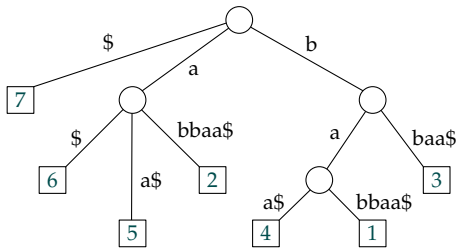
Distributed Compressed Suffix Arrays

Jouni Sirén

University of Helsinki, Department of Computer Science

1 Suffix Trees



Suffix trees are used for indexing and analyzing textual data. Their drawback is their size: usually 10–20 times the size of the text.

2 Space-Efficient Alternatives

	SA	BWT
$	7	a
a$	6	a
aa$	5	b
abbaa$	2	b
baa$	4	b
babbaa$	1	$
bbaa$	3	a

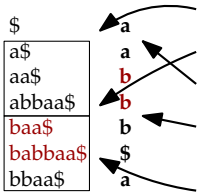
BWT takes the same space as the text.

Suffix aa$ starts at position 5 and after character **b**.

SA is usually 4–8 times the size of the text.

Suffix arrays (SA) support the core functionality of suffix trees. They can be simulated by using the *Burrows-Wheeler Transform (BWT)*.

3 Using the BWT for Pattern Matching



Suffixes starting with **a** at positions 2 to 4.

Suffixes starting with **b** at positions 5 to 7.

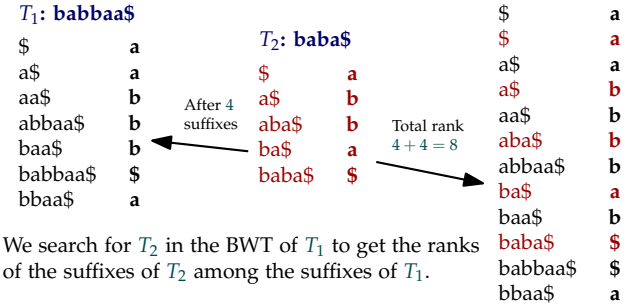
No **b**s before the suffixes starting with **a**.

2 **b**s before the suffixes starting with **b**.

Suffixes starting with **ba** at positions $5+0=5$ to $5+2-1=6$.

Suffix array values and text substrings can be extracted from the BWT efficiently by *sampling* some SA values.

4 Merging BWTs



We search for T_2 in the BWT of T_1 to get the ranks of the suffixes of T_2 among the suffixes of T_1.

5 The Compressed Suffix Array (CSA)

We use compressed *bit vectors* to encode the BWT.

BWT	**a**	**a**	**a**	**b**	**b**	**b**	**b**	**a**	**b**	**$**	**$**	**a**
Ψ_a	1	1	1	0	0	0	0	1	0	0	0	1
Ψ_b	0	0	0	1	1	1	1	0	1	0	0	0

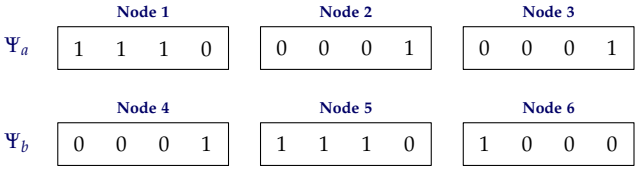
With *run-length encoded* bit vectors, the size of the CSA is bounded by the number of edit operations required to construct the texts. On the Finnish language Wikipedia with full version history, 42 GB of text is compressed to a 2.1 GB index.

6 CSA Construction

```
buildIndexForTexts(T_1, T_2, ..., T_p)
    A ← buildCSA(T_1)              # Any algorithm will do
    for i = 2 to p do
        B ← buildCSA(T_i)
        I ← sort(search(A, T_i))   # Ranks of suffixes of T_i in A
        A ← merge(A, B, I)         # Merge BWTs
    return A
```

The algorithm is reasonably fast and very space-efficient, making it possible to construct CSAs for much larger collections than before.

7 Distributed Implementation



Network latency becomes the bottleneck for individual queries. Massive parallelism is required for good performance.

References

Veli Mäkinen, Gonzalo Navarro, Jouni Sirén, and Niko Välimäki: **Storage and Retrieval of Individual Genomes**. In *RECOMB 2009*, Springer LNCS 5541, pp. 121–137, 2009. Extended version accepted to *Journal of Computational Biology*.

Jouni Sirén: **Compressed Suffix Arrays for Massive Data**. In *SPIRE 2009*, Springer LNCS 5721, pp. 63–74, 2009. Source code available at `http://www.cs.helsinki.fi/group/suds/rlcsa/`.