A computational framework for determining run-maximal strings

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

Department of Computing and Software
McMaster University, Hamilton, Ontario, Canada

LSD/LAW 2012, King's College, London, UK
February 9-10 2012

Outline

Motivation and background

- We investigate the function
$$\rho_d(n) = \max\{\ \boldsymbol{r}(x) \mid x \text{ is a } (d,n)\text{-string}\ \}$$
$\boldsymbol{r}(x)$ denotes the number of runs in a string x
(d,n)-string denotes a string of length n with exactly d distinct symbols.
- We introduce the notion of r-cover and show how it can be used for recursive computational determination of $\rho_d(n)$.
- r-covers can be used as a computational framework for an efficient computation of the maximum number of runs.

Baker+Deza+F (*On the structure of run-maximal strings*) introduced the notion of an *r-cover* as a means to represent the distribution of the runs in a string and thus describe the structure of run-maximal strings.

The straightforward assertion that a run-maximal string has an r-cover – except possibly a single weak point – holds only when the size of the alphabet is not kept fixed.

However, the approach can be adapted inductively to handle situations with fixed alphabets and can be used to speed up generation of the pool of the strings containing a run-maximal string.

Definition

An *r-cover* of a string $x = x[1 \mathinner{..} n]$ is a sequence of primitively rooted squares $\{ S_i = (s_i, e_i, p_i) \mid 1 \le i \le m \}$ so that

(1) none of the S_i's, $2 \le i \le m$ are left-shiftable;

(2) for any $1 \le i < m$, $s_i < s_{i+1}$ and $d_i \le s_{i+1}+1$, i.e. two consecutive squares are either adjacent or overlap;

(3) $\displaystyle\bigcup_{1 \le i \le m} S_i = x$;

(4) for any run (s, e, p) of x there is $1 \le i \le m$ so that S, the leading square of the run is a substring of S_i, denoted by $S \subseteq S_i$.

A string which has an r-cover is referred to as *r-covered*.

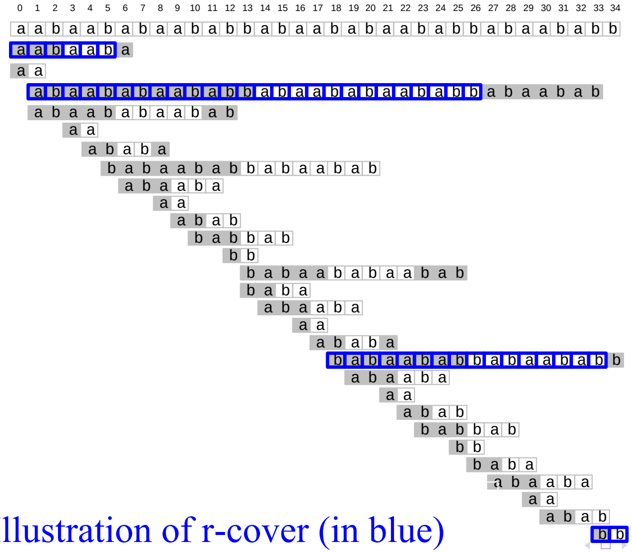

Illustration of r-cover (in blue)

$(d, n{-}d)$ table

	$n-d$: 1	2	3	4	5	6	7	8	9	10	11	12
d = 2	1	2	2	3	4	5	5	6	7	8	$\rho_2(13)$	.
3	1	2	3	3	4	5	6	6	7	8	$\rho_3(14)$	.
4	1	2	3	4	4	5	6	7	7	8	$\rho_4(15)$	.
5	1	2	3	4	5	5	6	7	8	8	.	.
6	1	2	3	4	5	6	6	7	8	9	.	.
7	1	2	3	4	5	6	7	7	8	9	.	.
8	1	2	3	4	5	6	7	8	8	9	.	.
9	1	2	3	4	5	6	7	8	9	9	.	.
10	1	2	3	4	5	6	7	8	9	10	$\rho_{10}(21)$	.
11	.	.	.	.	.	.	.	.	.	.	$\rho_{11}(22)$	.

The main diagonal, the second diagonal

There are several reasons for organizing the table in this unorthodox manner:

- The regularities the table exhibits point to several ways of possible induction
- Point to a proper upper bound: $\rho_d(n) \leq n{-}d$
- Allow a reduction of the problem, as the behaviour of the items on the main diagonal determines the behaviour of all entries.

Basic properties of $\rho_d(n)$ function

- $\rho_d(n) \leq \rho_{d+1}(n+1)$ for $n \geq d \geq 2$
- $\rho_d(n) \leq \rho_d(n+1)$ for $n \geq d \geq 2$
- $\rho_d(n) < \rho_{d+1}(n+2)$ for $n \geq d \geq 2$
- $\rho_d(n) = \rho_{d+1}(n+1)$ for $2d \geq n \geq d \geq 2$
- $\rho_d(n) \geq n-d$, $\rho_d(2d+1) \geq d$ and $\rho_d(2d+2) \geq d+1$ for $2d \geq n \geq d \geq 2$
- $\rho_{d-1}(2d-1) = \rho_{d-2}(2d-2) = \rho_{d-3}(2d-3)$ and $0 \leq \rho_d(2d) - \rho_{d-1}(2d-1) \leq 1$ for $d \geq 5$

- $\{\rho_d(n) \leq n-d \text{ for } n \geq d \geq 2\} \iff \{\rho_d(2d) \leq d \text{ for } d \geq 2\}$
- $\{\rho_d(n) \leq n-d \text{ for } n \geq d \geq 2\} \iff \{\rho_d(9d) \leq 8d \text{ for } d \geq 2\}$
- $\{\rho_d(n) \leq n-d \text{ for } n \geq d \geq 2\} \iff \{\rho_d(2d+1) - \rho_d(2d) \leq 1 \text{ for } d \geq 2\}$
- $\{\rho_d(2d+1) \leq d \text{ for } d \geq 2\} \implies \{\rho_d(2d) = d \text{ and } \rho_d(n) \leq n-d-1 \text{ for } n > 2d \geq 4\}$
- $\{\rho_d(2d) = \rho_d(2d+1) \text{ for } d \geq 2\} \implies \{\rho_d(n) \leq n-d-1 \text{ and } \rho_d(2d) = d \text{ for } n > 2d \geq 4\}$
- $\{\rho_d(2d) = \rho_d(2d+1) \text{ for } d \geq 2\} \implies \{$square-maximal $(d, 2d)$-strings are, up to relabelling, unique and equal to $a_1a_1a_2a_2a_2 \ldots a_da_d\}$.

Basic properties of r-covered strings

Lemma

If a string x is r-covered, its r-cover is unique.

Lemma

If a string x has an r-cover, then it is singleton free.

Definition

We say that the symbol $x[i]$ *destroys k runs in x*, if

(a) $i = 1$ and $\boldsymbol{r}(x) - \boldsymbol{r}(x[2 \mathinner{..} n]) = k$, or

(b) $i = n$ and $\boldsymbol{r}(x) - \boldsymbol{r}(x[1 \mathinner{..} n{-}1]) = k$, or

(c) $1 < i < n$ and $\boldsymbol{r}(x) - \boldsymbol{r}(x[i \mathinner{..} i{-}1]) - \boldsymbol{r}(x[i{+}1 \mathinner{..} n]) = k$.

Definition

A singleton-free (d, n)-string x is *t-dense*, $t \geq 1$, if

(a) $x[1]$ destroys strictly more than $t-\rho_d(n-1)$ runs in x;

(b) $x[i]$, $1 < i < n$, destroys strictly more than $t-\boldsymbol{r}(x[1 \mathinner{..} i-1])-\rho_{d'}(n-i)$ runs in x, where $d' = |\mathcal{A}(x[1 \mathinner{..} n]) - \mathcal{A}(x[1 \mathinner{..} i])| \leq d$;

(c) $x[n]$ destroys strictly more than $t-r(x[1 \mathinner{..} n-1])$ runs in x.

Lemma

If a singleton-free (d, n)-string is not t-dense, then $\mathbf{r}(x) \leq t$.

Lemma

Let x be a (d, n)-string. If any $x[i]$ destroys at least one run in x, then x has an r-cover.

Note that for an (d, n)-string having an r-cover implies being singleton free, however it does not imply that every $x[i]$ destroys at least one run, even though it is very close to it.

The notion of density was designed so it can be checked using only knowledge of an initial segment of a string and use the knowledge of the $\rho_d(n)$ function for the previous values of n. The following lemma shows how the density of an r-covered string x can be verified incrementally using the members of the r-cover.

Lemma

Let $\{\ S_i = (s_i, e_i, p_i) \mid 1 \le i \le m\ \}$ *be the r-cover of an* (d, n)*-string* x*. Then* x *is* t*-dense if and only if*

(a) *for any* $1 \le i < m$*, for any* $s_i \le j < s_{i+1}$*,* $x[j]$ *destroys strictly more than* k *runs in* x*, where* $k =$
$$\begin{cases} t-\rho_d(e_i-1) & \text{if } j = 0 \\ t-\boldsymbol{r}(x[1\mathinner{..}j-1])-\rho_{d'}(e_i-j) & \text{if } 0 < j \text{ and } d' = |\mathcal{A}(x[1\mathinner{..}j]) - \mathcal{A}(x[1\mathinner{..}n])| \end{cases}$$

(b) *for any* $s_m \le j \le e_m$*,* $x[j]$ *destroys strictly more than* k *runs in* x*, where* $k =$
$$\begin{cases} t-\boldsymbol{r}(x[1\mathinner{..}j-1])-\rho_{d'}(n-j) & \text{if } j < n \text{ and } d' = |\mathcal{A}(x[1\mathinner{..}j]) - \mathcal{A}(x[1\mathinner{..}n])| \\ t-\boldsymbol{r}(x[1\mathinner{..}n-1]) & \text{if } j = n \end{cases}$$

Lemma

Let $\rho_d(n_1) + \rho_d(n_2) \leq \rho_d(n_1+n_2)$ for any $n_1+n_2 = n-1$. If a singleton-free run-maximal (d, n)-string x does not have an r-cover, then $\rho_d(n) = \rho_d(n-1)$.

Lemma

If a run-maximal (d, n)-string has a singleton, then either $\rho_d(n) = \rho_d(n-1)$ or $\rho_d(n) = \rho_{d-1}(n-1)$.

Corollary

Let x be a run-maximal $(d, 2d)$-string, $d \geq 2$. If x has any singletons, they can all be moved to the beginning or the end of the string without affecting the number of runs in x.

Computational framework

Heuristic for a lower bound $\rho_d^-(n)$

The higher the value of $\rho_d^-(n)$ that we can compute easily, the less computational effort must be spent on determining $\rho_d(n)$.

For $d = 2$, generate $\mathcal{L}_2(n)$, the set of (d, n)-strings which are: r-covered, balanced over every prefix (the frequencies of *a*'s and *b*'s differ by at most a predefined constant), and contain no triples (*aaa* or *bbb*). Then

$$\rho_2^-(n) = \max \{ \ \rho_2(n-1), \ \ \rho_2(n-2)+1, \max_{x \in \mathcal{L}_2(n)} \boldsymbol{r}(x) \ \}.$$

This heuristic was found to be very good when tested against the known binary run-maximal strings: Franek & Smyth up to 34, and Kolpakov & Kucherov up to 60. Note that $\rho_2^-(25) < \rho_2(25)$ since the only run-maximal $(2, 25)$-string contains a triple.

For $d \geq 3$, we simply set $\rho_d^-(n) = \rho_{d-1}(n)+1$.

Generating r-covered strings

Rather than generating strings, we generate their r-covers. The generation proceeds by extending the partially built r-cover in all possible ways. Every time a potential square of the r-cover is to be extended by one position, all previously used symbols and the first unused symbol are tried. For each symbol, the frequency counter is checked that the symbol does not exceed $n+2-2d$. Once a symbol is used, the frequency counter is updated. When the whole r-cover is generated, the counter is checked whether all d symbols occurred in the resulting string; if not, the string is rejected.

For computational efficiency reasons we opted instead for a user-stack controlled backtracking implemented as a co-routine `Next()` allowing us to call the co-routine repeatedly to produce the next string. Note that the strings are generated in a lexico-graphic order.

- The generator for the first square and any other square in the r-cover that is adjacent to the previous square is generated by iterative calls to `Next()` producing all the possible generators. Each generator is checked for the additional properties (must be primitive, is not left-shiftable, did not create an intermediate square in the intermediate string, etc.) before it is accepted.

- For subsequent square, if the new square is overlapping the previous square, its generator may be partially or fully determined. If it is partially determined, iterative calls to `Next()` are used to generate all possible completions of the generator. The complete generator is checked and accepted or rejected.
- In addition, if the density of the string being generated is to be checked.

Recursive computation of $\rho_d(n)$

- verify that $\rho_d(n_1)+\rho_d(n_2) \leq \rho_d(n-1)$ for any $n_1+n_2 = n-1$.
- compute $\rho_d^-(n)$
- generate $\mathcal{R}$, the set of all $\rho_2^-(d)$-dense r-covered (d, n)-strings

Consider a singleton-free run-maximal string $x \notin \mathcal{R}$. Either x does not have an r-cover, in which case $\rho_d(n) = \rho_d(n{-}1)$, or x has an r-cover and is not $\rho^-_d(n)$-dense, in which case by$\rho_d(n) \leq \rho^-_d(n)$. Therefore

$$\rho_d(n) = \max \{\ \rho^-_d(n),\ \rho_d(n{-}1),\ \rho_{d-1}(n{-}1),\ \max_{x \in \mathcal{R}_k} \boldsymbol{r}(x)\ \}.$$

$\mathcal{R} \subseteq \{\ x \mid x \text{ is a } (d, n)\text{-string}\ \}$ and from the computational results, the decrease in size is very significant, in many cases $\mathcal{R}$ is empty.

Recursive computation of $\rho_d(2d)$

For computation of values on the main diagonal we can use r-covers satisfying additional conditions.

Definition

The r-cover $\{\ S_i = (s_i, e_i, p_i) \mid 1 \le i \le m\ \}$ of $x = x[1\ ..\ n]$ satisfies the *parity condition* if for any $1 \le i < m$, $\mathcal{A}(x[1\ ..\ e_i-1]) \cap \mathcal{A}(x[s_{i+1}+1\ ..\ n]) \subseteq \mathcal{A}x[s_{i+1}\ ..\ e_i]$.

Lemma

The singleton-free part of a run-maximal $(d, 2d)$-string with all its singletons at the end has an r-cover satisfying the parity condition.

With additional assumptions, the previous lemma can be strengthen to exclude non-overlapping squares from the r-cover.

Lemma

Let $\rho_{d'}(2d') = d'$ for any $d' < d$. Either $\rho_d(2d) = d$ or for every run-maximal $(d, 2d)$-string x with v singletons all at the end, $v \le d - 2$, its singleton-free part $x[1 \,..\, 2d - v]$ has an r-cover satisfying the parity condition and which has no adjacent non-overlapping squares.

Since the number of runs in a singleton-free $(d, 2d)$-string is at most d, we do not need to consider the singleton-free strings.

We can consider only $(d, 2d)$-strings that have singletons at the end. Since $\rho_d(2d) > \rho_{d-1}(2d-2)$, we can set $\rho_d^-(2d) = \rho_{d-1}(2d-2)+1$ and thus consider only the strings that have the non-singleton part $\rho_d^-(2d)$-dense.

Moreover, we can only consider strings whose r-covers of the non-singleton part satisfy the parity condition with no adjacent non-overlapping squares.

The number of singletons must be at least $\lceil \frac{7d}{8} \rceil$.

For every $\lceil \frac{7d}{8} \rceil \leq v \leq d-2$, let $\mathcal{T}_v$ denote the set of all singleton-free $\rho^-_d(2d))$-dense r-covered $(d, 2d-v)$-strings whose r-covers satisfy the parity condition and have no adjacent non-overlapping squares. Then

$$\rho_d(2d) = \max\left(d, \max\left\{\max_{x \in \mathcal{T}_v} \boldsymbol{r}(x) \ : \ \left\lceil \frac{7d}{8} \right\rceil \leq v \leq d-2\right\}\right)$$

Conclusion and future research

We first presented the notion of r-covers as a structural generalization of a uniform distribution of runs in a string. Then we showed that only for r-covered strings we do not know the exact value of the maximum number of runs and hence only r-covered strings must be examined to determine $\rho_d(n)$.

Based on these observations, we presented a fast and efficient computational framework with significantly reduced search space for computations of $\rho_d(n)$ based on the notion of density.

THANK YOU