Vol.2020-AL-176 No.6
2020/1/30

パラメタ化文字列に対する接尾辞トレイ

藤里 法輝[1]　中島 祐人[1]　稲永 俊介[1,2]　坂内 英夫[1]　竹田 正幸[1]

概要：パターン照合の一種に，文字列の構造に着目した「パラメタ化文字列照合（Parameterized Pattern Matching; PPM）問題」がある．以下に PPM 問題を定義する．互いに素なアルファベット Π, Σ に対して，$\Sigma \cup \Pi$ 上の文字列を p-文字列と呼ぶ．Π 上のある全単射 f が存在し，p-文字列 x の Π の要素であるすべての文字を f によって置換して得られる p-文字列が p-文字列 y と一致するとき，2 つの p-文字列 x, y は p-マッチするという．PPM 問題とは，入力としてパターン p-文字列 P とテキスト p-文字列 T が与えられたとき，P と p-マッチする T の部分文字列の開始位置をすべて出力する問題である．本研究では PPM 問題のための索引構造であるパラメタ化接尾辞トレイ (Parameterized Suffix Tray; PSTray) を新たに提案する．PSTray を用いることで $O(m + \log(|\Sigma| + |\Pi|) + occ)$ 時間の PPM に対するクエリ時間を達成する．ここで，m はパターン長，occ は出力サイズである．この結果は，PPM 問題に対する既存のどの索引構造のクエリよりも理論的に高速である．また，T のパラメタ化接尾辞木が与えられたとき，$O(n)$ 時間，領域で PSTray を構築するアルゴリズムを提案する．ただし，n は入力テキスト長である．

キーワード：文字列アルゴリズム，パラメタ化パターン照合，索引データ構造

1. 導入

パターン照合問題における索引構造として，接尾辞木 [13]，接尾辞配列 [11] やポジションヒープ [6], [10] などが知られている．例えば接尾辞木を用いた際のクエリ時間は $O(m \log(|\Sigma|) + occ)$，接尾辞配列と最長共通接頭辞配列 [11] を用いた際のクエリ時間は $O(m + \log n + occ)$ である．ここで，m, n, occ, Σ はそれぞれ，パターン長，テキスト長，パターン出現回数，アルファベットである．また Cole らは，接尾辞木，接尾辞配列，最長共通接頭辞配列の 3 つ組を用いた索引構造である接尾辞トレイ [3] を提案した．接尾辞トレイのクエリ時間は $O(m + \log(|\Sigma|) + occ)$ であり，上記のいずれの結果に比べて理論的に高速なクエリを達成している．

パターン照合問題には様々な亜種が提案および研究されており，その一つである**パラメタ化パターン照合** (Parameterized Pattern Matching; PPM) は文字列の構造に焦点を当てたパターン照合であり，Baker [1] によって導入された．Σ と Π を互いに素なアルファベットとする．$(\Sigma \cup \Pi)^*$ の要素をパラメタ化文字列，もしくは簡単のために p-文字列とよぶ．Π 上のある全単射 f が存在し， p-文字列 x の Π の要素であるすべての文字を f によって置換して得られる p-文字列が p-文字列 y と一致するとき，2 つの p-文字列 x, y は p-**マッチ**するという．パラメタ化パターン照合問題とは，テキスト p-文字列 T とパターン p-文字列 P が与えられたとき，P と p-マッチする T の部分文字列の開始位置をすべて出力することである．パラメタ化パターン照合はソフトウェアのメンテナンス [1] や剽窃の検出などに応用されうる．

パラメタ化パターン照合問題に対しても，通常の文字列のパターン照合問題と同様に，効率的に解くための索引構造が提案されている．例えば，パラメタ化ポジションヒープ [5], [7]，パラメタ化接尾辞配列 [4], [8]，パラメタ化接尾辞木 [2] などが知られている．例えばパラメタ化接尾辞木を用いたときのクエリ時間は $O(m \log(|\Sigma| + |\Pi|) + occ)$ であり，パラメタ化接尾辞配列とパラメタ化最長共通接頭辞配列 [4], [8] を用いたときのクエリ時間は $O(m + \log n + occ)$ である．ただし，m, n, occ, Σ, Π はそれぞれ，パターン長，テキスト長，パターン出現回数，定数文字集合，変数文字集合である．

本研究ではパラメタ化接尾辞木，パラメタ化接尾辞配列，パラメタ化最長共通接頭辞配列の 3 つ組を用いた索引構造である**パラメタ化接尾辞トレイ**を新たに提案する．この索引構造はパラメタ化接尾辞木から $O(n)$ 時間，領域で構築できることを示す．また，$O(m + \log(|\Sigma| + |\Pi|) + occ)$ で

[1] 九州大学
Kyushu University
[2] JST さきがけ
JST PRESTO

クエリ応答できることも示す．このクエリ時間は既存のどのパラメタ化パターン照合問題に対する索引構造のクエリ時間よりも理論的に高速な結果である．

2. 準備

本章では，本稿で使用する用語の説明などを行う．文字列 t の長さを $|t|$ と表記する．文字列 $t = xyz$ に対し，x, y, z をそれぞれ t の**接頭辞**，**部分文字列**，**接尾辞**と呼ぶ．$0 \leq i, j \leq n$ に対し，$t[i:], t[:i], t[i:j]$ はそれぞれ位置 i から始まる t の接尾辞と位置 i で終わる t の接頭辞と位置 i から始まり位置 j で終わる t の部分文字列である．ただし，$i > j$ のとき $t[i:j] = \epsilon$ とする．文字列 t に対し，逆文字列 t^R とは $1 \leq i \leq |t|$ を満たす任意の i について $t[i] = t^R[n+1-i]$ が成り立つ文字列のことである．Σ, Π を互いに素な文字集合とする．$\Sigma \cup \Pi$ 上の文字列を p-文字列と呼ぶ．

3. パラメタ化パターン照合

本章では，パラメタ化パターン照合問題について説明する．p-マッチとは以下のように定義される．

定義 1 ([1])．同じ長さの p-文字列 w_1 と w_2 が p-マッチするとは，ある全単射 $f : \Sigma \cup \Pi \to \Sigma \cup \Pi$ が存在し，以下の条件を満たすことである;

$$\forall c \in \Sigma, c = f(c), w_2[i] = f(w_1[i])\ (\forall i, 1 \leq i \leq |w_1|).$$

つまり，f は Σ に含まれる文字については恒等写像であるが，Π に含まれる文字についてはその限りではない．p-文字列 x, y が p-マッチするとき，$x \approx y$ と表記する．例えば，$\Sigma = \{A, B\}, \Pi = \{x, y, z\}$ とすると，p-文字列 $X = xyzAxxxByzz$ と $Y = zxyAzzzBxyy$ について $X \approx Y$ である．なぜならば，X 中の x を z に，y を x に，z を y に置き変えることで 2 つの文字列が合致するためである．

次に，パラメタ化パターン照合問題を以下のように定義する．

定義 2 ([1])．パラメタ化パターン照合（Parameterized Pattern Matching; PPM）問題とは，テキスト p-文字列 T とパターン p-文字列 P が与えられたとき，P と p-マッチする T の部分文字列の開始位置をすべて出力することである．

例えば，$\Sigma = \{A\}, \Pi = \{x, y, z\}$ について，テキスト $T = xyzAxxxAyyzAzx$ とパターン $P = yAzz$ を入力とすると，出力するテキスト中の位置の集合は $\{3, 7\}$ である．

p-文字列の p-マッチは，直前符号 $prev$ と最小符号 spe により，確認できることが知られている．

定義 3 ([1])．p-文字列 w の直前符号 $prev(w)$ は以下のように定義される $\Sigma \cup \{0, \ldots, |w|-1\}$ 上の文字列である．ただし，$1 \leq i \leq |w|$ とする．

$$prev(w)[i] = \begin{cases} w[i] & (w[i] \in \Sigma) \\ 0 & (w[i] \in \Pi \text{ and } \\ & \quad w[j] \neq w[i] \text{ for any } 1 \leq j \leq i) \\ i - j & \text{otherwise} \end{cases}$$

定義 4. p-文字列 w の最小符号化 $spe(w)$ は w と p-マッチする辞書式順序最小の p-文字列である．

任意の p-文字列 w_1 と w_2 に対して，$prev(w_1) = prev(w_2)$ が成り立つことと，w_1 と w_2 が p-マッチすることは同値である．また，$spe(w_1) = spe(w_2)$ が成り立つことと，w_1 と w_2 が p-マッチすることは同値である．例えば，$\Sigma = \{A, B\}, \Pi = \{x, y, z\}$ とし，$X = yxzAyyyBxzz$，$Y = zxyAzzzBxyy$ とすると，$prev(X) = 000A411B771 = prev(Y)$ でありかつ，$spe(X) = xyzAxxxByzz = spe(Y)$．また，直前符号化と最小符号化は共に文字列の長さに線形時間で計算できる．

4. パラメタ化接尾辞トレイ

パラメタ化接尾辞トレイ（Parameterised Suffix Tray; PSTr）をこの章で定義する．PSTr を定義するため，パラメタ化接尾辞木（Parameterised Suffix Tree; PST），パラメタ化接尾辞配列（Parameterised Suffix Array; PSA），パラメタ化最長共通接頭辞配列（Parameterised Longest Common Prefix array; PLCP）をそれぞれ定義する．

PST を以下に定義する．

定義 5 ([2])．$\$ \in \Sigma$ を文字列の末尾のみに現れる特別な文字とする．p-文字列 t に対するパラメタ化接尾辞木 $PST(t)$ とは，$t\$$ の接尾辞を直前符号化した文字列集合を表すコンパクトトライである．

PST の例を図 1 に示す．

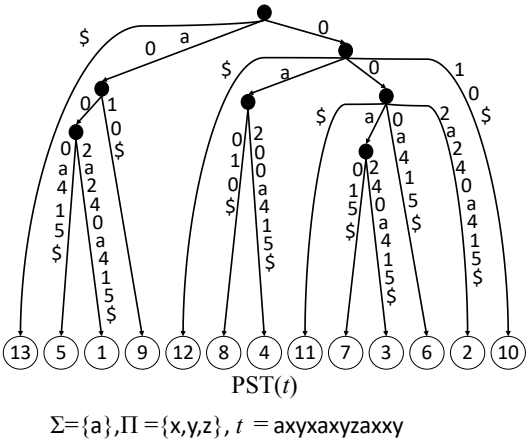

図 1 $\Sigma = \{a\}, \Pi = \{x, y, z\}, t = axyaxyzaxxy$ としたときのパラメタ化接尾辞木 $PST(t)$.

PSA を以下に定義する．

定義 6. p-文字列 t に対する $PSA(t)$ とは $t\$$ のすべての真

の接尾辞を直前符号化したものを辞書式順に並べたときの開始位置の列を格納した配列である．

PLCP を以下に定義する．

定義 7. 長さ n の p-文字列 t に対する $PLCP(t)[1:n+1]$ とは以下を満たす配列である．

$PLCP(t)[1]=0$

$i \in 2,\cdots,n$ のとき，$PLCP(t)[i]$ は $prev(t[PSA[i-1]:n+1])$ と $prev(t[PSA[i]:n+1])$ の最長の共通する接頭辞の長さである．

図 2 に PSA，PLCP の例を示す．

i	$PSA[i]$	$prev(x[PSA[i]..n])$	$PLCP[i]$
1	11	\$	0
2	10	0 \$	0
3	6	0 0 1 A 2 \$	1
4	2	0 0 1 A 4 3 1 A 2 \$	4
5	1	0 0 2 1 A 4 3 1 A 2 \$	2
6	3	0 1 A 0 3 1 A 2 \$	1
7	7	0 1 A 2 \$	3
8	4	0 A 0 3 1 A 2 \$	1
9	8	0 A 2 \$	2
10	9	A 0 \$	0
11	5	A 0 0 1 A 2 \$	2

図 2 例は $t=$ `stssAtssAs` の PSA と PLCP である．ただし，$\Sigma=\{\mathtt{A},\$\},\Pi=\{\mathtt{s},\mathtt{t}\}$.

また，PSTr を定義するため π 配列と π 頂点と呼ばれる配列と PST 上の頂点を定義する．以降，任意の文字列について，パラメタ化接尾辞木上である文字列でたどり着く頂点とその文字列を同一視することがある．

定義 8. t を p-文字列とする．$PST(t)$ 上の任意の頂点 v について v を頂点とする部分木の葉の数が $|\Pi|+|\Sigma|$ 以上のとき v を π 頂点と呼ぶ．

定義 9. t を p-文字列, x を $\Pi\cup\Sigma$ の要素, $prev(v)$ を $PST(t)$ 上の π 頂点とする．このとき，v に対する π 配列 $A_{prev(v)}$ を以下のように定義される長さ $|\Sigma|+|\Pi|$ の配列とする．

$A_{prev(v)}[x]$ は $prev(spe(v)x)$ を接頭辞にもつ $prev(v)$ の子供の位置を指し示す．ただし，$prev(v)$ の子供のうち辞書式順序昇順で i 番目が $prev(vy)$ であるとき $prev(v)$ の子供 $prev(vy)$ の位置を i と定義する．

定義 10. p-文字列 t に対するパラメタ化接尾辞トレイ $PSTr(t)$ とは，$PST(t)$ の任意の π 頂点 v について π 配列 A_v を付加したものと，$PSA(t)$，$PLCP(t)$ の 3 つ組である．

次に，以上のように定義した PSTr の領域計算量について議論する．

補題 1. $PST(t)$ 上の π 頂点の個数は $O(\frac{n}{|\Pi|+|\Sigma|})$ である．

補題 2. t を長さ n の p-文字列とする．t に対するパラメタ化接尾辞トレイの領域は $O(n)$ である．

これらの補題は，基本的には通常の文字列に対する接尾辞トレイの議論をパラメタ化文字列に拡張することで証明できる．

5. パラメタ化パターン照合のアルゴリズム

本章では，前章で定義した PSTr を用いたパラメタ化パターン照合について説明する．基本的には通常の文字列に対する接尾辞トレイを用いたパターン照合を拡張できるが，パラメタ化照合による問題点が生じる箇所が存在する．

定義 11. PST 上の頂点 $prev(v)$ の子孫の葉のうち辞書式順序最小のものと最大のものがテキストの各接尾辞を直前符号化した文字列集合の中で辞書式順序昇順にそれぞれ i,j 番目であるとき，$I_v=(i,j)$ と定義する．

PSTr の任意の頂点 v には I_v が記録されているものとする．この作業は全体で $O(n)$ の時間と領域で可能である．

文字列長 m のパターン p が与えられたとする．$prev(p), spe(p)$ の計算をする．この計算は $O(m)$ で可能である．PSTr 上で $prev(p)$ を辿るとする．頂点 $prev(p[:i])(1\le i\le m)$ が σ 頂点であるとき自身のどの辺を辿れば良いかを $A_{prev(p[:i])}[spe(p)[i+1]]$ を用いれば定数時間で見つけられる（π 配列の定義より）．よって，$prev(p)$ が π 頂点の接頭辞であるならば $O(m+occ)$ でパターン照合可能である．$prev(p)$ が π 頂点の接頭辞でないならばどこかで親は π 頂点であるが，自身は π 頂点ではない頂点 $prev(p[:i])$ が存在する．

補題 3. ([4]) 文字列長 m のパターン p とテキスト t において PPM 中の出力すべてが $PSA(t)[i:j]$ に出現するような i,j が与えられたとき $PSA(t), PLCP(t)$ があれば $O(m+\log(i-j)+occ)$ でパターンの出現位置を出力できる．ただし，occ はパターン出現回数である．

上の補題より $PSA(t)$ と $PLCP(t)$ を用いて $I_{prev(p[:i])}$ をもとに探索すれば良い．この場合の計算時間の解析をする．頂点 $prev(p[:i])$ は $O(m)$ 時間で計算できる．ここで，$I_{prev(p[:i])}=(j,k)$ とする．$prev(p[:i])$ は π 頂点でないため $k-j<|\Pi|+|\Sigma|$ である．$PSA(t)$ と $PLCP(t)$ を用いて $prev(p)$ を探すのは $k-j<|\Pi|+|\Sigma|$ より $O(m+\log(|\Sigma|+|\Pi|)+occ)$ である．以上より次のことが言える．

定理 1. パラメタ化接尾辞トレイを用いてパラメタ化パターン照合を $O(m+\log(|\Pi|+|\Sigma|)+occ)$ で解くことができる．ただし，パターン長，出現回数，定数アルファベット，変数アルファベットをそれぞれ m, occ, Σ, Π とする．

6. PSTr の構築

本章では，PSTr の構築について説明する．説明の便利のために以下の 2 つを定義する．

定義 12. q,r を p-マッチする p-文字列とする．$f_{q,r}$ を p-マッチに関する q から r への変換を表す全単射な関数とする．

例えば，$\Pi = x, y, z$ とし，p-マッチする 2 つの文字列 $q = xyxzyyxz$ と $r = zxzyxxzy$ は q 中の x を z に y を x に z を y に変換すると合致するため $f_{q,r}(x) = z$，$f_{q,r}(y) = x$，$f_{q,r}(z) = y$ となる．

以下では，テキスト t 中に出現する Π に含まれる文字の種類数を π とする．

定義 13. q を p-文字列とする．π を q 中に出現する Π の要素の種類数とする．x を Π の要素とする．x の q 中での最左出現位置を $i_{q,x}$ とする．

q に対する初出現配列 $fpos(q)$ は以下のように定義される長さ π の配列である;

$$fpos(q)[x] = i_{q,x}.$$

p-文字列 t に対する PSTr の構築について説明する．$PST(t)$ はすでに与えられているものとする．このとき $PSA(t), PLCP(t)$ は，$PST(t)$ を用いて線形時間で構築できることが知られている．よって，以下では $PST(t)$ 上の任意の π 頂点 v に対し A_v を構築することについてのみ議論する．はじめに π 頂点の判定について説明する．以下の補題を用いることで，すべての π 頂点を計算できる．

補題 4. ([3]) t を長さ n の p-文字列とする．$PST(t)$ 上のすべての π 頂点を $O(n)$ でマークすることができる．

よって，線形時間の前処理を $PST(t)$ に行えば $PST(t)$ 上の任意の頂点について π 頂点かどうかを定数時間で判定できる．次に，$prev(t[i:])$ の祖先である π 頂点 $prev(v)$ に $A_{prev(v)}$ を付加することについて考える．

補題 5. s を p-文字列とする．$prev(s)[|s|] = k \in \mathcal{N}$ とする．このとき，$spe(s)[|s|] = spe(s)[|s| - k]$ が成り立つ．

証明. $prev(s)[|s|] = k$ であるため直前符号化の定義より，$s[|s|] = s[|s| - k]$ が成り立つ．最小符号化の定義より $spe(s) \approx s$ である．また，$spe(s) \approx s$ であるため，p-マッチの定義より $spe(s)[|s|] = spe(s)[|s| - k]$. □

つまり，$prev(v)$ のもつ辺のラベルの 1 文字目を k とするとき，$spe(v)[|v| - k + 1]$ を調べることができれば π 頂点 $prev(v)$ に $A_{prev(v)}$ を付加できる．v と p-マッチする t の部分文字列 v' の開始位置 $m_{v'}$ と $f_{v',spe(v)}$ を求められれば $spe(v)[|v| - k + 1]$ は定数時間で求まる（$f_{v',spe(v)}$ の定義より）．よって，$m_{v'}$，$f_{v',spe(v)}$ を求める方法を考える．次の補題を用いる．

補題 6. $PST(t)$ 上の任意の π 頂点 $prev(v)$ について v と p-マッチする部分文字列 v' の出現位置 $m_{v'}$ と $fpos(v)$ があれば $f_{v',spe(v)}$ をならし $O(|\pi|)$ 時間で計算することができる．

証明. Π の要素のうち j 番目に辞書式順序の小さい文字を x_j とする．v' に出現する文字のうち出現位置が l 番目に辞書式順序が小さい文字を y_l とする．基数ソートを用いることで，$PST(t)$ 上の π 頂点に対する初出現関数を $O(n)$ でソートできるため，$fpos(t[j:])$ をならし $O(|fpos(t[j:])|)$ で辞書式順にソートできる．よって任意の k について，定数時間で y_k を得ることができるため，$f_{v',spe(v)}(y_k) = x_k$ となるように関数を構成すればよい． □

次に，v と p-マッチする t の部分文字列の開始位置 m_v，$fpos(v)$ を求めることを考える．

補題 7. x を v 中に存在する Π の要素とする．$prev(t[j:])$ を $prev(v)$ の子孫の葉とする．このとき $fpos(v)[x] = fpos(t[j:])[x]$ かつ v と p-マッチする t の部分文字列は位置 j に出現する．

証明. 上記の定義および議論より，ただちに証明できる． □

よって，上の補題より $prev(v)$ の子孫の葉 $prev(t[i:])$ に対し，$fpos(t[i:])$ を求めることで出現位置を計算できる．$fpos(t[i:])$ は次のように求める．

補題 8. q を p-文字列とする．x, y を $x \neq y$ を満たす Π 上の文字とする．このとき，次が成り立つ; $fpos_{xq}(x) = 1$, $fpos_{xq}(y) = fpos_q(y)$.

したがって，上の補題を用いることで $fpost[i+1:]$ から定数時間で $fpost[i:]$ を計算することができる．

全体のアルゴリズムは以下のようになる．

ステップ 1：$PST(t)$ の任意の π 頂点 $prev(v)$ に，v と p-マッチする部分文字列 v' が出現する位置 m_v と $fpos(v)$ を付加する．

$fpos(v)$ と m_v を $prev(v)$ に付加する具体的な方法ついて述べる．仮定として，$prev(t[n:])$ から $prev(t[i+1:])$ の祖先である任意の π 頂点 v' には $fpos(v')$ と $m_{v'}$ がそれぞれ計算されており，$fpos(t[i+1:])$ はすでに計算されているものとする．$prev(t[i:])$ の祖先の π 頂点 v_1 に $fpos(v_1)$ と m_{v_1} を計算することを考える．まず，$fpos(t[i+1:])$ を $fpos(t[i:])$ に更新する．補題 8 より，この更新は定数時間でできる．祖先の π 頂点を文字列深さ順に遡り，v_1 に $fpos(t[i:])$ をコピーし $fpos(v_1)$ とする．また，$j_{v_1} = i$ とする．ただし，すでに $fpos(v_1)$，m_{v_1} が計算されているのならば何もしない．$fpos(v_1)$ が計算されているならば，仮定より $prev(t[n:])$ から $prev(t[i+1:])$ の葉の祖先のため，$prev(v_1)$ の任意の祖先 $prev(v_2)$ にも当然 $fpos(v_2)$，j_{v_2} が計算されているため何もしなくても良い．

ステップ 2：$fpos(v)$ を辞書式順にソートする．補題 6 より $f_{v,spe(v)}$ を $f_{v,spe(v)}$ の大きさに対してならし線形時間で計算可能である．

ステップ 3：$m_{v'}$ とソートされた $fpos(v)$ をもとに $f_{v',spe(v)}$ を計算する．

ステップ 4：$f_{v',spe(v)}$ をもとに A_v を計算する．

$f_{t[i:],spe(t[i:])}$ を用いて $t[i]$ 上の π 頂点に A_v を計算する．

それぞれのステップはテキスト長について線形時間で計算できる．よって以下が成り立つ．

定理 2. t を長さ n の p-文字列とする．$PST(t)$ が与えられれば t に対するパラメタ化接尾辞トレイを $O(n)$ 時間，領域で構築できる．

7. まとめと今後の課題

本研究では，パラメタ化パターン照合を高速に行うための新たな索引構造であるパラメタ化接尾辞トレイを提案した．パラメタ化接尾辞木からパラメタ化接尾辞トレイを $O(n)$ 時間領域で構築することができ，クエリ時間は $O(m + \log(|\Sigma| + |\Pi|) + occ)$ である．パラメタ化パターン照合問題だけではなくデカルト木照合問題 [12] や順序同型照合問題 [9] でも接尾辞木や接尾辞配列を模倣した索引構造が提案されている．よってデカルト木照合問題や順序同型照合問題でも同様の手法を用いて高速な索引構造を作ることができるかどうかが今後の課題として考えられる．

参考文献

[1] B. S. Baker. A theory of parameterized pattern matching: algorithms and applications. In *STOC 1993*, pages 71–80, 1993.

[2] B. S. Baker. Parameterized pattern matching: Algorithms and applications. *J. Comput. Syst. Sci.*, 52(1):28–42, 1996.

[3] R. Cole, T. Kopelowitz, and M. Lewenstein. Suffix trays and suffix trists: Structures for faster text indexing. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Automata, Languages and Programming*, pages 358–369, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[4] S. Deguchi, F. Higashijima, H. Bannai, S. Inenaga, and M. Takeda. Parameterized suffix arrays for binary strings. In J. Holub and J. Žďárek, editors, *Proceedings of the Prague Stringology Conference 2008*, pages 84–94, Czech Technical University in Prague, Czech Republic, 2008.

[5] Diptarama, T. Katsura, Y. Otomo, K. Narisawa, and A. Shinohara. Position heaps for parameterized strings. In *Proc. CPM 2017*, pages 8:1–8:13, 2017.

[6] A. Ehrenfeucht, R. M. McConnell, N. Osheim, and S.-W. Woo. Position heaps: A simple and dynamic text indexing data structure. *Journal of Discrete Algorithms*, 9(1):100–121, 2011.

[7] N. Fujisato, Y. Nakashima, S. Inenaga, H. Bannai, and M. Takeda. Right-to-left online construction of parameterized position heaps. In *Proc. PSC 2018*, pages 91–102, 2018.

[8] N. Fujisato, Y. Nakashima, S. Inenaga, H. Bannai, and M. Takeda. Direct linear time construction of parameterized suffix and LCP arrays for constant alphabets. *CoRR*, abs/1906.00563, 2019.

[9] J. Kim, P. Eades, R. Fleischer, S. Hong, C. S. Iliopoulos, K. Park, S. J. Puglisi, and T. Tokuyama. Order preserving matching. *CoRR*, abs/1302.4064, 2013.

[10] G. Kucherov. On-line construction of position heaps. *J. Discrete Algorithms*, 20:3–11, 2013.

[11] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 319–327, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[12] S. G. Park, A. Amir, G. M. Landau, and K. Park. Cartesian tree matching and indexing. *CoRR*, abs/1905.08974, 2019.

[13] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (Swat 1973)*, SWAT '73, pages 1–11, Washington, DC, USA, 1973. IEEE Computer Society.