BWT Computation Algorithms

Robert Bakarić
rbakaric@irb.hr
bakaric@evolbio.mpg.de
02.09.2015
LCP-1.0

Abstract

The BWT is used in many lossless data compression schemes, one of which is the Julian Seward's bzip2 program. It is usually computed by sorting all suffixes of a string S according to lexicographic order after which a sequence of characters at positions preceding sorted suffixes, is recorder. This sequence is known as Burrows-Wheeler transform.

Contents

1 Installation

The simplest way to compile this program is to:

1. Unpack the BWT package (`BWT-XXX.tar.gz`):

   ```
   tar -xvzf BWT-XXX.tar.gz
   ```

2. Change the current directory to `BWT-XXX`:

   ```
   cd BWT-XXX/
   ```

3. Build the program for your system :

   ```
   perl Makefile.PL
   ```

4. Compile the program:

   ```
   make
   ```

5. Test:

   ```
   make test
   ```

6. Install the program:

   ```
   (sudo) make install
   ```

2 Input files

Each BWT computation program takes a simple ASCII txt file and computes its transform. An example of the input file can be found in `./BWT-xxx/demo` and it should look like this:

```
Test:

thisisatesttotestthecorrectnessofLCPKasaialgorithm
```

3 Program options

I order to see program options type:

```
perl ./src/apps/BWT.pl -h
```

Expected output:

```
Usage:

-i input - ASCII file
-t terminating symbol
```

4 Functions and Modules

SA::SuffixArray module:

`new` : Constructor. Creates a new suffix array object.

(Ex.:my $sa = SA::SuffixArray→new();)

`Sort_Suffixes` : Function returns the lexicographically sorted array of indexes corresponding to starting positions of string suffixes. Function is a simple quicksort based sorting algorithm with O(n log n) worst case runtime behaviour.

(Ex.: my @suftab = $sa→Sort_Suffixes(array ⇒ $array);)

BWT::Simple module:

`new` : Constructor. Creates a new BWT object.

(Ex.:my $bwt = BWT::Simple→new();)

`BWT_encode` : Function requires a sorted suffix array (suftab) and a string (string) both as array references. As a result it returns the computed BWT array.

(Ex.: my $bwtencode = $bwt→BWT_encode(suftab ⇒ suftab, string ⇒ array);)

`BWT_decode` : Once encoded, BWT can be decoded back using BWT_decode function. The function requires BWT encod string and a string terminating character. (Ex.: my $bwtdecode = $bwt→BWT_decode(bwt ⇒ $bwtencode, terminator ⇒ $terminator);)

5 Example

A minimal example demonstrating the usage of BWT.pl demo program:

```
./src/apps/BWT.pl -i demo/Test

This is my BWT:
mLPfCsiKseehrnttoltttashrahtsgctrooaiiseeeoat$ictss

This is my BTW decode:
thisisatesttotestthecorrectnessofLCPKasaialgorithm$
```

5.1 Simple.pm

Adding the `use BWT::Simple` module file to your program will allow you to include all the functions described in section 4. A minimal example:

```
use BWT::Simple;
use SA::SuffixArray;

my $bwt1S = BWT::Simple->new();
my $sa = SA::SuffixArray->new();

# -----        Compute suffix array        ----- #
my @suftab = $sa->Sort_Suffixes(array => \@array);
```

```
# -----              BWT encode              ----- #
my $bwtencode = $bwt1S->BWT_encode(suftab => \@suftab,
                                   string => \@array);

# -----              BWT decode              ----- #
my $bwtdecode = $bwt1S->BWT_decode(bwt => $bwtencode,
                                   terminator => $terminator);
```

6 Acknowledgement

1. Adjeroh, D., Bell, T. and Mukherjee, A. 2008. The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching. Springer US.

7 Future work

Upon request!