BERNOULLI CONVOLUTIONS: A COMBINATORIAL APPROACH

JULIA DAVIS, MICHELLE DELCOURT, AND ZEBEDIAH ENGBERG

Abstract. In this paper we consider a combinatorial analogue of the Bernoulli convolution problem from real analysis. We discuss several innovative algorithms for computing the sequences in this new approach. In particular, these algorithms assist us in gathering data regarding the maximum values and give an improvement on the best known bound. This work was completed as part of the Clemson University REU, an NSF funded program[1].

1. Introduction

The classic Bernoulli convolution problem in analysis has an elegant combinatorial analogue which we now describe. Consider the two maps $\mathrm{dup}_n, \mathrm{shf}_n : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ defined by

$$\mathrm{dup}_n : (a_1, a_2, ..., a_{n-1}, a_n) \longmapsto (a_1, a_1, a_2, a_2, ..., a_{n-1}, a_{n-1}, a_n, a_n, \overbrace{0, ..., 0}^{n \text{ times}}) \tag{1}$$

$$\mathrm{shf}_n : (a_1, a_2, ..., a_{n-1}, a_n) \longmapsto (\overbrace{0, ..., 0}^{n \text{ times}}, a_1, a_1, a_2, a_2, ..., a_{n-1}, a_{n-1}, a_n, a_n). \tag{2}$$

The names "dup" and "shf" reference the duplication and shifting of the coordinates. Consider the finite sequences of increasing length given by $B_0 = (1)$ and $B_{n+1} = \mathrm{dup}_n(B_n) + \mathrm{shf}_n(B_n)$. In this paper, we are primarily interested in the rate at which the maximum m_n of B_n is growing with n. We develop three independent algorithms: the first gives a recursive method for computing the entire sequence B_n when n is small, the second gives a method for computing given entries of B_n when n is larger, and the third improves the best known bound on the growth of m_n. Additionally, we provide numerical data and put forth several conjectures concerning various properties of the sequence B_n.

1.1. **Motivation.** Classically, Bernoulli convolutions have been studied as a problem in real analysis. A Bernoulli convolution is a measure obtained as an infinite convolution of Bernoulli measures [1]. This concept was first studied by Jessen and Wintner [4]. In our research, we look at an equivalent description of the classic Bernoulli convolution problem. For $0 < q < 1$, consider the functional equation

$$F(t) = \frac{1}{2}F\left(\frac{t-1}{q}\right) + \frac{1}{2}F\left(\frac{t+1}{q}\right) \tag{3}$$

for t on the interval $I_q := [-1/(1-q), 1/(1-q)]$. It can be shown that there is a unique continuous solution $F_q(t)$ to the above equation; for a very good introduction to this refer to Chapter 5 in *Experimental Mathematics in Action* [1]. For a more in depth report on what is known, refer to *Sixty years of Bernoulli convolutions* [6].

The major question regarding the solutions of (3) is that of determining the values of q that make $F_q(t)$ absolutely continuous and the values that make $F_q(t)$ singular. When $0 < q < 1/2$, Kershner and Wintner [5] have shown that $F_q(t)$ is always singular. For these values of q, the solution $F_q(t)$

Date: July 5, 2008.

[1] This research was supported by NSF grant DMS-0552799.

is an example of a *Cantor function*, a function that is constant almost everywhere. It is also easy to see that for $q = 1/2$, the solution $F_q(t)$ is absolutely continuous. The case when $q > 1/2$ is much harder and more interesting. In 1939, Erdős [3] showed that if q is of the form $q = 1/\theta$ where θ is a Pisot number, then $F_q(t)$ is again singular. There is little else that is known for other values of $q > 1/2$. One interesting result due to Solomyak [7] is that almost every $q > 1/2$ yields a solution $F_q(t)$ that is absolutely continuous. Hence it is surprising that no actual example of such a q is known. Specifically, the obvious case when $q = 2/3$ remains a mystery.

Rather than looking at the function $F_q(t)$, one can also consider its derivative $f_q(t) = F_q'(t)$. Upon differentiating, the functional equation for $F_q(t)$ gives the following equation for $f_q(t)$:

$$f(t) = \frac{1}{2q} f\left(\frac{t-1}{q}\right) + \frac{1}{2q} f\left(\frac{t+1}{q}\right). \tag{4}$$

The question of the existence of an absolutely continuous solution $F_q(t)$ to (3) is equivalent to the the existence of an $L^1(I_q)$ solution $f_q(t)$ to (4).

In [1], Girgensohn asks the question of computing $f_q(t)$ for various values of q. The author considers starting with an arbitrary initial function $f^0(t) \in L^1(I_q)$ and iterating the transform

$$T_q : f(t) \longmapsto \frac{1}{2q} f\left(\frac{t-1}{q}\right) + \frac{1}{2q} f\left(\frac{t+1}{q}\right) \tag{5}$$

to gain a sequence of functions $f^0, f^1, f^2, \dots$. If this sequence converges, then it converges to the solution of (4).

Neil Calkin [2] then looked at the above process for $q = 2/3$. Rather then working on the interval I_q, we shift the entire interval to $[0, 1]$ for simplicity. The transform T_q now becomes the transform $T : L^1([0, 1]) \longrightarrow L^1([0, 1])$ where

$$T : f(x) \longmapsto \frac{3}{4} f\left(\frac{3x}{2}\right) + \frac{3}{4} f\left(\frac{3x-1}{2}\right). \tag{6}$$

Intuitively, this transform (6) takes two scaled copies of $f(x)$: one on the interval $[0, 2/3]$ and the other on $[1/3, 1]$, and adds them. The scaling factor of $3/4$ gives us that

$$\int_0^1 f(x)dx = \int_0^1 Tf(x)dx,$$

in other words the average value of $Tf(x)$ is the same as that of $f(x)$. In this setting, the question to be answered is: starting with the function $f^0(x) = 1$, does the iteration determined by the transform in (6) converge to a bounded function?

1.2. **Combinatorial approach.** Instead of viewing T as a transform on $[0, 1]$, we consider the combinatorial analogue initially mentioned. The sequences $\mathrm{dup}(B_n)$ and $\mathrm{shf}(B_n)$ defined in (1) and (2) are analogous to the two shifted copies of the function on $[0, 1]$. We start with the sequence $B_0 = (1)$. We recursively generate sequences given by

$$B_{n+1} = \mathrm{dup}_n(B_n) + \mathrm{shf}_n(B_n). \tag{7}$$

We now discuss some notation that we find useful throughout this paper. We call B_n the *Bernoulli sequence on level n*. Likewise, we refer to the map $(\mathrm{dup}_n + \mathrm{shf}_n) : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ seen in (7) as the process of *duplicate, shift, add* or DSA for short. When indexing n^{th} level Bernoulli sequence, we start from 0 rather than 1. We write

$$B_n = (b_0, b_1, ..., b_{3^n-1}).$$

The fact that B_n has a total of 3^n terms follows directly from the definition of dup_n and shf_n in (1) and (2).

$$\begin{array}{ccccc} 1 & \longrightarrow & 1 & 1 & \\ & & & 1 & 1 \\ \hline & & 1 & 2 & 1 \end{array}$$

$$\begin{array}{ccccccccccccc} 1 & 2 & 1 & \longrightarrow & 1 & 1 & 2 & 2 & 1 & 1 & & & \\ & & & & & & & 1 & 1 & 2 & 2 & 1 & 1 \\ \hline & & & & 1 & 1 & 2 & 3 & 2 & 3 & 2 & 1 & 1 \end{array}$$

$$\begin{array}{cccccccccccccccccccccccccccccccccccccc} 1 & 1 & 2 & 3 & 2 & 3 & 2 & 1 & 1 & \longrightarrow & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 3 & 3 & 2 & 2 & 1 & 1 & 1 & 1 & & & & & & & & & \\ & & & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 3 & 3 & 2 & 2 & 1 & 1 & 1 & 1 \\ \hline & & & & & & & & & & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 2 & 3 & 4 & 4 & 3 & 4 & 3 & 4 & 4 & 3 & 2 & 3 & 3 & 2 & 2 & 1 & 1 & 1 & 1 \end{array}$$

FIGURE 1. This shows the process of DSA at three low levels. At the top, we see how to compute the first level from the zeroth. In the middle, we see how to compute the second level from the first. Finally we see how to compute the third level from the second. This figure also demonstrates the $O(3^n)$ growth in the length of B_n.

1.3. **Overview of the problem.** As students in the 2008 Clemson University REU, Julia Davis, Michelle Delcourt, and Zebediah Engberg worked under the direction of Neil Calkin and Kevin James on several questions regarding the Bernoulli sequences B_n. One of the major problems we considered was putting a reasonable bound on the growth rate of $m_n := \max(B_n)$. It is immediate that the maximum value must occur on the middle third of the Bernoulli sequence. Furthermore, because the sequence is palindromic, the maximum must occur on the first half of the middle third. Hence we gathered a significant amount of data on the entries in this range. It is easy to see that the mean $\mu(B_n) = (4/3)^n$ (see Section 5.1). The major question that we considered was if m_n also grows like $O((4/3)^n)$. This is the combinatorial analogue of the existence of a bounded solution to the functional equation (4). Beginning with a computational approach, we address this and other related questions throughout this paper.

2. Recursive algorithms

2.1. **The naive algorithm: DSA.** The process of *duplicate, shift, add* gives a naive method for computing Bernoulli sequences. Despite the simplicity in describing this process, DSA is not computationally feasible for large values of n. One concern with the DSA method is that each level B_n has 3^n terms—each successive level computation takes three times as long as the previous. Likewise, at each successive level we must store three times as many entries as on the previous level. Using the DSA method, we have been able to compute B_n up to $n = 20$. With better computing facilities we could possibly push this method several levels further, but certainly not significantly further. For a graphical representation of various levels, see Figure 2 and Figure 3 in the Appendix.

2.2. **The sophisticated algorithm: DEM.** We now consider an alternate approach that addresses some of the issues arising with the DSA algorithm. The process we call *double, enlarge, merge*, abbreviated DEM, is a way of encoding the Bernoulli sequence B_n as a sequence of length $2(2^n-1)$. The advantage with DEM is that the sequence grows in size like 2^n as opposed to the 3^n size increase required for the DSA process. The DEM algorithm is based on the observation that in a given Bernoulli sequence, many individual entries are consecutively repeated. Rather than keeping consecutive repeats, we only keep the entries where the Bernoulli sequence either increases or decreases. It is important to note that when comparing two consecutive entries in a Bernoulli sequence, the jump between these entries will never be greater than one. This can easily be proved inductively. Given the Bernoulli sequence B_n, the DEM representation is $(d_1, ..., d_r)$ where d_i is the index of the i^{th} jump in B_n up to a sign. Suppose the i^{th} jump occurs at index j in B_n, that is b_j and b_{j+1} are different. Then

$$d_i = \begin{cases} j & \text{if } b_j < b_{j+1} \\ -j & \text{if } b_j > b_{j+1}. \end{cases}$$

For example, the DEM representation of $B_2 = (1,1,2,3,2,3,2,1,1)$ is $(2,3,-4,5,-6,-7)$.

We now describe how to translate the process of DSA to this new representation. The process of *duplicate* becomes *double*: each element from the original list is multiplied by two. The process of *shift* becomes *enlarge*: each element from the original list is modified by 3^n. If the element is positive, we add 3^n, for negative elements we subtract 3^n. The process of add translates to *merge*: we discard the original elements and concatenate the new lists attained in the *double* and *enlarge* processes. We then add two additional elements $-2(3^n)$ and 3^n to the list. Finally, we merge sort the elements according to their absolute value.

2.3. **Data.** Using the DEM method, we have been able to compute the Bernoulli sequence up to level $n = 26$. For each of these levels, the maximum value is of particular interest. Refer to Table 1 in the Appendix for numerical data on these maximums.

3. A Polynomial Approach to DSA

3.1. **Translating DSA as a polynomial recursion.** By encoding these sequences as coefficients of polynomials, the process of *duplicate, shift, add* gives a particularly nice recursive relation among the polynomials. Let $B_n = (b_0, b_1, ..., b_t)$ be the Bernoulli sequence on level n where $t = 3^n - 1$. Consider the polynomial $p_n(x) := b_0 + b_1x + ... + b_tx^t$.

We create a dictionary that translates the process of DSA into recursive relations among the polynomials. We see that the duplication $b_0, b_0, b_1, b_1, ..., b_r, b_r$ corresponds to the polynomial $(1 + x)p_n(x^2)$. Shifting the sequence 3^n places to the right corresponds to multiplication by x^{3^n}. By adding the duplicate and the shift of the sequence, we get the sequence on the next level. This yields the recurrence relation

$$p_{n+1}(x) = (1+x)p_n(x^2)\left(1+x^{3^n}\right).$$

3.2. **Explicit Formula.** This formula allows us to explicitly solve for $p_n(x)$.

Theorem 3.1. *The polynomials $p_n(x)$ satisfy*

$$p_n(x) = \prod_{i=0}^{n-1}\left(1+x^{2^i}\right)\prod_{j=0}^{n-1}\left(1+x^{2^{n-1}(3/2)^j}\right). \tag{8}$$

Proof. We proceed by induction on n. When $n = 1$, we have that

$$1 + 2x + x^2 = (1 + x)(1 + x) = (1 + x^{2^0})(1 + x^{2^0(3/2)^0}).$$

Now assume the formula holds for $p_n(x)$. We will show that it holds for $p_{n+1}(x)$. We have

$$\begin{aligned} p_{n+1}(x) &= (1+x)p_n(x^2)\left(1+x^{3^n}\right) \\ &= (1+x)\prod_{i=0}^{n-1}\left(1+(x^2)^{2^i}\right)\prod_{j=0}^{n-1}\left(1+(x^2)^{2^{n-1}(3/2)^j}\right)\left(1+x^{3^n}\right) \\ &= (1+x)\prod_{i=0}^{n-1}\left(1+x^{2^{i+1}}\right)\prod_{j=0}^{n-1}\left(1+x^{2^n(3/2)^j}\right)\left(1+x^{2^n(3/2)^n}\right) \\ &= \prod_{i=0}^{n}\left(1+x^{2^i}\right)\prod_{j=0}^{n}\left(1+x^{2^n(3/2)^j}\right). \end{aligned}$$

□

3.3. **A Bound on the Coefficients.** By factoring p_n in a clever way, we can put a bound on how fast the coefficients grow with the level n.

Theorem 3.2. ([2]) *On level n, the maximum value $m_n = O((\sqrt{2})^n)$.*

Proof. To start, define polynomials q_n, r_n, s_n by

$$q_n(x) = \prod_{i=0}^{n-1}\left(1+x^{2^i}\right) \qquad s_n(x) = \prod_{\substack{1\le j\le n-1 \\ j \text{ odd}}}\left(1+x^{2^{n-1}(3/2)^j}\right)$$

$$r_n(x) = \prod_{\substack{1\le j\le n-1 \\ j \text{ even}}}\left(1+x^{2^{n-1}(3/2)^j}\right) = \prod_{j=1}^{\lfloor (n-1)/2\rfloor}\left(1+x^{2^{n-1}(9/4)^j}\right).$$

We see that

$$p_n(x) = q_n(x)\left(1+x^{2^{n-1}}\right)r_n(x)s_n(x).$$

Consider the polynomial $q_n(x)r_n(x)$. Because $9/4 > 2$, we have distinct powers of x when we expand $q_n(x)r_n(x)$. In other words, the coefficients are all either 0 or 1. Hence the coefficients of $q_n(x)r_n(x)(1 + x^{2^{n-1}})$ are all either 0, 1, or 2. In particular, the coefficients are bounded. On the other hand, there are at most $n/2$ terms in the product defining $s_n(x)$. Hence there are at most $2^{n/2}$ nonzero terms in the polynomial $s_n(x)$ since we have 2 choices from each term in the product. Therefore the coefficients of $p_n(x)$ are all $O(2^{n/2}) = O((\sqrt{2})^n)$. □

How far can this bound be improved? Looking at numerical data in Table 1 in the Appendix, it appears as if $m_n = O((4/3)^n)$. Also see the plot in Figure 6. For more on improving this bound, see Section 4.

3.4. **An algorithmic implementation: PIP.** The fact that our sequence can be realized as the coefficients of an explicitly defined polynomial provides us with an algorithm for computing isolated points on high levels. The algorithms DSA and DEM are useful for computing entire levels, but this becomes impossible for large n due to the recursive nature of the algorithm. The following algorithm, which we call PIP, allows us to compute with much larger n values. The algorithm is nonrecursive—we need no previous levels to compute entries on B_n. The name PIP stands for *polynomial isolated point* because this algorithm computes the entry corresponding to a given index and given level number.

Our algorithm is based on the following idea. Suppose $S = \{a_1, ..., a_n\}$ is a set of distinct positive integers. Consider the polynomial

$$f(x) = \prod_{i=1}^{n}(1 + x^{a_i}) = \sum_{j=1}^{m} \alpha_j x^j.$$

Then α_j is the number of ways to write j as a sum of distinct elements from S. This idea is applicable to the coefficients of our polynomial because our polynomial is a product of terms of the form $(1+x^a)$. The only difficulty that arises is that the powers of x in the terms in the product are not distinct—indeed the term $(1+x^{2^{n-1}})$ is repeated twice. To this end we do not view S in (9) as a set. This poses no problem; the above argument goes through when thinking of S as a sequence or ordered tuple. Hence we get that the coefficient b_j of x^j in $p_n(x)$ (which is the j^{th} entry on the n^{th} Bernoulli sequence) is precisely the number of ways that j can be written as a sum of distinct terms in the sequence

$$S = \{1, 2, 4, ..., 2^{n-2}, 2^{n-1}, 2^{n-1}, 2^{n-2}3, 2^{n-3}3^2, ..., 2^2 3^{n-3}, 2^1 3^{n-2}, 3^{n-1}\}. \tag{9}$$

We now outline an algorithm that can be used to calculate the entry b_j for a fixed level n. The entire algorithm is based on the following ideas:

- Let $S = \{a_1, ..., a_n\}$ where each $a_i > 0$. Let $\mathrm{N}_S(k)$ denote the number of ways to write k as a sum of elements from S. Then for any $i \in \{1, ..., n\}$, the following holds
$$\mathrm{N}_S(k) = \mathrm{N}_{S\setminus\{a_i\}}(k) + \mathrm{N}_{S\setminus\{a_i\}}(k - a_i). \tag{10}$$
- If $k > \sum_{s\in S} s$, then $\mathrm{N}_S(k) = 0$.
- If $k < 0$, then $\mathrm{N}_S(k) = 0$.
- Let S be as in (9). We see that if $0 < k < 2^{n-1}$, then $\mathrm{N}_S(k) = \mathrm{N}_{S'}(k)$ where $S' = \{1, 2, 4, ..., 2^{n-2}\}$ since all other elements of S are too large. However, every k with $0 < k < 2^{n-1}$ can be written uniquely as a sum from elements of S'; this is simply the binary expansion of k.

3.5. **Data.** In Table 2 and Table 3 found in the Appendix, we use the PIP algorithm to compute isolated points on high levels. We first pick $\alpha \in [0, 1]$. We then consider the index $k = \lceil \alpha(3^n - 1)\rceil$. We compute the entry b_k at index k for levels $n = 1, 2, ..., 40$ (we have used PIP to compute entries on levels as high as $n = 70$). Finally we take the quotient of b_k by $(4/3)^n$. Table 2 shows evenly spaced α values within the first half of the middle third of the sequence (remember that this is the only interesting piece of B_n). Table 3 shows an accumulation of α values as α gets close to $1/2$. Also of interest is the flowchart in Figure 4 which shows an explicit example of our implementation of the PIP algorithm in use. Further data gathered using PIP is found in Table 1 containing the maximum values (and approximate maximum values) of the first 32 levels.

4. An Improvement on the Bound

4.1. **Background.** Seeing that our sequence on level n has length 3^n, we naturally index it by the first 3^n nonnegative integers. In certain circumstances, it is advantageous to normalize the indexing in such a way that each index is on the interval $[0,1]$. To this end, we can simply take the image of $k \in \{0,1,2,...,3^n-1\}$ under the map $k \mapsto k/3^n$. This normalization scheme has the advantage that each level can be made to live on $[0,1]$; indeed for a subset $S \subset [0,1]$, we define

$$\Gamma_n(S) = \max_{x\in\bar{S}} g_n(x)$$

where $\bar{S} = S \cap \{0, 1/3^n, 2/3^n, ..., (3^n-1)/3^n\}$ and $g_n(x)$ denotes the n^{th} level Bernoulli sequence where now $x \in [0,1]$. In other words,

$$g_n\left(\frac{k}{3^n}\right) = b_k \qquad \text{for } k = 0,1,...3^n-1. \tag{11}$$

Consider the two maps predup, preshf : $[0,1]\cup\emptyset \longrightarrow [0,1]\cup\emptyset$ defined by

$$\text{predup} : x \longmapsto \begin{cases} (3/2)x & \text{if } x \in\ [0,2/3] \\ \emptyset & \text{otherwise} \end{cases} \qquad \text{preshf} : x \longmapsto \begin{cases} (3/2)(x+1/3) & \text{if } x \in\ [1/3,1] \\ \emptyset & \text{otherwise.} \end{cases}$$

Also predup$(\emptyset) = \emptyset$ and predup$(\emptyset) = \emptyset$. The normalization scheme is now apparent—the functions predup and preshf are defined independently of n.

For a given entry $g_n(x)$ on the n^{th} level, $g_n(x)$ can be written as a sum of two entries on level $n-1$ provided x is in the middle third. If x is in the first third or the last third, then $g_n(x)$ only comes from one entry on the previous level. The maps predup and preshf give the preimage of the index x on the duplicated copy of the $(n-1)^{th}$ level and the shifted copy of the $(n-1)^{th}$ level, respectively. This explains the strange names "predup" and "preshf". These maps reflect the behavior of the indices in the duplication process and shifting. Indeed, in this new setting the process of *duplicate, shift, add* translates to the equation

$$g_{n+1}(x) = g_n(\text{predup}(x)) + g_n(\text{preshf}(x)) \tag{12}$$

where $g_n(\emptyset) = 0$.

4.2. **Description of Algorithm.** We wish to give a reasonable bound on the growth of $\Gamma_n := \Gamma_n([0,1])$ in terms of n. Realize that Γ_n is equivalent terminology for m_n; the new notation reinforces the fact that we are now thinking of the indices of B_n as living on $[0,1]$. The following procedure gives a method for computing a positive real number θ so that $\Gamma_n = O(\theta^n)$. The exact number θ will depend on several factors discussed in the remarks in Section 4.4.

(1) Write the interval $[0,1]$ as a union

$$[0,1] = \bigcup_{j=1}^{w} D_j \tag{13}$$

where each D_j is a closed interval.

(2) Now fix an index j and let $D = D_j$. Consider the image of D under predup and preshf. We pullback D one level to obtain the two intervals predup(D) and preshf(D). Note that one of predup(D) or preshf(D) maybe be empty, but this poses no problem. Pulling D back two levels, we are left with the four intervals

$$(\text{predup}(\text{predup})(D)),\ \ (\text{preshf}(\text{predup}(D)),\ \ (\text{predup}(\text{preshf}(D)),\ \ (\text{preshf}(\text{preshf})(D)).$$

Continuing in this manner, we are left with 2^r intervals after considering the image of D under all pullbacks to level $n-r$. Call these intervals $E_1, E_2, ..., E_{2^r}$.

(3) For each $i = 1, 2, ..., 2^r$, consider the particular interval E_i. If E_i is empty, we disregard E_i. For technical reasons, if E_i only contains 0 or 1, we also disregard it. The reason for this is that $\Gamma_n(\{0\}) = 1$ and $\Gamma_n(\{1\}) = 0$. Hence intervals will contribute nothing—this will be clear in the proof. Otherwise there exists a $y \in E_i$ so that $|y - 1/2| \leq |w - 1/2|$ for all other $w \in E_i$. In other words, we select the element from E_i that is closest to 1/2. If $y > 1/2$, replace y with $1 - y \in (0, 1/2]$ in the following.

(4) Consider the sequence of rationals $a_0, a_1, a_2, ...$ where

$$a_k = \frac{1}{3}\left(\frac{2}{3}\right)^k.$$

We see that $a_k \to 0$ as $k \to \infty$, hence there exists $k = k_i$ so that $a_{k+1} \leq y < a_k$.

(5) For this index i, define the monomial $\text{mon}_i(X) := X^{n-r-k_i} \in \mathbb{R}[X]$. If $E_i = \emptyset, \{0\}$, or $\{1\}$, then set $\text{mon}_i(X) = 0$. Recall that $D = D_j$ and consider the polynomial

$$f_j(X) = X^n - \sum_{i=1}^{2^r} \text{mon}_i(X). \tag{14}$$

Note that f_j is not necessarily a polynomial. If n is small, then the powers $n - r - k$ of X may turn out to be negative. To this end, we assume that n is large enough so that f_j is indeed a polynomial. Increasing the value of n only increases the multiplicity of the root at $X = 0$, this is irrelevant to us. For each $j = 1, 2, ..., w$, let θ_j be the greatest real root of $f_j(X)$.

(6) Let $\theta = \max_j \theta_j$.

After computing θ, we have $\Gamma_n = O(\theta^n)$.

4.3. **Verification of Algorithm.** After first reading the above algorithm, it is not clear how or even why it works. We now prove that this algorithm does indeed provide a bound for Γ_n.

Theorem 4.1. *Suppose θ is found using the above method. Then $\Gamma_N = O(\theta^N)$ for all positive integers N.*

Proof. We proceed by induction on N. Whenever N is small, Γ_N is a bounded integer. Hence $\Gamma_N < C\theta^N$ for large enough constant C. This establishes the base case; moreover it establishes the bound for any N less than some fixed positive integer. Before we begin the inductive step, we revisit our algorithm. We will see that the above algorithm is essentially the computation involved in the inductive step.

Using the aforementioned notation, we see that in step 1 we have

$$\Gamma_n = \max_{j \in \{1,...,w\}} \Gamma_n(D_j).$$

Now using the properties of the functions predup and preshf as in (12), in step 2 we obtain the rather crude estimate:

$$\Gamma_n(D_j) = \Gamma_n(D) \leq \sum_{i=1}^{2^r} \Gamma_{n-r}(E_i).$$

Consider the sequence $g_h(x)$ on some level h. There is very little we know a priori regarding the location of the maximum. But one very trivial statement we can make is that the maximum occurs on $[1/3, 2/3]$; in the process of *duplicate, shift, add* the first third of $g_h(x)$ is simply the first half of $g_{h-1}(x)$. By extending this inductively, if $x \in [0, (1/3)(2/3)^k]$, then $g_h(x)$ is can be realized as a term of the $(h-k)^{th}$ level. A similar argument applies for $x \in [1-(1/3)(2/3)^k, 1]$. We obtain that

$$g_h(x) \le \Gamma_{h-k}.$$

In step 3, we are determining the $y \in E_i$ that is closest to 1/2. In step 4, we compute the largest value of k so that every $x \in E_i$ satisfies either $x \le (1/3)(2/3)^k$ or $x \ge 1-(1/3)(2/3)^k$. This gives the estimate

$$\Gamma_{n-r}(E_i) \le \Gamma_{n-r-k_i}.$$

Putting these two inequalities together, we obtain

$$\Gamma_n(D) \le \sum_{i=1}^{2r} \Gamma_{n-r-k_i}.$$

We now tackle the inductive step in the proof. Assume that $\Gamma_n < C\theta^n$ for all $n < N$. We wish to show that $\Gamma_N < C\theta^N$. We start

$$\begin{aligned} \Gamma_N &= \max_j \Gamma_N(D_j) \\ &\le \max_j \sum_{i=1}^{2^r} \Gamma_{N-r-k_i} \\ &< \max_j \sum_{i=1}^{2^r} C\theta^{N-r-k_i} \\ &= C \max_j \left(\theta^N - f_j(\theta)\right) \end{aligned}$$

where the last equality comes from (14). Since $f_j(X)$ is a monic polynomial, $f_j(X) \to \infty$ as $X \to \infty$. In particular, $f_j(X) \ge 0$ for all $X \ge \theta_j$, with θ_j the largest real root of $f_j(X)$. Since $\theta \ge \theta_j$ for all j, it follows that $f_j(\theta) \ge 0$ for all j. Hence

$$C \max_j \left(\theta^N - f_j(\theta)\right) \le C \max_j \theta^N = C\theta^N.$$

□

4.4. **Remarks on algorithm.** There is much to say regarding this algorithm. Refer to Figure 7 in the Appendix for an example of the implementation of this algorithm. In step 1 of the outline, we only need to consider breaking up the interval $[1/3, 1/2]$ in (13) as a union of closed intervals because the maximum is guaranteed to occur on this interval. Also, steps 3 and 4 could be omitted from the algorithm—the resulting bound would not be as strong, but we would still have a bound. If this were the case, we would just set $\text{mon}_i(X) = X^{n-r}$ if $E_i \ne \emptyset$ and $\text{mon}_i(X) = 0$ if $E_i = \emptyset$.

As part of our project, we coded the above algorithm using Python. The code can easily be broken up and shipped out to many computing nodes; indeed this is exactly what is being done in step 1. We have found that the majority of the running time is spent in steps 3 and 4. Specifically, for a given interval D, a computer must repeat steps 3 and 4 at most 2^r times. In actuality, because some of the intervals E_i become empty under repeated applications of predup and preshf, the number of nonempty E_i will not be quite this high. However, the number of nonempty intervals still grows exponentially.

There are two essentially different parameters that we have control over when running this algorithm. We are able to choose exactly how we break up the interval $[1/3, 1/2]$ in (13) and we have able to choose the pullback number r defined in step 2. There is a computational trade off when setting these two values—it is hard to make both large at once. After testing various ways of breaking up $[1/3, 1/2]$ in (13) and various pullback numbers r, we find that a small increase in r will yield a better bound then a finer granulation in (13).

4.5. **Data.** After writing our code in Python, we ran many jobs using the high-throughput computing facilities of Condor at Clemson University. We submitted tens of thousands of jobs daily attempting to improve the bound on m_n. Before implementing this algorithm, the best known bound on m_n was the $O((\sqrt{2})^n)$ bound given in Section 3.3. We succeeded in significantly improving the bound. For a summary of our work, see Table 4 in the Appendix. Although this algorithm may not be able to actually achieve the bound $m_n = O((4/3)^n)$, it succeeds in bringing us much closer.

5. Statistical Properties of Bernoulli Convolutions

5.1. **The mean value of the Bernoulli sequence.** It is straightforward to see that the mean $\mu(B_n) = (4/3)^n$. To see this, let $S_n = \sum_i b_i$ where $b_i \in B_n$.

Proposition 5.1. ([2])

$$S_n = 4^n$$

Proof. We use induction on n. When $n = 0$, we have $S_0 = 1 = 4^0$. Assume that $S_{n-1} = 4^{n-1}$. Consider the n^{th} level Bernoulli sequence B_n. By definition, $B_n = \text{dup}_{n-1}(B_{n-1}) + \text{shf}_{n-1}(B_{n-1})$. However, the sum of elements in $\text{dup}_{n-1}(B_{n-1})$ is $2S_{n-1}$. A same statement is true with "shf" in place of "dup". Hence,

$$S_n = 2S_{n-1} + 2S_{n-1} = 4(4^{n-1}) = 4^n.$$

□

Because B_n has 3^n entries, as an immediate corollary we have $\mu(B_n) = (4/3)^n$.

5.2. **The variance and sum of squares of the Bernoulli sequence.** Using the formula for the variance of a set of data and the mean formula provided above, we were able to bound the sum of squares from above and below.

Proposition 5.2.

$$4^{2n} \geq \sum_{i=0}^{3^n-1} b_i^2 \geq \frac{4^{2n}}{3^n}$$

Proof. Let σ^2 denote the variance of B_n, and let $t = 3^n - 1$. Then

$$\begin{aligned}
\sigma^2 &= \frac{1}{3^n}\sum_{i=0}^{t}\left(b_i - \left(\frac{4}{3}\right)^n\right)^2 \\
&= \frac{1}{3^n}\sum_{i=0}^{t}\left(b_i^2 - 2\left(\frac{4}{3}\right)^n b_i + \left(\frac{4}{3}\right)^{2n}\right) \\
&= \left(\frac{1}{3^n}\sum_{i=0}^{t}\left(b_i^2 - 2\left(\frac{4}{3}\right)^n b_i\right)\right) + \left(\frac{4}{3}\right)^{2n} \\
&= \left(\frac{1}{3^n}\sum_{i=0}^{t} b_i^2\right) - 2\left(\frac{4}{3}\right)^{2n} + \left(\frac{4}{3}\right)^{2n} \\
&= \left(\frac{1}{3^n}\sum_{i=0}^{t} b_i^2\right) - \left(\frac{4}{3}\right)^{2n} && (15) \\
& && (16)
\end{aligned}$$

Since $\sigma^2 \geq 0$, we get $\sum_{i=0}^{t} b_i^2 \geq \frac{4^{2n}}{3^n}$.

Also it is easy to see that

$$4^{2n} = \left(\sum_{i=0}^{t} b_i\right)^2 \geq \sum_{i=0}^{t} b_i^2.$$

Thus

$$4^{2n} \geq \sum_{i=0}^{t} b_i^2 \geq \frac{4^{2n}}{3^n}.$$

□

This provides an upper and a lower bound for the sum of the squares. Using the upper bound on the sum of squares together with equation (15), the following upper bound on the variance can be achieved:

$$\left(\frac{4}{3}\right)^{2n}(3^n - 1) \geq \sigma^2.$$

In other words,

$$\frac{4^{2n}}{3^n}\left(1 - \frac{1}{3^n}\right) \geq \sigma^2.$$

We believe that the variance is in fact $O((4/3)^{2n})$ and that the sum of squares is $O((16/3)^n)$. See Figure 5 for a data plot supporting this claim.

5.3. **A nice property on the sum of squares.** Consider an arbitrary Bernoulli level $B_n = (b_0, b_1, ..., b_t)$ where $t = 3^n - 1$. Let $m = t/2$ so that b_m represents the midpoint of B_n. Because B_n is a palindrome, $b_i = b_{t-i}$. Using this notation, we can prove the following:

Proposition 5.3. *$\sum_i b_i^2$ is divisible by 4 if and only if b_m is divisible by 4.*

Proof. Assume b_m is divisible by 4. By proposition 5.1, we know that $\sum_{i=0}^{t} b_i = 4^n$. Let E denote the sum of the even terms b_i for $i = 0, 1, ..., m-1$. Let D denote the sum of the odd terms b_i for $i = 0, 1, ..., m-1$. Also, let E^2 denote the sum of the squares of the even terms on the same range, and D^2 denote the sums of the squares of the odd terms on the same range. Then,

$$\begin{aligned} 4^n &= \sum_{i=0}^{t} b_i \\ &= 2(b_0 + b_1 + ... + b_{m-1}) + b_m \\ &= 2E + 2D + b_m \qquad (17) \end{aligned}$$

Clearly $2E \equiv 0 \mod 4$, and by assumption $b_m \equiv 0 \mod 4$. This implies that $2D \equiv 0 \mod 4$. In turn, this implies that the number of odd terms is even. Similarly, we can write

$$\begin{aligned} \sum_{i=0}^{t} b_i^2 &= 2(b_0^2 + b_1^2 + ... + b_{m-1}^2) + b_m^2 \\ &= 2E^2 + 2D^2 + b_m^2 \end{aligned}$$

Clearly $2E^2 \equiv 0 \mod 4$. Since the number of odd terms is even, $2D^2 \equiv 0 \mod 4$. By assumption, $b_m^2 \equiv 0 \mod 4$. Thus, $\sum_i b_i^2 \equiv 0 \mod 4$. Thus $\sum_i b_i^2$ is divisible by 4.

Now assume $\sum_i b_i^2$ is divisible by 4. From above, we have $\sum_i b_i^2 = 2E^2 + 2D^2 + b_m^2 \equiv 0 \mod 4$. Clearly $2E^2 \equiv 0 \mod 4$. If the number of odd terms is even, then $2D^2 \equiv 0 \mod 4$ and $b_t^2 \equiv 0 \mod 4$ (we are done). Assume the number of odd terms is odd, and then D^2 is odd. Then $2D^2 = 2(2d+1)$ for some integer d. Then $2(2d+1) + b_m^2 \equiv 0 \mod 4$. This implies that $b_m^2 + 2 \equiv 0 \mod 4$, which implies that $b_m^2 \equiv 2 \mod 4$. This is a contradiction—every square number is congruent to 0 or 1 modulo 4. Thus the number of odd terms must be even. Then $2D^2 \equiv 0 \mod 4$, forcing $b_m^2 \equiv 0 \mod 4$, or, by 17, forcing b_m to be divisible by 4. □

5.4. **Using PIP to compute the sum of squares.** In this section we describe how a slight modification of the PIP algorithm can be used to compute the sum of squares for a given Bernoulli sequence. The way in which we use PIP comes from the following observation.

Consider the polynomial $p_n(x) = \sum_i b_i x^i$ as defined in Section 3.1. Let $t = 3^n - 1$. Since B_n is palindromic, we have $b_i = b_{t-i}$. Consider the coefficient s_t of x^t in $p_n(x)^2$. When expanding $p_n(x)^2$, we see that

$$s = \sum_{i=0}^{t} b_i b_{t-i} = \sum_{i=0}^{t} b_i^2.$$

Hence computing the sum of squares is equivalent to computing the middle coefficient in $p_n(x)^2$. Because $p_n(x)^2$ is of the form mentioned in Section 3.4, we can use PIP to compute this coefficient. The only part of the algorithm we change is the sequence S as defined in Section 3.4. This sequence is now

$$S = \{1, 1, 2, 2, ..., 2^{n-2}, 2^{n-2}, 2^{n-1}, 2^{n-1}, 2^{n-1}, 2^{n-1}, 2^{n-2}3, 2^{n-2}3, ..., 2^1 3^{n-2}, 2^1 3^{n-2}, 3^{n-1}, 3^{n-1}\}$$

which can be seen by looking at the product form of $p_n(x)$ in Section 3.2.

6. Conclusion

Studying Bernoulli convolutions through the lens of combinatorics sheds much new insight on the subject. For some of our work, this point of view is irrelevant (for example the proof of the bound improvement in Section 4). However, many of our algorithms would not have been discovered without combinatorial thinking (for example the PIP algorithm). The combinatorial point of view is a very simple way to think about Bernoulli convolutions (the *duplicate, shift, add* method could be explained to a small child), but a computer has trouble computing more than a handful of Bernoulli sequences. In particular, studying Bernoulli convolutions via combinatorics has led to the discovery and development of three elegant algorithms (DEM, PIP, and the bound improvement algorithm of Section 4).

In this paper, we have shown that the maximums satisfy $m_n = O((1.34000224903)^n)$, improving upon the previously best known bound of $O((\sqrt{2})^n)$. We conjecture $m_n = O((4/3)^n)$. Using our three algorithms, we have sufficient data to support this claim. These are questions we have been unable to answer: Can our bound improvement algorithm be pushed to further lower the bound given more computational power? Is it possible to conclusively prove our conjecture? Furthermore, is there an explicit formula to describe m_n for any arbitrary level? What else can be said regarding the global behavior of the Bernoulli sequence B_n? We have succeeded in providing partial answers of these questions through our algorithms and data.

7. Appendix

Figure 2. This shows some of the low level Bernoulli sequences generated using DSA. In the following plots, the horizontal axis gives the index and the vertical axis gives the entry corresponding to that index in a Bernoulli sequence. Below we show the Bernoulli sequence B_7.

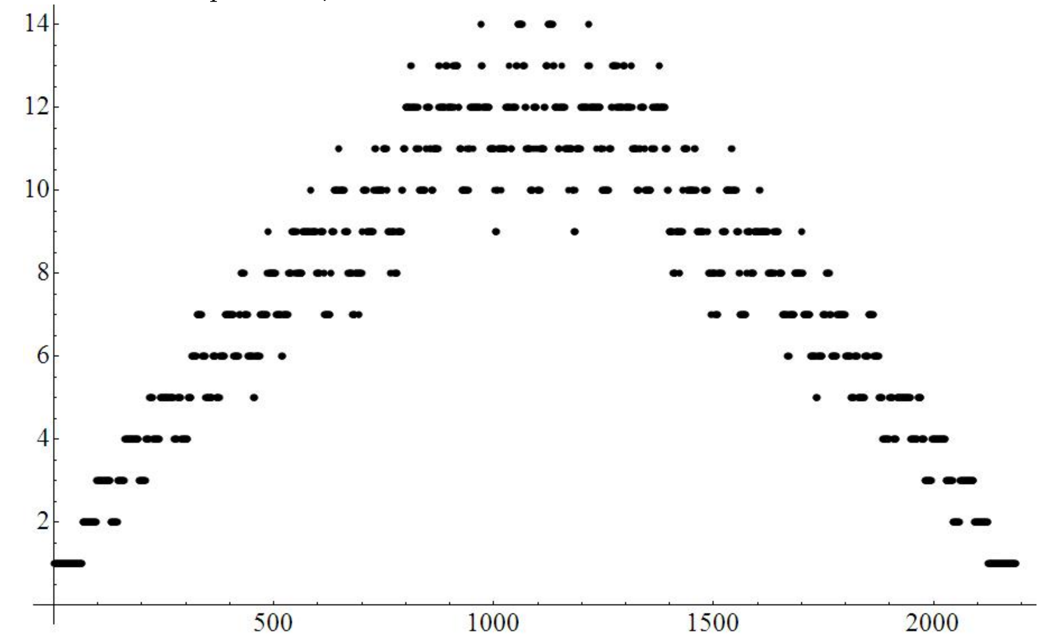

Figure 3. Below we show the Bernoulli sequence B_{11}.

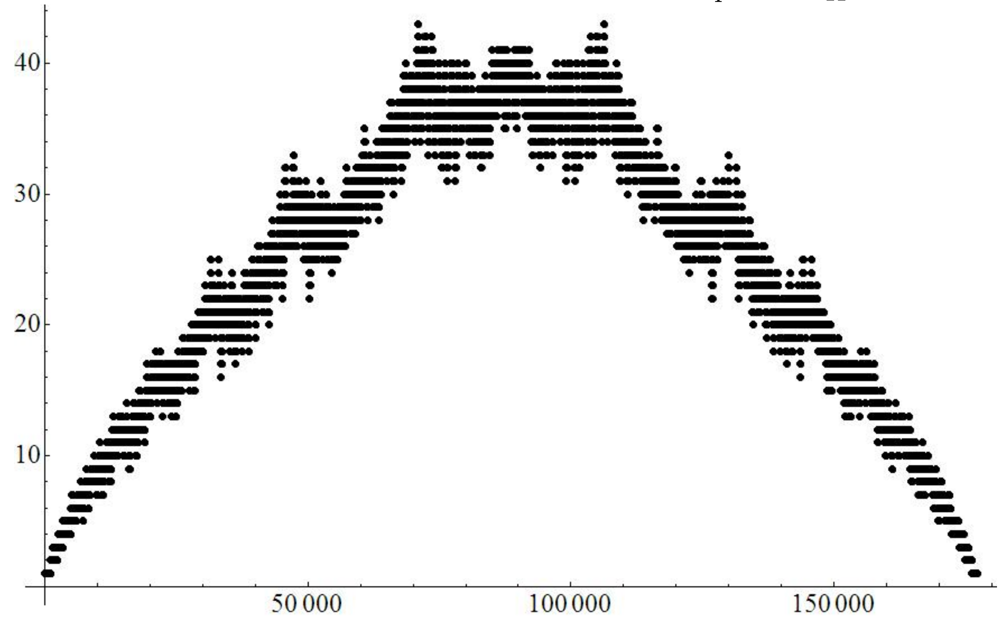

TABLE 1. The maximums m_n appear to be growing like $O((4/3)^n)$. Looking at the convergence of $m_n(3/4)^n$ provides evidence for this claim. Note that PIP is a powerful tool used to calculate isolated points on B_n; however, the sequences quickly become too large to analyze every individual point. We consider the indices on the first half of the middle third of B_n. We start by calculating the point one third of the way through and increment through by a constant. When the index increment is one, we calculate all of the values. This gives us a global picture of the behavior of B_n. Hence the maximum values m_n for $n > 26$ are not exact, but we do believe they are only slightly under the actual maximum. This accuracy of course dissolves for larger index increments.

Level n	Maximum Value m_n	$m_n(3/4)^n$	Computational Method	Index Increment
0	1	1	DSA	1
1	2	1.5	DSA	1
2	3	1.6875	DSA	1
3	4	1.6875	DSA	1
4	6	1.8984375	DSA	1
5	8	1.8984375	DSA	1
6	11	1.957763672	DSA	1
7	14	1.868774414	DSA	1
8	18	1.802032471	DSA	1
9	25	1.877117157	DSA	1
10	33	1.858345985	DSA	1
11	43	1.816110849	DSA	1
12	56	1.773875713	DSA	1
13	75	1.781794801	DSA	1
14	99	1.763976853	DSA	1
15	131	1.750613395	DSA	1
16	176	1.763976853	DSA	1
17	232	1.743931662	DSA	1
18	309	1.742052425	DSA	1
19	410	1.733595860	DSA	1
20	545	1.728310507	DSA	1
21	728	1.731481719	DEM	1
22	962	1.716022061	DEM	1
23	1283	1.716468012	DEM	1
24	1705	1.710782128	DEM	1
25	2266	1.705263476	DEM	1
26	3024	1.706768563	DEM	1
27	≈ 4025	–	PIP	20000
28	≈ 5357	–	PIP	200000
29	≈ 7170	–	PIP	400000
30	≈ 9531	–	PIP	2000000
31	≈ 12670	–	PIP	80000000
32	≈ 16914	–	PIP	200000000

Table 2. In this table, we consider various values for α appearing in the top row. Then we compute the k^{th} entry b_k of the Bernoulli sequence B_n for $k = \lceil \alpha(3^n - 1) \rceil$. Finally, we compute $b_k(3/4)^n$ for $n = 1, 2, ..., 40$. The data suggests this quantity converges as n grows large.

Level	0.36	0.38	0.4	0.42	0.44	0.46	0.48	0.5
1	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000
2	1.687500	1.687500	1.687500	1.687500	1.687500	1.125000	1.125000	1.125000
3	1.265625	1.687500	1.687500	1.687500	1.687500	1.265625	1.265625	1.687500
4	1.265625	1.582031	1.582031	1.265625	1.582031	1.265625	1.582031	1.898438
5	1.186523	1.186523	1.661133	1.423828	1.898438	1.423828	1.423828	1.898438
6	1.245850	1.423828	1.779785	1.423828	1.779785	1.423828	1.423828	1.779785
7	1.201355	1.334839	1.735291	1.601807	1.601807	1.334839	1.601807	1.601807
8	1.201355	1.301468	1.802032	1.701920	1.501694	1.401581	1.501694	1.802032
9	1.201355	1.351524	1.877117	1.802032	1.501694	1.351524	1.576778	1.802032
10	1.182584	1.407838	1.858346	1.689405	1.464151	1.520465	1.520465	1.576778
11	1.309289	1.435995	1.816111	1.562700	1.393759	1.520465	1.647170	1.689405
12	1.330407	1.393759	1.742199	1.615494	1.488789	1.520465	1.647170	1.710523
13	1.354164	1.401679	1.710523	1.591737	1.472950	1.520465	1.591737	1.520465
14	1.336346	1.425436	1.674887	1.567979	1.496708	1.478890	1.621433	1.639251
15	1.296256	1.469981	1.643706	1.576888	1.496708	1.496708	1.630342	1.630342
16	1.312960	1.433231	1.603615	1.573548	1.533457	1.493367	1.583570	1.583570
17	1.315466	1.450771	1.578559	1.563525	1.525940	1.510906	1.563525	1.623661
18	1.324862	1.460167	1.578559	1.544733	1.516544	1.482718	1.572921	1.668762
19	1.314996	1.429160	1.594063	1.530638	1.530638	1.496812	1.577149	1.640574
20	1.319224	1.436559	1.585606	1.525353	1.534867	1.474614	1.569750	1.617318
21	1.305747	1.446073	1.591156	1.534074	1.541209	1.479370	1.560236	1.584020
22	1.309314	1.455586	1.589372	1.539425	1.539425	1.478776	1.566182	1.641102
23	1.311098	1.446221	1.581345	1.547898	1.547898	1.485019	1.577331	1.650913
24	1.312436	1.453914	1.585358	1.551243	1.535189	1.483012	1.578334	1.609440
25	1.310178	1.455419	1.592382	1.563785	1.537446	1.478748	1.576579	1.610443
26	1.304910	1.448270	1.603482	1.563409	1.533495	1.479877	1.581470	1.609690
27	1.308438	1.448129	1.609832	1.559881	1.526864	1.478607	1.577237	1.601789
28	1.308967	1.444848	1.615017	1.556601	1.532790	1.485486	1.579142	1.612160
29	1.315555	1.448896	1.616049	1.554379	1.529853	1.476517	1.576046	1.618192
30	1.308114	1.451872	1.616346	1.554914	1.527055	1.471516	1.577594	1.610810
31	1.304274	1.449729	1.617150	1.550718	1.531565	1.470757	1.577639	1.608846
32	1.304073	1.447017	1.616983	1.552593	1.536219	1.473034	1.575094	1.610252
33	1.307213	1.447419	1.616631	1.551990	1.531498	1.474767	1.572030	1.613467
34	1.308757	1.447589	1.615463	1.549579	1.532515	1.474315	1.575458	1.615463
35	1.308474	1.449637	1.616141	1.549989	1.529139	1.473750	1.574780	1.617582
36	1.308125	1.449245	1.617667	1.550127	1.530548	1.474640	1.574123	1.616396
37	1.307886	1.450556	1.617897	1.552057	1.530150	1.474251	1.574417	1.617445
38	1.307892	1.450008	1.619012	1.551682	1.529727	1.473464	1.574763	1.617743
39	1.307544	1.450535	1.619714	1.550310	1.529070	1.473370	1.575921	1.617233
40	1.307356	1.450019	1.620015	1.550705	1.529959	1.475150	1.575666	1.616817

Table 3. In this table, we consider various values for α appearing in the top row. Then we compute the k^{th} entry b_k of the Bernoulli sequence B_n for $k = \lceil \alpha(3^n - 1) \rceil$. Finally, we compute $b_k(3/4)^n$ for $n = 1, 2, ..., 40$. The data suggests this quantity converges as n grows large.

Level	0.49	0.499	0.4999	0.49999	0.499999	0.4999999	0.49999999	0.5
1	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000	1.500000
2	1.125000	1.125000	1.125000	1.125000	1.125000	1.125000	1.125000	1.125000
3	1.687500	1.687500	1.687500	1.687500	1.687500	1.687500	1.687500	1.687500
4	1.898438	1.898438	1.898438	1.898438	1.898438	1.898438	1.898438	1.898438
5	1.661133	1.898438	1.898438	1.898438	1.898438	1.898438	1.898438	1.898438
6	1.601807	1.779785	1.779785	1.779785	1.779785	1.779785	1.779785	1.779785
7	1.601807	1.601807	1.601807	1.601807	1.601807	1.601807	1.601807	1.601807
8	1.601807	1.802032	1.802032	1.802032	1.802032	1.802032	1.802032	1.802032
9	1.501694	1.802032	1.802032	1.802032	1.802032	1.802032	1.802032	1.802032
10	1.633092	1.633092	1.576778	1.576778	1.576778	1.576778	1.576778	1.576778
11	1.604935	1.647170	1.689405	1.689405	1.689405	1.689405	1.689405	1.689405
12	1.552141	1.615494	1.710523	1.710523	1.710523	1.710523	1.710523	1.710523
13	1.544222	1.663008	1.544222	1.520465	1.520465	1.520465	1.520465	1.520465
14	1.621433	1.621433	1.621433	1.603615	1.621433	1.639251	1.639251	1.639251
15	1.603615	1.603615	1.643706	1.603615	1.616979	1.630342	1.630342	1.630342
16	1.623661	1.593593	1.573548	1.563525	1.573548	1.583570	1.583570	1.583570
17	1.623661	1.623661	1.578559	1.601110	1.616144	1.623661	1.623661	1.623661
18	1.623661	1.601110	1.589834	1.651849	1.663124	1.668762	1.668762	1.668762
19	1.632117	1.581378	1.577149	1.623661	1.636345	1.640574	1.640574	1.640574
20	1.630003	1.601462	1.572921	1.598291	1.614147	1.617318	1.617318	1.617318
21	1.643481	1.600669	1.600669	1.562615	1.576885	1.584020	1.584020	1.584020
22	1.632183	1.616129	1.619696	1.601858	1.630399	1.641102	1.641102	1.641102
23	1.637535	1.614791	1.610777	1.606764	1.640210	1.650913	1.650913	1.650913
24	1.630511	1.602416	1.602416	1.596396	1.601412	1.611446	1.609440	1.609440
25	1.621731	1.603670	1.602165	1.596897	1.598402	1.611196	1.610443	1.610443
26	1.631702	1.602918	1.601224	1.609690	1.606304	1.609690	1.609690	1.609690
27	1.630574	1.608138	1.600942	1.611525	1.604752	1.604329	1.602212	1.601789
28	1.622002	1.607080	1.606763	1.614700	1.611842	1.611525	1.612477	1.612160
29	1.626764	1.611763	1.609382	1.614144	1.616049	1.618668	1.618906	1.618192
30	1.630812	1.615096	1.605632	1.612596	1.612775	1.612596	1.611346	1.610810
31	1.628535	1.610319	1.602551	1.610989	1.613132	1.614204	1.609784	1.608846
32	1.632352	1.613567	1.604828	1.612061	1.615476	1.614170	1.611156	1.610252
33	1.627028	1.615426	1.605858	1.614522	1.613392	1.616932	1.615124	1.613467
34	1.626538	1.612469	1.602919	1.613994	1.613655	1.615407	1.615576	1.615463
35	1.625719	1.613344	1.605165	1.612539	1.612582	1.613938	1.616565	1.617582
36	1.624850	1.614934	1.607941	1.612963	1.614457	1.613472	1.616968	1.616396
37	1.624334	1.612820	1.606050	1.613321	1.616062	1.614751	1.617373	1.617445
38	1.625716	1.613327	1.606497	1.612701	1.615669	1.614900	1.617868	1.617743
39	1.624809	1.613479	1.606439	1.612902	1.615745	1.615584	1.617032	1.617233
40	1.625315	1.613187	1.607183	1.613257	1.614746	1.614786	1.618094	1.616817

FIGURE 4. The above flowchart illustrates this algorithm. It shows the computation of $\mathrm{N}_S(k)$ for $k = 12$ and $S = \{1, 2, 4, 4, 6, 9\}$. As the diagram suggests, $\mathrm{N}_S(k) = 3$, corresponding to the fact that there are three boxes containing the word answer++.

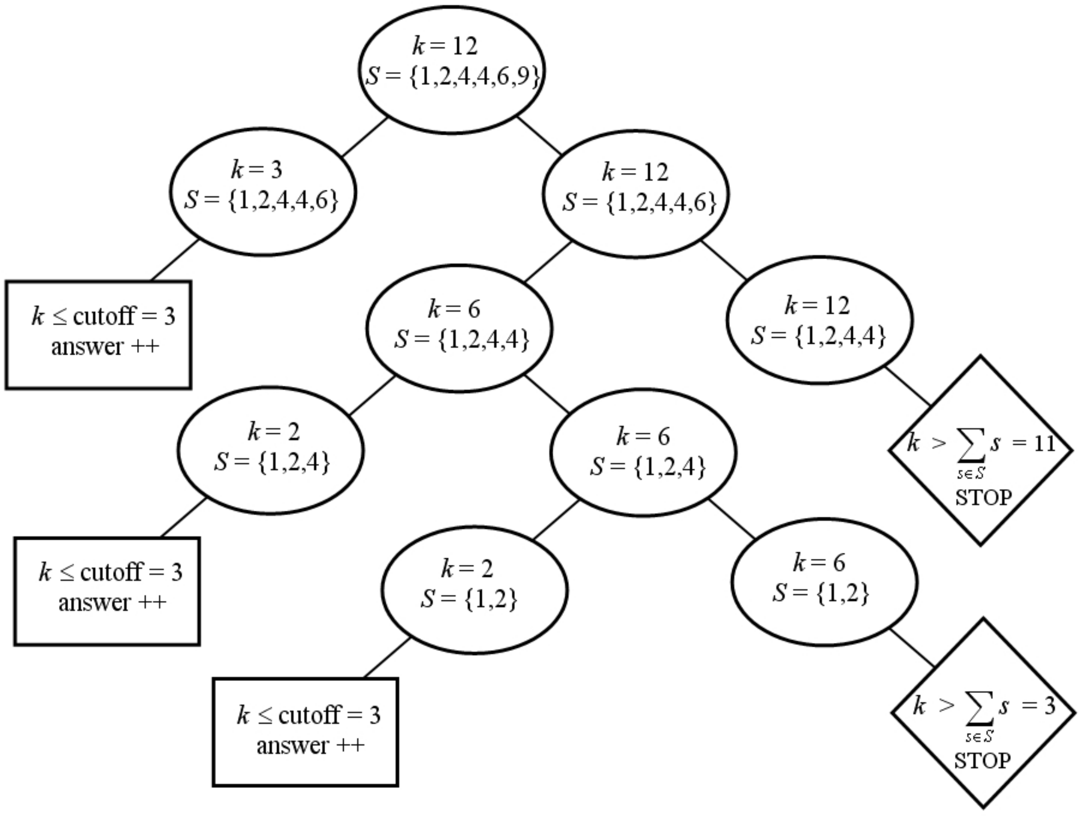
$k = 12$ $S = \{1,2,4,4,6,9\}$
$k = 3$ $S = \{1,2,4,4,6\}$
$k = 12$ $S = \{1,2,4,4,6\}$
$k \le$ cutoff $= 3$ answer ++
$k = 6$ $S = \{1,2,4,4\}$
$k = 12$ $S = \{1,2,4,4\}$
$k = 2$ $S = \{1,2,4\}$
$k = 6$ $S = \{1,2,4\}$
$k > \sum_{s \in S} s = 11$ STOP
$k \le$ cutoff $= 3$ answer ++
$k = 2$ $S = \{1,2\}$
$k = 6$ $S = \{1,2\}$
$k \le$ cutoff $= 3$ answer ++
$k > \sum_{s \in S} s = 3$ STOP

FIGURE 5. This plot shows the quotient of the sum of the squares by $(4/3)^n$ for n up to 25.

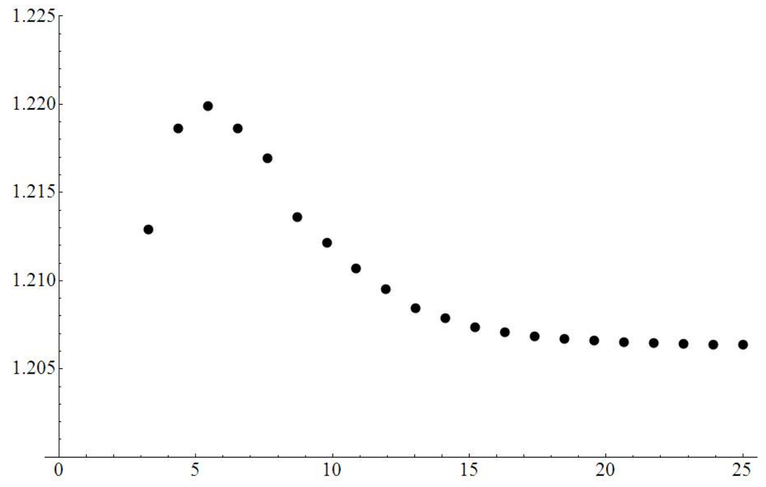

FIGURE 6. This graph shows the quotient of the maximum m_n by $(4/3)^n$ for n up to 27. Are these points approaching a limit?

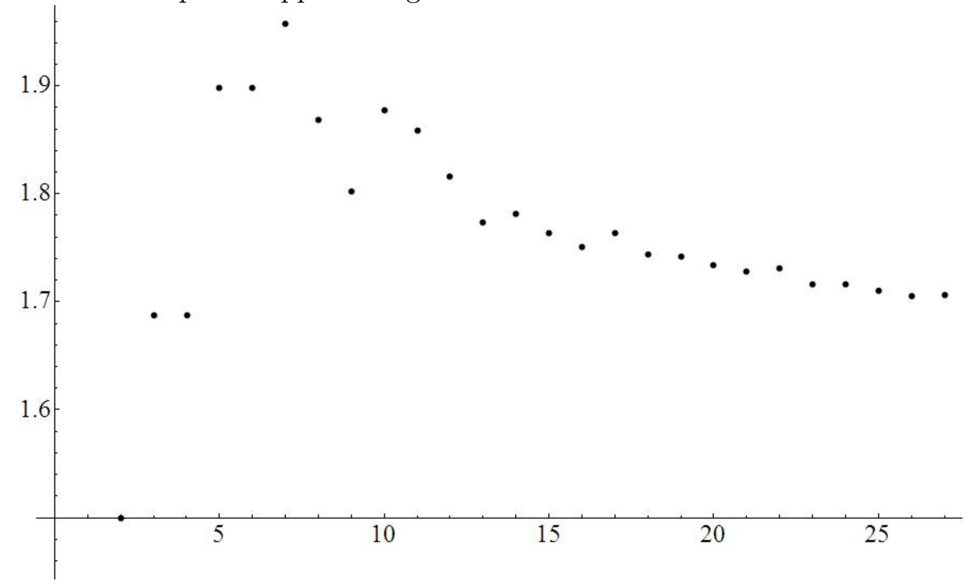

FIGURE 7. This diagram provides a good visualization of the logic behind the better bound algorithm. Here we the pullback number $r = 2$ and each E_i has length 1/27 as indicated by the vertical lines. The numbers h on each interval E_i satisfy $\Gamma_{n-2}(E_i) \leq \Gamma_{n-h}$ as in steps 3 and 4 in the PIP algorithm.

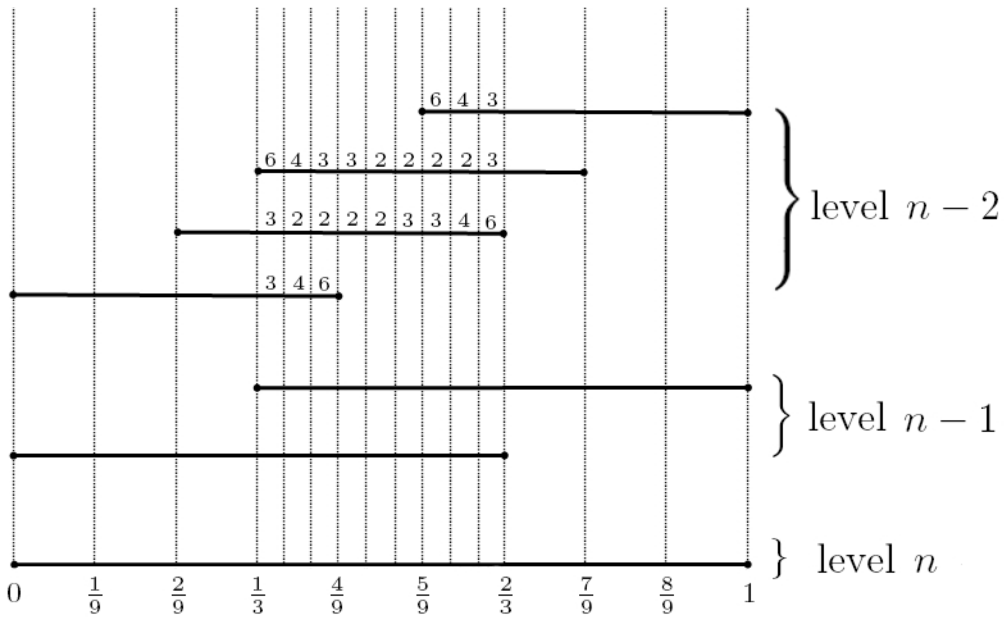

Table 4. These are the bounds coming from the algorithm in Section 4.2. We choose various pullback numbers r as seen in the table below. Our break up the interval $[1/3, 1/2]$ so that each pullback interval E_i has width according to the values shown below. The last entry in this table is the best bound we were able to achieve with the computational power available to us. The values beneath the line at level 22 mark the bounds we achieved after we parallelized our algorithm among thousands of nodes.

Pullback number r	Width of interval E_i	Bound θ
1	1/360	1.57795430509
2	1/360	1.41421356237
3	1/360	1.38027756910
4	1/360	1.38581622915
5	1/360	1.36533552385
6	1/360	1.36558440642
7	1/360	1.35579220586
8	1/360	1.35605185929
9	1/360	1.35327499283
10	1/360	1.34930156389
11	1/360	1.34798572414
12	1/360	1.34740387405
13	1/360	1.34614197103
14	1/360	1.34530126253
15	1/210	1.34428051641
16	1/150	1.34411942747
17	1/150	1.34308122018
18	1/150	1.34308363659
19	1/150	1.34237196822
20	1/150	1.34203310608
21	1/150	1.34147328569
22	1/150	1.34113965590
23	1/1200	1.34034466363
23	1/1800	1.34030034266
23	1/2001	1.34028774655
23	1/2100	1.34028422268
24	1/150	1.34045678181
24	1/600	1.34010969510
24	1/1050	1.34000224903

8. Acknowledgments

Special thanks to:
Jobby Jacob for numerous suggestions and patience,
Neil Calkin for motivation on the problem and guidance,
Kevin James for assistance and support,
Our fellow REU students for friendship and comic relief,
Dan Warner for suggesting the use of Python,
Matt Saltzmann for computational advice,
Deniz Savas for help with parallel computing,
The National Science Foundation for their generous grant (DSM-0552799) enabling us to do this research.

References

[1] David H. Bailey, Jonathan M. Borwein, Neil J. Calkin, Roland Girgensohn, D. Russell Luke, and Victor H. Moll. *Experimental mathematics in action.* A K Peters Ltd., Wellesley, MA, 2007.
[2] Neil Calkin. Counting kings, collectings coupons, and other applications of linear algebra to combinatorics. *Clemson University REU Colloquium Lecture*, 2008.
[3] Paul Erdős. On a family of symmetric Bernoulli convolutions. *Amer. J. Math.*, 61:974–976, 1939.
[4] Børge Jessen and Aurel Wintner. Distribution functions and the Riemann zeta function. *Trans. Amer. Math. Soc.*, 38(1):48–88, 1935.
[5] Richard Kershner and Aurel Wintner. On Symmetric Bernoulli Convolutions. *Amer. J. Math.*, 57(3):541–548, 1935.
[6] Yuval Peres, Wilhelm Schlag, and Boris Solomyak. Sixty years of Bernoulli convolutions. In *Fractal geometry and stochastics, II (Greifswald/Koserow, 1998)*, volume 46 of *Progr. Probab.*, pages 39–65. Birkhäuser, Basel, 2000.
[7] Boris Solomyak. On the random series $\sum \pm\lambda^n$ (an Erdős problem). *Ann. of Math. (2)*, 142(3):611–625, 1995.

Julia Davis, Department of Mathematics, Grove City College, Grove City, PA 16127
E-mail address: `davisjl1@gcc.edu`

Michelle Delcourt, School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332
E-mail address: `mdelcourt3@gatech.edu`

Zebediah Engberg, School of Natural Science, Hampshire College, Amherst, MA
E-mail address: `zee05@hampshire.edu`