Algorithmic Bioinformatics Burrow-Wheeler Algorithm

Måns Magnusson, Farzon Nosiri

2010-05-06

Abstract

BLAST has been used for many years when aligningn short reads against genomes but it is not fast enough any more. We describe BWA: A fast and accurate short read alignment with Burrows-Wheeler transform.
By Hang Le and Richard Durbin

1 Introduction

Given a query W with respect to a database(genome) BWA can, after some preprocessing, give back an exact match in time $\mathcal{O}(|W|)$.

1.1 Preprocessing

- Let Σ be an lexicographic alphabet, for example $\Sigma = \{A, C, G, T\}$, with \$ being the smallest element, the rest can be in any order.
- Let $X = x_0 x_1 \ldots x_{n-1}$ where $x_i \in \Sigma$, $0 \leq i \leq n-2$ and $x_{n-1} = \$$.
- We say that $X[i] = x_i$, $0 \leq i \leq n-1$, is the i:th symbol of X, $X^{\geq i} = x_i \ldots x_{n-1}$ is a suffix string of X and $X^{\geq i, \leq j} = x_i \ldots x_j$.
- A suffix array(SA) for X is an array S where $S[i]$ is the start position of the i:th smallest suffix of X.
- The Burrows-Wheeler Transform of X is defined as follows:

$$B[i] = \begin{cases} \$ & \text{if } S[i] = 0, \\ X(S[i]-1) & \text{otherwise.} \end{cases}$$

- We also define the length of string X as $|X|$ and therefore we have that $|X| = |B| = n$.

Example

Our genome = googol so X = googol$

$Positions$	$Suffixes$
0	googol$
1	oogol$g
2	ogol$go
3	gol$goo
4	ol$goog
5	l$googo
6	$googol

Sorting $\Longrightarrow$

i	$S(i)$	$Suffixes$	$StartPositions$
0	6	$googo	l
1	3	gol$go	o
2	0	googol	$
3	5	l$goog	o
4	2	ogol$g	o
5	4	ol$goo	g
6	1	oogol$	g

Here we have sorted the suffixes in lexicographical order.
The positions of the first symbols form the suffixarray $S(i) = (6,3,0,5,2,4,1)$ and the concatenation of the last symbols of the circulated strings gives the BWT string $B[i]$ = lo$oogg.

End of example

Observe

Each occurence of W is in a interval of the Suffix Array S.

We will search for the so called SA interval of W.

Definition 1. *The SA interval of W is $[\underline{R}(W), \overline{R}(W)]$ where*

$$\underline{R}(W) = min\{k\text{: } W \text{ is a prefix of } X^{\geq S(k)}\} \tag{1}$$
$$\overline{R}(W) = max\{k\text{: } W \text{ is a prefix of } X^{\geq S(k)}\} \tag{2}$$

Observe

All occurences of W in X have startposition in

$$\{S(k) : \underline{R}(W) \leq k \leq \overline{R}(W)\}$$

Moreover $\underline{R}(W) \leq \overline{R}(W)$ if and only if W occur in X.

Theorem (Ferragine, Manzini, 2000)

Let $c(a)$ = The number of i such that X_i is lexicographically smaller than $a \in \Sigma$
and let
$O(a,i)$ = The number of occurences of a in $B^{\geq 0, \leq i}$
then

$$\underline{R}(aW) = c(a) + O(a, \underline{R}(W) - 1) + 1$$
$$\overline{R}(aW) = c(a) + O(a, \overline{R}(W))$$

Where aW is the symbol a concatenated to the string W.
For example if our original query was $W = ogo$ then in the theorem above $a = o$ and $W = go$.

In particular for the empty string ε, we have that $\underline{R}(\varepsilon) = 0$ and $\overline{R}(\varepsilon) = n - 1$

Observations

(i) All suffixes starting with a symbol which is lexicographically smaller than a will appear before aW

$$\Rightarrow \underline{R}(aW) \geq c\,(a)$$

(ii) Some suffixes in $[0, \underline{R}(W) - 1]$ are preceeded by an a

$$\Rightarrow \underline{R}(aW) \geq c(a) + O(a, \underline{R}(W) - 1)$$

Moreover any suffixes preceeding aW is of the type (i) or (ii) so

$$\Rightarrow \underline{R}(aW) = c(a) + O(a, \underline{R}(W) - 1)$$

(iii) The number of suffixes in $\underline{R}(W), \overline{R}(W)$ that are preceeded by a is

$$O(a, \overline{R}(W) - 1) - O(a, \underline{R}(W) - 1)$$

so

$$\overline{R}(W) = c(a) + O(a, \overline{R}(W))$$

1.2 Algorithm

We make a call like ExRecur(W, i, k, l) or InexRecur(W, i, z, k, l) where

- W is our query
- i = $|W| - 1$
- k, l is our SA-intervall so k = 0 and l = n - 1
- z is a maximum allowing differences (mismatches or gaps)

Algorithm 1 *Calculate exact recursion*

$Exrecur(W, i, k, l)$
if $i < 0$ **then**
  return $[k, l]$
end if
if $k \leq 0$ **then**
  $K \leftarrow C(W_i) + O(W_i, k - 1) + 1$
  $l \leftarrow C(W_i) + O(W_i, l)$
end if
return $ExRecur(W, i - 1, k, l)$

Algorithm 2 *Calculate inexact recursion*

```
InexRecur(W, i, z, k, l)
if z < 0 then
    return ∅
end if
if i < 0 then
    return [k, l]
end if;
I ← ∅
for σ ∈ {A, C, G, T} do
    k ← C(σ) + O(σ, k − 1) + 1
    l ← C(σ) + O(σ, l)
    if k ≤ 1 then
        I ← I ∪ InexRecur(W, i, z − 1, k, l)
        if σ = W_i then
            I ← I ∪ InexRecur(W, i − 1, z, k, l)
        else
            I ← I ∪ InexRecur(W, i − 1, z − 1, k, l)
        end if
    end if
end for
```