MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

ZERO-SUM FREE SETS WITH SMALL SUM-SET

GAUTAMI BHOWMIK, IMMANUEL HALUPCZOK,
AND JAN-CHRISTOPH SCHLAGE-PUCHTA

ABSTRACT. Let A be a zero-sum free subset of $\mathbb{Z}_n$ with $|A| = k$. We compute for $k \leq 7$ the least possible size of the set of all subset-sums of A.

1. Introduction and Results

For an abelian group G and a subset B of G, we define the sum-set of B as $\Sigma(B) := \{\sum_{b\in C} b \mid C \subset B, C \neq \emptyset\}$ for the set of all subset sums of B (excluding the empty subset). We say that B is *zero-sum free* if $0 \notin \Sigma(B)$. In this note we are only interested in finite cyclic groups, and we write $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ for $n \geq 2$.

Define $f_n(k) = \min |\Sigma(B)|$, where B runs over all zero-sum free subsets of $\mathbb{Z}_n$ of cardinality k, and set $f(k) := \min_n f_n(k)$. If there are no zero-sum free sets of cardinality k in $\mathbb{Z}_n$, we set $f_n(k) = \infty$. This function arises naturally when considering the structure of zero-sum free sequences in $\mathbb{Z}_n$ with not too many repetitions. For example, Gao and Geroldinger [3] showed that a sequence $a_1, \ldots, a_m$ in $\mathbb{Z}_n$ with $m > \delta n$ contains a sub-sequence adding up to 0, provided that no element occurs in the sequence more than ϵn times, where ϵ is a constant depending only on δ, its precise value being determined by f_n.

The following proposition summarises everything which is already known about f (as far as we know).

Proposition 1.1. (1) *If $f_n(k) < \infty$, then $f_n(k) \leq \binom{k+1}{2}$.*
(2) *We have $f(1) = 1$, $f(2) = 3$, and if $n \geq 6$ then $f_n(3) = 5$ for n even and $f_n(3) = 6$ for n odd.*
(3) *We have $f(k) \geq 2k$ for $k \geq 4$, and $f(k) \geq \frac{1}{9}k^2$ for any k.*
(4) *If p is prime, then $f_p(k) \geq \min\{\binom{k+1}{2} - \delta, \frac{p+3}{2}\}$, where $\delta = \begin{cases} 0, & k \equiv 0\ (2) \\ 1, & k \equiv 1\ (2) \end{cases}$*

The first statement is obtained by taking $B = \{1, 2, \ldots, k\}$. Notice that in particular, if $n > \binom{k+1}{2}$ then B has no zero-sum subsets and $f_n(k) \leq \binom{k+1}{2}$. The second statement is straightforward. The third one is due to Eggleton and Erdős [2] and Olson [5, Theorem 3.2], respectively, and the fourth one is due to Olson [4, Theorem 2].

In this note we describe the computation of $f_n(k)$ for $k \leq 7$ and all n. For $k = 7$, it took 18 hours of CPU-time; the same algorithm solved $k = 6$ within 2 minutes. Hence, even if we only assume exponential growth of the running time,

2000 *Mathematics Subject Classification.* Primary 11B75; Secondary 11B50.
The second author was supported by the Fondation sciences mathématiques de Paris.

which appears somewhat optimistic, the case $k = 8$ would require some serious improvements of the algorithm.

Our main result is the following

Theorem 1.2. *We have $f(4) = 8$, $f(5) = 13$, $f(6) = 19$, and $f(7) = 24$.*

In fact, we computed $f_n(k)$ for $k \leq 7$ and all n; the results of these computations are listed in the following table.

How to read the table:

The last column gives an example of a set B of k elements which has no non-empty zero-sum subset and which has the number of different subset sums specified in the third column. The second column specifies the conditions on n for this example to work. Some of the examples of B are only specified for some fixed n_0; it is clear how to turn this into an example for any multiple of n_0.

Thus one gets: if the condition in the second column is satisfied, then $f_n(k)$ has at most the value given in the table. Using a computer we checked that there are no other examples making $f_n(k)$ smaller.

The boldface values in the third column are the values of $f(k)$.

k	cond. on n	$f_n(k)$	Example
2	$n \geq 4$	**3**	$\{1, 2\} \subset \mathbb{Z}_n$
3	$n \geq 6$ $2 \mid n$	**5**	$\{1, \frac{1}{2}n, \frac{1}{2}n + 1\} \subset \mathbb{Z}_n$
	$n \geq 7$	6	$\{1, 2, 3\} \subset \mathbb{Z}_n$
4	$9 \mid n$	**8**	$\{3, 1, 4, 7\} \subset \mathbb{Z}_9$
	$n \geq 10$ $2 \mid n$	9	$\{1, 2, \frac{1}{2}n, \frac{1}{2}n + 1\} \subset \mathbb{Z}_n$
	$n \geq 12$ $3 \mid n$	9	$\{1, \frac{1}{3}n, \frac{1}{3}n + 1, \frac{2}{3}n + 1\} \subset \mathbb{Z}_n$
	$n \geq 11$	10	$\{1, 2, 3, 4\} \subset \mathbb{Z}_n$
5	$n \geq 14$ $2 \mid n$	**13**	$\{1, 2, \frac{1}{2}n, \frac{1}{2}n + 1, \frac{1}{2}n + 2\} \subset \mathbb{Z}_n$
	$15 \mid n$	14	$\{-1, 2, 3, 4, 5\} \subset \mathbb{Z}_{15}$
	$n \geq 16$	15	$\{1, 2, 3, 4, 5\} \subset \mathbb{Z}_n$
6	$n \geq 20$ $2 \mid n$	**19**	$\{1, 2, 3, \frac{1}{2}n, \frac{1}{2}n + 1, \frac{1}{2}n + 2\} \subset \mathbb{Z}_n$
	$21 \mid n$	20	$\{-1, 2, 3, 4, 5, 6\} \subset \mathbb{Z}_{21}$
	$n \geq 22$	21	$\{1, 2, 3, 4, 5, 6\} \subset \mathbb{Z}_n$
7	$25 \mid n$	**24**	$\{5, 10, 1, 6, 11, 16, 21\} \subset \mathbb{Z}_{25}$
	$n \geq 26$ $2 \mid n$	25	$\{1, 2, 3, \frac{1}{2}n, \frac{1}{2}n + 1, \frac{1}{2}n + 2, \frac{1}{2}n + 3\} \subset \mathbb{Z}_n$
	$27 \mid n$	26	$\{1, -2, 3, 4, 5, 6, 7\} \subset \mathbb{Z}_{27}$
	$n \geq 30$ $3 \mid n$	27	$\{1, 2, \frac{1}{3}n, \frac{1}{3}n + 1, \frac{1}{3}n + 2, \frac{2}{3}n + 1, \frac{2}{3}n + 2\} \subset \mathbb{Z}_n$
	$n \geq 30$ $5 \mid n$	27	$\{1, \frac{1}{5}n, \frac{1}{5}n + 1, \frac{2}{5}n, \frac{2}{5}n + 1, \frac{3}{5}n + 1, \frac{4}{5}n + 1\} \subset \mathbb{Z}_n$
	$n \geq 29$	28	$\{1, 2, 3, 4, 5, 6, 7\} \subset \mathbb{Z}_n$

It is clear that $f_n(k)$ is either ∞ or less than n, so in particular $f_n(k) = \infty$ if $n \leq f(k)$. On the other hand, for any $n > f(k)$ the table does give an example which yields $f_n(k) < \infty$. Thus we get:

Corollary 1.3. *If $k \leq 7$, then $f_n(k) = \infty$ if and only if $n \leq f(k)$.*

When n is prime and large compared to $\binom{k+1}{2}$, then Proposition 1.1 (4) yields the precise value of $f_n(k)$ up to δ. The following corollary of the table suggests that for n prime, we have $f_n(k) = \binom{k+1}{2}$ even when n is not much bigger than $\binom{k+1}{2}$.

Corollary 1.4. *If $k \leq 7$ and p is prime, then $f_p(k) \geq \binom{k+1}{2}$.*

In fact, our main motivation to carry out these computations was that we needed this corollary ([1], Lemma 9) to deduce some general results on zero-sum free sequences in $\mathbb{Z}_p \times \mathbb{Z}_p$ and Proposition 1.1 (4) was not strong enough to deal with small primes.

2. Description of the algorithm

We are looking for an algorithm which, for a given k, determines $f_n(k)$ for all n. First we describe how to turn the problem into an algorithmically decidable one (i.e. how to treat infinitely many values of n simultaneously); then we shall describe how to reduce the amount of computation so as to solve the problem in real time.

Suppose first that k, ℓ and n are fixed and that we want to check whether there exists a zero-sum free set $B = \{b_1, \dots b_k\} \subset \mathbb{Z}_n$ consisting of k distinct elements such that $|\Sigma(B)| = \ell$. Such a set B yields an equivalence relation $\sim$ on the set of non-empty subsets of the index set $\{1, \dots, k\}$, defined by $I \sim I' \iff \sum_{i\in I} b_i = \sum_{i\in I'} b_i$, and this equivalence relation has precisely ℓ equivalence classes. Moreover, the elements b_i of B form a solution of the system of linear equations and inequations $\mathcal{E}(\sim)$ (in the variables $x_1, \dots, x_k$), which we define as follows:

(1) For each $i \neq j$, take the inequation $x_i \neq x_j$.
(2) For each $I \subset \{1, \dots, k\}$, $I \neq \emptyset$, take the inequation $\sum_{i\in I} x_i \neq 0$.
(3) For each pair $I, I' \subset \{1, \dots, k\}$, take $\sum_{i\in I} x_i = \sum_{i\in I'} x_i$ or $\sum_{i\in I} x_i \neq \sum_{i\in I'} x_i$, depending on whether $I \sim I'$ or not.

On the other hand, any solution in $\mathbb{Z}_n$ of this system $\mathcal{E}(\sim)$ defines a set B solving the original problem.

Note that the coefficients of $\mathcal{E}(\sim)$ are elements of $\mathbb{Z}$ and that $\mathcal{E}(\sim)$ does not depend at all on n, thus to determine $f_n(k)$ simultaneously for all n, we can proceed as follows. Suppose k is given. Iterate through all equivalence relations $\sim$ (on the set of non-empty subset of $\{1, \dots, k\}$) and for each one consider the corresponding system $\mathcal{E}(\sim)$. Determine the set of n such that $\mathcal{E}(\sim)$ has a solution modulo n. (We still have to describe how to do this.) Then we compute

$$f_n(k) = \min\{\text{number of equivalence classes of } \sim \mid \mathcal{E}(\sim) \text{ has a solution modulo } n\}.$$

To determine the set of n for which $\mathcal{E}(\sim)$ has a solution modulo n, the idea is that every such solution yields a solution of $\mathcal{E}(\sim)$ in $\mathbb{Q}/\mathbb{Z}$ by dividing all variables by n. Thus we first determine all solutions of $\mathcal{E}(\sim)$ in $\mathbb{Q}/\mathbb{Z}$; such a solution then yields a solution in $\mathbb{Z}_n$ if and only if multiplying by n removes all denominators. In other words, we embed the groups $\mathbb{Z}_n$ in $\mathbb{Q}/\mathbb{Z}$ and then check which solutions in $\mathbb{Q}/\mathbb{Z}$ already lie in $\mathbb{Z}_n$.

To solve $\mathcal{E}(\sim)$ over $\mathbb{Q}/\mathbb{Z}$, the algorithm proceeds as follows. First it considers only the equations. It brings them into upper triangular form by working over $\mathbb{Z}$. (Thus the original system and the triangular one will have the same set of solutions in any abelian group.) Then we use the equations to successively determine some of the variables as linear functions in the other ones; in $\mathbb{Q}/\mathbb{Z}$, this works as follows. Suppose we are using the equation $a_i x_i + \sum_{j>i} a_j x_j = 0$ to determine x_i (without loss $a_i > 0$). In $\mathbb{Q}/\mathbb{Z}$, there are a_i different solutions to this:

$$x_i = \frac{1}{a_i}\Big(-\sum_{j>i} a_j x_j + \ell_i \Big) \quad \text{where } \ell_i \in \{0, 1, \dots a_i - 1\}.$$

After treating all variables, for each tuple $(\ell_i)_i$ we get a linear expression of each non-free variable in the free ones. In the remainder of the algorithm, each tuple $(\ell_i)_i$ is treated separately.

Now we symbolically plug these linear expressions into the inequations $L \neq R$ of $\mathcal{E}(\sim)$. If one gets identically $L = R$, then this is not a solution of $\mathcal{E}(\sim)$ in $\mathbb{Q}/\mathbb{Z}$; if on the other hand one does not get identically $L = R$ for any of the inequations, then almost all values in $\mathbb{Q}/\mathbb{Z}$ for the free variables x_i yield a solution of $\mathcal{E}(\sim)$ in $\mathbb{Q}/\mathbb{Z}$, which means that by multiplying by appropriate n, we find solutions in $\mathbb{Z}_n$. The computer prints all those solutions of $\mathcal{E}(\sim)$ in $\mathbb{Q}/\mathbb{Z}$ (the linear expressions of the non-free variables in the free ones), and we manually check the necessary conditions on n to make the example work.

Here is an example program output. For $k = 5$, one solution of a system corresponding to 13 equivalence classes was $x_1 = \frac{1}{2} + x_4, x_2 = \frac{1}{2} + \frac{1}{2}x_4, x_3 = \frac{1}{2}x_4, x_4 = x_4, x_5 = \frac{1}{2}$. Choosing $x_4 = \frac{2}{n}$ and then multiplying the whole solution by n yields a solution of the equations in $\mathbb{Z}_n$, provided that n is even. (For odd n, we do not get any solution due to $x_5 = \frac{1}{2}$.) To check that the inequations are satisfied, too, it suffices to verify that the resulting set $B = \{x_1, \dots, x_5\}$ indeed consists of k different elements and is zero-sum free, and that the sumset $\Sigma(B)$ has cardinality 13; this is the case when $n \geq 14$. Thus we get the line $k = 5, n \geq 14, 2|n$ in the table.

The problem is now finite, but the number of systems of equations which we have to consider is of magnitude the number of equivalence relations on a set of 2^k elements, that is, even for $k = 4$ we would have to check about 10^{10} cases. Since each single case requires a considerable amount of computation, this would already stretch our resources. We now describe how to make the algorithm faster.

First note that by Proposition 1.1 (1), it suffices to consider equivalence relations with at most $\ell_{\max} = \frac{k(k+1)}{2} - 1$ equivalence classes.

Next, we reduce the number of times we have to solve $\mathcal{E}(\sim)$ over $\mathbb{Q}/\mathbb{Z}$: whenever $\mathcal{E}(\sim)$ contains an inequation $L \neq R$ such that $L = R$ lies in the $\mathbb{Z}$-lattice generated by the equations in $\mathcal{E}(\sim)$, then $\mathcal{E}(\sim)$ is unsolvable in any abelian group, so we may skip the remainder of the computation in this case. Call the system $\mathcal{E}(\sim)$ an *almost-example* if we don't skip it.

(Note that checking the almost-example condition is not sufficient to prove that a system of equations and inequations has solutions modulo some n; for example, $\mathcal{E} = \{x_1 \neq 0, x_2 \neq 0, x_1 \neq x_2, 2x_1 = 2x_2 = 0\}$ has no solution modulo any n, but none of $x_1 = 0$, $x_2 = 0$, $x_1 = x_2$ lies in $\langle 2x_1 = 0, 2x_2 = 0\rangle_{\mathbb{Z}}$.)

It turned out that the number of almost-examples is very small. For example, for $k = 7$ and $\ell \leq 27$ there are only 19 of them (up to permutation of the set B), so there is no need to optimise any part of the algorithm treating the almost-examples. And even though the search for almost-examples finds some almost-examples in several different shapes given by permutations of B, removing duplicates takes almost no time compared to the main search. Thus in what follows, we only describe how to optimize the search for almost-examples.

The program starts with a system $\mathcal{E}$ consisting only of the inequations (1) and (2). Then it recursively adds equations and inequations of the form (3) to $\mathcal{E}$. (For each pair $I, I' \subset \{1, \dots k\}$, it separately tries the corresponding equation and the corresponding inequation.) As soon as $\mathcal{E}$ gets inconsistent, the program stops in this branch; by "inconsistent" we mean, as described above, that there exists an

inequation whose negation lies in the lattice generated by the equations. If we manage to treat all pairs $I, I' \subset \{1, \dots k\}$ without $\mathcal{E}$ becoming inconsistent, then the resulting system corresponds to an equivalence relation $\sim$ and it is an almost-example.

The program also stops if it can easily prove that the final equivalence relation will have more than $\ell_{\max}$ equivalence classes. To this end, it searches for a maximal anti-clique using a greedy approach. Start with an empty anti-clique $\mathcal{A}$. Iterate through all subsets $I \subset \{1, \dots, k\}$. If for all $I' \in \mathcal{A}$, $\mathcal{E}$ is inconsistent with the equation $\sum_{i \in I} x_i = \sum_{i \in I'} x_i$, then add I to $\mathcal{A}$. The cardinality of the set $\mathcal{A}$ obtained in this way is a lower bound for the number equivalence classes we will finally get. Whether this method yields good bounds heavily depends on the order in which the subsets I are considered. We will describe the order below.

We can greatly reduce the computation time by exploiting symmetry coming from permutations of the elements of B. We use the following general method: we fix, once and for all, a totally ordered set Γ, and for each complete system $\mathcal{E}$ we choose a function $v_{\mathcal{E}} \colon \{1, \dots, k\} \to \Gamma$ such that $v_{\sigma(\mathcal{E})}(\sigma(i)) = v_{\mathcal{E}}(i)$ for any permutation $\sigma \in S_k$. (Here, $\sigma(\mathcal{E})$ means that the variables are permuted.) We may then restrict our search to those $\mathcal{E}$ for which $v_{\mathcal{E}}$ is (weakly) increasing. During the computation, the program computes lower and upper bounds for the values $v_{\mathcal{E}}(i)$ (for $1 \leq i \leq k$) and stops whenever these bounds imply that $v_{\mathcal{E}}$ will not be increasing.

The function $v_{\mathcal{E}}$ which we use is:

$$\begin{aligned} v_{\mathcal{E}}(i) = (&\text{number of equations in } \mathcal{E} \text{ of the form } x_i = a + b, \\ &\text{number of equations in } \mathcal{E} \text{ of the form } a = x_i + b, \\ &\text{number of equations in } \mathcal{E} \text{ of the form } x_i + a = b + c) \in \mathbb{N}^3. \end{aligned}$$

We use the lexicographical order on these tuples, where the first entry is the most significant one.

It is important to choose a good order in which to try to add equations and inequations during the recursion, so that we get contradictions as early as possible. A good approach is to start with equations between one and two element sums: on the one hand, such equations imply a lot of other equations. On the other hand, if only few such equations exist, then we already get a lot of different sums, which is helpful to prove that there are two many equivalence classes. Therefore, the program first treats the equations of the form $x_i = x_{i'} + x_{i''}$, then the equations of the form $x_i + x_{i'} = x_{i''} + x_{i'''}$, and the remaining ones only afterwards. Another advantage of this order is that we early get good bounds for $v_{\mathcal{E}}$ and thus are able to apply the symmetry conditions.

Now we can explain the order in which the above anti-clique $\mathcal{A}$ is built: as we expect to have a lot of inequations between one and two element sums, the program tries these sums first when constructing the anti-clique.

Finally, we mention some of the data structures used to work more efficiently with $\mathcal{E}$.

- Do not perform any consistency check of $\mathcal{E}$ with another equation or inequation twice; always store the old results.
- Keep track of the equivalence relation defined by the equations which we already added to $\mathcal{E}$. If I and I' are equivalent, then concerning consistency

of $\mathcal{E}$ we do not need to distinguish between I and I', i.e. we are able to use remembered consistency results more often.

Also update the equivalence relation when we accidentally stumble over an equation which follows from $\mathcal{E}$.

- Each time an equation is added to $\mathcal{E}$, immediately put the system of equations into upper triangular form; this allows us to check quickly whether an inequation lies in the $\mathbb{Z}$-lattice generated by the equations.

References

1. G. Bhowmik, I. Halupczok, J.-C. Schlage-Puchta, The structure of maximal zero-sum free Sequences, *Acta Arith.* to appear.
2. R. B. Eggleton, P. Erdős, Two combinatorial problems in group theory, *Acta Arith.* **21** (1972), 111-116.
3. W. Gao, A. Geroldinger, On the structure of zerofree sequences, *Combinatorica* **18** (1998), 519–527.
4. J. E. Olson, An addition theorem modulo p, *J. Combinatorial Theory* **5** (1968), 45–52.
5. J. E. Olson, Sums of sets of group elements, *Acta Arith.* **28** (1975), 147–156.

Université de Lille 1, Laboratoire Paul Painlevé UMR CNRS 8524, 59655 Villeneuve d'Ascq Cedex, France
E-mail address: bhowmik@math.univ-lille1.fr

Institut für Mathematische Logik und Grundlagenforschung, Universität Münster, Einsteinstrasse 62, 48149 Münster, Germany
E-mail address: math@karimmi.de

Mathematisches Institut, Eckerstr. 1, 79104 Freiburg, Germany
E-mail address: jcp@math.uni-freiburg.de