Computational approaches to the maximum number of distinct squares

Mei Jiang
Joint work with Antoine Deza and Frantisek Franek

Advanced Optimization Laboratory
Department of Computing and Software
McMaster University

November 22, 2011

Outline

Outline

Basic Notation

- A **square** is a repetition with power of 2, encoded as $(s, p, 2)$, **distinct squares** means only the types of the squares are counted, **primitively rooted distinct squares** means the generator itself is not a repetition. i.e. $x = aababaa$.
- $\boldsymbol{\sigma_d(n)}$ denotes the maximum number of primitively rooted distinct squares over all strings of length n containing exactly d distinct symbols.
- A **singleton** refers to a symbol in a string that occurs exactly once, a **pair** occurs exactly twice.

d-step Approach

We introduced a *d*-step approach to investigate the problem of distinct squares in relationship to the alphabet of the string [4].

d \ n - d	1	2	3	4	5	6	7	8	9	10	...
1	**1**	1	1	1	1	1	1	1	1	1	...
2	1	**2**	2	3	3	4	5	6	7	7	...
3	1	2	**3**	3	4	4	5	6	7	8	...
4	1	2	3	**4**	4	5	5	6	7	8	...
5	1	2	3	4	**5**	5	6	6	7	8	...
6	1	2	3	4	5	**6**	6	7	7	8	...
7	1	2	3	4	5	6	**7**	7	8	8	...
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**	...
...	...	...	...	...	...	...	...	...	...	...	...

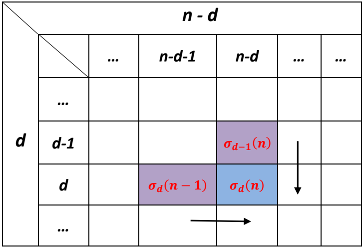
(d, n-d) Table: $\sigma_d(n)$ with $1 \le d \le 10$ and $1 \le n - d \le 10$

Outline

1. Introduction
2. *R*-cover
3. Computation on distinct squares
4. Program
5. Future work

Motivation

- Finding square-maximal strings by brute force algorithm is time consuming with large d and n.
- The goal is to prune down the search space of the square-maximal strings as much as possible.
- $(d,n\text{-}d)$ table shows $\sigma_d(n)$ is either larger than or equal to the previous computed values.



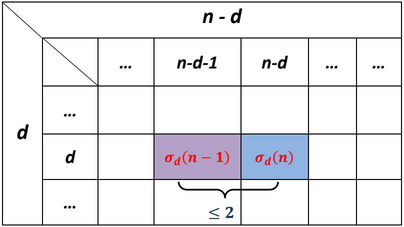
R-cover

We define a set of squares $\{R_i = (s_i, p_i, 2) : 1 \leq i \leq m\}$ is an R-cover of a string x if:

1. each R_i is a distinct primitively rooted square in x;
2. for every $R_i, 1 \leq i \leq m$, R_i is unique in $x[1..s_i + 2p_i - 1]$;
3. for every R_i and $R_{i+1}, 1 \leq i < m$, R_{i+1} is not in $x[s_i + 2p_i + 1..n]$;
4. for every R_i and $R_{i+1}, 1 \leq i < m$, $s_i < s_{i+1} \leq s_i + 2p_i$ and $s_i + 2p_i - 1 < s_{i+1} + 2p_{i+1} - 1$;
5. for all $1 \leq j \leq n$, there exists an $1 \leq i \leq m$ such that $s_i \leq j \leq s_i + 2p_i - 1$;

Double *R*-cover

- By Fraenkel-Simpson [1], there are at most two right most distinct squares starting at the same position in a string.
- $\sigma_d(n)$ is increased at most 2 compare to the previous computed value $\sigma_d(n-1)$.
- Further prune down the search space of square-maximal strings.



Double R-cover (cont.)

Proposition 1

A square-maximal string x with d symbols and length n has $\sigma_d(n) = \sigma_d(n-1) + 2$, x satisfy the **double R-cover density condition**: every letter in x occurs in at least two distinct squares.

Proof.

Suppose x does not meet the double R-cover density condition: there exist a letter in x occurs in only one distinct square. The removal of this letter (form string y), will destroy at most one square of x. Therefore $\sigma_d(n) = s(x) \leq s(y) + 1 \leq \sigma_d(n-1) + 1$ which leads to a contradiction. □

Outline

Along the row

- Motivation
  1. achieve better upper bound for $\sigma_d(n)$ with given d, i.e. $\sigma_2(n) \leq 2n - 47$ with $\sigma_2(37) = 27$;
  2. conjecture exact formula of $\sigma_d(n)$ with $d = 2$ row.
- Search space is narrowed down on strings satisfy double R-cover density.
- Compare to $\sigma_d(n-1)$, if rule out the possibility of $\sigma_d(n)$ increasing by 2 and by 1 in double R-cover strings search space, then search only one R-cover string that increases by 1 is sufficient.

Main diagonal

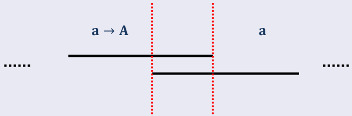
- Motivation
  1. prove the conjecture of $\sigma_d(n) \leq n - d$ is true up to certain d;
  2. achieve better upper bound of $\sigma_d(n)$ for all d and n, i.e. $\sigma_d(n) \leq 2n - d_0 - 2d$, where d_0 is the maximum where $\sigma_{d_0}(2d_0) = d_0$ is known;
- Computationally with smaller search space of strings:
  1. satisfy double R-cover density;
  2. contains at least $\lceil \frac{2d}{3} \rceil$ singletons[4], *i.e. computationally, we only need to generate strings with* $d = \lfloor \frac{d}{3} \rfloor$ *and* $n = d + \lfloor \frac{d}{3} \rfloor$;
  3. no pairs[4];
  4. parity condition.

Main diagonal (cont.)

Proposition 2

A square-maximal string x with d symbols and length $2d$ contains an R-cover satisfies the **parity condition**: the overlap of any two R-cover squares must contain the symbols that occur in both of the non-overlapping parts.

Proof.



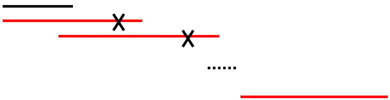
		n - d				
		...	*d-1*	*d*	...	...
d	...	...				
	d-1		$\sigma_{d-1}(2d-2)$			
	d		...	$\sigma_d(2d)$		
	...		$\sigma_{d+1}(2d)$		...	
	...					...

Suppose there exist such symbol a, change every occurrences of a to A in the left non-overlapping part to form y. Thus, $\sigma_d(2d) = s(x) \leq s(y) \leq \sigma_{d+1}(2d) = \sigma_{d-1}(2d-2)$, a contradiction. □

Outline

Description

- Build double R-cover strings:
  1. With all possible periods, consider all primitive generators and extend it to form the first square;
  2. Calculate the weak point by far;
  3. Build the next square within range which satisfies:
     - the generator is primitive;
     - is unique by far.
  4. Repeat step 2 and 3 and until reach desire length;
  5. When desire length is reached, ensure there is no weak point.



- Apply the appropriate conditions: i.e. parity condition when computing the main diagonal value.

Preliminary Result

- Program was implemented in C++, and run on Advol5 (8× Quad-Core AMD Opteron 8356) and Advol3 (16× Dual Core AMD Opteron 885) server.
- Along the row:
  - $\sigma_2(37)$ was computed in about 5 days;
  - did not return any result for weeks by brute force program.
- On the main diagonal:
  - $\sigma_{16}(32)$ (equivalent to $\sigma_5(21)$) was computed in about 1 hour and 40 minutes;
  - took weeks to run $\sigma_{10}(20)$ by brute force program.

Future work

Future work

- Use heuristic search on R-cover strings. i.e. apply conditions that likely to return square maximal strings with increase of 1.
- Current program recomputes all R-cover strings for length $n-1$ when computes length n. Develop a mechanism to be able to extend R-cover strings from shorter ones.
- Parallelize the program to speed up the computation.
- Because of the nature of the R-cover strings for distinct squares, we have to allow intermediate squares crossing the R-cover squares which is the bottleneck of the program.
- If the first and the last R-cover squares are the same for a given string and its reversal string, then we could avoid generating duplicated strings by restricting $p_1 \leq p_m$.

References

- A. S. Fraenkel and J. Simpson, *How Many Squares Can a String Contain?*, Journal of Combinatorial Theory Series A, 82, 1 (1998), 112-120.
- L. Ilie, *A simple proof that a word of length n has at most 2n distinct squares*, Journal of Combinatorial Theory Series A, 112, 1 (2005) 163-164.
- L. Ilie, *A note on the number of squares in a word*, Theoretical Computer Science, 380, 3 (2007), 373-376.
- A. Deza, F. Franek, and M. Jiang, *A d-step approach for distinct squares in strings*, Lecture Notes in Computer Science, Volume 6661 (2011) 77-89.

THANK YOU!