Vol.2020-CE-157 No.6
2020/11/7

コンピュータの整列処理で正解との距離は単調に減少するか?

山口 琢[1]　大場 みち子[2]

概要：学習者の演習など知的プロセスを測定・分析・評価する研究では、「適切なプロセスでは、プロセスの進展に伴って正解との距離が単調に減少する」ことを前提とする場合がある。この前提は妥当だろうか? われわれは、コンピュータによる整列を題材に、これを確かめた。1 から 10 の 10 個の数をクイックソートなどの整列アルゴリズムでコンピュータに整列させ、途中の数列を記録した。そして、途中の数列と正解の数列との距離を計算して変化を調べた。距離としてはレーベンシュタイン距離やケイリー距離などを採用した。コンピュータによる整列では、プロセスの途中で正解との距離が遠ざかることがあるし、距離 0 すなわち正解に達しても、さらにプロセスが続いて遠ざかることがあると分かった。単調減少の前提は、無条件に妥当ではないと考えられる。今後は、プログラム・コードの並べ替えパズルを題材に、人による並べ替えについても単調減少の前提を検証する。

Does the Distance to the Correct Answer Decrease Monotonically in the Sorting Process by a Computer?

Taku Yamaguchi[1]　Michiko Oba[2]

1. 背景

特にプログラミング教育において、学習者の演習プロセスを測定・分析して、学習者のつまづきや理解を測ろうという研究が行われるようになってきた [1][2][3][4][5]。

人の知的プロセスを測定・分析・評価するこのような研究では、「適切なプロセスでは、プロセスの進展に伴って正解との距離が単調に減少する」ことを前提とする場合がある。ここで「単調に減少する」とは広い意味で、$x < y$ のときに $f(x) \leq f(y)$ であること、言い換えると単調非増加することを指して、あるステップで距離が変わらない場合も含むこととする。

プロセス (過程) を対象とする学習分析では、プロセスをデータ化/測定する手法や測定データを分析する手法などが提案される。人の考えを直接見ることはできないが、分析に基づく推定と正答率との相関を調べたり、発話プロトコルと照合したりすることで提案手法が評価される (**図 1**)。編集距離を使う研究では、この測定・分析で、プロセスの各ステップでの正解との距離を計算し、それら距離の変化を分析して学習者の状態を推定する。

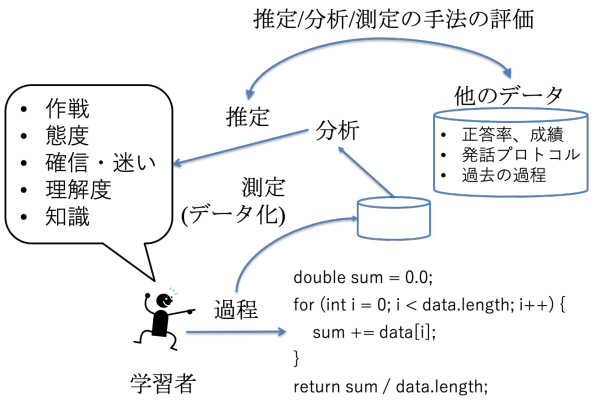

図 1　学習分析の研究の構造

[1] フリー
Independent Researcher

[2] 公立はこだて未来大学システム情報科学部
Faculty of Systems Information Science, Future University Hakodate

ⓒ 2020 Information Processing Society of Japan

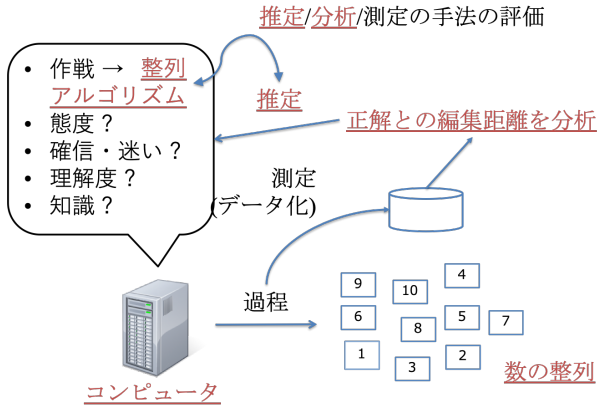

図 2　コンピュータによる整列を題材に、正解との編集距離の変化を調査

2.　目的

人が知的な処理を行うとき、「適切なプロセスでは、プロセスの進展に伴って正解との編集距離が単調に減少する」という前提は妥当だろうか?。本研究の目的は、コンピュータによる整列を題材に、人に対するこの前提の妥当性を検証することである。図 1 での学習者、すなわち人をコンピュータにおきかえて、コンピュータが数を整列するプロセス (過程) を測定し、正解すなわち整列済み数列 (数の並び) との距離を計算して単調に減少するかを検証する (図 **2**)。

整列を題材にするのは、プログラミングよりも単純なプロセスだからである。また、コンピュータによる整列を調べるのは、中で何をやっているかアルゴリズムが明白で、かつ、デバッガを使ったりコードを書き換えることでプロセスの途中の状態を詳細に調べられるからである。

人の知的プロセスに関する前提を検証するため、コンピュータによる処理を測定・分析するにあたって、条件を設定する。これについては 4 節で述べる。

人による整列とコンピュータによる整列が同じなのか？については検証が必要であるが、それは別稿に譲り、本稿の範囲外とする。

3.　関連する研究

3.1　正解との距離に基づく分析

図 1 の測定・分析で編集距離を使う研究では、プロセスの各ステップでの正解との距離を計算し、それら距離の変化を調べて学習者の状態を推定する。「適切なプロセスでは成果との距離が単調に減少する」ことを前提にすると、途中で距離が増加する/広がる場合、学習者に何らかの問題が起きていると推定できる。例えば、被験者/学習者は Tinker してる (下手にいじくり回してる) と推定することがある (図 **3**)。

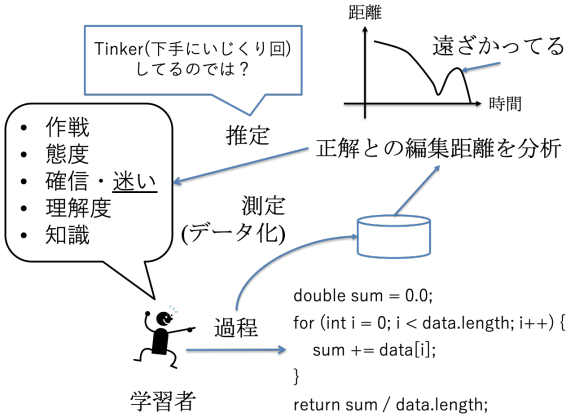

図 3　正解から遠ざかることに基づく推定

3.2　コンピュータによる整列

データを一定のルールに従って並べ替えることを整列、またはソート (sort) といい [6]、単純だが知的なプロセスである。その研究は古く、コンピューターによる整列はコンピューターの歴史と共にあると言ってよい。

コンピュータによる整列は、次のような論点で研究されてきた:

- 結果が正しいのは当たり前
- 性能、ステップ数、処理時間
- メモリ所要量
- 安定性。同じ順序だった複数のデータの順序が、整列の前後で変わるかどうか。
- データ操作の手法: データの位置の替え方、交換、挿入、選択、マージ、など
- データ操作の作業領域: 数列の内部か、外部か

本稿あるいは学習分析の観点からは、コンピュータによる整列は次の特徴を持つ:

- 必ず正解する。プロセスだけが異なる。正答率の変化で評価するとか正解を教師として機械学習するといった、学習分析でよく採用される「結果に基づく分析」手法を使えない。
- アルゴリズムが明らかである。本人にインタビューして「何考えてましたか？」と聞くまでもない。(何より自分で組んだアルゴリズムで動いているのだから。)
- 再現できる。人間による整列よりもコンピューターによる整列の方が再現性が高い。
- 網羅性が高い。(人間に並べ替え可能な)10 個程度の数であれば、パソコンで全順列を網羅できる。

3.3　人による並べ替え操作の測定・分析

我々はこれまで、並べ替えプログラミング・パズルのジグソー・コードを開発して、学習者が並べ替えプログラミングするとき、並べ替え操作を測定した。操作の傾向を分析して、学習者へアドバイスすべき内容を明らかにした [1][3]。

ⓒ 2020 Information Processing Society of Japan

ジグソー・コードは、PC やスマホの UI で広く採用されているドラッグ&ドロップ操作でプログラムの断片 (ピース) を並べ替えて完成させる。開始、ドラッグ開始、ドロップ、完成ボタンというタイミングで、ピース全体の並び順や時刻などを記録する。ドラッグやドロップでは、操作対象のピースも記録する。

4. アプローチ

単調減少の前提の妥当性を、コンピュータによる数の整列を題材に検証する。このアプローチは、コンピュータに整列させる問題の作問と正解、検証の対象とする整列アルゴリズム、プロセスの途中で正解との距離を測る測定タイミング、採用する距離、距離の変化の評価方法で構成される。

4.1 作問と正解

1 から 10 の 10 個の数を昇順に並べ替えることを問題とする。全順列 ${}_{10}P_{10} = 3,628,800$ 個の問題をコンピュータに解かせる。

これら問題の正解は (1,2,3,4,5,6,7,8,9,10) である。問題の中には、最初から正解で並べ替え不要のものも含まれる。

4.2 整列アルゴリズム

整列アルゴリズムとして、バブルソート、クイックソート、ヒープソート、マージソート、選択ソート、挿入ソートの 6 個を検証する。

挿入ソートはドラッグ&ドロップと同じ操作でデータを操作する。本稿の範囲外であるが、ジグソー・コードの並べ替え操作と同じなので、将来の参考のために検証対象とする。

コンピュータに実行させる整列プログラムは、文献 [6] のアルゴリズムを Python で実装する。

4.3 測定タイミング

並べ替えの進展に伴う正解との距離の変化を調べるために、タイミングを決めて、その瞬間の数列のスナップショットを記録する。そのタイミングでは、数列に欠けている数があってはならないし、数が重複していてもならない。これには、3.2 節で述べた、整列アルゴリズムのデータ操作の手法やデータ操作の作業領域が関係する。

バブルソート、ヒープソート、クイックソート、選択ソートは、処理の途中で数を交換 (入れ替え、swap) するが、その交換後に数列を記録する。

マージソートは、整列済の数列を 1 つの数列にマージするが、このマージの直後に数列を記録する。

挿入ソートは、数を挿入した直後に数列を記録する。

destination	2, 5, 3, 1, 4
start	1, 2, 3, 4, 5
swap(1, 4) *	<u>4</u>, 2, 3, <u>1</u>, 5
swap(1, 5)	<u>5</u>, 2, 3, 1, <u>4</u>
swap(1, 2)	<u>2</u>, <u>5</u>, 3, 1, 4
number of cycles	3
	(* swapの引数は添字)

図 4 ケイリー距離の例図 N キャプション

4.4 距離

学習分析の研究では、学習アプリケーションで可能な操作に応じて、編集距離を定義する場合がある [2]。整列問題はデータ操作を限定しない。そのうえで、前節で述べたとおり、整列アルゴリズムは、置換、マージ、挿入といった様々な独自の規則でデータを操作する。人の知的プロセスに関する研究では、現状、これらアルゴリズムに相当する考え方を突き止めることが目標の 1 つであり、事前にアルゴリズムを知って距離を決めることはできない。本稿では、アルゴリズムごとに距離を限定することをせず、いくつかの距離を採用して全てのアルゴリズムに適用して分析する。

距離としてはレーベンシュタイン距離 (Levenshtein distance)、ハミング距離 (Hamming distance)、ケイリー距離 (Cayley distance) およびウラム距離 (Ulam distance) の 4 個を採用する。レーベンシュタイン距離以外は、同じ文字数の文字などに適用される距離である。

レーベンシュタイン距離は、1 要素の挿入・削除・置換 (その要素を任意の別の要素に置き換える) 操作を何回繰り返すと、一方の並びから他方の並びへ変わるかの操作回数である。断りなく編集距離というとレーベンシュタイン距離を指すことが多い。

ハミング距離は、2 つの並びの同じ位置にあって異なる要素の個数である。(2,5,3,1,4) と (1,2,3,4,5) の距離は、3 だけが同じで他は異なるので、4 である。

ケイリー距離は「交換」操作によってのみ並びを変換して、一方の並びから他方の並びにいたるまでの変換の最小回数を距離とするものである [7]。この操作は、クイックソートなどが行う交換に該当する。**図 4** はケイリー距離の例である。destination と start との距離を測っている。swap(1, 4) は交換操作を表し、1 や 4 は各数の添字である。前の行に交換操作を行った結果が swap() の行に表示されている。swap(1, 4) は、添字 1 と添字 4 を交換、すなわち添字 1=1、添字 4=4 を交換することを表す。交換操作で交換された数に下線が引かれている。start から 3 回の交換で destination と同じになったので、ケイリー距離は 3 である。

ケイリー距離は、Python ならば SymPy ライブラリの Permutations[8] を使って、R であれば PerMallows パッ

表 1　正解との距離の変化

	Levenshtein 距離		Hamming 距離		Caley 距離		Ulam 距離	
	増加した問題数	割合	増加した問題数	割合	増加した問題数	割合	増加した問題数	割合
バブルソート	2,743,264	75.6%	3,512,825	96.8%	3,624,619	99.9%	0	0.0%
ヒープソート	3,628,800	100.0%	3,628,800	100.0%	3,628,800	100.0%	3,628,800	100.0%
挿入ソート	1,339,176	36.9%	1,964,215	54.1%	3,443,878	94.9%	0	0.0%
マージソート	1,852,755	51.1%	2,432,789	67.0%	3,426,453	94.4%	0	0.0%
クイックソート	1,034,478	28.5%	0	0.0%	1,089,872	30.0%	872,908	24.1%
選択ソート	0	0.0%	0	0.0%	0	0.0%	0	0.0%

ケージ [9] を使って計算できる。SymPy は代数計算のライブラリである。

ウラム距離の操作は、スマホの UI などのドラッグ&ドロップに該当する。ウラム距離は、目的の並びになるまでの最小の操作数である [7]。これは、挿入ソートが行う操作に該当する。ウラム距離は、並びの長さから最長共通部分 (Longest Common Subsequence) を引いた値に等しい [7] ので、これを Python で計算した。R であれば PerMallows パッケージ [9] を使って計算できる。

4.5　実験と距離の変化の評価

4.2 節の整列アルゴリズムを実装するプログラムに、4.3 節のタイミングで数列を出力するコードを挿入する。これらのプログラムで 4.1 節の問題を整列させて、各タイミングでの数列を記録する。これら記録しておいた整列中の数列と正解との距離を、後で 4.4 節の距離で測る。こうすることで、後から別の距離で分析しなおすこともできる。

こうして得られた距離の列の中で、前の距離よりも大きくなる場合があるかどうかを調べる。

5.　結果

全ての整列アルゴリズムで、全ての距離について、正解との距離が遠ざかる場合があることが分かった (表 1)。

ヒープソートは、全ての問題について、すべての距離で、処理のどこかで正解との距離が増加する。

選択ソートは、全ての問題について、すべての距離で、処理の途中で正解との距離が単調減少した。

6.　考察

本稿の範囲で、ヒープソートが不適切なアルゴリズムだという理由はない。正解との距離が単調減少しないからといって、ヒープソートが不適切だとは言えない。

本稿の範囲で、選択ソートが他のアルゴリズムと異なって優秀だという理由はない。正解との距離が必ず単調減少するからといって、選択ソートが適切だとは言えない。

距離が遠ざかるとき、コンピュータは試行錯誤してるのか？あるいはアルゴリズムは何か特別な処理をしてるのか？正解に近づく場合と遠ざかる場合とで、実行されるコードは同じである。仮に、それぞれの場合で実行されるコードが異なれば、近づく場合は順調、遠ざかる場合は不調といった解釈への道が拓けるかもしれない。しかし、そうではない。

図 5 はクイックソートの途中で正解との距離が広がった例である。一番左の列の (9,10) などは数列に対する置換操作で、その上の行の数列で 9 と 10 を入れ替えた結果、その行の数列になったことが分かる。(9,10) の置換では距離が縮まったが、次の (6, 8) の置換ではレーベンシュタイン距離が広がった。(9,10) の置換にいたるまで処理 (for 文、while 文といった繰り返しや if 文の判定) も、(6, 8) の置換にいたる処理も同じプログラム・コードによるものである。

正解との距離が単調減少することを、良い整列アルゴリズムの条件とするならば、では距離を確かめながら処理することは妥当か？例えば、各整列アルゴリズムにおいて、置換などの並べ替えを行う前に「並べ替え後の正解との距離を調べて、遠ざかる場合は並べ替えしない」という処理をすれば、単調減少するだろうか。これは上手くいかない。なぜならば、正解が分からないからこそ整列するからである。正解を知っていて、それとの距離を確かめられるなら、そもそも整列不要である。従って、正解を知らずに正解との距離を知るアルゴリズムが必要となる。それを実装したとき、その整列アルゴリズムはもはや、元の整列アルゴリズムとは別物であろう。

図 6 では、距離が 0、すなわち正解に達した後でも処理を続けている。ここでも、正解に達したかを確かめながら処理を進めれば、このような事態を避けられるが、それはできないのは同様である。

その他の例を付録の A.1 に載せる。

6.1　結論

以上の考察を踏まえると、コンピュータによる整列において「適切なプロセスでは正解との距離が単調に減少する」とは言えない。

単調減少を前提とするためには、何らかの条件が必要だろう。本稿であれば、選択ソート以外のアルゴリズムを採用することが不適切と言える条件、あるいは距離として Ulam 距離を採用することが妥当と言える条件である。

ⓒ 2020 Information Processing Society of Japan

swap-vl		levenshtein	hamming	cayley	ulam
start	1, 2, 3, 4, 5, 7, 8,10, 6, 9	4 ####	5 #####	4 ####	2 ##
(9,10)	1, 2, 3, 4, 5, 7, 8, 9, 6,10	2 ##	4 ####	3 ###	1 #
(6, 8)	1, 2, 3, 4, 5, 7, 6, 9, 8,10	3 ###	4 ####	2 ##	2 ##
(6, 7)	1, 2, 3, 4, 5, 6, 7, 9, 8,10	2 ##	2 ##	1 #	1 #
(8, 9)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
answer	1, 2, 3, 4, 5, 6, 7, 8, 9,10				

図 5　クイックソートの距離の変化

swap-vl		levenshtein	hamming	cayley	ulam
start	1, 2, 3, 4, 9, 7,10, 5, 6, 8	6 ######	6 ######	5 #####	3 ###
(2, 1)	2, 1, 3, 4, 9, 7,10, 5, 6, 8	8 ########	8 ########	6 ######	4 ####
(3, 2)	3, 1, 2, 4, 9, 7,10, 5, 6, 8	8 ########	9 #########	7 #######	4 ####
(4, 1)	3, 4, 2, 1, 9, 7,10, 5, 6, 8	9 #########	10 ##########	8 ########	5 #####
(4, 3)	4, 3, 2, 1, 9, 7,10, 5, 6, 8	9 #########	10 ##########	7 #######	6 ######
(9, 3)	4, 9, 2, 1, 3, 7,10, 5, 6, 8	8 ########	10 ##########	8 ########	5 #####
(9, 4)	9, 4, 2, 1, 3, 7,10, 5, 6, 8	8 ########	10 ##########	9 #########	5 #####
(7, 2)	9, 4, 7, 1, 3, 2,10, 5, 6, 8	9 #########	10 ##########	8 ########	5 #####
(10, 7)	9, 4,10, 1, 3, 2, 7, 5, 6, 8	9 #########	9 #########	7 #######	5 #####
(10, 9)	10, 4, 9, 1, 3, 2, 7, 5, 6, 8	9 #########	9 #########	8 ########	5 #####
(5, 1)	10, 4, 9, 5, 3, 2, 7, 1, 6, 8	9 #########	9 #########	7 #######	6 ######
(5, 4)	10, 5, 9, 4, 3, 2, 7, 1, 6, 8	8 ########	8 ########	6 ######	7 #######
(6, 4)	10, 5, 9, 6, 3, 2, 7, 1, 4, 8	9 #########	9 #########	7 #######	6 ######
(6, 5)	10, 6, 9, 5, 3, 2, 7, 1, 4, 8	9 #########	9 #########	6 ######	7 #######
(8, 3)	10, 6, 9, 5, 8, 2, 7, 1, 4, 3	9 #########	9 #########	7 #######	8 ########
(8, 6)	10, 8, 9, 5, 6, 2, 7, 1, 4, 3	8 ########	9 #########	8 ########	7 #######
(10, 3)	3, 8, 9, 5, 6, 2, 7, 1, 4,10	7 #######	8 ########	7 #######	5 #####
(9, 3)	9, 8, 3, 5, 6, 2, 7, 1, 4,10	6 ######	7 #######	6 ######	5 #####
(7, 3)	9, 8, 7, 5, 6, 2, 3, 1, 4,10	8 ########	9 #########	7 #######	6 ######
(9, 4)	4, 8, 7, 5, 6, 2, 3, 1, 9,10	7 #######	8 ########	6 ######	5 #####
(8, 4)	8, 4, 7, 5, 6, 2, 3, 1, 9,10	7 #######	8 ########	5 #####	5 #####
(6, 4)	8, 6, 7, 5, 4, 2, 3, 1, 9,10	8 ########	8 ########	4 ####	6 ######
(8, 1)	1, 6, 7, 5, 4, 2, 3, 8, 9,10	6 ######	6 ######	3 ###	4 ####
(7, 1)	7, 6, 1, 5, 4, 2, 3, 8, 9,10	7 #######	7 #######	4 ####	4 ####
(3, 1)	7, 6, 3, 5, 4, 2, 1, 8, 9,10	6 ######	6 ######	3 ###	5 #####
(7, 1)	1, 6, 3, 5, 4, 2, 7, 8, 9,10	4 ####	4 ####	2 ##	3 ###
(6, 1)	6, 1, 3, 5, 4, 2, 7, 8, 9,10	5 #####	5 #####	3 ###	3 ###
(5, 1)	6, 5, 3, 1, 4, 2, 7, 8, 9,10	5 #####	5 #####	4 ####	4 ####
(6, 2)	2, 5, 3, 1, 4, 6, 7, 8, 9,10	4 ####	4 ####	3 ###	2 ##
(5, 2)	5, 2, 3, 1, 4, 6, 7, 8, 9,10	3 ###	3 ###	2 ##	2 ##
(4, 2)	5, 4, 3, 1, 2, 6, 7, 8, 9,10	4 ####	4 ####	3 ###	3 ###
(5, 2)	2, 4, 3, 1, 5, 6, 7, 8, 9,10	3 ###	3 ###	2 ##	2 ##
(4, 2)	4, 2, 3, 1, 5, 6, 7, 8, 9,10	2 ##	2 ##	1 #	2 ##
(4, 1)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
(3, 1)	3, 2, 1, 4, 5, 6, 7, 8, 9,10	2 ##	2 ##	1 #	2 ##
(3, 1)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
(2, 1)	2, 1, 3, 4, 5, 6, 7, 8, 9,10	2 ##	2 ##	1 #	1 #
(2, 1)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
answer	1, 2, 3, 4, 5, 6, 7, 8, 9,10				

図 6　ヒープソートの距離の変化
距離 0、すなわち正解になった後で遠ざかっている

7.　残された課題

本稿は「人の知的プロセスの分析に正解との距離は役に立たない」と主張しているわけではない。整列アルゴリズムと距離の組み合わせについて、なぜ表 1 の結果になるのか？、特に 100%や 0%のセルの存在は興味深い。なぜ、全ての問題について正解との距離が増えることがあるのか。なぜ、全ての問題について正解との距離が増えないのか。これについては、今後の研究に委ねる。

本稿を踏まえて、人による並べ替えについても検証する。表 1 の中の 51%を「処理の途中で距離が増加する確率」と読み替えられるのは、全ての問題が同じ確率で起きる場合である。現実の、人による並べ替えでは異なるかもしれない。3.3 節で述べた研究 [1][3] のデータについて、正解との距離が単調に減少するか検証する。

謝辞　本研究は JSPS 科研費 20H01728 の助成を受けたものである。

ⓒ 2020 Information Processing Society of Japan

参考文献

[1] 中村 陽太, 大場 みち子, 山口 琢, 伊藤 恵, 学習進度に対応するパズルを利用したプログラミング思考過程の分析, 情報処理学会研究報告コンピュータと教育 (CE), 2019-CE-151(1), 2019-09-28
http://id.nii.ac.jp/1001/00199559/

[2] 森永 笑子, 松本 慎平, 林 雄介, 平嶋 宗, カード操作方式によるプログラミング学習支援システムにおける学習者の活動に基づく学習課題の特徴分析, 電気学会 C 部門 情報システム研究会 (CIS), 2019-11-10
http://id.nii.ac.jp/1031/00126864/

[3] 中村 陽太, 大場 みち子, 山口 琢, 伊藤 恵, 授業進度に対応するパズルを利用したプログラミング思考過程の分析と教育支援システムの開発, 情報処理学会 第 82 回全国大会講演論文集, 2020-02-20
http://id.nii.ac.jp/1001/00205909/

[4] Salil Maharjan and Amruth Kumar, Using Edit Distance Trails to Analyze Path Solutions of Parsons Puzzles, Educational Data Mining 2020 (EDM 2020), 2020-07-10
https://educationaldatamining.org/files/conferences/EDM2020/papers/paper_163.pdf

[5] 伊東大輝, 島川博光, コードパズルによるプログラミング的思考力を考慮した理解度の推定, 情報科学技術フォーラム (FIT2020) 講演論文集, 2020

[6] 奥村晴彦, ［改訂新版］C 言語による標準アルゴリズム事典, 技術評論社, 2018
https://gihyo.jp/dp/ebook/2018/978-4-7741-9787-6

[7] 神嶌敏弘, 順序の距離と確率モデル, 人工知能学会 データマイニングと統計数理研究会,2009 文献本体
http://id.nii.ac.jp/1004/00000187/

[8] SymPy Development Team, SymPy 1.6.2 documentation, 2020-08-09.
https://docs.sympy.org/latest/index.html

[9] CRAN, PerMallows: Permutations and Mallows Distributions, 2017-04-28
https://cran.r-project.org/package=PerMallows

付　　録

A.1 距離の変化の例

整列処理の途中での距離の変化の例を示す。**図 A.1** から図 A-6 で、一番左の列は置換等の操作を表す。例えば図 A.1 では、start 行の数列に対し、3 と 8 を交換した (3,8) 結果が start の 1 つ下の行である。

操作		levenshtein	hamming	cayley	ulam
start	1, 2, 4,10, 9, 5, 7, 6, 8, 3	7 #######	7 #######	5 #####	4 ####
(3, 8)	1, 2, 4,10, 9, 5, 7, 6, 3, 8	7 #######	7 #######	6 ######	4 ####
(3, 6)	1, 2, 4,10, 9, 5, 7, 3, 6, 8	7 #######	7 #######	5 #####	4 ####
(3, 7)	1, 2, 4,10, 9, 5, 3, 7, 6, 8	7 #######	8 ########	6 ######	4 ####
(3, 5)	1, 2, 4,10, 9, 3, 5, 7, 6, 8	8 ########	8 ########	7 #######	4 ####
(3, 9)	1, 2, 4,10, 3, 9, 5, 7, 6, 8	7 #######	8 ########	6 ######	4 ####
(3,10)	1, 2, 4, 3,10, 9, 5, 7, 6, 8	7 #######	8 ########	5 #####	4 ####
(3, 4)	1, 2, 3, 4,10, 9, 5, 7, 6, 8	6 ######	6 ######	4 ####	3 ###
(6, 7)	1, 2, 3, 4,10, 9, 5, 6, 7, 8	4 ####	6 ######	5 #####	2 ##
(5, 9)	1, 2, 3, 4,10, 5, 9, 6, 7, 8	4 ####	6 ######	4 ####	2 ##
(5,10)	1, 2, 3, 4, 5,10, 9, 6, 7, 8	4 ####	5 #####	3 ###	2 ##
(6, 9)	1, 2, 3, 4, 5,10, 6, 9, 7, 8	4 ####	5 #####	4 ####	2 ##
(6,10)	1, 2, 3, 4, 5, 6,10, 9, 7, 8	4 ####	4 ####	3 ###	2 ##
(7, 9)	1, 2, 3, 4, 5, 6,10, 7, 9, 8	3 ###	3 ###	2 ##	2 ##
(7,10)	1, 2, 3, 4, 5, 6, 7,10, 9, 8	2 ##	2 ##	1 #	2 ##
(8, 9)	1, 2, 3, 4, 5, 6, 7,10, 8, 9	2 ##	3 ###	2 ##	1 #
(8,10)	1, 2, 3, 4, 5, 6, 7, 8,10, 9	2 ##	2 ##	1 #	1 #
(9,10)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
answer	1, 2, 3, 4, 5, 6, 7, 8, 9,10				

図 A·1 バブルソートでの正解との距離の変化
途中で距離が増加している

操作		levenshtein	hamming	cayley	ulam
start	10, 1, 5, 8, 9, 2, 3, 4, 6, 7	8 ########	10 ##########	8 ########	4 ####
(1,)	1,10, 5, 8, 9, 2, 3, 4, 6, 7	8 ########	9 #########	7 #######	4 ####
(5,)	1, 5,10, 8, 9, 2, 3, 4, 6, 7	8 ########	9 #########	6 ######	4 ####
(8,)	1, 5, 8,10, 9, 2, 3, 4, 6, 7	8 ########	9 #########	7 #######	4 ####
(9,)	1, 5, 8, 9,10, 2, 3, 4, 6, 7	8 ########	9 #########	8 ########	4 ####
(2,)	1, 2, 5, 8, 9,10, 3, 4, 6, 7	8 ########	8 ########	6 ######	4 ####
(3,)	1, 2, 3, 5, 8, 9,10, 4, 6, 7	6 ######	7 #######	4 ####	3 ###
(4,)	1, 2, 3, 4, 5, 8, 9,10, 6, 7	4 ####	5 #####	4 ####	2 ##
(6,)	1, 2, 3, 4, 5, 6, 8, 9,10, 7	2 ##	4 ####	3 ###	1 #
(7,)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
answer	1, 2, 3, 4, 5, 6, 7, 8, 9,10				

図 A·2 挿入ソートでの正解との距離の変化
途中で距離が増加している

操作		levenshtein	hamming	cayley	ulam
start	1, 2, 3, 4, 5, 7, 9, 6,10, 8	4 ####	5 #####	4 ####	2 ##
(6, 7)	1, 2, 3, 4, 5, 6, 9, 7,10, 8	4 ####	4 ####	3 ###	2 ##
(7, 9)	1, 2, 3, 4, 5, 6, 7, 9,10, 8	2 ##	3 ###	2 ##	1 #
(8,10)	1, 2, 3, 4, 5, 6, 7, 9, 8,10	2 ##	2 ##	1 #	1 #
(8, 9)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
answer	1, 2, 3, 4, 5, 6, 7, 8, 9,10				

図 A·3 クイックソートでの正解との距離の変化
距離が単調に減少している

ⓒ 2020 Information Processing Society of Japan

操作		levenshtein	hamming	cayley	ulam
start	10, 1, 5, 8, 9, 2, 3, 4, 6, 7	8 ########	10 ##########	8 ########	4 ####
(1,10)	1,10, 5, 8, 9, 2, 3, 4, 6, 7	8 ########	9 #########	7 #######	4 ####
(2,10)	1, 2, 5, 8, 9,10, 3, 4, 6, 7	8 ########	8 ########	6 ######	4 ####
(3, 5)	1, 2, 3, 8, 9,10, 5, 4, 6, 7	7 #######	7 #######	5 #####	4 ####
(4, 8)	1, 2, 3, 4, 9,10, 5, 8, 6, 7	5 #####	5 #####	4 ####	3 ###
(5, 9)	1, 2, 3, 4, 5,10, 9, 8, 6, 7	4 ####	4 ####	3 ###	3 ###
(6,10)	1, 2, 3, 4, 5, 6, 9, 8,10, 7	3 ###	3 ###	2 ##	2 ##
(7, 9)	1, 2, 3, 4, 5, 6, 7, 8,10, 9	2 ##	2 ##	1 #	1 #
(8, 8)	1, 2, 3, 4, 5, 6, 7, 8,10, 9	2 ##	2 ##	1 #	1 #
(9,10)	1, 2, 3, 4, 5, 6, 7, 8, 9,10	0	0	0	0
answer	1, 2, 3, 4, 5, 6, 7, 8, 9,10				

図 A·4 選択ソートでの正解との距離の変化
距離が単調に減少している