LCP Computation Algorithms

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

Abstract

The longest common prefix (LCP) array is an auxiliary data structure to the suffix array. The array containes 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] and Puglis's log linear PLCP, LCP based solution [2].

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. Install the program:

   ```
   (sudo) make install
   ```

2 Input files

Each LCP computation program takes a simple ASCII txt 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:

```
TestLCP:

thisisatesttotestthecorrectnessofLCPKasaialgorithm
```

3 Program options

I order to see program options type:

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

Expected output:

```
Usage:

-i input - ASCII file
-q quite - quite mode
-t terminating symbol
```

4 Functions and Modules

LCP::Kasai module:

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

(Ex.:my $lcp = LCP::Kasai→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 lagorithm with O(n log n) worst case runtime behaviour.

(Ex.: my @suftab = $lcp→_sort_suffixes(array ⇒ $array);)

`_kasai` : Function requires a sorted suffix array (suftab) and a string (string) both as array references. As a result it returns the computed LCP array (height - term used by Kasai et al.) and a rank array (an array invers to the suffix array)

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

5 Example

A minimal example demonstrating the usage of LCPKasai demo program:

```
perl ./src/apps/LCP.pl -i demo/TestLCP

#Hight:0 0 0 1 0 0 0 1 2 0 0 1 1 0 1 1

#Rank:15 5 8 11 7 10 1 13 3 12 14 4 6 9 2 0

#Position:15 6 14 8 11 1 12 4 2 13 5 3 9 7 10 0

0 16 $
0 7 atestfile$
0 15 e$
1 9 estfile$
0 12 file$
0 2 hisisatestfile$
0 13 ile$
1 5 isatestfile$
2 3 isisatestfile$
0 14 le$
0 6 satestfile$
1 4 sisatestfile$
1 10 stfile$
0 8 testfile$
1 11 tfile$
1 1 thisisatestfile$
```

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;

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


# -----        Compute suffix array        ----- #
my @suftab = $lcp->_sort_suffixes(array => \@array);

# -----         Compute lcp array          ----- #
my ($height,$sufinv) =$lcp->_kasai(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.

7 Future work