5A-05

複数の順列に共通して現れるパターンの列挙法

井上 祐馬 †
北海道大学 大学院情報科学研究科 †

湊 真一 ‡
北海道大学 大学院情報科学研究科 ‡

1. はじめに

順列は 1 から n までの整数がちょうど 1 度ずつ出現する長さ n の数列である．順列にはパターン照合問題が存在し，テキスト順列 τ にパターン順列 π がマッチするとは，「π と相対順序が一致するような τ の部分列が存在すること」と定義される．厳密には次節で定義する．順列パターンは，解が順列となる様々な問題の特徴付けに用いられ，例えば一時記憶としてスタックのみを使用するソート (スタックソート) で整列可能な入力は，順列 $(2,3,1)$ にマッチしない順列に対応する [3].

本稿では，複数のテキストすべてに共通してマッチするパターンを全列挙するアルゴリズムを提案する．提案手法は，順列集合を圧縮して効率的に処理するために，順列決定グラフ [2] というデータ構造を用いることで高速化を実現している．計算機実験により，ナイーブな手法に比べ 10 倍以上高速に動作することを確認した．

2. 順列パターン

テキスト順列 $\tau = (\tau_1,\ldots,\tau_n)$ がパターン順列 $\pi = (\pi_1,\ldots,\pi_m)$ を含むとは，あるインデックス $1 \le i_1 < \cdots < i_m \le n$ が存在して，$\tau_{i_x} < \tau_{i_y}$ のとき，かつそのときに限り $\pi_x < \pi_y$ であることをいう．例えばテキスト $\tau = (4,2,1,3)$，パターン $\pi = (3,1,2)$ のとき，インデックス 1,3,4 に対する τ の部分列 $(4,1,3)$ がパターン $(3,1,2)$ に対応するため，τ は π を含む．

本稿では，複数のテキストに共通するパターンをすべて列挙する問題について考える．2 つのテキスト順列の最長共通パターンを見つける問題が NP 完全であることが知られている [1] など，1 つの共通パターンを求めることでさえ難しい問題であり，全共通パターンの列挙に取り組んだ先行研究は筆者の知る限りない．

3. 順列決定グラフ

順列決定グラフは順列の集合を圧縮表現するデータ構造である．集合に含まれる各順列をある分解法で分解する手順を二分決定木で表し，それを圧縮することで順列決定グラフが得られる．本稿では，分解法として左ローテーションを採用した Rot-πDD [2] を用いる．ここで，区間 $[x,y]$ に対する左ローテーションは，x 番目の要素を y 番目に移動し，$x+1$ 番目から y 番目までの要素を左に 1 つ移動する操作であり，以降 $\rho_{x,y}$ と書く．

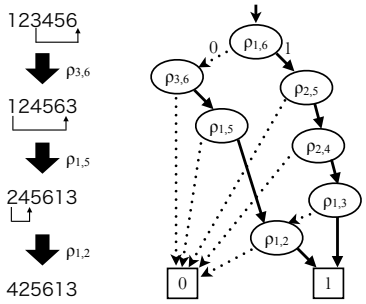

図 1　Rot-πDD の例

Rot-πDD の例を図 1 に示す．Rot-πDD の元となる決定木は，左ローテーション $\rho_{x,y}$ を表す内部節点と，$\rho_{x,y}$ を行うか否かに対応する 1-枝と 0-枝を持つ．まず整列された順列 $(1,\ldots,n)$ を持って，木を降り始める．木を降る際，1-枝を通るなら節点に書かれた $\rho_{x,y}$ を行い，0-枝を通るなら行わない．順列に対し左ローテーションを繰り返して終端節点に辿り着いたとき，対応する順列が集合に含まれるなら $\top$ 節点，含まれないなら $\bot$ 節点を終端とする．この決定木に対して 2 つの圧縮ルール (1) 1-枝が $\bot$ 節点を指す節点を削除，(2) 等価な節点 ($\rho_{x,y}$, 0-枝の先，1-枝の先がそれぞれ同じ) をすべて 1 つに統合，を適用すると，圧縮された Rot-πDD が得られる．

Rot-πDD は 2 つの Rot-πDD 同士の演算が可能で，2 つの順列集合の共通集合を求める演算や，片方の集合の各順列の並びに従ってもう片方の集合の各順列を並び替える演算などが存在する．演算の効率は順列の個数ではなく Rot-πDD の節点数に依存している．

4. 提案法

提案法では 1 つの順列に含まれる全パターンを格納する Rot-πDD を構築する．それぞれの順列に含まれるパターンを表す Rot-πDD を構築すれば，それらの共通集合を Rot-πDD の演算で計算することで共通パターンが列挙できるためである．

Generation of Common Patterns in Permutations
† Yuma Inoue, Hokkaido University
‡ Shin-ichi Minato, Hokkaido University

1つの順列に含まれるパターンを表すRot-πDDは，長さkの全パターンのRot-πDDから長さ$k-1$の全パターンのRot-πDDを構築する，という手順の繰り返しで構築する．入力の順列πの長さをnとすると，長さnの唯一のパターンπのみを表すRot-πDDから始める．

長さkのあるパターンπから長さ$k-1$のパターンを作るには，πの要素を1つ削除し，残った要素の順序を整理すればよい．これを長さkの全パターンの全要素に対して行えば，重複を許して長さ$k-1$の全パターンを列挙できる．長さkのパターンから要素iを削除するには，(1) iをkに，すべてのi以上の要素jを$j-1$に変更する，(2) kを一番右のk番目に寄せて削除する，という手順を踏むことで可能になる．一見冗長に思えるが，この手順はRot-πDDの構造をうまく利用できる．

手順 (1) は，長さ$k-1$の昇順の順列のi番目にkを挿入した順列$\sigma_{i,k}$を考えることで簡単に実現できる．例えばパターン$(4,2,1,3)$の2を4に，3と4をそれぞれ2と3に変更し，$(3,4,1,2)$を作る場合を考える．これは，$\sigma_{2,4}=(1,4,2,3)$を，元のパターン$\pi=(4,2,1,3)$に従って並べ替える，すなわち$\sigma_{2,4}$のπ_i番目をi番目に移動すればよい．例えば$\sigma_{2,4}$の$\pi_1=4$番目は3なので，移動後の1番目が3になる．このようにすると，目的の$(3,4,1,2)$が得られる．$\sigma_{i,k}$は右ローテーションに対応するため，隣接の左ローテーションの繰り返し$\rho_{k-1,k}\cdots\rho_{i,i+1}$で表現できる．よって，長さ$k$のパターン集合を表すRot-$\pi$DDと$\{\sigma_{1,k},\ldots,\sigma_{k,k}\}$を表すRot-$\pi$DDから，並べ替えの演算1回で手順 (1) が完了する．

手順 (2) ではkをk番目に寄せたいが，ここでRot-πDDの構造では最初$(1,\ldots,k)$から降り始めていたことを思い出すと，最初からkはk番目にあったのに，後の左ローテーションによって移動していることがわかる．よって，$\rho_{x,k}$を$\rho_{x,k-1}$に変更することで，kがk番目から動かなくなり，結果的にkをk番目に移動させたことになる．ただし，$\rho_{x,k}$の次の左ローテーション (=1-枝の先) が$\rho_{x',k-1}$の場合，こちらも$\rho_{x',k-2}$に変更する，というように，区間がkから連続する限り変更し続ける．

まとめると，順列πに含まれる長さkの全パターンを表すRot-πDD P_kの構築は，$\{\pi\}$のみを表すP_nから始め，(1) $P_k \leftarrow P_{k+1}$と$\{\sigma_{1,k},\ldots,\sigma_{k,k}\}$での並び替え演算結果の$\pi$DD，(2) P_kの根から区間がkから連続する節点を修正，でできる．各手順はそのままでは効率的とは言えないが，Rot-πDDの演算の利用で高速化が期待できる．

5. 実験

提案法を実装し，ナイーブな手法 (テキストの全部分列を抜き出し，順序整理したものをパターンとして抽出) と比較した．C++で実装し，3.20GHz CPU，64 GBメモリのマシンで実行した．入力は同じ長さnのランダムな順列2つとし，2つの順列の共通パターンを全列挙した．

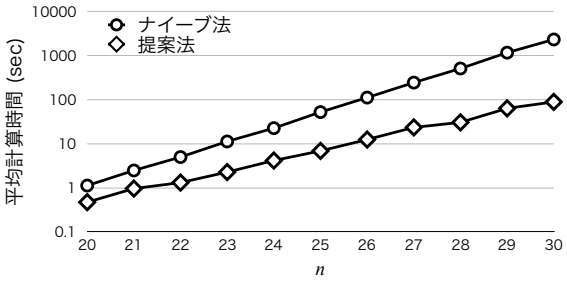

図2　各手法の実行時間の比較

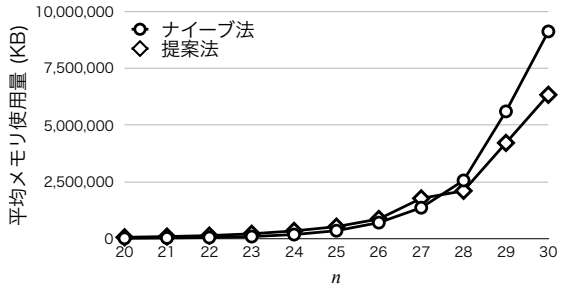

図3　各手法の使用メモリの比較

図2に実行時間の比較結果を示す．2つの順列の長さnが大きくなるにつれ実行時間の差が広がることから，提案法が長い順列に対して相対的に効率がよいことがわかる．実験中最も長い$n=30$については，ナイーブ法の約20倍高速に動作したことが確認できる．

図3に使用メモリの比較結果を示す．実行時間同様，順列が長いほど消費メモリに差がついた．$n=30$のケースでナイーブ法より3GBほどメモリ効率がよかった．

6. おわりに

本稿では，順列決定グラフRot-πDDを用いた複数の順列に共通するパターンの全列挙手法を提案した．提案手法により，ランダムな順列に対しナイーブ法の10倍以上高速な共通パターン列挙が可能となった．

謝辞

本研究はJSPS科研費15J01665および15H05711の助成を受けたものです．

参考文献

[1] Mathilde Bouvel and Dominique Rossin: “The Longest Common Pattern Problem for two Permutations.” Pure Mathematics and Applications, 17 (1–2), pp. 55–69, 2006.
[2] Yuma Inoue and Shin-ichi Minato: “An Efficient Method for Indexing All Topological Orders of a Directed Graph.” In Proc. of 25th International Symposium on Algorithms and Computation (ISAAC), pp. 103–114, 2014.
[3] Donald E. Knuth: “The Art of Computer Programming, Volume 1 : Fundamental Algorithms.” Addison-Wesley, 1968.