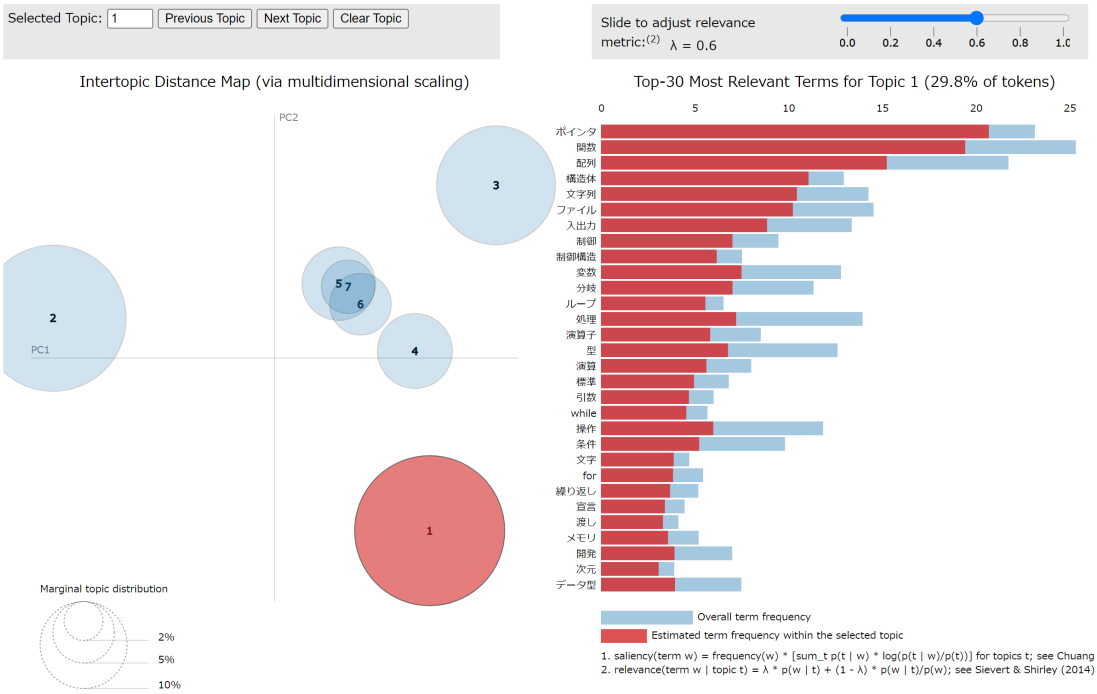


図 5: pyLDAvis による LDA 実行結果の可視化（トピック数 7 の場合）

図 5 の左側には，各トピックを多次元尺度構成法（MDS: Multi-Dimensional Scaling）により 2 次元に簡約し，円として配置した結果が表示されている．円の大きさは各トピックに属する文書（ここではシラバス）の合計を，円どうしの距離はトピック間距離を表す．各トピックはクリックによる選択が可能で，選択中のトピックは赤色でハイライトされる．この例ではトピック 1 が選択状態にある（図 5 中央下）．また図 5 の右側には選択中のトピックを特徴づける単語が表示されており，トピックの解釈に活用可能である．各単語の横には以下の 2 つの観点に基づく出現頻度が横棒グラフ形式で記載されている．

- 観点 1：文書全体におけるその単語の出現頻度.（青色）
- 観点 2：選択トピック中におけるその単語の出現頻度.（赤色）

[11] https://github.com/scikit-learn/scikit-learn
[12] LDA の実行時は，ユーザが総トピック数を指定する．
[13] https://github.com/bmabey/pyLDAvis

pyLDAvis には，2 観点のどちらによる順位付けを重視して単語を表示するか，その比率を指定する “relevance” というパラメータが導入されており，ユーザは図 5 右上に設けられたスライダーを用いることで，0 から 1 の範囲でその値を設定できる．この例では，relevance は 0.6 に指定されている．なお，0 に設定すれば観点 1 のみによる順位付けで，逆に 1 に設定すれば観点 2 のみによる順位付けで単語が表示される．

まずトピック数を 2〜10 に変更してトピックの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．トピック数 7 の場合の可視化結果を例として，その様子を図 6 に示す．

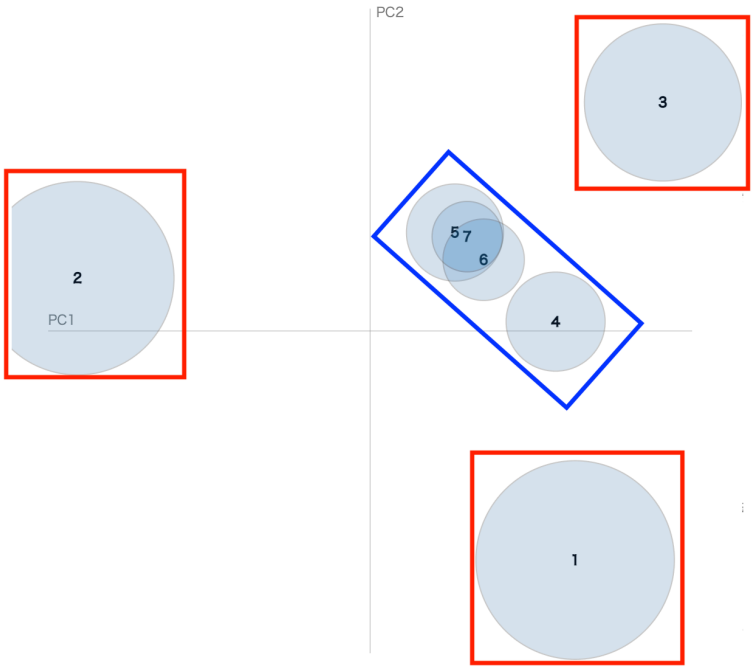


図 6: トピック数 7 の場合におけるトピックの分布傾向（赤枠が 3 つの大トピックを，青枠がその他の小トピック群を示す）

また，3 大トピックと小トピック群はそれぞれ以下のような傾向を持つ．

- 3 大トピック（図 6 赤枠）
  - 円のサイズが大きい（多数のシラバスがそのトピックに属する）．
  - トピック数によらず，互いに分離して配置されている（トピックの類似度が低い）．
- 小トピック群（図 6 青枠）
  - 円のサイズが小さい（そのトピックに属するシラバスが少ない）．

– 互いに比較的近い位置にある（トピックが類似している）.
– 表 3 に示すような単語を含む場合が多い．これらはプログラムの実行環境に関連するものと考えられる.

表 3: 小トピック群に含まれる単語の例

unix	シェル	端末	システム	コンパイラ
仮想	バイナリ	環境	コマンド	コンソール

なお，トピック数 11〜30 における実行結果も確認したところ，複数の小トピックが 3 大トピックのすぐ付近でクラスタ化される場合があったものの，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（群）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.

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，これらのトピックに含まれる単語は，様々な講義で共通して扱われる教育項目であるとみなせる．また各トピック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．したがって本報告では，3 大トピックをもとにカテゴリを策定するとともに，それぞれの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画面右側に表示される上位 30 の単語を記録する．このとき，パラメータ relevance の値は 0.6 に設定する.
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. 各カテゴリ内の単語を整理し，主要単元群とする．具体的には以下を実行した.
    - 同義の単語をまとめる.（“ソート” と “整列” など）
    - 包含関係にある単語を整理する.（“データ構造” と “スタック” など）

なお，手順 2 においてパラメータ relevance の値を 0.6 としたのは，開発者によるユーザ調査の結果，トピックの解釈において 0.6 が最適値であるという結果が得られている [5] ためである.

3.1.3 回答

3.1.2 節に示したアプローチにより，プログラミング学習の学習単元を得た．RQ1 の回答として，得られた単元の一覧を表 4 に示す.

表 4: 学習単元の一覧（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）

(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	

3.2 RQ2：学習単元の中でタグにできるものは何か？

RQ2 への回答は，まずタグの設計方針を定め，RQ1 で得られた学習単元（表 4）のうち，方針に適合するもののみを選別することで実現する．これにより，問題に付与するタグの全体像を明示する．

3.2.1 アプローチ

まず，以下の通りタグの設計方針を定めた．

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．
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．
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）．

分類結果が自明なタグは，分類により新たな知見が得られずノイズになりうると判断し，方針 1, 2 を設けている．また表 4 の学習単元は階層構造を持つため，そのままタグに反映するとタグ間に包含関係（依存関係）が生じ，分類結果の把握が困難になることが予想される．よって方針 3 を設けた．

続いて，上記の設計方針に従いタグを策定する．その際，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．

方針 1 への対応　全ての問題を解くにあたって必ず用いる技術について，関連する単元を除外する．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することで解答することが前提となっている．したがって演算・変数・入出力といったカテゴリ 2（表 4b）の一部単元に関しては，全ての問題で要求されるとし，タグの策定対象から外した．

方針 2 への対応　プログラミングコンテストの問題で用いることが想定されていない技法について，関連する単元を除外する．プログラミングコンテストの問題は様々な言語で解答可能であり，その中には手続き型言語も含まれる．よってオブジェクト指向特有の技法が問題として要求されることはないとし，オブジェクト指向言語にまつわる単元として，カテゴリ 3（表 4c）全体をタグの策定対象から除外した．

方針 3 への対応　階層構造を崩し，抽象度を高める方向で統一を行う．表 4 における第 1 レベル[14]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 レベルの単元

[14]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 つとしている．

を策定対象にすることとした．また，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整理を行った．

3.2.2 回答

3.2.1 節に示したアプローチにより，問題に付与するタグを得た．RQ2 の回答として，策定したタグの一覧を表 5 に示す．各カテゴリは表 4 のものを踏襲している．

表 5: 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	

3.3 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？

RQ3 へは，RQ2 で得られたタグ（表 5）を問題に付与し，分類を行うことで回答する．タグと問題に生じた多対多の関係から，問題の難易度別に見たタグの分布傾向，及びタグごとに見た問題の分布傾向の 2 つを軸に分析を行う．

3.3.1 分類対象

本報告では，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することとした．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．

3.3.2 アプローチ

RQ3 への回答ステップは大きく 2 つに分かれる．各ステップの詳細について以下で述べる．

STEP1: 参照情報の収集　タグ付与の判断にあたって必要となる，問題の特徴を表す情報を収集する．本報告では，分類対象の各問題に付随する以下の 3 情報に着目した．

1. **問題文**
   問題設定や制約，入出力仕様，入出力の例などが記載されており，ユーザはこれらの情報をもとに解答を行う．“問題として意図されている” 情報が得られることを期待した．

2. **解説文**
   問題の解答方針や詳細手法，解答にあたって求められるテクニック等が記載されている．執筆は AtCoder 公式により行われており，“問題として意図されている” 情報が問題文よりも明示的に得られることを期待した．

3. **解答コード**
   問題への解答としてユーザが提出したソースコードであり，“解答にあたって実際に用いられている” 情報が得られることを期待した．本報告では，C 言語で書かれ，かつ問題に正解したもの全てを解答コードの対象とする．

問題文・解説文については，問題ページ及び解説ページへアクセスし，スクレイピングによりテキストデータの収集を実行した．ただし公式解説が複数ある場合は，解説一覧ページの最も上にあるリンクを採用した．

また解答コードに関しては，AtCoder Problems [15] で公開されている提出情報ファイルを基にソースコードページへアクセスし，スクレイピングによりソースコードを収集した．AtCoder Problems とは有志により作成された Web アプリで，AtCoder に提出されたソースコードをクロールして管理している．提出情報ファイルには提出 ID や問題 ID，言語，提出結果といった情報が含まれており，問題 ID・言語・提出結果で絞り込んだ提出情報について，提出 ID とコンテスト ID を基にソースコードページにアクセスすることが可能となる．

上記に従い，2022 年 12 月 18 日時点における最新 108 コンテスト [16] の A〜C 問題（計 324 問）について，問題文・解説文・解答コードを収集した．

STEP2：タグの付与　表 5 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していく．タグの付与方針はカテゴリ 1 のタグ（表 5a）とカテゴリ 2 のタグ（表 5b）で大別されるため，以下ではそれぞれに分けて述べる．

カテゴリ 1 のタグ付与方針　“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” 場合，その問題はタグ X の学習に適しているとし，タグ X を付与する．問題の意図を読み取るにあたり，問題文・解説文を参照する．

[15] https://kenkoooo.com/atcoder/

[16] 解説が HTML 形式で統一されていたため，このような区切りを設けた（それ以前のコンテストでは，解説が PDF 形式であった）．

なお，カテゴリ 1 に属するタグはアルゴリズム等に関する抽象的な概念であり，同様のことを実現していても，その実装が異なる場合が考えられる．よって，あるパターンに当てはまるか否か，という単純な尺度で解答コードの傾向を掴む方法は有効でないと判断し，解答コードは参照しないこととした．

ある問題に対し，カテゴリ 1 に属するあるタグ X を付与するかの判断は，大まかに以下の流れに従う．

1. 問題文・解説文にタグ X のキーワードが含まれるか検索する．タグごとに設定したキーワードは表 6 の通りである．
2. キーワードがヒットした場合，問題文・解説文の内容を確認し，タグ X に関連する単元の利用意図が読み取れればタグ X を付与する．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．

カテゴリ 2 のタグ付与方針　“タグ Y に関連する単元を利用して問題に解答したユーザが多い” 場合，その問題はタグ Y の学習に適しているとし，タグ Y を付与する．解答の傾向を把握するにあたり，各解答コードを参照する．

カテゴリ 2 に属する単元は一部言語に依存したものである一方，プログラミングコンテストの問題はその仕様上，解答言語が指定されることはない．したがってカテゴリ 2 の単元について，その利用意図を問題文や解説文から読み取ることは困難であると判断し，問題文及び解説文は参照しないこととした．

ある問題に対し，カテゴリ 2 に属するタグ Y を付与するかの判断は，大まかに以下の流れに従う．

1. 問題の解答コードを逐次参照し，タグ Y のマッチ条件を満たすコード数を集計する．タグごとに設定したマッチ条件は，表 7 に示す通りである．
2. マッチ条件を満たすコード数が解答コード全体の過半数であった場合，タグ Y を付与する．

各解答コードの参照は機械的な手法を用いた．その詳細は次のとおりである．

まず，解答コードから AST（抽象構文木）を作成する．これには Python ライブラリ Clang Python Bindings [17] を使用した．

続いて AST を先行順に探索し，到達したノードをリスト化する．このとき，以下のルールを設けた．

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

表 6:（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 7:（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か
ループ	for 文，while 文，do-while 文のいずれかが使用されているか
構造体	構造体が宣言されているか
ポインタ	ポインタが宣言されているか
配列	配列が宣言されているか（ただし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，さらにそれがプログラム実行時に呼び出されるか
メモリ	malloc, calloc, realloc 関数のいずれかが呼び出されているか

1. ライブラリなどの外部ファイル内を参照しているノードは，それを根とした部分木の探索をスキップする．

2. 未使用関数のノードを根とした部分木の探索をスキップする．

上記ルールの適用により，コード全体のうち以下の部分のみを参照することが可能となる．

- グローバル宣言部
- 使用関数の内部

これらは解答コードの実行にあたって必要とされた要素のため，タグ付与の判断材料となる．なお，使用関数及び未使用関数とは，以下の手順で特定される関数とする．

1. 図 7a に示すような，コード中で宣言された関数の呼び出し関係を表す木を作成する．この例では main 関数から関数 A と関数 B が，関数 B から関数 C と関数 D が，そして関数 E から関数 F が呼び出されている．

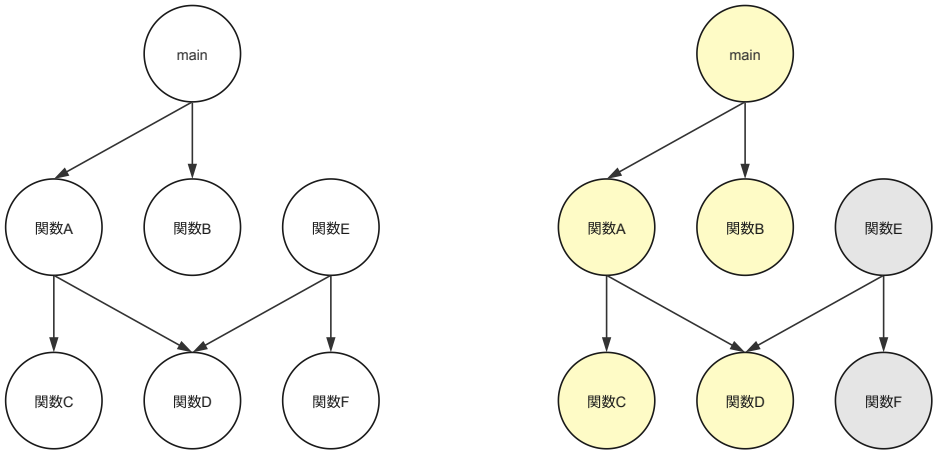
2. main 関数から辿って到達可能な関数を使用関数とし，そうでない関数を未使用関数とする．図 7a における使用関数と未使用関数を特定した様子を，図 7b に示す．

最後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ものとして扱う．ただし**関数**タグに関しては，main 関数以外に使用関数が 1 つでもあった場合にマッチ条件を満たすものとする．

3.3.3 回答

3.3.2 節に示したアプローチに従い，AtCoder Beginner Contest の A・B・C 問題に対しタグ付けを行った．タグ付けの対象は各難易度ごとに 108 問ずつ，計 324 問である．RQ3 の回答として，その付与結果を以下に述べる．なお，それぞれの結果を踏まえた詳細な考察については 4.3 節で行う．

はじめに，結果の全体像を示す．表 8 は，各タグが付与された問題数と全体に占める割合を表す．なお，合計して 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ことに起因する．また，全体の分布を表したものが図 8 である．



(a) 関数の呼び出し関係を表す木

(b) 特定した様子（黄色のノードが使用関数を，灰色のノードが未使用関数を示す）

図 7: 使用関数及び未使用関数の特定

表 8: 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%

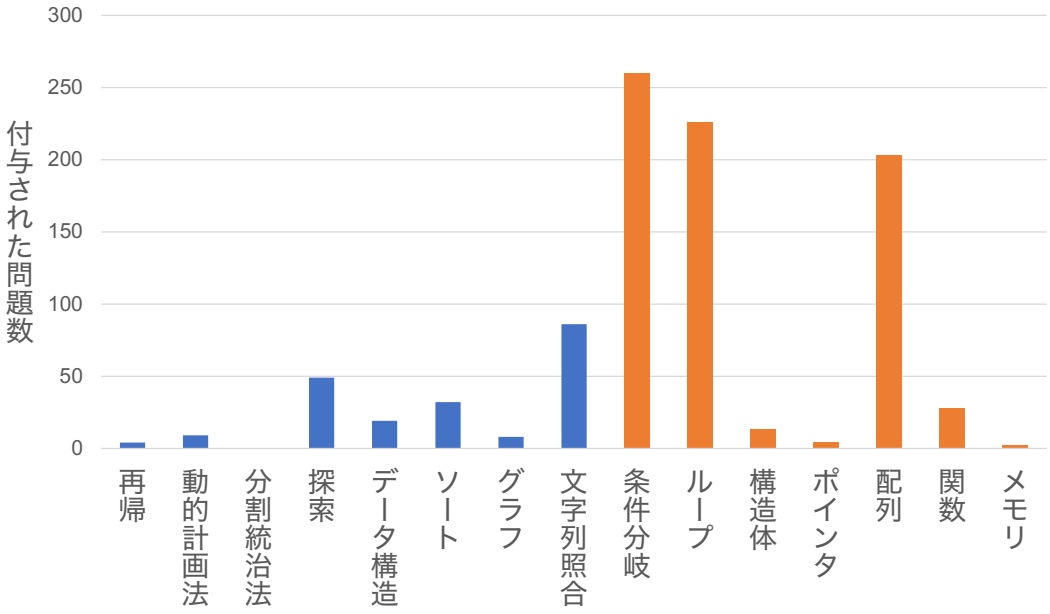


図 8: 各タグが付与された問題数の分布（青はカテゴリ 1 のタグを，橙はカテゴリ 2 のタグを示す）

問題とタグの間に生じる対応関係を示に先立ち，難易度ごとにドメインを区切り，各タグが付与された問題数とドメイン内に占める割合を集計した．その結果が表 9〜11 である．なお，合計して 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ことに起因する．

表 9: （A 問題）各タグが付与された問題数とドメイン内に占める割合

(a) カテゴリ 1

タグ名	問題数	割合
文字列照合	31	28.7%
ソート	5	4.6%
探索	3	2.8%
データ構造	1	0.9%
再帰	1	0.9%
動的計画法	0	0.0%
分割統治法	0	0.0%
グラフ	0	0.0%

(b) カテゴリ 2

タグ名	問題数	割合
条件分岐	71	65.7%
配列	28	25.9%
ループ	26	24.1%
関数	1	0.9%
構造体	0	0.0%
メモリ	0	0.0%
ポインタ	0	0.0%

表 10: （B 問題）各タグが付与された問題数とドメイン内に占める割合

(a) カテゴリ 1

タグ名	問題数	割合
文字列照合	31	28.7%
探索	12	11.1%
ソート	8	7.4%
データ構造	6	5.6%
グラフ	5	4.6%
動的計画法	2	1.9%
再帰	0	0.0%
分割統治法	0	0.0%

(b) カテゴリ 2

タグ名	問題数	割合
ループ	98	90.7%
条件分岐	92	85.2%
配列	83	76.9%
構造体	2	1.9%
ポインタ	1	0.9%
メモリ	1	0.9%
関数	1	0.9%

表 11:（C 問題）各タグが付与された問題数とドメイン内に占める割合

(a) カテゴリ 1

タグ名	問題数	割合
探索	34	31.5%
文字列照合	24	22.2%
ソート	19	17.6%
データ構造	12	11.1%
動的計画法	7	6.5%
グラフ	3	2.8%
再帰	3	2.8%
分割統治法	0	0.0%

(b) カテゴリ 2

タグ名	問題数	割合
ループ	102	94.4%
条件分岐	97	89.8%
配列	92	85.2%
関数	26	24.1%
構造体	11	10.2%
ポインタ	3	2.8%
メモリ	1	0.9%

最後に，問題とタグの間に生じる対応関係を示す．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 9 及び図 10 に示す．また，タグから問題への対応として，各タグについて，タグが付与された問題数の難易度別分布を図 11 に示す．

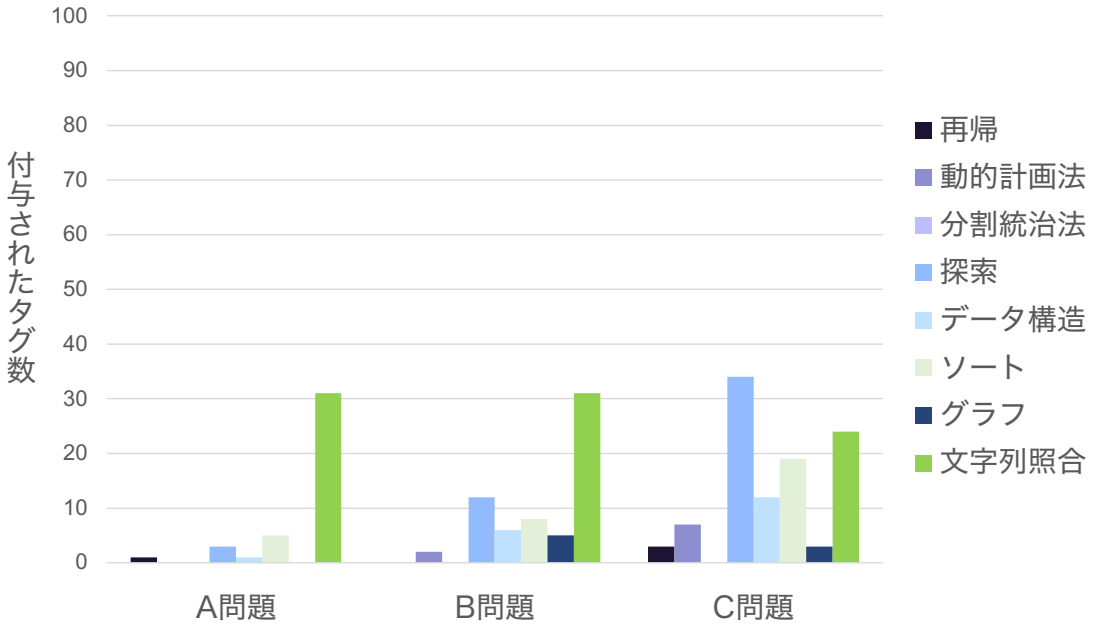


図 9: 各難易度の問題に付与されたタグの種類別分布（カテゴリ 1）

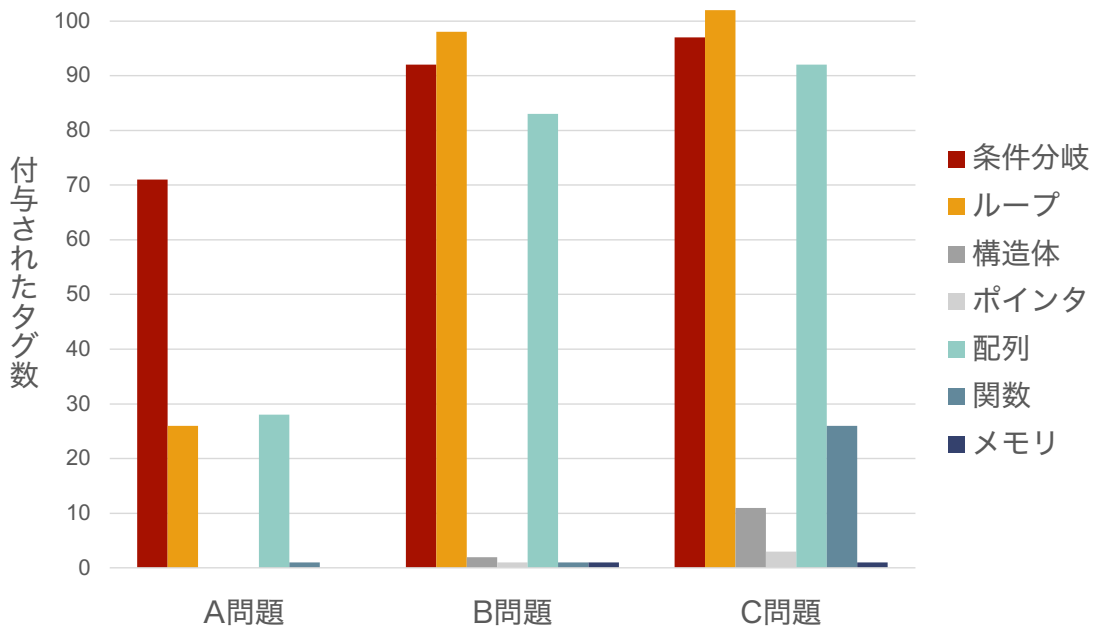


図 10: 各難易度の問題に付与されたタグの種類別分布（カテゴリ 2）

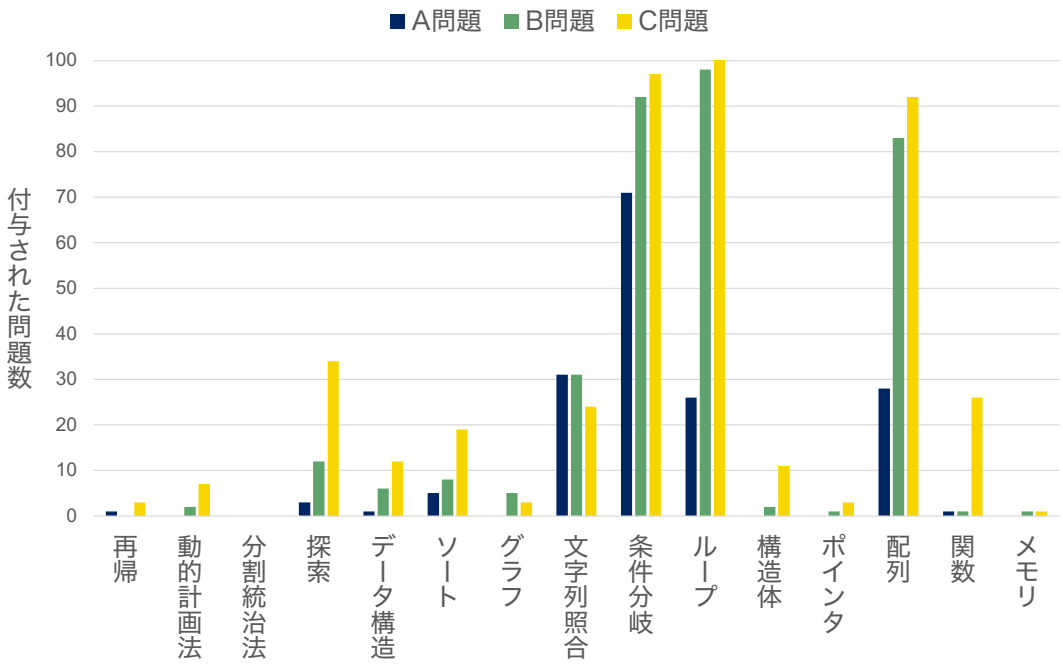


図 11: 各タグが付与された問題数の難易度別分布

4　考察

4.1　学習単元の各カテゴリの解釈

RQ1 で得られた学習単元（表 4）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といえる．これは単元のカテゴリのもととなった 3 大トピック（図 6）の解釈でもある．実際のシラバスを確認したところ，カテゴリ 1 及びカテゴリ 2 に属する項目の両者を扱う授業も見られた．よってこれら 2 つのカテゴリは，必ずしも完全に分割され独立した授業で扱われるわけではなく，互いに関係していることがうかがえる．これは，分析の際に適用した LDA が，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ことに起因すると考えられる．

4.2　タグの策定方針

RQ2 ではタグの策定方針として，明らかに全ての問題に付くタグを除外した．この設計は，そういったタグは分類の際のノイズになる，との考えによるものである．しかし別の視点で考えると，ある問題 X に着目したときに以下を区別できない．

- 本来全問題に付くため問題 X にも付くが，タグ集合から除外されているため付いていないタグ
- 単に問題 X に付いていないタグ

すなわち，問題を単体として見たとき，それが有する属性をタグで網羅できない．例えば任意の問題を取ってきたときに，それは**入出力**の単元に関する属性を持つが，**入出力**というタグは（付与対象から除外されているため）付いていない．本報告では，タグと問題の対応関係から問題全体の傾向を知るためにこのような形としたが，ランダムに問題を抽出しその問題単体が持つ属性を知る，といったユースケースを考える場合は，タグの設計方針を変更する必要がある．

また，タグ策定の際には抽象化レベルを上げる方向で統一を図ったが，逆により詳細な区分を設けてタグを設計することも考えられる．その利点として，用いられる技法が明確になり，ピンポイントでの学習に役立つ可能性があることが挙げられる．一方でタグ数そのものが増加すると，全体像の把握が困難になるという欠点も存在すると思われる．

4.3 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を掴み，それらが生じた理由の考察や，利用検討者に向けた問題利用に関する情報の提示を行う．以下では表 5 のタグのうち，カテゴリ 1 に属するものを “アルゴリズムに関するタグ”，カテゴリ 2 に属するものを “基礎事項に関するタグ” として扱う．

タグの付与傾向に見られる難易度ごとの問題の特徴　まず図 9 より，カテゴリ 1 のタグの付与傾向は (1) A・B 問題と (2) C 問題の 2 つに大別される．前者は全体としての付与数が少なく，**文字列照合**とそれ以外のタグで付与数の差が生じているのに対し，後者では全体の付与数が増加し，その中で最多のタグは**探索**となっている．また，**ソート**タグの付与数も増えている一方で，**文字列照合**タグが少々付与されづらくなっている．すなわち，A・B 問題に比べ，C 問題にはアルゴリズムに関する問題が多く含まれ，特に**探索**や**ソート**を用いる問題の増加により上記の傾向が生まれているといえる．

次に図 10 より，カテゴリ 2 のタグの付与傾向としては，(1) A 問題と (2) B・C 問題の 2 つに分かれる．前者は**条件分岐**タグのみが突出して多く，次いで**ループ**と**配列**が同程度に付与されている一方で，後者はこれら 3 つのタグの付与数が同程度かつ顕著に見られる．また，A 問題はそれ以外に比べ，全体としてのタグ付与数が少ない．そこで実際に問題や解説を確認したところ，A 問題はその基礎事項を利用し，習得することを目的とした問題が多いのに対し，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問題が多い，という傾向の違いが見られた．すなわち，難易度の上昇とともに各問題で要求される処理が複雑化していることから，A 問題に比べ B・C 問題には基礎事項に関する問題が多く含まれ，特に**ループ**及び**配列**を用いる問題が急増したと考えられる．なお，**関数**及び**構造体**タグの付与数変化という点では，B・C 問題の間にも差異は見られる．

付与された問題の難易度傾向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傾向（図 11）から，タグの種別は大まかに以下に分かれる．

1. A〜C 問題それぞれが同程度の割合のタグ.（例：**文字列照合・条件分岐**）
2. B・C 問題の割合が高いタグ.（例：**ループ・配列**）
3. C 問題のみ割合が高いタグ.（例：**探索・ソート・構造体・関数**）
4. その他.（付与数そのものが少ない等）

問題の難易度と各タグの属するカテゴリに着目して考えると，まず 1 に関連する単元は，難易度によらず，一般に求められる基本技術であるといえる．学習難度自体は低く，初学者向

きの単元とも解釈できる．

続いて 2 に関する単元は，ある程度複雑な処理を行うにあたり必須となる技法であると考えられる．B 問題では座標計算や数式・不等式の処理技術が，C 問題以降ではさらに複雑な数式やアルゴリズムの実現が求められるため，それらの問題を解くにあたって必要な手段として用いられ，このような傾向が生じたものと解釈できる．プログラムがある程度複雑になれば，条件分岐やループといった技術は当然必要になるであろう．単元としての難易度は中〜高といえ，ある程度の経験者であれば学習可能であると考えられる．

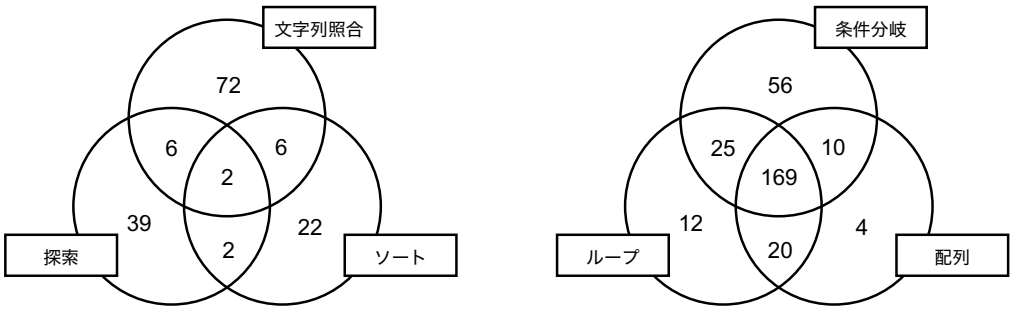
最後に 3 に関連する単元は，難易度が高く，アルゴリズムの使用や複雑な実装を求められたときに必要となる技術であると考えられる．傾向が生じた原因として，C 問題ではより複雑な解答が求められることが挙げられる．例えば，解答プログラム自体の規模が大きくなり，main 関数以外の関数に処理を分割したり，構造体の導入によりデータを適切に管理したりしないとうまく解けないような問題が増加したといえる．学習難度が高いため，初学者の学習には向かない単元である．

頻出タグの付与関係に見られるカテゴリの特徴　表 8 及び図 8 より，カテゴリ 1 及びカテゴリ 2 全体で付与傾向に差が生じていることが読み取れる．カテゴリ 1 のタグは全体として付与された問題数が少なく，カテゴリ中では最も多くの問題に付与された**文字列照合**タグも，全体の 4 分の 1 程度に留まっている．一方でカテゴリ 2 のタグは全体として付与された問題数が多いものの，二極化が見られた．**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 割を切っていることが分かる．

このような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．各カテゴリの頻出 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 12 に示す．1 つの円が 1 つのタグに対応し，円内部の数字はそのタグが付与された問題数を表す．各円の共通部分に記載された数字は，それらのタグが同時に付与された問題数である．

まずカテゴリ 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見られることから，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 問の半数を上回る．なお，その 169 問の



(a) カテゴリ 1　　**(b)** カテゴリ 2

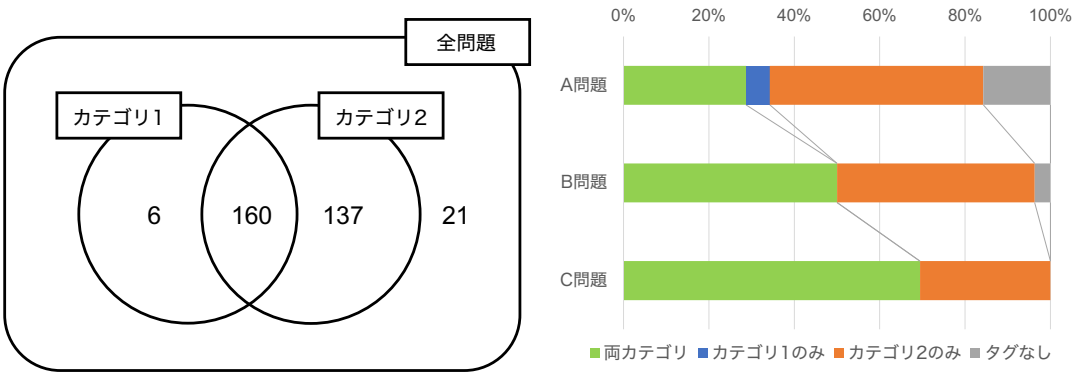
図 12: 各カテゴリにおける頻出 3 タグの付与有無に着目した問題数の分布

うちおよそ半数の 83 問が C 問題であり，次いで B 問題は 75 問，A 問題は 11 問に留まった．すなわち，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．対して，**条件分岐**タグに関しては，独立部分が 56 問と他の 2 タグと比べて多く，そのうち 44 問が A 問題であった．よって，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ものと考えられる．

カテゴリ間の付与関係に見られる問題の特徴　2 つのカテゴリ間の関係を知るため，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，その問題数の分布を調査した．全体の分布，及び難易度ごとの付与パターンの割合を示したものが図 13 である．

全 324 問のおよそ半数に両カテゴリのタグが付与され，残り半数弱にはカテゴリ 2 のタグのみが付与されており，これら 2 パターンの問題が全体の大多数を占めることが分かる．すなわち，分類した問題の性質は大きく以下の 2 種に分けられる．

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．



(a) 問題全体の分布　　**(b)** 難易度別の 4 パターンの割合

図 13: カテゴリごとのタグの付与有無に着目した問題数の分布

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.

前者は 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のに対し，後者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．これは，問題の難易度の上昇とともに用いる技法が複雑化するという点で，妥当な結果であるといえる．また，単純な基礎事項のみを学習したい場合は前者の問題に，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後者に着目することで，目的の問題を入手しやすくなると考えられる.

ここで，少数ながら存在した残りの 2 パターンについて，どのような問題があったかを以下に記す.

1. カテゴリ 1 のみが付与された問題
    - 解説にて別解としてアルゴリズムを用いた解法が紹介されていたものの，実際に解答した人の大半は単純な演算のみで解答した問題.
2. タグが付与されなかった問題
    - 入力値の型を変換するだけの問題.
    - 入力値を直接計算して出力するだけの問題.

なお，タグが付与されなかった問題が生じたのは，RQ2 にて全ての問題に付与されるタグを除外したことに起因する.

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 のタグを調べることで，それぞれのアルゴリズムを実現する際に，どのような基礎事項が重視されるのかが分かると考えられる．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 12 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**及び**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といえる．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持つことを確認できた.

表 12: カテゴリ 1 の頻出 3 タグに共起するカテゴリ 2 のタグとその割合

	共起タグ 1	共起タグ 2	共起タグ 3
文字列照合（86 問）	配列（87.2%）	条件分岐（80.2%）	ループ（74.4%）
探索（49 問）	条件分岐（89.8%）	ループ（89.8%）	配列（77.6%）
ソート（32 問）	配列（96.9%）	ループ（90.6%）	条件分岐（87.5%）

4.4　問題を利用する際の方針

これまでの考察を踏まえて得られた，AtCoder Beginner Contest の A〜C 問題をプログラミング学習に利用する際の方針について，複数の観点に分けてそれぞれ提示する．

各難易度の傾向

- A **問題**（難易度低）
  基礎事項の習得を目的とした単純な問題が多く見られ，それらの技術を単体で用いて解答するケースが大半である．アルゴリズムに関する単元を扱う問題はあまり見られない．

- B **問題**（難易度中）
  ある程度複雑な処理が求められるようになり，複数の基礎事項を組み合わせて解答することが必要となる．アルゴリズムに関する単元については，A 問題と同程度である．

- C **問題**（難易度高）
  アルゴリズムに関する問題が大きく増え，B 問題よりも多くの基礎事項を組み合わせた解答が求められる．

- 初学者などは，難易度の低い問題を選択することで複雑な問題を回避でき，単純な基礎事項の学習に専念できる．

- 同じ問題数で少しでも多くの単元を学びたい場合は，難易度の高い問題に優先して取り組むといい．

アルゴリズムに関する単元

- **文字列照合**
  どの難易度でも比較的多く問題が手に入るため，習熟度に応じた選択が可能であり，学習に適している．文字列の完全一致といった単純な照合でまず練習する場合は A 問題を，複雑なパターンマッチに取り組む場合は B・C 問題を選択するとよい．

- **探索・データ構造・ソート**
  C 問題で比較的多く入手できるが，A・B 問題にはあまり見られない．単元としての難易度が高く，初学者にはあまり適さない．C 問題に限れば問題量は確保できるので，経験者の学習には適する．

- **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く，学習にはあまり適さない．

- 様々なアルゴリズムに関する学習をまとめて行いたい場合，C 問題を重点的に学習すると効果的である．
- アルゴリズムに関す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．何か 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，複数のアルゴリズムを一括で学べるような問題はあまり含まれない．
- アルゴリズムに関する問題を学習する際は，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

- **条件分岐**
  難易度を問わず大量の問題が手に入るため，学習に非常に適している．特に A 問題の多くはそれ単体での学習が可能であるため，初学者は A 問題を用いれば効果的である．**ループ**等と組み合わせ，複雑な条件分岐処理の実践に取り組みたい場合は，B・C 問題を選択すればよい．
- **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であり，学習に適している．**条件分岐**同様，習熟度に応じた難易度選択が可能である．
- **関数・構造体**
  難易度の低い問題がほとんど手に入らず，初学者には向かない．一方 C 問題では一定数手に入るため，難度の高い学習単元である．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．すなわち，それらの単元は本報告における分類対象での学習には適さない．
- **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ため，学習状況や学習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．A 問題ではその技法を用い，習得することを目的とした問題が多いのに対し，B・C 問題では複雑な数式やアルゴリズムを実現する手段とされている問題が多い．
- 基礎事項に関する単元のみの学習にまとめて取り組みたい場合，A 問題を重点的に学習するとよい．

- 基礎事項に関する問題をまとめて入手した際，その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．一方，どれか 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．

4.5 妥当性への脅威

4.5.1 内的妥当性

シラバスデータの収集方法　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の記法が統一化されていない状況により，すべての収集作業を手動で行うこととなった．カリキュラムマップの確認によるプログラミング教育の把握や，シラバスに記載された項目の収集に関して，判断が主観的である可能性がある．完全な機械化によるアクセスが実現できれば，恣意性を取り除き，より統一性を持たせた収集が可能になるものと考えられる．

シラバスデータの分析方法　RQ1 ではシラバスデータの特徴把握にあたり，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，その事前処理や可視化方法によって結果が大きく左右されることを否定できない．例えば前処理の形態素解析の段階で固有名詞が分断される等，本来の表記意図と異なる単語がコーパス中に含まれる割合が高くなると，分析の精度が下がってしまう．可視化方法についても，多次元尺度法の種類により受ける印象が異なることが考えられる．本報告で用いた PCoA（Principal Coordinate Analysis）のほか，MMDS（Metric Multi-dimensional Scaling）や t-SNE（t-distributed Stochastic Neighbor Embedding）といった手法が挙げられ，データの特徴に合わせた選択が必要となる．さらに，Word Cloud といった全く別の可視化方法も候補の一つになるといえる．

また，最終的に単元を整理する過程について，階層構造が導入されている．これは LDA の適用で得られた各トピックには含まれていなかった要素であり，何をもって単元を入れ子にするかは主観に基づくものである．

タグの策定方法　RQ2 におけるタグ策定について，各工程が主観的である可能性を否定できない．単元の除外について，例えば本報告では表 4c に示すカテゴリ 3 の単元を全て除外したが，このうち “スレッド” や “例外処理” といった単元については，オブジェクト指向にまつわるものと判断するか否かが分かれる可能性がある．また，階層構造をフラットにするという工程については，例として本報告では表 4a に示す “データ構造” 配下の単元を抽象化の対象としたが，“データ構造” だけでは抽象化レベルが高すぎると判断した場合，配下の単元をタグにすることになる．

タグの付与方法　RQ3 におけるタグの付与方法は，問題によってはその実態を反映できていない可能性がある．まずカテゴリ 1 のタグ付与方法に関して，問題文・解説文が表 6 のキーワードにヒットしたものについてしか確認していないため，本来問題として意図されているものの，その明確な記述が問題文及び解説文になされていないものについて，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，内容確認における主観性の混在を否定できない．

続いてカテゴリ 2 のタグ付与方法に関しては，表 7 に示すマッチ条件のうち一部を宣言部のみで判断しているため，それらの項目を実際に使用しているかが考慮できていない．さらに各解答コードは 1 回以上のマッチで条件を満たすため，全体としてタグが付きやすい傾向にある．

また，カテゴリ間で付与方法が完全に異なるため，それらのタグの付与状況を単純に比較することは妥当ではない可能性がある．より妥当性のある，統一されたタグの付与方法を検討する必要がある．

4.5.2　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．私立大学を含めた場合，収集結果の傾向が異なる可能性がある．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 A〜C 問題のうち，計 324 問のみである．問題数や難易度の範囲を拡大し，AtCoder Beginner Contest 全体を対象と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．また，プログラミングコンテストの問題の実態把握という観点では，AtCoder 内の上級者向けコンテスト，及び Codeforces 等のプログラミングコンテストサイトも踏まえたドメインでの分類と比較が求められる．

5　まとめ

本報告では，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た．問題の分類に先立って学習単元を明示するため，大学の情報系学科におけるプログラミング教育に着目し，そのシラバスデータの収集・分析により類似する単元群を得た．得られた学習単元に即して分類タグを策定し，プログラミングコンテストの１つである AtCoder Beginner Contest の A〜C 問題に対して付与することで，タグと問題の間に生じる双方向の対応関係を分析した．

まず，問題の難易度に応じて付与されるタグの傾向は異なり，難易度の低い問題は単純な基礎事項のみで解答可能なことや，難易度が高くなると問題が複雑化し，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，タグごとに付与された問題の難易度傾向にも差異があり，難易度を問わず問題が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．そうした情報をもととし，問題利用に際しての方針をいくつか提示した．

今後の課題としては，まず，さらに難易度の高い問題を含めた分類対象の拡大が挙げられる．アルゴリズムに関する問題は C 問題で増加傾向が見られたことから，D 問題以降にはさらに多く含まれるのではないかと予想される．

また，他のプログラミングコンテストでもタグ付けによる分類を行い，傾向を比較することも課題の一つである．例えば，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て学ぶことのできる問題量の比較が着目点として挙げられる．

さらに，本報告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といえる．クラスや継承などのオブジェクト指向特有の技法を用いて解答するほうが好ましい問題があるのか，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，といった点に注目可能である．

謝辞

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 肥後 芳樹 教授には，ご多忙の中，研究活動に対して多くの貴重な御助言や御指導を賜りました. 心より深く感謝申し上げます.

立命館大学情報理工学部 吉田 則裕 教授には，ご多忙の中，研究の着想に関して貴重なご助言や御指導を賜りました. 心より深く感謝いたします.

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 准教授には，研究の着想から研究活動の直接の御指導，論文の執筆に至るまで，あらゆる場面で多くの御指導を賜りました. 松下 誠 准教授の適切な御指導により，本論文を完成させることができました. 心より深く感謝申し上げます.

大阪大学大学院情報科学研究科 神田 哲也 助教には，研究室での発表や評価内容について，大変貴重な御意見・御助言を賜りました. 多くの御助言を頂いた神田助教に心より深く感謝いたします.

最後に，様々な御指導・御助言を頂いた大阪大学大学院情報科学研究科コンピュータサイエンス専攻肥後研究室の皆様に，心より深く感謝申し上げます.

A　付録：情報系学科リスト

本報告でシラバスの収集対象とした国立大学の情報系学科を，表 13 に示す.

表 13: シラバス収集の対象とした国立大学の情報系学科

北海道大学 工学部 情報エレクトロニクス学科
北見工業大学 工学部 地球未来デザイン工学科
山形大学 工学部 情報・エレクトロニクス学科
岩手大学 理工学部 システム創成工学科
弘前大学 理工学部 電子情報工学科
東北大学 工学部 電気情報物理工学
秋田大学 理工学部 数理・電気電子情報学科
お茶の水女子大学 理学部 情報科学
信州大学 工学部 電子情報システム工学科
千葉大学 理学部 数学・情報数理学
埼玉大学 工学部 情報工学科
宇都宮大学 工学部 基盤工学科
山梨大学 工学部 コンピュータ理工学科
東京大学 理学部 情報科学科
東京工業大学 工学院 情報通信系
東京工業大学 情報理工学院 情報工学系
東京工業大学 情報理工学院 数理・計算科学系
東京農工大学 工学部 知能情報システム工学科
横浜国立大学 理工学部 数物・電子情報系学科
筑波大学 情報学群 情報メディア創成学類
筑波大学 情報学群 情報科学
筑波技術大学 保健科学部 情報システム学科
筑波技術大学 産業技術学部 産業情報学科
群馬大学 情報学部 情報学科
茨城大学 工学部 情報工学科
電気通信大学 情報理工学域 I 類 （情報系）
三重大学 工学部 総合工学科
京都大学 工学部 情報学科

表は次ページへ続く

前ページからの続き

京都工芸繊維大学 工芸科学部 設計工学域
名古屋大学 情報学部 コンピュータ科学科
名古屋工業大学 工学部 情報工学科
和歌山大学 システム工学部 システム工学科
大阪大学 基礎工学部 情報科学科
大阪大学 工学部 電子情報工学科
富山大学 工学部 工学科
岐阜大学 工学部 電気電子・情報工学
神戸大学 工学部 情報知能工学科
福井大学 工学部 電気電子情報工学科
豊橋技術科学大学 工学部 情報・知能工学課程
静岡大学 情報学部 情報科学科
山口大学 工学部 知能情報工学科
山口大学 理学部 物理・情報科学科
岡山大学 工学部 情報・電気・数理データサイエンス系
島根大学 総合理工学部 知能情報デザイン学科
徳島大学 理工学部 理工学科
愛媛大学 工学部 工学
香川大学 創造工学部 創造工学科
高知大学 理工学部 情報科学科
鳥取大学 工学部 電気情報系学科
九州大学 工学部 電気情報工学
九州大学 理学部 物理学科
九州工業大学 情報工学部 情報・通信工学科
九州工業大学 情報工学部 知能情報工学科
佐賀大学 理工学部 理工学科
大分大学 理工学部 共創理工学科
熊本大学 工学部 情報電気工学科
琉球大学 工学部 工学科
長崎大学 情報データ科学部 情報データ科学科
鹿児島大学 工学部 先進工学科

参考文献

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. 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/, 2022.

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

[4] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, to Read. In *Proceedings of the Seventeenth Conference on Hypertext and Hypermedia*, HYPERTEXT '06, p. 31–40, New York, NY, USA, 2006. Association for Computing Machinery.

[5] Carson Sievert and Kenneth E. Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pp. 63–70, 2014.

[6] Jakob Voss. Tagging, Folksonomy & Co - Renaissance of Manual Indexing?, 2007.

[7] AtCoder プレスリリース. 日本最大のプログラミングコンテストサイト AtCoder 全世界での登録者数が 30 万人を突破. https://prtimes.jp/main/html/rd/p/000000034.000028415.html, 2021.

[8] 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.

[9] 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.

[10] 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—. 自然言語処理研究会研究報告, pp. NL–229–15. 情報処理学会, 2016.

[11] 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 (NLP2017), pp. NLP2017–B6–1. 言語処理学会, 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.

[13] 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.

[14] 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.

[15] 文部科学省. 大学設置基準等の一部を改正する省令等の施行について（通知）. https://www.mext.go.jp/b_menu/hakusho/nc/07091103.htm.

[16] 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.

[17] 文部科学省 中央教育審議会. 学士課程教育の構築に向けて（答申）, 2008.