# コード共有プラットフォームにおける戦略性を考慮した評価指標の提案と実装

前田 新太郎[*1], 古池 謙人[*1], 東本 崇仁[*2]
*1 東京工芸大学大学院工学研究科, *2 東京工芸大学工学部

# Implementation of Quality Indicator based on Strategy in Code Sharing Platform

Shintaro Maeda[*1], Kento Koike[*1], Takahito Tomoto[*2]
*1 Graduate School of Engineering, Tokyo Polytechnic University,
*2 Faculty of Engineering, Tokyo Polytechnic University

プログラミング学習では，コードをより良いものへ洗練することが重要である．これまで，受講者の多様なプログラミングのレベルに着目し，レベルの近いコードを提示するコード共有プラットフォームを開発してきた．本稿ではより正確な受講者ごとのレベルを評価するべく，コードの振舞いを用いて戦略の近さも考慮した評価指標を提案・実装する．提案手法では，自身より発展したコードや無駄な動きが少ないコードの数値化を試みる．

キーワード: プログラミング学習，他者からの学び，コード共有，仮想ロボットプログラミング

## 1. はじめに

プログラミング学習において，自身の記述したコードをより良いものへ近づける，コードの洗練活動は重要である．本研究ではこれまで，プログラミングのレベルが異なる様々な学習者が受講されることが期待されるプログラミング講義に着目し，学習者のレベルに近いコードが共有される環境をコード共有プラットフォームとして開発してきた[(1)]．プログラミングについて初めて学ぶ学習者から独学で学んできた学習者といった，様々なレベル差のある学習者のプログラミングに関する知識を収集・評価することで，学習者のレベルに応じたコードを教師の負荷なく共有することが実現される．学習者は自身のレベルの近いコードが共有されるため，他者のコードから学ぶことができ，自身のコードの洗練活動につながる．さらに，この活動を繰り返すことで，漸進的な学習になることが期待される．

開発してきたシステムは，仮想上で動作するロボットプログラミングを学習題材としている．学習者は，ロボットに対して，「畑に移動」，「畑に種植え」，「成長した作物の収穫」といった動きが行えるようにコードを記述する．さらに，システム上では，ロボットプログラミングを基にした評価指標とその評価によるランキング機能を備えており，学習者のランクに近いコードのみ共有されるように制御する．

本システムは，小規模な実験室環境上[(2)]や授業実践による大規模な環境上[(3)]での評価を既に実施しており，一定の学習効果が確認された．しかしながら，評価指標のみ用いたレベルの近さを評価するランキング機能では，必ずしも学習者の戦略に沿ったコードが共有されるとは限らない．

そこで本研究では戦略を考慮する仕組みをランキング機能に拡張する形で提案・実装する．具体的には，コードの実行時の振舞いを戦略として定義し，学習者の記述したコードの振舞いに類似しているか評価する．


**図 1 開発してきたシステム画面の例**

## 2. コードの洗練活動

### 2.1 コードの洗練における学習題材

コードをより良いものへ近づける洗練活動は重要である．一般的なプログラミング講義においても，講義内で，良いコードの例をプロジェクターやディスプレイなどで投影し，どの点が良いのか解説する場合や，教師自らが，より良いコードに関する書き方や手法などを，学習者の前で披露する場合がある．このように，コードの洗練活動の重要性は確かなものであるが，多くのケースにおいて，学習者は適切な学習につながらないことが懸念される．例えば，教師から教授される良いコードの書き方については，別の学習者が記述したコードや参考書に書かれているコードといった，学習者自身のコードをベースとした解説ではない．そのため学習者は，自身のコードと解説されたコードの差異を見出すことができずに，学習につながらない事が考えられる．一方で，教師が学習者一人ひとりのコードを評価し，より良いコードの記述とその解説をすることは，一対多で実施されるプログラミング講義において現実的ではない．

そこで本研究では，様々なプログラミングのレベルを持つ学習者が受講されるプログラミング講義に着目した．プログラミング講義は通常，プログラミングに関して初めて触れる学習者や，独学で学んできた学習者など，幅広いレベルを持つ学習者が多く受講する．この学習者の知識をコード共有という形で活用することで，学習者にレベルの近いコードを提示することができるのではないかと考えた．学習者は自身のレベルに近いコードが提示されることから，自身のコードと共有されたコードとの差分を学び，自身のコードに取り入れることができる．つまり本研究では，他者のコードから学び，そこから自身のコードに学んだ要素を取り組み，洗練する活動を促す学習活動を支援する．

### 2.2 他者からの学びによる学習効果と提案手法

本研究では，他者のコードから学び，自身のコードを洗練する活動を促す学習環境を提案している．他者のコードからの学びによる学習効果は多くの研究から報告されている[4][5]．その中で特に，プログラミングの熟達者の記述したコードを読むことは，学習者において，より良い学習につながることが報告されている[6]．このように，教育上，熟達者の記述したコードから学ぶことによる学習効果は明らかであるが，初学者がファーストステップとして熟達者のコードを読むことは，互いのレベル差から学習につながらない．この問題の解決には，学習者のレベルに少し近いコードを与えることが望ましいと考える．

本研究では，プログラミング講義を受講する学習者を対象に，互いのコードを共有することで，より良い

コードに近づける，洗練活動を促す学習デザインを提案した．

## 3. 先行研究 - コード共有プラットフォーム

### 3.1 システム概要

2 章にて述べたように，本研究ではこれまで，他者のコードから学び，自身のコードの洗練活動を促すコード共有プラットフォームを開発してきた．

開発したシステム（図 1）は，農場をテーマとした仮想上で動作するロボットプログラミングで，作物を収穫させ，スコアを競い合うゲームのような仕様となっている．学習者は，ロボットに対して，畑を巡回しながら種植えと収穫を行うようなプログラミングをする．ロボットが成長した作物を収穫すると，後述する収穫ポイントを獲得することができる．

### 3.2 課題設計

他者のコードから学び，自身のコードへ学習した要素を取り入れるためには，その他者のコードがどのように動作しているかを理解する必要がある．本システムでは仮想ロボットプログラミングにより，学習者の記述したコードの実行時動作を振舞いとして可視化することにより活動の支援をする．

### 3.3 評価指標

学習者にレベルの近いコードを共有するためには，コードのレベルを評価することが重要である．本システムでは，ロボットプログラミングを基に，評価の指標を定めた．具体的には，「収穫数」，「コスト」，「合計スコア」の 3 つである．「収穫数」は，作物の収穫できた数を表す．「コスト」は，ロボットに対して何かしらの行動をするたびに増加するステップ数を表す．「合計スコア」は，「収穫数」から「コスト」を引いた結果を表す．コードの評価は一意に定まらず，コードの再利用性やコードの見やすさといった複数の観点が考えられるが，開発したシステムは，多くの作物を収穫でき，よりコストを抑えたコードを良いコードとして定めている．また，コードの書き方のような，静的な評価は考慮されておらず，実行時動作のような，動的な評価を取り入れている．

### 3.4 ランキング機能

本研究では，他者のコードからの学びに着目している．学習者の学習対象となるコードは，その学習者のレベルに近いことが望ましい．そこで本研究では，3.3 節にて説明した評価指標による評価を用いて，ランキング順にソートし，学習者のランクに近いコードのみ閲覧できるランキング機能を提案した．

実際に開発したランキング機能を図 2 に示す．ランキング機能は，評価指標による指標ごとのランキングを学習者に提示する．学習者は，閲覧したいランクを選択すると，そのランクの実際のコードが画面右部に提示される．ランキング機能の制約としては，学習者のひとつ上のランクから下のすべてのコードが閲覧できるようになっている．つまり，図を例に，自身のランクが 22 位だった場合には，21 位から下のすべてのコードが閲覧できる．ここで，20 位のランクを選択したとしても，コードは提示されない．

## 4. 戦略を考慮したコード共有プラットフォーム

### 4.1 概要

本研究ではこれまで，他者のコードから学び，自身のコードを洗練する活動を促すコード共有プラットフォームを開発してきた．また，開発したシステムは，実験室環境における予備評価実験と実践環境下における授業実践にて，一定の学習効果が確認された．その一方で，本研究の重要視するレベルの近いコードのみ

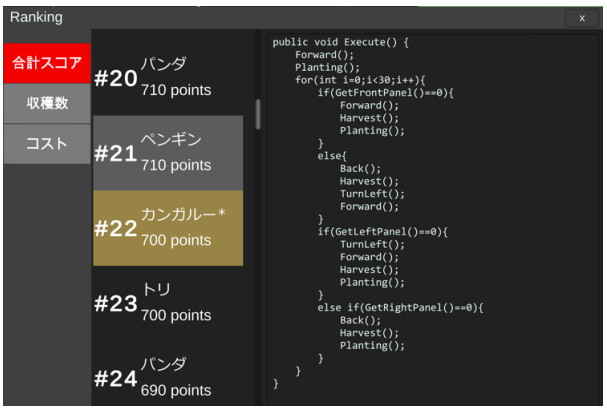


図 2 ランキング機能の画面の例

共有する仕組みに関して，未だ改善の余地があると考える．具体的には，レベルの近いコードとして学習者に共有する際に，「戦略」を考慮する必要があるのではないかと考えた．

戦略の重要性について図 3 のロボットの移動に関する戦略を例に説明する．図の例では，A さんは，畑の外縁をロボットが移動するような戦略をとっており，B さんは，敷き詰められたすべての畑をロボットが移動するような戦略をとっている．このとき，例のようなロボットの移動方法について思考途中である C さんに対して，例え，A さんのコードを共有したとしても，C さんは自身の戦略と異なることから，学習につながらない恐れがある．その一方で，B さんの戦略は C さんの戦略に類似していることから，B さんのコードを C さんへ共有することは B さんにとって，有益な学習につながるのではないかと考える．つまり，このような戦略を考慮することで，よりレベルの近さを考慮したコード共有が実現できる．

また，一般的なプログラミングにおける戦略の例として，ソートアルゴリズムがあげられる．ソートアルゴリズムは，あるデータの集合を昇順などに並び替えることであるが，この並び替える方法が戦略に値すると考える．例えば，一次元配列のデータにおいて，隣接する値同士を比較し，入れ替える処理を繰り返すことによりソートを実現するバブルソートや，ある値から最後の値まで順に照らし合わせて，最小値を見つけた際に先頭の要素と交換する選択ソートなど，要素を並び替えるソートアルゴリズムに対して複数の戦略が存在する．このように，開発してきたシステムの課題設計のみならず，一般的なプログラミングにおいてもある目的に対して複数の戦略が考えられる．

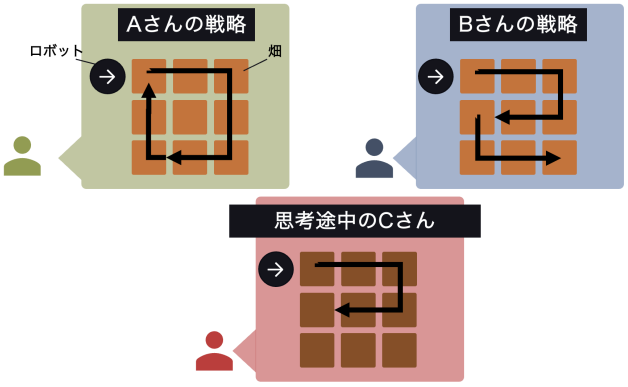


図 3 学習者ごとに異なる戦略の例

そこで本研究では，戦略の重要性に着目して，その戦略を考慮した評価機構の提案とその評価を自身と他者と比較し，類似度を評価する仕組みを提案する．類似度を評価することで，学習者の得点の近さに加えて戦略的な近さも考慮したレベルの近いコードを共有することができる．これにより，学習者は従来のシステムと比べてより高い学習効果を生み出すことが期待される．

### 4.2 提案手法

学習者の記述したコードの戦略を評価することは，そのコードのアルゴリズムを用いることが適切だと考える．本研究では戦略を評価する手法として，コードの実行時の動作から生成される，振舞いを用いることを提案する．具体例を図 4 に示す．図の例では，ある学習者の記述したコードに対して，そのコードの振舞いをデータ系列のような形で示している．図のように，コードが Forward(); Forward(); Forward(); Planting();であった場合，生成される振舞いとしては F→F→F→P となる．

本研究では，コードの戦略を評価する観点として，コードの構造（記述したコード）ではなく，コードの意味や機能（生成された振舞い）が近い．つまり，図の例では Forward();が複数回呼ばれているが， for による繰り返し処理に置き換えたとしても，同じ振舞いであることから，異なる戦略として評価しない．

図 5 のコード A では，関数を一種のベタ書きのよう

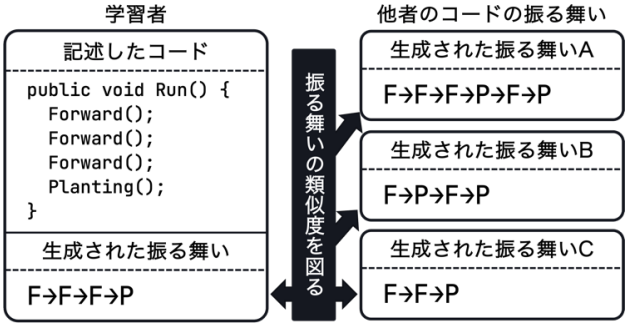


図 4 類似度を図る具体例

に呼び出しているが， B コードでは，for を用いてより簡潔に仕上げている．このように，コードの構造上ではコード A とコード B では大きく異なっているが，振舞いを比較すると，同一となっている．このとき，戦略を評価する手法としてコードの構造を比較するような手法を用いた場合では，同一の振舞いにも関わらず，戦略の異なるコードとして評価されてしまう．よって，本研究では，ある機能をどのように達成しようとしているかを測ることができる振舞いを用いて戦略の評価をする．自身と他者との振舞いを比較することで，どのぐらいそれぞれの戦略が近いかを類似度として算出する．類似度の算出には，自身の振舞いと他者の振舞いの中にある一つ一つの処理を比較し，一致する回数を用いる．比較の際には，一回の探索による一致数で終わらせずに，直近の組み合わせを外した状態で再度探索する，バックトラック法を用いる．バックトラック法により，複数回探索し，最も多く振舞いの処理が一致した回数を類似度の算出に採用する．一致した回数が定まると次に，自身の振舞いの数/一致した回数で適合率，他者の振舞いの数/一致した回数で再現率をそれぞれ算出する．そして最後に，適合率と再現率の調和平均として F 値を算出する．この F 値を本研究では，類似度として定義する．

図 6 に類似度の算出例を示す．図の例では，A さんと B さんの振舞いを比較している．まず，2 つの振舞いの一致する数を調べるために，バックトラック法を用いて複数回探索を行っている．探索の結果から，一致した振舞いの数は 5 個であることから，適合率は 5/5，再現率は 12/5 という結果となった．最後に，適合率と再現率の調和平均として，F 値は 0.59 という結果となった．つまり，A さんのコードは B さんのコードに 0.59，類似していることになる．このような流れで本研究では自身と他者との振舞いの類似度を測る．

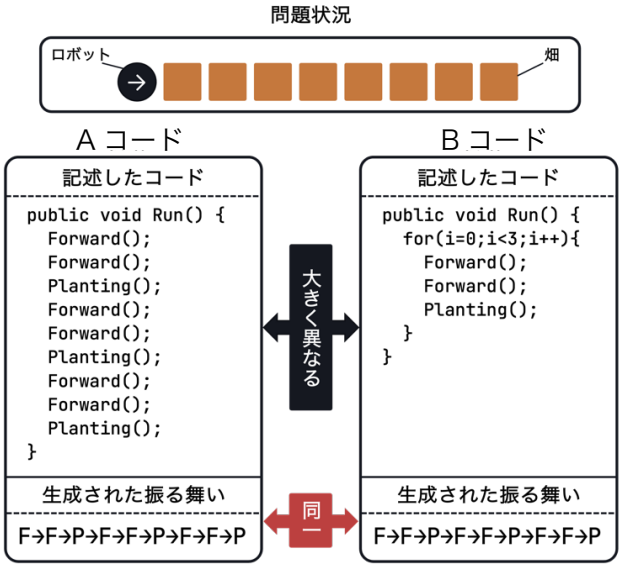


**図 5 コードの構造と振舞いの比較の例**

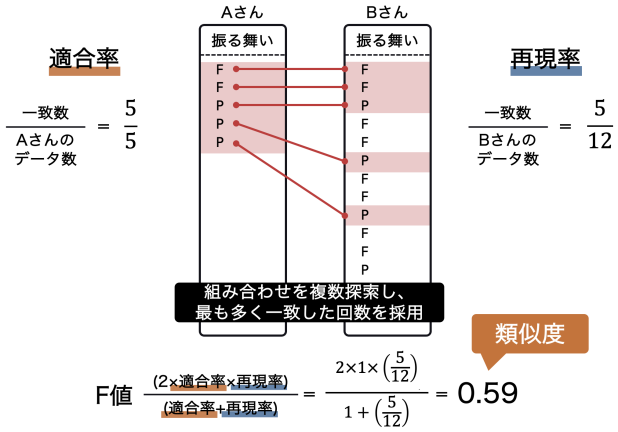


**図 6 類似度の算出方法の例**

## 5. ケーススタディ

### 5.1 検証概要

本研究では，4 章にて提案した類似度の算出手法をシステムに実装および評価する．今回はケーススタディとして，これまでの予備評価実験と授業実践にて学習者が記述したコードを用いて，水たまりパネルを回避するアルゴリズムが求められる問題 3 を対象とする．評価方法としては，ある無作為に選択した学習者のコードを基準にランキングを生成する．ランキングは，従来の戦略を考慮しないランキングと提案手法による類似度を用いた戦略を考慮するランキングの 2 つである．戦略を考慮するランキングでは，一定の類似度（0.3 以上）が満たされないとランキングに表示されない制約を設定する．この 2 つのランキングを比較し考察する．

### 5.2 考察

図 7 は無作為に選択した実験時に学習者が作成したコードである．このコードを基に，2 つのランキング

を生成した．従来のランキング機能により提示されるコード群と提案手法に沿って戦略を考慮したコード群を図 8 に示す．

図 8 の生成されたランキングの結果から，まず従来のランキングについて，すべての類似度が 0.15 を下回る結果となった．このことから，本研究のあげた従来のランキング機能の問題点である，学習者の異なる戦略のコードが共有されてしまう問題について，ケーススタディではあるものの明らかになった．つまり，従来のランキング機能では，異なる戦略のコードが学習者に共有されることから学習につながらなかった可能性が考えられる．次に，類似度を考慮したランキングの結果について，従来のランキングと比較すると，すべてのコードの類似度が高い傾向にあることがわかった．これは，従来のランキングでは，ロボットの移動方法に関する戦略が様々であることに対して，類似度を考慮したランキングでは，遠くの畑へ移動しない戦略をとっていることから，このように差が生じていると考える．特に，移動以外の戦略として，2 位の 10 行目から 12 行目のコードにおいて，複数回，TurnLeft(); 関数を呼び出されていることがわかる．これは，関数を呼び出すたびに作物が成長するシステムの仕様を利用した戦略だと考えられ，このランキングの基準である図 7 のコードに置いても 14 行目にて TurnLeft();

**ランキングの基となるコード**

**合計スコア: 573**

```
1  public void Execute() {
2
3    forward();
4
5    for(int i=0; i< 4; i++){
6      planting();
7
8      forward();
9      planting();
10
11     forward();
12     planting();
13
14     turnleft();
15     turnleft();
16     turnleft();
17     turnleft();
18     turnleft();
19     turnleft();
20     turnleft();
21     turnleft();
22     turnleft();
23     turnleft();
24     turnleft();
25     turnleft();
26
27     harvest();
28
29     back();
30     harvest();
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

**図 7 ランキング生成において基準としたコード**

を繰り返し行っていることから，提案手法のシステムは適切に戦略を考慮したランキングが生成できていると考える．

以上，考察から，提案した類似度の算出方法および類似度を用いて生成したランキングは，学習者の記述したコードの戦略を適切に評価できていることがわかった．

## 6. 今後の課題

本研究では，コードの振舞いを戦略として，自身と他者との振舞いを比較することにより，類似度を算出する仕組みを提案した．

今後の課題として，今回のケーススタディでは，著者等による主観的な評価であることから，より信頼性を高めるために，プログラミング熟達者を対象とした検証実験の実施が挙げられる．このとき，今回のケーススタディと同様に，従来のランキングと提案システムのランキングを提示し，振舞いについてシステムが適切に評価されているのか調査する．

## 7. おわりに

本研究ではこれまで，コードの洗練活動を促すコード共有プラットフォームを開発・評価してきた．評価から，一定の学習効果が示唆されたが，ランキング機能によるコードの提示機能については改善の余地があると考えられた．

そこで本研究では，学習者は高得点を目指そうとする際に戦略が存在することに着目して，戦略を考慮したランキング機能を提案した．具体的には，学習者の記述したコードの振舞いを他者との比較を行い，類似度として算出する．本稿ではケーススタディとして，これまでに実施した実験にて学習者が作成したコードを用いて，従来のランキングのコード群と提案手法による類似度を考慮したランキングを比較および考察した．

考察から，従来のランキングでは，全く異なるロボットの動作をするコードが集まった結果に対し，類似度を考慮したランキングでは，ロボットの動作が類似

したコードが集まる結果となった.

今後の課題として，プログラミング熟達者による評価実験が挙げられる.

## 謝辞

本研究の一部は JSPS 科研費 JP22K12322，JP21H03565，JP20H01730 の助成による.

## 参 考 文 献

(1) 前田新太郎, 古池謙人, 東本崇仁: “ロボットプログラミングを題材にした競争型知識共有プラットフォームの提案と実装”, 人工知能学会第 91 回先進的学習科学と工学研究会, pp.87-92 (2021)

(2) 前田新太郎, 茂木誠拓, 古池謙人, 東本崇仁: “仮想ロボットプログラミングを用いたコード共有プラットフォームの開発と評価”, 教育システム情報学会誌, Vol. 40, No. 3, in press, (2023)

(3) 前田新太郎, 古池謙人, 東本崇仁: “仮想ロボットプログラミングを対象としたコードの洗練活動を促す知識共有プラットフォームの実践利用の分析”, 人工知能学会第 95 回先進的学習科学と工学研究会, pp.1-6 (2022)

(4) 東本崇仁, 赤倉貴子: “提案するプログラムトレース課題のための学習支援システムの開発とその実践”, 電子情報通信学会論文誌 D, Vol. 101, No. 6, pp. 810–819, (2018)

(5) Teresa, B. and Carsten, S.: “The use of code reading in teaching programming”, In Proceedings of the 13th Koli Calling international conference on computing education research, pp. 3–11 (2013)

(6) Iwona, M. and Grace, T.: “Befriending computer programming: A proposed approach to teaching introductory programming”, Informing Science: International Journal of an Emerging Transdiscipline, Vol. 4, No. 1, pp. 277–289 (2007)

## 従来のランキング

### 1位 合計スコア: 580 類似度: 0.15 適合率: 0.16 再現率: 0.14

```
public void Execute() {
    Forward();
    Planting();
            while(true){
        for(int i=0; i<2;i++){
          if(GetFrontPanel()==1){
            TurnLeft();
                    }


          Forward();
          if(GetGrowthLevel()==0){


            Planting();
                    }
          if(GetGrowthLevel()>=5){

            Harvest();
            Planting();


                    }
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→
前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

他者の振舞い

前進→種植→G 前→前進→ G 成→種植→ G 成→ G 前→前進→ G 成→
種植→ G 成→左回→ G 前→前進→ G 成→種植→ G 成→ G 前→前進→
G 成→種植→ G 成→左回→ G 前→前進→ G 成→種植→ G 成→ G 前→
前進→ G 成→種植→ G 成→左回→ G 前→前進→ G 成→種植→ G 成→
G 前→前進→ G 成→ G 成→収穫→種植→左回→ G 前→前進→ G 成→
G 成→収穫→種植→ G 前→前進→ G 成→ G 成→収穫→種植→左回→
G 前→前進→ G 成→ G 成→収穫→種植→ G 前→前進→ G 成→ G 成→
収穫→種植→左回→ G 前→前進→ G 成→ G 成→収穫→種植→ G 前→
前進→ G 成→ G 成→収穫→種植→左回→ G 前→前進→ G 成→ G 成→
収穫→種植→ G 前→前進→ G 成→ G 成→収穫→種植→左回→ G 前→

### 2位 合計スコア: 579 類似度: 0.15 適合率: 0.13 再現率: 0.18

```


public void Execute()
{
turnleft();forward();
forward();
turnright();forward();planting();
turnright();forward();planting();
forward();planting();
turnleft();planting();
forward();planting();
forward();planting();
turnleft();forward();planting();
forward();planting();
forward();planting();
turnright();forward();planting();
turnright();forward();planting();
turnleft();forward();
turnleft();forward();
turnright();forward();
turnright();forward();
turnright();forward();

forward();harvest();
turnright();forward();harvest();
turnleft();forward();harvest();
turnleft();forward();harvest();
forward();harvest();
forward();harvest();
turnright();forward();harvest();
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→
前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

他者の振舞い

左回→前進→前進→右回→前進→種植→右回→前進→種植→前進→
種植→左回→種植→前進→種植→前進→種植→左回→前進→種植→
前進→種植→前進→種植→右回→前進→種植→右回→前進→種植→
左回→前進→左回→前進→右回→前進→右回→前進→右回→前進→
前進→収穫→右回→前進→収穫→左回→前進→収穫→左回→前進→
収穫→前進→収穫→前進→収穫→右回→前進→収穫→前進→収穫→
前進→収穫→右回→前進→収穫→前進→収穫→

### 3位 合計スコア: 573 類似度: 0.13 適合率: 0.10 再現率: 0.18

```
public void Execute() {

  Forward();
  Planting();


  Forward();
  Planting();


  Forward();
  Planting();

  TurnLeft();


  Forward();
  Planting();


  Forward();
  Planting();


  Forward();
  Planting();
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→
前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

他者の振舞い

前進→種植→前進→種植→前進→種植→左回→前進→種植→前進→
種植→前進→種植→前進→前進→前進→前進→前進→前進→前進→
前進→前進→前進→前進→左回→左回→前進→前進→前進→前進→
前進→前進→前進→前進→前進→前進→前進→収穫→前進→収穫→
前進→収穫→前進→収穫→右回→前進→収穫→前進→収穫→

## 類似度を考慮したランキング

### 1位 合計スコア: 720 類似度: 0.36 適合率: 0.38 再現率: 0.34

```
public void aExecute() {
  bool isLeft = true;
  while(true) {
    Forward();
    Planting();
    if (GetFrontPanel() != 0) {
      if (isLeft) {
        TurnLeft();
        Forward();
        TurnLeft();
        Planting();
        isLeft = false;
      } else {
        TurnRight();
        Forward();
        TurnRight();
        Planting();
        isLeft = true;
      }
    }
    if (GetBackPanel() == -1) {

    }
  }
}

public void Execute() {
  Forward();
  while (true) {
    for (int i = 0; i < 2; i+=1) {
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→
前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

他者の振舞い

前進→種植→前進→収穫→種植→前進→収穫→左回→左回→種植→
前進→収穫→種植→前進→収穫→左回→左回→種植→前進→収穫→
種植→前進→収穫→左回→左回→種植→前進→収穫→種植→前進→
収穫→左回→左回→種植→前進→収穫→種植→前進→収穫→左回→
左回→種植→前進→収穫→種植→前進→収穫→左回→左回→種植→
前進→収穫→種植→前進→収穫→左回→左回→種植→前進→収穫→
種植→前進→収穫→左回→左回→種植→前進→収穫→種植→前進→
収穫→左回→左回→種植→前進→収穫→種植→前進→収穫→左回→
左回→種植→前進→収穫→種植→前進→収穫→左回→左回→種植→
前進→収穫→種植→前進→収穫→左回→左回→種植→前進→収穫→

### 2位 合計スコア: 717 類似度: 0.74 適合率: 0.75 再現率: 0.74

```
public void Execute() {
    //Your code here!
Forward();
Planting();
Forward();
Planting();
Forward();
Planting();

for (int i=0; i<=19; i++) {
TurnLeft();
}

Harvest();
Back();
Harvest();
Back();
Harvest();

Planting();
Forward();
Planting();
Forward();
Planting();


for (int i=0; i<=19; i++) {
TurnLeft();
}
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→
前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

他者の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

### 3位 合計スコア: 630 類似度: 0.36 適合率: 0.38 再現率: 0.34

```
public void Execute() {
    //Your code here!
while(true){

Forward();
Harvest();
Planting();


Forward();
Harvest();
Planting();


Forward();
Harvest();
Planting();


TurnLeft();
TurnLeft();


Forward();
Harvest();
Planting();


Forward();
Harvest();
Planting();


Forward();
Harvest();
Planting();


Forward();


TurnLeft();
```

自身の振舞い

前進→種植→前進→種植→前進→種植→左回→左回→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→収穫→前進→
収穫→前進→収穫→種植→前進→種植→前進→種植→左回→左回→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
収穫→前進→収穫→前進→収穫→種植→前進→種植→前進→種植→
左回→左回→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→収穫→前進→収穫→前進→収穫→種植→前進→種植→
前進→種植→左回→左回→左回→左回→左回→左回→左回→左回→
左回→左回→左回→左回→収穫→前進→収穫→前進→収穫→

他者の振舞い

前進→収穫→種植→前進→収穫→種植→前進→収穫→種植→左回→
左回→前進→収穫→種植→前進→収穫→種植→前進→左回→左回→
前進→収穫→種植→前進→収穫→種植→前進→収穫→種植→左回→
左回→前進→収穫→種植→前進→収穫→種植→前進→左回→左回→
前進→収穫→種植→前進→収穫→種植→前進→収穫→種植→左回→
左回→前進→収穫→種植→前進→収穫→種植→前進→左回→左回→
前進→収穫→種植→前進→収穫→種植→前進→収穫→種植→左回→
左回→前進→収穫→種植→前進→収穫→種植→前進→収穫→種植→
前進→収穫→種植→前進→収穫→種植→前進→左回→左回→
左回→前進→収穫→種植→前進→収穫→種植→前進→左回→左回→

図 7 生成されたランキングの結果