Vol.2022-AL-186 No.3
2022/1/27

デカルト木部分列照合問題の高速なアルゴリズム

大泉 翼[1,a)]　有村 博紀[2,b)]

概要：デカルト木照合（Cartesian tree matching）は，値の大小関係に基づいてテキストの連続な部分文字列とパターンの照合を行う近似照合問題の一種であり，2019 年に Park と, Amir, Landau, Park らによって提案された．本研究では，デカルト木照合問題をテキストの非連続な部分系列に拡張して，デカルト木部分列照合問題を提案する．さらに，この問題を $O(mn \log n)$ 時間と $O(mn)$ 領域で解く効率の良いアルゴリズムを与える．このアルゴリズムは，動的計画法と区間最小値クエリを組み合わせることで，先行研究の加井らによるアルゴリズムの時間計算量 $O(mn^3)$ を大幅に改善している．

キーワード：文字列照合，部分系列照合，デカルト木，動的計画法，区間最小値クエリ

Efficient Algorithms for Cartesian Tree Subsequence Matching

Tsubasa Oizumi[1,a)]　Hiroki Arimura[2,b)]

Abstract: Park, Amir, Landau, and Park proposed in 2019 an approximate matching problem for numerical sequences, called the Cartesian tree matching problem, that asks to find a consecutive substring of a text string that matches a pattern string based on their Cartesian trees. In this paper, we study an extension of the Cartesian tree matching problem from consecutive substrings to non-consecutive subsequences, called the Cartesian tree subsequence matching problem. Then, we present an efficient dynamic programming algorithm that runs in $O(mn \log n)$ time and $O(mn)$ space, which improves on the previous $O(mn^3)$ time and space algorithm by Kai *et al.*

Keywords: subsequence matching, cartesian tree, dynamic programming, range minimum query,

1. はじめに

1.1 背景

文字列照合問題 (*string matching problem*) は，情報科学における基礎的な問題の一つであり，与えられた文字集合 Σ 上において，与えられたパターン P に照合するような入力テキスト T の**断片** (*fragment*) を見つける問題として定式化される．テキストの断片としては，連続な**部分文字列** (*substring*) と非連続な**部分列** (*subsequence*) の両方を対象として研究されている．

部分文字列照合に関して，パターンと部分文字列の間にさまざまな近似照合を許すことで，各種の応用に合わせて，次のような様々なタイプのパターンに対する照合問題が研究されている：パラメータ化文字列 (parameterized pattern) [8], [11] [*1]，スワップ付きパターン (pattern with swaps) [1]，オーバーラップパターン (overlap pattern) [2]，入れ替えパターン (jumbled pattern) [5].

1.2 数値列テキストに対する照合問題

とくに，2010 年ごろから数値列テキストに対する拡張が注目を集めている [10], [12]．例えば，株価のパターン分析では，株価の値そのものではなく，株価の変動や値の大小

[1] 北海道大学 大学院情報科学院
Graduate School of Information Science and Technology, Hokkaido University
[2] 北海道大学 大学院情報科学研究院
Faculty of Information Science and Technology, Hokkaido University
[a)] monkukui@eis.hokudai.ac.jp
[b)] arim@ist.hokudai.ac.jp

[*1] コードのメンテナンスに用いられる

関係 $\leq$ で照合したい．このような要求に対して，**順序保存パターン照合** (*order-preserving matching, OPPM*) [10] が提案されている．OPPM は，全ての位置の対の大小関係を考慮するので，応用によっては制約が厳しすぎる場合がある [12].

そこで, 照合制約を緩めた照合モデルとして, Park ら [12] は 2019 年に，デカルト木（Cartesian Tree）を用いたパターン照合である**デカルト木照合** (*Cartesian tree matching, CTM*) を提案した．デカルト木は，与えられた数値列から最小値を順に削除することで得られる二分木であり，最小値の構造に着目した数値列の近似表現の一種である．CTM では，テキストの部分文字列とパターンのデカルト木が同じときに照合が成功する．CTM は，連続する部分文字列に対してのみ照合を行うが，実際においては，パターンが連続して出現するとは限らない.

1.3 非連続な部分列照合への拡張

一方で，現実の数値列は測定誤差や不定値を含むため，いくつかのデータを飛ばして，不連続な部分列への照合を許した数値列テキスト照合が望ましい．**順序保存部分列パターン照合** (*order-preserving subsequence matching, OPSM*) 問題 [9] は，OPPM を部分列に拡張した照合問題である．連続列に対する OPPM が線形時間計算可能なのに対して, 非連続な部分列に対する OPSM は NP 困難である [3]．OPSM と密接に関連した問題として, **最長増加部分列** (*longest increasing subsequence, LIS*) 問題がある．これは, 長さ n の数値列 T を受け取り, 最長の非減少な部分列を見つける問題である．Crochemore と Porat [6] は，長さ k の LIS の一つを $O(n \log\log k)$ 時間で計算するアルゴリズムを与えている．これらの非連続な部分列に対する照合問題は，群論における置換群に関する**順列照合** (*permutation matching*) や**回避パターン** (*avoidable pattern*) の研究と密接な関係がある [3], [6] [*2].

1.4 研究目的

本稿で考察する**デカルト木部分列照合** (*Cartesian tree subsequence matching, CTSM*) 問題は，2021 年に加井ら [13] によって提案された数値列の近似照合問題であり，テキストのとびとびの部分列にパターンが照合することを許すようにデカルト木照合（CTM）を拡張したものである．後の節で示すように，CTSM の照合制約は，OPSM の照合制約の自然な拡張（緩和）になっている．一方で，OPSM が NP 困難であるのに対して, 加井ら [13] は CTSM が多項式時間計算可能であることを示した[*3].

本稿では，デカルト木部分列照合（CTSM）問題を考察し，数値列に対する他の近似照合との関係を明らかにし，さらにその時間計算量を改善し，効率良いアルゴリズムを与えることを目的とする.

1.5 主結果

具体的には，以下の結果を得た.

- CTSM 問題と，部分列を対象とした計算問題である OPSM と LIS との関係を調べた.
- 動的計画法に基づいて，CTSM を $O(mn^2)$ 時間と $O(mn)$ 領域で解く効率良いアルゴリズムを与えた（3 節, Algorithm 2）.
- さらに，区間最小値問クエリ構造（Range Minimum Query）を用いて，時間計算量を $O(mn \log n)$ 時間に改善する方法を与えた（4 節, Algorithm 3）.

上記の結果は，既存手法である加井ら [13] のアルゴリズム CTSeqM の時間計算量 $O(mn^3)$ を大幅に改善している．考察として, 2.5 節の観察から, LIS 問題を CTSM 問題に帰着でき，LIS 問題の現在最良の時間計算量は $O(n \log\log n)$ であることから，LIS 問題の計算量の大幅な改善がない限り，$o((m+n)\log\log m)$ 時間（例えば $m+n$ の線形時間）で解くような CTSM 問題の時間計算量の大幅な改善は難しいと思われる.

2. 準備

本節では, 数列に関する基本的な記法と, デカルト木の定義を述べる．さらに, 本論文で考察する問題（*CTSM*）を定義する．非負整数全体を $\mathbb{N} = \{0, 1, 2, \ldots\}$ で表す．任意の非負整数 $0 \leq i \leq j$ に対して, $\mathbb{N}$ の部分集合 $[i] = \{1, \ldots, n\}$ と離散区間 $[i, j] = \{i, i+1, \ldots, j\}$ を定める.

2.1 数列に関する記法

全順序 $\leq$ をもつ**アルファベット** (*alphabet*) を $\Sigma = \{x, y, \ldots\}$ とし，各要素 $x \in \Sigma$ を**文字**（*character*）という．本稿では，Σ を非負整数全体 $\mathbb{N}$ とする.

Σ^* で，Σ 上の長さ 0 以上の文字列の全体を表し，ε で長さ 0 の空文字列を表す． 長さ n の**文字列** (*string*) $S = S[1] \ldots S[n] = S[1, n] \in \Sigma^*$ に対して，$S[i]$ は i 番目の文字を表し，$|S| = n$ で長さを表す．任意の $0 \leq i \leq j \leq n$ に対して，$S[i, j]$ は開始位置が i で終了位置が j の**部分文字列** (*substring*) を表す.

本項の数値列照合では，一般性を失うことなく文字列 S の全ての文字は互いに異なると仮定する [10] [*4]．仮定の下で，S に出現する任意の値 $x \in \Sigma$ に対して，条件 $S[i] = x$ を満たす一意な添字を $\mathtt{idx}(S, x) := i$ で表す．列 S に含まれる値の最小値を $\min S := \min\{S[i] \mid i \in [n]\}$．と書く.

[*2] 実際，OPSM 問題は，置換群における順列照合問題と同一の問題であることが知られている.
[*3] 加井ら [13] によって，時間計算量 $O(mn^3)$ と，空間計算量 $O(mn^2)$ のアルゴリズム CTSeqM が提案された.
[*4] もし S に同じ文字が 2 回以上出現するときは，元の文字 c と位置 i の対 $ci = (c, i)$ を新たな文字として拡張し，仮定を満せる.

長さ n の任意の文字列を $S[1,n]$ とおく．任意の $0 \le m \le n$ に対して，$\mathcal{I}_m^n$ で，$1 \le i_1 < \cdots < i_m \le n$ を満たす昇順の添え字列 $I = (i_1, \ldots, i_m) \in [n]^*$ の全体からなる集合を表す．明らかに $|\mathcal{I}_m^n| = \binom{n}{m}$ である．S の**部分系列** (*subsequence*) とは，S から文字を 0 個以上取り除き，残りの文字を元の順序で連結して得られる文字列をいう．形式的には，添え字列 $I = (i_1, \ldots, i_m) \in \mathcal{I}_m^n$ に対して，$S_I = S[i_1] \ldots S[i_m] \in \Sigma^m$ は，I **に対応する** S **の部分系列**である．

2.2 デカルト木

全順序アルファベット上の文字列 $S[1,n]$ に対する**デカルト木**（*Cartesian tree*）$CT(S)$ とは，頂点数が n の順序付き二分木であり，次のように定義される．簡便のため，任意の添字区間 $[l,r] = \{l, l+1, \ldots, r\}$ に対して，表記 $S[l,r]$ で，部分文字列自身 $S[l,r] \in \Sigma^*$ とその添字区間 $[l,r]$ の対を表すと考える[*5]．部分文字列 $S[l,r]$ の**最小値の最左添字** (*left-most index of the minimum value*) とは，S の最小値 $S[i]$ を与える添字 $i \in [l,r]$ の中で，最左位置の添字 $i_{\min} = \texttt{LMIN-IDX}(S[l,r]) := \arg\min_{i \in [l,r]}(S[i], i)$ をいう．ただし，値と添字の順で比較し，辞書順 $\le$ をとるものとする．

定義 1. 長さ n の文字列 $S = S[1,n]$ の**デカルト木** $CT(S) = CT(S[1,n])$ とは，n 個の頂点 $1, \ldots, n$ をもち，任意の部分文字列 $S[i,j]$ $(1 \le i \le j \le n)$ に対して，次のように帰納的に定義される二分木である．

- $S[i,j]$ が空文字列のとき（$j = i-1$）：$CT(S[i,i-1])$ は空木 *null* である．
- $S[i,j]$ が空文字列でないとき（$i \le j$）：$S[l,r]$ の最小値の最左添字 $i_{\min} := \texttt{LMIN-IDX}(S[l,r])$ に対して，$CT(S[i,j])$ は以下のように再帰的に定義される二分木である．
  - 根は $v = i_{\min}$ である
  - $i_{\min}$ の左部分木は $CT(S[1, i_{\min}-1])$ である
  - $i_{\min}$ の右部分木は $CT(S[i_{\min}+1, n])$ である
  - v の左の子と右の子を，それぞれ，$v.\texttt{L}$ と $v.\texttt{R}$ で表す．

文字列 S のデカルト木を，$CT(S[1,n])$ と定める．定義より，文字列に対してデカルト木は常に一意に定まる．デカルト木の例を図 1 に示す．

デカルト木に関する記法を導入する．文字列 S のデカルト木 $C = CT(S[1,n])$ と任意の $v \in [1,n]$ に対して，C の**頂点** v **を根とする部分木**を $C[v]$ で表す．このとき，$I_v = [l_v, r_v]$ と $P_v := P[l_v, r_v] \in \Sigma^*$ で，それぞれ，部分木 $C[v]$ に対応する**添字区間**と**部分文字列**を表わす．

[*5] 実際には，部分文字列 $Q = S[i,j]$ で，その領域 $dom(Q) := [i,j]$ と写像 $Q : [i,j] \to \Sigma$ の対を考えることに対応する．

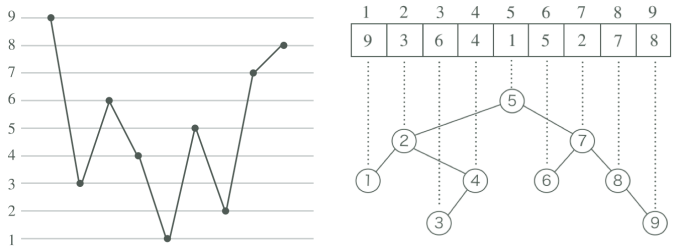

図 1 文字列 $S = (9,3,6,4,1,5,2,7,8)$ の例と，対応するデカルト木 $CT(S)$．$S[1,9]$ の最小値は $S[5]$ であるので，頂点 5 が根，$CT(S[1,4])$ が左部分木，$CT(S[6,9])$ が右部分木となる．

Algorithm 1 CTSM を解くナイーブアルゴリズム

Require: 長さ n のテキスト T，長さ m のパターン P
Ensure: $CT(T_I) = CT(P)$ を満たす添え字列 $I = (i_1, \ldots, i_m)$ $(1 \le i_1 < \cdots < i_m \le n)$ が存在するか

```
for I = (i_1, ..., i_m) ∈ I_{n,m} do
    if CT(T_I) = CT(P) then
        return true
return false
```

2.3 デカルト木部分列照合問題

任意の非負整数 $0 \le m \ge n$ に対して，長さ n の文字列（テキスト）T と長さ m の文字列（パターン）P を考える．$CT(T_I) = CT(P)$ を満たす T の添字列 $I \in \mathcal{I}_m^n$ が存在するとき（すなわち，この条件を満たす T の部分系列 T_I が存在するとき），P が T に**デカルト木照合する**と言い，$P \sqsubseteq T$ と表す．また，木 $C = CT(P)$ が T に照合するといい，$C \sqsubseteq T$ と表すこともある．

本論文で考察する問題を述べる．

定義 2. デカルト木部分列照合問題（*Cartesian tree subsequence matching problem, CTSM*）とは，長さ n の文字列（テキスト）T と長さ m の文字列（パターン）P を入力として受け取り，P が T にデカルト木照合するかどうかを判定する問題である．

問題 *CTSM* を，テキストとパターンの要素がラベルをもつ**ラベル付きデカルト木部分列照合問題** (*Labeled CTSM, LCTSM*) に拡張可能である．このとき，入力として T と P に加えて，テキストとパターンに対するラベル関数 $l_T : [1,n] \to \Sigma$ と $l_P : [1,m] \to \Sigma$ を受け取り，解となる添字列 $I = (i_1, \ldots, i_m)$ は，ラベルを保存する，すなわち，任意の $i \in I$ に対して $l_T(i) = l_P(\texttt{idx}(P,i))$ を満たすものとする．本稿で与える CTSM の解法は，簡単な修正で LCTSM に拡張できるので，詳細はフルペーパーに譲る．

2.4 ナイーブアルゴリズム

CTSM を解くナイーブな方法を Algorithm 1 に示す．

Algorithm 1 は，すべての添え字列 $I = (i_1, \ldots, i_m)$ に対して，文字列 T_I と P から実際にデカルト木を構築する．得られた 2 つのデカルト木が一致するかどうかは，根から深さ優先探索を行い，訪れた頂点の行きがけ順が一致する

かどうかを判定すれば良い．

ここで Algorithm 1 に関する定理を与える．

定理 1. 入力として，長さ n のテキスト T と，長さ m のパターン P が与えられたとき，Algorithm 1 の時間計算量は $O(m2^n)$ である．

証明. Algorithm 1 の 1 行目で，$(1 \leq i_1 < \cdots < i_m \leq n)$ を満たす添え字列長さ m の添え字列 $I = (i_1, \ldots, i_m)$ を列挙している．$(1 \leq i_1 < \cdots < i_m \leq n)$ となる $I = (i_1, \ldots, i_m)$ は ${}_n\mathrm{C}_m$ 通り存在するので，Algorithm 1 の 2 行目は ${}_n\mathrm{C}_m$ 回実行される．二項定理より，${}_n\mathrm{C}_m \leq 2^n$ である．また，長さ m の数列からデカルト木を構成するのにかかる時間計算量 [7] は $O(m)$ であるので，Algorithm 1 の時間計算量は $O(m2^n)$ となる． □

2.5 他の数値列照合問題との関係

以下では，照合問題 $\pi \in \{\mathrm{CTSM}, \mathrm{OPSM}, \ldots\}$ に関して，パターン P がテキスト T に照合するとき，$P \sqsubseteq^{\pi} T$ と書く．

順序保存部分列照合問題（*Order-preserving subsequence matching problem, OPSM*）とは，全順序付きアルファベット Σ 上で，長さ n のテキスト T と長さ m のパターン P を入力として受け取り，条件 $P[j] \leq P[k] \iff T[i_j] \leq T[i_k], \forall j, k \in [1, m]$ を満たす T の添字列 $I = (i_1, \ldots, i_m) \in \mathcal{I}_m^n$ が存在するかどうかを判定する問題である．上記の条件を満たす I が存在するとき，$P \sqsubseteq^{\mathrm{OPSM}} T$ と書く．

定義 3. 長さパラメータ付きの**最長増加部分列問題**とは，全順序アルファベット Σ 上の長さ n の文字列（テキスト）$T[1, n] \in \Sigma^*$ と，非負整数 $0 \leq m \leq n$ を入力として受け取り，条件 $T[i_1] \leq \ldots \leq T[i_m]$ を満たす長さ m の非減少部分列 $I = (i_1, \ldots, i_m) \in \mathcal{I}_m^n$ を見つける問題である．

Crochemore と Porat [6] は，長さパラメータ m に対する LIS 問題は $O(n \log \log k)$ 時間計算可能なことを示している．

はじめに，CTSM と OPSM の関係を考察しよう．

補題 1. $P \sqsubseteq^{\mathrm{OPSM}} T$ ならば $P \sqsubseteq^{\mathrm{CTSM}} T$ が成立する．

証明. もし T の長さ m の部分列 T_I と P において，OPSM の意味で全ての添字対 $j, k \in [m]$ について値の大小関係が一致するならば，$CT(T_I)$ と $CT(P)$ は同一となる．よって，結果が直ちに導かれる． □

CTSM 問題と長さパラメータ付き LIS 問題との関係を考察しよう．簡単な観察から，LIS 問題において T が長さ m の非減少部分列を持つことと，任意の問題 $\pi \in \{\mathrm{CTSM}, \mathrm{OPSM}\}$ において，長さ m の昇順列 $P[1, m] = (1, 2, \ldots, m)$ が T に対して，$P \sqsubseteq^{\pi} T$ が成立することは同値である．よって，CTSM または OPSM を $t(n)$ 時間と $s(n)$ 領域で解くアルゴリズムがあれば，長さパラメータ付きの LIS 問題は $t = O(t(n) + m)$ 時間と $s = O(s(n) + m)$ 領域で解けることがわかる．

3. 提案手法：デカルト木部分列照合

本節では，問題 *CTSM* に対して動的計画法を適用することで，Algorithm 1 より高速なアルゴリズムを与える．

3.1 基本アイディア

アルゴリズムの基本アイディアを述べる．

まず，パターン P のデカルト木 C を構成する．パターン P から構成されるデカルト木を C と書く．

CTSM を効率よく解くために，**極小出現区間**を定義する．

定義 4. パターン P と添え字 $i = \{1, \ldots, n\}$ に対して，次の条件を満たす $I = (i_1, \ldots, i_k)$ が存在するとき，区間 $X = [l, r]$ を P の i に対する**出現区間**という．

- $l = i_1 < \ldots < i_k = r$
- $CT(T_I) = CT(P)$
- $T[i] = \min(T_I)$

定義 5. パターン P と添え字 $i = \{1, \ldots, n\}$ に対して，P の i に対する**極小出現区間**とは，次を満たす区間 $X = [l, r]$ である．

- $[l, r]$ は P の i に対する出現区間である
- $[l', r'] \subset [l, r]$ を満たすような，P の i に対する出現区間 $[l', r']$ が存在しない

出現区間と，極小出現区間の例を図 3.1 に示す．

ここで，極小出現区間に関する重要な補題を与える．

補題 2. P と $i = \{1, \ldots, n\}$ に対して，P の i に対する極小区間は高々 1 つである．

証明. P と $i = \{1, \ldots, n\}$ に対して，P の i に対する極小出現区間が 2 つ存在すると仮定する．このような 2 つの区間を $X = [l, r]$ と $Y = [l', r']$（$X \neq Y$）と置く．一般性を失わずに，$l \leq l'$ とする．

$r' < r$ のとき，$[l', r'] \subset [l, r]$ となり，区間 $[l, r]$ が極小出現区間であることに矛盾する．よって，$r < r'$ が成り立つ．

ここで，定義 4 より，$CT(T_I) = CT(P)$ かつ $T[i] = \min(T_I)$ を満たすような $I = (l, \ldots, i, \ldots, r)$ が存在する．同様に，$CT(T_{I'}) = CT(P)$ かつ $T[i] = \min(T_{I'})$ を満たすような $I' = (l', \ldots, i, \ldots, r')$ が存在する．$CT(T_I) = CT(T_{I'}) = P$ より，$CT(T_{(i+1,\ldots,r)}) = CT(T_{(i+1,\ldots,r')})$ が成り立つので，

$$\begin{aligned} CT(T_{I'}) &= CT(T_{(l',\ldots,i,i+1,\ldots,r')}) \\ &= CT(T_{(l',\ldots,i,i+1,\ldots,r)}) \\ &= CT(P) \end{aligned}$$

となる．よって，区間 $[l', r]$ は，P の i に対する出現区間となり，$[l', r] \subset [l', r']$ であることから $[l', r']$ が極小出現区間であることに矛盾する．

以上により，P と $i = \{1, \ldots, n\}$ に対する極小出現区間

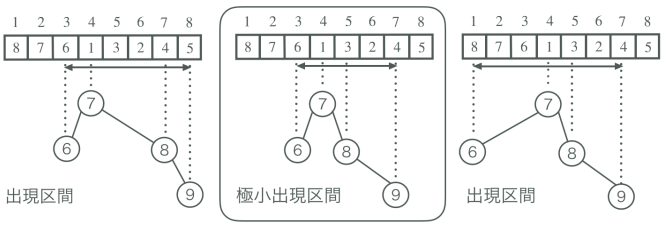

図 2 テキスト $T = (8, 7, 6, 1, 3, 2, 4, 5)$ と図のデカルト木 $CT(P)[7]$ に対して，$CT(P)[7]$ の 4 に対する出現区間の例として，左から $[3, 8]$，$[3, 7]$，$[1, 7]$ があげられる．区間 $[3, 7]$ は $CT(P)[7]$ の 4 に対する極小出現区間である．

は高々 1 つであることが示された． □

便宜的に，P の i に対する極小出現区間 $X = [l, r]$ が存在しないとき，$l = -\infty, r = \infty$ と表すことにする．

系 1. *P の i に対する極小出現区間を $[l, r]$ とし，$P[..\mathrm{ind}(\min(P))]$ の i に対する極小出現区間を $[l', r']$，$P[\mathrm{ind}(\min(P))..]$ の i に対する極小出現区間を $[l'', r'']$ とする．このとき，次が成り立つ．*

- $l = l'$
- $r = r''$
- $r' = l'' = i$

3.2 動的計画法の適用

動的計画法を用いて *CTSM* を解く方法を Algorithm 2 に示す．

DP テーブルの定義を述べる．

定義 6. $v \in \{1, \ldots, m\}$ と $i \in \{1, \ldots, n\}$ に対して， DP テーブルを次のように定義する．

$$L(v, i) := P_v[..\mathrm{ind}(\min(P_v))] \text{ の } i \text{ に対する極小出現区間の左端} \tag{1}$$

$$R(v, i) := P_v[\mathrm{ind}(\min(P_v))..] \text{ の } i \text{ に対する極小出現区間の右端} \tag{2}$$

P_v の i に対する極小出現区間が存在するとき，系 1 より区間 $[L(v, i), R(v, i)]$ は，P_v の i に対する極小出現区間となる．

図 3 に，系 1 や DP テーブルの定義 1 と定義 2 の補足説明を示す．

アルゴリズムは, 動的計画法によって DP テーブル $L(v, i)$ と $R(v, i)$ を計算することで，すべての $v \in \{1, \ldots, m\}$ と $i \in \{1, \ldots, n\}$ に対する $L(v, i)$ と $R(v, i)$ を得ることができる．

ここで，DP テーブルに関する系を述べる．

系 2. 式 1,2 で定義された DP テーブルを計算し終えたとする．$P \subseteq_R T$ となる必要十分条件は，$L(r, i) \neq -\infty$ かつ $R(r, i) \neq \infty$ を満たす $i \in \{1, \ldots, n\}$ が存在することである．

系 2 より，動的計画法により L と R の計算を終えたとき，直ちに *CTSM* を解くことができる．

Algorithm 2 動的計画法を用いて CTSM を解くアルゴリズム

Require: 長さ n のテキスト T，パターン P のデカルト木 $C = (V, E, r)$
Ensure: $CT(T_I) = C$ を満たす添え字列 $I = (i_1, \ldots, i_m)$（$1 \leq i_1 < \cdots < i_m \leq n$）が存在するか

```
function COMPUTE_LEFT_MAX(v, L, R)
    for i ∈ {1, ..., n} do
        for j ∈ {1, ..., i − 1} do
            if T[i] < T[j] ∧ R(v.L, j) < i then
                L(v, i) ← max(L(v, i), R(v.L, j))          ▷ 式 3
function COMPUTE_RIGHT_MIN(v, L, R)
    for i ∈ {1, ..., n} do
        for j ∈ {i + 1, ..., n} do
            if T[i] < T[j] ∧ i < L(v.R, j) < i then
                R(v, i) ← min(R(v, i), L(v.R, j))          ▷ 式 4
function UPDATE_TABLE(v)                  ▷ Input: v ∈ {1, ..., m}
    if v.L = null then
        for i ∈ {1, ..., n} do L(v, i) ← i                 ▷ 式 3
    else
        call COMPUTE_LEFT_MAX(v, L, R)
    if v.R = null then
        for i ∈ {1, ..., n} do R(v, i) ← i                 ▷ 式 4
    else
        call COMPUTE_RIGHT_MIN(v, L, R)
//アルゴリズム本体
for i ∈ {1, ..., n} do
    for v ∈ {1, ..., m} do
        L(v, i) ← −∞
        R(v, i) ← ∞
for each v ∈ {1, ..., m} in CT(P) in the post order do
    call UPDATE_TABLE(r)
for i ∈ {1, ..., n} do
    if L(r, i) ≠ ∞, R(r, i) ≠ −∞ then
        return true
return false
```

動的計画法を用いた DP テーブルの更新方法について述べる．まず，すべての $v \in \{1, \ldots, m\}$ と $i \in \{1, \ldots, n\}$ に対して，$L(v, i) \leftarrow -\infty, R(v, i) \leftarrow \infty$ と初期化する．次に，以下の更新式に従って，v に対して C の葉からボトムアップに DP テーブルを計算する．

$$L(v, i) := \begin{cases} i, & \text{if } v.\mathrm{L} = \text{null}, \\ \max\limits_{\substack{1 \leq j \leq n \\ T[i] < T[j]}} \{L(v.\mathrm{L}, j) \mid R(v.\mathrm{L}, j) < i\}, & \text{otherwise.} \end{cases} \tag{3}$$

$$R(v, i) := \begin{cases} i, & \text{if } v.\mathrm{R} = \text{null}, \\ \min\limits_{\substack{1 \leq j \leq n \\ T[i] < T[j]}} \{R(v.\mathrm{R}, j) \mid i < L(v.\mathrm{R}, j)\}, & \text{otherwise.} \end{cases} \tag{4}$$

ここで，アルゴリズムの正当性に関する補題を与える．

補題 3. すべての $i = \{1, \ldots, n\}$ に対して，$L(v.\mathrm{L}, i)$ と $R(v.\mathrm{L}, i)$ の計算がすべて完了しているものとする．

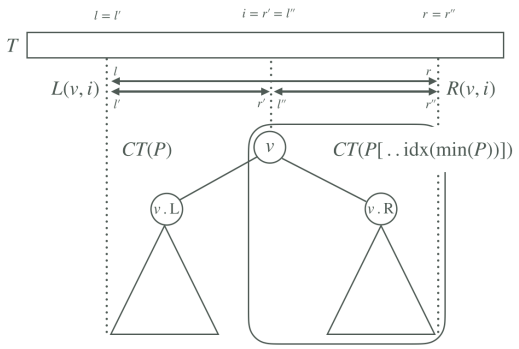

図 3 P の i に対する極小出現区間を $[l, r]$ とし, P[..idx(min(P))] の i に対する極小出現区間を $[l', r']$, P[idx(min(P))..] の i に対する極小出現区間を $[l'', r'']$ とする. $CT(P$[idx(min(P))..]) は, $CT(P)$ から $CT(P)[v.\mathrm{L}]$ を取り除いた木を表す. DP テーブル $L(v, i), R(v, i)$ は, 頂点 v と $T[i]$ を固定したときの極小出現区間を表す.

このとき, 式 3 と式 4 を用いて, すべての $i \in \{1, \ldots, n\}$ に対して $L(v, i)$ と $R(v, i)$ を計算することができる.

証明. 式 3 の正当性を示す.

$v.\mathrm{L} = \mathrm{null}$ のとき, P_v[..ind(min(P_v))] $= (P_v[1])$ となるので, 任意の $i = \{1, \ldots, n\}$ に対して, P_v[..ind(min(P_v))] の i に対する極小出現区間は $[i, i]$ となる. よって, $L(v, i) = i$ となる.

$v.\mathrm{L} \neq \mathrm{null}$ のとき,

$L(v, i) = \max_{1 \leq j \leq n, T[i] < T[j]} \{L(v.\mathrm{L}, j) \mid R(v.\mathrm{L}, j) < i\}$ が成り立つことを示すには, 次の条件満たす $j \in \{1, \ldots, n\}$ が存在することを示せば良い.

- $T[i] < T[j]$
- $L(v, i) = L(v.\mathrm{L}, j)$
- $R(v.\mathrm{L}, j) < i$

P_v[..ind(min(P_v))] の i に対する極小出現区間を $X = [l, i]$ としたとき, $v.\mathrm{L} \neq \mathrm{null}$ なので, $T[i] < T[j]$ かつ $\mathrm{ind}(T_I, T[j]) = \mathrm{ind}(P_{v.\mathrm{L}}, \mathrm{min}(P_{v.\mathrm{L}}))$ を満たすような $I = (l, \ldots, j, \ldots, r, i)$ が存在する.

また, デカルト木の定義より, $CT(T_{(l,\ldots,j,\ldots,r)}) = CT(P_{v.\mathrm{L}})$ が成り立つので, 区間 $[l, r]$ は $P_{v.\mathrm{L}}$ の j に対する出現区間である.

ここで, $P_{v.\mathrm{L}}$ の j に対する極小出現区間を $[l', r']$ と置くと, $l \leq l'$ と $r' \leq r$ が成り立つ. $l < l'$ を仮定すると, 添え字列 $I' = (l', \ldots, j, \ldots, r', i)$ に対して, $CT(T_{I'}) = CT(P_v$[..ind(min(P_v))] が成立するので, P_v[..ind(min(P_v)) の i に対する極小出現区間が $[l, i]$ であることに矛盾する. よって, $l' \leq l$ が成立する. 以上により, $l = l'$ かつ $r' \leq r < i$ が成り立つ.

l, l', r' は, それぞれ $L(v, i)$, $L(v.\mathrm{L}, j)$, $R(v.\mathrm{L}, j)$ に対応するので, $T[i] < T[j]$ かつ $L(v, i) = L(v.\mathrm{L}, j)$ かつ $R(v.\mathrm{L}, j) < i$ を満たすような $j \in \{1, \ldots, n\}$ が存在することが示された.

式 4 についても同様の議論ができる.

以上により, 補題 3 が示された. □

3.3 計算量解析

Algorithm 2 の計算量に関する定理を与える.

定理 2. 入力として, 長さ n のテキスト T と長さ m のパターン P が与えられたとき, Algorithm 2 の時間計算量は $O(mn^2)$ であり, 空間計算量は $O(mn)$ である.

証明. 時間計算量について議論する. Algorithm 2 の 21 行目から 24 行目のループ回数は mn 回であり, L と R の初期化にかかる時間計算量は $O(mn)$ である. Algorithm 2 の 25 行目では, パターン P のデカルト木 C の帰りがけ順で, 葉からボトムアップにテーブル L とテーブル R を計算する. C の頂点数は m 個であるので, 関数 **UPDATE_TABLE** は丁度 m 回呼び出される.

Algorithm 2 の 2 行目から 5 行目で, n に関する二重ループを実行するので, 関数 **COMPUTE_LEFT_MAX** の時間計算量は $O(n^2)$ となる. 関数 **COMPUTE_RIGHT_MIN** についても同様に, 時間計算量は $O(n^2)$ となる.

関数 **COMPUTE_LEFT_MAX** と関数 **COMPUTE_RIGHT_MIN** は, それぞれについて, 高々 m 回実行されるので, Algorithm 2 の時間計算量は, $O(mn^2)$ となる.

空間計算量について議論する. L と R は共に, 長さ mn の作業領域を要する. 頂点数 m のデカルト木を深さ優先探索で巡回するのに必要な作業領域は $O(m)$ であるので, 空間計算量は $O(mn + m) = O(mn)$ となる.

以上により, Algorithm 2 の時間計算量は $O(mn^2)$ であり, 空間計算量は $O(mn)$ であることが示された. □

Algorithm 2 の簡単な拡張で, ラベル付きデカルト木部分列照合問題を解くことができる. 具体的には, 式 3 と式 4 の max の下付き文字の条件に, $l_T(j) = l_P(v)$ を追加すると良い.

ここで, 次のような系を得る.

系 3. 入力として長さ n の文字列 T と, 長さ m の文字列 P, テキストに対するラベル $l_T : \{1, \ldots, n\} \to \Sigma$, パターンに対するラベル $l_P : \{1, \ldots, m\} \to \Sigma$ が与えられたとき, 時間計算量が $O(mn^2)$ で, 空間計算量が $O(mn)$ の問題 *CTSM* を解くアルゴリズムが存在する.

4. 区間最小値クエリ問題を適用した時間計算量の改善

本節では, Algorithm 2 を改良し, 時間計算量を改善する方法を述べる. DP テーブルの更新順序を工夫し, 区間最小値クエリ問題を効率よく処理するデータ構造を用いる

Algorithm 3 区間最小値クエリ構造を用いて $L(v,i)$ を計算するアルゴリズム．**COMPUTE_RIGHT_MIN** についても，同様に構成できる．

Require: $v \in \{1,\dots,m\}$
Ensure: すべての $i = \{1,\dots,n\}$ に対して，$L(v,i)$ を計算する
1: **function** COMPUTE_LEFT_MAX(v, L, R)
2: **for** $i \in \{1,\dots,n\}$ **do**
3: $\mathrm{max\_left}[r] \leftarrow -\infty$
4: 数列 max_left を用いて，区間最大値クエリ構造を初期化する
5: **for** $i \in \{1,\dots,n\}$ for ascending order of T **do**
6: $\mathtt{L}(v,i) \leftarrow \mathtt{range\_max}(1,n)$
7: $\mathtt{update\_max}(R(v.\mathrm{L},i), L(v.\mathrm{L},i))$

ことで，時間計算量を $O(mn\log n)$ に改善する．

4.1 区間最小値クエリ問題

区間最小値クエリ問題（*Range Minima Problem*）とは，与えられた長さ n の配列に対して，以下のクエリに答える問題である．

- 最小値更新クエリ $\mathtt{update\_min}(i,x) : i \in \{1,\dots,n\}$ と $x \in \mathbb{N}$ を受け取り，$T[i] \leftarrow \min(T[i],x)$ を実行する．
- 区間最小値クエリ $\mathtt{range\_min}(l,r) : l,r \in \{1,\dots,n\}$ を受け取り，$\min\{T_l,\dots,T_r\}$ を出力する．

同様に，最大値更新クエリ $\mathtt{update\_max}(i,x)$ と，区間最大値クエリ $\mathtt{range\_max}(l,r)$ も定義する．

区間最小値クエリ問題を効率よく処理するデータ構造が知られている [4]．$O(n)$ 前処理時間と領域を用いて，各クエリに対して $O(\log n)$ 時間を達成するデータ構造である．

4.2 基本アイディア

Algorithm 3 に，すべての $i \in \{1,\dots,n\}$ に対して，$L(v,i)$ を計算する関数 **COMPUTE_LEFT_MAX** を示す．この関数を用いて，Algorithm 2 の同名の関数と置き換えることができる．関数 **COMPUTE_RIGHT_MIN** についても，同様に構成できる．

アルゴリズムの基本アイディアを述べる．Algorithm 2 と同様に，$CT(P)$ の葉からボトムアップに DP テーブル $L(v,i)$ と $R(v,i)$ を計算する．

Algorithm 2 では，$L(v,i)$ を式 3 に従って計算するのに $O(n)$ 時間かかる．これを改善するために，データ構造を用いて高速化する．

$T[i] < T[j]$ を満たすような $j \in \{1,\dots,n\}$ に対して，区間 $[L(v.\mathrm{L}), R(v.\mathrm{L},j)]$ からなる集合を $\mathcal{S}(v,i)$ とする．つまり，

$$\mathcal{S}(v,i) = \bigcup_{\substack{1\le j<i \\ T[i]<T[j]}} \{[L(v.\mathrm{L},i), R(v.\mathrm{L},i)]\}$$

とする．

このとき，式 3 より，$L(v,i)$ は，$i < r$ を満たす区間 $[l,r] \in \mathcal{S}(v,i)$ のうち，l の最大値となる（図 4）．

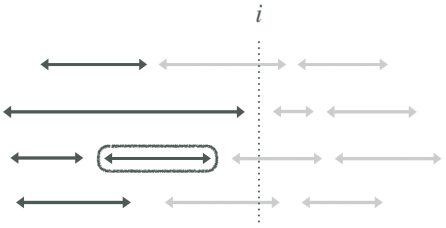

図 4 $r < i$ を満たす区間 $[l,r]$ のうち，l が最大なものが $L(v,i)$ となる．

$i \in \{1,\dots,n\}$ を $T[i]$ が降順となるような順序で，以下の手順を実行することで，すべての $i \in \{1,\dots,n\}$ に対して $L(v,i)$ を計算することができる．

(1) 集合 $\mathcal{S}$ から $r < i$ を満たす最大の l を見つける．
(2) $\mathcal{S}$ に区間 $[L(v.\mathrm{L},i), R(v.\mathrm{L},i)]$ を挿入する．

以上の 2 つの操作は，区間最小値クエリ構造を用いて効率よく処理することができる．具体的には，区間最小値クエリ構造に長さ n の配列 max_left を格納し，次のようにデータ構造へのクエリ処理と対応づける．$\mathrm{max\_left}[r]$ は，終点が r の区間の中で l の最大値を表す．

(1) $L(v,i) \leftarrow \mathtt{range\_max}(1,i)$
(2) $\mathtt{update\_max}(R(v.\mathrm{L},i), L(v.\mathrm{L},i))$

各 $i = \{1,\dots,n\}$ に対して range_max と update_max をちょうど一度ずつ実行するので，$L(v,i)$ を計算するのにかかる時間は $O(\log n)$ となる．

4.3 計算量解析

Algorithm 3 の計算量に関する定理を与える．

補題 4. 入力として，$v \in \{1,\dots,m\}$ が与えられたとき，Algorithm 3 の時間計算量は $O(n\log n)$ である．

証明. Algorithm 3 の 2 行目から 3 行目における，配列 max_left の初期化にかかる時間計算量は $O(n)$ である．Algorithm 3 の 4 行目における，区間最大値クエリ構造の初期化にかかる時間計算量は $O(n)$ である．Algorithm 3 の 5 行目では，あらかじめ $(T[i],i)$ の組を昇順でソートしておくことで，$i = \{1,\dots,n\}$ に対して，$T[i]$ が昇順となるような順序で実行できる．各 $i = \{1,\dots,n\}$ に対して，range_max と update_max をちょうど 1 回実行しているので，Algorithm 3 の 5 行目から 7 行目にかかる時間計算量は $O(n\log n)$ となる．

以上により，Algorithm 3 の時間計算量は $O(n\log n)$ となる． □

DP テーブル R についても同様の議論が可能である．したがって，本論文の主結果である，次の定理を与える．

定理 3. 入力として，長さ n のテキスト T と長さ m のパターン P が与えられたとき，問題 *CTSM* を解く提案アルゴリズムの時間計算量は $O(mn\log n)$ であり，空間計算量は $O(mn)$ である．

証明. 補題 4 より明らか. □

系 3 と同様に，次のような系を得る．

系 4. 入力として長さ n の文字列（テキスト）T と，長さ m の文字列（パターン）P，テキストに対するラベル $l_T : \{1,\ldots,n\} \to \Sigma$，パターンに対するラベル $l_P : \{1,\ldots,m\} \to \Sigma$ が与えられたとき，時間計算量が $O(mn \log n)$ で，空間計算量が $O(mn)$ の問題 $CTSM$ を解くアルゴリズムが存在する．

5. 実験

本節では，提案手法の実験結果を示し，その考察を行う．

デカルト木部分列照合問題（$CTSM$）を解くための 3 つのアルゴリズムに対して，入力のテキスト T とパターン P のサイズを変化させて実行し，速度を計測した．ナイーブな手法である Algorithm 1 と，提案手法である 3 を比較した．

5.1 環境および設定

本研究の実験では，以下の計算機（CPU Intel(R) Core(TM) i5-7360U CPU @ 2.30GHz，8 GiB メモリ，macOS Mojave 10.14.6）を用いた．すべてのプログラムは Apple LLVM version 10.0.1 (clang-1001.0.46.4) でオプション-O3 を用いて，コンパイルした..

本研究の実験において，デカルト木部分列照合問題（$CTSM$）の入力である長さ n のテキスト T と長さ m のパターン P には，文字列の長さ $\ell \in \{n, m\}$ に対して，$\{1,\ldots,\ell\}$ のすべての順列を一様ランダムに生成して得られた人工データを用いた．

実験には，C++で実装した次のプログラムを用いた．以下では，入力サイズを n と m で表す．

- ナイーブ: 2.4 の Algorithm 1 に示した $O(m2^n)$ 時間の素朴なアルゴリズム．
- 既存手法: 先行研究 [13] の $O(mn^3)$ 時間の CTSeqM アルゴリズム．
- 提案手法: 3 節の Algorithm 2 に，4 の Algorithm 3 を組み合わせて得られた $O(mn \log n)$ 時間のアルゴリズム．

これらの 3 つのアルゴリズムに対して，テキスト T のサイズ n を変化させて，計算が終了するまでの速度を計測した．パターン長 m は，$[1, n]$ の範囲でランダムに生成し，実行時間は 10 ケースの平均値を採用した．

5.2 実験結果

実験の結果を表 1 に示す．NA は，実行時間が 1 時間を超えたものについて，計測を打ち切ったことを示す．

この結果より，提案アルゴリズムの方が高速に動作することがわかる．

表 1 デカルト木部分列照合問題（$CTSM$）を解くのにかかった時間（秒）

テキスト長 n	ナイーブ	既存手法	提案手法
10	0.0318	0.04387	0.0052
100	NA	0.3046	0.0135
300	NA	10.3467	0.0639
500	NA	258.1598	0.1194
1000	NA	NA	0.4255

6. 終わりに

本論文では，デカルト木照合問題を部分列に拡張したデカルト木部分列照合問題を定義した．さらに，動的計画法を用いて効率よく解くアルゴリズムを与えた．

今後の課題として，本問題を近似文字列照合問題への拡張や，値の変動を許した問題への拡張を考え，その問題を解くことが考えられる．

参考文献

[1] A. Amir, Y. Aumann, G. M. Landau, M. Lewenstein, and N. Lewenstein. Pattern matching with swaps. *Journal of algorithms*, 37(2):247–266, 2000.

[2] A. Amir, R. Cole, R. Hariharan, M. Lewenstein, and E. Porat. Overlap matching. *Information and Computation*, 181(1):57–74, 2003.

[3] P. Bose, J. F. Buss, and A. Lubiw. Pattern matching for permutations. *IPL*, 65(5):277–283, 1998.

[4] G. S. Brodal, P. Davoodi, and S. S. Rao. Path minima queries in dynamic weighted trees. In *Workshop on Algorithms and Data Structures*, pages 290–301, 2011.

[5] P. Burcsi, F. Cicalese, G. Fici, and Z. Lipták. Algorithms for jumbled pattern matching in strings. *Int'l J. of Foundations of Comput. Sci.*, 23(02):357–374, 2012.

[6] M. Crochemore and E. Porat. Fast computation of a longest increasing subsequence and application. *Information and computation*, 208(9):1054–1059, 2010.

[7] J. Fischer and V. Heun. Theoretical and practical improvements on the rmq-problem, with applications to lca and lce. In *CPM 2006*, pages 36–48. Springer, 2006.

[8] J. Garay and A. Kiayias. Sok: A consensus taxonomy in the blockchain era. In *Cryptographers' Track at the RSA Conference*, pages 284–318. Springer, 2020.

[9] P. Gawrychowski and P. Uznański. Order-preserving pattern matching with k mismatches. *Theoretical Computer Science*, 638:136–144, 2016.

[10] J. Kim, P. Eades, R. Fleischer, S.-H. Hong, C. S. Iliopoulos, K. Park, S. J. Puglisi, and T. Tokuyama. Order-preserving matching. *Theoretical Computer Science*, 525:68–79, 2014.

[11] J. Mendivelso, S. V. Thankachan, and Y. J. Pinzón. A brief history of parameterized matching problems. *Discret. Appl. Math.*, 274:103–115, 2020.

[12] S. G. Park, A. Amir, G. M. Landau, and K. Park. Cartesian tree matching and indexing. *CPM 2019*, 2019.

[13] 加井丈志, 光吉健汰, 古谷勇, 有村博紀. デカルト木照合の部分系列への拡張. 研究報告アルゴリズム (AL), 情報処理学会, 2021(12):1–7, 2021.