Computing Periodicities in Strings — A New Approach

Bill Smyth[*†]

The most efficient methods for computing repetitions or repeats in a string $\boldsymbol{x} = \boldsymbol{x}[1..n]$ all depend on the prior computation of a suffix tree/array ST$\boldsymbol{x}$/SA$\boldsymbol{x}$. Although these data structures can be computed in asymptotic $\Theta(n)$ time, nevertheless in practice they involve significant overhead, both in time and space. Since the number of repetitions/repeats in $\boldsymbol{x}$ can be reported in a way that is at most linear in string length, it should therefore be possible to devise less roundabout means of computing repetitions/repeats that take advantage of their infrequent occurrence. This talk provides background for these ideas and explores the possibilities for more efficient computation of periodicities in strings.

* Algorithms Research Group, Department of Computing & Software, McMaster University
† Department of Computing, Curtin University

Why are Periodicities Interesting?

- Often long sections of DNA are copied, exactly or approximately, from one section of the genome to another; it is important to identify these copies and their context in a gene or chromosome.

- Many data compression algorithms depend on identifying repeating sections of text that are either long or frequent or both; these can be coded into shorter substrings that allow the text to be compressed.

- Repeating substrings, exact or approximate, may be of interest in decryption.

- Repeating motifs/phrases, exact or approximate, are studied by musicologists.

Kinds of Periodicity

In this talk, we confine ourselves to substrings that repeat **exactly**:

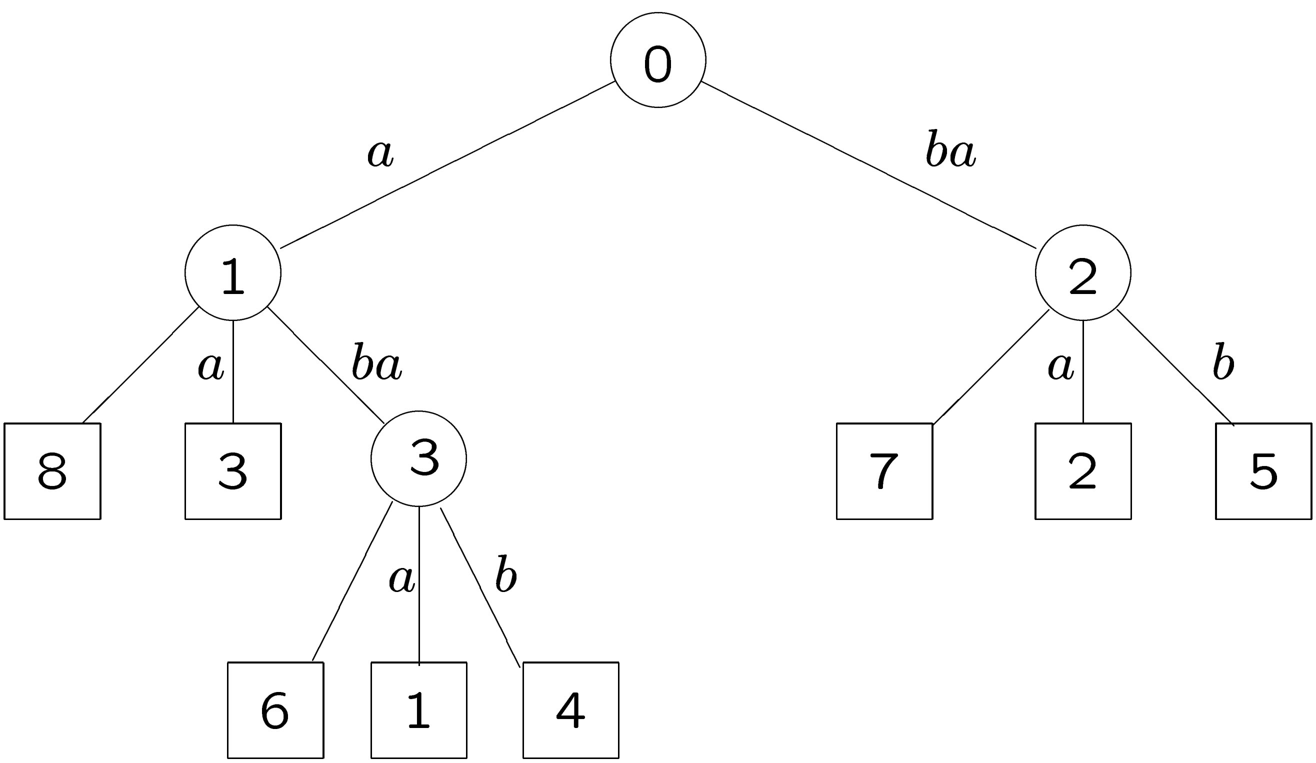
- **repetitions** (adjacent repeating substrings);
- **runs** ("super-repetitions");
- **repeats** (repeating substrings, not necessarily adjacent).

Computing approximate repetitions is much harder: the best algorithm is a stringological *tour de force* that requires $O(n^2 \log n)$ time.

Even exact repetitions require a lot of work; all the algorithms use **suffix trees/arrays**.

What is a Suffix Tree/Array?

$$
\begin{array}{rcccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\boldsymbol{x} = & a & b & a & a & b & a & b & a \\
\mathsf{SA}_{\boldsymbol{x}} = & 8 & 3 & 6 & 1 & 4 & 7 & 2 & 5 \\
\mathsf{lcp}_{\boldsymbol{x}} = & - & 1 & 1 & 3 & 3 & 0 & 2 & 2
\end{array}
$$



Repetitions

Suppose a string $\boldsymbol{x} = \boldsymbol{x}[1..n]$ is given:

repetition:	a substring $\boldsymbol{x}[i..i{+}pe{-}1] = \boldsymbol{u}^e$, $\lvert\boldsymbol{u}\rvert = p$ and $e \geq 2$.
$\boldsymbol{u}^e$ **irreducible**:	$\boldsymbol{u}$ itself is not a repetition.
$\boldsymbol{u}^e$ **maximal**:	neither $\boldsymbol{x}[i{-}p..i{-}1]$ nor $\boldsymbol{x}[i{+}pe..i{+}p(e{+}1){-}1] = \boldsymbol{u}$.

All repetitions discussed are both irreducible and maximal; they are fully specified by the triple (i, p, e).

generator:	$\boldsymbol{u}$
period:	p
exponent:	e

$$\begin{array}{cccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boldsymbol{x} = & a & b & a & a & b & a & b & a \end{array}$$

Repetitions $(1, 3, 2) = (aba)^2$, $(3, 1, 2) = a^2$, $(4, 2, 2) = (ab)^2$, and $(5, 2, 2) = (ba)^2$, all **squares**.

<u>Repetitions</u> (continued)

A naïve reporting of all the squares in a string would require $\Theta(n^2)$ time in the worst case — reporting all squares in $\boldsymbol{x} = a^6$ necessitates $\lfloor 6^2/4 \rfloor$ outputs:

$$\boldsymbol{x}[i]^2, 1 \le i \le 5;\ \ \boldsymbol{x}[i..i{+}1]^2, 1 \le i \le 3;\ \ \boldsymbol{x}[1..3]^2.$$

There are three "classical" $O(n \log n)$ algorithms [Crochemore (1981), Apostolico & Preparata (1983), Main & Lorentz (1984)] for computing all repetitions in the (i,p,e) encoding. [C81] & [AP83] essentially use suffix trees, [ML84] is divide-&-conquer. All are (in a sense) asymptotically optimal because the Fibostring $\boldsymbol{f_K}$

$$\boldsymbol{f_0} = b,\ \boldsymbol{f_1} = a;\ \ \boldsymbol{f_k} = \boldsymbol{f_{k-1}} \boldsymbol{f_{k-2}},\ k = 2, 3, \ldots, K$$

actually contains $\Theta\Big(|\boldsymbol{f_K}| \log |\boldsymbol{f_K}|\Big)$ repetitions.

Runs

Consider

$$\boldsymbol{x} = \cdots b \ \ \overset{i}{a}baabaab\ b \cdots$$

We report repetitions $(i, 3, 2)$, $(i+1, 3, 2)$, $(i+2, 3, 2)$. But $(i, 3, 2)$ implies the other two because $(aba)^2ab$ is followed by ab, a prefix of the generator aba.

Given a (maximal, irreducible) repetition (i, p, e):

- (i, p, e) is **left-extendible** (LE) if $(i-1, p, 2)$ is a square; otherwise NLE.
- The **tail** is the greatest integer t satisfying $\forall\ j \in 0..t$, $(i+j, p, e)$ is a repetition.

Then a **run** (**maximal periodicity**) [Main 1989] is a 4-tuple (i, p, e, t) where (i, p, e) is an NLE repetition of tail t.

<u>Runs</u> (continued)

Let $\rho(n)$ be the maximum number of runs that can occur in any string of length n. Then [Kolpakov & Kucherov (2000)]

$$\rho(n) \leq k_1 n - k_2 \sqrt{n} \log_2 n,$$

where k_1 & k_2 are universal positive constants of unknown size. K&K show that all runs in $\boldsymbol{x}$ can be computed in linear time (on an **indexed** (integer) alphabet):

- compute the suffix tree $T_{\boldsymbol{x}}$ [Farach (1997)];
- compute the LZ-factorization [Lempel-Ziv (1976)];
- compute the leftmost runs [Main (1989)];
- compute the remaining runs [K&K (2000)].

Repeats

A **repeat** in $\boldsymbol{x}$ is a tuple

$$M_{\boldsymbol{x},\boldsymbol{u}} = (p; i_1, i_2, \ldots, i_e),$$

where $e \geq 2$, $1 \leq i_1 < i_2 < \cdots < i_e \leq n$, and
$\boldsymbol{u} = \boldsymbol{x}[i_1..i_1{+}p{-}1] = \boldsymbol{x}[i_2..i_2{+}p{-}1] = \cdots = \boldsymbol{x}[i_e..i_e{+}p{-}1]$,
with **generator** $\boldsymbol{u}$, **period** p, **exponent** e.

Note that possibly, for some $j \in 1..e{-}1$, $i_{j+1} - i_j = p$ (repetition) or $i_{j+1} - i_j < p$ (overlap).

- $M_{\boldsymbol{x},\boldsymbol{u}}$ is **maximal** if for every

  $$i \in 1..n \text{ and } i \notin \{i_1, i_2, \ldots, i_e\},$$

  we are assured that $\boldsymbol{x}[i..i{+}p{-}1] \neq \boldsymbol{u}$.
- $M_{\boldsymbol{x},\boldsymbol{u}}$ is **left-extendible** (LE) if

  $$(p; i_1{-}1, i_2{-}1, \ldots, i_e{-}1)$$

  is a repeat.
- $M_{\boldsymbol{x},\boldsymbol{u}}$ is **right-extendible** (RE) if

  $$(p; i_1{+}1, i_2{+}1, \ldots, i_e{+}1)$$

  is a repeat.

BUT:

- maybe $k_1 = 10^{10}$ — the K&K proof is non-constructive;
- K&K provide convincing experimental evidence that in fact $\rho(n) < n$;
- the algorithm is complicated and not space-efficient (though better using suffix arrays).

Recall:

If $\sigma(n)$ is the maximum number of *distinct* squares that can occur in a string of length n, then [Fraenkel & Simpson (1998), Ilie (2005)]: $\sigma(n) < 2n$.

So here is the problem: $\sigma(n) \leq \rho(n) <$???

Hope:

By resolving the fundamental theoretical problem, we will (finally) understand periodicity better and therefore be able to design a simple direct all-runs algorithm.

In order that $\rho(n) > n$, it is necessary that two runs (squares) occur at some positions i — we therefore suppose that two squares occur at i and seek to restrict the squares that can occur in a neighbourhood of i. There seems to be only one result of this kind:

Lemma 1 *[Lothaire (2002)] Let $\boldsymbol{u}^2$ be a repetition, and suppose $\boldsymbol{w} \neq \boldsymbol{u}^k$ for any $k \geq 1$. If $\boldsymbol{u}^2$ is a prefix of $\boldsymbol{w}^2$, in turn a proper prefix of $\boldsymbol{v}^2$, then $w \leq v - u$.*

(We use $\boldsymbol{x}$ for the string, x for its length.)

Definition 2 *A square* $\boldsymbol{u}^2$ *is said to be* **irreducible** *if* $\boldsymbol{u}$ *is not a repetition.*

Definition 3 *A square* $\boldsymbol{u}^2$ *is said to be* **regular** *if no prefix of* $\boldsymbol{u}$ *is a square.*

Definition 4 *A square* $\boldsymbol{u}^2$ *is said to be* **minimal** *if no proper prefix of* $\boldsymbol{u}^2$ *is a square.*

Lemma 5 *If* $\boldsymbol{u}^2$ *is minimal, then* $\boldsymbol{u}^2$ *is regular; if* $\boldsymbol{u}^2$ *is regular, then* $\boldsymbol{u}^2$ *is irreducible.*

Lemma 6 *If* $\boldsymbol{v}^2$ *is irreducible with regular proper prefix* $\boldsymbol{u}^2$*, then*

$$v > \max\{u+1, 3u/2\}.$$

Lemma 7 *If* $\boldsymbol{x} = \boldsymbol{v}^2$ *is irreducible with regular proper prefix* $\boldsymbol{u}^2$*,* $v < 2u$*, then*

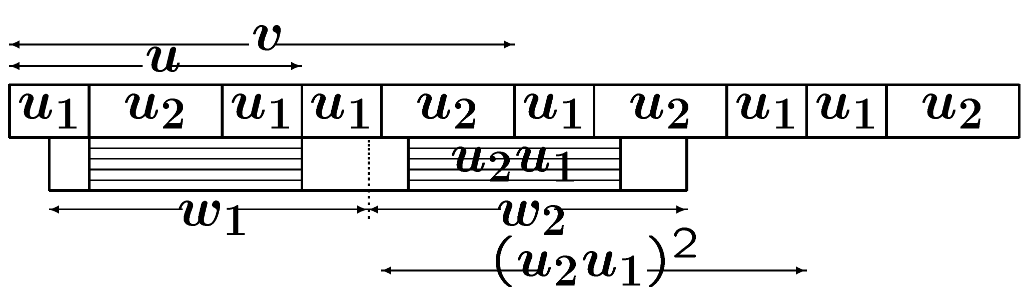
$$\boldsymbol{x} = \boldsymbol{u_1 u_2 u_1 u_1 u_2 u_1 u_2 u_1 u_1 u_2},$$

where $u_1 = 2u-v, u_2 = 2v-3u$.

Pièce de Résistance:

Lemma 8 (NPL) *If $\boldsymbol{x}$ has regular prefix $\boldsymbol{u}^2$ and irreducible prefix $\boldsymbol{v}^2$, $u < v < 2u$, then for every $w \in u+1..v-1$ and for every $k \in 0..v-u-1$, $\boldsymbol{x}[k+1..k+2w]$ is not a square.*

Case I (easy case: k small):



Notes

- Lemma 8 extends in an obvious way to runs.
- Lemma 8 applies only trivially to the cases $u = 1$ and $u = 2$: for $u = 1$, $v \geq 3 > 2u$, while for $u = 2$, $v \geq 5 > 2u$, contrary to the requirement of the lemmas that $v < 2u$.
- For all $u \geq 3$, the hypothesis of the lemma can be satisfied — for example, if $\boldsymbol{u} = aba$ of length 3, $\boldsymbol{v}$ may be $abaab$ of length $5 < 2 \times 3$.
- Thus Lemma 8 can be thought of as restricting the occurrences of squares when the second square at some position is small.
- We have extended to cases where $w \in v-u+1..u-1$.
- Our next project is to apply the NPL!