Sampled LCP Array

Jouni Sirén / CPM 2010

Outline

1. **Introduction**
2. Irreducible LCP Algorithm
3. Sampled LCP Array
4. Conclusions

Suffix Trees

- Important data structures in string processing and bioinformatics.
- In practice: 10 – 20 bytes / character with 32-bit pointers (Kurtz 1999).
- For large data sets, we want something much smaller but still efficient.

Space-Efficient Alternatives

SA	BWT	
12	I	$
11	P	I$
8	S	IPPI$
5	S	ISSIPPI$
2	M	ISSISSIPPI$
1	$	MISSISSIPPI$
10	P	PI$
9	I	PPI$
7	S	SIPPI$
4	S	SISSIPPI$
6	I	SSIPPI$
3	I	SSISSIPPI$

SA: $(\log n + \log \sigma)$ bits / char
BWT: $\log \sigma$ bits / char

Only support a part of ST functionality!

Compressed Suffix Trees

- Many proposals have three components:
  - Compressed suffix array (CSA)
  - Longest common prefix (LCP) array
  - Tree topology
- In practice: Cánovas, Navarro (SEA 2010)

Longest Common Prefix Array

SA	BWT	LCP	
12	I	0	$
11	P	0	I$
8	S	1	IPPI$
5	S	1	ISSIPPI$
2	M	4	ISSISSIPPI$
1	$	0	MISSISSIPPI$
10	P	0	PI$
9	I	1	PPI$
7	S	0	SIPPI$
4	S	2	SISSIPPI$
6	I	1	SSIPPI$
3	I	3	SSISSIPPI$

Longest Common Prefix Array

SA	BWT	LCP	
12	I	0	$
11	P	0	I$
8	S	1	IPPI$
5	S	1	ISSIPPI$
2	M	4	ISSISSIPPI$
1	$	0	MISSISSIPPI$
10	P	0	PI$
9	I	1	PPI$
7	S	0	SIPPI$
4	S	2	SISSIPPI$
6	I	1	SSIPPI$
3	I	3	SSISSIPPI$

Both preceded by 'S'

Previous suffixes are also adjacent.

Longest Common Prefix Array

SA	BWT	LCP	
12	I	0	$
11	P	0	I$
8	S	1	IPPI$
5	S	1	ISSIPPI$
2	M	4	ISSISSIPPI$
1	$	0	MISSISSIPPI$
10	P	0	PI$
9	I	1	PPI$
7	S	0	SIPPI$
4	S	2	SISSIPPI$
6	I	1	SSIPPI$
3	I	3	SSISSIPPI$

Suffix and left match preceded by same character.

Permuted LCP Array

SA	BWT	PLCP	
1	$	0	MISSISSIPPI$
2	M	4	ISSISSIPPI$
3	I	3	SSISSIPPI$
4	S	2	SISSIPPI$
5	S	1	ISSIPPI$
6	I	1	SSIPPI$
7	S	0	SIPPI$
8	S	1	IPPI$
9	I	1	PPI$
10	P	0	PI$
11	P	0	I$
12	I	0	$

LCP[x] = PLCP[SA[x]]

Compressing the LCP Array

- PLCP[i] + 2i is strictly increasing, and can be represented as a bit vector of length 2n.
    - Sadakane (2007)
- There are at most R runs of ones in the bit vector, so we can use run-length encoding.
    - Fischer, Mäkinen, Navarro (2009)

Outline

1. Introduction
2. **Irreducible LCP Algorithm**
3. Sampled LCP Array
4. Conclusions

Maximal and Minimal Values

SA	BWT	PLCP		MAX	MIN
1	$	0	MISSISSIPPI$	X	X
2	M	4	ISSISSIPPI$	X	
3	I	3	SSISSIPPI$		
4	S	2	SISSIPPI$		
5	S	1	ISSIPPI$		X
6	I	1	SSIPPI$	X	X
7	S	0	SIPPI$	X	X
8	S	1	IPPI$	X	X
9	I	1	PPI$	X	X
10	P	0	PI$	X	X
11	P	0	I$	X	X
12	I	0	$	X	X

Irreducible LCP Algorithm

- Kärkkäinen, Manzini, Puglisi (CPM 2009)
- Find and compute maximal values.
    - Their sum is at most $2n \log n$.
- Other values: $PLCP[i+1] = PLCP[i] - 1$
- PLCP construction: $O(n \log n)$ time, negligible working space.
- Requires the text in memory and SA on disk.
- How to use a CSA instead?

Compressed Suffix Arrays

- Function Ψ: $SA[\Psi(x)] = SA[x] + 1$
  - Scan CSA in text order: Start from $(i, x) = (i, SA^{-1}[i])$ and iterate with $(i+1, \Psi(x))$.
- Array C: $C[c]$ is the number of occurrences of characters in $\{ 0, \ldots, c-1 \}$.
  - Function Ψ is strictly increasing in every $\Psi_c = [C[c-1] + 1, C[c]]$.
- SA samples: Compute $SA[x] = i$ and $SA^{-1}[i] = x$ when i is a multiple of parameter d.
  - $SA[x] = SA[\Psi^k(x)] - k$.

Modified Algorithm

- Scan the CSA in text order, starting from $(1, SA^{-1}[1])$.
- If PLCP[i] is maximal, compute it by scanning the CSA from (i, x) and its left match (j, x-1).
- Otherwise PLCP[i] = PLCP[i-1] – 1.
- Time complexity is $O(t_\psi \, n \log n)$, and working space is still negligible.

Identifying Maximal Values

- Suffix i of rank x and its left match j of rank $x – 1$.
- PLCP[i] is non-maximal iff BWT[x] = BWT[x-1].
- PLCP[i] is non-maximal iff the ranks of suffixes $i – 1$ and $j – 1$ are in the same Ψ_c.
- Let y be the rank of suffix $i – 1$. PLCP[i] is non-maximal iff $y – 1$ and y are in the same Ψ_c, and $\Psi(y-1) = x – 1$.

In Practice

Name	Size	Time	MB / s
English	400 MB	1688 s	0.24
Fiwiki	400 MB	327 s	1.22
DNA	385 MB	3475 s	0.11
Yeast	409 MB	576 s	0.71

Outline

Using PLCP with a CSA

- We want to retrieve $LCP[x] = PLCP[SA[x]]$.
- To get $SA[x]$, we iterate $\Psi^k(x)$ for $k = 0, 1, \ldots$, until we find a sampled $SA[\Psi^k(x)]$ value.
- Then $LCP[x] = PLCP[SA[\Psi^k(x)] - k]$.
- Time complexity is $O(d\ t_\Psi)$.
  - Typical values of d are 16 to 64 for regular texts, and 128 to 512 for highly repetitive texts.
  - t_Ψ is roughly 1 μs in practice.

A Faster Approach

- Cánovas, Navarro (SEA 2010)
- LCP values are usually small, so we can compress the LCP array directly.
- For regular texts, we get 6 – 8 bits / character.
    - PLCP is at most 2 bits / character.
- LCP[x] can be retrieved in less than 1 μs.

Sampled LCP Array (1 / 2)

- PLCP[i] = PLCP[i+1] + 1, if i is non-minimal.
- PLCP[i] = PLCP[i+k] + k, if i + k is the next minimal value.

Maximal and Minimal Values

SA	BWT	PLCP		MAX	MIN
1	$	0	MISSISSIPPI$	X	X
2	M	4	ISSISSIPPI$	X	
3	I	3	SSISSIPPI$		
4	S	2	SISSIPPI$		
5	S	1	ISSIPPI$		X
6	I	1	SSIPPI$	X	X
7	S	0	SIPPI$	X	X
8	S	1	IPPI$	X	X
9	I	1	PPI$	X	X
10	P	0	PI$	X	X
11	P	0	I$	X	X
12	I	0	$	X	X

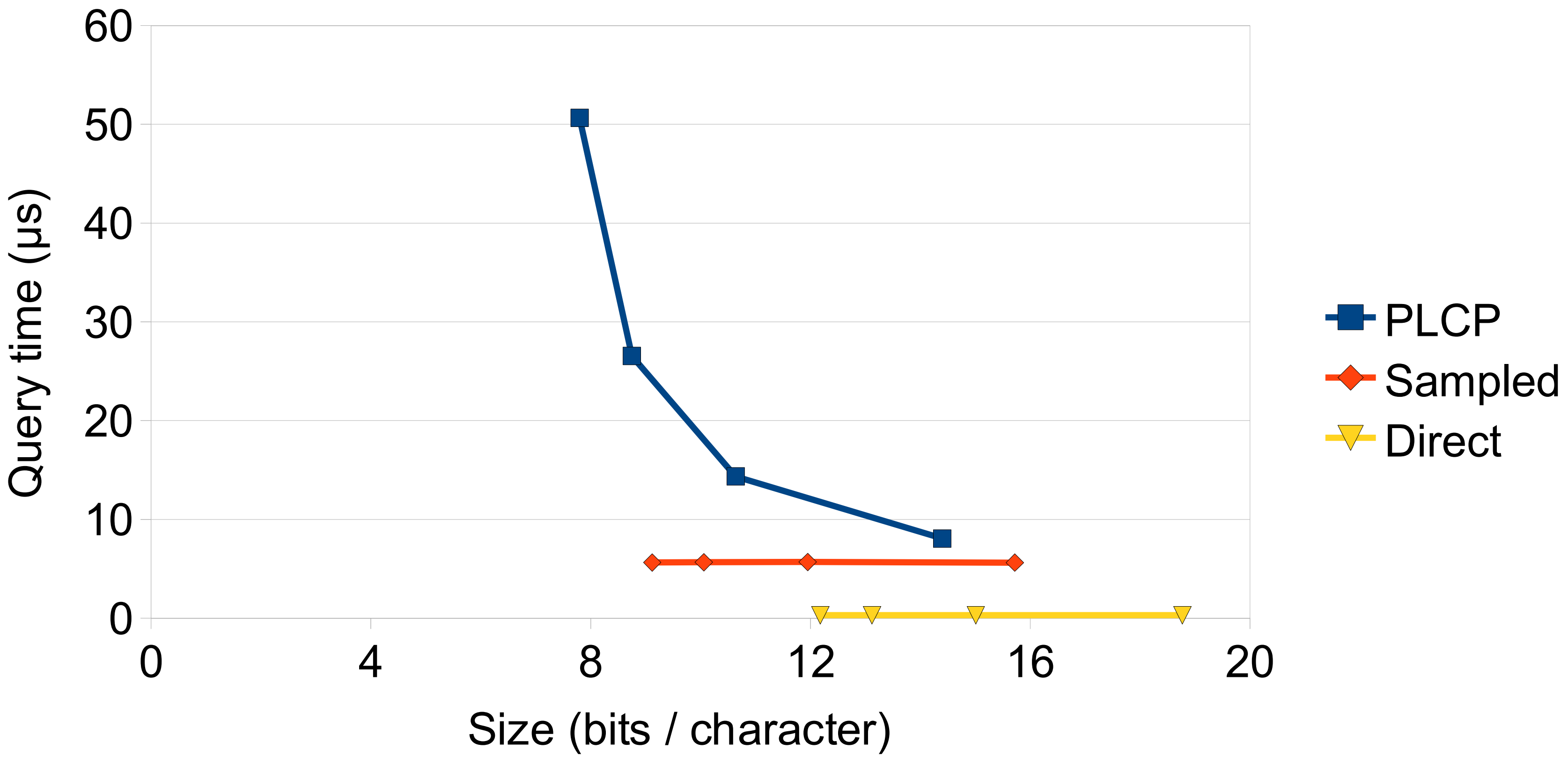
Sampled LCP Array (1 / 2)

- PLCP[i] = PLCP[i+1] + 1, if i is non-minimal.
- PLCP[i] = PLCP[i+k] + k, if i + k is the next minimal value.
- We store the R minimal PLCP values in SA order.
  - Additional samples may be needed for performance.
- To get LCP[x], we iterate $\Psi^k(x)$ for k = 0, 1, …, until we find a sampled LCP[$\Psi^k(x)$] value.
- Then LCP[x] = LCP[$\Psi^k(x)$] + k.

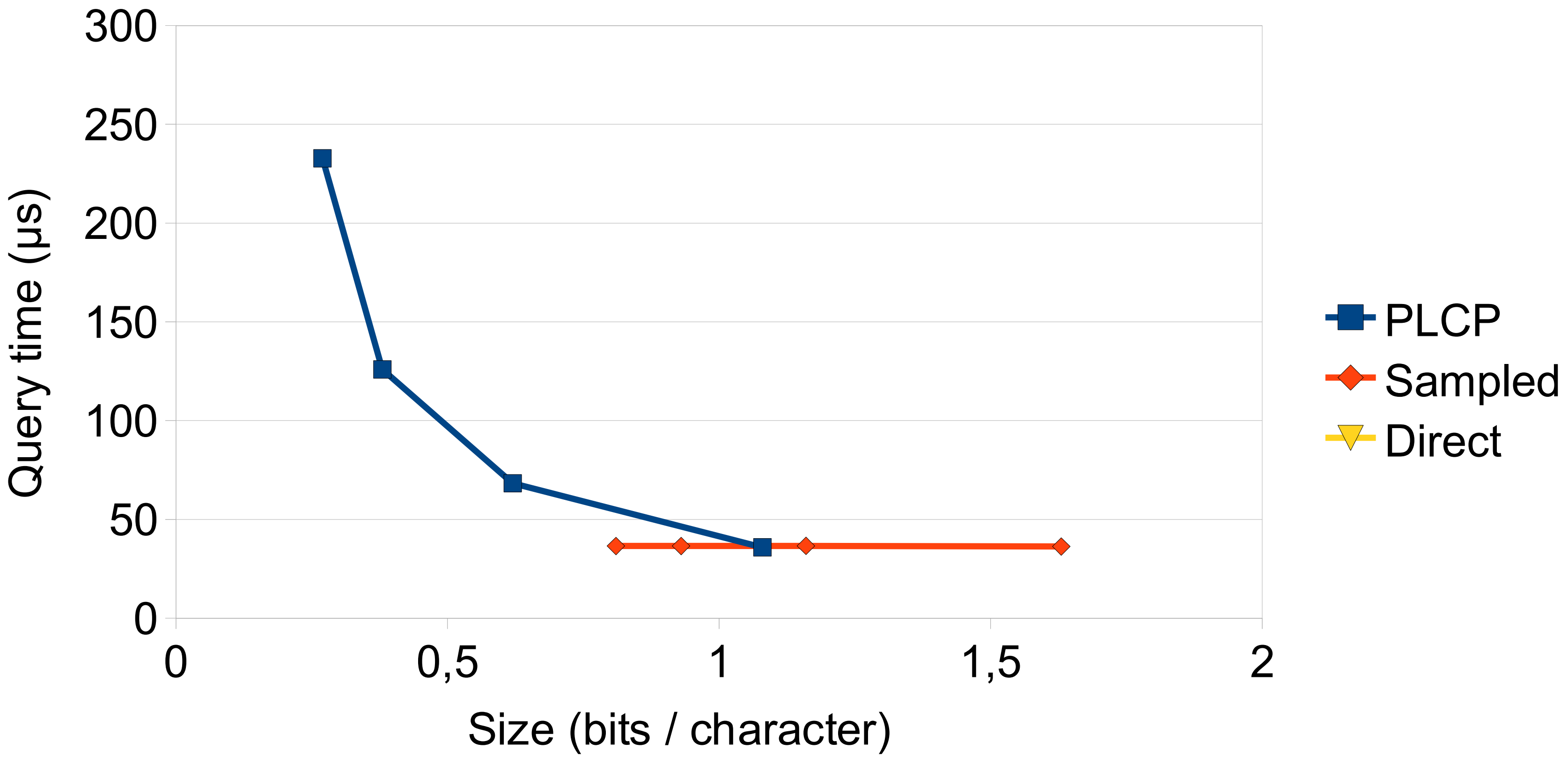
Sampled LCP Array (2 / 2)

- The size of the minimal samples scales with R.
  - Somewhat larger in practice than a run-length encoded PLCP.
- In regular texts, 20 – 40 % of the LCP values are minimal.
  - LCP values can be retrieved several times faster than by using a PLCP array.
- On highly repetitive texts, the performance is determined by the number of extra samples.

Experiments: English



Experiments: Fiwiki



Outline

1. Introduction
2. Irreducible LCP Algorithm
3. Sampled LCP Array
4. **Conclusions**

Conclusions

- We can construct the (P)LCP array directly from a CSA with little extra working space.
- Construction speed is similar to direct CSA construction.
- The LCP array can be sampled in a similar way as the suffix array.
- On regular texts, the sampled LCP array offers better time/space trade-offs than the PLCP array.