Computing All Repeats of a Partial Word

K. Sasikala[1], V.R. Dare[2] and D.G. Thomas[2]

[1]Department of Mathematics, St. Joseph's College of Engineering
Chennai - 119, India. Email : sasikalaveerabadran@yahoo.co.in

[2]Department of Mathematics, Madras Christian College
Chennai - 59, India

Abstract

In this paper, we describe two algorithms to identify the repeating subwords in a given partial word $w_0 = w_0[1, \ldots, n]$. The first algorithm uses the suffix tree and the second algorithm uses the valency tree. Both algorithms take linear time to identify the repeating subwords of a partial word.

1 Introduction

Partial words appear in comparing genes. Indeed, alignment of two strings can be viewed as a construction of two partial words that are compatible in a sense that was described in [2]. The computation of all repeating substrings in a given string $x = x[1 \ldots n]$ is a problem with various application areas, most notably data compression; cryptography and computational biology [1, 3]. Motivated by these studies, in this paper we describe two algorithms to identify all repeating subwords of a given partial word.

2 Preliminaries

In this section, we give a short review of some basic notions on partial words that will be used throughout this study [2].

Partial words are strings of symbols from a finite alphabet that may have a number of "do not know" symbols. Similar to the definition of a word as a total function from $\{1, \ldots, |w|\}$ to Σ, a partial word w is defined as a partial function from $\{1, \ldots, |w|\}$ to Σ. The positions, where $w(n)$ (the n^{th} letter of w) is not defined for $n < |w|$ are called holes of w. If $D(w)$ stands for the domain of w, then the set of holes of w denoted by $H(w)$ is the set of numbers in $\{1, \ldots, |w|\} \backslash D(w)$.

A word over Σ is a partial word over Σ with an empty set of holes (we sometimes refer to words as full words). For any partial word u over Σ, $|u|$ denotes its length. In particular $|\lambda| = 0$.

We denote by W_0 the set Σ^*, and for every integer $i \geq 1$, by W_i the set of partial words over Σ with exactly i holes. $W = \bigcup_{i \geq 0} W_i$ is the set of all partial words over Σ with an arbitrary number of holes.

If u is a partial word of length n over Σ, then the companion of u (denoted by $u_\Diamond$) is the total function $u_\Diamond : \{1, \ldots, n\} \to \Sigma \cup \{\Diamond\}$ defined by

$$u_\Diamond(i) = \begin{cases} u(i) & \text{if } i \in D(u) \\ \Diamond & \text{otherwise} \end{cases}$$

The symbol $\Diamond \notin \Sigma$ is viewed as a "do not know" symbol. The word $u_\Diamond = abb\Diamond b\Diamond bb$ is the companion of the partial word u of length 8 where $D(u) = \{1, 2, 3, 5, 7, 8\}$ and $H(u) = \{4, 6\}$.

The reverse of a word $u = a_0 a_1 \ldots a_{n-1}$ is $rev(u) = a_{n-1} \ldots a_1 a_0$. The reverse of a partial word u is $rev(u)$. The reverse of a set $X \subset W$ is the set $rev(X) = \{rev(u) | u \in X\}$.

Definition 2.1 *If u and v are two partial words of equal length then u is said to be contained in v, denoted by $u \subset v$, if $D(u) \subset D(v)$ and $u(i) = v(i)$ for all $i \in D(u)$. The partial words u and v are compatible, denoted by $u \uparrow v$ if there exists a partial word w such that $u \subset w$ and $v \subset w$.*

We now recall the concept of special factors of a partial word introduced in [5, 4].

Definition 2.2 *Let w be a partial word over an alphabet Σ. A word u is a factor or subword of w if there exist words or partial words p, q over Σ such that $w = puq$. $F(w)$ denotes the set of all its factors (subwords) and $alph(w)$, the set of all letters of the alphabet Σ occurring in w.*

Example 2.1 *Let $w = ab\Diamond baa\Diamond a$. The factors of w are $\lambda, a, b, ab, ba, aa, baa$.*

Let the cardinality of Σ i.e., $card(\Sigma) = d$ and w be a partial word over the alphabet Σ. Let u be a factor of w. Since w is a partial word, the factor u can be followed either by a letter of Σ or by the special symbol $\Diamond$. We now define the positive and negative right valence of u.

For any factor u of w, we consider the maximal subset, with respect to inclusion, R of Σ, such that $uR \subseteq F(w)$. Thus for all letters $x \in R$, one has $ux \in F(w)$. We now introduce the maps (i) $v_{r+} : F(w) \to Z^+$ (the set of all non-negative integers) defined for each $u \in F(w)$ as $v_{r+}(u) = card(R)$. v_{r+} is called the positive right valence of u. (ii) $v_{r-} : F(w) \to Z^-$ (the set

of all negative integers) defined for each $u \in F(w)$ as $v_{r-}(u) = -m$, if $u\Diamond$ occurs in m places in w is called the negative right valence of u.

Similarly one can define the left valence.

For any factor u of w, we consider the maximal subset, with respect to inclusion, L of Σ such that $Lu \subseteq F(w)$, so that for all letters $x \in L$, $xu \in F(W)$. (i) $v_{l+} : F(w) \rightarrow Z^+$ is defined as for each $u \in F(w)$, $v_{l+}(u) = card(Lu) = card(L)$. (ii) $v_{l-} : F(w) \rightarrow Z^-$ is defined as for each $u \in F(w)$, $v_{l-}(u) = -m$, if $u\Diamond$ occurs in m places in w.

Definition 2.3 *A factor u of a partial word w is said to be right special if it is followed by at least two distinct letters of Σ, i.e, $v_{r+}(u) \geq 2$.*

Similarly, left special factors can be defined.

The empty subword λ of w has according to the definition a right and a left valence equal to $card(alph(w))$.

For any $u \in F(w)$, $-m \leq v_{r-}, v_{l-} \leq -1$; $0 \leq v_{r+}, v_{l+} \leq d$ where $m \in N$, $card(\Sigma) = d$.

Definition 2.4 *A factor u of a partial word w is said to be bispecial if it is both right and left special.*

Remark 2.1

1. *For any factor u of w, if $v_{r-}(u) = v_{l-}(u) = -1$, then u is bordered by holes.*

2. *For any factor u of w if either $v_{r-}(u) = -1$ or $v_{l-}(u) = -1$, then it is either preceded or followed by a hole.*

3. *For any factor u of w if (i) u is the suffix of w and (ii) it is either not repeated or only followed by a hole in w, then its right valence is defined as $v_{r+}(u) = 0$. Similarly $v_{l+}(u) = 0$, if (i) u is a prefix of w and ii) it is either not repeated or only preceded by a hole in w.*

Example 2.2 *Let $w = ab\Diamond ba\Diamond aa\Diamond ba$.*
$F(w) = \{a, b, ab, ba, aa\}$
$v_{r-}(a) = -2, v_{r-}(b) = -1, v_{r-}(ab) = -1, v_{r-}(ba) = 0, v_{r-}(aa) = -1$.

3 Computing all Repeats of a Partial Word using Suffix Arrays

In this section, we describe an algorithm to identify all repeating factors of a given partial word by using the notion of a suffix tree.

Definition 3.1 *Let w be a partial word. A repeat of u (subword) of a partial word w is a tuple $M_{w,u} = (p; i_1, i_2, \ldots, i_r), r \geq 2$, where*

$$u = w[i_1, \ldots, i_1 + p - 1] = w[i_2, \ldots, i_2 + p - 1] = \cdots = w[i_r, \ldots, i_r + p - 1].$$

Then u is said to be a repeating substring of w and u is a generator of $M_{w,u}$. Again, $p = |u|$ is the period of the repeat. If all the occurrences of u in w are counted in $M_{w,u}$, then $M_{w,u}$ is said to be complete and the tuple is written as $M^*_{w,u}$.

Example 3.1 *Let $w = abab\Diamond baaba\Diamond baaab$*
$u = aba$, $p = 3$, $M_{w,u} = (3; 1, 8)$, $u = w[1, 2, 3] = w[8, 9, 10]$. Since all the occurrences of aba are counted in $M_{w,u}$, we have $M^*_{w,u} = (3; 1, 8)$.

Definition 3.2 *A repeat $M_{w,u} = (p; i_1, i_2, \ldots, i_r)$ is said to be left-extendable (respectively right-extendable) if $(p+1; i_1 - 1, i_2 - 1, \ldots, i_r - 1)$ (respectively, $(p+1; j_1, j_2, \ldots, j_r)$) is a repeat. If $M_{w,u}$ is neither left-extendable nor right-extendable then it is said to be nonextendable. We abbreviate these terms as LE, RE and NE respectively.*

Example 3.2 *Let $w = abab\Diamond baaba\Diamond baaab$ and $u = ba$.*
Then $M_{w,u} = (2; 2, 6, 9, 12)$, $M_{w,aba} = (3; 1, 8)$, $M_{w,baa} = (3; 6, 12)$. Therefore $u = ba$ is left-extendable as well as right-extendable. $M_{w,baa} = (3; 6, 12)$ is neither left-extendable nor right-extendable. Hence it is nonextendable.

The repeats which are right-extendable and left-extendable are substrings of NE repeating subwords. Therefore in order to find all repeating subwords of a partial word, it suffices to find only NE repeating subwords.

In the last few years, several algorithms have been proposed that employ suffix trees to compute all the NE repeats in a given string x defined on an ordered alphabet [3]. We employ suffix array and suffix tree to compute all repeats of a partial word.

We give an algorithm to compute the suffix array of a partial word w.

Definition 3.3 *The prefix of the partial word upto the first hole of w is called as the initial hole box. The suffix of w after the last hole of w is called as the terminal hole box. A factor of w, which is preceded as well as followed by a hole is called as a proper hole box.*

Let w be a partial word of length n. Let n_1 be the number of holes in w and the positions of the holes be $h_1, h_2, \ldots, h_{n_1}$. Since there are n_1 holes, the number of proper hole boxes is $n_1 - 1$. Let HB_1 be the initial hole box, $HB_2, \ldots, HB_{n_1}$ be the proper hole boxes and HB_{n_1+1} be the terminal hole box.

Definition 3.4 *Let $w = w[1, \ldots, n]$ be a given partial word and n_1 be the number of holes in w. The suffix array $\lambda_w = \lambda[1, \ldots, n - n_1]$ is an array of integers, such that $\lambda[i] = j$, if i^{th} smallest suffix of the hole boxes of w is either $w[j, \ldots, h_k - 1]$ where h_k is the first hole followed by the sequence of letters from the j^{th} position or $w[j, \ldots, n]$ where there is no hole after the j^{th} position.*

For effective use of suffix arrays, a second array π_w is also computed that gives the length of the longest common prefix (lcp) of adjacent entries in λ_w.

Remark 3.1 *If a partial word starts with a hole, then there is no initial hole box and if it ends with a hole, then there is no terminal hole box.*

Algorithm 1 (Compute suffix array and array π_w)
Given the partial word w

1. *Compute the hole boxes of w*

$$\begin{aligned} HB_1 &= w[1, \ldots, h_1 - 1] \\ &\vdots \\ HB_i &= w[h_{i-1} + 1, \ldots, h_i - 1] \\ &\vdots \\ HB_{n_1} &= w[h_{n_1 - 1} + 1, \ldots, h_{n_1} - 1] \\ HB_{n_1+1} &= w[h_{n_1} + 1, \ldots, n] \end{aligned}$$

2. *Compute the suffixes of hole boxes*
   $h_0 = 0, h_{n_1+1} = n + 1$
   For $i = 1$ *to* $n_1 + 1$
   $suffixes(HB_i) = \{w[h_i - 1], w[h_i - 2, h_i - 1], \ldots, w[h_{i-1} + 1, \ldots, h_i - 1]\}$

3. *Compute the suffix array λ_w. Arrange the suffixes (HB_i) for $i = 1$ to $n_1 + 1$ in ascending lexicographical order. The suffix array $\lambda_w = \lambda[1, \ldots, n - n_1]$ is an array of integer, such that $\lambda[i] = j$, if i^{th} smallest suffix of the hole boxes of w is either $w[j, \ldots, h_k - 1]$ where h_k is the first hole followed by the sequence of letters from the j^{th} position or $w[j, \ldots, n]$ where there is no hole after the j^{th} position.*

4. *Compute the array π_w*
   $\pi_w(1) = 0$, $\pi_w(i) = lcp(\lambda_w(i - 1), \lambda_w(i))$, *for* $i = 2$ *to* $n - n_1$.

Example 3.3 *Let* $w = abab\Diamond baaba\Diamond baaab\$$
$n = |w| = 16$, $n_1 = 2$
$D(w) = \{1, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16\}$
$H(w) = \{5, 11\}$, $h_1 = 5, h_2 = 11$.
$HB_1 = abab$, $HB_2 = baaba$, $HB_3 = baaab$.
suffixes of (HB_1) *is* $\{b, ab, bab, abab\}$
suffixes of (HB_2) *is* $\{a, ba, aba, aaba, baaba\}$
suffixes of (HB_3) *is* $\{b, ab, aab, aaab, baaab\}$.
The ascending lexicographical order of the suffixes of hole boxes is $\{a, aaab, aab, aaba, ab, ab, aba, abab, b, b, ba, baaab, baaba, bab\}$.

Thus λ_w, *specifies the starting positions of the suffixes of hole boxes of* w *in ascending lexicographical order.*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\lambda_w =$	10	13	14	7	3	15	8	1	4	16	9	12	6	2

π_w *is given below*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\lambda_w =$	10	13	14	7	3	15	8	1	4	16	9	12	6	2
$\pi_w =$	•	1	2	3	1	2	2	3	0	1	1	2	3	2

For example lcp(7,3) = 1, as a is the longest common prefix of aaba and ab.

Lemma 3.1 *Let* $\hat{w}$ *be the reverse* $w[n]\ w[n-1]\ \ldots\ w[1]$ *of the partial word* w. *Then a repeat* $M_{w,u}$ *is LE if and only if* $M_{\hat{w},\hat{u}}$ *is RE.*

We now give the suffix tree of a partial word and its reversal. Non Right Extendable (NRE) repeats of a partial word are identified by its suffix tree (as well as by its suffix array). The suffix tree represents only the distinct prefixes of each suffix of hole boxes of w. The internal (◯) nodes are recognizable as the longest common prefix (lcp) values contained in the array π_w, while the leaf (□) nodes represent the starting positions i of the suffixes of hole boxes $HB_i, i = 1, 2, \ldots, n_1 + 1$, as specified in the array λ_w.

As in every suffix tree, the lcp value at the root of each subtree applies to all the leaf nodes of that subtree. The lcp values also specify the complete repeats in w, which are NRE.

Example 3.4 *Suffix tree of* $w = abab\Diamond baaba\Diamond baaab\$$ *is shown in Figure 1.*
Since $H(w) = \{5, 11\}$, *these positions are not indicated in the tree structure.*

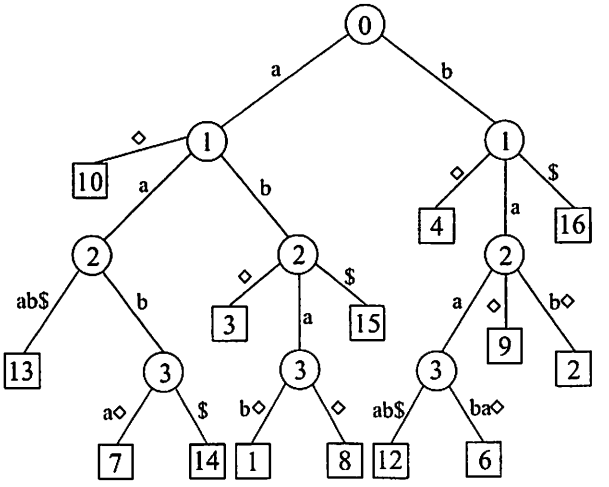

Figure 1: Suffix tree of w

The NRE repeats of w *are* $w[7,8,9] = aab = w[14,15,16]$,
$w[1,2,3] = aba = w[8,9,10]$, $w[12,13,14] = baa = w[6,7,8]$.

Thus the NRE repeating subwords are aab, aba *and* baa.

Let us consider the reverse of w, $\hat{w} = baaab\Diamond abaab\Diamond baba\$$.
$HB_1 = baaab$, $HB_2 = abaab$, $HB_3 = baba$
suffixes of (HB_1) *is* $\{b, ab, aab, aaab, baaab\}$
suffixes of (HB_2) *is* $\{b, ab, aab, baab, abaab\}$
suffixes of (HB_3) *is* $\{a, ba, aba, baba\}$

The suffixes of hole boxes of $\hat{w}$ *can be written in the ascending lexicographical order as follows.*
$\hat{w} = \{a, aaab, aab, aab, ab, ab, aba, abaab, b, b, ba, baaab, baab, baba\}$.
The suffix tree of $\hat{w}$ *is shown in Figure 2.*

Thus $\lambda_{\hat{w}}$, *specifies the starting positions of the suffixes of hole boxes of* $\hat{w}$ *in ascending lexicographical order.*
$H(\hat{w}) = \{6, 12\}$. *Position of holes are not indicated in the suffix tree.*
The NRE repeats of $\hat{w}$ *are*
$\hat{w}[3,4,5] = \hat{w}[9,10,11] = aab$
$\hat{w}[7,8,9] = \hat{w}[14,15,16] = aba$
$\hat{w}[8,9,10] = \hat{w}[1,2,3] = baa$.
Thus the NRE repeats of w *and* $\hat{w}$ *are* $\{aab, aba, baa\}$.

Lemma 3.1, suggests a straight forward approach to compute all the NE repeats in w.

- *Compute all the NRE repeats of* w *and all the NRE repeats of* $\hat{w}$.

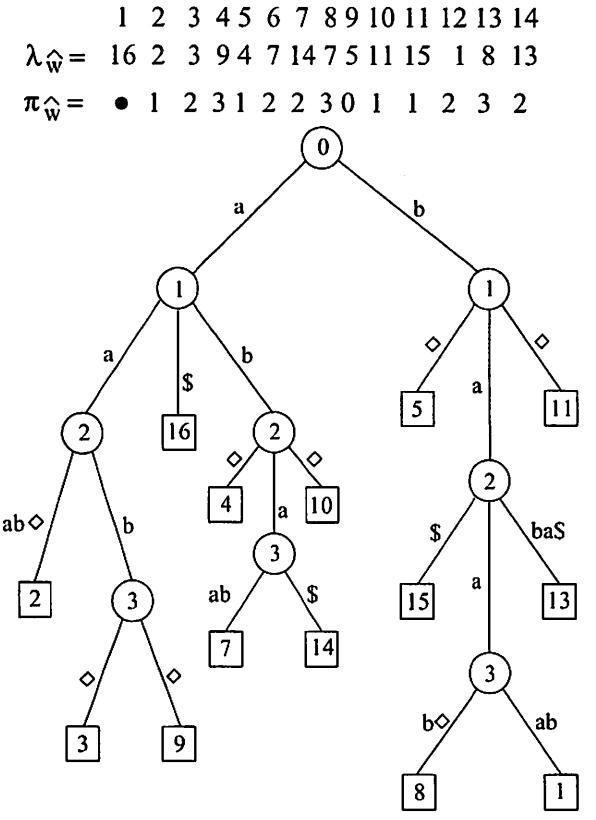

Figure 2: Suffix tree of $\hat{w}$

- *Compare the NRE repeats of w and $\hat{w}$ to identify the repeats in both lists. They are the NE repeats of w. Thus NE repeating subwords of w are aab, aba, baa.*

Computing the NE Tree

From Lemma 3.1, we know that the NRE repeats in $\hat{w}$ identify NLE repeats in w. Suppose that some non-zero lcp node $\hat{p}$ in $T_{\hat{w}}$ has as child at position node $\hat{i}$. Then in $\hat{w}$, there exists an NRE repeating substring $\hat{u} = \hat{w}[\hat{i}, \ldots \hat{i} + \hat{p} - 1]$, that is, the reverse of an NLE repeating substring

$$u = w[n - (\hat{i} + \hat{p} - 2), \ldots, n - (\hat{i} - 1)] \text{ in } w.$$

Thus the assignment $i \leftarrow n - (\hat{i} + \hat{p} - 2)$ identifies the starting position i of an NLE repeating substring u in w of length $\hat{p}$.

Definition 3.5 *A substring u of w is maximal NE repeating substring of w if*

1. *u occurs at least twice in w*
2. *u is not a proper substring of any repeating substring of w.*

Lemma 3.2 *If $u = w[i, \ldots, i+p-1]$ is a maximal NE repeating substring of w, then the position node i occurs as a child of the lcp node p in T_w and $n-(i+p-2)$ occurs as a child of p in $T_{\hat{w}}$.*

This lemma can be easily proved using the transformation on $i \leftarrow n-(i+p-2)$, and which tells us that a maximal NE repeating substring must be identifiable in both T_w and $T_{\hat{w}}$ and provides us with a simple strategy for identifying all NE repeating substrings, that is, find the largest value of p, then locate all their ancestors in T_w.

Algorithm 2 (Compute the NE tree using suffix tree)
Given the suffix trees T_w and $T_{\hat{w}}$, compute the NE tree of $w[1, \ldots, n]$.

1. *traverse T_w to create a table POINTER[i], that for each position i in w points to the corresponding node in T_w.*
2. *for every lcp node p in T_w do*
   $NE[p] \leftarrow False$
3. *for every parent-child pair $(\hat{p}, \hat{i})$ in $T_{\hat{w}}$ $(\hat{p} > 0)$ do*
   Here use POINTER[i]
   if $i = n-(\hat{i}+\hat{p}-2)$ is a child of lcp node $\hat{p}$ in T_w then
   while not NE[$\hat{p}$] do
   NE[$\hat{p}$] $\leftarrow$ true
   if $\hat{p} \neq 0$ then
   $\hat{p} \leftarrow$ parent of $\hat{p}$ in T_w
4. *traverse T_w deleting every sub tree rooted at an lcp node p for which NE[p] = FALSE*

Algorithm 2 is expressed in terms of four steps. Step (1) is a traversal of T_w that sets up a table enabling each position node i in T_w to be accessed later in constant time. In Step (2) a Boolean variable corresponding to each marked node p in T_w is initialized to FALSE, indicating that no terminal nodes in the subtree rooted at p have currently been identified as NE. Step (3) processes every parent-child pair $(\hat{p}, \hat{i})$ in $T_{\hat{w}}$, testing to determine whether or not the equivalent parent-child pair $(\hat{p}, n-(\hat{i}+\hat{p}-2))$ exists in T_w, if so, then the marked node $\hat{p}$ and all its ancestors in T_w must be NE accordingly, until an ancestor is found that is already NE, $\hat{p}$ and its

ancestors in T_w are identified as NE. A final step traverses T_w to eliminate all subtrees rooted at any node p for which $NE[p]$ = FALSE, the remaining tree is the NE tree of w.
Figure 3 displays the tree modified by the transformation $i \leftarrow n-(i+\hat{p}-2)$ for the partial word $w = abab\Diamond baaba\Diamond baaab\$$.

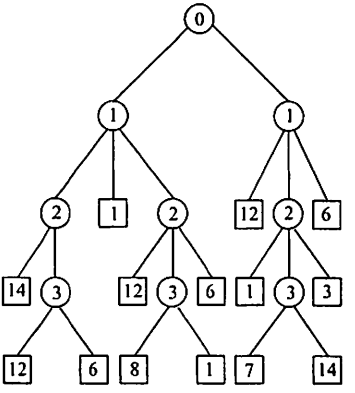

Figure 3: $T_{\hat{w}}$ with leaves labeled by the transformation $i \leftarrow n-(\hat{i}+\hat{p}-2)$

The NE tree of w is shown in Figure 4.

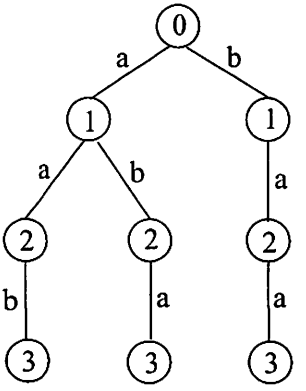

Figure 4: NE tree for w

Thus the repeating subwords of the given partial word w are $a, b, aa, ab, ba, aab, aba, baa$.

Time Complexity

In Algorithm 1 (Compute suffix array λ_w and π_w)

1. The time taken to compute the hole boxes $HB_1, HB_2, \ldots, HB_{n_1+1}$ of the partial word w is n, where $n = |w|$.

2. Let $|HB_1| = m_1$, $|HB_2| = m_2$, $\ldots$, $|HB_{n_1+1}| = m_{n_1+1}$. The time taken to compute the suffixes of the hole box HB_j is $\frac{m_j(m_j+1)}{2}$, $j = 1, 2, \ldots, n_1 + 1$. The total time taken to compute the suffixes of all the hole boxes is $\sum_{j=1}^{n_1+1} \frac{m_j(m_j+1)}{2} = M$, that is, the number of suffixes is M.

3. To arrange the suffixes in ascending lexicographical order, the Algorithm 1 takes M units of time.

4. The Algorithm 1 takes $2M$ units of time to compute the suffix array λ_w and π_w. Therefore the Algorithm 1 takes $n + 4M$ units of time.

5. In Algorithm 2 (Compute the NE tree using suffix tree), the time taken to locate the positions and the lcp values is $2n$ units of time and the time taken to identify the suffixes is M units of time. Therefore the time taken to construct the suffix tree is $2n + M$ units of time, similarly the time taken to construct $T_{\tilde{w}}$ is $2n + M$ units of time.

6. Let $n_2 < n$ be the number of nodes of NE tree. To traverse the NE tree the time taken is $n + n_2 < 2n$ units of time. Similarly to delete the nodes from NE tree the time taken is $2n$ units of time. Therefore the total time taken by the Algorithms 1 and 2 is $n + 4M + 2n + M + 2n + M + 2n + 2n = 9n + 6M = O(n)$.

4 Valence Tree of a Partial Word

In this section, the valence tree of a partial word is given. The NE tree of a partial word is identified by the valencies of the factors of w.

Remark 4.1 *A factor u of a partial word w is a repeating factor if any one of the following conditions are satisfied.*

1. *u is right special.*

2. *u is left special.*

3. *u is hole special.*

4. *$v_{r_+}(u) = m$, $v_{r_-}(u) = -n$, $m = 0, 1$, $n \geq 1$.*

Let us use the following procedure to find the repeating factors of a partial word w using valence tree.

- Find $F(w)$.
- Find their valencies.
- If the valence of a factor u of w satisfy any one of the conditions given in Remark 4.1, then u is a repeating factor.

Algorithm 3 (Compute the NE tree using valence tree)
Given the valence tree VT_w, compute the NE tree of $w[1,\ldots,n]$.

1. *Traverse VT_w to create a table $POINTER[i]$ that position i in w points to the corresponding node in VT_w.*
2. *For every valence node i and valence node (i,j) in VT_w do $NE[i] \leftarrow False$, $NE(i,j) \leftarrow False$.*
3. *For every node i in VT_w*
   Here use $POINTER[i]$
   if valence node (i) or valence node (i,j) satisfies one of the following conditions:
   (a) $v_{r_+}(i) \geq 2$.
   (b) $v_{r_-}(i) = -n, n \geq 2$.
   (c) $v_{r_+}(i) = m, v_{r_-}(j) = -n, m = 0,1, n \geq 1$. *(For simplicity, we write this condition as $v_r(i,j) = (m,-n)$, where $m = 0,1$, $n \geq 1$).*

   then $NE(i) \leftarrow True$ or $NE(i,j) \leftarrow True$.
4. *Traverse VT_w deleting every subtree rooted at a node for which $NE(i) \leftarrow False$ or $NE(i,j) \leftarrow False$.*

The valence tree represents only the distinct prefixes of each factor of w, and the circular node represents the valencies of the factors of w.

Example 4.1 *Let $w = abab\Diamond baaba\Diamond baaab$*
$F(w) = \{a, b, ab, ba, aa, aba, bab, baa, aaa, aab, abab, baab, aaba, baaa, aaab, baaba, baaab\}$
The valence tree of w is shown in Figure 5.

Hence the repeating subwords of w are $a, b, aa, ab, ba, aab, aba, baa$.
The NE tree of a partial word is shown in Figure 6.

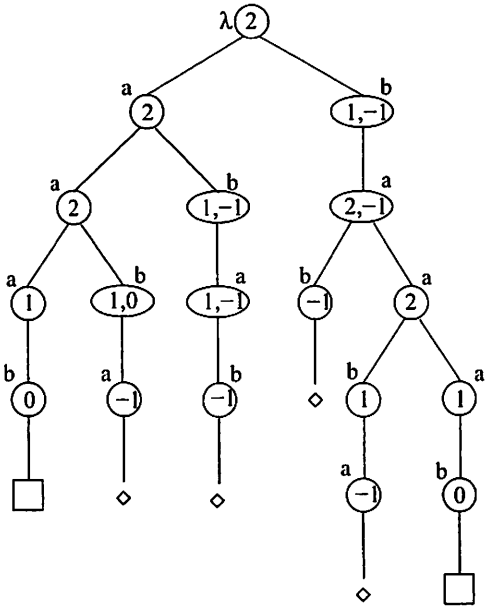

Figure 5: Valence tree of a partial word w

Time Complexity

1. The time taken to compute the hole boxes of w is n, where $n = |w|$.
2. The time taken to compute the factors of w is nML, where ML is the length of the maximal subword of w.
3. To find the valencies of the factors, the time taken is nM_1, where M_1 is the number of factors of w.
4. To construct the valence tree, the time taken is M_1 units of time.
5. To compute the NE tree, the time taken to traverse the tree is n_2, where n_2 is the number of nodes of NE tree.

Therefore the total time taken is
$n + nML + nM_1 + M_1 + n_2 = 2n + n_2 + (n+1)M_1 = O(n)$ where n_2, $ML < n$.

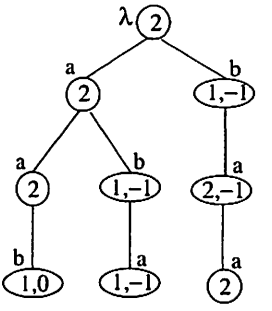

Figure 6: NE tree of a partial word w

Conclusion

We describe Algorithm 2 and Algorithm 3 to identify the repeating subwords in a given partial word $w = w[1, \ldots, n]$. Algorithm 2 uses the suffix tree and Algorithm 3 uses the valence tree. Both algorithms take linear time to identify the repeating subwords of a partial word.

Comparing Algorithm 2 and Algorithm 3 given in this Chapter to compute the repeating subwords of a partial word w, we found that Algorithm 2 is simple and number of steps in the computation of NE tree is less in Algorithm 3.

References

[1] Aldo De Luca. On the combinatorics of finite words. *Theoretical Computer Science*, 218:13–39, 1999.

[2] Berstel. J and Boasson. L. Partial words and a theorem of Fine and Wilf. *Theoretical Computer Science*, 218:135–141, 1999.

[3] Frantisek Franek, Willium F Smyth, and Yudong Tang. Computing all repeats using suffix arrays. *Journal of Automata, Languages and Combinatorics*, 8:579–591, 2003.

[4] Sasikala. K and Dare. V.R. Special factors of partial words and trapezoidal partial words. In *Graphs, Combinatorics, Algorithms and Applications*, pages 135–143. Narosa Publishing House, 2005.

[5] Sasikala. K, Kalyani. T, Dare. V.R, and Abisha. P.J. Subword complexity of partial words. In *Proceedings of NCMCM 2003*, pages 253–259. Allied Publishers, 2003.