Implementing and Experimentally Evaluating the Complexity of the Burrows-Wheeler Transform

Gary Hoppenworth

Introduction:

Burrows and Wheeler's seminal paper [1] introduced the Burrows-Wheeler Transform and created a new wave of applications and theoretical research. The Burrows-Wheeler transform is a reversible transformation on a text that requires only constant additional space. The Burrows-Wheeler transform allows "free" compression of text since for large texts, the Burrows-Wheeler transform can aggregate repeated sequences in the text allowing us to use methods such as run length compression to decrease the space needed for the text. This is useful for certain applications in bioinformatics that contain highly repetitive text sequences that are very long. In these applications, speed and minimal storage are paramount -- this is where Burrows-Wheeler transform shines in applications. The Burrows-Wheeler transform takes linear time using suffix arrays in the implementation. In this experimental report, I implement the Burrows-Wheeler Transform and I experimentally evaluate the Burrows-Wheeler transform's time and space complexity.

Preliminaries and Definitions:

Given a Text T we perform the following:

1. Sort all circular shifts of T into lexicographical order.
2. Extract the last column of each shift.
3. Record the index of the shift corresponding to the original text T.

This yields the completed Burrows-Wheeler Transform, which can be completely reversed as necessary.

Evaluating the Performance of the Burrows-Wheeler Transform:

I wish to determine whether or not the Burrows-Wheeler Transform increases the amount of compression via runs of a random text T. I am both interested in a larger number of runs, as well as larger length runs. Therefore I will use the H-index of runs in text T and BWT(T). In this situation, I define the H-index to be the largest number H such that T contains H runs of length H or longer.

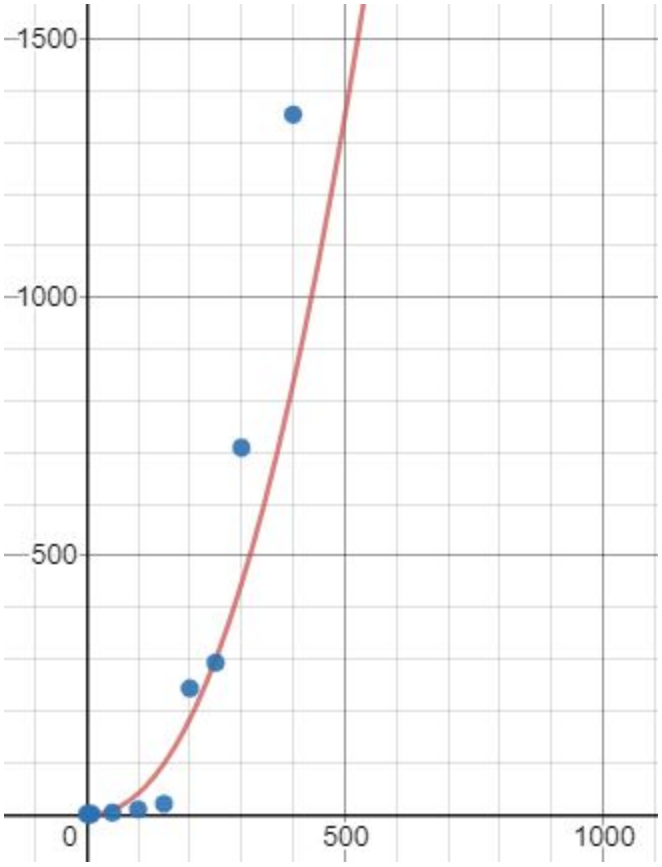
Implementation:

I implemented the Burrows-Wheeler Transform in python. I use the naive algorithm to perform the Burrows-Wheeler Transform, as opposed to the linear time algorithm that uses a suffix array. Likewise, I perform the naive algorithm to reverse the BWT as well. The theoretical complexity of my implementation should be O(n^2 log n) where n is the length of the text T. We will use randomly generated texts with alphabets of size five.

Experimental Results:

Burrows-Wheeler Transform Time Complexity:

I evaluated the speed of the Burrows-Wheeler Transform on input texts of length 100, 500, 1000, 5000, 10000, 15000, 20000, and 25000 words.



The x axis contains the length of the input text (in hundreds of symbols), and the y axis contains the run time of the Burrows-Wheeler Transform (in hundredths of a second).

Does the BWT really help compress random strings?

Size of the Text	H-Index of the Text	H-index of BWT of Text
100	2	2
500	3	3
1000	3	4
5000	4	4
10,000	5	5
15,000	5	5
20,000	5	5

Recall as stated above that the H-index is a measurement of the number of runs in the string. The H-index for T and BWT(T) is approximately the same, with BWT(T) performing better than T only for the text of size 1000. The H-index seems to indicate that the BWT doesn't actually allow for that much more compression in terms of number of runs and size of runs for random strings over an alphabet of size 5. However, perhaps a more fine-grained approach could yield greater insight into compression provided by the

Analysis:

Our experimental results suggest that the widespread popularity of the Burrows-Wheeler transform may not be well-deserved. The BWT does not increase the H-index of the text significantly, even for texts of size 20000. Our naive algorithm was relatively fast on small texts, however it began to slow dramatically for larger texts. On an input text of size 45,000, my laptop ran out of memory and could not finish the Burrows-Wheeler Transform.

Conclusion:

Further research should be done to verify whether or not the Burrows-Wheeler Transform grants any compression random input texts. Perhaps there are mathematical arguments that can be made that would “justify” the experimental results I observed. Additionally, experimental research should be performed on the run time of the efficient implementation of the Burrows-Wheeler Transform that is linear in the length of the input string T. The Burrows-Wheeler Transform is known to have better performance at compression for large T, so future research should further explore the performance of the transform on large input texts.

References:

[1] .Burrows, Michael, and David J. Wheeler. "A block-sorting lossless data compression algorithm." (1994).