

Finding Large 3-free Sets I: The Small n Case

William Gasarch^a

^a*University of Maryland at College Park, Department of Computer Science,
College Park, MD 20742*

James Glenn^b

^b*Loyola College in Maryland, Dept. of Computer Science, 4501 N. Charles St,
Baltimore, MD 21210*

Clyde P. Kruskal^c

^c*University of Maryland at College Park, Department of Computer Science,
College Park, MD 20742*

Abstract

There has been much work on the following question: given n , how large can a subset of $\{1, \dots, n\}$ be that has no arithmetic progressions of length 3. We call such sets *3-free*. Most of the work has been asymptotic. In this paper we sketch applications of large 3-free sets, present techniques to find large 3-free sets of $\{1, \dots, n\}$ for $n \leq 250$, and give empirical results obtained by coding up those techniques. In the sequel we survey the known techniques for finding large 3-free sets of $\{1, \dots, n\}$ for large n , discuss variants of them, and give empirical results obtained by coding up those techniques and variants.

Key words: 3-free sets, Arithmetic Sequence, Arithmetic Progression, van der Waerden's Theorem, non-averaging sets

Contents

| | | |
|-----|-----------------------|---|
| 1 | Introduction | 3 |
| 1.1 | Historical Background | 3 |

Email addresses: gasarch@cs.umd.edu (William Gasarch), jglenn@cs.loyola.edu (James Glenn), kruskal@cs.umd.edu (Clyde P. Kruskal).

| | | |
|------|---|----|
| 2 | Our Results and A Helpful Fact | 6 |
| 3 | Applications | 7 |
| 3.1 | The Diagonal Queens Domination Problem | 7 |
| 3.2 | Matrix Multiplication | 7 |
| 3.3 | Application to Communication Complexity | 8 |
| 3.4 | Linearity Testing | 9 |
| 4 | The Base 3 Method | 9 |
| 5 | Simple Upper Bounds via Splitting | 12 |
| 6 | Exact Values via Backtracking | 13 |
| 6.1 | Basic Backtracking Algorithms | 13 |
| 6.2 | Backtracking Algorithm with Information | 15 |
| 6.3 | Testing if a string is 3-free | 20 |
| 6.4 | Coding Techniques to Speed up our Program | 22 |
| 6.5 | Empirical Results | 22 |
| 7 | Upper Bounds via Linear Programming | 23 |
| 8 | Lower Bounds via Thirds Method | 26 |
| 9 | Other methods | 27 |
| 9.1 | The Concatenation Method | 28 |
| 9.2 | The Greedy Vertex Cover Method | 28 |
| 9.3 | The Randomization Method | 30 |
| 10 | The Values of $sz(n)$ for Small n | 30 |
| 11 | Acknowledgments | 30 |
| 12 | Appendix I: Comparison to Known Results | 31 |
| 12.1 | Compare to First Website | 31 |

| | | |
|------|---|----|
| 12.2 | The Second and Third Website | 32 |
| 13 | Appendix II: Tables for Small n | 33 |
| 14 | Appendix III: Comparing Methods For Finding Large 3-free Sets | 36 |
| 15 | Appendix IV: When did Linear Programming Help? | 46 |

1 Introduction

1.1 Historical Background

The motivation for this paper begins with van der Waerden's theorem:

Definition 1

- (1) Let $[n]$ be the set $\{1, \dots, n\}$.
- (2) An arithmetic progression of length k is a sequence of numbers of the form

$$a, a + d, a + 2d, \dots, a + (k - 1)d.$$

- (3) A k -AP is an arithmetic progression of length k .

Theorem 2 (van der Waerden (41) but see also (18)) For all k , for all c , there exists $W(k, c)$ such that for all c -colorings of $[W(k, c)]$ there exists a monochromatic k -AP.

The numbers $W(k, c)$ are called *van der Waerden numbers*. In the original proof of van der Waerden's theorem the upper bounds on $W(k, c)$ were quite large. Erdos and Turan (13) wanted smaller upper bounds on $W(k, c)$. They made a conjecture that would imply van der Waerden's theorem, hoping that a proof of this conjecture would yield smaller upper bounds. They conjectured the following:

For every $k \in \mathbb{N}$, $\lambda > 0$, for large enough n , for every $A \subseteq [n]$

$$|A| \geq \lambda n \Rightarrow A \text{ has a } k\text{-AP.}$$

The $k = 3$ case of this conjecture was originally proven by Roth (18; 30; 31) using analytic means. The $k = 4$ case was proven by Szemerédi (18; 38) (see also Gowers' proof (16)) by a combinatorial argument. Szemerédi (39) later proved the whole conjecture with a much harder proof. His proof used van der Waerden's Theorem and hence did not provide smaller

bounds on the van der Waerden numbers. Furstenberg (14) provided a very different proof using Ergodic theory. His proof was nonconstructive and hence provided no upper bounds on the van der Waerden numbers. Gowers (17) provided an analytic proof that yielded much smaller upper bounds for the van der Waerden Numbers.

We are concerned with the $k = 3$ case.

Roth's theorem (30) (but see also (18)) is as follows:

For all λ , for large enough n , for all $A \subseteq [n]$,

$$|A| \geq \lambda n \Rightarrow A \text{ has a 3-AP .}$$

Roth later (31) improved his result:

For all n , for all $A \subseteq [n]$,

$$|A| \geq \Omega\left(\frac{n}{\log \log n}\right) \Rightarrow A \text{ has a 3-AP .}$$

Szemerédi (40) (but see also (20)) and Heath-Brown (24) proved the following:

There exists c such that, for all n , for all $A \subseteq [n]$,

$$|A| \geq \Omega\left(\frac{n}{(\log n)^c}\right) \Rightarrow A \text{ has a 3-AP.}$$

Szemerédi obtained $c = 1/20$. Bourgain (5) (but see also (19)) has shown that, for all ϵ , $c = \frac{1}{2} - \epsilon$ works. In the same paper he showed the following stronger result:

For large enough n , for all $A \subseteq [n]$,

$$|A| \geq \Omega\left(n \sqrt{\frac{\log \log n}{\log n}}\right) \Rightarrow A \text{ has a 3-AP.}$$

The theorems stated above are all refinements of the following statement:

if $A \subseteq [n]$ is 'large enough' then A has a 3-AP.

The question arises, how large can $A \subseteq [n]$ be without having a 3-AP? We give a brief history of the known results in order of increasing quality (of the bounds), which differs from chronological order. Proof sketches of all the results stated here will be given in the sequel.

The following Theorem appeared in (13) but they do not take credit for it; hence we can call it folklore. We will describe it in Section 4.

For all n

$$(\exists A \subseteq [n])[A \text{ has no 3-AP and } |A| \geq \Omega(n^{\log_3 2}) \sim \Omega(n^{0.63})]$$

The following theorem was proven by Rozsa (32) in 1999 and is not as good as results obtained earlier; however, it is of some interest as will be described in the sequel.

For every $\epsilon > 0$, for all n ,

$$(\exists A \subseteq [n])[A \text{ has no 3-AP and } |A| \geq \Omega(n^{(\log_3 3)^{-\epsilon}}) \sim \Omega(n^{0.68})].$$

The following theorem was proven by Salem and Spencer (35).

For every $\epsilon > 0$, for all n ,

$$(\exists A \subseteq [n])[A \text{ has no 3-AP and } |A| \geq \Omega(n^{1-\frac{1+\epsilon}{\lg \lg n}})].$$

Behrend (2) has the best result currently. Moser (28) obtained the same result in a slightly different way that he claims is more constructive than Behrend's method.

There exists a constant c such that, for all n ,

$$(\exists A \subseteq [n])[A \text{ has no 3-AP and } |A| \geq \Omega(n^{1-c/\sqrt{\log n}})].$$

We introduce terminology we will use throughout this paper and state the above theorems using it.

Definition 3 For $k \in \mathbb{N}$, a set A is k -free if it does not have any arithmetic progression of size k .

Definition 4 Let $\text{sz}(n)$ be the maximum size of a 3-free subset of $[n]$. ('sz' stands for Szemerédi.)

Combining the results of Bourgain and Behrend mentioned above we have the following: There exist constants c_1, c_2, c such that, for all n ,

$$c_1 n^{1-c/\sqrt{\log n}} \leq \text{sz}(n) \leq c_2 n \sqrt{\frac{\log \log n}{\log n}}.$$

In this paper we discuss and implement techniques for finding exact values, and upper and lower bounds, on $\text{sz}(n)$ for $n \leq 250$. We obtain the following.

- (1) Exact values of $\text{sz}(n)$ for $1 \leq n \leq 187$.
- (2) Upper and lower bounds for $\text{sz}(n)$ for $188 \leq n \leq 250$.

Prior empirical studies have been done by Erdos and Turan (13), Wagstaff (42), and Wroblewski (43). Erdos and Turan (13) computed $\text{sz}(n)$ for $1 \leq n \leq 21$. Wagstaff (42) computed $\text{sz}(n)$ for $1 \leq n \leq 52$ (and also looked at 4-free and 5-free sets). Wroblewski (43) has on his website, in different terminology, the values of $\text{sz}(n)$ for $1 \leq n \leq 150$, and has lower bounds for $n \leq 25, 958$. We compute $\text{sz}(n)$ for $1 \leq n \leq 187$ and get close (but not matching) upper and lower bounds for $188 \leq n \leq 250$. We also obtain new lower bounds on $\text{sz}(n)$ for three values of n . Since Wroblewski's website uses a different notation than our paper we discuss the comparison in Appendix I.

2 Our Results and A Helpful Fact

Section 3 provides a short summary of how 3-free sets have been used in mathematics and computer science. Section 4 describes *The Base 3 Method* for obtaining large (though not optimal) 3-free sets. Section 5 describes *The Splitting Method* for obtaining upper bounds on $\text{sz}(n)$. Both the Base 3 method and the Splitting Method are easy; the rest of our methods are more difficult. Section 6 describes a backtracking method for obtaining $\text{sz}(n)$ exactly. It is used to obtain all of our exact results. Section 7 describes how to use linear programming to obtain upper bounds on $\text{sz}(n)$. All of our upper bounds on $\text{sz}(n)$ come from a combination of splitting and linear programming. Section 8 describes *The Thirds Method* for obtaining large 3-free sets. It is used to obtain all of our large 3-free sets beyond where backtracking could obtain exact answers. Section 9 describes methods for obtaining large 3-free sets whose results have been superseded by backtracking and the Thirds method; nevertheless, they may be useful at a later time. Section 10 describes our empirical results. The results themselves are in Appendices 1,2,3, and 4.

In the sequel we will summarize and unify several known methods for obtaining large 3-free sets of $[n]$ when n is large and give the results of empirical studies. We will also look at Roth's theorem empirically to obtain upper bounds on $\text{sz}(n)$.

The next two facts are trivial to prove; however, since we use them throughout the paper we need a shorthand way to refer to it:

Fact 5 *Let $x < y < z$. Then x, y, z is a 3-AP iff $x + z = 2y$.*

Fact 6 *If A is 3-free and c is a constant then $A + c = \{x + c \mid x \in A\}$ is 3-free.*

In light of Fact 5 3-free sets are sometimes called *non-averaging sets*. For example Wroblewski (43) and Moser (28) use the term.

3 Applications

We sketch four applications of 3-free sets. The first is a combinatorics problem about chess and the other three are applications in theoretical computer science.

3.1 The Diagonal Queens Domination Problem

How many queens do you need to place on an $n \times n$ chess board so that every square is either occupied or under attack? How many queens do you need if you insist that they are on the main diagonal? The former problem has been studied in (21) and the latter in (7). It is the diagonal problem that is connected to 3-free sets.

Theorem 7 *Let $\text{diag}(n)$ be the minimal number of queens needed so that they can be placed on the main diagonal of an $n \times n$ chessboard such that every square is either occupied or under attack. Then, for $n \geq 2$, $\text{diag}(n) = n - \text{sz}(\lceil n/2 \rceil)$.*

A more natural version of the problem is to ask “how many queens do you need to place on the main diagonal so that every non-diagonal square is under attack?” Using this version, Theorem 7 holds for all $n \geq 0$.

This paper will give exact values for $\text{sz}(n)$ for $1 \leq n \leq 187$, and hence exact values for $\text{diag}(n)$ for $1 \leq n \leq 374$. The theorems listed in Section 1 (also surveyed in the sequel paper) imply that, for large n , you need ‘close to’ n queens.

3.2 Matrix Multiplication

It is easy to multiply two $n \times n$ matrices in $O(n^3)$ steps. Strassen showed how to lower this to $O(n^{2.87})$ (37) (as described in many algorithms textbook, e.g. (10; 12; 25; 27; 29)). The basis of this algorithm is a way to multiply two 2×2 matrices using only 7 multiplications (but 18 additions). The best matrix multiplication algorithm known takes $O(n^{2.36})$ steps (9). It uses 3-free sets to guide the multiplication of smaller matrices. The algorithm is quite complicated.

The algorithm needs 3-free sets of size $n^{1-o(1)}$. The theorems listed in Section 1 imply that such sets exist. Unfortunately larger 3-free sets will not lead to better matrix multiplication

algorithms. However, larger sets that satisfy other combinatorial properties will lower the matrix multiplication exponent. See (8).

3.3 Application to Communication Complexity

Definition 8 Let f be any function from $\{0, 1\}^L \times \{0, 1\}^L \times \{0, 1\}^L$ to $\{0, 1\}$.

- (1) A protocol for computing $f(x, y, z)$, where Alice has x, y , Bob has x, z , and Carol has y, z , is a procedure where they take turns broadcasting information until they all know $f(x, y, z)$. (This is called ‘the forehead model’ since we can think of Alice having z on her forehead, Bob having y on his forehead, and Carol having x on her forehead. Everyone can see all foreheads except his or her own.)
- (2) Let $d_f(L)$ be the number of bits transmitted in the optimal deterministic protocol for f . This is called the multiparty communication complexity of f . (The literature usually denotes $d_f(L)$ by $d(f)$ with the L being implicit.)

Definition 9 Let $L \in \mathbf{N}$. For the function we are about to define, we view elements of $\{0, 1\}^L$ as L -bit numbers in base 2. Let $f : \{0, 1\}^L \times \{0, 1\}^L \times \{0, 1\}^L \rightarrow \{0, 1\}$ be defined as

$$f(x, y, z) = \begin{cases} 1 & \text{if } x + y + z = 2^L; \\ 0 & \text{otherwise.} \end{cases}$$

The multiparty communication complexity of f was studied by (6) (see also (26) and (3)). They used it as a way of studying branching programs. A careful analysis of the main theorem of (6) yields the following.

Theorem 10 Let f be the function in Definition 9.

(1)

$$d_f(L) = O\left(\log\left(\frac{L2^L}{\text{sz}(2^L)}\right)\right).$$

(2) It is known that $\text{sz}(2^L) \geq 2^{L-c\sqrt{L}}$ ((2) or the sequel). Hence $d_f(L) \leq O(\sqrt{L})$.

Using the sphere method to generate large 3-free sets (see (2) or the sequel) the protocol’s complexity is asymptotically $2\sqrt{2L}$. Empirically (3) the complexity is bounded above by $3.1\sqrt{L}$.

3.4 Linearity Testing

One ingredient in the proofs about probabilistically checkable proofs (PCPs) has been linear testing (1; 34). Let $\text{GF}(2^n)$ be the finite field on 2^n elements (GF stands for ‘Galois Field’). Given a black box for a function $f : \text{GF}(2^n) \rightarrow \mathbb{Z}_2$ we want to test if it is linear. One method, first suggested by (4), is to pick $x, y \in \text{GF}(2^n)$ at random and see if $f(x + y) = f(x) + f(y)$. This test can be repeated to reduce the probability of error.

We want a test that, for functions f that are ‘far from’ linear, will make fewer queries to obtain the same error rate. The quantity $d(f)$ (different notation from the $d_f(L)$ in the last section) is a measure of how nonlinear f is. The more nonlinear f is, the smaller $d(f)$ is (see (36; 23)).

In (36) the following was suggested: Let $G = (V, E)$ be a graph on k vertices. For every $v \in V$ pick $\alpha(v) \in \text{GF}(2^n)$ at random. For each $(u, v) \in E$ test if $f(\alpha(u) + \alpha(v)) = f(\alpha(u)) + f(\alpha(v))$. Note that this test makes k random choices from $\text{GF}(2^n)$ and $|E|$ queries. In (36) they showed that, using this test, the probability of error is $\leq 2^{-|E|} + d(f)$.

In (23) a graph is used that obtains probability of error $\leq 2^{-k^2 - o(1)} + d(f)^{k^{1 - o(1)}}$. The graph uses 3-free sets. It is a bipartite graph (X, Y, E) such that the following happens.

- There exists a partition of $X \times Y$ into $O(k)$ sets of the form $X_i \times Y_i$. We denote these $X_1 \times Y_1, X_2 \times Y_2, \dots, X_k \times Y_k$.
- For all i , the graph restricted to $X_i \times Y_i$ is a matching (i.e., it is a set of edges that do not share any vertices).

This is often expressed by saying that the graph is the union of $O(k)$ induced matchings.

We reiterate the construction of such a graph from (23) (which is reiterated from (33)). Let $A \subseteq [k]$ be a 3-free set. Let $G(A)$ be the bipartite graph on vertex sets $U = [3k]$ and $V = [3k]$ defined as the union over all $i \in [k]$ of $M_i = \{(a + i, a + 2i) \mid a \in A\}$. One can check that each M_i is an induced matching.

4 The Base 3 Method

Imagine trying to generate a 3-free set using the greedy method. This means you would go through the numbers and put one in if it does not cause a 3-free set. If you do this for the first 27 numbers you get the following: $\{1, 2, 4, 5, 9, 10, 12, 13, 27\}$. What do these numbers all have in common? If you express them in base 3 then their digits will be 0 and 1 (never 2). This motivates the *Base 3 Method*.

Throughout this section $\text{sz}(n)$ will be the largest 3-free set of $\{0, \dots, n\}$ instead of $\{1, \dots, n\}$.

The following method appeared in (13) but they do not take credit for it; hence we can call it folklore. Let $n \in \mathbf{N}$. Let

$$A_n = \{m \mid 0 \leq m \leq n \text{ and all the digits in the base 3 representation of } m \text{ are in the set } \{0, 1\}\}.$$

We will later show that A_n is 3-free and $|A_n| \approx 2^{\log_3 n} = n^{\log_3 2} \approx n^{0.63}$.

Example: Let $n = 116 = 1 \times 3^4 + 1 \times 3^3 + 0 \times 3^2 + 2 \times 3^1 + 2 \times 3^0$. Hence n in base 3 is 11022. We list the elements of A_{116} in several parts.

(1) The elements of A_{116} that have a 1 in the fifth place (coefficient of 3^4) are

$$\{10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011\}.$$

This has the same cardinality as the set obtained by subtracting 3^4 from all of the elements (removing the leading 1):

$$\{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011\}.$$

which is A_{116-3^4} .

(2) The elements of A_{116} that have a 0 in the fifth place are the 2^4 numbers

$$\{0000, 0001, \dots, 1111\}.$$

The above example illustrates one case of how to compute the size of A_n . If n has k digits in base 3 then there are clearly 2^{k-1} elements in A_n that have 0 in the k th place (the coefficient of 3^{k-1}). How many elements of A_n have a 1 in the k th place? In the case above it is $|A_{n-3^{k-1}}|$. This is not a general formula as the next example shows.

Example: Let $n = 113 = 1 \times 3^4 + 2 \times 3^3 + 1 \times 3^2 + 1 \times 3^1 + 2 \times 3^0$. Hence n in base 3 is 12112. We list the elements of A_{113} in several parts.

(1) The elements of A_{113} that have a 1 in the fifth place are

$$\{10000, 10001, 10010, 10011, 10100, 10101, \dots, 11111\}.$$

This has 2^4 elements.

(2) The elements of A_{113} that have a 0 in the fifth place are the 2^4 numbers

$$\{0000, 0001, \dots, 1111\}.$$

The above example illustrates the following scenario: If n has k digits in base 3 then there are clearly 2^{k-1} elements in A_n that have 0 in the k th place. How many elements of A_n have

a 1 in the k th place? If $3^{k-1} + \dots + 3^0 \leq n$ then every sequence of 0's and 1's of length k that begins with a 1 is in A_n . This is an additional 2^{k-1}

The two examples demonstrate the two cases that can occur in trying to compute the size of A_n . The following definition and theorem formalize this.

Definition 11 *Let S be defined as follows. Let $n \in \mathbb{N}$. Let k be the number of base 3 digits in n . (Note that $k = \lceil \log_3(n+1) \rceil$ except when $n = 0$ in which case $k = 1$.)*

• $S(0) = 1$ and

•

$$S(n) = 2^{k-1} + \begin{cases} 2^{k-1} & \text{if } 3^{k-1} + \dots + 3^0 \leq n ; \\ S(n - 3^{k-1}) & \text{otherwise.} \end{cases}$$

Theorem 12 *Let $n \in \mathbb{N}$ (n can be 0)*

(1) A_n has size $S(n)$.

(2) A_n is 3-free.

(3) $\text{sz}(n) \geq \Omega(n^{\log_3 2}) \sim \Omega(n^{0.63})$

Proof:

(1) We show that A_n is of size $S(n)$ by induction on n . If $n = 0$ then $A_0 = \{0\}$ which is of size $S(0) = 1$.

Inductively assume that, for all $0 \leq m < n$, A_m is of size $S(m)$.

Let k be the number of base 3 digits in n . There are several cases.

Case 1: $n \geq 3^{k-1} + \dots + 3^0$. Note that *every* element of

$$\{0 \dots 0, 0 \dots 1, 0 \dots 10, 0 \dots 11, \dots, 1 \dots 11\}$$

(all numbers of length k in base 3) is in A_n , and A_n cannot have any more elements. Hence A_n is of size $2^k = S(n)$.

Case 2: $n < 3^{k-1} + \dots + 3^0$. Note that the k th digit in base 3 is 1 since if it was 2 we would be in case 1, and if it was 0 then the number would only need $k-1$ (or less) digits in base 3.

(a) We count the numbers of the form $1b_{k-1} \dots b_0$ such that $b_{k-1}, \dots, b_0 \in \{0, 1\}$ and $1b_{k-1} \dots b_0 \leq n$. This is equivalent to asking that the number (in base 3) $b_{k-1} \dots b_0 \leq n - 3^{k-1}$, which is in $A_{n-3^{k-1}}$. Hence we have a bijection between the elements of A_n that begin with 1 and the set $A_{n-3^{k-1}}$. Inductively this is $S(n - 3^{k-1})$.

(b) We count the numbers of the form $0b_{k-1} \dots b_0$ such that $b_{k-1} \dots b_0 \in \{0, 1\}$ and $1b_{k-1} \dots b_0 \leq n$. Since the k th digit in base 3 of n is 1, there are clearly 2^{k-1} elements of this form.

Hence we have A_n is of size $S(n - 3^{k-1}) + 2^{k-1} = S(n)$.

(2) We show that A_n is 3-free. Let $x, y, z \in A_n$ form a 3-AP. Let x, y, z in base 3 be $x = x_{k-1} \cdots x_0$, $y = y_{k-1} \cdots y_0$, and $z = z_{k-1} \cdots z_0$. By the definition of A_n , for all i , $x_i, y_i, z_i \in \{0, 1\}$. By Fact 5 $x + z = 2y$. Since $x_i, y_i, z_i \in \{0, 1\}$ the addition is done *without carries*. Hence we have, for all i , $x_i + z_i = 2y_i$. Since $x_i, y_i, z_i \in \{0, 1\}$ we have $x_i = y_i = z_i$, so $x = y = z$.

(3) When $n = 3^{k-1} + 3^{k-2} + \cdots + 3^0$ then, by item 2,

$$\text{sz}(n) \geq |A_n| = S(n) = 2^k = 2^{\lceil \log_3(n+1) \rceil} \approx n^{\log_3 2} \approx n^{0.63}.$$

Using this one easily obtains that, for all n (not just of the form $3^{k-1} + \cdots + 3^0$)

$$\text{sz}(n) \geq \Omega(n^{\log_3 2}) \sim \Omega(n^{0.63}).$$

■

5 Simple Upper Bounds via Splitting

Theorem 13

- (1) For all n_1, n_2 , $\text{sz}(n_1 + n_2) \leq \text{sz}(n_1) + \text{sz}(n_2)$.
(2) For all n , $\text{sz}(kn) \leq k \cdot \text{sz}(n)$.

Proof:

1) Let A be a 3-free subset of $[n_1 + n_2]$ of size $\text{sz}(n_1 + n_2)$. Let $A_1 = A \cap [1, n_1]$ and $A_2 = A \cap [n_1 + 1, n_1 + n_2]$. Since A_1 is a 3-free subset of $[n_1]$, $|A_1| \leq \text{sz}(n_1)$. Since A_2 is the translation of a 3-free subset of $[n_2]$, $|A_2| \leq \text{sz}(n_2)$. Hence

$$|A| = |A_1| + |A_2| \leq \text{sz}(n_1) + \text{sz}(n_2).$$

2) This follows from part (1). ■

Since we will initially not know $\text{sz}(n_1)$ and $\text{sz}(n_2)$, how can we use this theorem? We will often know upper bounds on $\text{sz}(n_1)$ and $\text{sz}(n_2)$ and this will provide upper bounds on $\text{sz}(n_1 + n_2)$.

Assume we know upper bounds on $\text{sz}(1), \dots, \text{sz}(n-1)$. Call those bounds $\text{usz}(1), \dots, \text{usz}(n-1)$. Then $\text{usz}(n)$, defined below, is an upper bound on $\text{sz}(n)$.

$$\text{usz}(n) = \min\{\text{usz}(n_1) + \text{usz}(n_2) \mid n_1 + n_2 = n\}$$

This is the only elementary method we have for getting upper bounds on $\text{sz}(n)$. We will look at a sophisticated method, which only works for rather large n , in the sequel.

6 Exact Values via Backtracking

In this section we describe several backtracking algorithms for finding $\text{sz}(n)$. All of them will use depth first search. The key differences in the algorithms lie in both how much information they have ahead of time and the way they prune the backtrack tree. Most of the algorithms find $\text{sz}(1), \dots, \text{sz}(i-1)$ before finding $\text{sz}(i)$.

Throughout this section we will think of elements of $\{0, 1\}^*$ and finite sets of natural numbers interchangeably. The following notation makes this rigorous.

Notation: Let $\sigma \in \{0, 1\}^n$.

- (1) We identify σ with the set $\{i \mid \sigma(i) = 1\}$.
- (2) If $1 \leq i \leq j \leq n$ then we denote $\sigma(i) \cdots \sigma(j)$ by $\sigma[i \dots j]$.
- (3) σ has a 3-AP means there exists a 3-AP x, y, z such that $\sigma(x) = \sigma(y) = \sigma(z) = 1$.
- (4) σ is 3-free means that σ does not have a 3-AP.
- (5) $\#(\sigma)$ is the number of bits set to 1 in σ . Note that it is the number of elements in the set we identify with σ .
- (6) Let $\sigma = \alpha\tau$ where $\alpha, \tau \in \{0, 1\}^*$. Then α is a *prefix* of σ , and τ is a *suffix* of σ . In particular $\alpha 0$ is α concatenated with 0, and $\alpha 1$ is α concatenated with 1.
- (7) ϵ is the empty string.

We will need an algorithm to test if a given string is 3-free. Let `THREE_FREE` be such a test. We will describe our implementation of this in Section 6.3.

For all of the algorithms in this section we will present a main algorithm that calls a DFS, and then present the DFS.

6.1 Basic Backtracking Algorithms

In our first algorithm for $\text{sz}(n)$ we do a depth first search of $\{0, 1\}^n$ where we eliminate a node α if α is not 3-free.

```

BASIC( $n$ )
   $sz(n) = 0$ 
  BASIC_DFS( $\epsilon, n$ )
  Output( $sz(n)$ )
END OF ALGORITHM

```

```

BASIC_DFS( $\alpha, n$ )
  If  $|\alpha| = n$  then
     $sz(n) = \max(sz(n), \#(\alpha))$ 
  Else
    BASIC_DFS( $\alpha 0, n$ ) (Since  $\alpha$  is 3-free, so is  $\alpha 0$ )
    If THREE_FREE( $\alpha 1$ ) then BASIC_DFS( $\alpha 1, n$ )
END OF ALGORITHM

```

The algorithm presented above will find $sz(n)$ but is inefficient. The key to the remaining algorithms in this section is to cut down on the number of nodes visited. In particular, we will not pursue $\alpha 0$ if we can guarantee that any 3-free suffix of $\alpha 0$ will not have enough 1's in it to make it worth pursuing.

Assume we know $sz(1), \dots, sz(n-1)$. By Theorem 13, $sz(n) \in \{sz(n-1), sz(n-1) + 1\}$. Hence we need to determine if $sz(n) = sz(n-1) + 1$.

Assume there exists a 3-free set $A \in \{0, 1\}^n$ with $\#(A) = sz(n-1) + 1$ and prefix α . Then

$$A = \alpha\tau \text{ where } |\tau| = n - |\alpha| \text{ and}$$

$$\#(\alpha) + \#(\tau) = sz(n-1) + 1.$$

Since τ is 3-free we know that $\#(\tau) \leq sz(n - |\alpha|)$. Therefore if α is the prefix of a 3-free set of $[n]$ of size $sz(n-1) + 1$ then

$$\#(\alpha) + sz(n - |\alpha|) \geq sz(n-1) + 1$$

Notation:

$$\text{POTB}(\alpha, n) = \begin{cases} \text{TRUE} & \text{if } \#(\alpha) + sz(n - |\alpha|) \geq sz(n-1) + 1; \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

The POT stands for Potential: does α have the potential to be worth pursuing? The B stands for Basic, since we are using it in the Basic algorithm.

We now have two tests to eliminate prefixes: $\text{THREE_FREE}(\alpha)$ and $\text{POTB}(\alpha, n)$. If α ends in a 0 then we do not need to test $\text{THREE_FREE}(\alpha)$. If α ends in a 1 then we do not need to test $\text{POTB}(\alpha, n)$.

```

BASIC2( $n$ )
   $\text{sz}(n) = \text{sz}(n - 1)$ 
  BASIC_DFS2( $\epsilon, n$ )
  Output( $\text{sz}(n)$ )
END OF ALGORITHM

```

```

BASIC_DFS2( $\alpha, n$ )
  If  $|\alpha| = n$  then
    if  $\#(\alpha) = \text{sz}(n - 1) + 1$  then
       $\text{sz}(n) = \text{sz}(n - 1) + 1$ 
      Exit BASIC_DFS2 and all recursive calls of it
    Else
      If  $\text{POTB}(\alpha 0, n)$  then BASIC_DFS2( $\alpha 0, n$ )
      If  $\text{THREE\_FREE}(\alpha 1)$  then BASIC_DFS2( $\alpha 1, n$ )
  END OF ALGORITHM

```

6.2 Backtracking Algorithm with Information

Definition 14 For all $i \in \mathbb{N}$ let $\text{TF}(i)$ be the set of all 3-free sets of $[i]$.

Let L and m be parameters. We will later take them to be $L = 25$ and $m = 80$. We will do the following to obtain information in two phases, which will be used to prune the depth first search tree.

Phase I: Find $\text{TF}(L)$.

Phase II: For each $\sigma \in \text{TF}(L)$, for each $n \leq m$, find the size of the largest 3-free set of $\{0, 1\}^{L+n}$ that begins with σ .

Phase I: Find $\text{TF}(L)$

In phase I we find *all* 3-free sets of $[L]$ by using the following recurrence. We omit the details of the program.

$$\begin{aligned} \text{TF}(0) &= \{\epsilon\} \\ \text{TF}(L) &= \{\alpha 0 \mid \alpha \in \text{TF}(L - 1)\} \cup \{\alpha 1 \mid \alpha \in \text{TF}(L - 1) \wedge \text{THREE_FREE}(\alpha 1)\} \end{aligned}$$

Phase II: Generating More Information

In this phase we gather the following information: for every $\sigma \in \text{TF}(L)$, for every $n \leq m$, we find the $\rho \in \{0, 1\}^n$ such that $\text{THREE_FREE}(\sigma\rho)$ and $\#(\rho)$ is maximized; then let $\text{NUM}(\sigma, n) = \#(\rho)$. Note that $\text{NUM}(\sigma, n)$ is the maximum number of 1's that can be in a string that extends σ by n bits while keeping the entire string 3-free. $\text{NUM}(\sigma, n)$ counts the 1's in the extension but not in σ . The main point of the phase is to find $\text{NUM}(\sigma, n)$ values; we do not keep the ρ 's that are encountered. We do not even calculate sz values in the algorithm; however, we can (and do) easily calculate some sz values after this phase.

It is easy to see that, for all $\sigma \in \text{TF}(L)$, $\text{NUM}(\sigma, 0) = 0$. Hence we only discuss the case $n \geq 1$. The algorithm will be given an input n , $1 \leq n \leq m$ and will try to find, for every $\sigma \in \text{TF}(L)$, $\text{NUM}(\sigma, n)$.

Before trying to find $\text{NUM}(\sigma, n)$, where $1 \leq n \leq m$, we have computed the following:

- (1) $\text{TF}(L)$ from phase I.
- (2) For all $\sigma' \in \text{TF}(L)$, for every $n' < n$, $\text{NUM}(\sigma', n')$.

It is easy to see that $\text{NUM}(\sigma, n) \in \{\text{NUM}(\sigma, n-1), \text{NUM}(\sigma, n-1) + 1\}$. Let $\alpha \in \{0, 1\}^{\leq L+m}$ be such that σ is a prefix of α . We will want to pursue strings α that have a chance of showing $\text{NUM}(\sigma, n) = \text{NUM}(\sigma, n-1) + 1$.

Assume $A \in \{0, 1\}^{L+n}$ is such that A is 3-free, A has prefix α (hence prefix σ), and

$$\#(A) = \#(\sigma) + \text{NUM}(\sigma, n-1) + 1.$$

Note that such an A will show that $\text{NUM}(\sigma, n) = \text{NUM}(\sigma, n-1) + 1$ with the last n bits of A playing the role of ρ in the definition of $\text{NUM}(\sigma, n)$. Rewrite α as $\beta\sigma'$ where $\beta \in \{0, 1\}^{\leq m}$ and $\sigma' \in \{0, 1\}^L$. Note that

$$A = \beta\sigma'A[|\beta| + L + 1 \dots n + L] = \alpha A[|\beta| + L + 1 \dots n + L].$$

Hence

$$\#(A) = \#(\alpha) + \#(A[|\beta| + L + 1 \dots n + L]).$$

We bound $\#(A)$ from above. Since we know α , we know $\#(\alpha)$. (Now is the key innovation.) Note that $A[|\beta| + L + 1 \dots n + L]$ is a string of length $n - |\beta|$ such that $\sigma'A[|\beta| + L + 1 \dots n + L]$ is 3-free. Hence

$$\#(A[|\beta| + L + 1 \dots n + L]) \leq \text{NUM}(\sigma', n - |\beta|)$$

therefore

$$\#(A) = \#(\alpha) + \#(A[|\beta| + L + 1 \dots n]) \leq \#(\alpha) + \text{NUM}(\sigma', n - |\beta|).$$

By our assumption $\#(A) = \#(\sigma) + \text{NUM}(\sigma, n - 1) + 1$, so

$$\#(\sigma) + \text{NUM}(\sigma, n - 1) + 1 = \#(A) \leq \#(\alpha) + \text{NUM}(\sigma', n - |\beta|).$$

Hence

$$\#(\alpha) + \text{NUM}(\sigma', n - |\beta|) \geq \#(\sigma) + \text{NUM}(\sigma, n - 1) + 1.$$

We define a potential function that uses this test.

$$\text{POTG}(\sigma, \alpha, n) = \begin{cases} \text{TRUE} & \text{if } \#(\alpha) + \text{NUM}(\sigma', n - |\beta|) \geq \#(\sigma) + \text{NUM}(\sigma, n - 1) + 1; \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

```
INIT_GATHER( $n$ )
  For every  $\sigma \in \text{TF}(L)$ 
     $\text{NUM}(\sigma, 0) = 0$ 
  END OF ALGORITHM
```

```
GATHER( $n$ ) (Assume  $n \geq 1$ .)
  For every  $\sigma \in \text{TF}(L)$ 
     $\text{NUM}(\sigma, n) = \text{NUM}(\sigma, n - 1)$ 
    GATHER_DFS( $\sigma, \sigma, n$ )
  END OF ALGORITHM
```

```
GATHER_DFS( $\sigma, \alpha, n$ ) ( $\sigma$  is of length  $L$ )
  If  $|\alpha| = n$  then
    If  $\#(\alpha) = \text{NUM}(\sigma, n - 1) + 1$  then
       $\text{NUM}(\sigma, n) = \text{NUM}(\sigma, n - 1) + 1$ 
      Exit GATHER_DFS and all recursive calls of it.
    Else
      If POTG( $\sigma, \alpha_0, n$ ) then GATHER_DFS( $\sigma, \alpha_0, n$ )
      If THREE_FREE( $\alpha_1$ ) then GATHER_DFS( $\sigma, \alpha_1, n$ )
  END OF ALGORITHM
```

Now that we have the values $\text{NUM}(\sigma, n)$ for all n , $0 \leq n \leq m$ we can compute $\text{sz}(n)$.

FIND_{sz}(n)

If $n \leq L$ then $\text{sz}(n) = \text{NUM}(0^L, n)$ If $L < n \leq m + L$ then $\text{sz}(n) = \max\{\#(\sigma) + \text{NUM}(\sigma, n - L)\}$
 END OF ALGORITHM

Phase III: Using the Information Gathered

We will present the algorithm for $n \geq m + L + 1$. We devise a potential function for prefixes.

Assume there exists a 3-free set $A \in \{0, 1\}^n$ with $\#(A) = \text{sz}(n - 1) + 1$ and prefix α . Rewrite α as $\beta\sigma'$ where $\beta \in \{0, 1\}^*$ and $\sigma' \in \{0, 1\}^L$. Note that

$$A = \beta\sigma'A[|\beta| + L + 1 \dots n].$$

Hence

$$\#(A) = \#(\beta\sigma') + \#(A[|\beta| + L + 1 \dots n]) = \#(\alpha) + \#(A[|\beta| + L + 1 \dots n]).$$

We bound $\#(A)$ from above. Clearly we know $\#(\beta\sigma') = \#(\alpha)$. (Now is the key innovation.) Note that $\sigma'A[|\beta| + L + 1 \dots n]$ is a 3-free string of length $L + n - |\beta| - L = n - |\beta|$ that has $\sigma' \in \text{TF}(L)$ as a prefix. Hence

$$\#(A[|\beta| + L + 1 \dots n]) \leq \text{NUM}(\sigma', n - |\beta| - L).$$

Therefore

$$\#(A) \leq \#(\alpha) + \text{NUM}(\sigma', n - |\beta| - L).$$

Since $\#(A) = \text{sz}(n - 1) + 1$ we have

$$\text{sz}(n - 1) + 1 = \#(A) \leq \#(\alpha) + \text{NUM}(\sigma', n - |\beta| - L).$$

Hence

$$\#(\alpha) + \text{NUM}(\sigma', n - |\beta| - L) \geq \text{sz}(n - 1) + 1.$$

If $n - |\beta| - L \leq m$ then $\text{NUM}(\sigma', n - |\beta| - L)$ has been computed and we use this test. If $n - |\beta| - L > m$ then we cannot use this test; however in this case there are several weaker bounds we can use.

Test T1: We use sz . Since

$$\#(A[|\beta| + L + 1 \dots n]) \leq \text{sz}(n - |\beta| - L)$$

we define $T1(\alpha)$ as follows

$$T1(\alpha) : \#(\alpha) + \text{sz}(n - |\beta| - L) \geq \text{sz}(n - 1) + 1.$$

Note that this is the same test used in POTB.

Test T2: We use NUM and sz . Note that $A[|\beta| + 1, \dots, |\beta| + m]$ is a string of length m that extends σ' . Hence

$$\#(A[|\beta| + 1, \dots, |\beta| + m]) \leq \text{NUM}(\sigma', m).$$

Clearly

$$\#(A[|\beta| + m + 1, \dots, n]) \leq \text{sz}(n - m - |\beta|).$$

Hence

$$\#A[|\beta| + 1, \dots, n] \leq \text{NUM}(\sigma', m) + \text{sz}(n - m - |\beta|).$$

We define $T2(\alpha)$ as follows:

$$T2(\alpha) : \#(\alpha) + \text{NUM}(\sigma', m) + \text{sz}(n - m - |\beta|) \geq \text{sz}(n - 1) + 1.$$

Test T3: We use forbidden numbers. In Section 6.3 we will see that associated with the bit string α will be a set of forbidden numbers. These are all numbers f , $|\alpha| < f \leq n$, such that, viewing α as a set (that is, take the bit positions that are a 1), $\alpha \cup \{f\}$ has a 3-AP. Let c be the number of numbers that are *not* forbidden. If α can be extended to a 3-free set of $[n]$ that has $\text{sz}(n - 1) + 1$ elements in it then we need the following to be true.

$$T3(\alpha) : \#(\alpha) + c \geq \text{sz}(n - 1) + 1.$$

Notation: Let $\sigma' \in \{0, 1\}^L$, $\alpha \in \{0, 1\}^*$, $n \in \mathbf{N}$, and $|\alpha| < n$. Let $\alpha = \beta\sigma'$. Then

$$\text{POT}(\alpha, n) = \begin{cases} \text{TRUE} & \text{if } n - |\beta| - L \leq m \text{ and} \\ & \#(\alpha) + \text{NUM}(\sigma', n - |\beta| - L) \geq \text{sz}(n - 1) + 1; \\ \text{TRUE} & \text{if } n - |\beta| - L > m \text{ and} \\ & T1(\alpha) \wedge T2(\alpha) \wedge T3(\alpha); \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

```
FINAL(n)
  sz(n) = sz(n - 1)
  For every  $\sigma \in \text{TF}(L)$ 
    FINAL_DFS( $\sigma, n$ )
  Output(sz(n))
END OF ALGORITHM
```

```
FINAL_DFS( $\alpha, n$ )
  If  $|\alpha| = n$  then
    If  $\#(\alpha) = \text{sz}(n - 1) + 1$  then
      sz(n) = sz(n - 1) + 1
      Exit FINAL_DFS and all recursive calls to it
    Else (In what follows we know  $|\alpha| < n$ .)
      If POT( $\alpha 0, n$ ) then FINAL_DFS( $\alpha 0, n$ )
      If THREE_FREE( $\alpha 1$ ) then FINAL_DFS( $\alpha 1, n$ )
  END OF ALGORITHM
```

6.3 Testing if a string is 3-free

In the above algorithms we called a procedure called THREE_FREE. We do not have such a procedure. Instead we have a process that does the following.

- A string is being constructed bit by bit.
- While constructing it we need to know if adding a 1 will cause it to no longer be 3-free.

We describe this process.

- (1) We are building α which will be a string of length at most n . We maintain both the string α and the array of forbidden bits f .
- (2) Assume α is currently of length i . If $k \geq i + 1$ and $f_k = 1$ then setting $\alpha(k) = 1$ would create a 3-AP in α .
- (3) Initially α is of length 0 and f is an array of n 0's.
- (4) (This is another key innovation.) Assume that we have set $\alpha(1) \cdots \alpha(i-1)$. Conceptually

maintain α and f as follows

$$\begin{array}{ccccccc} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{i-2} & \alpha_{i-1} & \\ f_n & \cdots & f_{2i+1} & f_{2i} & f_{2i-1} & f_{2i-2} & \cdots & f_{i+1} & f_i \end{array}$$

(5) If we append 0 to α then the new α and f are

$$\begin{array}{cccccccc} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{i-2} & \alpha_{i-1} & 0 & \\ f_n & \cdots & f_{2i+2} & f_{2i+1} & f_{2i-2} & f_{2i-1} & \cdots & f_{i+3} & f_{i+2} & f_{i+1} \end{array}$$

(6) If we want to append 1 to α we do the following:

(a) Shift f one bit to the right.

$$\begin{array}{ccccccc} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{i-2} & \alpha_{i-1} & \\ f_n & \cdots & f_{2i} & f_{2i-1} & f_{2i-2} & f_{2i-3} & \cdots & f_{i+2} & f_{i+1} \end{array}$$

(b) The bit string α remains as the above diagram, and f is replaced by the bitwise OR of α and f . (The bits of f that do not correspond to bits of α remain the same.) We denote the new f by f' .

(c) Shift α one bit to the left and append a 1 to it.

$$\begin{array}{cccccccc} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{i-2} & \alpha_{i-1} & 1 & \\ f'_n & \cdots & f'_{2i} & f'_{2i-1} & f'_{2i-2} & \cdots & f'_{i+3} & f'_{i+2} & f'_{i+1} \end{array}$$

We leave it to the reader to verify that this procedure correctly sets f . Note that this procedure is very fast since the main operations are bit-wise ORs and SHIFTS.

In the DFS algorithms above we often have the line

If `THREE_FREE(α 1)` then `DFS(α 1)` (where `DFS` is one of the DFS algorithms).

As noted above we do not have a procedure `THREE_FREE`. So what do we really do? We use the forbidden bit array. For example, lets say that the first 99 bits of α are known and the forbidden bit pattern from 100 to 108 is as follows.

$$\begin{array}{cccccccc} \cdots & 100 & 101 & 102 & 103 & 104 & 105 & 106 & 107 & 108 \\ f'_n & \cdots & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

We are pondering extending α by 0 or 1. But note that the next place to extend α is a forbidden bit. In fact, the next four places are all forbidden bits (these are the four rightmost

bits). Hence we automatically put 0's in the next four places. After that we do the recursive calls to the DFS procedure.

We illustrate this by showing how we really would code BASIC_DFS.

Definition 15 *Let $\alpha, f \in \{0, 1\}^*$ such that f is the forbidden bit array for α . Let $b \in \{0, 1\}$. Then $\text{ADJUST}(\alpha, f, b)$ is the forbidden bit array that is created when b is appended to α . The details were described above.*

```

BASIC_DFS( $\alpha, f, n$ )
  If  $|\alpha| = n$  then
     $\text{sz}(n) = \max\{\text{sz}(n), \#(\alpha)\}$ 
    Exit BASIC_DFS
  Else
    While ( $f_{|\alpha|+1} = 1$ ) and ( $|\alpha| \leq n$ )
       $\alpha = \alpha 0$ 
      BASIC_DFS( $\alpha 0, \text{ADJUST}(\alpha, f, 0), n$ )
      BASIC_DFS( $\alpha 1, \text{ADJUST}(\alpha, f, 1), n$ )
END OF ALGORITHM

```

6.4 Coding Techniques to Speed up our Program

If there is a 3-free set $A \in \{0, 1\}^n$ such that $\#(A) = \text{sz}(n-1) + 1$ then $A(1) = A(n) = 1$ (otherwise there would be a 3-free subset of $[n-1]$ of size $\text{sz}(n-1) + 1$). We use this as follows.

- (1) In BASIC and BASIC2 we can start with 1 instead of ϵ . We can also end with a 1.
- (2) In FINAL we need only begin with the $\sigma \in \text{TF}(L)$ that begin with 1. (GATHER is unaffected since we need to gather information about all σ including those that begin with 0.)
- (3) In the procedure THREE_FREE we test if σ is 3-free, we are actually testing if $\sigma \cup \{n\}$ is 3-free.

6.5 Empirical Results

The test

$$\#(\alpha) + \text{NUM}(\sigma', n - |\beta| - L) \geq \text{sz}(n-1) + 1.$$

cut down on the number of nodes searched by a factor of 10. The tests $T1$ and $T2$ were useful but not dramatic. The test $T3$ did not seem to help much at all.

The method enabled us to find exact values up to $\text{sz}(187)$.

7 Upper Bounds via Linear Programming

We describe how linear programming was used to get better upper bounds on $\text{sz}(n)$. The actual improvement obtained is in Appendix IV.

We rephrase the problem of finding a large 3-free set of $[n]$ as an integer programming problem:

Maximize: $x_1 + \cdots + x_n$

Constraints:

$x_i + x_j + x_k \leq 2$ for $1 \leq i < j < k \leq n$ where i, j, k is a 3-AP.

$0 \leq x_i \leq 1$

Say that (x_1, \dots, x_n) is a solution. Then the set

$$A = \{i \mid x_i = 1\}$$

is a 3-free set of size $\text{sz}(n)$. Hence we can talk about solutions to this integer programming problem, and 3-free sets A , interchangeably.

The general integer programming problem is NP-complete. We have tried to use IP packages to solve this but the problem is too big for them. The two we used are actually parts of LP packages, CPLEX (11) and GLPK (15). However, we can use linear programming, and these packages, to get upper bounds on $\text{sz}(n)$.

If the integer program above is relaxed to be a linear program, and the max value for $x_1 + \cdots + x_n$ was s , then we would know $\text{sz}(n) \leq s$. We will use this linear program, with many additional constraints, to obtain upper bounds on values of $\text{sz}(n)$ for which we do not have exact values.

If we just use the relaxation of the integer programming problem given in the last section then the upper bounds obtained are worse than those obtained by the splitting method. Hence we will need to add more upper bound constraints. For example, if we know that $\text{sz}(100) \leq 27$ and we are looking at $\text{sz}(200)$ we can put in the constraints

$$x_1 + \cdots + x_{100} \leq 27$$

$$x_2 + \cdots + x_{101} \leq 27$$

⋮

$$x_{100} + \cdots + x_{199} \leq 27$$

$$x_{101} + \cdots + x_{200} \leq 27$$

$$x_1 + x_3 + x_5 + \cdots + x_{199} \leq 27$$

More generally, if we know $\text{sz}(i)$ for $i \leq m$ then, for every $3 \leq i \leq m$, we have the constraints

$$x_{b_1} + \cdots + x_{b_i} \leq \text{sz}(i) \text{ such that } b_1 < \cdots < b_i \text{ is an } i\text{-AP} .$$

Putting in all of these constraints caused us linear programs that took too long to solve. However, the constraints based on $\text{sz}(100) = 27$ are intuitively more powerful than the constraints based on $\text{sz}(3) = 2$. Hence we put in fewer constraints. However, it turned out that putting in all constraints that used the values of $\text{sz}(i)$ for $20 \leq i \leq 187$ yielded programs that ran quickly. But there was another problem— These programs always resulted in numbers bigger than our upper bounds on $\text{sz}(n)$ based on splitting, hence the information was not useful.

We then put in *lower bound constraints*. For example, if we want to see if $\text{sz}(187) = 41$ we can have the constraint

$$x_1 + \cdots + x_{187} = 41.$$

We can also have constraints based on known lower values of sz . For example, since $\text{sz}(100) = 27$ a 3-free set of $[187]$ of size 41 would need to have

$$x_{101} + \cdots + x_{187} \geq 14$$

since otherwise

$$x_1 + \cdots + x_{187} \leq 40.$$

We then put in all lower bound constraints. This always resulted in either finding the conjectured value (which was not helpful) or finding that the feasible region was empty. In the

latter case we know that the conjectured value cannot occur.

We now formalize all of this.

INPUT:

- n
- $\text{usz}(1), \dots, \text{usz}(n-1)$ (upper bound on sz).
- t . (We want to show $\text{sz}(n) < t$.)

OUTPUT: Either “ $\text{sz}(n) \leq t-1$ ” or “NO INFO”

We will add the following constraints.

New Upper Constraints using Known Values of sz

For every i , $3 \leq i \leq m$, we have the constraints

$$x_{b_1} + \dots + x_{b_i} \leq \text{sz}(i) \text{ such that } b_1 < \dots < b_i \text{ is an } i\text{-AP} .$$

New Lower Constraints Based on $\text{usz}(i)$

From the upper bound constraints we have

$$x_{b_1} + \dots + x_{b_i} \leq \text{sz}(i) \text{ such that } b_1 < \dots < b_i \text{ is an } i\text{-AP} .$$

If A is to have t elements in it we need

$$\sum_{j \notin \{b_1, \dots, b_i\}} x_j \geq t - \text{sz}(i) \text{ such that } b_1 < \dots < b_i \text{ is an } i\text{-AP} .$$

New Lower Constraints Based on Prefixes

We want to know if there is a 3-free set $A \subseteq \{1, \dots, n\}$ with $\#(A) \geq t$. Let L be a parameter. We consider every $\sigma \in \{0, 1\}^L$ that could be a prefix of A . In order to be a prefix it must satisfy the following criteria (and even then it might not be a prefix).

- σ is 3-free.
- For every i , $1 \leq i \leq L$, let τ_i be the i -length prefix of σ . Then

$$\#(\tau_i) + \text{sz}(n-i) \geq t.$$

- σ begins with a 1. We can assume this since if there is such a 3-free set that does not begin with 1 then we can shift it.

Definition 16 *If σ satisfies the criteria above then $\text{GOOD}(\sigma)$ is TRUE , else it is false.*

For each such σ such that $\text{GOOD}(\sigma) = \text{TRUE}$ we create a linear program that has the following additional constraints.

$$x_i = \sigma(i) \text{ for all } i, 1 \leq i \leq L.$$

$$x_{L+1} + x_{L+2} + \cdots + x_n \geq t - \#(\sigma).$$

If *every* such linear program returns a value that is $\leq t-1$ then we can say that $\text{sz}(n) \leq t-1$. If *any* return a value that is $\geq t$ then we cannot make any conclusions.

Using $L = 30$ we improved many of the upper bounds obtained by the splitting method. This value of L was chosen partially because of issues with word-size.

In Table 2 we describe which values we obtained improvements on.

8 Lower Bounds via Thirds Method

The large 3-free sets that are found by the methods above all seem to have many elements in the first and last thirds but very few in the middle third. This leads to the following heuristic to find a large 3-free set.

Given a large 3-free set $A \subseteq [m]$ one can create a 3-free set of $[3m-1]$ in the following way: $A \cup B$ where B is the set A shifted to be a subset of $\{2m+1, \dots, 3m\}$. You can then try to include some elements from the middle; however, most of the elements of the middle will be excluded.

We could take different 3-free sets of $[m]$ for A and B . In fact, we could go through *all* large 3-free sets of $[m]$.

In practice we do not use the maximum 3-free set of $[m]$. We sometimes found larger 3-free sets of $[m]$ by using 3-free sets of size between $m - \log m$ and $m + \log m$ that are of size within one or two of maximum. This leads to most of the remaining middle elements being forbidden; hence, searching for the optimal number that can be placed is easy. There is nothing sacrosanct about $\log m$ or being within one or two of maximum. We only used this technique for numbers between 3 and 250; for larger values of m other parameters may lead to larger 3-free sets. We know that for $m \leq 187$ the thirds method always found a set of size $\text{sz}(m)$, and also the best known value for $m \leq 300$.

Using this technique we obtained the following results.

- (1) $\text{sz}(194) \geq 41$. (This was known by (43).)
- (2) $\text{sz}(204) \geq 42$. (This is new.)
- (3) $\text{sz}(209) \geq 43$. (This was known by (43).)
- (4) $\text{sz}(215) \geq 44$. (This was known by (43).)
- (5) $\text{sz}(227) \geq 45$. (This is new.)
- (6) $\text{sz}(233) \geq 46$. (This is new.)
- (7) $\text{sz}(239) \geq 47$. (This was known by (43).)
- (8) $\text{sz}(247) \geq 48$. (This was known by (43).)

The three free set that showed $\text{sz}(204) \geq 42$ is

$\{1, 3, 8, 9, 11, 16, 20, 22, 25, 26, 38, 40, 45, 46, 48, 53, 57, 59, 62, 63, 127, 132, 134, 135, 139, 140, 147, 149, 150, 152, 156, 179, 181, 182, 186, 187, 189, 194, 198, 200, 203, 204\}$.

The three free set that showed $\text{sz}(227) \geq 45$ is

$\{1, 2, 6, 8, 12, 17, 19, 20, 24, 25, 27, 43, 45, 51, 54, 55, 58, 60, 64, 72, 76, 79, 129, 145, 147, 154, 155, 159, 160, 167, 169, 170, 172, 176, 201, 202, 206, 208, 212, 217, 219, 220, 224, 225, 227\}$.

The three free set that showed $\text{sz}(233) \geq 46$ is

$\{1, 4, 5, 11, 13, 14, 16, 26, 29, 30, 35, 50, 52, 58, 61, 62, 68, 73, 76, 77, 80, 82, 97, 137, 152, 154, 157, 158, 161, 166, 172, 173, 176, 182, 184, 199, 204, 205, 208, 218, 220, 221, 223, 229, 230, 233\}$.

9 Other methods

We present methods for constructing large 3-free sets that were tried but ended up not being as good as Backtracking or the Thirds Method. These methods, or modifications of them, may prove useful later. In Appendix III we compare them to each other and to the optimal results that are known.

9.1 The Concatenation Method

The following theorem is similar in proof to Theorem 13.

Definition 17 If B is a set and $m \in \mathbb{N}$ then an m -translate of B is the set $\{x+m \mid x \in B\}$.

We need the following simple fact.

Fact 18 Let $n = n_1 + n_2$. Let \mathcal{A}_1 be the set of all 3-free subsets of $[n_1]$. Let \mathcal{A}_2 be the set of all 3-free subsets of $[n_2]$. If A is a 3-free subset of $[n_1 + n_2]$ then $A = A_1 \cup A_2$ where $A_1 \in \mathcal{A}_1$ and A_2 is an n_1 -translate of some element of \mathcal{A}_2 .

Definition 19 If $n, k \in \mathbb{N}$ then $\mathcal{E}_{n,k}$ is the set of 3-free subsets of $[n]$ that contain both 1 and n and have size k .

The following assertions, stated without proof, establish the usefulness of the \mathcal{E} 's in computing $\text{sz}(n)$:

- (a) $|\mathcal{E}_{1,0}| = 0$, $|\mathcal{E}_{1,1}| = 1$. (This is used at the base of a recursion.)
- (b) if $n \geq 2$ then $|\mathcal{E}_{n,0}| = 0$, $|\mathcal{E}_{n,1}| = 0$, and $|\mathcal{E}_{n,2}| = 1$. (This is used at the base of a recursion.)
- (c) if $\mathcal{E}_{n,k} \neq \emptyset$ then $\text{sz}(n) \geq k$;
- (d) if $\mathcal{E}_{n,k} = \emptyset$ where $k, n > 1$ then $\mathcal{E}_{n,l} = \emptyset$ for all $l > k$; and
- (e) if $\mathcal{E}_{n,k} = \emptyset$ and $k, n > 1$ then $\text{sz}(n) < k$.

The sets that comprise $\mathcal{E}_{n,k}$ can be obtained from $\mathcal{E}_{m,l}$ where $m < n$ and $l < k$. Let $A \in \mathcal{E}_{n,k}$. Partition A into $A_1 = A \cap \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$ and $A_2 = A \cap \{\lfloor \frac{n}{2} \rfloor + 1, \dots, n\}$. Let x be the largest element of A_1 and let y be the smallest element of A_2 . Then $A_1 \in \mathcal{E}_{x,|A_1|}$ and A_2 is a $(y-1)$ -translation of an element of $\mathcal{E}_{n-y+1,|A_2|}$. This can be used to obtain a Dynamic Program to find $\mathcal{E}_{n,k}$.

This method requires too much time and space to be useful for finding $\text{sz}(n)$. However, it is useful if you want to find many large 3-free sets of $[n]$.

9.2 The Greedy Vertex Cover Method

We can rephrase our problem as that of finding the maximum independent set in a hypergraph.

Definition 20

- (1) A hypergraph is a pair (V, E) such that E is a collection of subsets of V . The elements

- of V are called vertices. The elements of E are called hyperedges.
- (2) A 3-uniform hypergraph is one where all of the hyperedges have exactly three vertices in them.
- (3) If $H = (V, E)$ is a hypergraph then \bar{H} , the complement of H , is $(V, \mathcal{P}(V) - E)$ where $\mathcal{P}(V)$ is the powerset of V .
- (4) If $H = (V, E)$ is a hypergraph then an independent set of H is a set $U \subseteq V$ such that

$$(\forall U' \subseteq U)[U' \notin E].$$

- (5) If $H = (V, E)$ is a hypergraph then a vertex cover of H is a set $U \subseteq V$ such that

$$(\forall e \in E)(\exists v \in U)[v \in e].$$

Note 1 If U is a vertex cover of H then \bar{U} is an independent set of H .

Let $G = (V, E)$ be the following 3-uniform hypergraph.

$$V = \{1, \dots, n\};$$

$$E = \{(i, j, k) : (i < j < k) \wedge i, j, k \text{ form a 3-AP}\}.$$

The largest independent set in this hypergraph corresponds to the largest 3-free set of $[n]$. Unfortunately the independent set problem, even for the simple case of graphs, is NP-complete. In fact, approximating the maximum independent set is known to be NP-hard (22). It is possible that our particular instance is easier.

We have used the greedy method for vertex cover on our hypergraph; the complement of the cover gives a (not necessarily good) solution quickly. To compute the greedy vertex cover, at each step one selects the vertex in G with highest degree. If there is a tie either take the first one found or break the tie randomly. We will comment on this later. Once a vertex is selected it is removed from the graph along with all its incident edges. This process continues until no edges remain in G . For each of the $O(n)$ removals we find the vertex with highest degree in $O(n)$ time, so the greedy vertex cover can be found in $O(n^2)$ time.

We have actually described two algorithms here: VC-Det where you pick the first vertex (so this is deterministic) and VC-Rand where you pick at random. Both methods are fast. VC-Rand seems to give larger sets, as can be seen in Appendix III. VC-Det comes within at most 8 of optimal and VC-Rand comes within at most 6 of optimal, in the range $1 \leq n \leq 250$.

9.3 The Randomization Method

As noted in Section 4 the Greedy method starting at $1,2,3,\dots$ is the Base 3 method. What if you did not start with 1? What if you picked numbers at random rather than in order? This is the essence of the randomized method.

- 1) Randomly permute $1, \dots, n$ to get a_1, \dots, a_n .
- 2) Set $S = \emptyset$.
- 3) For $i = 1$ to n add a_i to S if doing so does not create a 3-AP in S

The running time is $O(n^2)$ using appropriate data structures. The method is fast and, as evidenced in Appendix III, yields 3-free sets that are at most 7 away from optimal, in the range $1 \leq n \leq 250$.

10 The Values of $sz(n)$ for Small n

Appendix II contains tables of results small n . A lower bound of X on $sz(n)$ means that there is a 3-free set of $[n]$ of size X . An upper bound of X on $sz(n)$ means that no set of $[n]$ of size X is 3-free. When we have exact values for $sz(n)$ they were obtained by backtracking as described in Section 6. When we have upper and lower bounds they are obtained by the thirds method, splitting, and linear programming.

- (1) Tables 1 and 2 gives exact values for $sz(n)$ for $1 \leq n \leq 187$. We obtained these results by backtracking.
- (2) Table 3 gives upper and lower bounds for $188 \leq n \leq 250$. The upper bounds for $188 \leq n \leq 250$ were obtained by Theorem 13 and the linear programming upper bound technique described in Section 7. The lower bounds for $188 \leq n \leq 250$ were obtained by the thirds-method described in Section 8.

11 Acknowledgments

We would like to thank Jon Chapin, MohammadReza Ghodsi, (this is not a typo- his name really is MohammedReza with that capital R), Walid Gomaa, Kin Keung Ma, Elnatan Reisner, Andre Utis, and the referees for proofreading and commentary. We would also like to thank Ryan Farrell, Tasha Innis, and Howard Karloff for help on some of the integer and linear programming methods, and Robert Kleinberg for discussions of matrix multiplication and 3-free sets. The first author would like to thank NSF grant CCR-01-05413.

12 Appendix I: Comparison to Known Results

There are several websites that contain results similar to ours:

- <http://www.math.uni.wroc.pl/~jwr/non-ave/index.htm>
- <http://www.research.att.com/~njas/sequences/A065825>
- <http://www.research.att.com/~njas/sequences/A003002>

12.1 Compare to First Website

The first website is about *Nonaveraging sets search*. A *nonaveraging set* is what we have been calling a 3-free set. They study the problem in a different way.

Definition 21 For $m \in \mathbb{N}$, $a(m)$ is the least number so that there is a nonaveraging subset of $\{1, \dots, a(m)\}$ of size m .

The following are easily verified.

Fact 22

$$\text{sz}(a(m)) \geq m.$$

$\text{sz}(n) \geq m$ iff $a(m) \leq n$. Hence large 3-free sets yield upper bounds on $a(m)$ and vice-versa.

If $\text{sz}(n) < m$ then $a(m) > n$.

If $\text{sz}(n) = m - 1$ and $\text{sz}(n + 1) = m$ then $a(m) = n$.

At the website they have exact values for $a(m)$ for $m \leq 35$, and upper bounds for $a(m)$ (hence 3-free sets) for $m \leq 1024$. They have $a(35) = 150$ which yields $\text{sz}(150) = 35$.

We summarize the difference between our data and the websites above:

- (1) Our table yields the following new exact results.
 - (a) Stated in their terms: $a(37) = 163$, $a(38) = 167$, $a(39) = 169$, $a(40) = 174$. (They had $a(37) \leq 163$, $a(38) \leq 167$, $a(39) \leq 169$, $a(40) \leq 174$.)
 - (b) Stated in our terms:
 - (i) $\text{sz}(n) = 37$ for $163 \leq n \leq 164$
 - (ii) $\text{sz}(n) = 38$ for $165 \leq n \leq 167$
 - (iii) $\text{sz}(n) = 39$ for $169 \leq n \leq 173$
 - (iv) $\text{sz}(n) = 40$ for $174 \leq n \leq 187$

- (2) Our table yields the following new bounds.
- (a) Stated in their terms: $a(42) \leq 204$, $a(45) \leq 227$, $a(46) \leq 233$. (They had $a(42) \leq 205$, $a(45) \leq 228$, $a(46) \leq 234$.)
 - (b) Stated in our terms: $\text{sz}(204) \geq 42$, $\text{sz}(227) \geq 45$, $\text{sz}(233) \geq 46$.
- (3) Impressively, they have obtained lower bounds on $\text{sz}(n)$ for $300 \leq n \leq 25,958$. (They have a 3-free set of $[25, 958]$ of size 1024.) We have not considered this range.

12.2 The Second and Third Website

The second website is the entry on $a(n)$ in the Online Encyclopedia. The first website has the most current results. The third website is the entry in the Online Encyclopedia of $\text{sz}(n)$. It only has values up to $n = 53$.

13 Appendix II: Tables for Small n

| n | $sz(n)$ | n | $sz(n)$ | n | $sz(n)$ | n | $sz(n)$ |
|-----|---------|-----|---------|-----|---------|-----|---------|
| 1 | 1 | 26 | 11 | 51 | 17 | 76 | 22 |
| 2 | 2 | 27 | 11 | 52 | 17 | 77 | 22 |
| 3 | 2 | 28 | 11 | 53 | 17 | 78 | 22 |
| 4 | 3 | 29 | 11 | 54 | 18 | 79 | 22 |
| 5 | 4 | 30 | 12 | 55 | 18 | 80 | 22 |
| 6 | 4 | 31 | 12 | 56 | 18 | 81 | 22 |
| 7 | 4 | 32 | 13 | 57 | 18 | 82 | 23 |
| 8 | 4 | 33 | 13 | 58 | 19 | 83 | 23 |
| 9 | 5 | 34 | 13 | 59 | 19 | 84 | 24 |
| 10 | 5 | 35 | 13 | 60 | 19 | 85 | 24 |
| 11 | 6 | 36 | 14 | 61 | 19 | 86 | 24 |
| 12 | 6 | 37 | 14 | 62 | 19 | 87 | 24 |
| 13 | 7 | 38 | 14 | 63 | 20 | 88 | 24 |
| 14 | 8 | 39 | 14 | 64 | 20 | 89 | 24 |
| 15 | 8 | 40 | 15 | 65 | 20 | 90 | 24 |
| 16 | 8 | 41 | 16 | 66 | 20 | 91 | 24 |
| 17 | 8 | 42 | 16 | 67 | 20 | 92 | 25 |
| 18 | 8 | 43 | 16 | 68 | 20 | 93 | 25 |
| 19 | 8 | 44 | 16 | 69 | 20 | 94 | 25 |
| 20 | 9 | 45 | 16 | 70 | 20 | 95 | 26 |
| 21 | 9 | 46 | 16 | 71 | 21 | 96 | 26 |
| 22 | 9 | 47 | 16 | 72 | 21 | 97 | 26 |
| 23 | 9 | 48 | 16 | 73 | 21 | 98 | 26 |
| 24 | 10 | 49 | 16 | 74 | 22 | 99 | 26 |
| 25 | 10 | 50 | 16 | 75 | 22 | 100 | 27 |

Table 1
Values of $sz(n)$; 1-100 found by Backtracking

| n | $sz(n)$ | n | $sz(n)$ | n | $sz(n)$ | n | $sz(n)$ |
|-----|---------|-----|---------|-----|---------|-----|---------|
| 101 | 27 | 126 | 32 | 151 | 35 | 176 | 40 |
| 102 | 27 | 127 | 32 | 152 | 35 | 177 | 40 |
| 103 | 27 | 128 | 32 | 153 | 35 | 178 | 40 |
| 104 | 28 | 129 | 32 | 154 | 35 | 179 | 40 |
| 105 | 28 | 130 | 32 | 155 | 35 | 180 | 40 |
| 106 | 28 | 131 | 32 | 156 | 35 | 181 | 40 |
| 107 | 28 | 132 | 32 | 157 | 36 | 182 | 40 |
| 108 | 28 | 133 | 32 | 158 | 36 | 183 | 40 |
| 109 | 28 | 134 | 32 | 159 | 36 | 184 | 40 |
| 110 | 28 | 135 | 32 | 160 | 36 | 185 | 40 |
| 111 | 29 | 136 | 32 | 161 | 36 | 186 | 40 |
| 112 | 29 | 137 | 33 | 162 | 36 | 187 | 40 |
| 113 | 29 | 138 | 33 | 163 | 37 | | |
| 114 | 30 | 139 | 33 | 164 | 37 | | |
| 115 | 30 | 140 | 33 | 165 | 38 | | |
| 116 | 30 | 141 | 33 | 166 | 38 | | |
| 117 | 30 | 142 | 33 | 167 | 38 | | |
| 118 | 30 | 143 | 33 | 168 | 38 | | |
| 119 | 30 | 144 | 33 | 169 | 39 | | |
| 120 | 30 | 145 | 34 | 170 | 39 | | |
| 121 | 31 | 146 | 34 | 171 | 39 | | |
| 122 | 32 | 147 | 34 | 172 | 39 | | |
| 123 | 32 | 148 | 34 | 173 | 39 | | |
| 124 | 32 | 149 | 34 | 174 | 40 | | |
| 125 | 32 | 150 | 35 | 175 | 40 | | |

Table 2
Values of $sz(n)$; 101-187 found by Dynamic Programming

| n | low | high | n | low | high | n | low | high |
|-----|-----|------|-----|-----|------|-----|-----|------|
| 188 | 40 | 41 | 213 | 43 | 50 | 238 | 46 | 57 |
| 189 | 40 | 42 | 214 | 43 | 51 | 239 | 47 | 57 |
| 190 | 40 | 42 | 215 | 44 | 51 | 240 | 47 | 57 |
| 191 | 40 | 43 | 216 | 44 | 51 | 241 | 47 | 58 |
| 192 | 40 | 44 | 217 | 44 | 51 | 242 | 47 | 58 |
| 193 | 40 | 44 | 218 | 44 | 51 | 243 | 47 | 58 |
| 194 | 41 | 44 | 219 | 44 | 51 | 244 | 47 | 58 |
| 195 | 41 | 44 | 220 | 44 | 52 | 245 | 47 | 58 |
| 196 | 41 | 45 | 221 | 44 | 52 | 246 | 47 | 59 |
| 197 | 41 | 45 | 222 | 44 | 52 | 247 | 48 | 59 |
| 198 | 41 | 46 | 223 | 44 | 52 | 248 | 48 | 59 |
| 199 | 41 | 46 | 224 | 44 | 53 | 249 | 48 | 59 |
| 200 | 41 | 47 | 225 | 44 | 54 | 250 | 48 | 60 |
| 201 | 41 | 48 | 226 | 44 | 54 | | | |
| 202 | 41 | 48 | 227 | 45 | 55 | | | |
| 203 | 41 | 48 | 228 | 45 | 55 | | | |
| 204 | 42 | 48 | 229 | 45 | 55 | | | |
| 205 | 42 | 48 | 230 | 45 | 55 | | | |
| 206 | 42 | 48 | 231 | 45 | 56 | | | |
| 207 | 42 | 49 | 232 | 45 | 56 | | | |
| 208 | 42 | 49 | 233 | 46 | 56 | | | |
| 209 | 43 | 49 | 234 | 46 | 56 | | | |
| 210 | 43 | 49 | 235 | 46 | 56 | | | |
| 211 | 43 | 49 | 236 | 46 | 56 | | | |
| 212 | 43 | 50 | 237 | 46 | 56 | | | |

Table 3
Bounds on $\text{sz}(n)$

14 Appendix III: Comparing Methods For Finding Large 3-free Sets

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 2 | 2 | 2 | 2 | 2 |
| 4 | 3 | 3 | 3 | 2 | 3 |
| 5 | 4 | 4 | 4 | 3 | 4 |
| 6 | 4 | 4 | 4 | 4 | 4 |
| 7 | 4 | 4 | 4 | 4 | 4 |
| 8 | 4 | 4 | 4 | 4 | 4 |
| 9 | 4 | 5 | 5 | 4 | 5 |
| 10 | 5 | 5 | 5 | 5 | 5 |
| 11 | 6 | 6 | 6 | 5 | 6 |
| 12 | 6 | 6 | 6 | 6 | 6 |
| 13 | 7 | 7 | 7 | 6 | 7 |
| 14 | 8 | 8 | 8 | 7 | 8 |
| 15 | 8 | 8 | 8 | 8 | 8 |
| 16 | 8 | 8 | 8 | 8 | 8 |
| 17 | 8 | 8 | 8 | 8 | 8 |
| 18 | 8 | 8 | 8 | 8 | 8 |
| 19 | 8 | 8 | 8 | 8 | 8 |
| 20 | 8 | 8 | 8 | 8 | 9 |
| 21 | 8 | 8 | 8 | 9 | 9 |
| 22 | 8 | 8 | 8 | 9 | 9 |
| 23 | 8 | 8 | 8 | 9 | 9 |
| 24 | 8 | 8 | 8 | 9 | 10 |
| 25 | 8 | 9 | 9 | 10 | 10 |

Table 4
Comparing Methods for finding 3-free sets, 1-25

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 26 | 8 | 10 | 10 | 10 | 11 |
| 27 | 8 | 10 | 9 | 11 | 11 |
| 28 | 9 | 11 | 9 | 11 | 11 |
| 29 | 10 | 10 | 10 | 11 | 11 |
| 30 | 10 | 11 | 10 | 11 | 12 |
| 31 | 11 | 11 | 11 | 12 | 12 |
| 32 | 12 | 12 | 12 | 12 | 13 |
| 33 | 12 | 12 | 12 | 13 | 13 |
| 34 | 12 | 13 | 12 | 12 | 13 |
| 35 | 12 | 12 | 12 | 13 | 13 |
| 36 | 12 | 13 | 13 | 13 | 14 |
| 37 | 13 | 13 | 13 | 14 | 14 |
| 38 | 14 | 14 | 14 | 13 | 14 |
| 39 | 14 | 14 | 14 | 14 | 14 |
| 40 | 15 | 15 | 15 | 14 | 15 |
| 41 | 16 | 16 | 16 | 15 | 16 |
| 42 | 16 | 16 | 16 | 16 | 16 |
| 43 | 16 | 16 | 16 | 16 | 16 |
| 44 | 16 | 16 | 16 | 16 | 16 |
| 45 | 16 | 16 | 16 | 16 | 16 |
| 46 | 16 | 16 | 16 | 16 | 16 |
| 47 | 16 | 16 | 16 | 16 | 16 |
| 48 | 16 | 16 | 16 | 16 | 16 |
| 49 | 16 | 16 | 16 | 16 | 16 |
| 50 | 16 | 16 | 16 | 16 | 16 |

Table 5
Comparing Methods for finding 3-free sets, 26-50

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 51 | 16 | 16 | 16 | 16 | 17 |
| 52 | 16 | 16 | 16 | 17 | 17 |
| 53 | 16 | 16 | 16 | 17 | 17 |
| 54 | 16 | 16 | 16 | 17 | 18 |
| 55 | 16 | 16 | 16 | 18 | 18 |
| 56 | 16 | 16 | 16 | 18 | 18 |
| 57 | 16 | 16 | 16 | 18 | 18 |
| 58 | 16 | 16 | 16 | 18 | 19 |
| 59 | 16 | 16 | 16 | 19 | 19 |
| 60 | 16 | 16 | 16 | 18 | 19 |
| 61 | 16 | 17 | 16 | 18 | 19 |
| 62 | 16 | 16 | 16 | 19 | 19 |
| 63 | 16 | 17 | 16 | 19 | 20 |
| 64 | 16 | 17 | 17 | 20 | 20 |
| 65 | 16 | 18 | 17 | 19 | 20 |
| 66 | 16 | 17 | 17 | 19 | 20 |
| 67 | 16 | 17 | 16 | 19 | 20 |
| 68 | 16 | 17 | 16 | 19 | 20 |
| 69 | 16 | 16 | 16 | 20 | 20 |
| 70 | 16 | 17 | 16 | 19 | 20 |
| 71 | 16 | 18 | 16 | 20 | 21 |
| 72 | 16 | 18 | 17 | 20 | 21 |
| 73 | 16 | 19 | 17 | 20 | 21 |
| 74 | 16 | 18 | 18 | 21 | 22 |
| 75 | 16 | 19 | 17 | 21 | 22 |

Table 6
Comparing Methods for finding 3-free sets, 51-75

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 76 | 16 | 20 | 18 | 21 | 22 |
| 77 | 16 | 20 | 20 | 21 | 22 |
| 78 | 16 | 20 | 19 | 21 | 22 |
| 79 | 16 | 21 | 19 | 21 | 22 |
| 80 | 16 | 21 | 18 | 21 | 22 |
| 81 | 16 | 21 | 18 | 21 | 22 |
| 82 | 17 | 21 | 20 | 22 | 23 |
| 83 | 18 | 22 | 20 | 22 | 23 |
| 84 | 18 | 22 | 19 | 22 | 24 |
| 85 | 19 | 21 | 19 | 22 | 24 |
| 86 | 20 | 21 | 20 | 22 | 24 |
| 87 | 20 | 22 | 20 | 22 | 24 |
| 88 | 20 | 22 | 20 | 23 | 24 |
| 89 | 20 | 22 | 20 | 23 | 24 |
| 90 | 20 | 22 | 21 | 23 | 24 |
| 91 | 21 | 24 | 21 | 23 | 24 |
| 92 | 22 | 22 | 22 | 23 | 25 |
| 93 | 22 | 23 | 22 | 23 | 25 |
| 94 | 23 | 23 | 23 | 23 | 25 |
| 95 | 24 | 24 | 24 | 24 | 26 |
| 96 | 24 | 24 | 24 | 24 | 26 |
| 97 | 24 | 24 | 24 | 24 | 26 |
| 98 | 24 | 24 | 24 | 24 | 26 |
| 99 | 24 | 25 | 24 | 25 | 26 |
| 100 | 24 | 25 | 25 | 24 | 27 |

Table 7
Comparing Methods for finding 3-free sets, 76-100

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 101 | 24 | 26 | 25 | 25 | 27 |
| 102 | 24 | 26 | 24 | 24 | 27 |
| 103 | 24 | 25 | 24 | 25 | 27 |
| 104 | 24 | 25 | 24 | 25 | 28 |
| 105 | 24 | 26 | 24 | 25 | 28 |
| 106 | 24 | 26 | 25 | 25 | 28 |
| 107 | 24 | 26 | 26 | 26 | 28 |
| 108 | 24 | 26 | 25 | 26 | 28 |
| 109 | 25 | 27 | 27 | 26 | 28 |
| 110 | 26 | 27 | 26 | 26 | 28 |
| 111 | 26 | 27 | 27 | 27 | 29 |
| 112 | 27 | 27 | 27 | 26 | 29 |
| 113 | 28 | 28 | 28 | 26 | 29 |
| 114 | 28 | 28 | 28 | 27 | 30 |
| 115 | 28 | 29 | 28 | 27 | 30 |
| 116 | 28 | 29 | 28 | 26 | 30 |
| 117 | 28 | 29 | 28 | 27 | 30 |
| 118 | 29 | 29 | 28 | 27 | 30 |
| 119 | 30 | 30 | 29 | 27 | 30 |
| 120 | 30 | 30 | 29 | 27 | 30 |
| 121 | 31 | 31 | 30 | 28 | 31 |
| 122 | 32 | 32 | 32 | 28 | 32 |
| 123 | 32 | 32 | 32 | 29 | 32 |
| 124 | 32 | 32 | 32 | 28 | 32 |
| 125 | 32 | 32 | 32 | 28 | 32 |

Table 8
Comparing Methods for finding 3-free sets, 101-125

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 126 | 32 | 32 | 32 | 28 | 32 |
| 127 | 32 | 32 | 32 | 28 | 32 |
| 128 | 32 | 32 | 32 | 29 | 32 |
| 129 | 32 | 32 | 32 | 29 | 32 |
| 130 | 32 | 32 | 32 | 29 | 32 |
| 131 | 32 | 32 | 32 | 28 | 32 |
| 132 | 32 | 32 | 32 | 29 | 32 |
| 133 | 32 | 32 | 32 | 29 | 32 |
| 134 | 32 | 32 | 32 | 29 | 32 |
| 135 | 32 | 32 | 32 | 29 | 32 |
| 136 | 32 | 32 | 32 | 29 | 32 |
| 137 | 32 | 32 | 32 | 30 | 33 |
| 138 | 32 | 32 | 32 | 30 | 33 |
| 139 | 32 | 32 | 32 | 31 | 33 |
| 140 | 32 | 32 | 32 | 30 | 33 |
| 141 | 32 | 32 | 32 | 30 | 33 |
| 142 | 32 | 32 | 32 | 30 | 33 |
| 143 | 32 | 32 | 32 | 30 | 33 |
| 144 | 32 | 32 | 32 | 31 | 33 |
| 145 | 32 | 32 | 32 | 30 | 34 |
| 146 | 32 | 32 | 32 | 31 | 34 |
| 147 | 32 | 32 | 32 | 32 | 34 |
| 148 | 32 | 32 | 32 | 32 | 34 |
| 149 | 32 | 32 | 32 | 31 | 34 |
| 150 | 32 | 32 | 32 | 31 | 35 |

Table 9
Comparing Methods for finding 3-free sets, 126-150

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----|
| 151 | 32 | 32 | 32 | 32 | 35 |
| 152 | 32 | 32 | 32 | 31 | 35 |
| 153 | 32 | 32 | 32 | 32 | 35 |
| 154 | 32 | 32 | 32 | 31 | 35 |
| 155 | 32 | 32 | 32 | 31 | 35 |
| 156 | 32 | 32 | 32 | 32 | 35 |
| 157 | 32 | 32 | 32 | 32 | 36 |
| 158 | 32 | 32 | 32 | 33 | 36 |
| 159 | 32 | 32 | 32 | 32 | 36 |
| 160 | 32 | 32 | 32 | 32 | 36 |
| 161 | 32 | 32 | 32 | 32 | 36 |
| 162 | 32 | 32 | 32 | 32 | 36 |
| 163 | 32 | 32 | 32 | 32 | 37 |
| 164 | 32 | 32 | 32 | 32 | 37 |
| 165 | 32 | 32 | 32 | 32 | 38 |
| 166 | 32 | 32 | 32 | 33 | 38 |
| 167 | 32 | 32 | 32 | 34 | 38 |
| 168 | 32 | 32 | 32 | 33 | 38 |
| 169 | 32 | 32 | 32 | 33 | 39 |
| 170 | 32 | 32 | 32 | 33 | 39 |
| 171 | 32 | 32 | 32 | 33 | 39 |
| 172 | 32 | 32 | 32 | 33 | 39 |
| 173 | 32 | 32 | 32 | 33 | 39 |
| 174 | 32 | 32 | 32 | 33 | 40 |
| 175 | 32 | 32 | 32 | 33 | 40 |

Table 10
Comparing Methods for finding 3-free sets, 151-175

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----------|
| 176 | 32 | 33 | 32 | 34 | 40 |
| 177 | 32 | 33 | 32 | 33 | 40 |
| 178 | 32 | 32 | 32 | 33 | 40 |
| 179 | 32 | 33 | 32 | 34 | 40 |
| 180 | 32 | 33 | 32 | 34 | 40 |
| 181 | 32 | 33 | 32 | 34 | 40 |
| 182 | 32 | 33 | 32 | 34 | 40 |
| 183 | 32 | 33 | 32 | 34 | 40 |
| 184 | 32 | 33 | 32 | 35 | 40 |
| 185 | 32 | 33 | 32 | 35 | 40 |
| 186 | 32 | 33 | 32 | 35 | 40 |
| 187 | 32 | 34 | 32 | 35 | 40 |
| 188 | 32 | 34 | 33 | 35 | ≥ 40 |
| 189 | 32 | 35 | 33 | 34 | ≥ 40 |
| 190 | 32 | 34 | 33 | 35 | ≥ 40 |
| 191 | 32 | 34 | 34 | 36 | ≥ 40 |
| 192 | 32 | 35 | 33 | 35 | ≥ 40 |
| 193 | 32 | 35 | 34 | 35 | ≥ 40 |
| 194 | 32 | 35 | 33 | 35 | ≥ 41 |
| 195 | 32 | 36 | 32 | 36 | ≥ 41 |
| 196 | 32 | 35 | 35 | 36 | ≥ 41 |
| 197 | 32 | 34 | 33 | 37 | ≥ 41 |
| 198 | 32 | 34 | 32 | 36 | ≥ 41 |
| 199 | 32 | 35 | 33 | 36 | ≥ 41 |
| 200 | 32 | 35 | 33 | 37 | ≥ 41 |

Table 11
Comparing Methods for finding 3-free sets, 176-200

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----------|
| 201 | 32 | 34 | 32 | 36 | ≥ 41 |
| 202 | 32 | 34 | 33 | 36 | ≥ 41 |
| 203 | 32 | 34 | 32 | 36 | ≥ 41 |
| 204 | 32 | 34 | 32 | 36 | ≥ 42 |
| 205 | 32 | 35 | 32 | 36 | ≥ 42 |
| 206 | 32 | 34 | 32 | 37 | ≥ 42 |
| 207 | 32 | 34 | 32 | 37 | ≥ 42 |
| 208 | 32 | 34 | 32 | 36 | ≥ 42 |
| 209 | 32 | 34 | 32 | 37 | ≥ 43 |
| 210 | 32 | 34 | 32 | 37 | ≥ 43 |
| 211 | 32 | 34 | 32 | 38 | ≥ 43 |
| 212 | 32 | 34 | 32 | 37 | ≥ 43 |
| 213 | 32 | 35 | 32 | 37 | ≥ 43 |
| 214 | 32 | 35 | 33 | 37 | ≥ 43 |
| 215 | 32 | 36 | 34 | 38 | ≥ 44 |
| 216 | 32 | 37 | 34 | 38 | ≥ 44 |
| 217 | 32 | 36 | 34 | 39 | ≥ 44 |
| 218 | 32 | 37 | 34 | 38 | ≥ 44 |
| 219 | 32 | 37 | 35 | 38 | ≥ 44 |
| 220 | 32 | 37 | 34 | 38 | ≥ 44 |
| 221 | 32 | 37 | 36 | 38 | ≥ 44 |
| 222 | 32 | 37 | 36 | 39 | ≥ 44 |
| 223 | 32 | 37 | 35 | 38 | ≥ 44 |
| 224 | 32 | 37 | 35 | 38 | ≥ 44 |
| 225 | 32 | 37 | 36 | 38 | ≥ 44 |

Table 12
Comparing Methods for finding 3-free sets, 201-225

| n | Base3 | VC-Rand | VC-Det | Rand | Opt |
|-----|-------|---------|--------|------|-----------|
| 226 | 32 | 38 | 36 | 38 | ≥ 44 |
| 227 | 32 | 38 | 37 | 39 | ≥ 45 |
| 228 | 32 | 38 | 37 | 39 | ≥ 45 |
| 229 | 32 | 39 | 38 | 39 | ≥ 45 |
| 230 | 32 | 40 | 40 | 39 | ≥ 45 |
| 231 | 32 | 40 | 40 | 39 | ≥ 45 |
| 232 | 32 | 40 | 39 | 38 | ≥ 45 |
| 233 | 32 | 40 | 38 | 39 | ≥ 46 |
| 234 | 32 | 40 | 38 | 39 | ≥ 46 |
| 235 | 32 | 41 | 38 | 39 | ≥ 46 |
| 236 | 32 | 42 | 38 | 40 | ≥ 46 |
| 237 | 32 | 42 | 37 | 40 | ≥ 46 |
| 238 | 32 | 41 | 37 | 40 | ≥ 46 |
| 239 | 32 | 41 | 36 | 40 | ≥ 47 |
| 240 | 32 | 41 | 37 | 40 | ≥ 47 |
| 241 | 32 | 41 | 36 | 40 | ≥ 47 |
| 242 | 32 | 41 | 38 | 40 | ≥ 47 |
| 243 | 32 | 41 | 39 | 40 | ≥ 47 |
| 244 | 33 | 42 | 38 | 40 | ≥ 47 |
| 245 | 34 | 42 | 40 | 41 | ≥ 47 |
| 246 | 34 | 42 | 39 | 40 | ≥ 47 |
| 247 | 35 | 42 | 37 | 41 | ≥ 48 |
| 248 | 36 | 43 | 38 | 41 | ≥ 48 |
| 249 | 36 | 43 | 36 | 40 | ≥ 48 |
| 250 | 36 | 44 | 37 | 41 | ≥ 48 |

Table 13
Comparing Methods for finding 3-free sets, 226-250

15 Appendix IV: When did Linear Programming Help?

Using Backtracking we obtained $sz(n)$ for $1 \leq n \leq 187$. Using splitting we then obtained upper and lower bounds on $sz(n)$ for $188 \leq n \leq 250$. We then improved some of the upper bounds using the Linear Programming technique described in section 7. The table in Appendix II is the result of all of these techniques. In this section we list the values where linear programming was used to get a better upper bound than splitting.

| n | splitting bound on $sz(n)$ | LinProg bound on $sz(n)$ | improvement |
|-----|----------------------------|--------------------------|-------------|
| 211 | 50 | 49 | 1 |
| 213 | 51 | 50 | 1 |
| 217 | 52 | 51 | 1 |
| 218 | 52 | 51 | 1 |
| 219 | 53 | 51 | 2 |
| 220 | 53 | 52 | 1 |
| 221 | 53 | 52 | 1 |
| 222 | 53 | 52 | 1 |
| 223 | 54 | 52 | 2 |
| 224 | 54 | 53 | 1 |
| 228 | 56 | 55 | 1 |
| 229 | 56 | 56 | 1 |
| 230 | 56 | 55 | 1 |
| 245 | 57 | 56 | 1 |

References

- [1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *Journal of the ACM*, 45, 1998. Prior version in FOCS92.
- [2] F. Behrend. On set of integers which contain no three in arithmetic progression. *Proc. of the National Academy of Science (USA)*, 23:331–332, 1946.
- [3] R. Beigel, W. Gasarch, and J. Glenn. The multiparty communication complexity of exact- t : improved bounds and new problems. In *Proceedings of the 31th International*

- Symposium on Mathematical Foundations of Computer Science 2001*, Stara Lesna, Slovakia, 2006.
- [4] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993.
 - [5] J. Bourgain. On triples in arithmetic progression. *Geometric and Functional Analysis*, 9:968–984, 1999.
 - [6] A. Chandra, M. Furst, and R. Lipton. Multiparty protocols. In *Proceedings of the Fifteenth Annual ACM Symposium on the Theory of Computing*, Boston MA, pages 94–99, 1983.
 - [7] E. Cockayne and S. Hedetniemi. On the diagonal queens domination problem. *Journal of Combinatorial Theory, Series A*, 42:137–139, 1986.
 - [8] Cohn, Kleinberg, Szegedy, and Umans. Group-theoretic algorithms for matrix multiplication. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, Pittsburgh PA, 2005.
 - [9] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990. Earlier Version in STOC87.
 - [10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, 2003.
 - [11] Cplex package for linear and integer programming. www.ilog.com/products/cplex/.
 - [12] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw Hill, 2008.
 - [13] P. Erdos and P. Turan. On some sequences of integers. *Journal of the London Mathematical Society*, 11(2):261–264, 1936.
 - [14] H. Furstenberg. Ergodic behaviour of diagonal measures and a theorem of Szemerédi on arithmetic progressions. *J. Analyse Math.*, pages 204–256, 1977.
 - [15] Glpk package for linear and integer programming. www.gnu.org/software/glpk/glpk.html.
 - [16] W. Gowers. A new proof for Szemerédi’s theorem for arithmetic progressions of length four. *Geometric and Functional Analysis*, 8:529–551, 1998.
 - [17] W. Gowers. A new proof of Szemerédi’s theorem. *Geometric and Functional Analysis*, 11:465–588, 2001.
 - [18] R. Graham, A. Rothchild, and J. Spencer. *Ramsey Theory*. Wiley, 1990.
 - [19] B. Green. On triples in arith. prog., 1999. www.dpmms.cam.ac.uk/~bjg23/papers/bourgain.pdf.
 - [20] B. Green. Progressions of length 3 following Szemerédi, 1999. www.dpmms.cam.ac.uk/~bjg23/papers/newszem.pdf.
 - [21] R. Guy. *Unsolved problems in number theory*. Springer Verlag, New York, 1981.
 - [22] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, Burlington VT, 1996. Also at Electronic Colloquium on Computational Complexity www.eccc.uni-trier.de/eccc.
 - [23] J. Hastad and A. Wigderson. Simple analysis of graph tests for linearity. *Random Structures and Algorithms*, 22, 2003. Prior version in Complexity 2001.
 - [24] D. Heath-Brown. Integer sets containing no arithmetic progressions. *Proc. of the London*

- Math Society*, 35(2):385–394, 1987.
- [25] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley, 2006.
 - [26] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.
 - [27] A. Levitin. *Design and analysis of algorithms*. Addison-Wesley, 2007.
 - [28] L. Moser. On non-averaging sets of integers. *Canadian Journal of Mathematics*, 5:245–252, 1953.
 - [29] G. Rawlins. *Compared to what? An introduction to the analysis of algorithms*. Freeman, 1992.
 - [30] K. Roth. Sur quelques ensembles d’entiers. *C.R. Acad. Sci Paris*, 234:388–3901, 1952.
 - [31] K. Roth. On certain sets of integers. *Journal of the London Mathematical Society*, 28:104–109, 1953.
 - [32] I. Ruzsa. Erdos and the numbers. *Journal of Number Theory*, pages 115–163, 1999.
 - [33] I. Ruzsa and E. Szemerédi. Triple systems with no six points carrying three triangles. In *Combinatorics, Fifth Hungarian Colloquium, Keszthely*, 1976.
 - [34] S. A. S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45, 1998. Prior version in FOCS92.
 - [35] R. Salem and D. Spencer. On set of integers which contain no three in arithmetic progression. *Proc. of the National Academy of Science (USA)*, 28:561–563, 1942.
 - [36] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the Thirty-second Annual ACM Symposium on the Theory of Computing*, Portland OR, pages 191–199, 2000.
 - [37] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14:354–356, 1969.
 - [38] E. Szemerédi. On sets of integers containing no four elements in arithmetic progression. *Acta Math. Sci. Hung.*, 20:89–104, 1969.
 - [39] E. Szemerédi. On sets of integers containing no k elements in arithmetic progression. *Acta Arith.*, 27:299–345, 1986.
 - [40] E. Szemerédi. Integer sets containing no arithmetic progressions. *Acta Math. Sci. Hung.*, 56:155–158, 1990.
 - [41] B. van der Waerden. Beweis einer Baudetschen vermutung. *Nieuw Arch. Wisk.*, 15:212–216, 1927.
 - [42] S. Wagstaff. On k -free sequences of integers. *Mathematics of Computation*, pages 767–771, 1972.
 - [43] J. Wroblewski. Nonaveraging sets search. www.math.uni.wroc.pl/~jwr/non-ave/index.htm.