

---

# 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 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: Kasai's linear time LCP construction strategy [1] and Puglis's log linear PLCP, LCP based solution [2].

## **Contents**

<b>1</b>	<b>Installation</b>	<b>2</b>
<b>2</b>	<b>Input files</b>	<b>2</b>
<b>3</b>	<b>Program options</b>	<b>2</b>
<b>4</b>	<b>Functions and Modules</b>	<b>2</b>
<b>5</b>	<b>Example</b>	<b>3</b>
5.1	Kasai.pm . . . . .	6
5.2	Karkkainen.pm . . . . .	6
<b>6</b>	<b>Acknowledgement</b>	<b>7</b>
<b>7</b>	<b>Future work</b>	<b>7</b>

## 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:

Test:

```
thisisatesttotestthecorrectnessofLCPKasaialgorithm
```

## 3 Program options

In order to see program options type:

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

Expected output:

Usage:

```
-i input - ASCII file
-q quite - quite mode
-k Kasao = k , Karkkainine = kkk
-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();)

**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 inverses to the suffix array)

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

#### **LCP::Karkkainen module:**

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

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

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

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

## **5 Example**

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

```
./src/apps/LCP.pl -i demo/Test -k k
#Hight:0 0 0 0 0 1 1 1 0 1 0 2 1 2 4 0 0 0 1 1 0
1 2 1 0 0 0 0 1 2 1 0 1 1 0 2 1 1 1 3 0 5 1 2 2 1 1 1 2

#Rank:45 19 23 37 22 36 8 43 15 41 50 48 31 42 14 40
49 44 18 11 9 30 34 32 12 10 47 27 13 39 38 28 16 3
1 4 2 7 35 5 21 6 25 17 29 33 24 46 20 26 0

#Position:50 34 36 33 35 39 41 37 6 20 25 19 24 28
14 8 32 43 18 1 48 40 4 2 46 42 49 27 31 44 21 12 23
45 22 38 5 3 30 29 15 9 13 7 17 0 47 26 11 16 10

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 atestttotestthecorrectnessofLCPKasaialgorithm$
0 21 correctnessofLCPKasaialgorithm$
1 26 ctnessofLCPKasaialgorithm$
0 20 ecorrectnessofLCPKasaialgorithm$
2 25 ectnessofLCPKasaialgorithm$
1 29 essofLCPKasaialgorithm$
2 15 estthecorrectnessofLCPKasaialgorithm$
4 9 estttotestthecorrectnessofLCPKasaialgorithm$
0 33 fLCPKasaialgorithm$
0 44 gorithm$
0 19 hecorrectnessofLCPKasaialgorithm$
1 2 hisisatestttotestthecorrectnessofLCPKasaialgorithm$
1 49 hm$
0 41 ialgorithm$
1 5 isatestttotestthecorrectnessofLCPKasaialgorithm$
2 3 isisatestttotestthecorrectnessofLCPKasaialgorithm$
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 satestttotestthecorrectnessofLCPKasaialgorithm$
1 4 sisatestttotestthecorrectnessofLCPKasaialgorithm$
1 31 sofLCPKasaialgorithm$
1 30 ssofLCPKasaialgorithm$
1 16 stthecorrectnessofLCPKasaialgorithm$
3 10 stttotestthecorrectnessofLCPKasaialgorithm$
0 14 testthecorrectnessofLCPKasaialgorithm$
5 8 testttotestthecorrectnessofLCPKasaialgorithm$
1 18 thecorrectnessofLCPKasaialgorithm$
2 1 thisisatestttotestthecorrectnessofLCPKasaialgorithm$
2 48 thm$
1 27 tnessofLCPKasaialgorithm$
1 12 totestthecorrectnessofLCPKasaialgorithm$
1 17 tthecorrectnessofLCPKasaialgorithm$
2 11 ttotestthecorrectnessofLCPKasaialgorithm$

```

```

perl ./src/apps/LCP.pl -i demo/Test -k kkk
#Hight:0 0 0 0 0 0 1 1 1 0 1 0 2 1 2 4 0 0 0 1 1 0
1 2 1 0 0 0 1 2 1 0 1 1 0 2 1 1 1 3 0 5 1 2 2 1 1 1 2

PLCP: 2 1 2 1 1 2 1 5 4 3 2 1 1 0 2 1 1 1 0 0 0 2 1
0 2 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 2 1 0 0

```

```
#Position:50 34 36 33 35 39 41 37 6 20 25 19 24 28 14 8
32 43 18 1 48 40 4 2 46 42 49 27 31 44 21 12 23 45 22 38
5 3 30 29 15 9 13 7 17 0 47 26 11 16 10
```

```
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 essoofLCPKasaialgorithm$
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 ($hight,$rank) = $lcp->Kasai(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 ($lcparray,$plcparray) = $lcp->Karkkainen(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.

## 7 Future work