A note on the structure of run-maximal strings

A. Baker, A. Deza, and F. Franek

Department of Computing and Software
McMaster University, Hamilton, Ontario

African StringMasters, Stellenbosch University,
Stellenbosch, South Africa 27-31 July, 2009

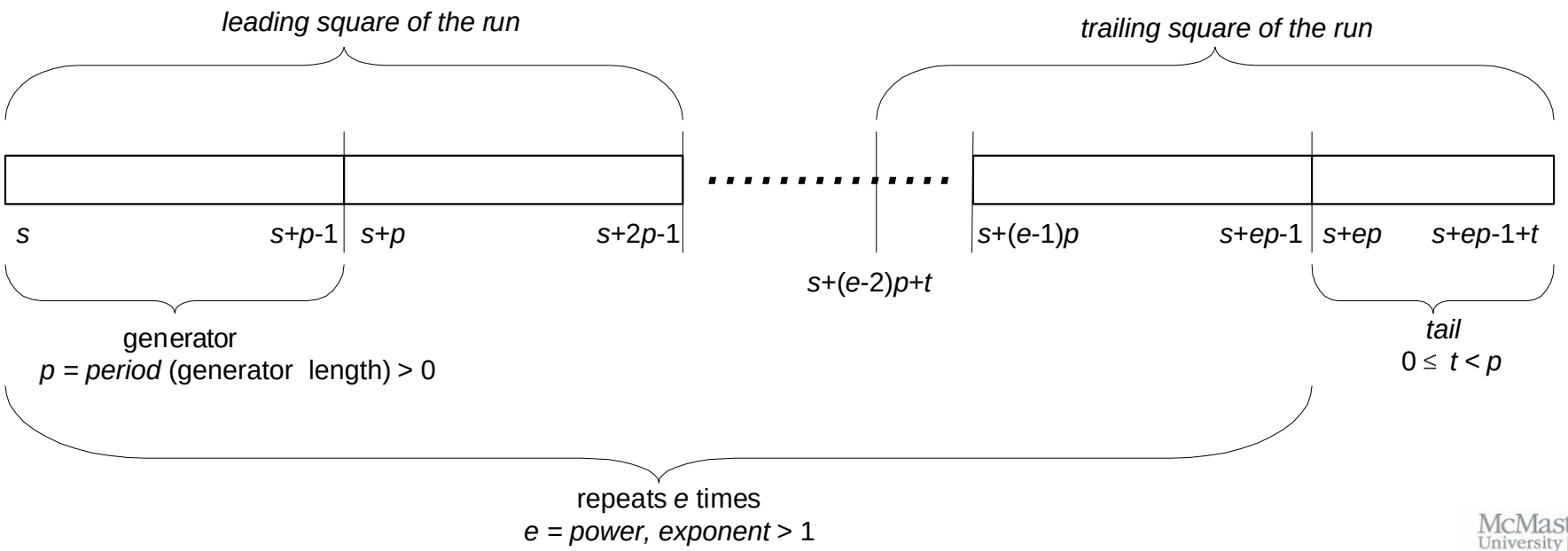
McMaster University

Outline

McMaster University

Background

A run, a maximal fractional repetition in a string was conceptually introduced by *Main* in 1989. The term was coined by *Iliopoulos*, *Moore*, and *Smyth* in 1997.



Though there may be at most $O(n \log n)$ repetitions in a string (*Crochemore* 1981), it was hoped that the more concise notation of runs will eliminate the need to list all repetitions.

In 2000, *Kolpakov* and *Kucherov* showed that there are at most $O(n)$ runs in a string.

Several authors (*Kolpakov* and *Kucherov*, *Smyth* et al.) formulated several conjectures on the nature of the maximum number of runs in a string, based on computational results (*Kolpakov* and *Kucherov* up to length 32, *Franek* and *Smyth* up to length 35).

McMaster University

$$r(x) = \#\text{ of runs in a string } x$$

$$\rho(n) = \max\{r(x) : |x| = n\}$$

Conjectures:

1. $\rho(n) \leq n$
2. $\rho(n+1) \leq \rho(n)+2$
3. for any n, there is a binary cube-free string x of length n so that $\rho(n) = r(x)$.

McMaster University

- Upper bound:
  1. $\rho(n) \leq 6.3n$, then improved to $3.44n$, *Rytter* 2006
  2. $\rho(n) \leq 1.6n$ *Crochemore* and *Ilie* 2008
  3. The current value stands at $\rho(n) \leq 1.029n$
     `http://www.csd.uwo.ca/~ilie/runs.html`

- Upper bound for microruns (runs with a bounded period):
  1. $\rho_{\leq 9}(n) < n$, *Crochemore* and *Ilie* 2008
  2. $\rho_{\leq 9}(n) < 0.971n$, *Giraud* 2008
  3. $\rho_{\leq p}(n) < n$, $p = 9, 10, 11$ *Franek* and *Holub* 2008
  4. $\rho_{\leq 60}(n) < 0.93n$, *Ilie* 2009

McMaster University

- Lower bound (asymptotic):
  1. $\rho(n) \geq 0.927n$, *Franek*, *Simpson*, and *Smyth* 2003, *Franek* and *Yang* 2008
  2. $\rho(n) \geq 0.944565n$, *Matsubara et al.* 2008
  3. The current value stands at $\rho(n) \geq 0.944575n$ by *Puglisi* and *Simpson*

     `http://www.shino.ecei.tohoku.ac.jp/runs/`

McMaster University

Structure of run-maximal strings

The objective is to describe in structural or combinatorial terms strings that exhibit the maximum number of runs.

In essence, a *run-maximal string has an R-cover, possibly a single “weak point", and satisfies a density condition*.

This structure immediately yields:

- a highly effective compression scheme for run-maximal strings, and
- a way to reduce dramatically the search space for run-maximal strings.

McMaster University

Usually, we encode a repetition as (s, p, e), where s is the starting position, p is the period, and e is the exponent. Alternatively, we can encode it as (s, p, d) where d is the ending position of the repetition; the mutual transformations are $d = s+ep-1$ and $e = (d-s+1) \;/\; p$

Usually, we can encode a run as (s, p, e, t) where s is the starting position, p is the period, e is the exponent, and t is the tail. Alternatively, we can encode it as (s, p, d) where d is the ending position of the run; the mutual transformations are $d = s+ep-1+t$ and $e = (d-s+1) \;/\; p, \quad t = (d-s+1) \;\%\; p$

McMaster University

Definition (R-cover)

Let x be a string. $\{ R_i = (s_i, p_i, d_i) \mid 1 \leq i \leq m \}$ is an *R-cover* of x if

a) Each R_i is a left-maximal square in x, i.e. either $R_i = x[0..k]$ or $R_i = x[v..k]$ and $x[v-1] \neq x[\frac{v+k-1}{2}]$.

b) For every R_i and R_{i+1}, $1 \leq i < m$, $s_{i+1} \leq d_i < d_{i+1}$.

c) For every R_i and R_{i+1}, $1 \leq i < m$, their overlap $x[s_{i+1}..d_i]$ is maximal such that (b) holds.

d) For any run R' in x and its leading square R, there is an $1 \leq i \leq m$ so that $R \subseteq R_i$.

e) $x = \bigcup_{1 \leq i \leq m} R_i$.

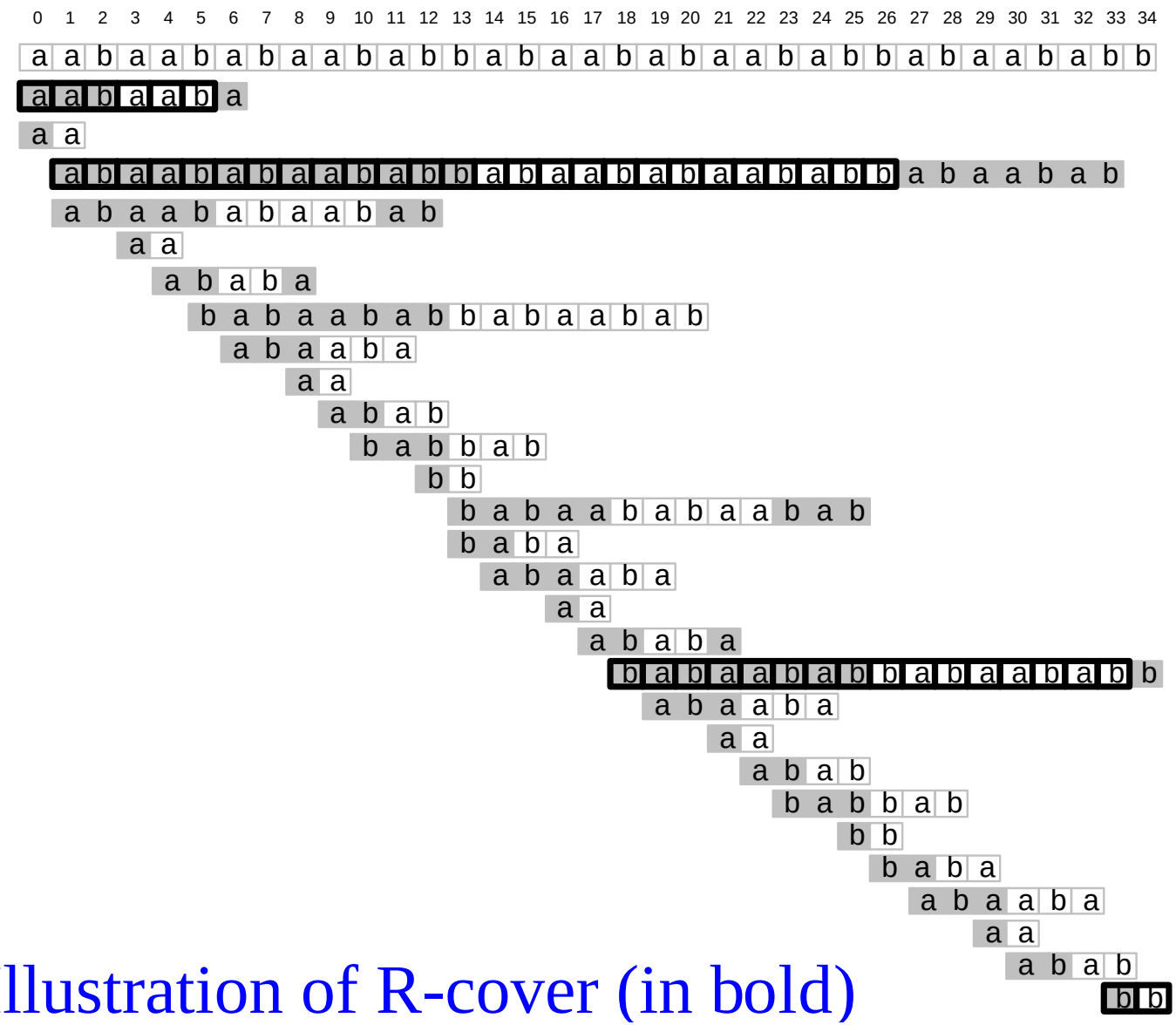


Illustration of R-cover (in bold)

Theorem (Structure of run-maximal strings)

Let $x = x[0..n]$ be a run-maximal string. Then either

- *x has an R-cover, or*
- *there are a string y that has an R-cover and a letter w so that $x = wy$, or*
- *there are a string y that has an R-cover and a letter w so that $x = yw$, or*
- *there are strings y_1 and y_2 that have R-covers and a letter w so that $x = y_1 w y_2$.*

Moreover, x satisfies the density condition: for any $k < n$, removing $x[0..k-1]$ destroys at least $\rho(k)$ runs.

McMaster University

Proof of the theorem

Lemma

Let x and y be strings. Then there is a transformation τ of the alphabet of y, so that $|\tau(y)| = |y|$, $r(\tau(y)) = r(y)$, and $r\Big(x \,||\, \tau(y)\Big) \geq r(x) + r(\tau(y)) = r(x) + r(y)$.

Proof.

Let $\mathcal{A}$ be the alphabet of x, and let $\mathcal{B} = \{b_1, \cdots, b_k\}$ be the alphabet of y. Take $\mathcal{C} = \{c_1, \cdots, c_k\}$ so that $\mathcal{A} \cap \mathcal{C} = \emptyset$. Define $\tau(b_i) = c_i$. Now, no runs can get “glued" together. □

McMaster University

Definition (weak point)

Let $x = x[0..n]$ be a string. A position i is a *weak point* if $r(x[0..i-1]) + r(x[i+1..n]) = r(x[0..n])$ ("removing" the position does not destroy any run).

Lemma

A run-maximal string $x = x[0..n]$ has at most one weak point.

Proof.

By contradiction assume two different weak points, i and j. By the previous lemma, there are transformations τ and θ so that
$r\Big(x[0..i-1]) \,||\, \tau(x[i+1..j-1]) \,||\, \theta(\tau(x[j+1..n]))\Big) \geq$
$r(x[0..i-1]) + r(x[i+1..j-1]) + r(x[j+1..n]) = r(x[0..n])$.
Let $a \neq x[0]$, let
$y = aa \,||\, x[0..i-1] \,||\, \tau(x[i+1..j-1]) \,||\, \theta(\tau(x[j+1..n]))$. Then
$|y| = |x|$ and
$r(y) \geq 1 + r(x[0..i-1]) + r(\tau(x[i+1..j-1])) + r(\theta(\tau(x[j+1..n]))) \geq$
$1 + r(x) > r(x)$, a contradiction with x being run-maximal. □

Lemma

Let $x = x[0..n]$ have no weak point. Then x has an R-cover.

Proof.

0 is not a weak point, so there must be a run destroyed by 0. Take all such runs, take the one with the largest period, and set R_0 to its leading square.
We follow by induction.
Assume to have $\{R_i : r \leq k\}$. Let D be the ending of R_k. If $D = n$, we are done.
If $D < n$, then $D+1$ is not a weak point, so there is a run destroyed by $D+1$. Among all runs destroyed by $D+1$, choose the ones with the leftmost start, among them choose the one with the largest period. Set R_{k+1} to its leading square. □

Now we can conclude the proof of the main theorem. A run-maximal string x has at most one weak point. The weak point-free segments have R-covers.

The density condition is straightforward: if $x[0..k-1]$ destroys fewer than $\rho(k)$ runs, we can replace $x[0..k-1]$ with a run-maximal string y for $k-1$, but then $r(y \mid\mid x[k..n]) > r(x)$ contradicting run-maximality of x.

Illustrations

For illustration, we used large (hopefully run-maximal) strings from `http://www.shino.ecei.tohoku.ac.jp/runs/` by *Matsubara et al.*

First we used the string of length 125 characters. Its R-cover given as [*starting position*, *period*, *ending position*] follows.

[0,24,47]
[16,32,79]
[48,37,121]
[109,8,124]

McMaster University

Then we used the string of length 139632 characters:

[0,34227,68453]
[21754,45341,112435]
[67095,34227,135548]
[113749,11114,135976]
[127159,4781,136720]
[131940,3609,139157]
[136862,1172,139205]
[138273,505,139282]
[138778,380,139537]
[139158,218,139593]
[139424,93,139609]
[139554,32,139617]
[139594,13,139619]
[139610,8,139625]
[139618,5,139627]
[139624,3,139629]
[139630,1,139631]

McMaster University

Then we used the string of length 184973 characters:

[0,79568,159135]
[113749,25837,165422]
[139586,19504,178593]
[159090,11114,181317]
[172500,4781,182061]
[177281,3609,184498]
[182203,1172,184546]
[183614,505,184623]
[184119,380,184878]
[184499,218,184934]
[184765,93,184950]
[184895,32,184958]
[184935,13,184960]
[184951,8,184966]
[184959,5,184968]
[184965,3,184970]
[184968,2,184971]
[184971,1,184972]

McMaster University

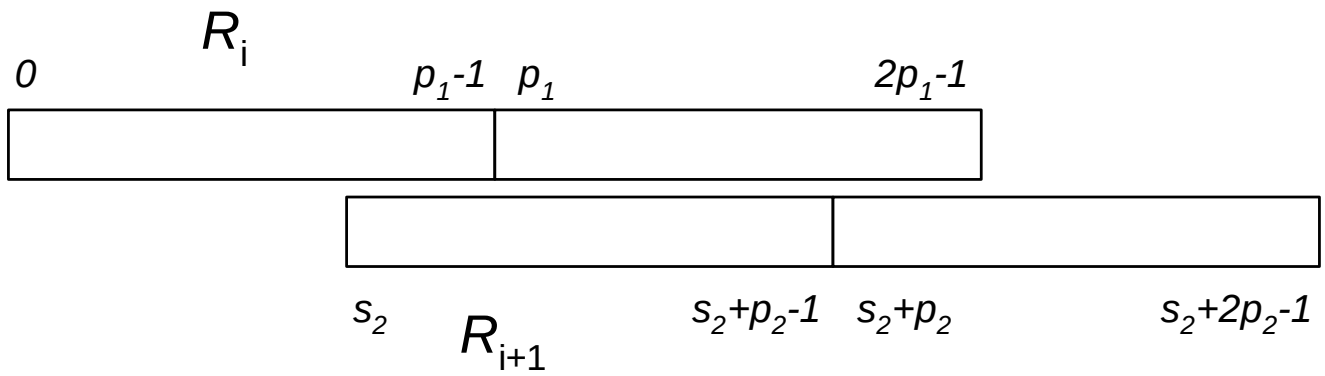
A compression scheme for strings with R-covers

The R-covers allow for a very effective compression scheme with 40-50% reduction rate. In this section we describe the compression scheme.

Consider a string x and its R-cover $\{ R_i \mid 1 \leq i \leq m \}$.

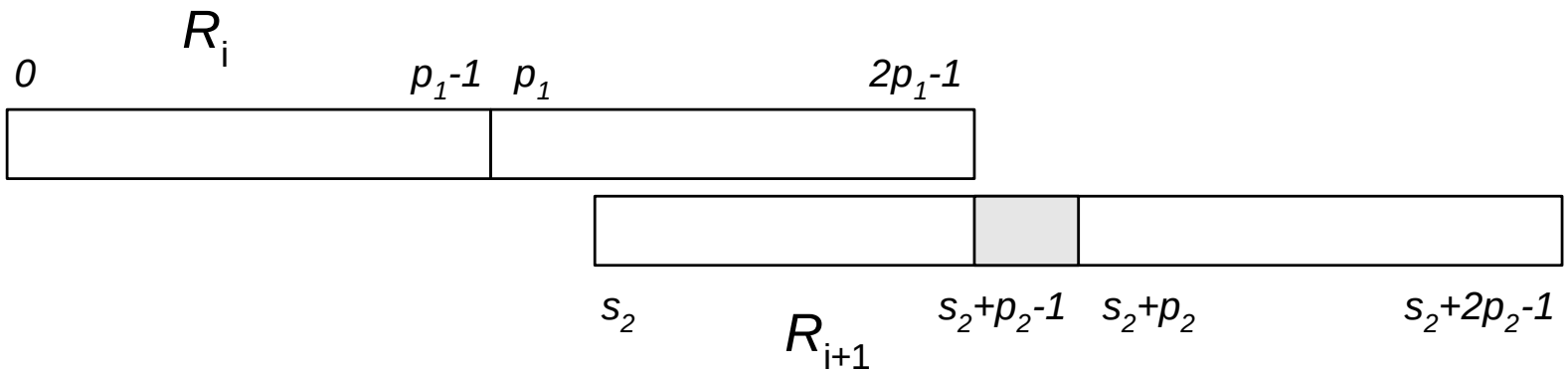
We record the period of R_1, its generator, and the offset where R_2 starts relative to the beginning of R_1 (0 if there is no R_2). If there is no R_2, we end there. If there is R_2, we record the period of R_2, the “completion of its generator", and the offset where R_3 starts relative to the beginning of R_2 (0 if there is no R_3), and so on.

What do we mean by “completion of the generator" of R_{i+1}? The two basic relative configurations of R_i and R_{i+1} are depicted in the following two diagrams.



In this configuration, the generator of R_{i+1} is in fact subsumed in R_i and does not need to be recorded as it can be easily determined, and so the “completion of the generator" is empty.

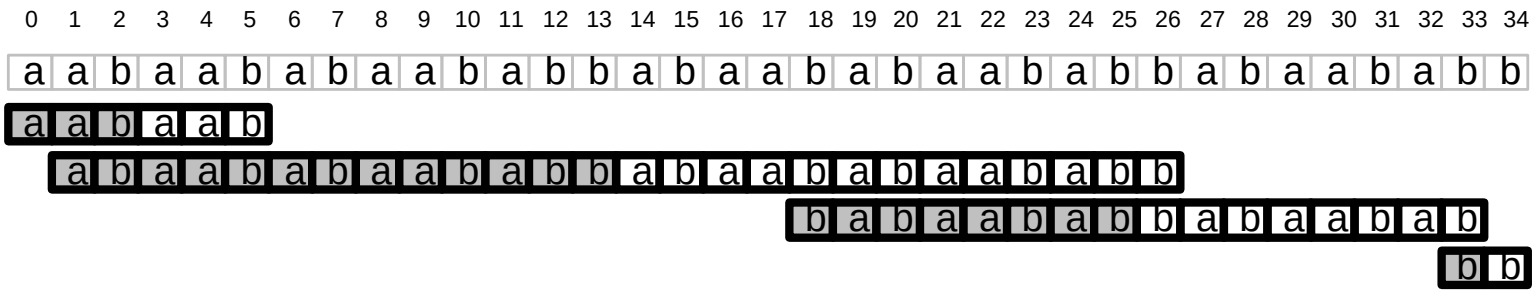
In this configuration, only a portion of the generator of R_{i+1} can be determined from R_i, the shaded part in the diagram must be recorded, and that is the “completion of the generator".



Let us recall a binary string

aabaababaababbabaababaabababbabaababb

of length 35 and its R-cover previously shown for illustration.



The compression of the string is given by
[3, *aab*, 1, 13, *abaababb*, 17, 8, 16, 1, 0].

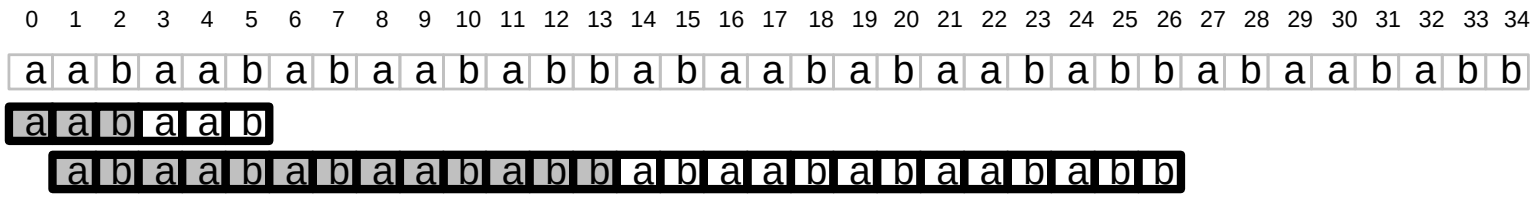
How can we decompress it?

1. We start with having the IPGN (initial part of the next generator) set to empty.
2. We read the first item, 3, and it tells us that the generator will have length 3, and since IPGN is empty, we read the next 3 characters to complete the generator `aab`, and we build the first square `aabaab`.
3. We read the next item, 1, and it tells us that the next square starts at the offset of 1, which gives 5 of the first characters of the generator, thus we set IPGN to `abaab`.

$$[\ 3, \mathit{aab}, 1, 13, \mathit{abaababb}, 17, 8, 16, 1, 0\ ]$$

McMaster University

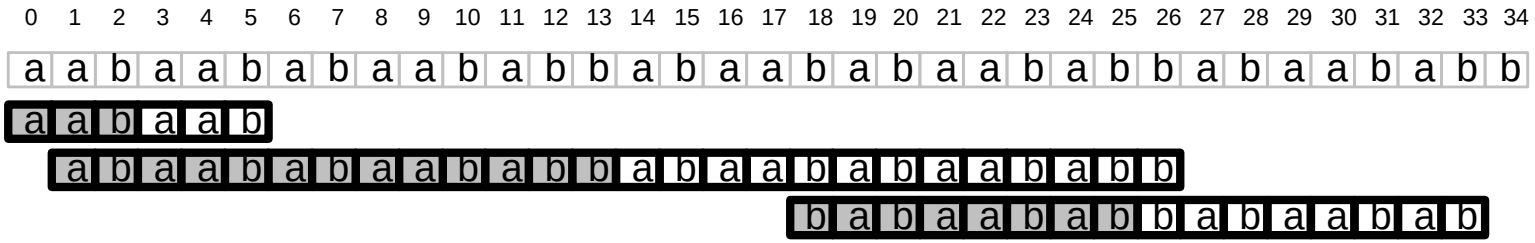
4 We read the next item, 13, which tells us that we do not have a complete generator (out of 13 we only have 5 characters), so we read the next 8 characters `abaababb` to complete the generator to `abaababaababb` and to complete the second square to `abaababaababbabaababaababb`. This gives us



[3, *aab*, 1, 13, *abaababb*, 17, 8, 16, 1, 0]

5 We read the next item, 17, which tells us that the next square starts at the offset of 17 from the beginning of the second square, so we set IPGN to `babaababb`.

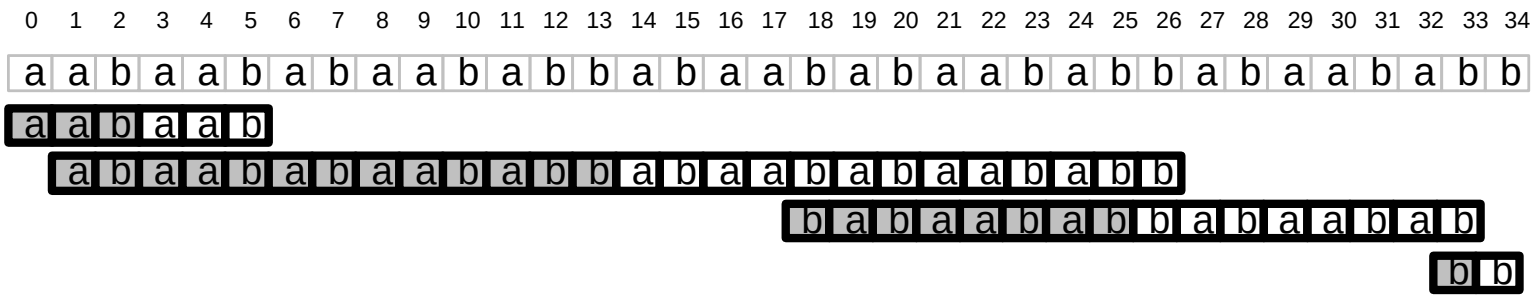
6 We read the next item, 8, which tells us that the generator will have size 8. Since IPGN has size ≥ 8, we can determine the generator as `babaabab` and complete the square to `babaababbabaabab`. This gives us



[3, *aab*, 1, 13, *abaababb*, 17, 8, 16, 1, 0]

7 We read the next item, 16, which tells us that the next square starts at the offset of 16. It gives us IPGN `b`.

8 We read the next item, 1, which tells us that the generator has length 1. Since the size of IPGN ≥ 1, we do not to read any more letters, compute the generator as `b`, complete the square to `bb`. This gives us



9 We read the next item,0, which tells us that there is no further square, we are done.

[3, *aab*, 1, 13, *abaababb*, 17, 8, 16, 1, 0]

Search strategy for run-maximal strings

Having generated a string with an R-cover, we can extend it in all possible ways with another square. We do not need to compute all runs in the extended string, all we need is to “unify" runs in the last square and the newly added square.

Conclusion

- We showed that run-maximal strings are completely covered with one or two R-covers
- This fact leads to an efficient compression scheme for run-maximal strings
- Further research will concentrate on finding if this structure of run-maximal strings can help resolve any of the three major conjectures concerning run-maximal strings.

McMaster University

THANK YOU

McMaster University