The Burroughs Wheeler Transform: An Example

Cavan Reilly

October 28, 2019

Table of contents

Computing the BWT: an example

Some descriptions of the algorithm introduce a special character (e.g. $) that is lexicographically smaller than all other letters, then append this letter to the string before starting the rest of the transform.

This just guarantees that the first row of M is the original string so that one needn't store the row of M that is the original string.

To see how the BWT works, let's consider the consensus sequence just upstream of the transcription start site in humans (from Zhang 1998):

                    GCCACC

We will use the modification that introduces the special character.

BWT example

First add the special character to the end of the sequence we want to transform, denoted S with length N.

```
GCCACC$
```

then get all of the rotations:

```
$GCCACC
C$GCCAC
CC$GCCA
ACC$GCC
CACC$GC
CCACC$G
GCCACC$
```

BWT example

then sort them (i.e. alphabetize these sequences)

```
$GCCACC
ACC$GCC
C$GCCAC
CACC$GC
CC$GCCA
CCACC$G
GCCACC$
```

so the BWT is just the last column, typically denoted L, and we will refer to element i of this array as $L[i]$ for $i = 1, \ldots, N$ and ranges of elements via $L[1, \ldots, k]$.

$$L = \texttt{CCCCAG\$}$$

BWT example

Importantly one can invert this transformation of the string to obtain the original string.

This was first developed in the context of data compression, so the idea was to transform the data, then compress it (which can be done very efficiently after applying this transformation), then one can invert the transformation after uncompressing.

BWT example

Suppose we have the BWT of a genome and want to count the number of occurrences of some pattern, e.g. `AC`, our *query sequence*.

We can do this as follows.

First, create a table that for each letter has the count of the number of occurrences of letters that are lexically smaller than that letter. For us this is just

x	$	A	C	G
$C(x)$	0	1	2	6

BWT example

Next we need to be able to determine the number of times that a letter x occurs in $L[1, \ldots, k]$ for any k, which we will denote $O(x, k)$.

Note that all of these calculations can be done once and stored for future use.

Now search for the last letter of our query sequence, which is C, over the range of rows given by $[C(x) + 1, \ldots, C(x + 1)]$ where x is C so the range is [3,6].

Note that these are the rows of M that start with a C although we don't actually need to store all of M to determine this (it just follows from the alphabetization).

BWT example

Then process the second to last letter, which for our example is `A`.

To determine the range of rows of M to search we look at $[C(x) + O(x, s_1 - 1) + 1, \ldots, C(x) + O(x, s_2)]$ where x is `A` and s_1 and s_2 are the endpoints of the interval from the previous step. Now

$$O(\mathtt{A}, 3 - 1) = O(A, 2) = 0$$

and

$$O(\mathtt{A}, 6) = 1$$

so the interval is $[1 + 0 + 1, \ldots, 1 + 1] = [2, 2]$ then since this is the end of the query sequence we determine the number of occurrences by looking at the size of the range of this interval, which is here just 1.

BWT example

If the range becomes empty or if the lower end exceeds the upper end the query sequence is not present.

For example if we query on CCC and proceed as previously, the first step is the same (since this also ends in C), so then searching for C again we find

$$[C(\texttt{C}) + O(\texttt{C}, s_1 - 1) + 1, C(x) + O(\texttt{C}, s_2)]$$

but $O(C, 2) = 2$ and $O(C, 6) = 4$
so the interval is

BWT example

$[2 + 2 + 1, 2 + 4] = [5, 6]$
which has size 2 (as there are 2 instances of CC in the sequence),
so then proceed as in step 2 again to get

$$[C(\texttt{C}) + O(\texttt{C}, 5 - 1) + 1, C(\texttt{C}) + O(\texttt{C}, 6)]$$

which is

$$[2 + 4 + 1, 2 + 4] = [7, 6]$$

indicating that this sequence is not present.

BWT example

For aligning short sequences to a genome we need to know where the short segments align, hence we need to determine the location of the start of the sequence in the genome.

If the locations of some of the letters are known we can just trace back to those locations as follows.

First, note that we know the first column of the array M. We know this due to 2 facts:

1. the first column will be strictly alphabetized
2. every letter in the sequence is represented in the last column L.

BWT example

For our problem we have L=CCCCAG\$ and so the first column is just F=\$ACCCCG.

As our sequence is of length N, then if we knew the permutation of $1, \ldots, N$ that mapped F to L we would know the entire sequence (as we will see shortly).

That one can determine this mapping is the key ingredient in computing the inverse BWT. Call this map T where we have

$$F\big(T(i)\big) = L(i).$$

To find T, first consider M and a new array, call it M', that is obtained from M by taking the last element from each row and making it the first element but otherwise changing nothing.

Here is M':

```
C$GCCAC
CACC$GC
CC$GCCA
CCACC$G
ACC$GCC
GCCACC$
$GCCACC
```

Inverting the BWT

Note that every row of M has a corresponding row in M' and like M M' has all cyclic rotations of the original sequence S.

Also note that the relative order of the rows with the same first letter are the same in both arrays: rows 3, 4, 5 and 6 in M become rows 1, 2, 3 and 4 in M'. We can use this insight to determine the mapping from F to L, i.e. T.

To determine T, we proceed sequentially as follows. $L(1) = $ C and this is the first C in L so $T(1) = i$ where $F(i)$ is the first C in F so looking at F we see that $i = 3$.

BWT example

Then continue with this thinking: $T(2) = i$ and $L(2)$ is the second C in L, but then looking at F we see that the second C in F occurs at position 4, so that $T(2) = 4$.

Continuing in this fashion we find that

$$T = (3, 4, 5, 6, 2, 7, 1).$$

BWT example

We can then reconstruct the sequence in reverse order.

We know the last element of the sequence is the last element of the first row of M which is the first element of L, which is here C.

Then we proceed sequentially using the mapping T as follows

$$S(N - i) = L\big(T^{i}(1)\big)$$

where $T^{i}(x) = T^{i-1}\big(T(x)\big)$, i.e. repeatly apply the mapping T to itself.

BWT example

so

$$S(N-1) = L\big(T(1)\big) = L(3) = \texttt{C}$$

$$S(N-2) = L\big(T(3)\big) = L(5) = \texttt{A}$$

$$S(N-3) = L\big(T(5)\big) = L(2) = \texttt{C}$$

$$S(N-4) = L\big(T(2)\big) = L(4) = \texttt{C}$$

$$S(N-5) = L\big(T(4)\big) = L(6) = \texttt{G}$$

If we were just trying to find the location of a particular match, like AC we could just stop at that point rather than recover the entire sequence.

FM index

The FM index is a compressed version of L, C and O along with a compressed version of a set of indices with known locations in S.

This allows one to trace back to a known location.

So one could align short reads to a genome by first computing the BWT and compressing the results.

Unfortunately many of the reads would not align due to sequencing errors and polymorphisms.

Hence one needs to allow some discrepancies between the genome and the short reads.

While this is adjustable, the default specifications for the program Bowtie are that there is a maximum of 2 mismatches in the first 28 bases and a maximum total quality score of 70 for mismatches on the PHRED scale.