Computing Patterns in Strings III: Intrinsic Patterns

Bill Smyth[1,2,3]

[1]Algorithms Research Group, Department of Computing & Software
McMaster University, Hamilton, Ontario, Canada
email: smyth@mcmaster.ca

[2]Digital Ecosystems & Business Intelligence Institute
Curtin University, Perth, Western Australia
email: b.smyth@curtin.edu.au

[3]Department of Computer Science
King's College London, UK

DEBII 2008

Outline

Outline

Outline

Outline

Outline

Abstract

In this, the last of three talks on computing patterns in strings, we discuss intrinsic patterns — those that occur in all strings, and that in particular are key data structures for the computation of specific and intrinsic patterns. We focus on the suffix tree (ST), the suffix array (SA), and the longest common prefix array (LCP) – structures whose construction and application have been developed and refined over 40 years by hundreds of researchers. These are just examples — there are many others.

One of the exciting aspects of this dynamic research area is the likelihood that 40 years in the future new, more powerful and more flexible data structures will surely have been discovered and put to use.

Outline

Applications

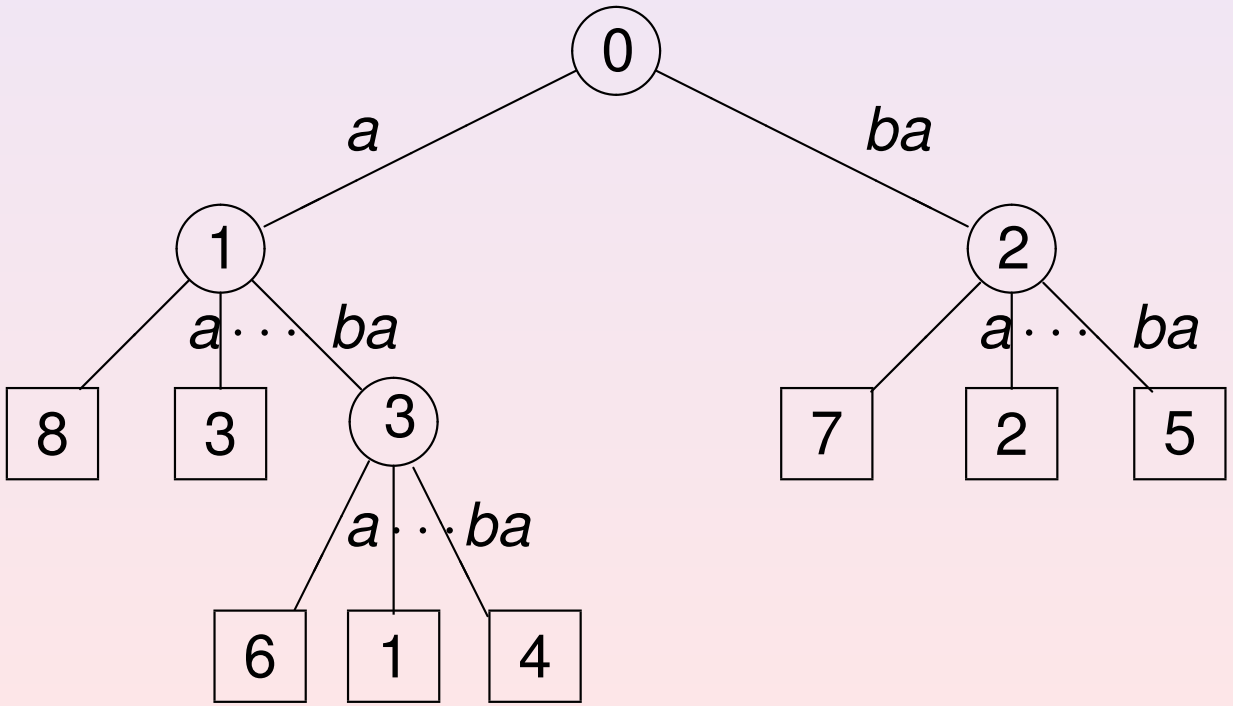
ST/SA/LCP are basic data structures for

- pattern-matching;
- computing repeats;
- computing repetitions;
- LZ decomposiition.

Outline

The Tree

	1	2	3	4	5	6	7	8
$\boldsymbol{f} =$	a	b	a	a	b	a	b	a



ST Checklist

- leaf nodes in lexorder
- internal nodes give lcp
- $O(n \log \alpha)$ construction time, where α is the alphabet size
- $O(n)$ space
- for construction and search: data structure at each node
- pattern-matching in time proportional to pattern-length m
- essential for repeats, repetitions, LZ

Marvellous!

BUT ... too much space!!

ST Space

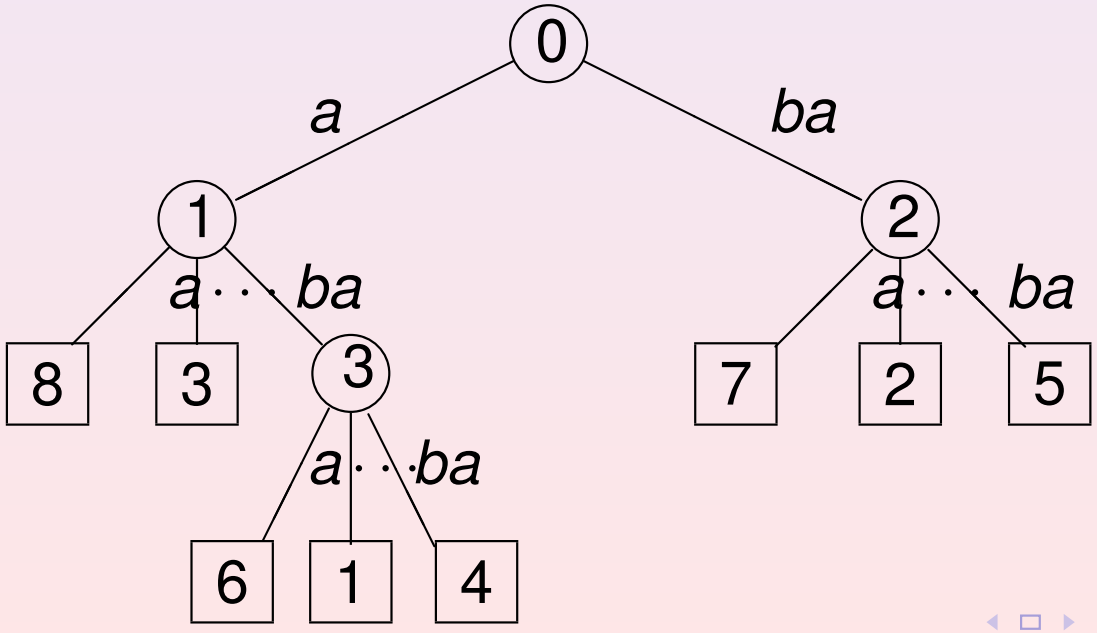
Usually each element of a string $\boldsymbol{x}$ requires one byte, often less (DNA two bits) — thus altogether n bytes. Even with Unicode, $2n$ bytes is usually enough.

- Every ST node is an integer: altogether about $8n$ bytes.
- Every edge in ST needs a pointer: at least $4n$ bytes.
- At each internal node, perhaps an array of letters or a search tree: αn bytes.

ST storage will generally be at least $15n$ bytes, often as much as $40n$ bytes or more. When $n = 10^{10}$, this is serious!

SA & LCP

	1	2	3	4	5	6	7	8
$x =$	a	b	a	a	b	a	b	a
$\mathrm{SA}_x =$	8	3	6	1	4	7	2	5
$\mathrm{lcp}_x =$	$-$	1	1	3	3	0	2	2



SA/LCP Advantages

- Each requires $4n$ bytes — if both are required, $8n$ bytes.
- SA can be computed quickly using only the space for $\boldsymbol{x}$ and SA itself.
- LCP can be computed using about $6n$ bytes, including $\boldsymbol{x}$ and LCP.
- SA/LCP can do everything ST can do, as fast or faster.

Outline

A Little Chronology

1973 ST invented; first algorithm published.

1985 "Myriad Virtues" paper published; several ST construction algorithms exist.

1990 SA/LCP arrays invented; inefficient algorithms for their construction proposed.

1999 First fast SA construction algorithm proposed.

2001 Fast LCP construction algorithm proposed, but $13n$ bytes of space required.

2003 Three linear-time algorithms for SA construction, but all are slow and require much space!

2004 Algorithms proposed that use SAs more efficiently than STs.

2007 Fast algorithms found to compute repeats/repetitions using SAs.

2007 More than 20 SA construction algorithms proposed.

2008 First SA construction algorithm discovered that is linear, fast and lightweight.

2008 Fast LCP algorithm propsed using only $6n$ bytes.

Looking Ahead ...

"We stand on the shoulders of giants."

Isaac Newton