6G-02

抽象構文木を利用したサンプルプログラムの難易度判定手法

漆原 宏丞 † 岸本 有生 ‡ 本多 佑希 * 兼宗 進 †

† 大阪電気通信大学 ‡ 大阪電気通信大学高等学校 四天王寺大学 *

1 はじめに

2022 年度から高等学校でプログラミングが必修化され、全ての高校生がプログラミングを学ぶことになった。プログラムの学習には適切な難易度で小さなサンプルプログラムを段階的に学習してくことが有効である。しかし、教員としては適切な難易度にしたつもりでも、学生にとってはまだ難しかったり、含まれている学習項目が多い可能性がある。そこで、授業を準備する教員が、用意したサンプルプログラムの難易度や、そこに含まれている学習項目を機械的に判定する手法を提案する。

2 既存の複雑度を計測するアルゴリズム

複雑度を計測する既存のアルゴリズムとしては、サイクロマティック複雑度 [1] と Halstead Volume[2] が代表的である。前者はソースコードを制御フローグラフに変換し、その辺やノードの数から算出される。これは、分岐やループが多いほど複雑であるという考えに基づいている。後者は、プログラム中のオペレータとオペランドの個数と種類数から算出される。これは、プログラム中の変数などの個数やオペレータが多いほどプログラムの規模が大きくなる傾向に基づいている。これらの指標は、基本的に大規模開発で使用され、授業で扱うようなレベルのソースコードは扱うことを想定されていない。

3 提案システムの概要

本報告では、表 1 のような難易度の設定を行った。

難易度の設定にあたって、今回は市販の Python の入門書 7 冊 [3][4][5][6][7][8][9] と、高校向けの Python の教科書 1 冊 [10] の計 8 冊を調査した。市販の入門書は、大学の Python の入門の授業で教科書に指定されているものを採用した。これらの書籍を調査し、扱われている順番やサンプルプログラムの規模などを参考にしている。

表 1: 今回定義した難易度

難易度	内容
1	関数呼び出し (print 関数など)
2	代入文、四則演算
3	リスト、インデックス指定
4	if、for、while、関数定義
5	for/while の中に if/for 関数定義の中に if/for/while

難易度の自動算出を行うため、Python の入門レベルのプログラムを対象に抽象構文木を解析し、そこに含まれているノードの種類に応じて重み付けを行うことで難易度判定を行った。抽象構文木とは、プログラムの構造を木構造で表現したものである。その木構造を解析し、例えば「関数呼び出し」ノードが見つかれば重み+1、if が見つかれば重み+3、のような処理を行う。設定した重みを表 2 に示す。今回は、四則演算は長さによらず、「1 つの計算式」として扱っている。つまり、「a + 1」や「a + b * c」などは同じ扱いとしている。

代入文などで入れ子になると重みが減少している理由としては、入れ子になっている時点でそれ単体での学習が済んでいると考えたためである。例えば「a = a + 1」という代入文を理解していれば、これが for 文などの中で使われていても、その部分を理解するのには対して困らないと考えている。if 文や for 文などが入れ子になっていた場合は、代入文ほどすんなり理解できるとは限らないので、0 にはしていない。

また、より複雑な構造、例えば「for 文の中に for 文があり、さらにその中に if 文」などの構造はアルゴリズム的な要素が強くなるため、本提案では難易度 5 より上と判断される。

Methods for determining the difficulty level of sample programs by AST

Kosuke URUSHIHARA† Tomonari KISHIMOTO† Yuki HONDA* Susumu KANEMUNE†

†Osaka Electro-Communication University

572-8530, Neyagawa, Japan

表 2: 設定した重み

内容	重み
関数呼び出し	1
四則演算	1
代入文	2 ただし入れ子になっていれば 0
リスト	2
インデックス指定	1
if 文	3 ただし入れ子になっていれば 1.5
for 文	3 ただし入れ子になっていれば 2
while 文	3.5 ただし入れ子になっていれば 2
比較演算	1
論理演算	2
関数定義	4.5
メソッド呼び出し	2

4 まとめと今後の予定

今回は、Python の入門レベルを対象にサンプルプログラムの難易度を自動判定するシステムを提案した。大学の授業で教科書に指定されている市販書籍や、高校向けの Python の教科書の内容を参考に、どんな内容をどの段階で扱うべきか検討した。入門レベルのプログラムの抽象構文木を解析し、検討した難易度に合うようノードの重み付けを行うことで、難易度の自動判定を行えるようにした。

今後は、この難易度判定が情報関係基礎や共通テストで出題されるプログラムに対してどの程度有効か検討し、改良を進める。また、サイクロマティック複雑度や Halstead Volume が入門レベルのプログラムに対してどの程度有効か検討し、可能であれば本システムに取り入れていく予定である。

参考文献

[1] THOMAS J. McCABE: ”A Complexity Measure”, IEEE Transaction on Software Engineering, Vol.SE-2, No.4, pp.308-320, (1976).

[2] Halstead, Maurice H.: ”Elements of Software Science.”, Elsevier North-Holland, Inc., (1977).

[3] Bill Lubanovic(著), 長尾高弘 (訳): “ 入門 Python 3 ”, オライリー・ジャパン, (2015).

[4] 渡辺宙志: “ ゼロから学ぶ Python プログラミング ”, 講談社, (2020).

[5] 国本大悟, 須藤秋良: “ スッキリわかる Python 入門 ”, インプレス, (2019).

[6] 喜多一, 森村吉貴, 岡本雅子: “プログラミング演習 Python 2021”, http://hdl.handle.net/2433/265459 (2022/08/29 アクセス).

[7] 東京大学 数理・情報教育研究センター: “ Python プログラミング入門 ”, https://sites.google.com/view/ut-python/ (2022/08/29 アクセス).

[8] ジョン V グッターク (著), 久保幹雄ほか (訳): “ Python 言語によるプログラミングイントロダクション ”, 近代科学社, (2017).

[9] 三谷純: “ Python ゼロからはじめるプログラミング ”, 翔泳社, (2021).

[10] 兼宗進, 間辺広樹, 本多佑希, 前田健太郎, 阿部百合, 漆原宏丞: “ Python 入門プログラミングの基礎から応用まで ”, 東京書籍, (2022).