A simpler proof of Crochemore-Ilie lemma concerning maximum number of runs in a string

F. Franek

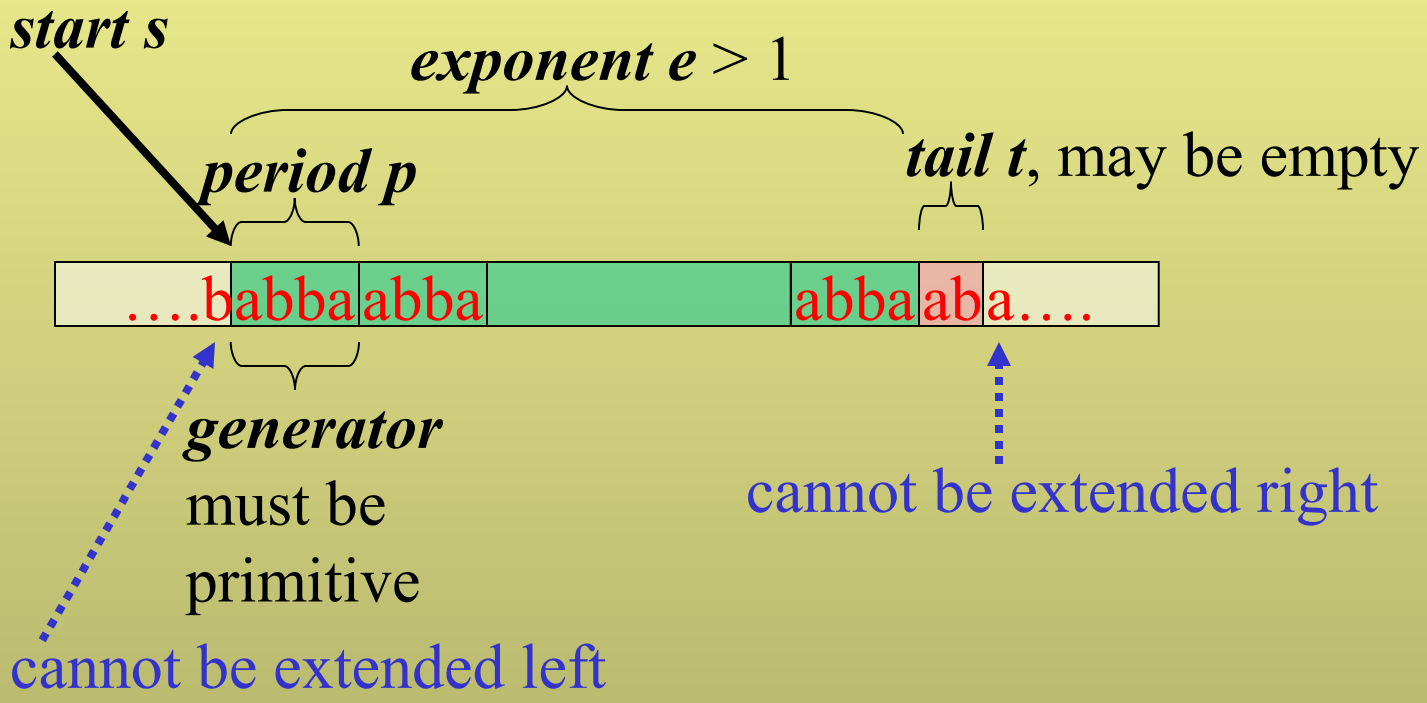
Dept. of Comp. and Software
McMaster University
Hamilton, Ontario, Canada

J. Holub

Dept. of Comp. Sci. and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Prague, Czech Republic

- Introduction of the problem of maximum number of runs in a string
- A brief history of results on bounds
- The current best upper bound by Crochemore and Ilie
- A key lemma in Crochemore-Ilie and its simpler proof using a different approach.
- Conclusion

A **run** in a string is a leftmost maximal (possibly fractional) repetition:



Naturally encoded as a 4-tuple (***s***,***p***,***e***,***t***)

An important (for computational reasons) and a natural question is **how many runs in a string**?

1981 *Crochemore*: There are $O(n \log n)$ integer runs (tails zero), attained on Fibonacci strings.

1989 *Main*: gave a linear-time algorithm to find all leftmost occurrences of runs.

1997 *Iliopoulos, Moore, & Smyth*: the number of runs in Fibonacci strings is linear.

2000 *Kolpakov & Kucherov*: the number of runs in strings is linear, however only existence of a linear constant was given, not its magnitude.

$r(x)$ = number of runs in a string x

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

2003/04 *Smyth at al*: several conjectures about $\rho(n)$

1. $\rho(n) < n$

   1a. $\lim_{n \longrightarrow \infty} \frac{\rho(n)}{n} = \frac{3}{1+\sqrt{5}} n = 0.927\ n$

2. $0 \leq \rho(n+1) - \rho(n) \leq 2$

3. $\rho(n)$ is attained by a binary cube-free string of length n

2006 *Rytter*: $\rho(n) < 5n$ later improved to $\rho(n) < 3.48n$ by *Puglisi*, *Simpson*, & *Smyth*, and again by *Rytter* to $\rho(n) < 3.44n$.

2007 *Crochemore* & *Ilie*: $\rho(n) < 1.6n$ and hinted how to lower it, may be to as low as $1.18n$.

As for the lower bound:

2003 *Franek, Simpson, & Smyth*: presented a recursive construction of a sequence of binary strings $\{xn\}$ such that

$$\lim_{n \longrightarrow \infty} \frac{r(xn)}{|xn|} = \frac{3}{1+\sqrt{5}}$$

2007 *Franek & Yang*: $\frac{3}{1+\sqrt{5}}\, n - \varepsilon$

is an asymptotic lower bound for $\rho(n)$ for any ε

Crochemore-Ilie method relies on two key theorems:

1. ***On average, there is at most one center of a δ-run ($2\delta \leq$ period $\leq 3\delta$) in each interval of length δ.***
2. ***There are $< n$ runs with periods ≤ 9 in a string of length n.***

This leads to a formula (for suitably selected intervals):

$$\rho(n) \leq n + \sum_{i=10}^{\infty} \frac{n}{\delta i} = n + \left(\frac{2}{10} \sum_{i=0}^{\infty} \left(\frac{2}{3} \right)^i \right) n$$

$$\rho(n) \leq n + 0.6\, n = 1.6\, n$$

The proof of the second theorem (*Lemma* 2 in the paper) is a complex combinatorial analysis comprising 512 cases of which only one is presented in the paper.

We present a simpler and a more straightforward proof of *Lemma* 2. However, it relies on a key fact that must be verified by computer.

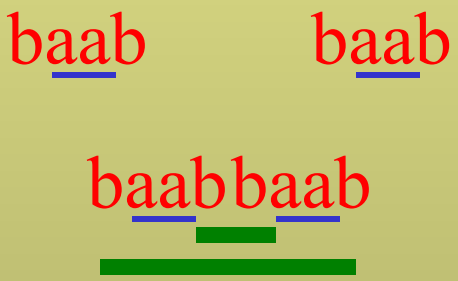
Why can't we estimate $\rho(n)$ by recursion?

(a) **# of runs can decrease**

concatenating two strings may "glue" two or more runs together

(b) **# of run can increase**

concatenating two strings may "glue" two fractions together, thus creating one or more new runs



We would like to use induction and take a string, break it into two and apply the induction hypothesis to the shorter fragments. For estimating an upper bound, we do not care about the runs that get broken into two separate runs (case a), but we must consider the runs that are destroyed by the breaking of the original string (case b).

However, there are places in a string where the analysis of the number of strings that get destroyed by the breaking is amenable to computer analysis.

The ***core*** of a run $r = (s,p,e,t)$ in a string x consists of all those positions i so that neither $x[s..i]$ is a run nor $x[(i+1)..(s+ep-1)]$ is a run.

In simple terms, breaking the string $x[1..n]$ into two fragments $x[1..i]$ and $x[i+1..n]$ for any i from the core of r will destroy r and all its subruns.

x = aabcdabaababbabbabbabbabbadacd the core is empty

x = aabcdabaababbabdacd the core is the "whole" run

x = aabcdabaababbabbadacd the core is a part of the run

Any run with $e > 3$ has an empty core. A run with $e = 3$ has a non-empty core only if $p > 1$. Any run with $e = 2$ has a non-empty core.

In essence, the core of a run is the intersection of the first and the last square of the run with the last element removed.

Lemma: There are at most n-8 runs of period ≤ 9 in a string x of length $n \geq 35$.

For $n=35$ verified computationally:

n	$\rho(n)$	$\rho_g(n)$	n	$\rho(n)$	$\rho_g(n)$	n	$\rho(n)$	$\rho_g(n)$	n	$\rho(n)$	$\rho_g(n)$
2	1	1	11	7	7	20	15	15	29	23	22
3	1	1	12	8	8	21	15	15	30	24	23
4	2	2	13	8	8	22	16	16	31	25	24
5	2	2	14	10	10	23	17	17	32	26	25
6	3	3	15	10	10	24	18	18	33	27	26
7	4	4	16	11	11	25	19	18	34	27	26
8	5	5	17	12	12	26	20	19	35	28	27
9	5	5	18	13	13	27	21	20			
10	6	6	19	14	14	28	22	21			

For higher n we continue by induction:

In the following, a ***small run*** *means a run with period* ≤ 9, ***rs(x)*** *denotes the number of small runs in a string* x.
Let x be a string of length $n \geq 35$.

If there is a position i in x that is not covered by a core of a small run, then $r_s(x[1..n]) \leq r_s(x[1..i]) + r_s(x[i+1..n])$

If both fragments are of size ≥ 35, then by induction hypothesis, $r_s(x[1..n]) \leq i - 8 + (n - i) - 8 = n - 16$.
Otherwise one of the fragments is of size ≥ 35 and the other is of size < 35 .

WLOG assume that $r_s(x[1..n])$ has size ≥ 35 . Then by induction hypothesis and the computational results, $r_s(x[1..n]) \leq i - 8 + \rho_9(n-i) \leq i - 8 + (n - i) = n - 8$.

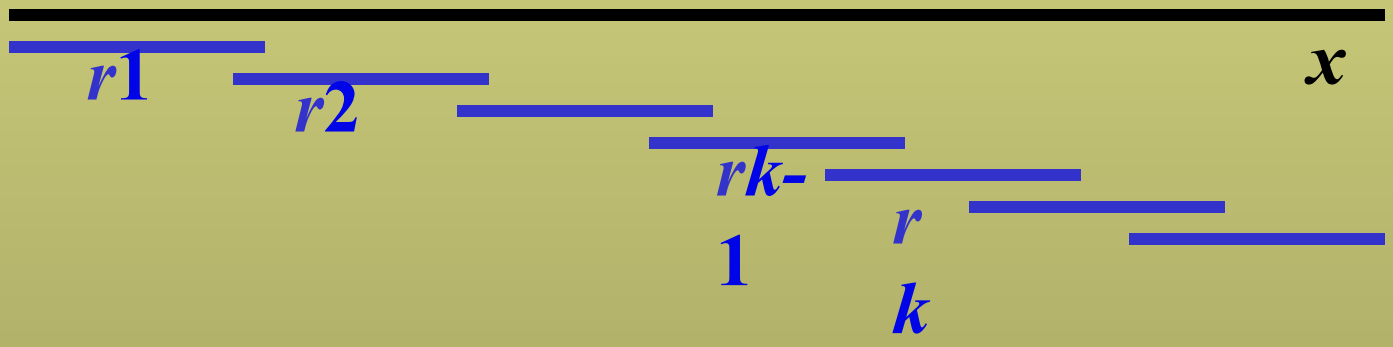
Assume that there is a position i that is covered only by a core of a single small run. Then $r_s(x[1..n])-1 \leq \rho_9(n-1) \leq (n-1)-8$. Hence $r_s(x[1..n]) \leq n-8$.

So we must assume that every position i is covered by cores of at least two small runs.

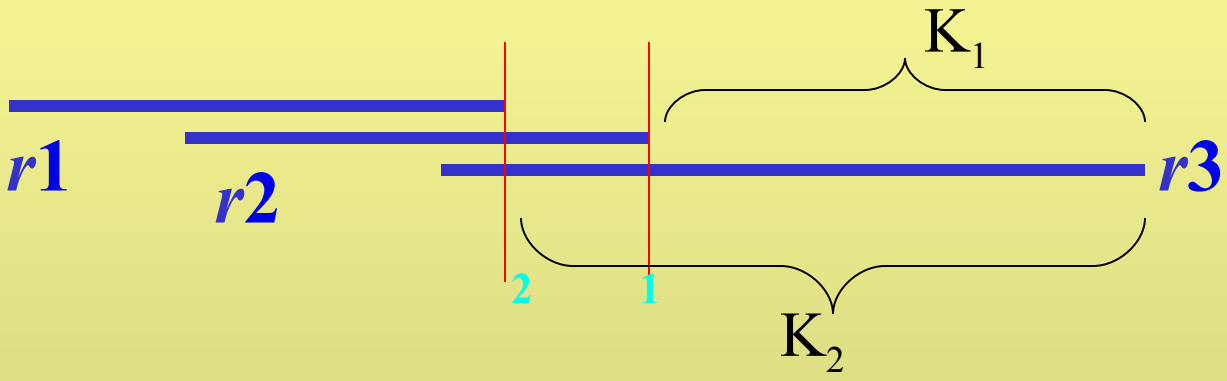
R-cover

If we order all small runs by their starting positions, and if they have the same starting position, by their period (the bigger goes first), we can cover string x by a sequence of squares $\{r_i\}$ so that

- for any small run in x, its first square is a square in some r_i.
- the intersection of r_i and r_{i+1} is non-empty.



Since $|x| \geq 35$, the r-cover ends with three squares:



Computer is used to analyze number of small runs "crossing" c_1 or being to the right of c_1, C_1. For most of the configurations r_2 and r_3, $C_1 \leq K_1$.

This is possible due to many constraints that limit the number of possibilities. For instance, if $\rho(|r_3|) \leq K_1$, it is automatically true and does not have to be considered. Also, only two squares need to be considered, r_3 and r_2.

For the few configurations of $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ for which C_1 exceeds K_1, we add the third square $\boldsymbol{r}_1$, and check C_2, the number of squares crossing $\boldsymbol{c}_2$ or to the right of $\boldsymbol{c}_2$. It turns out to be always $\leq K_2$.

Now we can finish the proof of the induction step:

Case 1: $C_1 \leq K_1$

$$\boldsymbol{r}_s(\boldsymbol{x}[1..\boldsymbol{n}]) \leq \boldsymbol{r}_s(\boldsymbol{x}[1..\boldsymbol{c}_1]) + C_1 \leq \boldsymbol{c}_1 - 8 + K_1 \leq \boldsymbol{n} - 8$$

Case 2: $C_2 \leq K_2$

$$\boldsymbol{r}_s(\boldsymbol{x}[1..\boldsymbol{n}]) \leq \boldsymbol{r}_s(\boldsymbol{x}[1..\boldsymbol{c}_2]) + C_2 \leq \boldsymbol{c}_2 - 8 + K_2 \leq \boldsymbol{n} - 8$$

QED

CONCLUSION

- The "improved" bound does not meaningfully improve the Crochemore-Ilie upper bound for $\boldsymbol{\rho(n)}$ (from $1.6\boldsymbol{n}$ to $1.6\boldsymbol{n}$ **-** 8).
- It simplifies the proof of Lemma 2, however relies on computer analysis.
- We conjecture that the computer verified observation that *either* $C_1 \leq K_1$ *or* $C_2 \leq K_2$ holds for any bounded size of periods (not just 9).
- The consequence of the conjecture will be a theorem *For any* $\boldsymbol{q}$*, there are* $< \boldsymbol{n}$ *runs with periods* $\leq \boldsymbol{q}$ *in a string of size* $\boldsymbol{n}$.

Thank you!