LCP Computation Algorithms

Robert Bakarić
rbakaric@irb.hr
bakaric@evolbio.mpg.de
02.09.2015
LCP-0.01

Abstract

The longest common prefix (LCP) array is an auxiliary data structure to the suffix array. The array contains lengths of the longest common prefixes (LCPs) between all pairs of consecutive suffixes in a sorted suffix array. The algorithms presented here are implementations of: Kasais linear time LCP construction strategy [1], Karkkainen's linear PLCP, LCP based solution [2] and Manzini's space efficient BWT based strategy.

Contents

1 Installation

The simplest way to compile this program is to:

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

   ```
   tar -xvzf LCP-XXX.tar.gz
   ```

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

   ```
   cd LCP-XXX/
   ```

3. Build the program for your system :

   ```
   perl Makefile.PL
   ```

4. Compile the program:

   ```
   make
   ```

5. Make test:

   ```
   make test
   ```

6. Install the program:

   ```
   (sudo) make install
   ```

2 Input files

Each LCP computation program takes a simple ASCII text file and computes an array of indexes corresponding to lexicographically ordered text suffixes from which then longest common prefix array is computed. An example of the input file can be found in `./LCP-xxx/demo` and it should look like this:

```
Test:

thisisatesttotestthecorrectnessofLCPKasaialgorithm
```

3 Program options

I order to see program options type:

```
perl ./bin/LCP.pl -h
```

Expected output:

```
Usage:

-i input - ASCII file
-q quite - quite mode
-k Kasai = k , Karkkainine = kkk, Manzini = m
-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);)

LCP::Kasai module:

`new` : Constructor. Creates a new longest common prefix object.

(Ex.:my $lcp = LCP::Kasai→new();)

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

(Ex.: my $lcparray =$lcp→lcp(suftab ⇒ $suftab, string ⇒ $array);)

LCP::Karkkainen module:

`new` : Constructor. Creates a new longest common prefix object.

(Ex.:my $lcp = LCP::Karkkainen→new();)

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

(Ex.: my $lcparray =$lcp→lcp(suftab ⇒ $suftab, string ⇒ $array);)

`plcp` : Function requires a sorted suffix array (suftab) and a string (string) both as array references. As a result it returns the computed PLCP array (an array of LCP values as they appear in a string)

(Ex.: my $plcparray =$lcp→plcp(suftab ⇒ $suftab, string ⇒ $array);)

LCP::Manzini module:

`new` : Constructor. Creates a new longest common prefix object.

(Ex.:my $lcp = LCP::Manzini→new();)

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

(Ex.: my $lcparray =$lcp→lcp(suftab ⇒ $suftab, string ⇒ $array);)

5 Example

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

```
./bin/LCP.pl -i demo/Test -k k

0 51 $
0 35 CPKasaialgorithm$
0 37 Kasaialgorithm$
0 34 LCPKasaialgorithm$
0 36 PKasaialgorithm$
0 40 aialgorithm$
1 42 algorithm$
1 38 asaialgorithm$
1 7 atesttotestthecorrectnessofLCPKasaialgorithm$
0 21 correctnessofLCPKasaialgorithm$
1 26 ctnessofLCPKasaialgorithm$
0 20 ecorrectnessofLCPKasaialgorithm$
2 25 ectnessofLCPKasaialgorithm$
1 29 essofLCPKasaialgorithm$
2 15 estthecorrectnessofLCPKasaialgorithm$
4 9 esttotestthecorrectnessofLCPKasaialgorithm$
0 33 fLCPKasaialgorithm$
0 44 gorithm$
0 19 hecorrectnessofLCPKasaialgorithm$
1 2 hisisatesttotestthecorrectnessofLCPKasaialgorithm$
1 49 hm$
0 41 ialgorithm$
1 5 isatesttotestthecorrectnessofLCPKasaialgorithm$
2 3 isisatesttotestthecorrectnessofLCPKasaialgorithm$
1 47 ithm$
0 43 lgorithm$
0 50 m$
0 28 nessofLCPKasaialgorithm$
0 32 ofLCPKasaialgorithm$
1 45 orithm$
2 22 orrectnessofLCPKasaialgorithm$
1 13 otestthecorrectnessofLCPKasaialgorithm$
0 24 rectnessofLCPKasaialgorithm$
1 46 rithm$
1 23 rrectnessofLCPKasaialgorithm$
0 39 saialgorithm$
2 6 satesttotestthecorrectnessofLCPKasaialgorithm$
1 4 sisatesttotestthecorrectnessofLCPKasaialgorithm$
1 31 sofLCPKasaialgorithm$
1 30 ssofLCPKasaialgorithm$
1 16 stthecorrectnessofLCPKasaialgorithm$
3 10 sttotestthecorrectnessofLCPKasaialgorithm$
0 14 testthecorrectnessofLCPKasaialgorithm$
5 8 testtotestthecorrectnessofLCPKasaialgorithm$
1 18 thecorrectnessofLCPKasaialgorithm$
2 1 thisisatesttotestthecorrectnessofLCPKasaialgorithm$
2 48 thm$
1 27 tnessofLCPKasaialgorithm$
1 12 totestthecorrectnessofLCPKasaialgorithm$
1 17 tthecorrectnessofLCPKasaialgorithm$
```

```
2 11 ttotestthecorrectnessofLCPKasaialgorithm$
```

```
perl ./bin/LCP.pl -i demo/Test -k kkk

0 51 $
0 35 CPKasaialgorithm$
0 37 Kasaialgorithm$
0 34 LCPKasaialgorithm$
0 36 PKasaialgorithm$
0 40 aialgorithm$
1 42 algorithm$
1 38 asaialgorithm$
1 7 atesttotestthecorrectnessofLCPKasaialgorithm$
0 21 correctnessofLCPKasaialgorithm$
1 26 ctnessofLCPKasaialgorithm$
0 20 ecorrectnessofLCPKasaialgorithm$
2 25 ectnessofLCPKasaialgorithm$
1 29 essofLCPKasaialgorithm$
2 15 estthecorrectnessofLCPKasaialgorithm$
4 9 esttotestthecorrectnessofLCPKasaialgorithm$
0 33 fLCPKasaialgorithm$
0 44 gorithm$
0 19 hecorrectnessofLCPKasaialgorithm$
1 2 hisisatesttotestthecorrectnessofLCPKasaialgorithm$
1 49 hm$
0 41 ialgorithm$
1 5 isatesttotestthecorrectnessofLCPKasaialgorithm$
2 3 isisatesttotestthecorrectnessofLCPKasaialgorithm$
1 47 ithm$
0 43 lgorithm$
0 50 m$
0 28 nessofLCPKasaialgorithm$
0 32 ofLCPKasaialgorithm$
1 45 orithm$
2 22 orrectnessofLCPKasaialgorithm$
1 13 otestthecorrectnessofLCPKasaialgorithm$
0 24 rectnessofLCPKasaialgorithm$
1 46 rithm$
1 23 rrectnessofLCPKasaialgorithm$
0 39 saialgorithm$
2 6 satesttotestthecorrectnessofLCPKasaialgorithm$
1 4 sisatesttotestthecorrectnessofLCPKasaialgorithm$
1 31 sofLCPKasaialgorithm$
1 30 ssofLCPKasaialgorithm$
1 16 stthecorrectnessofLCPKasaialgorithm$
3 10 sttotestthecorrectnessofLCPKasaialgorithm$
0 14 testthecorrectnessofLCPKasaialgorithm$
5 8 testtotestthecorrectnessofLCPKasaialgorithm$
1 18 thecorrectnessofLCPKasaialgorithm$
2 1 thisisatesttotestthecorrectnessofLCPKasaialgorithm$
2 48 thm$
1 27 tnessofLCPKasaialgorithm$
1 12 totestthecorrectnessofLCPKasaialgorithm$
1 17 tthecorrectnessofLCPKasaialgorithm$
2 11 ttotestthecorrectnessofLCPKasaialgorithm$
```

5.1 Kasai.pm

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

```
use LCP::Kasai;
use SA:SuffixArray;


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

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


my $lcp = LCP::Kasai->new();


# -----         Compute lcp array          ----- #
my $lcparray =$lcp->lcp(suftab => \@suftab,
                                  string => \@array);
```

5.2 Karkkainen.pm

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

```
use LCP::Karkkainen;
use SA:SuffixArray;


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

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


my $lcp = LCP::Karkkainen->new();


# -----         Compute lcp array          ----- #
my $lcparray =$lcp->lcp(suftab => \@suftab,
                                           string => \@array);

my $plcparray =$lcp->plcp(suftab => \@suftab,
                                           string => \@array);
```

5.3 Manzini.pm

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

```
use LCP::Manzini;
use SA:SuffixArray;


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

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


my $lcp = LCP::Manzini->new();


# -----         Compute lcp array          ----- #
my $lcparray =$lcp->lcp(suftab => \@suftab,
                                  string => \@array);
```

6 Acknowledgement

1. Kasai, Toru and Lee, Gunho and Arimura, Hiroki and Arikawa, Setsuo and Park, Kunsoo, Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications, Lecture Notes in Computer Science, Combinatorial Pattern Matching, 2001.
2. Karkkainen, Juha and Manzini, Giovanni and Puglisi, SimonJ. Permuted Longest-Common-Prefix Array. Lecture Notes in Computer Science,Combinatorial Pattern Matching, 2009.
3. Manzini, G. and Ferragina, P. Engineering a Lightweight Suffix Array Construction Algorithm. 2002.

7 Future work