Taking the "Convoluted" out of Bernoulli Convolutions

A Combinatorial Approach

Julia Davis, Michelle Delcourt, and Zebediah Engberg

Mini-Conference on Discrete Mathematics and Combinatorics
Clemson University

October 3, 2008

Acknowledgments

Acknowledgments

- Dan Warner, for suggesting the use of Python

Acknowledgments

- Dan Warner, for suggesting the use of Python
- Matt Saltzmann, for computational advice

Acknowledgments

- Dan Warner, for suggesting the use of Python
- Matt Saltzmann, for computational advice
- Deniz Savas, for help with parallel computing

Acknowledgments

- Dan Warner, for suggesting the use of Python
- Matt Saltzmann, for computational advice
- Deniz Savas, for help with parallel computing
- Janine Janoski, for help with Condor

Acknowledgments

- Dan Warner, for suggesting the use of Python
- Matt Saltzmann, for computational advice
- Deniz Savas, for help with parallel computing
- Janine Janoski, for help with Condor
- Our fellow REU students, for friendship and comic relief

Acknowledgments

- Dan Warner, for suggesting the use of Python
- Matt Saltzmann, for computational advice
- Deniz Savas, for help with parallel computing
- Janine Janoski, for help with Condor
- Our fellow REU students, for friendship and comic relief
- Clemson University, for allowing access to Condor and Palmetto

Acknowledgments

- Dan Warner, for suggesting the use of Python
- Matt Saltzmann, for computational advice
- Deniz Savas, for help with parallel computing
- Janine Janoski, for help with Condor
- Our fellow REU students, for friendship and comic relief
- Clemson University, for allowing access to Condor and Palmetto
- The NSF for their generous grant (DSM-0552799) enabling us to do this research.

Acknowledgments

And especially,

Acknowledgments

And especially,

- Jobby Jacob, for numerous suggestions and patience

Acknowledgments

And especially,

- Jobby Jacob, for numerous suggestions and patience
- Neil Calkin, for motivation on the problem and guidance

Acknowledgments

And especially,

- Jobby Jacob, for numerous suggestions and patience
- Neil Calkin, for motivation on the problem and guidance
- Kevin James, for assistance and support

Acknowledgments

And especially,

- Jobby Jacob, for numerous suggestions and patience
- Neil Calkin, for motivation on the problem and guidance
- Kevin James, for assistance and support

Motivation

Bernoulli convoluted

Bernoulli convoluted

A Bernoulli Convolution is the convolution

$$\mu_q(X) = b(X) * b(X/q) * b(X/q^2) * \ldots$$

Bernoulli convoluted

A Bernoulli Convolution is the convolution

$$\mu_q(X) = b(X) * b(X/q) * b(X/q^2) * \ldots$$

where b is the discrete Bernoulli measure concentrated at 1 and -1 each with weight $1/2$.

Bernoulli convoluted

A Bernoulli Convolution is the convolution

$$\mu_q(X) = b(X) * b(X/q) * b(X/q^2) * \ldots$$

where b is the discrete Bernoulli measure concentrated at 1 and -1 each with weight $1/2$.

Consider the distribution function we define as $F_q(t) = \mu_q([-\infty, t])$.

Bernoulli convoluted

A Bernoulli Convolution is the convolution

$$\mu_q(X) = b(X) * b(X/q) * b(X/q^2) * \ldots$$

where b is the discrete Bernoulli measure concentrated at 1 and -1 each with weight $1/2$.

Consider the distribution function we define as $F_q(t) = \mu_q([-\infty, t])$.

This is very confusing—we do not like it.

Functional equation

Functional equation

Instead consider the functional equation

Functional equation

Instead consider the functional equation

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

for t on the interval $I_q := [-1/(1-q), 1/(1-q)]$.

Functional equation

Instead consider the functional equation

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

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.

Absolutely continuous v. singular

Absolutely continuous v. singular

The major question regarding the solution of the previous equation is that of determining the values of q that make $F_q(t)$ absolutely continuous and the values that make $F_q(t)$ singular.

Absolutely continuous v. singular

The major question regarding the solution of the previous equation 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 $0 < q < 1/2$, it is known that $F_q(t)$ is always singular. For these values of q, the solution $F_q(t)$ is an example of a *Cantor function*, a function that is constant almost everywhere.

Absolutely continuous v. singular

The major question regarding the solution of the previous equation 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 $0 < q < 1/2$, it is known that $F_q(t)$ is always singular. For these values of q, the solution $F_q(t)$ is an example of a *Cantor function*, a function that is constant almost everywhere.

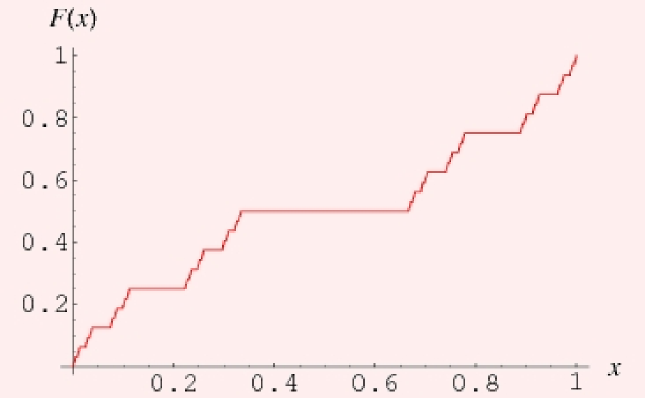
It is also easy to see that for $q = 1/2$, the solution $F_q(t)$ is absolutely continuous.

Devil's staircase

From www.mathworld.com, a plot of the Devil's staircase:

Devil's staircase

From www.mathworld.com, a plot of the Devil's staircase:



Pisot numbers

Pisot numbers

The case when $q > 1/2$ is much harder and more interesting.

Pisot numbers

The case when $q > 1/2$ is much harder and more interesting.

In 1939, Erdős showed that if q is of the form $q = 1/\theta$ with θ a *Pisot number*, then $F_q(t)$ is again singular.

Pisot numbers

The case when $q > 1/2$ is much harder and more interesting.

In 1939, Erdős showed that if q is of the form $q = 1/\theta$ with θ a *Pisot number*, then $F_q(t)$ is again singular.

A *Pisot number* is an algebraic integer greater than 1 in absolute value, whose conjugates are all less than 1 in absolute value.

Pisot numbers

The case when $q > 1/2$ is much harder and more interesting.

In 1939, Erdős showed that if q is of the form $q = 1/\theta$ with θ a *Pisot number*, then $F_q(t)$ is again singular.

A *Pisot number* is an algebraic integer greater than 1 in absolute value, whose conjugates are all less than 1 in absolute value.

The classic example is the golden ratio $\tau = (1 + \sqrt{5})/2$. Like all Pisot numbers, τ has the property that large powers of τ approach rational integers.

No actual example is known

No actual example is known

There is little else that is known for other values of $q > 1/2$.

No actual example is known

There is little else that is known for other values of $q > 1/2$.

One interesting result is that almost every $q > 1/2$ yields a solution $F_q(t)$ that is absolutely continuous.

No actual example is known

There is little else that is known for other values of $q > 1/2$.

One interesting result 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.

$q = 2/3$

$q = 2/3$

Specifically, the obvious case when $q = 2/3$ remains a

$q = 2/3$

Specifically, the obvious case when $q = 2/3$ remains a

mystery.

Differentiate

Differentiate

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 $f_q(t)$:

Differentiate

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 $f_q(t)$:

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

Differentiate

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 $f_q(t)$:

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

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

A new functional equation

A new functional equation

In *Experimental Mathematics in Action*, Girgensohn asks the question of computing $f_q(t)$ for various values of q.

A new functional equation

In *Experimental Mathematics in Action*, 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

A new functional equation

In *Experimental Mathematics in Action*, 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

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

A new functional equation

In *Experimental Mathematics in Action*, 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

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

to gain a sequence of functions $f_0, f_1, f_2, \ldots$. If this sequence converges, then it converges to the solution of the previous functional equation.

$q = 2/3$

$q = 2/3$

Neil Calkin 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.

$q = 2/3$

Neil Calkin 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

$q = 2/3$

Neil Calkin 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

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

The transform

The transform

Intuitively, this transform 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 transform

Intuitively, this transform 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

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

The transform

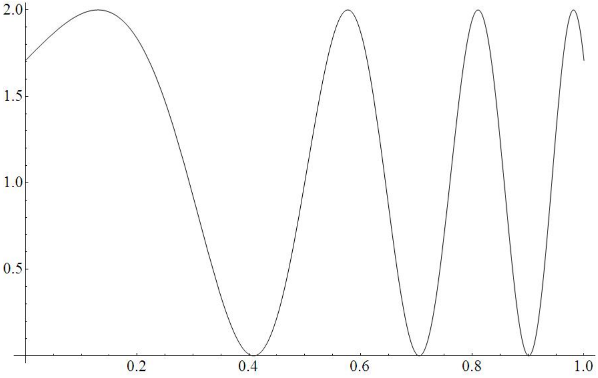
Intuitively, this transform takes two scaled copies of $f(x)$: one on the interval $[0, 2/3]$ and the other on $[1/3, 1]$, and adds them.

The scaling factor of $3/4$ gives us that

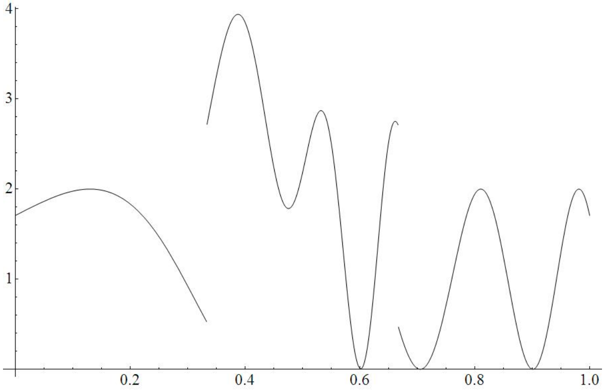
$$\int_0^1 f(x)dx = \int_0^1 Tf(x)dx.$$

In this setting, the question to be answered is: starting with the function $f_0(x) = 1$, does the iteration determined by this transform converge to a bounded function?

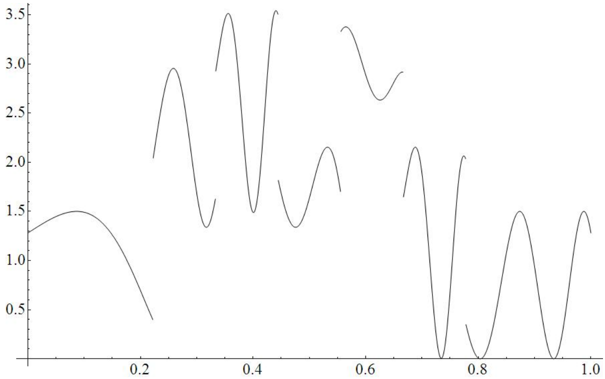
f_0



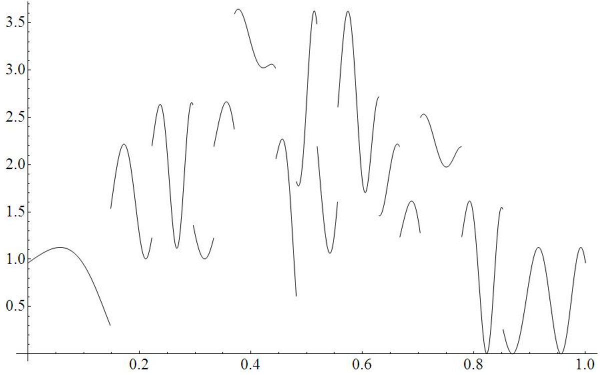
f_1



f_2



f_3



Recursive Algorithms

Recursive Algorithms

Duplicate, Shift, Add

Recursive Algorithms

Duplicate, Shift, Add

Looking at this problem through the lens of combinatorics

Duplicate, Shift, Add

Duplicate, Shift, Add

Instead of viewing T as a transform on $[0, 1]$, we consider a combinatorial analogue.

Duplicate, Shift, Add

Instead of viewing T as a transform on $[0, 1]$, we consider a combinatorial analogue.

Consider the two maps $\mathrm{dup}_n, \mathrm{shf}_n : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ defined by

Duplicate, Shift, Add

Instead of viewing T as a transform on $[0, 1]$, we consider a combinatorial analogue.

Consider the two maps $\mathrm{dup}_n, \mathrm{shf}_n : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ defined by

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

Duplicate, Shift, Add

Instead of viewing T as a transform on $[0,1]$, we consider a combinatorial analogue.

Consider the two maps $\mathrm{dup}_n, \mathrm{shf}_n : \mathbb{R}^n \longrightarrow \mathbb{R}^{3n}$ defined by

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

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

Duplicate, Shift, Add

The names “dup” and “shf” reference the duplication and shifting of the coordinates.

Duplicate, Shift, Add

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

Duplicate, Shift, Add

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

We are primarily interested in the rate at which the maximum $m_n := \max B_n$ is growing with n. We start with the sequence $B_0 = (1)$.

Duplicate, Shift, Add

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

We are primarily interested in the rate at which the maximum $m_n := \max B_n$ is growing with n. We start with the sequence $B_0 = (1)$.

Duplicate, Shift, Add

Duplicate, Shift, Add

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

Duplicate, Shift, Add

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

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

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

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

Duplicate, Shift, Add

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

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

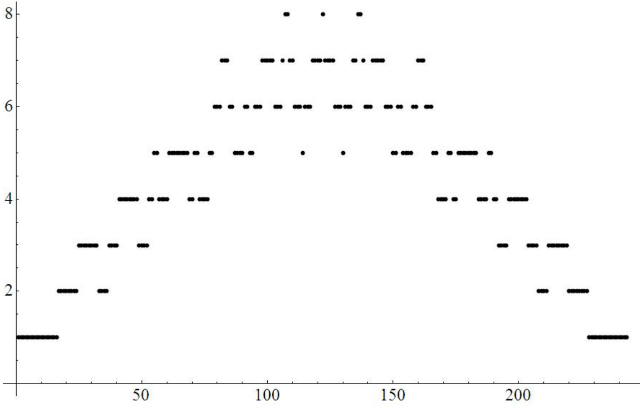
$$\begin{array}{ccccccccccccc} 1 & 2 & 1 & \longrightarrow & 1 & 1 & 2 & 2 & 1 & 1 & & & \\ & & & & & & & 1 & 1 & 2 & 2 & 1 & 1 \\ \hline & & & & 1 & 1 & 2 & 3 & 2 & 3 & 2 & 1 & 1 \end{array}$$

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

The first few maximums m_n are $1, 2, 3, 4, 6, 8, 11, \ldots$.

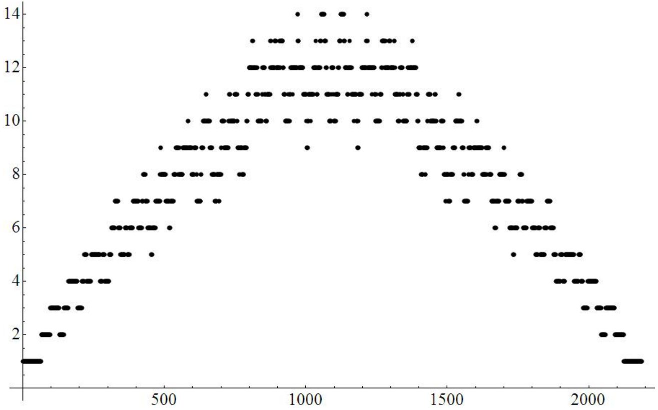
Level $n = 5$

The plot shows the index on the horizontal axis and the B_5 entry on the vertical axis.



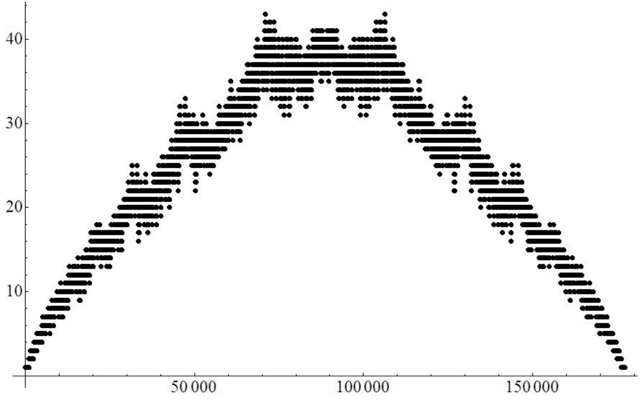
Level $n = 7$

The plot shows the index on the horizontal axis and the B_7 entry on the vertical axis.



Level $n = 11$

The plot shows the index on the horizontal axis and the B_{11} entry on the vertical axis.



A Useful Property

A Useful Property

It is straightforward to see that the mean $\mu(B_n) = (4/3)^n$.

A Useful Property

It is straightforward to see that the mean $\mu(B_n) = (4/3)^n$.

The reason for this is because under the DSA process, the length of a Bernoulli sequence grows by a factor of three while the sum of the terms increases by a factor of four.

Does m_n also grow like $(4/3)^n$?

Double, Enlarge, Merge

Recursive Algorithms Continued

Double, Enlarge, Merge

Double, Enlarge, Merge

Double, Enlarge, Merge

We now consider an alternate approach that addresses some of the issues arising with the DSA algorithm.

Double, Enlarge, Merge

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 $2(2^n - 1)$.

Double, Enlarge, Merge

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.

Double, Enlarge, Merge

Double, Enlarge, Merge

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.

Double, Enlarge, Merge

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.

Double, Enlarge, Merge

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.

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

Double, Enlarge, Merge

Double, Enlarge, Merge

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. Explicitly

Double, Enlarge, Merge

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. Explicitly

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

Double, Enlarge, Merge

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. Explicitly

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

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

Double, Enlarge, Merge

Double, Enlarge, Merge

We now describe how to translate the process of DSA to this new representation.

Double, Enlarge, Merge

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.

Double, Enlarge, Merge

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 doubled list is modified by 3^n. If the element is positive, we add 3^n, for negative elements we subtract 3^n.

Double, Enlarge, Merge

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 doubled 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 two 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.

Data

Using Palmetto...

Data

Using Palmetto...

Level n	Maximum Value	$m_n(3/4)^n$
0	1	1
1	2	1.5
2	3	1.6875
3	4	1.6875
4	6	1.8984375
5	8	1.8984375
6	11	1.957763672
7	14	1.868774414
8	18	1.802032471
9	25	1.877117157

Data

Level n	Maximum Value	$m_n(3/4)^n$
10	33	1.858345985
11	43	1.816110849
12	56	1.773875713
13	75	1.781794801
14	99	1.763976853
15	131	1.750613395
16	176	1.763976853
17	232	1.743931662
18	309	1.742052425

Data

Level n	Maximum Value	$m_n(3/4)^n$
19	410	1.733595860
20	545	1.728310507
21	728	1.731481719
22	962	1.716022061
23	1283	1.716468012
24	1705	1.710782128
25	2266	1.705263476
26	3024	1.706768563

Translating DSA as a Polynomial Recursion

Polynomial Approach

Translating DSA as a Polynomial Recursion

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.

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 $t = 3^n - 1$.

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 $p_n(x) := b_0 + b_1 x + ... + b_t x^t$.

Translating DSA as a Polynomial Recursion

Translating DSA as a Polynomial Recursion

We create a dictionary that translates the process of DSA into recursive relations among the polynomials.

Translating DSA as a Polynomial Recursion

We create a dictionary that translates the process of DSA into recursive relations among the polynomials.

We see that the duplication $b_0, b_0, b_1, b_1, ..., b_t, b_t$ corresponds to the polynomial $(1 + x)p_n(x^2)$.

Translating DSA as a Polynomial Recursion

We create a dictionary that translates the process of DSA into recursive relations among the polynomials.

We see that the duplication $b_0, b_0, b_1, b_1, ..., b_t, b_t$ 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}.

Translating DSA as a Polynomial Recursion

We create a dictionary that translates the process of DSA into recursive relations among the polynomials.

We see that the duplication $b_0, b_0, b_1, b_1, \ldots, b_t, b_t$ 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.

Translating DSA as a Polynomial Recursion

We create a dictionary that translates the process of DSA into recursive relations among the polynomials.

We see that the duplication $b_0, b_0, b_1, b_1, ..., b_t, b_t$ 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

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

Translating DSA as a Polynomial Recursion

This formula allows us to explicitly solve for $p_n(x)$.

Translating DSA as a Polynomial Recursion

This formula allows us to explicitly solve for $p_n(x)$.

Theorem

(Calkin-Lennard) The polynomials $p_n(x)$ satisfy

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

Proof

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

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

Proof

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

Proof

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

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

Proof

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

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

Proof

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

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

Proof

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

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

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.

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

(Calkin-Lennard) The maximum values satisfy $m_n = O((\sqrt{2})^n)$.

Proof

To start, define polynomials q_n, r_n, s_n by

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

Proof

To start, define polynomials q_n, r_n, s_n by

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

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

Proof

We see that

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

Proof

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 $q_n(x) r_n(x)$.

Proof

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.

Proof

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 $s_n(x)$.

Proof

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.

Proof

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 $O(2^{n/2}) = O((\sqrt{2})^n)$.

PIP

Polynomial Isolated Point

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.

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.

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.

An Algorithmic Implementation: PIP

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

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

An Algorithmic Implementation: PIP

Our algorithm is based on the following idea. Suppose $S = \{a_1, ..., a_n\}$ is a sequence of 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.

An Algorithmic Implementation: PIP

Our algorithm is based on the following idea. Suppose $S = \{a_1, ..., a_n\}$ is a sequence of 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 $(1 + x^a)$.

An Algorithmic Implementation: PIP

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

An Algorithmic Implementation: PIP

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

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

An Algorithmic Implementation: PIP

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:

An Algorithmic Implementation: PIP

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

An Algorithmic Implementation: PIP

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

$$\mathrm{N}_S(k) = \mathrm{N}_{S\setminus\{a_i\}}(k) + \mathrm{N}_{S\setminus\{a_i\}}(k - a_i).$$

An Algorithmic Implementation: PIP

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

$$\mathrm{N}_S(k) = \mathrm{N}_{S\setminus\{a_i\}}(k) + \mathrm{N}_{S\setminus\{a_i\}}(k - a_i).$$

- If $k > \sum_{s \in S} s$, then $\mathrm{N}_S(k) = 0$.

An Algorithmic Implementation: PIP

- If $k < 0$, then $\mathrm{N}_S(k) = 0$.

An Algorithmic Implementation: PIP

- If $k < 0$, then $\mathrm{N}_S(k) = 0$.
- 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.

Flowchart

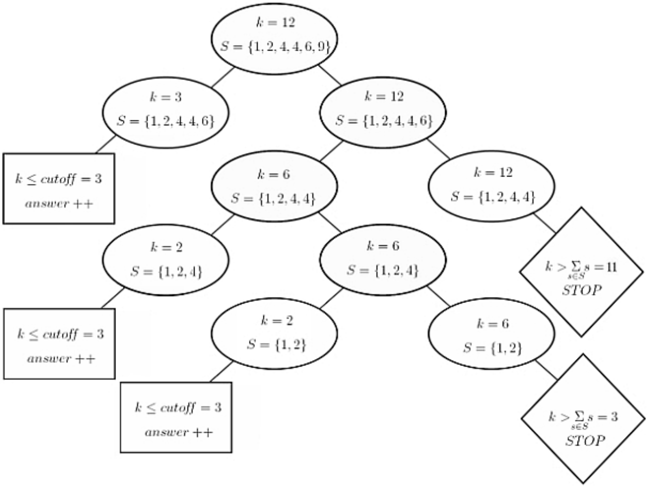
The following flowchart shows an explicit example of our implementation of the PIP algorithm in use.

Flowchart

The following flowchart shows an explicit example of our implementation of the PIP algorithm in use.

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* $++$

Flowchart


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

Data

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	0.38	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
2	1.687500	1.687500	1.687500	1.687500	1.125000	1.125000	1.125000
3	1.687500	1.687500	1.687500	1.687500	1.265625	1.265625	1.687500
4	1.582031	1.582031	1.265625	1.582031	1.265625	1.582031	1.898438
5	1.186523	1.661133	1.423828	1.898438	1.423828	1.423828	1.898438
6	1.423828	1.779785	1.423828	1.779785	1.423828	1.423828	1.779785
7	1.334839	1.735291	1.601807	1.601807	1.334839	1.601807	1.601807
8	1.301468	1.802032	1.701920	1.501694	1.401581	1.501694	1.802032
9	1.351524	1.877117	1.802032	1.501694	1.351524	1.576778	1.802032
10	1.407838	1.858346	1.689405	1.464151	1.520465	1.520465	1.576778
11	1.435995	1.816111	1.562700	1.393759	1.520465	1.647170	1.689405
12	1.393759	1.742199	1.615494	1.488789	1.520465	1.647170	1.710523
13	1.401679	1.710523	1.591737	1.472950	1.520465	1.591737	1.520465
14	1.425436	1.674887	1.567979	1.496708	1.478890	1.621433	1.639251
15	1.469981	1.643706	1.576888	1.496708	1.496708	1.630342	1.630342
16	1.433231	1.603615	1.573548	1.533457	1.493367	1.583570	1.583570
17	1.450771	1.578559	1.563525	1.525940	1.510906	1.563525	1.623661

Data

Level	0.38	0.4	0.42	0.44	0.46	0.48	0.5
18	1.460167	1.578559	1.544733	1.516544	1.482718	1.572921	1.668762
19	1.429160	1.594063	1.530638	1.530638	1.496812	1.577149	1.640574
20	1.436559	1.585606	1.525353	1.534867	1.474614	1.569750	1.617318
21	1.446073	1.591156	1.534074	1.541209	1.479370	1.560236	1.584020
22	1.455586	1.589372	1.539425	1.539425	1.478776	1.566182	1.641102
23	1.446221	1.581345	1.547898	1.547898	1.485019	1.577331	1.650913
24	1.453914	1.585358	1.551243	1.535189	1.483012	1.578334	1.609440
25	1.455419	1.592382	1.563785	1.537446	1.478748	1.576579	1.610443
26	1.448270	1.603482	1.563409	1.533495	1.479877	1.581470	1.609690
27	1.448129	1.609832	1.559881	1.526864	1.478607	1.577237	1.601789
28	1.444848	1.615017	1.556601	1.532790	1.485486	1.579142	1.612160
29	1.448896	1.616049	1.554379	1.529853	1.476517	1.576046	1.618192
30	1.451872	1.616346	1.554914	1.527055	1.471516	1.577594	1.610810
31	1.449729	1.617150	1.550718	1.531565	1.470757	1.577639	1.608846
32	1.447017	1.616983	1.552593	1.536219	1.473034	1.575094	1.610252
33	1.447419	1.616631	1.551990	1.531498	1.474767	1.572030	1.613467
34	1.447589	1.615463	1.549579	1.532515	1.474315	1.575458	1.615463
35	1.449637	1.616141	1.549989	1.529139	1.473750	1.574780	1.617582
36	1.449245	1.617667	1.550127	1.530548	1.474640	1.574123	1.616396
37	1.450556	1.617897	1.552057	1.530150	1.474251	1.574417	1.617445
38	1.450008	1.619012	1.551682	1.529727	1.473464	1.574763	1.617743
39	1.450535	1.619714	1.550310	1.529070	1.473370	1.575921	1.617233
40	1.450019	1.620015	1.550705	1.529959	1.475150	1.575666	1.616817

Better Bound

Normalize

Seeing that our sequence on level n has length 3^n, we naturally index it by the first 3^n nonnegative integers.

Normalize

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 $[0, 1]$.

Normalize

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 $k \mapsto k/3^n$.

Notation

To emphasize our new indexing scheme, let $g_n(x)$ denotes the n^{th} level Bernoulli sequence where now $x \in [0,1]$. In other words,

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

Notation

To emphasize our new indexing scheme, let $g_n(x)$ denotes the n^{th} level Bernoulli sequence where now $x \in [0,1]$. In other words,

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

For a subset $S \subset [0,1]$, we define

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

An in depth example

We now walk through an in depth example to demonstrate our algorithm.

An in depth example

We now walk through an in depth example to demonstrate our algorithm.

Each entry on level n can be written as a sum of entries of previous levels. In this particular example we write each entry on level n as a sum of entries on level $n - 3$.

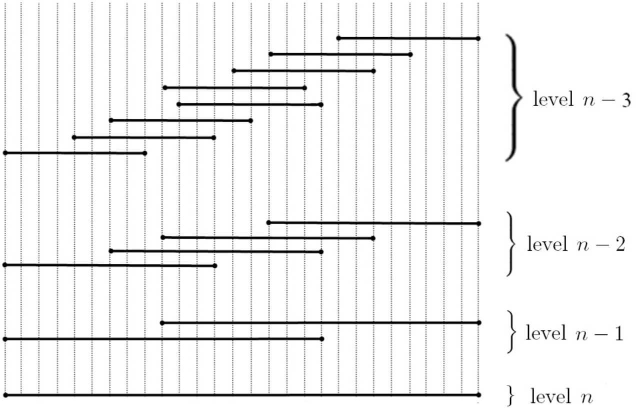
An in depth example

We now walk through an in depth example to demonstrate our algorithm.

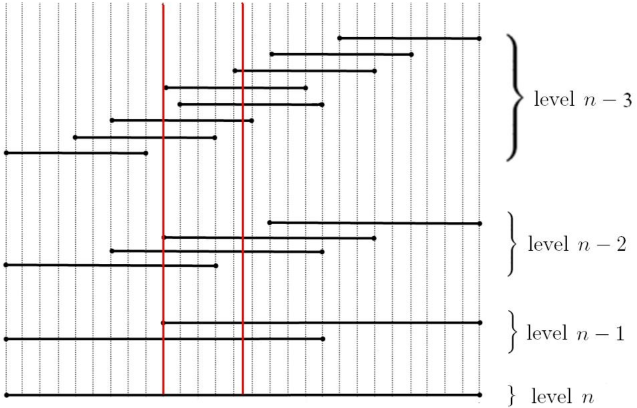
Each entry on level n can be written as a sum of entries of previous levels. In this particular example we write each entry on level n as a sum of entries on level $n-3$.

We break up the interval $[0, 1]$ into subintervals of length $1/81$. Let's see what we get.

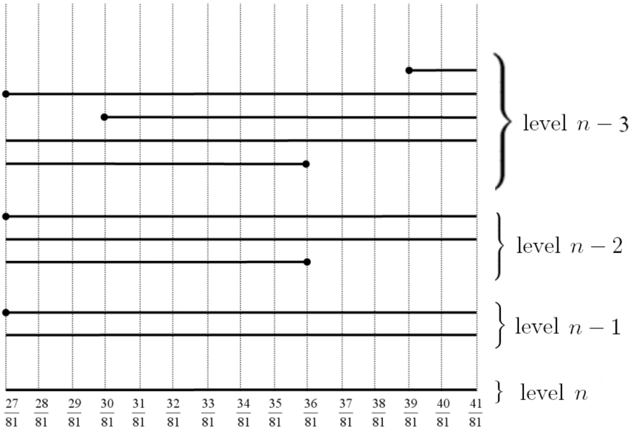
Pullback diagram



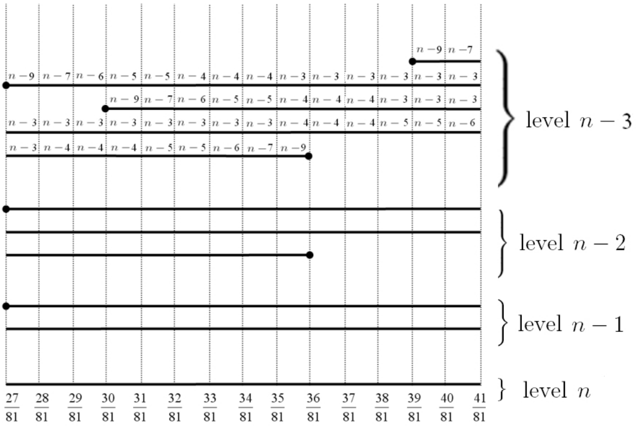
Pullback diagram



Pullback diagram



Pullback diagram



Largest real root

Interval	Polynomial	Largest real root
1	$x^n - 2x^{n-3} - x^{n-9}$	1.301688030
2	$x^n - x^{n-3} - x^{n-4} - x^{n-7}$	1.288452726
3	$x^n - x^{n-3} - x^{n-4} - x^{n-6}$	1.304077155
4	$x^n - x^{n-3} - x^{n-4} - x^{n-5} - x^{n-9}$	1.349240712
5	$x^n - x^{n-3} - x^{n-7} - 2x^{n-5}$	1.342242489
6	$x^n - x^{n-3} - x^{n-4} - x^{n-5} - x^{n-6}$	1.380277569
7	$x^n - x^{n-3} - x^{n-4} - x^{n-5} - x^{n-6}$	1.380277569
8	$x^n - x^{n-3} - x^{n-4} - x^{n-5} - x^{n-7}$	1.366811194
9	$x^n - x^{n-3} - 2x^{n-4} - x^{n-9}$	1.375394454
10	$x^n - x^{n-3} - 2x^{n-4}$	1.353209964
11	$x^n - x^{n-3} - 2x^{n-4}$	1.353209964
12	$x^n - 2x^{n-3} - x^{n-5}$	1.363964602
13	$x^n - 2x^{n-3} - x^{n-5} - x^{n-9}$	1.385877646
14	$x^n - 2x^{n-3} - x^{n-6} - x^{n-7}$	1.383834352

1.385877646

This example has given us the bound

1.385877646

This example has given us the bound

$$m_n = O(1.385877646^n)$$

1.385877646

This example has given us the bound

$$m_n = O(1.385877646^n)$$

The actual proof uses induction on n. The details are rather involved......if anyone is interested we would be glad to go through the proof after the talk.

Data

The above example can be generalized to give an algorithm that computes a θ so that $m_n = (\theta^n)$.

Data

The above example can be generalized to give an algorithm that computes a θ so that $m_n = (\theta^n)$.

After writing our code in Python, we ran many jobs on Condor. We submitted tens of thousands of jobs daily attempting to improve the bound on m_n.

Data

The above example can be generalized to give an algorithm that computes a θ so that $m_n = (\theta^n)$.

After writing our code in Python, we ran many jobs on Condor. 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 earlier. We succeeded in significantly improving the bound.

Data

The above example can be generalized to give an algorithm that computes a θ so that $m_n = (\theta^n)$.

After writing our code in Python, we ran many jobs on Condor. 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 earlier. We succeeded in significantly improving the bound.

We have shown $m_n = O(\theta^n)$ where $\theta = 1.33997599527$.
Specifically θ is a root of the polynomial

Data

The above example can be generalized to give an algorithm that computes a θ so that $m_n = (\theta^n)$.

After writing our code in Python, we ran many jobs on Condor. 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 earlier. We succeeded in significantly improving the bound.

We have shown $m_n = O(\theta^n)$ where $\theta = 1.33997599527$. Specifically θ is a root of the polynomial

$$X^{33} - 752X^8 - 520X^7 - 319X^6 - 231x^5 - 141X^4 - 101X^3 - 54X^2 - 50X - 83$$

Conclusion

Combinatorial point of view

Studying Bernoulli convolutions through the lens of combinatorics sheds much new insight on the subject.

Combinatorial point of view

Studying Bernoulli convolutions through the lens of combinatorics sheds much new insight on the subject.

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.

Combinatorial point of view

Studying Bernoulli convolutions through the lens of combinatorics sheds much new insight on the subject.

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).

Conclusion

Conclusion

We have shown that the maximums satisfy $m_n = O((1.33997599527)^n)$, improving upon the previously best known bound of $O((\sqrt{2})^n)$.

Conclusion

We have shown that the maximums satisfy $m_n = O((1.33997599527)^n)$, 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.

Questions

Questions

These are questions we have been unable to answer:

Questions

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?

Questions

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?

Questions

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?

Questions

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?

Questions

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?

However, we have provided partial answers for these questions through our algorithms and data.

Bound

1.41421356237

Bound

1.41421356237

1.33997599527

Bound

1.41421356237

1.33997599527

1.33333333333

Thank you for listening!