Computing the BWT and the LCP Array in Constant Space

Felipe A. Louza and Guilherme P. Telles

Institute of Computing, University of Campinas, Brazil

Motivation

- Suffix arrays may be used to solve many string matching and analysis problems.
- In larger scales, suffix arrays may be replaced by other indexes based on the Burrows-Wheeler transform, such as the FM-index.
- The LCP may be used together with the FM-index to solve some approximate string matching problems.

Our article

- Shows how to build the BWT and the LCP array in constant space and ${\cal O}(n^2).$
- Builds on the beautiful algorithm by Crochemore et al. to build the BWT in-place.
- Points out that other paths may be followed to perform the same construction, but this one is very simple.
- Presents some additional relations between the BWT and the LCP array.

Burrows-Wheeler transform

- ullet A reversible transformation that produces a permutation of a string S which tends to group the occurrences of a symbol into blocks.
- ullet To build the BWT sort all the cyclic rotations of S and get the last column.

i	cyclic rotations	sorted rotations	BWT
0	BANANA\$	\$BANANA	Α
1	\$BANANA	A\$BANAN	N
2	A\$BANAN	ANA\$BAN	N
3	NA\$BANA	ANANA\$B	В
4	ANA\$BAN	BANANA\$	\$
5	NANA\$BA	NA\$BANA	Α
6	ANANA\$B	NANA\$BA	Α

Suffix Array

• An array of integers that stores the order of every suffix of a string in lexicographic order.

	0	1	2	3	4	5	6
T	В	Α	N	Α	N	Α	\$

i	SA[i]	T[SA[i], n-1]
0	6	\$
1	5	A\$
2	3	ANA\$
3	1	ANANA\$
4	0	BANANA\$
5	4	NA\$
6	2	NANA\$

lcp, LCP array

- ullet The lcp between two strings S and T is the length of their largest common prefix.
- ullet The $L\!C\!P$ array stores the lcp value of consecutive suffixes in the suffix array.

i	SA[i]	LCP[i]	T[SA[i], n-1]
0	6	0	\$
1	5	0	A\$
2	3	1	ANA\$
3	1	3	ANANA\$
4	0	0	BANANA\$
5	4	0	NA\$
6	2	2	NANA\$

Relations

\overline{i}		F L	SA	LCP	BWT	T[SA[i], n-1]
0	BANANA\$	\$BANANA	6	0	Α	\$
1	\$BANANA	A\$BANAN	5	0	N	A\$
2	A\$BANAN	ANA\$BAN	3	1	N	ANA\$
3	NA\$BANA	ANANA\$B	1	3	В	ANANA\$
4	ANA\$BAN	BANANA\$	0	0	\$	BANANA\$
5	NANA\$BA	NA\$BANA	4	0	Α	NA\$
6	ANANA\$B	NANA\$BA	2	2	Α	NANA\$

- $\bullet \ L[i] = BWT[i] \ \text{and} \ F[i] = T[S\!A[i]].$
- The j-th symbol in the BWT is the symbol that precedes the j-th suffix (in order).
 - The 0-th symbol in the BWT precedes T[n-1].
- BWT[i] = T[SA[i] 1] if $SA[i] \neq 0$ or BWT[i] = \$ if SA[i] = 0.

Related results

	time	space (bits)	
In-place suffix sorting:	0(1)	$n\log n + n\log \sigma +$	
• ?	$O(n\log n)$	$O(\log n)$ SA + text + $O(1)$ words	
In-place BWT:	0(2)	$n\log\sigma + O(\log n)$	
• ?	$O(n^2)$	BWT + O(1) words	
In-place LCP:		$n\log n + n\log \sigma +$	
• ??	0	$O(\log n)$	
This work	$O(n^2)$	LCP + BWT + O(1) words	

- Proceeds by induction from the right to the left in T.
- Adds T[s], which corresponds to the suffix T_s .
- ullet The position of \$ gives the rank of T_s in the solution already constructed.

Let $BWT(T_s)$ be the BWT for T[s, n-1].

- \bullet proceed by induction from right to left in T
 - lacktriangle while maintaining the BWT of the current suffix T_s
 - ▶ Incremental step from $BWT(T_{s+1})$
 - * Replace \$ with new first character
 - * Insert the new suffix and its preceding character \$

<u>BWT</u>		sort	ed s	uffix	<u>es</u>	<u>BW</u>	<u>/T</u>	<u>501</u>	rted s	suffix	<u>(es</u>	
A N N	\$ A A	\$ N	A	\$		A N N		\$ N	Α	\$		
A \$	N N	A A	\$ N	Α	\$	\$ A	A N	N A	A \$	N	A	\$

Let $BWT(T_s)$ be the BWT for T[s, n-1].

- ullet proceed by induction from right to left in T
 - lacktriangle while maintaining the BWT of the current suffix T_s
 - ▶ Incremental step from $BWT(T_{s+1})$
 - * Replace \$ with new first character
 - * Insert the new suffix and its preceding character \$

<u>BWT</u>		<u>sort</u>	ed s	uffixe	<u>es</u>
Α	\$				
N	Α	\$			
N	Α	Ν	Α	\$	
Α	Ν	Α	\$		
\$	N	Α	Ν	Α	\$

<u>BWT</u>		sor	ted s	suffix	<u>ces</u>	
A N N	\$ A A	\$ N	А	\$		
\$	A	N	A	N	А	\$
A	Ν	Α	\$			
Λ	N.I	Λ.	N.I	Λ.	4	

Let $BWT(T_s)$ be the BWT for T[s, n-1].

- ullet proceed by induction from right to left in T
 - lacktriangle while maintaining the BWT of the current suffix T_s
 - ▶ Incremental step from $BWT(T_{s+1})$
 - * Replace \$ with new first character
 - * Insert the new suffix and its preceding character \$

<u>BWT</u>		sort	ed si	uffixe	<u>es</u>
A N	\$ A	\$			
N	A	Ŋ	Α	\$	
Δ	N	А	\$	Ф	
\$	N	Α	Ň	Α	\$

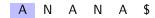
<u>BWT</u>		sor	ted s	suffix	<u>ces</u>	
A N N	\$ A A	\$ N	Α	\$		
\$, ,	N	Α	N	А	\$
Α	Ν	Α	\$			
۸	N.I	Λ.	N.I	Λ.	Φ.	

Let $BWT(T_s)$ be the BWT for T[s, n-1].

- ullet proceed by induction from right to left in T
 - lacktriangle while maintaining the BWT of the current suffix T_s
 - ▶ Incremental step from $BWT(T_{s+1})$
 - * Replace \$ with new first character
 - ★ Insert the new suffix and its preceding character \$

BWT sorted suffixes

A \$
N A \$
N A N A \$

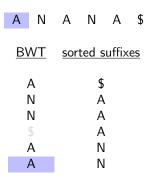


<u>BWT</u>		sor	ted s	suffix	<u>es</u>	
Α	\$					
N	Α	\$				
N	Α	Ν	Α	\$		
\$	Α	Ν	Α	Ν	Α	\$
Α	Ν	Α	\$			
Α	Ν	Α	Ν	Α	\$	

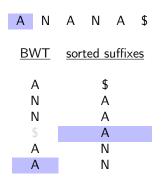
Ν

- The new suffix starts with the character that is being added.
- To determine its position, the algorithm counts:
 - smaller characters
 - preceding equal characters (rank)

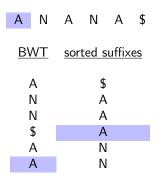
- The new suffix starts with the character that is being added.
- To determine its position, the algorithm counts:
 - smaller characters
 - preceding equal characters (rank)



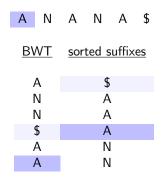
- The new suffix starts with the character that is being added.
- To determine its position, the algorithm counts:
 - smaller characters
 - preceding equal characters (rank)



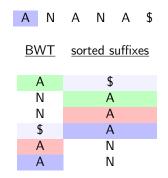
- The new suffix starts with the character that is being added.
- To determine its position, the algorithm counts:
 - smaller characters
 - preceding equal characters (rank)



- The new suffix starts with the character that is being added.
- To determine its position, the algorithm counts:
 - smaller characters
 - preceding equal characters (rank)



- The new suffix starts with the character that is being added.
- To determine its position, the algorithm counts:
 - smaller characters
 - preceding equal characters (rank)



BWT and LCP

- ullet Adding a character to the solution will demand computing the values of the LCP for T_s and its neighbors only.
- The first $L\!C\!P$ value involves T_s and the largest suffix T_a in $\{T_{s+1},\ldots,T_{n-1}\}$ that is smaller than T_s .
- The second LCP value involves T_s and the smallest suffix T_b in $\{T_{s+1}, \ldots, T_{n-1}\}$ that is larger than T_s .

BWT and LCP

2' Find the position p_{a+1} of the suffix T_{a+1} , such that suffix T_a has rank r in $BWT(T_{s+1})$, and compute:

$$\ell_a = \left\{ \begin{array}{ll} \mathit{RMQ}(p_{a+1},p) + 1 & \text{ if } T[p_{a+1}] = T[s] \\ 0 & \text{ otherwise.} \end{array} \right.$$

2" Find the position p_{b+1} of the suffix T_{b+1} , such that suffix T_b has rank r+1 in $BWT(T_{s+1})$, and compute:

$$\ell_b = \left\{ \begin{array}{ll} \mathit{RMQ}(p,p_{b+1}) + 1 & \text{ if } T[s] = T[p_{b+1}] \\ 0 & \text{ otherwise.} \end{array} \right.$$

4' Shift LCP[s+1,r] one position to the left, store ℓ_a in LCP[r] and if r+1 < n then store ℓ_b in LCP[r+1].

RMQ

- The RMQ (range minimum query) with respect to the *LCP* is the largest *lcp* value in an interval of the suffix array.
- $RMQ(i, j) = \min_{i < k < j} \{LCP[k]\}, \text{ for } 0 \le i < j < n.$
- Given a string T with length n and its $L\!C\!P$ array it is easy to see that $L\!C\!P(T_{S\!A[i]}, T_{S\!A[j]}) = R\!M\!Q(i,j).$

Let the current symbol be c = T[s].

BWT:

- - ightharpoonup replace \$ by c
- $\begin{array}{l} \textbf{2} \quad r = \text{rank of the} \\ \text{suffix } T_s \text{ using just} \\ \text{symbol } c \end{array}$

- $\bullet \ a = {\rm select} \ (r-1)^{th} \ {\rm suffix} \ {\rm in} \ BWT(T_s)$
 - $\blacktriangleright LCP[r] = min_{LCP}(a, p) + 1$ (if T[a] = T[p]
- $b = \text{select } (r+1)^{th} \text{ suffix in } BWT(T_s)$
 - $\qquad \qquad LCP[r+1] = min_{LCP}(p,b) + 1 \; (\text{if} \; T[p] = T[b]))$

		Α	Ν	Α	N	Α	\$	
		<u>BWT</u>		SOI	ted	suffix	<u>kes</u>	<u>LCP</u>
		Α	\$					0
b	\rightarrow	N	Α	\$				0
		N	Α	Ν	Α	\$		1
r	\rightarrow	\$	A	N	A	N	A	\$
а	\rightarrow	Α	Ν	Α	\$			0
р	\rightarrow	\$	Ν	Α	Ν	Α	\$	2

Let the current symbol be c = T[s].

BWT:

- - ightharpoonup replace \$ by c
- ② r = rank of the $\text{suffix } T_s \text{ using just}$ symbol c

- $\bullet \ a = {\rm select} \ (r-1)^{th} \ {\rm suffix \ in} \ BWT(T_s)$
 - ightharpoonup $LCP[r] = min_{LCP}(a,p) + 1$ (if T[a] = T[p]
- $b = \text{select } (r+1)^{th} \text{ suffix in } BWT(T_s)$
 - ► $LCP[r+1] = min_{LCP}(p, b) + 1$ (if T[p] = T[b])

		Α	N	Α	N	Α	\$	
		<u>BWT</u>		sor	ted :	suffix	<u>ces</u>	<u>LCP</u>
		Α	\$					0
b	\rightarrow	Ν	Α	\$				0
		Ν	Α	Ν	Α	\$		1
r	\rightarrow	\$	A	N	A	N	A	\$
а	\rightarrow	Α	Ν	Α	\$			0
p	\rightarrow	Α	N	Α	Ν	Α	\$	2

Let the current symbol be c = T[s].

BWT:

- - ightharpoonup replace \$ by c
- $\mathbf{0}$ r= rank of the suffix T_s using just symbol c

- $\bullet \ a = {\rm select} \ (r-1)^{th} \ {\rm suffix} \ {\rm in} \ BWT(T_s)$
- $\blacktriangleright \ LCP[r] = min_{LCP}(a,p) + 1 \ (\text{if} \ T[a] = T[p]$
- $b = \text{select } (r+1)^{th} \text{ suffix in } BWT(T_s)$
 - $\qquad \quad \blacktriangleright \ LCP[r+1] = min_{LCP}(p,b) + 1 \ (\text{if} \ T[p] = T[b])$

		Α	N	Α	Ν	Α	\$	
		<u>BWT</u>		SOI	ted s	suffix	<u>kes</u>	<u>LCP</u>
		Α	\$					0
b	\rightarrow	N	Ă	\$				0
		Ν	Α	Ν	Α	\$		1
r	\rightarrow	\$	Α	N	Α	Ν	Α	\$??
а	\rightarrow	Α	Ν	Α	\$			0
p	\rightarrow	Α	N	Α	Ν	Α	\$	2

Let the current symbol be c = T[s].

BWT:

- $p = position of $ in \\ BWT(T_{s+1})$
 - ightharpoonup replace \$ by c

- $a = \operatorname{select} (r-1)^{th} \operatorname{suffix} \operatorname{in} BWT(T_s)$
 - $LCP[r] = min_{LCP}(a, p) + 1$ (if T[a] = T[p])
- $b = \operatorname{select} (r+1)^{th} \operatorname{suffix} \operatorname{in} BWT(T_s)$

		Α	Ν	Α	N	Α	\$	
		<u>BWT</u>		sor	ted :	suffix	<u>kes</u>	<u>LCP</u>
		Α	\$					0
b	\rightarrow	N	Α	\$				0
		Ν	Α	N	Α	\$		1
r	\rightarrow	\$	Α	N	Α	Ν	Α	\$ 3
а	\rightarrow	Α	N	Α	\$			0
p	\rightarrow	Α	N	Α	Ν	Α	\$	2

Let the current symbol be c = T[s].

BWT:

- - ightharpoonup replace \$ by c

- $\bullet \ a = {\rm select} \ (r-1)^{th} \ {\rm suffix \ in} \ BWT(T_s)$
 - $LCP[r] = min_{LCP}(a, p) + 1$ (if T[a] = T[p])
- $b = \operatorname{select} (r+1)^{th} \operatorname{suffix} \operatorname{in} BWT(T_s)$
 - $\qquad LCP[r+1] = min_{LCP}(p,b) + 1 \text{ (if } T[p] = T[b] \text{)}$

		Α	Ν	Α	Ν	Α	\$	
		<u>BWT</u>		SOI	rted :	suffix	<u>kes</u>	<u>LCP</u>
		Α	\$					0
b	\rightarrow	N	A	\$				0
		N	Α	N	Α	\$		1
r	\rightarrow	\$	Α	N	Α	Ν	Α	\$ 3
а	\rightarrow	Α	N	Α	\$??
p	\rightarrow	Α	N	Α	Ν	Α	\$	2

Let the current symbol be c = T[s].

BWT:

- - ightharpoonup replace \$ by c

- $\bullet \ a = {\rm select} \ (r-1)^{th} \ {\rm suffix} \ {\rm in} \ BWT(T_s)$
 - ▶ $LCP[r] = min_{LCP}(a, p) + 1$ (if T[a] = T[p])
- $b = \operatorname{select} (r+1)^{th} \operatorname{suffix} \operatorname{in} BWT(T_s)$
 - $\qquad LCP[r+1] = min_{LCP}(p,b) + 1 \text{ (if } T[p] = T[b] \text{)}$

		Α	Ν	Α	Ν	Α	\$	
		<u>BWT</u>		SOI	rted	suffix	<u>(es</u>	<u>LCP</u>
		Α	\$					0
b	\rightarrow	N	A	\$				0
		N	Α	N	Α	\$		1
r	\rightarrow	\$	Α	N	Α	Ν	Α	\$ 3
а	\rightarrow	Α	N	Α	\$			0
p	\rightarrow	Α	N	Α	Ν	Α	\$	2

Final remarks

- Although ${\cal O}(n^2)$ these algorithms explore interesting relations among the structures.
- Only a constant number of additional variables is needed.
- The C code is quite short and clean.

```
1 void compute bwt lcp(unsigned char *T. int n. int *LCP) f
2 int i, p, r=1, s, p_a1, p_b1, l_a, l_b;
3 LCP[n-1] = LCP[n-2] = 0; // base cases
5 for (s=n-3: s>=0: s--) {
6
7
      /*steps 1 and 2*/
8
      p=r+1:
9
      for (i=s+1, r=0; T[i]!=END_MARKER; i++)
        if(T[i] <=T[s]) r++;
10
11
      for (; i < n; i++)
         if (T[i] <T[s]) r++;
13
14
      /*step 2'*/
      p_a1=p+s-1;
15
16
      1_a=LCP[p_a1+1];
      while (T[p_a1]!=T[s]) // RMQ function
18
        if (LCP[p_a1--]<1_a)
          1_a=LCP[p_a1+1];
19
20
      if (p_a1==s) 1_a=0:
21
      else 1_a++;
22
23
      /*step 2''*/
24
      p_b1=p+s+1:
25
      1_b=LCP[p_b1];
26
      while (T[p_b1]!=T[s] \&\& p_b1 < n) // RMQ function
27
       if (LCP[++p b1]<1 b)
28
          1_b=LCP[p_b1];
29
      if (p_b1==n) 1_b=0;
30
      else 1 b++:
31
32
      /*steps 3 and 4*/
33
      T[p+s]=T[s];
      for (i=s; i<s+r; i++) {
34
35
        T[i]=T[i+1];
36
        LCP[i]=LCP[i+1];
37
      T[s+r]=END_MARKER;
38
30
40
      /*step 4'*/
41
      LCP[s+r]=1_a:
42
      if (s+r+1<n) // If r+1 is not the last position
43
        LCP[s+r+1]=1_b:
44
     1
45 }
```

Thank you!

References I

The road not taken

IP-BWT

- 1 Find the position p of \$ in T[s+1,n-1]. Evaluating p-s gives the local rank of T_{s+1} that originally was starting at position s+1.
- 2 Find the local rank r of the suffix T_s using just symbol c=T[s]. To this end, sum the number of symbols in T[s+1,n-1] that are strictly smaller than c with the number of occurrences of c in T[s+1,p] and with s, obtaining r.
- 3 Store c into T[p], replacing \$.
- 4 Shift T[s+1,r] one position to the left. Write \$ at T[r].