

# Hardness of Detecting Abelian and Additive Square Factors in Strings

**Jakub Radoszewski** ✉ 

University of Warsaw, Poland  
Samsung R&D, Warsaw, Poland

**Wojciech Rytter** ✉ 


University of Warsaw, Poland

**Juliusz Straszyński** ✉ 

University of Warsaw, Poland

**Tomasz Waleń** ✉ 

University of Warsaw, Poland

**Wiktor Zuba** ✉ 

University of Warsaw, Poland

---

## Abstract

We prove 3SUM-hardness (no strongly subquadratic-time algorithm, assuming the 3SUM conjecture) of several problems related to finding Abelian square and additive square factors in a string. In particular, we conclude conditional optimality of the state-of-the-art algorithms for finding such factors.

Overall, we show 3SUM-hardness of (a) detecting an Abelian square factor of an odd half-length, (b) computing centers of all Abelian square factors, (c) detecting an additive square factor in a length- $n$  string of integers of magnitude  $n^{\mathcal{O}(1)}$ , and (d) a problem of computing a double 3-term arithmetic progression (i.e., finding indices  $i \neq j$  such that  $(x_i + x_j)/2 = x_{(i+j)/2}$ ) in a sequence of integers  $x_1, \dots, x_n$  of magnitude  $n^{\mathcal{O}(1)}$ .

Problem (d) is essentially a convolution version of the AVERAGE problem that was proposed in a manuscript of Erickson. We obtain a conditional lower bound for it with the aid of techniques recently developed by Dudek et al. [STOC 2020]. Problem (d) immediately reduces to problem (c) and is a step in reductions to problems (a) and (b). In conditional lower bounds for problems (a) and (b) we apply an encoding of Amir et al. [ICALP 2014] and extend it using several string gadgets that include arbitrarily long Abelian-square-free strings.

Our reductions also imply conditional lower bounds for detecting Abelian squares in strings over a constant-sized alphabet. We also show a subquadratic upper bound in this case, applying a result of Chan and Lewenstein [STOC 2015].

**2012 ACM Subject Classification** Theory of computation → Pattern matching

**Keywords and phrases** Abelian square, additive square, 3SUM problem

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2021.77

**Related Version** *Full Version:* <https://arxiv.org/abs/2107.09206>

**Funding** *Jakub Radoszewski:* Supported by the Polish National Science Center, grant no. 2018/31/D/ST6/03991.

*Juliusz Straszyński:* Supported by the Polish National Science Center, grant no. 2018/31/D/ST6/03991.

*Tomasz Waleń:* Supported by the Polish National Science Center, grant no. 2018/31/D/ST6/03991.

*Wiktor Zuba:* Supported by the Polish National Science Center, grant no. 2018/31/D/ST6/03991.

**Acknowledgements** The authors warmly thank Paweł Gawrychowski and Tomasz Kociumaka for helpful discussions.



© Jakub Radoszewski, Wojciech Rytter, Juliusz Straszyński, Tomasz Waleń, and Wiktor Zuba; licensed under Creative Commons License CC-BY 4.0

29th Annual European Symposium on Algorithms (ESA 2021).

Editors: Petra Mutzel, Rasmus Pagh, and Grzegorz Herman; Article No. 77; pp. 77:1–77:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

**Abelian squares.** An Abelian square, Ab-square in short (also known as a jumbled square), is a string of the form  $XY$ , where  $Y$  is a permutation of  $X$ ; we say that  $X$  and  $Y$  are Ab-equivalent. We are interested in factors (i.e., substrings composed of consecutive letters) of a given text string being Ab-squares.

► **Example 1.** The string

0 6 1 0 5 6 5 1 6 1 0 1 1 1 1 0 6 5 6 5 7 8 6 1 6 5 0 5 1 0 5 6 6 5 0 6 0 3 0 6 5 2

has exactly two Ab-square factors of length 12, shown above (but it has also Ab-squares of other lengths, e.g. 5665, 11, 1111, 011110).

Ab-squares were first studied by Erdős [16], who posed a question on the smallest alphabet size for which there exists an infinite Ab-square-free string, i.e., an infinite string without Ab-square factors. The first example of such a string over a finite alphabet was given by Evdokimov [18]. Later the alphabet size was improved to five by Pleasants [34] and finally an optimal example over a four-letter alphabet was shown by Keränen [27]. Further results on combinatorics of Ab-square-free strings and several examples of their applications in group theory, algorithmic music and cryptography can be found in [26] and references therein. Avoidability of long Ab-squares was also considered [36].

Strings containing Ab-squares were also studied. Motivated by another problem posed by Erdős [16], Entringer et al. [15] showed that every infinite binary string has arbitrarily long Ab-square factors. Fici et al. [19] considered infinite strings containing many distinct Ab-squares. A string of length  $n$  may contain  $\Theta(n^2)$  Ab-square factors that are distinct as strings, but contains only  $\mathcal{O}(n^{11/6})$  Ab-squares which are pairwise Abelian nonequivalent (correspond to different Parikh vectors), see [28]. It is also conjectured that a binary string of length  $n$  must have at least  $\lfloor n/4 \rfloor$  distinct [20] and nonequivalent [21] Ab-square factors. For more conjectures related to combinatorics of Ab-square factors of strings and circular strings, see [39].

Several algorithms computing Ab-square factors of a string are known. All Ab-squares in a string of length  $n$  can be computed in  $\mathcal{O}(n^2)$  time [13]. For a string over a constant-sized alphabet, all Ab-square factors of a string can be computed in  $\mathcal{O}(n^2/\log^2 n + \text{output})$  time and the longest Ab-square can be computed in  $\mathcal{O}(n^2/\log^2 n)$  time [29, 30]. Moreover, for a string of length  $n$  that is given by its run-length encoding consisting of  $r$  runs, the longest Ab-square that ends at each position can be computed in  $\mathcal{O}(|\Sigma|(r^2 + n))$  time [2] or in  $\mathcal{O}(rn)$  time [40]; both approaches require  $\Omega(n^2)$  time in the worst case.

In [37] a different problem of enumerating strings being Ab-squares was considered.

**Additive squares.** An additive square is an even-length string over an integer alphabet such that the sums of characters of the halves of this string are the same.

► **Example 2.** The following string has exactly 4 additive squares of length 10, as shown.

1 2 0 3 2 1 2 0 2 3 2 1 0 1 2 3

All of them except for the rightmost one are also Ab-squares. This string does not contain any longer additive square. Altogether this string has 8 additive square factors.

An Ab-square (over an integer alphabet) is an additive square, but not necessarily the other way around. Combinatorially, problems related to additive squares are hard, in particular avoiding additive squares seems more difficult than avoiding Ab-squares. There are infinitely many strings over  $\{0, 1, 2, 3\}$  avoiding Ab-squares, but there are only finitely many strings over the same alphabet avoiding additive squares; see [22].

In fact it is unknown if there are infinitely many strings over any finite integer alphabet avoiding additive squares [7, 25, 33]. For additive cubes the property was proved in [9] (see also [32]) however.

Nowadays, combinatorial study of Ab-square and additive square factors often involves computer experiments; see e.g. [9, 19, 36]. In addition to other applications, efficient algorithms detecting such types of squares could provide a significant aid in this research. In case of classic square factors (i.e., factors of the form  $XX$ ), a linear-time algorithm for computing them is known for a string over a constant [24] and over an integer alphabet [4, 12]. We show that, unfortunately, in many cases the existence of near-linear-time algorithms for detecting Ab-square and additive square factors is unlikely, based on conjectured hardness of the 3SUM problem.

**3SUM problem.** The problem asks if there are distinct elements  $a, b, c \in S$  such that  $a + b = c$  for a given set  $S$  of  $n$  integers; see [35]. It is a general belief that the following conjecture is true for the word-RAM model.

**3SUM conjecture.** There is no  $\mathcal{O}(n^{2-\epsilon})$  time algorithm for the 3SUM problem, for any constant  $\epsilon > 0$ .

A problem with input of size  $n$  is called 3SUM-hard if an  $\mathcal{O}(n^{2-\epsilon})$ -time solution to the problem implies an  $\mathcal{O}(n^{2-\epsilon'})$ -time solution for 3SUM, for some constants  $\epsilon, \epsilon' > 0$ .

#### Our results.

- We show that the problems of computing all centers of Ab-square factors and detecting an odd half-length Ab-square factor, called an *odd Ab-square* (consequently also computing all lengths of Ab-square factors), for a length- $n$  string over an alphabet of size  $\omega(1)$ , cannot be solved in  $\mathcal{O}(n^{2-\epsilon})$  time, for constant  $\epsilon > 0$ , unless the 3SUM conjecture fails. Weaker conditional lower bounds are also stated in the case of a constant-sized alphabet.
- For constant-sized alphabets, we show strongly sub-quadratic algorithms for these problems based on an involved result of [11] related to jumbled indexing.
- En route we prove that detection of a double 3-term arithmetic progression (see [8]) and additive squares in a length- $n$  sequence of integers of magnitude  $n^{\mathcal{O}(1)}$  is 3SUM-hard.

We obtain deterministic conditional lower bounds from a convolution version of 3SUM that is well-known to be 3SUM-hard.

**Related work.** In the jumbled indexing problem, we are given a text  $T$  and are to answer queries for a pattern specified by a Parikh vector which gives, for each letter of the alphabet, the number of occurrences of this letter in the pattern. For each query, we are to check if there is a factor of the text that is Ab-equivalent to the pattern (existence query) or report all such factors (reporting query). Chan and Lewenstein [11] showed a data structure that can be constructed in truly subquadratic expected time and answers existence queries in truly sublinear time for a constant-sized alphabet (deterministic constructions for very small alphabets were also shown). Amir et al. [3] showed under a 3SUM-hardness assumption that jumbled indexing with existence queries requires  $\Omega(n^{2-\epsilon})$  preprocessing time or  $\Omega(n^{1-\delta})$

queries for any  $\epsilon, \delta > 0$  for an alphabet of size  $\omega(1)$ . They also provided particular constants  $\epsilon_\sigma, \delta_\sigma$  for an alphabet of a constant size  $\sigma \geq 3$  such that, under a stronger 3SUM-hardness assumption, jumbled indexing requires  $\Omega(n^{2-\epsilon_\sigma})$  preprocessing time or  $\Omega(n^{1-\delta_\sigma})$  queries. We use the techniques from both results in our algorithm and conditional lower bound for Ab-squares, respectively. The lower bound of Amir et al. was later improved and extended to both existence and reporting variants and any constant  $\sigma \geq 2$  by Goldstein et al. [23, Section 7] with the aid of randomization. Moreover, recently an unconditional lower bound for the reporting variant was given in [1].

**Our techniques.** A subsequence of three distinct positions is a 3-term double arithmetic progression (3dap in short) if it is an arithmetic progression and the elements on these positions also form an arithmetic progression. The problem of finding a 3dap in a sequence is denoted by 3DAP. It is an odd 3dap if the first and the third positions are odd and the middle position is even. The corresponding problem is denoted by ODD-3DAP. First we reduce the convolution problem 3SUM (known to be 3SUM-hard) to the 3DAP problem via ODD-3DAP as an intermediate problem. This uses a divide-and-conquer approach and a partition of sets into sets avoiding *bad* arithmetic progression of length 3.

The 3DAP problem reduces in a simple way to detection of an additive square, showing that the latter problem is 3SUM hard.

Next, the 3DAP problem is encoded as a string. We follow the high-level idea from Amir et al. Instead of checking equality of numbers, we can check equality of their remainders modulo sufficiently many prime numbers. Then, each prime number corresponds to a distinct character. If the numbers are poly  $n$  then only  $\mathcal{O}(\log n)$  prime numbers are needed. However, there is a certain technical complication, already present in the paper of Amir et al., which needs an introduction of additional gadgets working as *equalizers*. The details, compared with construction of Amir et al., are different, mostly because in the end we want to ask about detection, not indexing.

Then we consider the problem of computing all centers of Ab-squares, this requires new gadgets. We show that computing all centers of Ab-squares is 3SUM-hard, as well as detection of any Ab-square which is *well centred*.

Later we extend this to detection of any odd Ab-square. We use a construction of a string over the alphabet of size 4 with no Ab-square. The input string is “shuffled” with such a string, with some separators added. This forces odd Ab-squares to be *well centered*, in this way we reduce the previously considered problem of detection of any well-centred Ab-square to the detection of any odd Ab-square. Ultimately, this shows that the latter problem is 3SUM-hard.

## 2 From Conv3SUM to finding double 3-term arithmetic progressions

For integers  $a, b$ , by  $[a, b]$  we denote the set  $\{a, \dots, b\}$ . We use the following convolution variant of the 3SUM problem that is 3SUM-hard; see [10, 31, 35] for both randomized and deterministic reductions. As already noted in [3], the range of elements can be made  $[-N^2, N^2]$  using a randomized hashing reduction from [5, 35].

CONV3SUM( $\bar{x}$ )

**Input:** A sequence  $\bar{x} = [x_1, \dots, x_N] \in [-N^2, N^2]$

**Output:** Yes if there are  $i \neq j$  such that  $x_i + x_j = x_{i+j}$ ; no otherwise.

Let us denote  $\text{mid}(a, b) = (a + b)/2$  and define the condition:

$$\Lambda_{\bar{x}}(i, j) = (i \neq j \wedge j - i \text{ is even} \wedge x_{\text{mid}(i, j)} = \text{mid}(x_i, x_j)).$$

We omit the subscript  $\bar{x}$  if it is clear from the context. The last part of the condition is equivalent to  $x_j - x_{\text{mid}(i, j)} = x_{\text{mid}(i, j)} - x_i$ .

Our first goal is to reduce the CONV3SUM problem to the following one with  $K = N^{\mathcal{O}(1)}$ .

Double 3-Term Arithmetic Progression, 3DAP( $\bar{x}$ )  
**Input:**  $\bar{x} = [x_1, \dots, x_n]$ , each of  $x_i$  is in  $[0, K]$ .  
**Output:**  $(\exists i, j) \Lambda(i, j)$ .

In Section 2.1 we obtain a reduction of CONV3SUM to an intermediate version of 3DAP with additional constraints on  $i, j$ , and in Section 2.2 we show how these constraints can be avoided.

## 2.1 From Conv3SUM to Odd-3DAP

Let us fix an integer sequence  $x_1, \dots, x_N$ . For an arithmetic progression (arithmetic sequence)  $\mathcal{I} = i_1, \dots, i_n$ , where  $1 \leq i_1 < \dots < i_n \leq N$ , i.e.  $i_2 - i_1 = \dots = i_n - i_{n-1}$ , we define the following extended functions.

$$\text{CONV3SUM}(\bar{x}, \mathcal{I}) = (\exists i_a, i_b \in \mathcal{I} : x_{i_a} + x_{i_b} = x_{i_a+i_b}, i_a < i_b)$$

$$\text{ODDCONV3SUM}(\bar{x}, \mathcal{I}) = (\exists i_a, i_b \in \mathcal{I} : x_{i_a} + x_{i_b} = x_{i_a+i_b}, a + b \text{ is odd}).$$

Note that it can happen that  $i_a + i_b \notin \mathcal{I}$ . For a fixed  $\bar{x}$  the input size is  $|\mathcal{I}|$ .

► **Lemma 3.** *An instance of CONV3SUM( $\bar{x}$ ) can be reduced to an alternative of  $\mathcal{O}(N)$  instances of ODDCONV3SUM( $\bar{x}, \mathcal{I}$ ) of total size  $\mathcal{O}(N \log N)$  in  $\mathcal{O}(N \log N)$  time.*

**Proof.** If  $\mathcal{I} = i_1, \dots, i_n$ , by  $\mathcal{I}_{\text{odd}}$  and  $\mathcal{I}_{\text{even}}$  we denote the subsequences  $i_1, i_3, \dots$  and  $i_2, i_4, \dots$ , respectively. We proceed recursively as shown in the following function CONV3SUM, with the first call to CONV3SUM( $\bar{x}, [1, 2, \dots, N]$ ).

---

**function** CONV3SUM( $\bar{x}, \mathcal{I}$ )

Comment:  $\mathcal{I}$  is an arithmetic progression

**if**  $|\mathcal{I}| \leq 2$  **then return false;**

**return** ODDCONV3SUM( $\bar{x}, \mathcal{I}$ )  $\vee$  CONV3SUM( $\bar{x}, \mathcal{I}_{\text{odd}}$ )  $\vee$  CONV3SUM( $\bar{x}, \mathcal{I}_{\text{even}}$ );

---

**Correctness.** Let  $\mathcal{I} = i_1, \dots, i_n$  and assume there are two indices  $a, b$  such that  $x_{i_a} + x_{i_b} = x_{i_a+i_b}$ . If  $a + b$  is odd, then ODDCONV3SUM( $\bar{x}, \mathcal{I}$ ) returns true. Otherwise both  $a, b$  are of the same parity, so  $i_a, i_b \in \mathcal{I}_{\text{odd}}$  or  $i_a, i_b \in \mathcal{I}_{\text{even}}$ . Consequently, the problem is split recursively into subproblems that correspond to  $\mathcal{I}_{\text{odd}}$  and  $\mathcal{I}_{\text{even}}$ .

**Complexity.** Let us observe that one call to CONV3SUM( $\bar{x}, \mathcal{I}$ ) creates an instance of ODDCONV3SUM of  $\mathcal{O}(|\mathcal{I}|)$  size in  $\mathcal{O}(|\mathcal{I}|)$  time ( $\bar{x}$  does not change). Let  $\#(n)$  and  $S(n)$  denote the total number and size of all instances of  $\mathcal{I}$  generated by CONV3SUM( $\bar{x}, \mathcal{I}$ ), when initially  $|\mathcal{I}| = n$ . We then have

$$\#(n) = \#(\lfloor n/2 \rfloor) + \#(\lceil n/2 \rceil) + 1 \quad \text{and} \quad S(n) = S(\lfloor n/2 \rfloor) + S(\lceil n/2 \rceil) + \Theta(n),$$

which yields  $\#(N) = \mathcal{O}(N)$  and  $S(N) = \mathcal{O}(N \log N)$ . The reduction takes  $\mathcal{O}(S(N))$  time. ◀

## 77:6 Hardness of Detecting Abelian and Additive Squares

We say that a 3-element arithmetic progression is a *good* progression if the middle element is even and two others are odd and introduce the following problem.

ODD-3DAP( $\bar{x}$ )

**Input:**  $\bar{x} = [x_1, \dots, x_n]$ , each of  $x_i$  is in  $[-\mathcal{O}(N^2), \mathcal{O}(N^2)]$ .

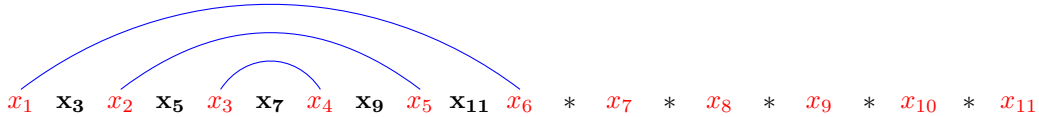
**Output:**  $(\exists i, j)$  [ $\Lambda(i, j)$  and  $(i, \text{mid}(i, j), j)$  is a good progression].

► **Lemma 4.**  $\text{ODD CONV 3SUM}(\bar{x}, \mathcal{I})$  is reducible in  $\mathcal{O}(|\mathcal{I}|)$  time and space to  $\text{ODD-3DAP}(\bar{y})$ , where  $|\bar{y}| = \mathcal{O}(|\mathcal{I}|)$ .

**Proof.** Let  $\mathcal{I} = i_1, \dots, i_n$ . Define  $\alpha_N = 2N^2 + 1$  and let  $\bar{y}$  be a sequence of length  $2n - 1$  that is created as follows:

1. put  $x_{i_1}, x_{i_2}, \dots, x_{i_n}$  at subsequent odd positions in  $\bar{y}$ ;
2. at each even position  $2j$ , put  $x_{i_j + i_{j+1}}$  or, if  $i_j + i_{j+1} > N$ , put  $\alpha_N$ .
3. multiply elements on even positions by 2.

After the first two steps  $\text{ODD CONV 3SUM}(\bar{x}, \mathcal{I})$  is equivalent to  $(\exists i, j) y_{\text{mid}(i, j)} = y_i + y_j$  for odd  $i, j$  and even  $\text{mid}(i, j)$ ; see Figure 1. Then, after the third step,  $\text{ODD CONV 3SUM}(\bar{x}, \mathcal{I})$  is equivalent to  $\text{ODD-3DAP}(\bar{y})$ . ◀



■ **Figure 1** The sequence constructed in Lemma 4 for  $\bar{x} = [x_1, x_2, x_3, \dots, x_{11}]$  and  $\mathcal{I} = (1, 2, \dots, 11)$  after the first two steps (\* denotes  $\alpha_{11}$ ). Note that the elements connected by arcs all have their sum of indices equal to 7; this is because  $\mathcal{I}$  is an arithmetic progression.

## 2.2 From Odd-3DAP to 3DAP

Our main tool in this subsection is partitioning a set of integers into progression-free sets. A set of integers  $A$  is called *progression-free* if it does not contain a non-constant three-element arithmetic progression. We use the following recent result that extends a classical paper of Behrend [6].

► **Theorem 5** ([14]). Any set  $A \subseteq [1, n]$  can be partitioned into  $n^{o(1)}$  progression-free sets in  $n^{1+o(1)}$  time.

► **Lemma 6.** We can construct in  $n^{1+o(1)}$  time a family  $\mathcal{F}$  of  $n^{o(1)}$  subsets of  $[1, n]$  satisfying:

- (a) Each good 3-element progression is contained in some  $S \in \mathcal{F}$ .
- (b) If  $S \in \mathcal{F}$ , then all 3-element arithmetic progressions in  $S$  are good.

**Proof.** Let us divide the elements from  $[1, n]$  into three classes:

$$\text{BLUE} = \{i \leq n : i \text{ is even}\},$$

$$\text{RED} = \{i \leq n : i \bmod 4 = 1\}, \quad \text{GREEN} = \{i \leq n : i \bmod 4 = 3\}.$$

Each element  $i \in [1, n]$  has the colour blue, red or green of its corresponding class. Each class forms an arithmetic progression.

A progression is called *multi-chromatic* if its elements are of three distinct colours. Let us observe that a 3-element progression is good if and only if it is multi-chromatic. Indeed, this is because if  $i, j \in \text{RED}$  (or  $\text{GREEN}$ ), then  $\text{mid}(i, j)$  is odd.

Now instead of good progressions we will deal with multi-chromatic progressions. We treat sets of integers as increasing sequences and for a set  $C = \{c_1, \dots, c_m\}$  we denote by  $C_{\text{odd}}$  and  $C_{\text{even}}$  the subsets  $\{c_1, c_3, \dots\}$  and  $\{c_2, c_4, \dots\}$ .

For example  $\text{BLUE}_{\text{odd}} = \{i \leq n : i \bmod 4 = 2\}$ ,  $\text{RED}_{\text{even}} = \{i \leq n : i \bmod 8 = 5\}$ .

Our construction works as follows:

1. Partition the set  $[1, n]$  into classes  $\text{BLUE}, \text{RED}, \text{GREEN}$ .
2. For each class  $C \in \{\text{BLUE}, \text{RED}, \text{GREEN}\}$  partition it in  $n^{1+o(1)}$  time into a family  $\mathcal{F}_C$  of  $n^{o(1)}$  progression-free sets with the use of Theorem 5.
3. Refine each partition  $\mathcal{F}_C$ , splitting each set  $X \in \mathcal{F}_C$  into two sets  $X \cap C_{\text{odd}}, X \cap C_{\text{even}}$ , so that for each set  $X$  in the new refined partition  $\mathcal{F}_C$  we have  $X \subseteq C_{\text{odd}}$  or  $X \subseteq C_{\text{even}}$ . Each family is still of size  $n^{o(1)}$ .
4. Return  $\mathcal{F} = \{X \cup Y \cup Z : X \in \mathcal{F}_{\text{BLUE}}, Y \in \mathcal{F}_{\text{RED}}, Z \in \mathcal{F}_{\text{GREEN}}\}$ .

**Proof of point (a).** Each multi-chromatic progression is contained in some  $S \in \mathcal{F}$  since each element of  $C$  is contained in a set from  $\mathcal{F}_C$ .

**Proof of point (b).** The proof is by contradiction. Assume that  $S \in \mathcal{F}$  contains a progression which is not multi-chromatic. There are two cases.

**Case 1:** the progression is monochromatic, hence it appears in a single set  $X \in \mathcal{F}_C$ . However every  $X$  is progression-free (step 2), hence such a progression cannot appear in any  $S \in \mathcal{F}$ ; a contradiction.

**Case 2:** the progression contains exactly two different colors. Observe that if  $i \bmod p = \text{mid}(i, j) \bmod p = r$ , then  $j \bmod p = r$  (if the middle element of progression belongs to the same class as one of the other elements, then the triple is monochromatic), hence the two-coloured arithmetic progression has to consist of  $i, j \in C$  and  $\text{mid}(i, j) \notin C$ .

Since  $i, j$  both belong to  $C_{\text{odd}}$  or  $C_{\text{even}}$  (step 3),  $\text{mid}(i, j)$  must belong to  $C$  (if  $i \bmod 2p = j \bmod 2p$ , then  $i \bmod p = \text{mid}(i, j) \bmod p$ ). Consequently, the progression cannot contain exactly two colours; a contradiction. ◀

Our next tool is a *deactivation* of a set of elements which indexes are not in a given set  $E$ , that is, omitting them in the computation of a solution. For  $E \subseteq [1, n]$  the operation  $\text{restr}(\bar{x}, E)$  replaces each element  $x_i$  on position  $i \notin E$  by  $5 \max\{\text{MAX}, n^2\} + i^2$ , where  $\text{MAX} = \max_{k \geq 1} |x_k|$ .

► **Lemma 7.**  $3\text{DAP}(\text{restr}(\bar{x}, E)) \iff (\exists i, j) \Lambda_{\bar{x}}(i, j) \wedge i, j, \text{mid}(i, j) \in E$ .

**Proof.** The  $(\Leftarrow)$  part is obvious, so it suffices to show  $(\Rightarrow)$ . If at least one, but not all, of  $i, j, \text{mid}(i, j)$  is not in  $E$ , then  $\Lambda_{\bar{y}}(i, j)$  cannot hold for  $\bar{y} = \text{restr}(\bar{x}, E)$  because  $y_{\text{mid}(i, j)}$  and  $\text{mid}(y_i, y_j)$  differ by at least  $\max\{\max_k \{|x_k|\}, n^2\}$  (for an exhaustive check of all the cases, see the full version). Otherwise, if all the positions  $i, j, \text{mid}(i, j)$  are not in  $E$ , then  $\Lambda_{\bar{y}}(i, j)$  does not hold because  $\text{mid}(i^2, j^2) - (\frac{i+j}{2})^2 = (\frac{i-j}{2})^2 > 0$  since  $i \neq j$ . ◀

An instance  $\bar{x}$  is called an **odd-half** instance if  $\Lambda(i, j)$  is false for  $i, j$  such that  $(j - i)/2$  is even (equivalently, for  $i, j$  such that  $i$  and  $\text{mid}(i, j)$  have the same parity). Efficient equivalence

$$\text{ODD-3DAP}(\bar{x}) \iff (\exists S \in \mathcal{F}) 3\text{DAP}(\text{restr}(\bar{x}, S))$$

follows now from Lemmas 6 and 7.

This produces only odd-half instances because only good progressions are left in the construction of Lemma 6. The instances have elements in  $[-\mathcal{O}(N^2), \mathcal{O}(N^2)]$ . We can increase all the elements by  $\mathcal{O}(N^2)$  so that they become non-negative. This implies:

► **Lemma 8.** *An instance of ODD-3DAP can be reduced in  $n^{1+o(1)}$  time to  $n^{o(1)}$  odd-half instances of 3DAP of total size  $n^{1+o(1)}$  and with elements up to  $K = \mathcal{O}(N^2)$ .*

Finally, we show that the resulting instances can be glued together to a single equivalent one.

► **Theorem 9.** *An instance of CONV3SUM can be reduced in  $N^{1+o(1)}$  time to an odd-half instance of 3DAP of size  $n = N^{1+o(1)}$  with elements up to  $K = N^{3+o(1)}$ .*

**Proof.** With Lemmas 3, 4, and 8 we obtain a reduction from CONV3SUM to  $N^{1+o(1)}$  odd-half instances of 3DAP of total size  $N^{1+o(1)}$ . The instances have elements in  $[0, \mathcal{O}(N^2)]$ . We will show that these instances can be reduced to a single odd-half instance of 3DAP of size  $N^{1+o(1)}$  with elements in the range  $[0, N^{3+o(1)}]$  in time  $N^{1+o(1)}$ . The resulting instance will return true if and only if at least one of the input instances does.

Let  $t = N^{1+o(1)}$  be the number of the instances of 3DAP, numbered 1 through  $t$ . We use Theorem 5<sup>1</sup> and pick the largest constructed progression-free set  $A \subseteq [1, m]$ , for some  $m$ . By the pigeonhole principle,  $|A| \geq m^{1-o(1)}$ . We select  $m$  that is large enough so that  $m^{1-o(1)} \geq t$ , so  $m = t^{1+o(1)} = N^{1+o(1)}$ , and trim the set  $A$  to the size  $t$ . Let  $A = \{a_1, \dots, a_t\}$ . For instance  $i$  we multiply all its elements by  $2m$  and add to each element the value  $a_i$ . Finally we concatenate all the instances.

If any of the input instances returns true, then so does the output instance, since multiplication by and addition of the same number to all elements cannot affect the outcome of a single instance. If none of the input instances returns true, then the only possibility for the output instance to return true is to contain a 3-element arithmetic progression with elements from multiple parts corresponding to the input instances. However, this is impossible since, taken modulo  $2m$ , the progression would form an arithmetic progression in the set  $A$ . ◀

► **Corollary 10.** *The general 3DAP problem is also 3SUM-hard.*

► **Remark 11.** Similarly as in CONV3SUM, techniques from [5] can be used to hash down the range in 3DAP to integers of magnitude  $\mathcal{O}(N^2)$  (cf. [3]), using randomization.

► **Remark 12.** The AVERAGE problem (introduced by J. Erickson [17]) asks if there are distinct elements  $a, b, c \in S$  such that  $a + b = 2c$  for a given set  $S$  of  $n$  integers. It was recently shown to be 3SUM-hard [14]. The 3DAP problem can be viewed as a convolution version of the AVERAGE problem<sup>2</sup>. The ideas based on almost linear hashing used in the reductions from 3SUM to CONV3SUM [35, 10] can be extended with some effort to reduce AVERAGE to 3DAP. We presented a different reduction that additionally directly leads to an instance of 3DAP with an odd-half property, which is essential in our proof of 3SUM-hardness of computing Ab-squares (see the proof of Lemma 19).

<sup>1</sup> Actually, a deterministic version of Behrend's construction from [14] or an earlier construction of Salem and Spencer [38] would suffice here.

<sup>2</sup> <https://cs.stackexchange.com/questions/10681/is-detecting-doubly-arithmetic-progressions-3sum-hard/10725#10725>



### 2.3 Hardness of detecting additive squares

If the alphabet is a set of integers, then a string  $W$  is called an *additive square* if  $W = UV$ , where  $|U| = |V|$  and  $\sum_{i=1}^{|U|} U[i] = \sum_{i=1}^{|V|} V[i]$ .

► **Theorem 13.** *Finding an additive square in a length- $N$  sequence composed of integers of magnitude  $N^{O(1)}$  is 3SUM-hard.*

**Proof.** We use Theorem 9 to reduce CONV3SUM to an instance of 3DAP of size  $n = N^{1+o(1)}$  with elements in the requested range. 3DAP returns true on an instance  $x_1, \dots, x_n$  if and only if the sequence  $x_2 - x_1, x_3 - x_2, \dots, x_n - x_{n-1}$  contains an additive square. As the reduction works in  $N^{1+o(1)}$  total time, the conclusion follows. ◀

## 3 From arithmetics to Abelian stringology

We use capital letters to denote strings and lower case Greek letters to denote sets of integers. We assume that the positions in a string  $S$  are numbered 1 through  $|S|$ , where  $|S|$  denotes the length of  $S$ . By  $S[i]$  and  $S[i..j]$  we denote the  $i$ th letter of  $S$  and the string  $S[i] \cdots S[j]$  called a factor of  $S$ . The reverse of string  $S$ , i.e. the string  $S[|S|] \cdots S[1]$ , is denoted as  $S^R$ . By  $\varepsilon$  we denote the empty string. By  $Alph(S)$  we denote the set of distinct letters in  $S$ .

We denote Ab-equivalence of  $U$  and  $V$  by  $U \cong V$ . For a string  $U$ , by  $Parikh(U)$  we denote the Parikh vector of  $U$ . Then  $U \cong V$  if and only if  $Alph(U) = Alph(V)$  and  $Parikh(U) = Parikh(V)$ .

We use an encoding of Amir et al. [3] based on the Chinese remainder theorem to connect CONV3SUM-type problems with Abelian stringology.

Let  $p_1 < p_2 < \dots < p_k$  be prime numbers. The Chinese remainder theorem states that if one knows the remainders  $r_1, r_2, \dots, r_k$  of an integer  $x$ , such that  $0 \leq x < \prod p_i$ , when dividing by  $p_i$ 's, then one can uniquely determine  $x$ . Assuming that the remainders of an integer  $x$  are  $r_1, r_2, \dots, r_k$ , we could encode  $x$  as a possibly short string  $\mathbf{a}_1^{r_1} \mathbf{a}_2^{r_2} \cdots \mathbf{a}_k^{r_k}$  over an alphabet  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$  (the symbols correspond to consecutive prime numbers).

For example for primes 2,3,5 the encoding of 11 would be  $\mathbf{a}_1^1 \mathbf{a}_2^2 \mathbf{a}_3^1$  since its remainders modulo 2,3,5 are 1,2,1, respectively. However, we are interested in encodings of subtractions of one number from another one, and it is more complicated.

Let  $\bar{x} = [x_1, \dots, x_n]$  be an instance of 3DAP and  $r_1^{(i)}, r_2^{(i)}, \dots, r_k^{(i)}$  be remainders of  $x_i$  modulo  $p_1, p_2, \dots, p_k$ . Like Amir et al. [3], we define for  $1 \leq i < n$  and  $1 \leq j \leq k$ ,

$$EXP_i(j) = r_j^{(i+1)} - r_j^{(i)} + d \text{ where } d = \max_{j=1}^k p_j, \quad \mathbf{S}_i = \mathbf{a}_1^{EXP_i(1)} \mathbf{a}_2^{EXP_i(2)} \cdots \mathbf{a}_k^{EXP_i(k)}.$$

We choose a sequence  $p_1, \dots, p_k$  of  $k$  distinct primes such that  $p_1 \cdots p_k > \max\{x_i\}$ . In this way we encode the difference  $x_j - x_i$ , for  $j > i$ , by a string  $\mathbf{S}_i \mathbf{S}_{i+1} \cdots \mathbf{S}_{j-1}$ . An obstacle is the potentially possible inequality  $(a \bmod p) - (b \bmod p) \neq (a - b) \bmod p$ . However a small correction is sufficient, due to the following observation.

► **Observation 14.**  $(a \bmod p) - (b \bmod p) + q = (a - b) \bmod p$ , where  $q \in \{0, p\}$ .

If we apply the encoding to an instance  $\bar{x} = x_1, \dots, x_n$  of 3DAP, we obtain a lemma that is analogous to [3, Lemma 1].

► **Lemma 15.**  $\Lambda(i, j)$  holds for  $i < j$ ,  $j - i$  even, iff for each  $t \in [1, k]$ , there are  $e_t, f_t \in \{0, p_t\}$ , such that

$$e_t + EXP_i(t) + EXP_{i+1}(t) + \cdots + EXP_{mid(i,j)-1}(t) = EXP_{mid(i,j)}(t) + EXP_{mid(i,j)+1}(t) + \cdots + EXP_{j-1}(t) + f_t.$$

## 77:10 Hardness of Detecting Abelian and Additive Squares

Let  $\Psi$  be a morphism such that  $\Psi(i) = \mathbf{a}_i^{p_i}$  for each  $i = 1, \dots, k$ . We treat a set  $U = \{u_1, \dots, u_w\}$  as a string  $u_1 \cdots u_w$ , where  $u_1 < u_2 < \dots < u_w$ . If we interpret the vector  $(EXP_i(1), EXP_i(2), \dots, EXP_i(k))$  as  $\mathbf{S}_i$ , then Lemma 15 directly implies the following fact.

► **Lemma 16.** *Assume  $i < j$  and  $j - i$  is even. Then*

$$\Lambda(i, j) \iff (\Psi(\alpha) \mathbf{S}_i \mathbf{S}_{i+1} \cdots \mathbf{S}_{\text{mid}(i,j)-1} \cong \mathbf{S}_{\text{mid}(i,j)} \cdots \mathbf{S}_{j-1} \Psi(\beta))$$

for some disjoint subsets  $\alpha, \beta$  of  $[1, k]$ .

### 4 Hardness of computing all centers of Ab-squares

We construct a text  $T$  over the alphabet  $\{\mathbf{a}_1, \dots, \mathbf{a}_k, \mathbf{b}, \bullet, \star, \#, \$\}$  such that 3DAP has a solution if and only if  $T$  contains an Ab-square with one of specified centers, so-called *well-placed* Ab-square.

First we extend each  $\mathbf{S}_i$  to have the same length  $M \geq \max_{i=1}^{n-1} |\mathbf{S}_i|$ , to be defined later. Intuitively, it is needed to control the number of  $\mathbf{S}_i$ 's in the strings from Lemma 16. We append  $M - |\mathbf{S}_i|$  occurrences of a letter  $\mathbf{b}$  to each  $\mathbf{S}_i$ . Let  $\mathbf{S}_i^I$  denote this modified string.

Lemma 16 immediately implies the following fact.

► **Lemma 17.** *Assume  $i < j$  and  $j - i$  is even. Then*

$$\Lambda(i, j) \iff (\mathbf{b}^e \Psi(\alpha) \mathbf{S}_i^I \mathbf{S}_{i+1}^I \cdots \mathbf{S}_{\text{mid}(i,j)-1}^I \cong \mathbf{S}_{\text{mid}(i,j)}^I \cdots \mathbf{S}_{j-1}^I \Psi(\beta) \mathbf{b}^f),$$

for some disjoint subsets  $\alpha, \beta$  of  $[1, k]$ , where  $e + |\Psi(\alpha)| = f + |\Psi(\beta)|$  with  $\min(e, f) = 0$ .

The parts  $\mathbf{b}^e \Psi(\alpha)$ ,  $\Psi(\beta) \mathbf{b}^f$  in the above lemma can be treated as *equalizers*. Let us note that in the above lemma we can assume that  $\max(e, f) \leq \max(|\Psi(\alpha)|, |\Psi(\beta)|) \leq kd$ .

A pair of disjoint sets  $\alpha, \beta$  that satisfies  $\alpha \cup \beta = [1, k]$  will be called a *2-partition* of  $[1, k]$ . For a 2-partition  $(\alpha, \beta)$  of  $[1, k]$ , we use the string

$$\Gamma(\alpha, \beta) = \# \alpha \$ \mathbf{b}^{kd} \# \beta \$,$$

called a  $\Gamma$ -string. If  $k = 4$ ,  $d = 7$ , an example of a  $\Gamma$ -string is  $\Gamma(2, 134) = \# 2 \$ \mathbf{b}^{28} \# 134 \$$ .

Let  $(\pi_1, \pi'_1), (\pi_2, \pi'_2), \dots, (\pi_m, \pi'_m)$  be the sequence of all  $m = 4^k$  pairs of  $\Gamma$ -strings. Define

$$U = \pi_m \pi_{m-1} \dots \pi_1, \quad V = \pi'_1 \pi'_2 \dots \pi'_m.$$

We have  $\{\pi_1, \dots, \pi_m\} = \{\pi'_1, \dots, \pi'_m\}$ , so  $U \cong V$ .

► **Observation 18.** *For disjoint subsets  $\alpha, \beta \subseteq [1, k]$  and integers  $0 \leq e, f \leq kd$ , there are decompositions  $U = U_1 \mathbf{b}^e \# \beta \$ U_2$  and  $V = V_1 \# \alpha \$ \mathbf{b}^f V_2$ , where  $U_2 \cong V_1$ .*

Let us recall the morphism  $\Psi$  such that  $\Psi(i) = \mathbf{a}_i^{p_i}$  for each  $i \in [1, k]$ . We define additionally  $\Psi(c) = c$  for  $c \in \{\mathbf{b}, \#, \$\}$  and set

$$\mathbf{B} = \Psi(U), \quad \mathbf{A} = \Psi(V), \quad M = |\mathbf{A}| = |\mathbf{B}|.$$

Let us observe that indeed  $\max_{i=1}^{n-1} |\mathbf{S}_i| \leq M$  holds since  $|\mathbf{S}_i| \leq kd + \sum_{j=1}^k p_j$  and the length of  $\Psi(W)$  for any  $\Gamma$ -string  $W$  is  $kd + \sum_{j=1}^k p_j + 4$ .

We add two new letters  $\bullet, \star$  and define the following string (the symbols “ $\downarrow$ ” are not parts of the string, but only show supposed centers of Ab-squares).

$$T = \bullet \mathbf{B} \star \mathbf{S}_1^I \mathbf{A} \bullet \downarrow \star \mathbf{B} \bullet \mathbf{S}_2^I \mathbf{A} \star \downarrow \bullet \mathbf{B} \star \mathbf{S}_3^I \mathbf{A} \bullet \downarrow \star \mathbf{B} \bullet \mathbf{S}_4^I \mathbf{A} \star \dots \quad (1)$$

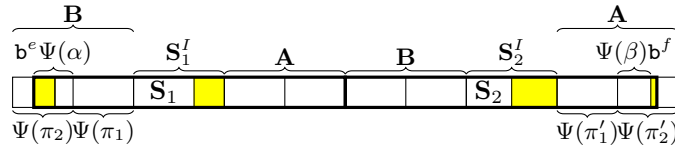


Figure 2 Internal structure of an Ab-square, shown in a thick box (proportions are symbolic), in  $\mathbf{B}\mathbf{S}_1^I\mathbf{A}\mathbf{B}\mathbf{S}_2^I\mathbf{A}$ . Here  $x_{mid(1,3)} = mid(x_1, x_3)$  and  $\mathbf{b}^e\Psi(\alpha)$ ,  $\Psi(\beta)\mathbf{b}^f$  are equalizers.

An Ab-square is called *well-placed* if its center is between the letters  $\bullet, \star$  in any order. Recall that, due to Theorem 9, we can assume that the input to 3DAP guarantees that only odd-half instances could have solutions.

► **Lemma 19.** *Assume  $\bar{x}$  is an odd-half instance. Then  $3DAP(\bar{x})$  has a solution if and only if  $T$  contains a well-placed Ab-square.*

**Proof.** Let  $\bar{x} = [x_1, \dots, x_n]$  be an odd-half instance of 3DAP. We show two implications.

( $\Rightarrow$ ) Assume that  $\Lambda(i, j)$  holds for  $\bar{x}$ . Lemma 17 implies that for strings  $W, Z$  such that  $W \cong Z$  we have

$$\mathbf{b}^e \# \Psi(\alpha) \$ W \star \mathbf{S}_i^I \mathbf{A} \bullet \star \mathbf{B} \bullet \mathbf{S}_{i+1}^I \mathbf{A} \star \cdots \bullet \mathbf{B} \star \mathbf{S}_{mid(i,j)-1}^I \mathbf{A} \bullet \cong \star \mathbf{B} \bullet \mathbf{S}_{mid(i,j)}^I \mathbf{A} \star \cdots \bullet \mathbf{B} \star \mathbf{S}_{j-2}^I \mathbf{A} \bullet \star \mathbf{B} \bullet \mathbf{S}_{j-1}^I Z \# \Psi(\beta) \$ \mathbf{b}^f \quad (2)$$

for some disjoint subsets  $\alpha, \beta$  of  $[1, k]$ , where  $e + |\Psi(\alpha)| = f + |\Psi(\beta)|$  with  $\min(e, f) = 0$ . Indeed, we use the fact that  $\mathbf{A} \cong \mathbf{B}$  and the counts of letters  $\bullet$  and  $\star$  on both hand sides are equal (because  $(j - i)/2$  is odd). By Observation 18, we obtain a well-placed Ab-square in  $T$  (or we obtain it after exchanging all letters  $\bullet$  with  $\star$ ).

( $\Leftarrow$ ) Assume that  $T$  has a well-placed Ab-square factor with center immediately after  $\bullet \mathbf{B} \star \mathbf{S}_t^I \mathbf{A} \bullet$  (the case that it is immediately after  $\star \mathbf{B} \bullet \mathbf{S}_t^I \mathbf{A} \star$  is symmetric). Let us investigate what can be the position  $s$  of the first letter of this Ab-square.

Recall that  $|\mathbf{S}_i^I| = |\mathbf{A}| = |\mathbf{B}| = M$  for each  $i \in [1, n - 1]$ , so  $T$  can be seen as composed of blocks of length  $M' = M + 1$ . We will check which of these blocks can contain  $s$ , by checking the counts of each of the letters  $\bullet, \star$  in both halves of the Ab-square. The positions of letters  $\bullet, \star$  in  $T$  repeat with period  $6(M + 1)$ , so it is sufficient to inspect the first 6 blocks on each side, as the remaining ones will behave periodically; see Figures 3 and 4.

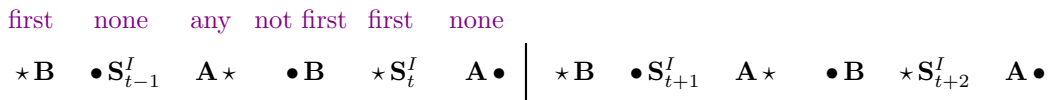


Figure 3 Which position in a block of  $T$  can be the starting position of a well-placed Ab-square with the designated center, just counting letters  $\bullet, \star$ .

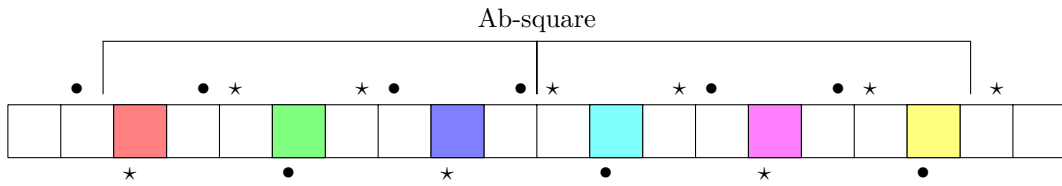
By counting letters  $\bullet, \star$  in both halves of the Ab-square, it can be readily verified that  $s$  cannot be in any block  $\mathbf{A} \bullet$  or  $\bullet \mathbf{S}_i^I$ ; if in any block  $\star \mathbf{B}$  or  $\star \mathbf{S}_i^I$ , it can only be the first position of the block; it cannot be the first position in a block  $\bullet \mathbf{B}$ ; and it can be in any position in a block  $\mathbf{A} \star$ .

Moreover,  $s$  cannot be the first position in a block  $\star \mathbf{B}$ , since this would imply, by Lemma 17, that  $\Lambda(i, j)$  holds for  $i$  such that the block  $\bullet \mathbf{S}_i^I$  immediately follows the  $\star \mathbf{B}$  block and  $j = 2t - i$ . However, in this case  $(j - i)/2$  is even, which is impossible.

If  $s$  is the first position of a block  $\star S_i^I$ , then this implies, again by Lemma 17, that  $\Lambda(i, j)$  holds for  $j = 2t - i$ . In this case  $(j - i)/2$  is odd, so this is a valid solution to the corresponding 3DAP instance.

We are left with the case that  $s$  belongs to a block  $\mathbf{A}\star$  or  $\bullet\mathbf{B}$  (and in case of  $\bullet\mathbf{B}$  does not coincide with the position of the letter  $\bullet$ ). Henceforth it suffices to count letters different from  $\bullet, \star$  in the halves. Each of the gadgets  $\mathbf{A}, \mathbf{B}$  is a concatenation of  $m$  Ab-equivalent strings of the form  $\# \Psi(\alpha) \$ \mathbf{b}^{kd} \# \Psi(\beta) \$$ , where  $\Psi(\alpha), \Psi(\beta)$  are composed of letters  $\mathbf{a}_i$  only. By counting the letters  $\#$  and  $\$$  in both halves of the Ab-square, we see that  $s$  can only be a position which holds the letter  $\mathbf{b}$  or  $\#$ .

Hence, the Ab-square is necessarily of the form (2), which, by Lemma 17, implies that  $\Lambda(i, j)$  holds, where  $\star S_i^I$  is the first such block after the position  $s$  and  $j = 2t - i$ . ◀



■ **Figure 4** The global structure of a fragment containing a well-placed Ab-square; there are three types of blocks:  $c\mathbf{B}$ ,  $cS_i^I$ ,  $\mathbf{A}c$ , where  $c$  is one of  $\bullet, \star$ . The blocks of the second type (which can be considered as essential blocks) are in color, each block is of length  $M + 1$  (recall that  $|\mathbf{A}| = |\mathbf{B}| = |S_i| = M$ ). The special letters  $\bullet, \star$  force each half of a well-placed Ab-square to contain a number of full  $S_i$ 's.

► **Theorem 20.** *Computing all positions that are centers of Ab-square factors in a length- $n$  string over an alphabet of size  $\omega(1)$  is 3SUM-hard.*

**Proof.** Due to Theorem 9 we can reduce CONV3SUM in  $N^{1+o(1)}$  time to an odd-half instance  $\bar{x}$  of 3DAP of size  $n = N^{1+o(1)}$  with elements in the range  $[0, N^{3+o(1)}]$ .

We construct the string  $T$  as shown in Eq. 1 for the sequence  $\bar{x}$ . Then Lemma 19 implies that 3DAP is a YES-instance if and only if  $T$  has a well-placed Ab-square. The string  $T$  has length  $\mathcal{O}(N^{1+o(1)}M)$ . Each of the strings  $\mathbf{A}, \mathbf{B}$  has length  $M$  and is composed of  $m = 4^k$  strings of length  $\mathcal{O}(kd)$ , i.e.,  $\Psi$ -images of  $\Gamma$ -strings.

Hence,  $M = \mathcal{O}(4^k kd)$ . We select  $k$  such that  $k = \omega(1)$  and simultaneously  $k = \mathcal{O}(\log N / \log \log N)$ . Then we have  $4^k k = N^{o(1)}$  and the  $k$  primes are of magnitude  $d = \mathcal{O}(N^{(3+o(1))/k}) = N^{o(1)}$  (we can choose  $k$  consecutive primes computed using Eratosthenes's sieve).

Overall  $|T| = N^{1+o(1)}$  and  $|\text{Alph}(T)| \leq k + 5 = \omega(1)$ . (One can obtain any alphabet up to  $\mathcal{O}(N)$  by appending distinct letters to  $T$ .) ◀

With the same argument for a constant-sized alphabet we obtain the following result.

► **Theorem 21.** *All positions that are centers of Ab-square factors in a length- $n$  string over an alphabet of size  $5 + k$ , for a constant  $k$ , cannot be computed in  $\mathcal{O}(n^{2 - \frac{6}{3+k} - \varepsilon})$  time, for a constant  $\varepsilon > 0$ , unless the 3SUM conjecture fails.*

## 5 Computing centers of Ab-squares for constant-sized alphabets

A set of vectors in  $[1, n]^d$  is called *monotone* if its elements can be ordered so that they form a monotone non-decreasing sequence on each coordinate.

► **Definition 22.** For sets  $\mathcal{A}$  and  $\mathcal{B}$  of vectors we define  $\mathcal{A} + \mathcal{B} = \{a + b : a \in \mathcal{A}, b \in \mathcal{B}\}$  and  $c \cdot \mathcal{A} = \{ca : a \in \mathcal{A}\}$ , and for a string  $W$  we define:  $P_{l,r}(W) = \{\text{Parikh}(W[1..k]) : l \leq k \leq r\}$ . Let us denote by  $|A|$  the length of a string corresponding to a Parikh vector  $A$ .

In the algorithm we use the following fact shown in [11]. The exact complexities can be found in [11, Theorem 3.1].

► **Fact 1** ([11]). Given three monotone sequences  $\mathcal{A}, \mathcal{B}, \mathcal{C}$  in  $[1, n]^d$  for a constant  $d$ , we can compute  $(\mathcal{A} + \mathcal{B}) \cap \mathcal{C}$  in  $\mathcal{O}(n^{2-\epsilon})$  expected time for a constant  $\epsilon > 0$ , or in  $\mathcal{O}(n^{2-\epsilon'})$  worst case time for a constant  $\epsilon' > 0$  if  $d \leq 7$ .

■ **Algorithm 1** CENTERS( $T$ ).

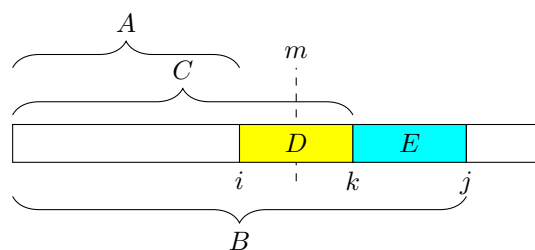
---

```

if  $|T| < 2$  return  $\emptyset$ ;
 $m = \lceil n/2 \rceil$ ;
 $\mathcal{A} = P_{0,m-1}(T)$ ;  $\mathcal{B} = P_{m,n}(T)$ ;  $\mathcal{C} = P_{0,n}(T)$ ;
 $\mathcal{M} = \{|C| : 2C \in (\mathcal{A} + \mathcal{B}) \cap 2 \cdot \mathcal{C}\}$ ;
 $T_{\text{left}} = T[1..m-1]$ ;  $T_{\text{right}} = T[m..n]$ ;
return  $\mathcal{M} \cup \text{CENTERS}(T_{\text{left}}) \cup \{k + m : k \in \text{CENTERS}(T_{\text{right}})\}$ 

```

---



■ **Figure 5**  $A \in \mathcal{A}$ ,  $B \in \mathcal{B}$ ,  $C \in \mathcal{C}$  denote Parikh vectors of the corresponding fragments. If  $A + B = 2C$ , then  $D = E$  and  $k = |C|$  is a center of an Ab-square.

► **Theorem 23.** For a string of length  $n$  over an alphabet of size  $d = \mathcal{O}(1)$ , we can compute centers of all Ab-squares and centers of all odd Ab-squares in expected time  $\mathcal{O}(n^{2-\epsilon})$  or in worst case time  $\mathcal{O}(n^{2-\epsilon'})$  if  $d \leq 7$ , for  $\epsilon > 0$ .

**Proof.** We use the above algorithm. Correctness of the algorithm is straightforward; see Figure 5. If

$$|A| < |B|, |C| = (|A| + |B|)/2, B = A + D + E, C = A + D$$

then  $A + B = 2C \iff 2A + D + E = 2A + 2D$ .

Consequently, after cancelling the same parts on both sides,  $A + B = 2C \iff E = D$ , equivalently if and only if the factor  $T[i..j]$  corresponding to  $DE$  is an Ab-square centred in  $k = |C|$ . The figure shows the case when  $k$  is in the right half of the strings; the other case is symmetric.

By Fact 1 the cost of the algorithm can be given by a recurrence

$$S(n) = 2 \cdot S\left(\frac{n}{2}\right) + \mathcal{O}(n^{2-\epsilon})$$

## 77:14 Hardness of Detecting Abelian and Additive Squares

which results in  $S(n) = \mathcal{O}(n^{2-\epsilon})$  for  $\epsilon > 0$ .

In case of of odd Ab-squares let

$$P_{l,r}^c(W) = \{\text{Parikh}(W[1..k]) : l \leq k \leq r, k \bmod 2 = c\}.$$

In the algorithm the statement  $\mathcal{M} = \{|C| : 2C \in (\mathcal{A} + \mathcal{B}) \cap 2 \cdot \mathcal{C}\}$  is executed for both  $c \in \{0, 1\}$ , with

$$\mathcal{A} = P_{0,m-1}^c(T), \quad \mathcal{B} = P_{m,n}^c(T), \quad \mathcal{C} = P_{0,n}^{1-c}(T).$$

Other parts of the algorithm, as well as its analysis, are essentially the same.  $\blacktriangleleft$

### 6 Detecting odd Ab-squares

Unfortunately the string  $T$  from Lemma 19 has many Ab-squares which are not well-placed. Our approach is to embed the (slightly) modified string  $T$  into a string which is a special composition of  $T$  and a combination of long quaternary Ab-square-free strings. The resulting string will fix the potential centers in specified locations. We use additional letters:  $\diamond$ ,  $\circ$  and  $0, \dots, 6$ .

We show first a fact useful in fixing Ab-squares in specified places (Lemma 26). Let  $P_{t-2}$  be any Ab-square-free string of length  $t - 2$  over alphabet  $\{3, 4, 5, 6\}$  (it can be constructed using Keränen's construction [27] of quaternary Ab-square-free strings). Let us define

$$U_{2t} = 0 P_{t-2} 1 2 P_{t-2}^R 0.$$

The following lemma is proved in the full version of the paper.

► **Lemma 24.** *The string  $(U_{2t})^m$  contains exactly the following Ab-squares:*

- (1) *of length divisible by  $4t$ ; and*
- (2) *with the center between two 0's and of all admissible even lengths other than  $(4q + 2)t$ , for an integer  $q \geq 0$ .*

For equal-length strings  $X, Y$  we define the string

$$\text{shuffle}_{\diamond}(X, Y) = X[1] \diamond Y[1] X[2] \diamond Y[2] X[3] \diamond Y[3] \dots$$

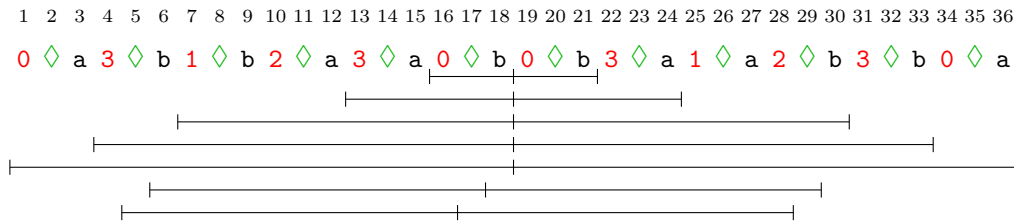
For example,  $\text{shuffle}_{\diamond}(\text{abc}, \text{ABC}) = \text{a} \diamond \text{Ab} \diamond \text{Bc} \diamond \text{C}$ .

The parity condition for half lengths of Ab-squares in the following observation justifies the usage of the additional letter  $\diamond$  in shuffle. Let  $U_{[X]}$  be the string resulting from  $U$  by removing all letters outside  $\text{Alph}(X)$ .

► **Observation 25.** *Assume  $X, Y$  are equal-length strings composed of disjoint sets of letters distinct from  $\diamond$  and  $W$  is an Ab-square in  $\text{shuffle}_{\diamond}(X, Y)$ . Then  $W_{[X]}, W_{[Y]}$  are Ab-squares in  $X, Y$ , respectively (we say that these Ab-squares are implied by  $W$ ). Moreover,  $|W_{[X]}|/2, |W_{[Y]}|/2, |W|/2$  are of the same parity.*

We say that an even-length factor of a string  $X$  is *centred at  $i$*  if it has its center between positions  $i$  and  $i + 1$  in  $X$ . By  $a \mid b$  and  $a \nmid b$  we denote that  $a$  divides  $b$  and  $a$  does not divide  $b$ . For an illustration of the following lemma, see Figure 6.

► **Lemma 26.** *Let  $X = (U_{2t})^{n-1}$ ,  $Y$  be a string of length  $|X|$  such that its alphabet is disjoint with  $\text{Alph}(X) \cup \{\diamond\}$ ,  $W = \text{shuffle}_{\diamond}(X, Y)$ , and let an integer  $\ell$  satisfy  $12t \nmid \ell$ . Then a length- $\ell$  factor of  $W$  is an Ab-square if and only if it is centred in  $W$  at  $r \equiv \{0, -1, -2\} \pmod{6t}$ ,  $Y$  contains an Ab-square factor of length  $\ell/3$  centred in  $Y$  at  $\lfloor r/3 \rfloor$ , and  $6t \nmid \ell$ .*



■ **Figure 6** Illustration of Lemma 26. Let  $X = (U_6)^2$ . The string  $Y = (\text{abba})^3$ , composed of black letters, contains many Ab-squares. However the string  $Z = \text{shuffle}_\diamond(X, Y)$  of length 36, shown above, contains only Ab-squares centred at 16, 17 or 18, as in the figure. The implied Ab-squares in  $Z$  are only those which are centred at positions 5 or 6 in  $Z$ .

**Proof.** By the disjointness of sets of letters in  $X, Y$  and  $\{\diamond\}$ , each Ab-square in  $W$  has length that is divisible by 3. The following claim is then readily verified (cf. Observation 25).

▷ **Claim 27.** For positive integer  $\ell$  such that  $6 \mid \ell$ , a length- $\ell$  factor of  $W$  centred at  $r$  is an Ab-square if and only if the length- $\ell/3$  factors in  $X$  and  $Y$  centred at  $\lfloor \frac{r+2}{3} \rfloor$  and  $\lfloor \frac{r}{3} \rfloor$ , respectively, are Ab-squares.

Let integer  $\ell > 0$  satisfy  $6 \mid \ell$  and  $12t \nmid \ell$ . We show two implications.

( $\Rightarrow$ ) If  $W$  contains an Ab-square factor of length  $\ell$  centred at some  $r$ , then the implied Ab-square factor of  $X$  has length  $\ell/3$ , where  $4t \nmid \ell/3$ , so by Lemma 24 it has its center between two 0's, i.e.,  $2t \mid \lfloor \frac{r+2}{3} \rfloor$ . Hence,  $r \equiv \{0, -1, -2\} \pmod{6t}$ .

Moreover,  $2t \nmid \ell/3$  also by Lemma 24. Finally, the implied Ab-square factor of  $Y$  indeed has length  $\ell/3$  and is centred at  $\lfloor r/3 \rfloor$ .

( $\Leftarrow$ ) Let  $r \equiv \{0, -1, -2\} \pmod{6t}$ ,  $6t \nmid \ell$ , and assume that  $Y$  contains an Ab-square factor of length  $\ell/3$  centred at  $\lfloor r/3 \rfloor$ . We have  $2t \mid \lfloor \frac{r+2}{3} \rfloor$  and  $2t \nmid \ell/3$ , so by Lemma 24 the string  $X$  contains an Ab-square factor of length  $\ell/3$  centred at  $\lfloor \frac{r+2}{3} \rfloor$ . Finally, the unary string  $\diamond^{2tm}$ , certainly contains an Ab-square factor of length  $\ell/3$  centred at  $\lfloor \frac{r+1}{3} \rfloor$ . By the claim,  $W$  contains an Ab-square of length  $\ell$  centred at  $r$  that implies the three Ab-squares. ◀

► **Theorem 28.** *Checking if a length- $n$  string over an alphabet of size  $\omega(1)$  contains an odd Ab-square is 3SUM-hard. Moreover, for a string over an alphabet of size  $14 + k$ , for a constant  $k$ , the same problem cannot be solved in  $\mathcal{O}(n^{2 - \frac{6}{3+k} - \epsilon})$  time, for a constant  $\epsilon > 0$ , unless the 3SUM conjecture fails.*

**Proof.** We use the technique of fixing Ab-squares from Lemma 26. Moreover, we make the following minor modifications upon the construction of Section 4:

- (1) Each fragment  $\mathbf{b}^{kd}$  is extended by one letter to  $\mathbf{b}^{kd+1}$ , and
- (2) the letters  $\bullet, \star$  are replaced each by two letters  $\bullet\circ, \star\circ$ , respectively.

Intuitively, (1) allows to extend Ab-squares considered in the proof of Lemma 19 by one letter  $\mathbf{b}$  to either side, and (2) makes  $|\mathbf{A}| = |\mathbf{B}| = |\mathbf{S}_i^j|$  even which facilitates the usage of Lemma 26 with  $Y = T$ . It can be verified by inspecting the proof that Lemma 19 still holds after these two changes. We refer to all the notions from Section 4 after these modifications.

It is enough now to show the following claim for  $X = (U_{2t})^{n-1}$ , where  $2t = |T|/(n-1)$ . We assume that  $n \geq 3$ .

▷ **Claim 29.** An odd-half instance of 3DAP is a YES-instance if and only if  $W = \text{shuffle}_\diamond(X, T)$  has an odd Ab-square factor.





2. Can one detect an additive square in a length- $n$  string over a constant-sized alphabet in  $\mathcal{O}(n^{2-\varepsilon})$  time, for some  $\varepsilon > 0$ ? We have shown 3SUM-hardness of this problem for an alphabet that is polynomial in  $n$ .

---

## References

- 1 Peyman Afshani, Ingo van Duijn, Rasmus Killmann, and Jesper Sindahl Nielsen. A lower bound for jumbled indexing. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 592–606. SIAM, 2020. doi:10.1137/1.9781611975994.36.
- 2 Amihod Amir, Alberto Apostolico, Tirza Hirst, Gad M. Landau, Noa Lewenstein, and Liat Rozenberg. Algorithms for jumbled indexing, jumbled border and jumbled square on run-length encoded strings. *Theoretical Computer Science*, 656:146–159, 2016. doi:10.1016/j.tcs.2016.04.030.
- 3 Amihod Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014. doi:10.1007/978-3-662-43948-7\_10.
- 4 Hideo Bannai, Shunsuke Inenaga, and Dominik Köppl. Computing all distinct squares in linear time for integer alphabets. In Juha Kärkkäinen, Jakub Radoszewski, and Wojciech Rytter, editors, *28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017, July 4-6, 2017, Warsaw, Poland*, volume 78 of *LIPICs*, pages 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.CPM.2017.22.
- 5 Ilya Baran, Erik D. Demaine, and Mihai Patrascu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008. doi:10.1007/s00453-007-9036-3.
- 6 Felix Adalbert Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences of the United States of America*, 32(12):331–332, 1946. doi:10.1073/pnas.32.12.331.
- 7 Tom C. Brown and Allen R. Freedman. Arithmetic progressions in lacunary sets. *Rocky Mountain Journal of Mathematics*, 17(3):587–596, 1987.
- 8 Tom C. Brown, Veselin Jungić, and Andrew Poelstra. On double 3-term arithmetic progressions. *Integers*, 14:A43, 2014. URL: <https://www.emis.de/journals/INTEGERS/papers/o43/o43.Abstract.html>.
- 9 Julien Cassaigne, James D. Currie, Luke Schaeffer, and Jeffrey O. Shallit. Avoiding three consecutive blocks of the same size and same sum. *Journal of the ACM*, 61(2):10:1–10:17, 2014. doi:10.1145/2590775.
- 10 Timothy M. Chan and Qizheng He. Reducing 3SUM to Convolution-3SUM. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA@SODA 2020, Salt Lake City, UT, USA, January 6-7, 2020*, pages 1–7. SIAM, 2020. doi:10.1137/1.9781611976014.1.
- 11 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.
- 12 Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Extracting powers and periods in a word from its runs structure. *Theoretical Computer Science*, 521:29–41, 2014. doi:10.1016/j.tcs.2013.11.018.
- 13 Larry J. Cummings and William F. Smyth. Weak repetitions in strings. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 24:33–48, 1997.

- 14 Bartłomiej Dudek, Paweł Gawrychowski, and Tatiana Starikovskaya. All non-trivial variants of 3-LDT are equivalent. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 974–981. ACM, 2020. doi:10.1145/3357713.3384275.
- 15 Roger C. Entringer, Douglas E. Jackson, and J.A. Schatz. On nonrepetitive sequences. *Journal of Combinatorial Theory, Series A*, 16(2):159–164, 1974. doi:10.1016/0097-3165(74)90041-7.
- 16 Paul Erdős. Some unsolved problems. *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 6:221–254, 1961.
- 17 Jeff Erickson. Finding longest arithmetic progressions, 1999. URL: <https://jeffe.cs.illinois.edu/pubs/arith.html>.
- 18 Aleksandr Andreevich Evdokimov. Strongly asymmetric sequences generated by a finite number of symbols. *Doklady Akademii Nauk SSSR*, 179(6):1268–1271, 1968.
- 19 Gabriele Fici, Filippo Mignosi, and Jeffrey O. Shallit. Abelian-square-rich words. *Theoretical Computer Science*, 684:29–42, 2017. doi:10.1016/j.tcs.2017.02.012.
- 20 Gabriele Fici and Aleksi Saarela. On the minimum number of abelian squares in a word. In Maxime Crochemore, James Currie, Gregory Kucherov, and Dirk Nowotka, editors, *Combinatorics and Algorithmics of Strings (Dagstuhl Seminar 14111)*, volume 4 (3), pages 34–35, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/DagRep.4.3.28.
- 21 Aviezri S. Fraenkel, Jamie Simpson, and Mike Paterson. On weak circular squares in binary words. In Alberto Apostolico and Jotun Hein, editors, *Combinatorial Pattern Matching, 8th Annual Symposium, CPM 97, Aarhus, Denmark, June 30 - July 2, 1997, Proceedings*, volume 1264 of *Lecture Notes in Computer Science*, pages 76–82. Springer, 1997. doi:10.1007/3-540-63220-4\_51.
- 22 Allen R. Freedman and Tom C. Brown. Sequences on sets of four numbers. *Integers*, 16:A33, 2016. URL: <http://math.colgate.edu/~integers/q33/q33.Abstract.html>.
- 23 Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPICs*, pages 45:1–45:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ESA.2016.45.
- 24 Dan Gusfield and Jens Stoye. Linear time algorithms for finding and representing all the tandem repeats in a string. *Journal of Computer and System Sciences*, 69(4):525–546, 2004. doi:10.1016/j.jcss.2004.03.004.
- 25 Lorenz Halbeisen and Norbert Hungerbühler. An application of van der Waerden’s theorem in additive number theory. *Integers*, 0:A7, 2000. URL: <http://math.colgate.edu/~integers/a7/a7.pdf>.
- 26 Veikko Keränen. A powerful abelian square-free substitution over 4 letters. *Theoretical Computer Science*, 410(38-40):3893–3900, 2009. doi:10.1016/j.tcs.2009.05.027.
- 27 Veikko Keränen. Abelian squares are avoidable on 4 letters. In Werner Kuich, editor, *Automata, Languages and Programming, ICALP 1992*, volume 623 of *Lecture Notes in Computer Science*, pages 41–52. Springer, 1992. doi:10.1007/3-540-55719-9\_62.
- 28 Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Maximum number of distinct and nonequivalent nonstandard squares in a word. *Theoretical Computer Science*, 648:84–95, 2016. doi:10.1016/j.tcs.2016.08.010.
- 29 Tomasz Kociumaka, Jakub Radoszewski, and Bartłomiej Wiśniewski. Subquadratic-time algorithms for abelian stringology problems. In Ilias S. Kotsireas, Siegfried M. Rump, and Chee K. Yap, editors, *Mathematical Aspects of Computer and Information Sciences - 6th International Conference, MACIS 2015, Berlin, Germany, November 11-13, 2015, Revised Selected Papers*, volume 9582 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2015. doi:10.1007/978-3-319-32859-1\_27.

- 30 Tomasz Kociumaka, Jakub Radoszewski, and Bartłomiej Wiśniewski. Subquadratic-time algorithms for abelian stringology problems. *AIMS Medical Science*, 4(3):332–351, 2017. doi:10.3934/ms.2017.3.332.
- 31 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287. SIAM, 2016. doi:10.1137/1.9781611974331.ch89.
- 32 Florian Lietard and Matthieu Rosenfeld. Avoidability of additive cubes over alphabets of four numbers. In Natasa Jonoska and Dmytro Savchuk, editors, *Developments in Language Theory - 24th International Conference, DLT 2020, Tampa, FL, USA, May 11-15, 2020, Proceedings*, volume 12086 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2020. doi:10.1007/978-3-030-48516-0\_15.
- 33 Giuseppe Pirillo and Stefano Varricchio. On uniformly repetitive semigroups. *Semigroup Forum*, 49:125–129, 1994. doi:10.1007/BF02573477.
- 34 Peter A. B. Pleasants. Non-repetitive sequences. *Mathematical Proceedings of the Cambridge Philosophical Society*, 68:267–274, 1970.
- 35 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.
- 36 Michaël Rao and Matthieu Rosenfeld. Avoiding two consecutive blocks of same size and same sum over  $\mathbb{Z}^2$ . *SIAM Journal on Discrete Mathematics*, 32(4):2381–2397, 2018. doi:10.1137/17M1149377.
- 37 Lawrence Bruce Richmond and Jeffrey O. Shallit. Counting abelian squares. *Electronic Journal of Combinatorics*, 16(1), 2009. URL: [http://www.combinatorics.org/Volume\\_16/Abstracts/v16i1r72.html](http://www.combinatorics.org/Volume_16/Abstracts/v16i1r72.html).
- 38 Raphaël Salem and Donald C. Spencer. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences of the United States of America*, 28(12):561–563, 1942. doi:10.1073/pnas.28.12.561.
- 39 Jamie Simpson. Solved and unsolved problems about abelian squares, 2018. arXiv:1802.04481.
- 40 Shiho Sugimoto, Naoki Noda, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Computing abelian string regularities based on RLE. In Ljiljana Brankovic, Joe Ryan, and William F. Smyth, editors, *Combinatorial Algorithms - 28th International Workshop, IWOCA 2017, Newcastle, NSW, Australia, July 17-21, 2017, Revised Selected Papers*, volume 10765 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2017. doi:10.1007/978-3-319-78825-8\_34.