PROBABILISTIC DIVIDE-AND-CONQUER – A NEW METHOD FOR EXACT SIMULATION – AND LOWER BOUND EXPANSIONS FOR RANDOM BERNOLLI MATRICES VIA NOVEL INTEGER PARTITIONS

by

Stephen Anthony DeSalvo

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(APPLIED MATHEMATICS)

May 2012

Copyright 2012 Stephen Anthony DeSalvo
Dedication

This dissertation is dedicated to my family.
Epigraph

Whatever you do will be insignificant, but it is very important that you do it.

– Mahatma Gandhi

Silence is sometimes the best answer.

– Dalai Lama XIV

"Don’t try to be a great man, just be a man,

and let history make its own judgments”

– Zefram Cochrane

A monkey is typing ...
Acknowledgements

First, I would like to thank the anonymous referees for the paper on random Bernoulli matrices, who found numerous typos and made some helpful suggestions for improving the presentation of the material.

I would like to thank Paul Newton for his guidance early on in my graduate studies. Also Martin Lo of the Jet Propulsion Laboratory was a source of inspiration and support.

I learned many valuable lessons from Max Jodeit during my last year as an undergraduate at the University of Minnesota, and I benefitted tremendously from all the time he spent with me during office hours.

Gary Rosen was also very supportive during my time at USC, as were many of the faculty in the department of mathematics. I also enjoyed the visits by Mario Martelli from the Claremont Graduate University, where I received not only valuable advice but also a chance to practice my Italian.

My fellow graduate students also gave me an invaluable perspective on math and the world, and I wish the best for them.

I was fortunate to befriend a great many individuals during my time as a graduate student, and I thank them for their support and I hope I was a source of support for them as well.
I would like to thank my siblings: Veronica, Paul, Chris and my parents, for their support over the years. My dad in particular was able to impart valuable advice from his days as a graduate student. My mom supported me in pretty much anything I decided to do, and made sure I had everything I needed while away from home. I thank my brother Chris for creating SpaceTime, which gave me a productive hobby of custom-designing and testing a high level programming language.

My wife Rehana has been especially understanding this final year, and her family has welcomed me into their lives. I especially thank Nazneen, Yousuf, Sara, and Farah.

A heartfelt thanks to Dana and David Dornsife, whom I have never met, but who generously donated to the College of Letters, Arts, and Sciences, and as a result there were many more dissertation fellowships awarded, myself being the recipient of one of them. This award freed me from teaching duties, which allowed me much more time to pursue my mathematical interests.

Finally, my sincerest thanks to my advisor, Richard Arratia, who very closely guided my development and chose problems that were fun and interesting to develop. I learned more about probabilistic intuition by simply listening to him speak his thoughts out loud than in any class or book, and I appreciate all the time he spent with me.
# Table of Contents

Dedication ii  
Epigraph iii  
Acknowledgements iv  
List Of Tables viii  
Abstract ix  

Chapter 1: Introduction 1  

Chapter 2: Integer Partitions 3  
2.1 History 3  
2.2 Introduction 4  
2.3 Generating function 5  
2.4 A series for $p(n)$ 7  
2.5 Numerical evaluation of $p(n)$ 8  

Chapter 3: Random Bernoulli Matrices 10  
3.1 Introduction 10  
3.2 Lower bound expansions 13  
3.3 Templates, Bernoulli orthogonal complements, and novel partitions 15  
3.4 Polynomial coefficients arising from inclusion-exclusion 35  
3.5 Interaction of left and right null vectors 45  

Chapter 4: Probabilistic Divide-And-Conquer 48  
4.1 Introduction 48  
4.1.1 Exact simulation 48  
4.1.2 Probabilistic divide-and-conquer, motivated by mix-and-match 51  
4.2 The basic lemma for exact sampling with divide-and-conquer 53  
4.2.1 Algorithmic implications of the basic lemma 55  
4.2.2 Use of acceptance/rejection sampling 56  
4.2.3 Caution: mix-and-match not yet enabled 58  
4.3 *Simple* matching enables mix-and-match 59
4.4 Algorithms for simulating random partitions of $n$.............. 62
  4.4.1 For baseline comparison: table methods.................. 63
  4.4.2 Waiting to get lucky.................................... 65
  4.4.3 Divide-and-conquer, by small versus large................ 68
    4.4.3.1 Divide-and-conquer with mix-and-match............. 70
    4.4.3.2 Roaming $x$........................................ 70
    4.4.3.3 Deliberately missing data.......................... 71
  4.4.4 Divide-and-conquer with a trivial second half........... 73
  4.4.5 Self-similar iterative divide-and-conquer: $p(z) = d(z)p(z^2)$...... 74
    4.4.5.1 Exploiting a parity constraint...................... 76
    4.4.5.2 The overall cost of the main problem and all its subproblems 78
    4.4.5.3 A variation based on $p(z) = p_{odd}(z)p(z^2)$........ 80
  4.5 For integer partitions: review, method of choice........... 82
    4.5.1 Method of choice for unrestricted partitions........... 82
    4.5.2 Complexity considerations............................. 84
    4.5.3 Partitions with restrictions............................ 87
    4.5.4 An eye for gathering statistics......................... 88
  4.6 For comparison: opportunistic divide-and-conquer
    with mix-and-match........................................... 89

Chapter 5: Opportunistic Probabilistic Divide-And-Conquer 92

References 94
List Of Tables

3.1 Novel partitions sorted by $r_\lambda$. Conjectured to be complete with respect to $r_\lambda \geq 38/256$. ................................. 30

3.2 Novel partitions of length 8, conjectured to be the complete list. ....... 33
Abstract

This thesis is divided into two areas of combinatorial probability: probabilistic divide-and-conquer, and random Bernoulli matrices via novel integer partitions.

Probabilistic divide-and-conquer is a new method of exact sampling that simulates from a set of objects by dividing each object into two disjoint parts, and pieces them together.

The study of random Bernoulli matrices is driven by the asymptotics of the probability that a random matrix whose entries are independent, identically distributed Bernoulli random variables with parameter 1/2 is singular. Our approach is an inclusion-exclusion expansion for this probability, defining a necessary and sufficient class of integer partitions as an index set to characterize all of the singularities.
Chapter 1

Introduction

An expert is a person who has made all the mistakes which can be made in a narrow field.

– Niels Bohr.

This thesis is primarily composed of two papers in combinatorial probability, one recently accepted to the Annals of Combinatorics on random Bernoulli matrices (RBM), which is posted on the ArXiv at http://arxiv.org/abs/1105.2834, and the other submitted to Combinatorics, Probability and Computing on probabilistic divide-and-conquer (PDC), also posted on the ArXiv at http://arxiv.org/abs/1110.3856.

Chapter 2 introduces some basic notation and results on integer partitions. Chapter 3 is based on the the paper submitted to the Annals of Combinatorics on the probability that a random Bernoulli matrix is singular, with corrections added from the anonymous referees. Chapter 4 is taken almost verbatim from the paper submitted to the journal Combinatorics, Probability, Computing on probabilistic divide-and-conquer.
These two disparate subjects have a common thread in this document, viz., integer partitions. Integer partitions are used as an example of an object that yields to efficient sampling methods under probabilistic divide-and-conquer, and they are also used as an index set to describe the singularities in an inclusion-exclusion expansion of the probability that a random Bernoulli matrix is singular.

Chapter 2 introduces integer partitions and is a very brief account of some of the major results that I found useful during the development of Chapter 4. A very nicely written book at the undergraduate level is [20], which is a must-read for anyone beginning a serious study of integer partitions. Integer partitions are a fundamental combinatorial structure and the information provided in Chapter 2 should be a sufficient starting point for understanding the other chapters.

Chapter 3 was developed shortly after a USC Mathematics Colloquium talk my advisor and I attended by Philip Matchett Wood on [10] in January 2011. We were drawn to the subject mainly because the index sets were integer partitions, and because no one had yet worked on the lower bound expansions in great detail.

Chapter 4 began by the posing of a simple question to the author by the author’s advisor, “how would you simulate a random integer partition quickly for as large an $n$ as possible?” Building upon the work of Fristedt [17] and [7], an idea was born to break up the task into two smaller tasks, simulate each one separately, and piece them back together.
Chapter 2

Integer Partitions

An equation means nothing to me unless it expresses a thought of God.
– Srinivasa Ramanujan

I am interested in mathematics only as a creative art.
– G. H. Hardy

2.1 History


However, the history that has driven many of the developments that we will find pertinent has its origins in a paper written by Hardy and Ramanujan [22] on estimating
the number of integer partitions of size $n$, denoted by $p(n)$. This collaboration led to many other discoveries, including Ramanujan’s congruence observations, but in particular it opened up the subject to analysis by introducing an integration technique now known as the circle method. In the words of Hardy and Ramanujan,

The idea which dominates this paper is that of obtaining asymptotic formulæ for $p(n)$ by a detailed study of the integral (1.21). This idea is an extremely obvious one; it is the idea which has dominated nine-tenths of modern research in the analytic theory of numbers: and it may seem very strange that it should never have been applied to this particular problem before. Of this there are no doubt two explanations. The first is that the theory of partitions has received its most important developments, since its foundation by Euler, at the hands of a series of mathematicians whose interests have lain primarily in algebra. The second and more fundamental reason is to be found in the extreme complexity of the behavior of the generating function $f(x)$ near a point of the unit circle.

2.2 Introduction

An integer partition is an unordered list of positive integers. An integer partition of size $n$ is a list of unordered positive integers whose sum is $n$. For example, the seven integer partitions of size 5 are $\{5\}, \{4,1\}, \{3,2\}, \{3,1,1\}, \{2,2,1\}, \{2,1,1,1\}, \{1,1,1,1,1\}$. It is standard to write the elements in decreasing order (reverse lexicographic) and to drop the commas and brackets when there is no danger of ambiguity, simply listing the elements consecutively, as in 5, 41, 3, 3, 2, 2, 1, 1, 1, 1, 1. A further notational convenience is to introduce exponential notation for the sake of the latter two partitions, so that we may write $5^1, 4^11^1, 3^12^1, 3^11^2, 2^21^1, 2^11^3, 1^5$ and if there is no confusion we may drop the superscript 1s as we please, so that our list may be more easily represented as 5, 41, 32, 312, 221, 213, 15.
The number of integer partitions of size $n$ is denoted by $p(n)$, where for example $p(5) = 7$. The first few values are (starting with $p(0) := 1$) $1, 1, 2, 3, 5, 7, 11, 15, 22, 30, 42$, where $p(10) = 42$. $p(n)$ grows very rapidly, as is evidenced by $p(100) = 190569292 \approx 1.9 \times 10^6$, and $p(1000) \approx 2.4 \times 10^{31}$. In fact, if we define $a_n \sim b_n$ to mean that $\lim_{n \to \infty} \frac{a_n}{b_n} = 1$, then it was first shown by Hardy and Ramanujan [22] that

$$p(n) \sim \frac{\exp(c_1 \sqrt{n})}{c_2 n},$$

where $c_1 = \sqrt{\frac{3}{2}}\pi$ and $c_2 = 4\sqrt{3}$. The proof of this result is quite technical, but proving for example, $p(n) < \exp(c_1 \sqrt{n})$ and $p(n) < \exp(c_1 \sqrt{n})/c_3 \sqrt{n}$ for some $c_3 > 0$, is easily established using standard elementary analysis; these cases are treated in [3].

### 2.3 Generating function

Generating functions play a significant role in the study of integer partitions. A good introduction to generating functions is the book *generatingfunctionology* by Herbert Wilf [52]. The idea of a generating function is to encode information into a lattice structure. For example, suppose we wanted to encode the list of numbers $a = \{a_0, a_1, \ldots, a_n\}$ into a single object in a way that makes them easily accessible. The ordinary generating function for the set $a$ would be the polynomial $g(x) = a_0 + a_1 x + a_2 x^2 + \ldots a_n x^n$, with $g(0) = a_0$, $g'(0) = a_1$, $g''(0)/2 = a_2$, $\ldots$, $g^{(n)}(0)/n! = a_n$. 
The generating function for \( p(n) \) is given by \( f(x) = \prod_{i \geq 1} (1 - x^i)^{-1} \). We can expand this function out as follows,

\[
\begin{align*}
f(x) &= \frac{1}{1-x} \frac{1}{1-x^2} \frac{1}{1-x^3} \cdots \\
&= (1 + x + x^2 + \ldots)(1 + x^2 + x^4 + \ldots)(1 + x^3 + x^6 + \ldots) \cdots \\
&= 1 + x + 2x^2 + 3x^3 + 5x^4 + 7x^5 + \ldots \\
&= \sum_{n=1}^{\infty} p(n)x^n.
\end{align*}
\]

The coefficients \( p(n) \) represent unrestricted partitions, but we could also ask for the number of partitions of size \( n \) with distinct parts. For example, the partitions of 5 with distinct parts are 5, 41, 32. The generating function is given by \( f_d(x) = (1 + x)(1 + x^2)(1 + x^3) \cdots \). A more general version places at most \( b_1 \) 1s, \( b_2 \) 2s, etc. For example, the number of partitions of 5 with at most 2 ones are 5, 41, 32, 311, 221. The generating function \( h(x) \) has the form

\[
h(x) = \prod_{i \geq 1} (1 + x + x^2 + \cdots x^{b_1})(1 + x^2 + \cdots x^{2b_2})(1 + x^3 + x^6 + \cdots + x^{3b_3}) \cdots .
\]

The generating function can be a source of identities, for instance,

\[
f_d(x) = \prod_{i} (1 + x^i) = \prod_{i} (1 + x^i) \frac{1-x^i}{1-x^i} = \prod_{i} \frac{1-x^{2i}}{1-x^i} = \prod_{i \ odd} \frac{1}{1-x^i} = f_o(x),
\]

where \( f_o(x) \) is the generating function for the number partitions with only odd parts.

This allows us to conclude that the number of partitions of \( n \) with distinct parts is equal to the number of partitions with only odd parts.
A good treatment on the use of generating functions in integer partitions is [2] and also [51].

2.4 A series for \( p(n) \)

The celebrated result of Hardy and Ramanujan is the asymptotic formula

\[
p(n) \sim \exp(c_1 \sqrt{n}) / c_2 n,
\]

but their work showed quite a bit more than just that. Their final result states that for any fixed \( n \), there exists an \( a > 0 \) such that

\[
p(n) = \sum_{q=1}^{a \sqrt{n}} A_q \phi_q + O(n^{-1/4}),
\]

where \( A_q \) and \( \phi_q \) are explicit, computable expressions. That is, once \( n \) is fixed, only \( O(\sqrt{n}) \) summands need to be considered before one is within 0.5 of the true answer. We do not obtain the value of \( a \) from their theory, however; so while it is of great theoretical insight and yields the asymptotic formula from Equation (2.1), from a computational point of view it does not allow one to compute this value with absolute certainty.

In 1937, however, Rademacher [45] published an infinite series for \( p(n) \) that converges, that is,

\[
p(n) = \sum_{q=1}^{\infty} A_q \phi_q,
\]
for similarly defined functions $A_q$ and $\varphi_q$. With a convergent series representation, this opens up the possibility of estimating the error involved in truncating the series, whereas the original series in Equation (2.2) is not convergent.

2.5 Numerical evaluation of $p(n)$

Shortly after the work of Rademacher, Lehmer [31, 33] worked on coming up with explicit upper bounds on a truncated series, bounding the remainder in a way that guaranteed the estimate was within 0.5 of the true answer. In particular, his Theorem 13 from [31] states that

**Theorem 1** (Lehmer). If only $2n^{1/2}/3$ terms of the Hardy-Ramanujan series be taken, the resulting sum will differ from $p(n)$ by less than 1/2, provided $n > 600$.

The cut-off value of 600 was chosen because at the time (ca. 1930s) there was a table of values of $p(n)$ for $n = 0, 1, 2, \ldots, 600$. He goes on to write that the $2/3$ can be improved upon, for example to $1/2$ if $n > 3600$, and this is generalized to

**Theorem 2** (Lehmer). Let $\delta > 1$ and let $c = \pi(2/3)^{1/2} = 2.565\ldots$. Then $p(n)$ is the nearest integer to the sum of the first $n^{1/2}/\delta$ terms of the Hardy-Ramanujan series provided

$$n > \frac{27^{1/2} e^{\delta}}{\delta^2} \left( \frac{\sinh(c \delta)}{c^3 \delta^3} + \frac{1}{6} \right)^3 = O(e^{3c\delta\delta^{-11}}).$$

A related question is for $\delta > 1$, how large does $n$ need to be so that the first $n^{1/2}/\delta$ terms of the Hardy-Ramanujan series are within a specified Tolerance of the true value? In the age of computers and floating point accuracy, it would be grossly inefficient to
attempt to compute a floating point result for $p(1000) = 2 \times 10^{31}$, accurate to say 16 digits, by computing all 32 digits accurately. The reason why this is a worthwhile endeavor is because the asymptotic expansions are heavily weighted in the first few terms. To again quote [22],

Taking $n = 100$, we found that the first six terms of our formula gave

\[
\begin{align*}
190568944.783 \\
+348.872 \\
-2.598 \\
+.685 \\
+.318 \\
-.064 \\
190569291.996,
\end{align*}
\]

while $p(100) = 190569292$; so that the error after six terms is only .004.

We then proceeded to calculate $p(200)$, and found

\[
\begin{align*}
3,972,998,993,185.896 \\
+36,282.978 \\
-87.555 \\
+5.147 \\
+1.424 \\
+0.071 \\
+0.000 \\
+0.043 \\
3,972,999,029,388.004,
\end{align*}
\]

and Major MacMahon’s subsequent calculations showed that $p(200)$ is, in fact,

\[
3,972,999,029,388.
\]

This demonstrates the striking principle of the Hardy-Ramanujan series, which is that the first term gives striking accuracy, with remaining terms contributing towards pinning down the smaller digits.
Chapter 3

Random Bernoulli Matrices

There is no philosophy which is not founded upon knowledge
of the phenomena, but to get any profit from this knowledge it
is absolutely necessary to be a mathematician.

– Daniel Bernoulli

A mathematician is a device for turning coffee into theorems.

– Paul Erdos

3.1 Introduction

To introduce our problem, we quote verbatim\(^1\) the opening 3 paragraphs of a paper by
Kahn, Komlós, and Szemerédi [24]:

\(^1\)Apart from correcting a typographical error, and using our own display equation numbering and reference numbering.
1.1. The problem. For $M_n$ a random $n \times n \pm 1$-matrix (“random” meaning with respect to uniform distribution), set

$$P_n = Pr(M_n \text{ is singular}).$$

The question considered in this paper is an old and rather notorious one: What is the asymptotic behavior of $P_n$?

It seems often to have been conjectured that

$$P_n = (1 + o(1)) n^2 / 2^{n-1},$$

that is, that $P_n$ is essentially the probability that $M_n$ contains two rows or two columns which are equal up to a sign. This conjecture is perhaps best regarded as folklore. It is more or less stated in [30] and is mentioned explicitly, as a standing conjecture, in [40], but has surely been recognized as the probable truth for considerably longer. (It has also been conjectured ([37]) that $P_n / (n^2 2^{-n}) \to \infty$.)

Of course the guess in (3.1) may be sharpened, e.g., to

$$P_n - 2 \left(\begin{array}{c} n \\ 2 \end{array}\right) 2^{-(n-1)} \sim 2^4 \left(\begin{array}{c} n \\ 4 \end{array}\right) \left(\frac{3}{8}\right)^n,$$

the right-hand side being essentially the probability of having a minimal row or column dependency of length 4.

Our paraphrase: (3.1) says, for a Bernoulli matrix to be singular, the most likely way is having a left or right null vector of the template 11, and (3.2) says that the second most likely way is having a left or right null vector of the template 1111.

When one continues the expansion (3.2) to higher order, two features emerge. First, the patterns, corresponding to 11, 1111, . . . , which we call templates, have a rich structure. The third most likely template is $1^6 = 111111$, with exponential decay $(5/16)^n$, and the fourth most likely template is $1^8$, with exponential decay $(35/128)^n$. The real pattern, which is not simply $1^{2m}$, emerges starting with the fifth template, 21111.

In the higher order continuation of (3.2), the second feature also first appears with the fifth term, which is the distinction between $2 \left(\begin{array}{c} n \\ 2 \end{array}\right) 2^{-(n-1)}$, which is the expected number of occurrences of a right or left null vector of template 11, and the probability of one
or more such occurrences. This is because the exponential decay rate for 21111, which is \((1/4)^n\), is small enough to force consideration of the probability that two null vectors from the template 11 appear.

In Section 3.3 we define what we call novel integer partitions, which include the templates from the previous paragraph, and we prove that this set is both necessary (Theorem 4) and sufficient (Theorem 2) for detecting singularities.

The natural extensions of (3.2) are our Conjectures 1 and 2, immediately below.

**Conjecture 1.** Let \(S\) denote the event that the \(n\) by \(n\) random Bernoulli matrix \(M = M_n\) is singular, with \(P_n = \mathbb{P}(S) = \mathbb{P}_n(S)\). Then for every \(\epsilon > 0\),

\[
\begin{align*}
\mathbb{P}(S \setminus D_{11}) &= o\left(\left(\frac{3 + \epsilon}{8}\right)^n\right) \\
\mathbb{P}(S \setminus (D_{11} \cup D_{1111})) &= o\left(\left(\frac{5 + \epsilon}{16}\right)^n\right) \\
\mathbb{P}(S \setminus (D_{11} \cup D_{1111} \cup D_{16})) &= o\left(\left(\frac{35 + \epsilon}{128}\right)^n\right) \\
\mathbb{P}(S \setminus (D_{11} \cup D_{1111} \cup D_{16} \cup D_{18})) &= o\left(\left(\frac{1 + \epsilon}{4}\right)^n\right) \\
\mathbb{P}(S \setminus (D_{11} \cup D_{1111} \cup D_{16} \cup D_{18} \cup D_{2111})) &= o\left(\left(\frac{63 + \epsilon}{256}\right)^n\right) \\
\mathbb{P}(S \setminus E_6) &= o\left(\left(\frac{15 + \epsilon}{64}\right)^n\right) \\
\mathbb{P}(S \setminus E_7) &= o\left(\left(\frac{231 + \epsilon}{1024}\right)^n\right) \\
\mathbb{P}(S \setminus E_8) &= o\left(\left(\frac{7 + \epsilon}{32}\right)^n\right)
\end{align*}
\]
and so on, where \( E_6 = D_{11} \cup D_{1111} \cup D_{16} \cup D_{18} \cup D_{21111} \cup D_{110} \), \( E_7 = E_6 \cup D_{216} \), and \( E_8 = E_7 \cup D_{112} \).

**Conjecture 2.** There is a list of novel partitions, which in order of exponential rate (3.5) is \( \lambda(1) = 11, \lambda(2) = 1111, \ldots, \lambda(5) = 21111, \ldots, \lambda(8) = 1^{12}, \ldots \), such that for every \( r > 0 \), there exists \( K > 0 \)

\[
P \left( S \setminus \bigcup_{i=1}^{K} D_{\lambda(i)} \right) = o(r^n).
\]

It was first shown that \( P_n \) decays exponentially, in [24], with an upper bound of .999\(^n\). This was later improved by Tao and Vu [47] to \( (.958 + o(1))^n \) and again [48] to \( (3/4 + o(1))^n \). (See also [49]). Recently Bourgain, Vu, and Wood [10] provided a further improvement to \( \left( \frac{1}{\sqrt{2}} + o(1) \right)^n \), which is currently the most accurate bound.

Section 3.2 presents an explicit lower bound expansion of \( P_n \), whose exponential decay rates are based on the novel integer partitions of Section 3.3. In Section 3.4 we derive the polynomial coefficients of our lower bound expansion. In Section 3.5 we give some bounds on the interaction of potential left and right null vectors, hoping to supply a tool for use in bounding \( \mathbb{P}(S \setminus D_{11}) \).

### 3.2 Lower bound expansions

The expansion in (3.2) can be continued by considering events \( D_{1111} \), that \( M \) has a left or right null eigenvector of the form \( e_i \pm e_j \pm e_k \pm e_\ell \), with \( D_{16}, D_{18}, D_{214}, D_{110}, D_{216}, \) and \( D_{112} \) defined similarly. Letting \( E_8 = D_{11} \cup D_{14} \cup D_{16} \cup D_{18} \cup D_{214} \cup D_{110} \cup D_{216} \cup D_{112} \), our expansion can be stated as
Theorem 1. For each \( n \),

\[ P_n \geq P(D_{11}) \geq 4 \left( \frac{n}{2} \right)^n - \left( 12 \left( \frac{n}{2} \right)^2 - 4 \left( \frac{n}{2} \right) \right) \left( \frac{1}{4} \right)^n. \]

For each \( n \), the event \( E = E_8 \) is a subset of the event that \( M \) is singular, hence trivially, \( P_n \geq P_n(E) \), and for all \( \epsilon > 0 \),

\[ P_n(E) = \sum_{i=1}^{8} Q_i(n) \left( \frac{1}{2} \right)^n + o \left( \left( \frac{7 + \epsilon}{32} \right)^n \right) \]

where

\[ Q_1(n) = 2^{2 \left( \frac{n}{2} \right)}, \quad Q_6(n) = 2^{10 \left( \frac{n}{10} \right)}, \]

\[ Q_2(n) = 2^{4 \left( \frac{n}{4} \right)}, \quad Q_7(n) = 2^{7 \left( \frac{7}{7} \right) \left( \frac{n}{7} \right)}, \]

\[ Q_3(n) = 2^{6 \left( \frac{n}{6} \right)}, \quad Q_8(n) = 2^{12 \left( \frac{n}{12} \right)}, \]

\[ Q_4(n) = 2^{8 \left( \frac{n}{8} \right)}, \]

\[ Q_5(n) = 2^{5 \left( \frac{5}{1} \right) \left( \frac{n}{5} \right)} - 4 \left( 2 \left( \frac{n}{2} \right)^2 + 8 \left( \frac{n}{4} \right) + 5 \left( \frac{n}{3} \right) \right). \]

Proof. This result follows easily from the combination of Lemmas 3 – 6, given in Sections 3.3 and 3.4.
3.3 Templates, Bernoulli orthogonal complements, and novel partitions

In the explicit expansions of Conjecture 1 and Theorem 1 there are exponentially decaying factors such as \((1/2)^n\), \((3/8)^n\), \((1/4)^n\), corresponding to the integer partitions 11, 1111, and 21111. When the expansion is carried out to high order, an obvious necessary condition for a partition \(\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)\) to appear is that it be fairly divisible, in the sense that for some combination of signs, \(0 = \lambda_1 \pm \lambda_2 \pm \cdots \pm \lambda_k\). However, this is not sufficient; some fairly divisible partitions, such as 211 and 321, will never appear. We call the partitions that eventually appear novel. The definitions below will let us characterize these novel partitions, and, to a limited extent, compute them explicitly.

**Definition 1. Integer partition, as a template for vectors.**

For a given partition \(\lambda\) with \(k\) parts, let \(V_\lambda \subset \mathbb{N} \times \mathbb{Z}^{k-1}\) denote the set of all vectors formed by reordering the parts of \(\lambda\), together with all combinations of plus and minus with the requirement that the first coordinate always has a plus.\(^2\)

If \(\lambda\) has \(c(i)\) parts of size \(i\), so that \(\text{len}(\lambda) := c(1) + c(2) + \cdots = k\), then

\[
| V_\lambda | = 2^{k-1} \frac{k!}{c(1)!c(2)! \cdots}.
\]

**Notation: coordinate injection, from \(\mathbb{R}^k\) to \(\mathbb{R}^n\).**

We often want to pad our vectors of length \(k\) with zeros, to get a vector of length \(n\). We say that a \(k\) by \(n\) matrix \(C\), with all entries 0 or 1, is a coordinate injection matrix, if

\(^2\)We write \(\mathbb{N} := \{1, 2, \ldots\}\) for the set of strictly positive integers.
every row has exactly one 1, and no column has more than one 1, and \( C_{ij} = C_{i'j'} = 1 \) with \( i < i' \) implies \( j < j' \). (This last requirement is imposed, since our \( V_\lambda \) already accounts for all rearrangements of the parts.) There are \( \binom{n}{k} \) such matrices. We speak of vectors of length \( n \), of the form \( vC \) for some \( v \in V_\lambda \) as having template \( \lambda \).

**Definition 2. Templates, used in \( n \) dimensions.**

We write \( V_\lambda^{(n)} \) for the subset of \( \mathbb{Z}^n \) of length \( n \) vectors with template \( \lambda \). Note, vectors in \( V_\lambda^{(n)} \) may have first coordinate zero, but the first non-zero coordinate must be strictly positive.

The number of vectors of length \( n \), having template \( \lambda \), is

\[
|V_\lambda^{(n)}| = \binom{n}{k} \quad |V_\lambda| = 2^{k-1} \frac{(n)_k}{c(1)!c(2)! \cdots},
\]

where we write \( (n)_k \) for \( n \text{ falling } k \).

For an integer partition \( \lambda \) with parts \( \lambda_1, \lambda_2, \ldots, \lambda_k \), and \( X = (\epsilon_1, \ldots, \epsilon_k) \) a vector of independent Bernoulli random variables, let \( \lambda \cdot X = \lambda_1 \epsilon_1 + \cdots + \lambda_k \epsilon_k \) denote the weighted sum, and define

\[
r_\lambda := \mathbb{P}(\lambda \cdot X = 0).
\]

We can then compute, for example, \( r_{11} = 1/2 \), \( r_{12m} = \frac{\binom{2m}{m}}{2^{2m}} \), \( r_{2111} = 1/4 \).

**Definition 3. Bernoulli orthogonal complement.**

For a vector \( v \in \mathbb{Z}^k \),

\[
v^\perp_B = \{ x \in \{-1, 1\}^k : v \cdot x = 0 \}.
\]
This definition can also be applied when \( v = \lambda = (\lambda_1, \ldots, \lambda_k) \) with \( \lambda_1 \geq \cdots \geq \lambda_k > 0 \) is an integer partition with \( k \) parts, in which case the probability \( r_\lambda \) defined by (3.5) is given by

\[
r_\lambda = \frac{|v^{\perp_B}|}{2^k}. \tag{3.6}
\]

**Remark 1.** Clearly \( x \in v^{\perp_B} \) iff \( -x \in v^{\perp_B} \), that is \( -v^{\perp_B} = v^{\perp_B} \). For a partition \( \lambda \), all \( v \) in \( V_\lambda \) have the same size \( |v^{\perp_B}| \) for their Bernoulli orthogonal complement. Indeed, the various sets \( v^{\perp_B} \) for \( v \in V_\lambda \) are related, by permuting the \( k \) coordinates, and applying, for some fixed \( I \subset \{2, 3, \ldots, k\} \), sign flips to all the coordinates indexed by \( I \). Hence, if \( \lambda^{\perp_B} \neq \emptyset \), then \( \{-1, 1\}^k = \bigcup_{v \in V_\lambda} v^{\perp_B} \).

**Definition 4.** The matrix \( A^{(\lambda)} \) for \( \lambda^{\perp_B} \).

For an integer partition \( \lambda \) of length \( k \), with \( 2p = |\lambda^{\perp_B}| > 0 \), the matrix \( A^{(\lambda)} \) for the Bernoulli orthogonal complement of \( \lambda \) is the \( k \) by \( p \) matrix whose columns are those elements of \( \lambda^{\perp_B} \) whose first coordinate is \( +1 \), taken in lexicographic order, with \( +1 \) preceding \( -1 \).
Example 1. Displaying the Bernoulli orthogonal complement.

When $\lambda = 1111$, we have

$$1111^\perp_B = \{ (+1, +1, -1, -1), (+1, -1, +1, -1), (+1, -1, -1, +1), (-1, -1, +1, +1), (-1, +1, -1, +1), (-1, +1, +1, -1) \}$$

$\{ + + - - , + - + - , + - - + , - - + + , - + - + , - + + - \}$

where the second representation omits the parentheses and commas for each $k$-tuple, and also shows only the signs.

Say that $\lambda$ has length $k$, and $2p = |\lambda^\perp_B|$. Showing only those elements of $\lambda^\perp_B$ that begin with $+$, and transposing, we have a $k$ by $p$ display, to be thought of as an economical representation of the set $\lambda^\perp_B$; we use this display in Example 4. Treating the same $k$ by $p$ array as a matrix, we have $A^{(\lambda)}$, as defined in Definition 4. For instance,

$$A^{(1111)} = \begin{pmatrix} + & + & + \\ + & - & - \\ - & - & - \\ - & - & + \end{pmatrix}.$$
Definition 5. Equivalence of templates.

For partitions $\lambda, \mu$ with the same number of parts, we say $\lambda \leftrightarrow \mu$ iff $\exists v \in V_\lambda, w \in V_\mu$, such that $v^{\perp B} = w^{\perp B}$. Clearly, this $\leftrightarrow$ is an equivalence relation on integer partitions. (Note, $\lambda \leftrightarrow \mu$ iff $\exists w \in V_\mu$ such that $\lambda^{\perp B} = w^{\perp B}$, that is, we need only apply rearrangement and sign flips to one of $\lambda, \mu$.)

Example 2. Equivalence is more than just multiples.

Trivially, scalar multiples of any partition are all equivalent to each other. But equivalence involves more. Let $\lambda = 321, \mu = 211$. Then $321 \leftrightarrow 211$ since

$$\mu^{\perp B} = \lambda^{\perp B} = \{ + - -, - + + \},$$

with no need to apply rearrangements or sign flips. Rearrangement and sign flips may change the Bernoulli complement. For instance,

$$V_\mu = \{(2,1,1), (1,2,1), (1,1,2), (2,1,-1), (1,2,-1), (1,1,-2),$$

$$(2,-1,1), (1,-2,1), (1,-1,2), (2,-1,-1), (1,-2,-1), (1,-1,-2)\}.$$

and with $v = (1,-2,-1) \in V_\mu$, we have

$$v^{\perp B} = \{ + + -, - - + \} \neq \mu^{\perp B}.$$
Example 3. Rearrangements are needed in the definition of equivalence.\(^3\)

The partitions

\[
\mu = 9 \ 7 \ 4 \ 4 \ 3 \ 1 \\
\lambda = 7 \ 5 \ 5 \ 4 \ 4 \ 3
\]

are such that \(\mu^\perp \neq \lambda^\perp\), but for \(v = (9, 7, 3, 4, 4, 1) \in V_\mu\), \(v^\perp = \lambda^\perp\), hence \(\mu \leftrightarrow \lambda\).

Definition 6. Reduction of templates.

For any partitions \(\mu, \lambda\) with \(\mu\) having \(m\) parts and \(\lambda\) having \(k\) parts, \(m \geq k > 0\), we say that \(\mu \rightarrow \lambda\) (read \(\mu\) reduces to \(\lambda\) or \(\mu\) implies \(\lambda\)) iff either

\[
k = m \text{ and } \exists v \in V_\lambda, \mu^\perp \subset v^\perp,
\]

(3.7)

or else

\[
\exists I \subset \{1, \ldots, m\}, \ v \in V_\lambda, \ \text{Proj}_{I} \mu^\perp \subset v^\perp.
\]

(3.8)

Clearly, the relation \(\rightarrow\) is transitive. Our use of the subset symbol \(\subset\) includes equality. We note that \((\lambda \rightarrow \mu \text{ and } \mu \rightarrow \lambda)\) iff \(\lambda \leftrightarrow \mu\), so that definitions 5 and 6 are compatible.

Remark 2. Definition 6 is set up so that it is obvious that if \(\mu \rightarrow \lambda\), and \(w \in V_\mu^{(n)}\), and \(M\) is an \(n\) by \(n\) Bernoulli matrix with \(wM = 0\), then there exists \(v \in V_\lambda^{(n)}\) with \(vM = 0\).

Definition 7. Strict reduction.

We define a relation of strict reduction, \(\mu \not\rightarrow \lambda\) (read \(\mu\) strictly reduces to \(\lambda\)) iff

\(^3\)This example was found by considering partitions of the form \((a + x_1, b + x_1, b, b, a - x_1, b - x_1)\) and \((a + x_2, a - x_2, b + x_2, b, b, b - x_2)\), where \(a \geq b, x_1, x_2\) are chosen so that the two b’s must cancel, but are in a different monotonic order in each partition. Here we have taken \(a = 6, b = 4, x_1 = 3, x_2 = 1\).
\( \mu \rightarrow \lambda \) and not \( \lambda \rightarrow \mu \). Hence, \( \mu \not\rightarrow \lambda \) iff (3.7) with proper subset containment of the Bernoulli complements, or (3.8) holds. Clearly, the relation \( \not\rightarrow \) is transitive and irreflexive.

**Example 4. Strict reduction using (3.7).**

\( \mu := 332211 \not\rightarrow \lambda := 221111 \).

\[
A^{(332211)} = \begin{pmatrix}
+ & + & + & + \\
+ & - & - & - \\
- & + & + & - \\
- & - & + & + \\
- & - & - & +
\end{pmatrix} \subset \begin{pmatrix}
+ & + & + & + & + \\
+ & - & - & - & - \\
- & + & + & - & - \\
- & - & - & + & - \\
- & - & - & + & +
\end{pmatrix} = A^{(221111)}
\]

Upon visual inspection, it is easily seen that each column of \( 332211 \perp B \) appears as a column in \( 221111 \perp B \).

**Example 5. Strict reduction using (3.8).**

The partition \( 211 \) reduces to the partition \( 11 \).

\[
A^{(211)} = \begin{pmatrix}
+ \\
-
\end{pmatrix}, \quad A^{(11)} = \begin{pmatrix}
+
\end{pmatrix}.
\]
Take $I = \{1, 2\}$, so that projection onto $I$ “forgets” the third coordinate in $211^{\perp B}$. We then have $\text{Proj}_I 211^{\perp B} = 11^{\perp B}$, and $211 \not\rightarrow 11$.

**Example 6. The consequence of not implying 11.**

If $\lambda$ does not imply 11, then every two rows of $A^{(\lambda)}$ are linearly independent. Thus, for every $i \neq j$, both $\lambda_i + \lambda_j$ and $\lambda_i - \lambda_j$ are expressible as a plus-minus combination of the remaining parts. Proposition 5 uses this in order to prove that $21111$ is the only novel partition of size 5.

There is a natural description of this principle in terms of coin weighing problems (see for example [1]). You have $k$ coins of various positive integer weights. Not implying 11 means that if an adversary selects any two coins and places them on the same or opposite sides of a balance scale, you can place all of the remaining coins on the scale so that it balances.

We now come to the definition that effectively governs explicit expansions such as those in Conjecture 1 and Theorem 1.

**Definition 8. Novel partitions.**

We call an integer partition $\lambda$ a novel partition if and only if there does not exist any other partition $\lambda'$ with $\lambda \not\rightarrow \lambda'$, and among all partitions equivalent to $\lambda$, in the sense of Definition 5, $\lambda$ is lexicographically first.

**Theorem 2** (Sufficiency of the set of novel partitions). The set of all novel partitions is sufficient, acting as possible left null vectors, to detect singularity for Bernoulli matrices $M$. That is, if such a matrix is singular, say of size $n$ by $n$, then there exists a novel partition $\lambda$ with $\text{len}(\lambda) \leq n$, and $v \in V^{(n)}_\lambda$ with $vM = 0$. 

22
Proof. If $M$ is singular, then there is a nonzero vector $w \in \mathbb{Z}^n$ with $wM = 0$. Taking absolute values of the coordinates, deleting zeros if they occur, and listing in nonincreasing order yields an integer partition $\lambda$, and $w \in V_{\lambda}^{(n)}$. If $\lambda$ is novel, we are done. If $\lambda$ is not novel, then it must reduce to a novel partition $\mu$, and then Remark 2 applies.

Theorem 3. Intrinsic characterization of novel partitions.

An integer partition $\lambda$ with $k$ parts is novel iff the matrix $A^{(\lambda)}$, specified in Definition 4, has rank $k - 1$, and $\text{gcd}(\lambda) = 1$.

Proof. Let $A \equiv A^{(\lambda)}$, with rows $r_1, \ldots, r_k$. To prove the only if direction, suppose rank $A < k - 1$. Then there exists $j < k$, and integers $c_1, \ldots, c_j \neq 0$, $\pi_1, \ldots, \pi_j$ distinct elements of $\{1, \ldots, k\}$, such that $c_1 r_{\pi_1} + \ldots + c_j r_{\pi_j} = 0$. Letting $v = (c_1, \ldots, c_j)$, we have $v \in V_{\mu}$, $\text{len}(\mu) = j < k$, and $\lambda \not\leftrightarrow \mu$, so that $\lambda$ is not novel.

If $\text{gcd}(\lambda) > 1$, then $\mu = \frac{1}{\text{gcd}(\lambda)} \lambda$, $\mu \leftrightarrow \lambda$, $\mu$ is earlier in lexicographic order, so $\lambda$ is not novel.

In the other direction, suppose rank $A = k - 1$, $\text{gcd}(\lambda) = 1$, but assume $\lambda$ is not novel. Then either

1. $\lambda \not\leftrightarrow \mu$, $\text{len}(\mu) < \text{len}(\lambda)$, but then there exists $v \in V_{\mu}^{(k)}$, with $k - 1$ or fewer nonzero components such that $vA = 0$, which implies rank $A < k - 1$;

2. $\lambda \not\leftrightarrow \mu$, $\text{len}(\mu) = \text{len}(\lambda)$, $\lambda \lessdot B \subseteq \mu \lessdot B$. Then $A^{(\mu)}$ and $A^{(\lambda)}$ both have rank $k - 1$, with $\mu A^{(\mu)} = 0$ and $\lambda A^{(\lambda)} = 0$. By the inclusion, we also have $\mu A^{(\lambda)} = 0$. But then if we consider the vector $v = \mu_1 \lambda - \lambda_1 \mu$ of length $k$ with first coordinate 0, $v$ has at most $k - 1$ nonzero entries, and $vA^{(\lambda)} = 0$. Since we assumed $A^{(\lambda)}$ has rank $k - 1$, we conclude $v = 0$.  

23
Corollary 1. An integer partition \( \lambda \) is either

1. novel (or a multiple of a novel),

2. implies a novel partition \( \mu \) of strictly smaller length, or

3. is not fairly divisible, i.e., \( \lambda^{\perp_B} = \emptyset \).

The only part that is not trivial is (2). We already showed that partitions like 332211 can strictly reduce to a partition of the same length, but without Theorem 3 it is not a priori obvious that there will always be a strict reduction in the length of the partition.

Theorem 4 (Minimality of the set of novel partitions). The family of novel partitions is minimal, in the sense that if any single one of the \( \lambda \)s in that set is removed, then the family, acting as possible left null vectors, does not detect all singularities.

Proof. For any \( v \in V_\lambda \), let \( 2p = |v^{\perp_B}| \); we can write \( v^{\perp_B} = \{x_1, x_2, \ldots, x_{2p}\} \), where \( x_i \in \{-1, 1\}^k \), \( x_i \neq x_j \), \( i \neq j \). As in Example 1, let \( A \equiv A(\lambda) \) denote the \( k \) by \( p \) matrix of rank \( k - 1 \) with columns given by \( x_1, x_2, \ldots, x_p \), the vectors with first entry positive.

1. If \( p < k \), then add \( k - p \) columns which are duplicates of column \( p \), and call this square matrix \( M \).

2. Else, if \( p > k \), since the rank of \( A \) is \( k - 1 \), the first \( k - 1 \) rows, denoted \( r_1, r_2, \ldots, r_{k-1} \), form an independent set in \( \mathbb{R}^p \). The existence of an independent set of \( p \) vectors in \( \mathbb{R}^p \), whose entries consist of plus and minus 1, is guaranteed by the existence of nonsingular Bernoulli matrices of all sizes; denote such a set as \( \{s_1, \ldots, s_p\} \). By the
basis extension theorem, re-indexing the $s_i$ as needed, there is an independent set of the form $\{r_1, r_2, \ldots, r_{k-1}, s_k, s_{k+1}, \ldots, s_p\}$. Replacing $s_k$ with $r_k$, the rows $r_1, r_2, \ldots, r_{k-1}, r_k, s_{k+1}, \ldots, s_p$ form a $p$ by $p$ Bernoulli matrix $M$ with rank $p - 1$.

3. Else, if $p = k$, then set $M = A$.

$M$ is thus a singular square matrix with entries either plus or minus 1. Denote the dimension of $M$ as $n$ by $n$; it has rank $n - 1$, and if $w \in V_{\mu}^{(n)}$ for some novel $\mu$ and $wM = 0$, then $w^\perp \supset v^\perp B$, which implies $w = v$.

**Remark 3.** In Theorem 4, it was necessary to specify testing for null vectors on one particular side, since in the case $n = 4$ any instance of singularity caused by 1111 in one direction carries with it a 11 in the other direction. In this case one could say that 1111 is not really necessarily to detect singularity.

We have just defined and characterized novel partitions, which form the foundation for the expansion in Theorem 1. The next set of theorems bounds the exponential decay from each term.

**Proposition 1** (Erdős, Littlewood, Offord [16]). Let $x_1, x_2, \ldots$ be real numbers, $|x_i| \geq 1$, and $\epsilon_1, \epsilon_2, \ldots$ be +1 or −1. Then the number of sums of the form $\sum_{i=1}^{k} x_i \epsilon_i$ which fall into an arbitrary open interval $I$ of length 2 does not exceed $\left(\frac{k}{\lfloor k/2 \rfloor}\right)$.

Taking $I = (-1, 1)$, an immediate consequence is that for any integer partition $\lambda$ with $k$ parts,

$$2^k r_{\lambda} = |\lambda^{\perp B}| \leq \left(\frac{k}{\lfloor k/2 \rfloor}\right),$$  \hspace{1cm} (3.9)
and in case \( k = 2m \) is even, the novel partition \( \lambda = 1^{2m} \) achieves equality with this upper bound.

A related theorem of Erdős [16] expands Proposition 1 by widening the target interval.

**Proposition 2** (Erdős [16]). Let \( r \) be any integer, the \( x_i \) real, \(|x_i| \geq 1\). Then the number of sums \( \sum_{i=1}^{k} \epsilon_i x_i \) which fall into the interior of any interval of length \( 2r \) is not greater than the sum of the \( r \) greatest binomial coefficients belonging to \( k \).

This proposition was proved by showing that the size of the union of \( r \) disjoint antichains in \( \{-1,1\}^k \) is at most the sum of the \( r \) largest binomial coefficients for \( k \); see [49], Proposition 7.7, and [9], Section 3, Exercise 7. As a corollary of this, we get

**Theorem 5.** Suppose \( \lambda \) is an integer partition with \( k \) parts, not all equal. Then \( 2^k r_\lambda = |\lambda^\perp| \) is at most the sum of the largest four binomial coefficients of \( k - 2 \). Hence, for \( k \geq 2 \) and even, \( \lambda = 1^k \) has \( |\lambda^\perp| > |\mu^\perp| \) for any partition \( \mu \) with \( k \) parts, not all equal, and for \( k \geq 5 \) and odd, \( \lambda = 2^1 1^{k-1} \) has \( |\lambda^\perp| \geq |\mu^\perp| \), for any partition \( \mu \) with \( k \) parts.

**Proof.** Fix \( i, j \) such that \( \lambda_i \neq \lambda_j \). Partition the set \( \lambda^\perp \) into four (possibly empty) subsets

\[
A = \{ x \in \lambda^{\perp} : x_i = 1, x_j = 1 \}, \\
B = \{ x \in \lambda^{\perp} : x_i = -1, x_j = -1 \}, \\
C = \{ x \in \lambda^{\perp} : x_i = 1, x_j = -1 \}, \\
D = \{ x \in \lambda^{\perp} : x_i = -1, x_j = 1 \}.
\]
These are disjoint antichains, since they specify four distinct target values for the sums $\sum' x_\ell \lambda_\ell$, where the sum is over the $k - 2$ indices other than $i, j$, and each $\lambda_\ell$ is strictly positive. Projecting out the two coordinates indexed by $i$ and $j$, we get 4 disjoint antichains in $\{-1, +1\}^{k-2}$.

\[\square\]


For all partitions $\lambda$ with exactly $k$ parts, with greatest common divisor 1, if $k \geq 4$ is even, the second largest probability $r_\lambda$ is achieved, uniquely, by $2^{2k-2}$, while if $k \geq 7$ is odd, the largest probability is achieved by $2^{1k-1}$ (already proved, as part of Proposition 5), and the second largest is achieved, uniquely, by $2^{3k-3}$.

We note that for $k \geq 5$ odd, it is trivial to check that $2^{1k-1}$ strictly beats $2^{3k-3}$, and with $k = 5$, $|22211^\perp B| = |32111^\perp B| = 6$.

Proposition 3. There are no novel partitions of size three.

Proof. By Theorem 3, any novel partition $\lambda$ with three parts must have rank $A^{(\lambda)} = 2$. Since 111 is not a valid template, the parts in $\lambda$ are not all equal, and so by Theorem 5, $|\lambda^\perp B| \leq 2$, which means that $A^{(\lambda)}$ has at most 1 column, and hence rank at most 1. \[\square\]

Proposition 4. The only novel partition of size 4 is 1111.

Proof. By Theorem 3, any novel partition $\lambda$ with four parts must have rank $A^{(\lambda)} = 3$. By Theorem 5, any novel partition $\lambda$ with four parts, not all equal, has $|\lambda^\perp B| \leq 4$, which means that $A^{(\lambda)}$ has at most 2 columns, and hence rank at most 2. If all parts of the partition are equal, then the requirement of $gcd(\lambda)$ equal 1 forces $\lambda = 1111$. This is indeed novel, with $A^{(1111)}$ given in Example 1. \[\square\]
Proposition 5. The only novel partition of size 5 is 21111.

Proof. Without loss of generality, assume \( \lambda = (a, b, c, d, e) \), where \( a \geq b \geq c \geq d \geq e > 0 \). As described in Example 6, in order to avoid implying 11, every pair of parts in \( \lambda \), when added or subtracted, must be a signed combination of the others, e.g.,

\[
\begin{align*}
  a + b &= \pm c \pm d \pm e \\
  b + c &= \pm a \pm d \pm e.
\end{align*}
\]

Let us look at the first equation. If any of the signs are negative, then monotonicity is necessarily broken. Thus, any novel partition of length five must have \( a + b = c + d + e \).

Similarly, we can look at \( b + c = \pm a \pm d \pm e \), and by a monotonicity argument we conclude that the only viable form is \( b + c = a \pm d \pm e \). We will look at each of these four cases separately.

1. 

\[
\begin{align*}
  a + b &= c + d + e \\
  b + c &= a + d + e
\end{align*}
\]

can be refined (by adding or subtracting one from the other) to \( b = d + e \) and \( a = c \). By monotonicity this means that \( a = b = c \), and hence our partition would be of the form \((d + e, d + e, d + e, d, e)\). However, we must have a solution to \( d - e = \pm (d + e) \pm (d + e) \pm (d + e) \), which would imply that \( e = 0 \).
2.

\[ a + b = c + d + e \]
\[ b + c = a + d - e \]

can be refined similarly to \( b = d \) and \( a = c + e \), which yields partitions of the form \((c + e, c, c, c, e)\). We must have a solution to \( c + 2e = \pm c \pm c \pm c \), which, to avoid implying \( e \leq 0 \), implies \( e = c \), and our template reduces to a multiple of 21111.

3.

\[ a + b = c + d + e \]
\[ b + c = a - d + e \]

can be refined to \( b = e \), \( a = c + d \), hence a multiple of 21111.

4.

\[ a + b = c + d + e \]
\[ b + c = a - d - e \]

forces \( b = 0 \).
Table 3.1: Novel partitions sorted by $r_\lambda$. Conjectured to be complete with respect to $r_\lambda \geq 38/256$. 

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\text{cen}(\lambda)$</th>
<th>$r_\lambda$</th>
<th>$256 \cdot r_\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1)</td>
<td>2</td>
<td>1/2</td>
<td>128.</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3/8</td>
<td>96.</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5/12</td>
<td>80.</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>35/128</td>
<td>70.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5/12</td>
<td>40.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>63/256</td>
<td>63.</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>15/64</td>
<td>60.</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>23/1024</td>
<td>57.75</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7/32</td>
<td>56.</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>7/32</td>
<td>56.</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>429/2048</td>
<td>53.625</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>105/512</td>
<td>52.5</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>13/64</td>
<td>52.</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>6435/32768</td>
<td>50.2734</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>99/512</td>
<td>49.5</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>49/256</td>
<td>49.</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3/16</td>
<td>48.</td>
</tr>
<tr>
<td>(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)</td>
<td>18</td>
<td>12155/65536</td>
<td>47.4805</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>3003/16384</td>
<td>46.9219</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>93/512</td>
<td>46.5</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>23/128</td>
<td>46.</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>46189/202144</td>
<td>45.1064</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>715/4096</td>
<td>44.6875</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>1419/8192</td>
<td>44.3438</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>11/64</td>
<td>44.</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>11/64</td>
<td>44.</td>
</tr>
<tr>
<td>(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)</td>
<td>22</td>
<td>88179/524288</td>
<td>43.0562</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>21879/131072</td>
<td>42.7324</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2717/16384</td>
<td>42.4531</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>675/4096</td>
<td>42.1875</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>21/128</td>
<td>42.</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>21/128</td>
<td>42.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>21/128</td>
<td>42.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>21/128</td>
<td>42.</td>
</tr>
<tr>
<td>(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)</td>
<td>24</td>
<td>676039/4194304</td>
<td>41.2623</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>20995/131072</td>
<td>41.0059</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>10439/65536</td>
<td>40.7773</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>81/512</td>
<td>40.5</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>323/2048</td>
<td>40.375</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>1287/8192</td>
<td>40.2188</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5/32</td>
<td>40.</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5/32</td>
<td>40.</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>5/32</td>
<td>40.</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>5/32</td>
<td>40.</td>
</tr>
<tr>
<td>(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)</td>
<td>26</td>
<td>1300875/8388608</td>
<td>39.6751</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>3232071/2097152</td>
<td>39.2681</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20111/131072</td>
<td>39.2793</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>627/4096</td>
<td>39.1875</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>5059/32768</td>
<td>39.1016</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>39/256</td>
<td>39.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>39/256</td>
<td>39.</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>623/4096</td>
<td>38.9375</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>155/1024</td>
<td>38.75</td>
</tr>
<tr>
<td>(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)</td>
<td>28</td>
<td>504575/33554432</td>
<td>38.2582</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>156009/1048576</td>
<td>38.0881</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>19/128</td>
<td>38.</td>
</tr>
</tbody>
</table>
Proposition 6. The only novel partitions of length 6 are 111111, 221111, 311111, 322111. The only novel partitions of length 7 are

\begin{align*}
2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 2 & 1 & 1 & 1 & 1 \\
3 & 2 & 1 & 1 & 1 & 1 & 1 \\
3 & 2 & 2 & 2 & 1 & 1 & 1 \\
3 & 3 & 2 & 1 & 1 & 1 & 1 \\
3 & 3 & 2 & 2 & 2 & 1 & 1 \\
4 & 1 & 1 & 1 & 1 & 1 & 1 \\
4 & 2 & 2 & 1 & 1 & 1 & 1 \\
4 & 3 & 2 & 2 & 1 & 1 & 1 \\
4 & 3 & 3 & 1 & 1 & 1 & 1 \\
4 & 3 & 3 & 2 & 2 & 1 & 1 \\
5 & 2 & 2 & 2 & 1 & 1 & 1 \\
5 & 3 & 3 & 2 & 1 & 1 & 1 \\
5 & 4 & 3 & 2 & 2 & 1 & 1 \\
\end{align*}

Proof. The same technique that was used in Proposition 5 can be continued for novel partitions of length 6, 7, etc., eliminating cases that imply 11. Mathematica [35] code was written to list all cases and reduce them. For the sake of economy in running time, we only considered the requirement that all four of $\lambda_1 \pm \lambda_2$ and $\lambda_{k-1} \pm \lambda_k$ be expressible plus-minus combination of the other $k - 2$ parts. When the reduction yields a space of dimension greater than one, the result may be viewed as what we call a meta-template,
e.g., \((a + b, a + b, b, b, a, a)\). The list of meta-templates includes all novel partitions and possibly others that are not novel. For \(k = 6\), the following candidates were returned:

\[
111111, 221111, 311111, 322111, 332211, 433211, 533221.
\]

We showed in Example 4 that \(332211 \not\leftrightarrow 221111\), and the ranks of \(A^{(433211)}\) and \(A^{(533221)}\) are 4, whereas the others have rank 5.

For \(k = 7\), a list of 14 templates was found; all of which turned out to be novel. Also, for \(k = 7\), 12 meta-templates were found; by hand inspection, 11 were easily shown to violate the monotonicity requirement that \(\lambda_1 \geq \lambda_2 \geq \cdots\). The remaining meta-template, \((a, b, a - b, d, e, h, h - d - e)\) is seen, by hand, to imply 11, since after the initial refinement to the form above one can apply the same technique again to the smallest two parts and reduce each case to either a monotonicity or positivity violation.

\[
\square
\]

**Lemma 3.** In order of decreasing \(r_{\lambda}\), the first eight novel partitions \(\lambda\) are

\[
11, 1111, 1^{6}, 1^{8}, 21111, 1^{10}, 21^{6}, 1^{12}.
\]

For novel partitions other than these eight, writing \(k\) for the number of parts

\[
\begin{align*}
k = 6, 7 & : \quad r_{\lambda} \leq \frac{60}{256}, \\
k = 8, 9 & : \quad r_{\lambda} \leq \frac{56}{256}, \\
k = 10, 11 & : \quad r_{\lambda} \leq \frac{52.5}{256}, \\
not 1^{14}, 1^{16}, \text{ and } k \geq 12 & : \quad r_{\lambda} \leq \frac{49.5}{256}.
\end{align*}
\]
Table 3.2: Novel partitions of length 8, conjectured to be the complete list.
Hence, aside from the first eight novel partitions, all other novel partitions have $r_{\lambda} \leq 56/256$. Observe that $\lambda = 1^{14}$ has $r_{\lambda} = 53.625/256$ and $\lambda = 1^{16}$ has $r_{\lambda} = 50.2734375/256$.

**Proof.** This follows immediately from Theorem 5. \qed

**Conjecture 4.** In order of decreasing $r_{\lambda}$, the novel partitions with $r_{\lambda} \geq 38/256$ are precisely those given in Table 3.3.

**Example 7.** The shortest novel arithmetic progression.

The partition $\lambda = (8, 7, 6, 5, 4, 3, 2, 1)$ is novel. It has $|\lambda^{\perp B}| = 14$, so $A^{(\lambda)}$ is an 8 by 7 matrix of rank 7. Examination of the $21 = 1 + 0 + 1 + 1 + 4 + 14$ novel partitions of lengths 2, 3, 4, 5, 6, 7 in Propositions 4 – 6 shows that this $\lambda$ is the shortest novel partition which is also an arithmetic progression.

**Conjecture 5.** There are exactly 122 novel partitions of length $k = 8$.

The list of 122 is given in Table 3.2. Our evidence in favor of this conjecture is that these 122, and no others, were found by a random survey, using Mathematica, of 420 million singular $n$ by $n$ matrices $M$, for $n = 8$. Of course, this is not a proof. For an exhaustive search, to guarantee that all novel partitions of length 8 have been found, one might observe that, with respect to the integer partitions underlying potential right and left null vectors, $M$ can be taken to have first row and first column all +1, so that it would suffice to examine $2^{49}$ matrices $M$.

**Remark 4.** The Mathematica command `NullSpace` applied to a singular $n$ by $n$ Bernoulli matrix $M$ returns a list of length $n$ vectors that forms a basis for the null space of $M$. Aside from the sign requirement in the first nonzero entry, these vectors have
always been of the form \( v \in V_\lambda^{(n)} \) for some novel partition \( \lambda \). One would like to prove a result about this, but since the basis returned by a generic null space algorithm is not unique, and hence implementation dependent, we will not pursue this idea further.

### 3.4 Polynomial coefficients arising from inclusion-exclusion

For events \( \{A_\alpha\}_{\alpha \in I} \), for a finite index set \( I \), and \( A = \cup_{\alpha \in I} A_\alpha \), the inclusion-exclusion formula states that

\[
P(A) = \sum_{\alpha \in I} P(A_\alpha) - \sum_{\{\alpha, \beta\} \subset I, \alpha \neq \beta} P(A_\alpha \cap A_\beta) + \sum P(A_\alpha \cap A_\beta \cap A_\gamma) + \cdots + (-1)^{|I|-1} P(\cap_{\alpha \in I} A_\alpha).
\] (3.10)

With \( W = \sum_{\alpha \in I} 1(A_\alpha) \), a sum of indicators of the events, the formula above may be expressed as

\[
P(A) = E W - E \left( W \binom{2}{2} \right) + E \left( W \binom{W}{3} \right) + \cdots + (-1)^{|I|-1} E \left( W \binom{W}{|I|} \right).
\]

The Bonferroni inequalities state that for events \( \{A_\alpha\}_{\alpha \in I} \), for a finite index set \( I \), and \( A = \cup_{\alpha \in I} A_\alpha \),

\[
P(A) \leq \sum_{\alpha \in I} P(A_\alpha)
\]

\[
P(A) \geq \sum_{\alpha \in I} P(A_\alpha) - \sum_{\{\alpha, \beta\} \subset I, \alpha \neq \beta} P(A_\alpha \cap A_\beta) + \cdots
\] (3.11)
Equation 3.11 is a lower bound, with the ... representing higher order bounds. A
variation of (3.11), with $B = \bigcup_{\beta \in I'} A_\beta$,

$$\mathbb{P}(A \setminus B) \geq \sum_{\alpha \in I} \mathbb{P}(A_\alpha) - \sum_{\{\alpha, \beta\} \subseteq I, \alpha \neq \beta} \mathbb{P}(A_\alpha \cap A_\beta) - \sum_{\alpha \in I, \beta \in I'} \mathbb{P}(A_\alpha \cap A_\beta),$$

(3.12)
is proved similarly.

For each integer partition $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$, let $D_\lambda$ denote the event that there
exists some set of indices $i_1, i_2, \ldots, i_k$, all distinct, such that $\lambda_1 e_{i_1} \pm \lambda_2 e_{i_2} \pm \ldots \pm \lambda_k e_{i_k}$ is
a null vector. Similarly, let $R_\lambda$ and $L_\lambda$ denote the sub event of $D_\lambda$ that the null vector is
a right or left null vector.

For example $D_{11}$ is the event that for some $i \neq j$, $e_i + e_j$ or $e_i - e_j$ is a left or right null
vector. We take $I = \binom{[n]}{2} \times \{-, +\} \times \{L, R\}$, so that $\alpha \in I$ specifies a set of two distinct
indices along with sign and direction bits. The event $A_\alpha$ corresponds to the occurrence
of a null vector of the form $\alpha$. For example, $\alpha = (\{2, 5\}, -, R) \in I$, and $A_\alpha$ is the event
that $e_2 - e_5$ is a right null vector.
Proposition 7. For $W = \sum_{\alpha \in I} 1(A_\alpha)$ with $I$ and the $A_\alpha$ as above, so that $D_{11} = \{W > 0\}$,

\[
\begin{align*}
\mathbb{E} W &= 4 \binom{n}{2} \left(\frac{1}{2}\right)^n, \\
\mathbb{E} \left(W^2\right) &= \left(12 \binom{n}{2}^2 - 4 \binom{n}{2}\right) \left(\frac{1}{4}\right)^n, \\
\mathbb{E} \left(W^3\right) &= 2^2 \binom{n}{3} \left(\frac{1}{4}\right)^n + \\
&\quad + 2^{3-3n} \left(\frac{13}{3} \binom{n}{2}^3 - 4 \binom{n}{2}^2 - 2 \binom{n}{2} - \frac{1}{3} \binom{n}{3} \binom{3}{2} - \binom{n}{3} \binom{n-2}{2}\right) \\
&= 4 \binom{n}{3} \left(\frac{1}{4}\right)^n + O(n^6 2^{-3n}).
\end{align*}
\]

Proof. Let $t = 2 \binom{n}{2} 2^{-n}$. Clearly,

\[
\mathbb{E} W = \sum_{\alpha \in I} P(e_i = \pm e_j) = |I| 2^{-n} = 4 \binom{n}{2} 2^{-n} = 2t.
\]

Let $I_R \subset I$ denote the subsets of $I$ with last coordinate $R$, with $I_L$ defined similarly. Then

\[
\mathbb{E} \left(W^2\right) = 2 \sum_{\alpha, \beta \in I_R} \mathbb{E} I_\alpha I_\beta + \sum_{\alpha \in I_R, \beta \in I_L} \mathbb{E} I_\alpha I_\beta =: G_1 + G_2.
\]

The first sum covers two null vectors on the same side, the second sum when they are on opposite sides. Focusing just on the sums, we have

\[
\begin{align*}
G_1 &= t^2 - 2^2 \binom{n}{2} 2^{-2n} = t^2 - t 2^{1-n}, \\
G_2 &= 2 \binom{n}{2} 2^{-n} \times 2 \binom{n}{2} 2^{-n+2} \times \frac{1}{2} = 2t^2.
\end{align*}
\]
The first equation (3.13) is found by considering all pairs on one side, and then taking away all pairs that share two rows along with any plus or minus combination. The second equation (3.14) is found by conditioning on one side first, say $\alpha \in I_R$, and then the other side picks up a boost from the conditioning on the values of four elements.

Finally, we have

$$6 \mathbb{E}\left( W^3 \right) = \sum_{(\alpha, \beta, \gamma) \text{ distinct}} 1(\alpha, \beta, \gamma) \mathbb{E} I_\alpha I_\beta I_\gamma =: F_1 + F_2$$

Here $F_1$ denotes the sum over all events where $\alpha, \beta, \gamma$ appear on the same side, and $F_2$ denotes the sum over events where two appear on one side, and one on the other. By choosing a side (left or right), we have for some $t_2, t_3, t_4$ functions of $n$,

$$\frac{F_1}{2} = t_3^3 - t_2 - t_3 - t_4 + 2^2 \binom{3}{2} \binom{n}{3} \left( \frac{1}{2} \right)^{2n}.$$  

The $t_3^3$ considers all triplets, and the $t_i$ considers the triplets that are supported on $i$ rows, $i = 2, 3, 4$, that need to be excepted. We have

$$t_2 = \binom{n}{2} 2^3 2^{-3n},$$

which chooses any two rows and all sign combinations. When three rows are supported, there are precisely $2^3 \binom{3}{1}^3$ combinations, but events of the form $\{e_i \pm e_j, e_i \pm e_k, e_j \pm e_k\}$ are null vectors} are sometimes valid. When they are valid, they have a probability of $(1/2)^{2n}$, hence excepting all events involving three rows we have
\[ t_3 = \binom{n}{3} 2^3 \binom{3}{1} 2^{-3n}, \]

and the term at the end adds back in the valid combinations supporting three rows. These events are of the form \( \{e_i \pm e_j \text{ and } e_i \pm e_k \text{ are null vectors}\} \), which imply one of \( e_j + e_k \) or \( e_j - e_k \) is a null vector as well. Finally, when four rows are supported, the exceptional cases are those in which two of \( \alpha, \beta, \gamma \) share two rows, and one does not share any, thus

\[ t_4 = \binom{3}{1} \binom{n}{2} \binom{n - 2}{2} 2^3 2^{-3n}. \]

For \( F_2 \), there are two choices for which side the solo index appears, and then three choices for which of \( \alpha, \beta, \gamma \) is this solo index. We have

\[ \frac{F_2}{6} = 4t_3 - \binom{n}{2} 2^2 \times \binom{n}{2} 2 \times 2^{2-3n}. \]

The factor of four comes from \( \mathbb{E} I_\alpha I_\beta I_\gamma = 4 \mathbb{E} I_\alpha \mathbb{E} I_\beta \mathbb{E} I_\gamma \), which is a boost from conditioning on an opposite side. The exceptional cases are those where the support of the non-solo pair lie on the same two rows, and includes all sign combinations. The solo index can be anything, and gets a conditional boost from being on the other side. \qed
Proposition 8. Recall that \( R_\lambda \) (and \( L_\lambda \), respectively) denotes the event that there is a right (left) null vector of template \( \lambda \). We have

\[
\mathbb{P}(R_{11} \setminus L_{11}) \geq 2 \left( \binom{n}{2} \right) \left( \frac{1}{2} \right)^n - \left( 12 \left( \binom{n}{2} \right)^2 - 4 \binom{n}{2} \right) \left( \frac{1}{4} \right)^n
\]

\[
\mathbb{P}(R_{11} \setminus L_{11}) = \mathbb{P}(R_{11}) - 8 \left( \binom{n}{2} \right)^2 \left( \frac{1}{4} \right)^n + O(n^6 2^{-3n}).
\]

Proof. The expansion follows along the same reasoning as Proposition 7 using (3.12); in particular, with \( t = 2 \binom{n}{2} 2^{-n} \) as before, we have

\[
\mathbb{P}(R_{11} \setminus L_{11}) \geq t - G_1 - G_2.
\]

A similar analysis can be undertaken for \( D_{1111} \); we omit the details.

Lemma 4.

\[
P(D_{11}) \geq 4 \left( \binom{n}{2} \right) \left( \frac{1}{2} \right)^n - \left( 12 \binom{n}{2}^2 - 4 \binom{n}{2} \right) \left( \frac{1}{4} \right)^n.
\]

\[
P(D_{11}) = 4 \left( \binom{n}{2} \right) \left( \frac{1}{2} \right)^n - \left( 12 \binom{n}{2}^2 - 4 \binom{n}{2} - 4 \binom{n}{3} \right) \left( \frac{1}{4} \right)^n + O(n^6 2^{-3n}),
\]

\[
P(D_{1111}) = 2^4 \left( \binom{n}{4} \right) (3/8)^n + O(n^5 (3/16)^n),
\]

\[
P(D_{16}) = 2^6 \left( \binom{n}{6} \right) \left( \frac{5}{16} \right)^n + O \left( n^7 \left( \frac{5}{32} \right)^n \right).
\]
\[
\begin{align*}
\mathbb{P}(D_{18}) &= 2^8 \left( \frac{n}{8} \right) \left( \frac{35}{128} \right)^n + O \left( n^9 \left( \frac{35}{256} \right)^n \right), \\
\mathbb{P}(D_{21111}) &= 2^5 \left( \frac{n}{5} \right) \left( \frac{1}{4} \right)^n + O \left( n^6 \left( \frac{1}{8} \right)^n \right), \\
\mathbb{P}(D_{110}) &= 2^{10} \left( \frac{n}{10} \right) \left( \frac{63}{256} \right)^n + O \left( n^{11} \left( \frac{63}{512} \right)^n \right), \\
\mathbb{P}(D_{215}) &= 2^7 \left( \frac{n}{7} \right) \left( \frac{60}{256} \right)^n + O \left( n^8 \left( \frac{60}{512} \right)^n \right), \\
\mathbb{P}(D_{12}) &= 2^{12} \left( \frac{n}{12} \right) \left( \frac{231}{1024} \right)^n + O \left( n^{13} \left( \frac{231}{2048} \right)^n \right).
\end{align*}
\]

**Proof.** The first two equations follow from Proposition 7. The rest are proved similarly, but we omit the details. \(\square\)

Next we move to probabilities \(\mathbb{P}(D_\lambda \cap D_\mu)\) for various choices of \(\lambda \neq \mu\). Observe that the events involved can be highly positively correlated; for example, with \(\lambda = 11, \mu = 1111\) we have \(\mathbb{P}(D_\lambda \cap D_\mu) / (\mathbb{P}(D_\lambda) \times \mathbb{P}(D_\mu))\) grows exponentially fast, as \((4/3)^n\).

**Proposition 9.** For two distinct novel partitions \(\lambda, \mu\), having \(j\) and \(k\) parts, respectively,

\[
\mathbb{P}(D_\lambda \cap D_\mu) \leq O \left( n^{k+j} \left( \max(r_\lambda, r_\mu)/2 \right)^n \right).
\]

The implicit constant in the big \(O\) varies with the choice of \(\lambda, \mu\).
Proof. Consider the event \( R_\lambda \cap R_\mu = \bigcup \{ vM = wM = 0 \} \), where the union is over \( v \in V_\lambda^{(n)}, w \in V_\mu^{(n)} \). The crucial ingredient is to show, with the notation of (3.5), that

\[
P( v \cdot X = w \cdot X = 0 ) \leq \max( r_\lambda, r_\mu ) / 2.
\] (3.15)

Without loss of generality, assume that the nonzero components of \( v \) are indexed by \( J \), so \( |J| = j \), and the nonzero components of \( w \) are indexed by \( K \), with \( |K| = k \). With \( I = J \cup K \) having size \( m = |I| \), the event of interest is based on \( m \) independent fair coins \( \epsilon_i, i \in I \), and can be expressed as

\[
\left\{ \sum_{i \in J} v_i \epsilon_i = 0 \right\} \cap \left\{ \sum_{i \in K} w_i \epsilon_i = 0 \right\}.
\] (3.16)

Case 1: \( J \neq K \). Without loss of generality, interchanging the \( \lambda \) and \( \mu \) if needed, \( I = \{1, 2, \ldots, m\} \) and \( m \in K \setminus J \). Condition on the values of the first \( m - 1 \) coins, with a configuration that satisfies \( \sum_{j \in J} v_j = 0 \). These configurations belong to the event \( vM = 0 \), and hence have probabilities summing to at most \( r_\lambda \). Each configuration, together with the requirement \( wM = 0 \), dictates the value needed for \( \epsilon_m \), which occurs with conditional probability \( 1/2 \). The possible exchange of \( \lambda, \mu \) at the start means that we have shown (3.15).

42
Case 2: $J = K$. Without loss of generality, rearranging the coordinates, and taking scalar multiples if needed, we can have $J = K = \{1, 2, \ldots, k\}$ and $a := v_k = w_k \neq 0$. The event in (3.16) simplifies to

$$\left\{ -a \epsilon_k = \sum_{i=1}^{k-1} v_i \epsilon_i = \sum_{i=1}^{k-1} w_i \epsilon_i \right\}.$$ 

From this we conclude

$$r_\lambda = \mathbb{P}(v \cdot X = 0) = \mathbb{P}\left( \sum_{i=1}^{k-1} v_i \epsilon_i \in \{\pm a\} \right) \geq 2 \mathbb{P}\left( \sum_{i=1}^{k-1} w_i \epsilon_i \in \{\pm a\} \right),$$

inequality arises since the second sum, with weights $w_i$, might not even be in $\{\pm a\}$, and the factor of 2 arises since when it is in the set, it dictates the choice of sign.

For the case where the potential null vectors are used on opposite sides, e.g., $L_\lambda \cap R_\mu$, we have

$$\mathbb{P}(L_\lambda \cap R_\mu) = O\left( \mathbb{P}(L_\lambda) \times \mathbb{P}(R_\mu) \right),$$

for the simple reason that conditioning of events of the form $Mw = 0$ with $w \in V^{(n)}_\mu$ only affects $k$ of the columns, giving the bound above, with constant $(1/r_\mu)^k$ as the implicit constant for the big O.

Proposition 9 involves a bound that can have exponential decay as large as $(1/4)^n$. For the sake of proving Theorem 1, with error term involving $(7/32)^n$, we need a stronger bound, as given below.
Lemma 5. For all novel partitions $\lambda, \mu$, having $j$ and $k$ parts, respectively, with $\lambda \neq \mu$, and neither partition equal to the partition 11, we have,

$$\mathbb{P}(D_{11} \cap D_{111}) = 2^3 \left( \frac{n}{4} \right)^2 \left( \frac{1}{4} \right)^n + O \left( n^5 \left( \frac{3}{16} \right)^n \right), \quad (3.17)$$

$$\mathbb{P}(D_{11} \cap D_\lambda) = O \left( n^{j+2} \left( \frac{3}{16} \right)^n \right), \quad (3.18)$$

$$\mathbb{P}(D_\lambda \cap D_\mu) = O \left( n^{k+j} \left( \frac{3}{16} \right)^n \right). \quad (3.19)$$

Proof. Equation (3.17) can be computed directly using inclusion exclusion, whereas Equations (3.18) and (3.19) use Proposition 9 with $\lambda = 1111$ since it is the most likely partition after 11.

Finally, we note a trivial lemma to simplify the coefficient for $4^{-n}$ in the expansion of Theorem 1,

Lemma 6.

$$\frac{1}{2} \left( \binom{n}{2} - \binom{n}{2} \right) = 3 \binom{n}{4} + 3 \binom{n}{3}. \quad (3.20)$$

Proof. Either simplify algebraically or note that the left hand side is the number of ways to choose any two unordered distinct pairs of unordered distinct pairs of $n$ objects. The right hand side counts the number of ways to select these pairs where all four indices are distinct and can be placed in 3 distinct configurations, and the second term counts the number of pairs that share a common index, of which there are 3 choices for the repeated index.
3.5 Interaction of left and right null vectors

Proposition 8 gives a lower bound on \( P(R_{11} \setminus L_{11}) = P(L_{11} \setminus R_{11}) \) which has, as a corollary,

\[
P(S \setminus L_{11}) \geq P(R_{11} \setminus L_{11}) \sim P(R_{11}) = P(L_{11}),
\]

and omitting the middle terms, and writing \( a_n \gtrsim c_n \) to mean that there exists \( b_n \) with \( a_n \geq b_n \) and \( b_n \sim c_n \), we have

\[
P(S \setminus L_{11}) \gtrsim P(L_{11}). \tag{3.21}
\]

Expressing (3.21) in terms of left null vectors, with the outer union on the left taken over all novel partitions \( \lambda \) of length less than or equal to \( n \), other than 11,

\[
P \left( \bigcup_{\lambda \neq 11} \bigcup_{v \in V^{(n)}_{\lambda}} \{vM = 0\} \right) \gtrsim P \left( \bigcup_{v \in V^{(n)}_{11}} \{vM = 0\} \right).
\]

Writing this with \( L_\lambda \) for the event that \( M \) has a left null vector of template \( \lambda \), the above display can be rewritten as

\[
\sum_{\lambda \neq 11} P(L_\lambda) \gtrsim P(L_{11}).
\]

We believe that to prove sharp upper bounds on \( P_n \), say as given by (3.1) or (3.3), it will be necessary to consider the effect of conditioning on \( D_{11}^c \). Propositions 10 and 11 might be a first step in this direction.
Proposition 10. Suppose that $\lambda$ is a novel partition of length $k$, with $k = n$. Let $2p = |\lambda^{\perp B}|$. Recall that $R_{11}$ is the event that our $n$ by $n$ matrix $M$ has a right null vector of the form $e_i \pm e_j$. For every $v \in V_\lambda$,

$$\frac{P(vM = 0 | R_{11}^c)}{P(vM = 0)} = \frac{(p)_n}{p^n}.$$

Proof. The hypothesis $k = n$ is essential: if $x$ denotes a column of $M$, then, thanks to $k = n$, we know that $x \in v^{\perp B}$. There are $p$ choices for the “direction” $\{-x, x\}$ with $x \in v^{\perp B}$, and different columns of $M$ must choose different directions, otherwise the event $R_{11}$ would occur. By giving the ratio of the conditional probability to the unconditional probability, factors of 2, for choosing between $x$ and $-x$, for each column, cancel. 

Proposition 11. Suppose that $\lambda$ is a novel partition of length $k$, with $k = n - 1$. Let $2p = |\lambda^{\perp B}|$. Recall that $R_{1111}$ is the event that our $n$ by $n$ matrix $M$ has a right null vector of the form $e_{j_1} \pm e_{j_2} \pm e_{j_3} \pm e_{j_4}$. For every $v \in V^{(n)}_\lambda$, as specified by Definition 2,

$$\frac{P(vM = 0 | (R_{11} \cup R_{1111})^c)}{P(vM = 0)} = \frac{(p)_n}{p^n} + \frac{1}{2p} \binom{n}{2} \frac{(p)_{n-1}}{p^{n-1}}.$$

Proof. The hypothesis $k = n - 1$ is essential. Without loss of generality, assume that $v_n = 0$, so $v = (w, 0)$ with $w \in V_\lambda$. Let $x = (y, s)$ denote a column of $M$, where $y$ gives the first $n - 1$ coordinates, and $s \in \{-1, +1\}$. Then, thanks to $k = n - 1$ and $v_n = 0$, we know that $y \in w^{\perp B}$. There are $p$ choices for the “direction” $\{-y, y\}$ — restricting to the first $n - 1$ coordinates, with $y \in w^{\perp B}$, and different columns of $M$ must choose different directions, apart from possibly one pair of columns, where the columns in a pair
may share the underlying $n - 1$ direction, but have opposite choices of $s$ for their $n$th coordinate. (If three columns share the underlying $n - 1$ direction, the event $R_{11}$ would occur; if two pairs of columns share, then event $R_{1111}$ would occur.)
Chapter 4

Probabilistic Divide-And-Conquer

\[ I, \text{ at any rate, am convinced that He does not throw dice.} \]

– Albert Einstein

\[ \text{God not only plays dice, He also sometimes} \]
\[ \text{throws the dice where they cannot be seen.} \]

– Stephen Hawking

4.1 Introduction.

4.1.1 Exact simulation

Exact simulation methods provide samples from a set of objects according to some given probability distribution. For many combinatorial problems, the given distribution is the uniform choice over all possibilities of a given size.
An important technique from von Neumann 1951, [50], which we review in Section 4.2.2, is acceptance/rejection sampling, where one samples from an easy-to-simulate distribution related to the desired distribution, and then rejects some samples. The precise recipe is to accept samples with probability proportional to the ratio of the desired probability to the probability under the easy-to-simulate distribution. The result is that upon acceptance, the sample is an exact sample from the desired distribution.

Another important technique is Markov chain coupling from the past (CFTP), where one keeps track of coalescing Markov chains starting from some time in the past, and constructs a long enough past that all chains have coalesced by time zero [44]. This is now a very lively subject, with over 160 papers described at

http://dimacs.rutgers.edu/~dbwilson/exact.html/

Divide-and-conquer is a basic strategy for algorithms. Notable examples include the Cooley-Tukey fast Fourier transform, attributable to Gauss [23], and Karatsuba’s fast multiplication algorithm [25], which surprised Kolmogorov, [26]. We note that these and other cases treated in textbooks on algorithms are deterministic. Randomized quicksort, see for example [12] Section 7.3, is the prototype of divide-and-conquer using randomness, but such algorithms can be thought of as a variation on the deterministic algorithm, applied to a permutation of the input data.

In this chapter, we propose a new method for exact sampling: probabilistic divide-and-conquer; here, the subdivision of the original problem is inherently random. We prove, in a couple of general settings, that this method does achieve exact samples.

Then we illustrate the use of this general method, with the target being to sample, for a given $n$, uniformly from the $p_n$ partitions $\lambda$ of the integer $n$. The starting point is
Fristedt’s construction, [17], with a random integer $T$ of size around $n$, such that, for a random partition of $T$, the counts of parts of various sizes are mutually independent; we review this in Section 4.4.2. The problem then is how to simulate efficiently, since the event $T = n$ is a rare event, as in [11].

In order of ontogeny these methods express a partition as $\lambda = (A, B)$, where

1. $A$ is the (list of) large parts, say $\sqrt{n}$ up to $n$, $B$ is the small parts, size 1 up to $\sqrt{n}$.

Mix-and-match may offer an additional speedup.

2. $B$ is the number of parts of size 1, $A$ is everything else. Hence the $B$ side of the simulation is trivial, with no calls to a random number generator. Nevertheless, there is a large speedup, by a factor asymptotic to $\sqrt{n}/c$, where $c = \pi/\sqrt{6}$.

3. In

$$p(z) = d(z)p(z^2),$$

with the classical $p(z) = \sum_{n \geq 0} p_n z^n = \prod_{i \geq 1} (1 - z^i)^{-1}$, and $d(z) = \prod_{i \geq 1} (1 + z^i)$ to enumerate partitions with distinct parts, $A$ corresponds to $d(z)$. This method iterates beautifully, reducing the target, $n$, by a factor of approximately 4 per iteration, with an acceptance/rejection cost of only roughly $2\sqrt{2}$, improved to $\sqrt{2}$ in Section 4.4.5.1. We have run this on a personal computer, with $n$ as large as $2^{49}$, and relative to the basic algorithm “waiting-to-get-lucky”, analyzed in Section 4.4.2, this version of divide-and-conquer achieves roughly a billion-fold speedup.\footnote{The RandomPartition function in Mathematica® [35] appears to hit the wall at around $n = 2^{30}$.}
4.1.2 Probabilistic divide-and-conquer, motivated by mix-and-match

For us, the random objects $S$ whose simulation might benefit from a divide-and-conquer approach are those that can be described as $S = (A, B)$, where there is some ability to simulate $A$ and $B$ separately. Specifically, we require that $A \in \mathcal{A}$, $B \in \mathcal{B}$, and that there is a function $h : \mathcal{A} \times \mathcal{B} \to \{0, 1\}$, so that for $a \in \mathcal{A}, b \in \mathcal{B}$, $h(a, b)$ is the indicator that “$a$ and $b$ fit together to form an acceptable $s$.” Furthermore, we require that $A$ and $B$ be independent, and that the desired $S$ be equal in distribution to $( (A, B) | h(A, B) = 1)$. This description, independent objects conditional on some event, may seem restrictive, but [7, 15] shows that very broad classes — combinatorial assemblies, multisets, and selections — fit into this setup.

Now imagine that one wants an honest sample of size $m$, that is, $S_1, S_2, \ldots, S_m$, to be independent, with the original $S$ along with $S_1, S_2, \ldots$ to be identically distributed. The pedestrian approach is to propose sample values $A_1, A_2, \ldots$ and $B_1, B_2, \ldots$, and to consider the indicators of aligned matching, that is, $h(A_1, B_1), h(A_2, B_2), \ldots$. One naturally has waiting times $T_1, T_2, \ldots$, with $T_1 := \min\{i \geq 1 : h(A_i, B_i) = 1\}$, and for $k > 1$, $T_k := \min\{i > T_{k-1} : h(A_i, B_i) = 1\}$. And of course, the $\ell$th sample found is $S_\ell := (A_{T_\ell}, B_{T_\ell})$.

In the case where $k = T_m$ is large, the scheme just described seems wasteful — we proposed $k$ values of $A \in \mathcal{A}$, and $k$ values of $B \in \mathcal{B}$, and hence might have $k^2$ opportunities for a match. That is, rather than just look for the aligned potential matches, scored by the $h(A_i, B_i)$ for $1 \leq i \leq k$, we might have considered the $h(A_i, B_j)$ for $1 \leq i, j \leq k$, with $k^2$ index pairs $(i, j)$. Indeed, this situation arises naturally in biological sequence matching,
[8, 6], where for two independent sequences of iid letters, in many but not all cases for the two marginal distributions, unrestricted rather than aligned matching effectively squares the number of ways to look for a match, and hence approximately doubles the length of the longest match found.

Of course, the difficulty with the general program “search all $k^2$ index pairs $ij$” is that conflicting matches might be found. Suppose, for example, that exactly two matches are found, with $A_i$ matching both $B_j$ and $B_{j'}$, with $j \neq j'$. It is easy to see that taking both $AB$ pairs ruins the iid nature of a sample. Also, though perhaps not as obvious, other strategies, such as suppressing both $AB$ pairs, or taking only the pair indexed by the lexicographically first of $ij, ij'$, or taking only one pair, based on an additional coin toss, introduce bias relative to the desired distribution for $S$.

There is a way to allow mix-and-match, and still get an honest sample\(^5\); details will be given in Section 4.3. The first step is to use rejection sampling to produce a list of $A$s, distributed as a sample from the distribution of $A$s biased by the chance that they would match, if a single $B$ were proposed. For us, it only became apparent later, that this use of rejection, even in the absence of mix-and-match, can be useful; Theorem 11 describes a surprising example.

\(^5\)under a condition on the structure of $h$, described by Lemma 8.
4.2 The basic lemma for exact sampling with divide-and-conquer

We assume throughout that

\[ A \in \mathcal{A}, \ B \in \mathcal{B} \text{ have given distributions,} \tag{4.1} \]

\[ A, B \text{ are independent,} \tag{4.2} \]

\[ h : \mathcal{A} \times \mathcal{B} \to \{0, 1\} \tag{4.3} \]

satisfies \( p := \mathbb{E} h(A, B) \in (0, 1], \tag{4.4} \)

where, of course, we also assume that \( h \) is measurable, and

\[ S \in \mathcal{A} \times \mathcal{B} \text{ has distribution } \mathcal{L}(S) = \mathcal{L}( (A, B) \mid h(A, B) = 1), \tag{4.4} \]

i.e., the law of \( S \) is the law of the independent pair \((A, B)\) conditional on having \( h(A, B) = 1 \).

Note that a restatement of (4.4), exploiting the hypothesis (4.3), is that for measurable sets \( R \subset \mathcal{A} \times \mathcal{B} \),

\[ \mathbb{P}(S \in R) = \frac{\mathbb{P}((A, B) \in R \text{ and } h(A, B) = 1)}{p}, \]

The requirement that \( p > 0 \) is not needed for divide-and-conquer to be useful; but rather, a choice we make for the sake of simpler exposition. In cases where \( p = 0 \), the conditional distribution, apparently specified by (4.4), needs further specification — this is known as Borel’s paradox.
or equivalently, for bounded measurable functions $g$ from $A \times B$ to the real numbers,

$$\mathbb{E} g(S) = \mathbb{E} (g(A, B)h(A, B))/p.$$  

Since we have assumed $p > 0$, this is elementary conditioning. This allows the distributions of $A$ and $B$ to be arbitrary: discrete, absolutely continuous, or otherwise.

The following lemma is a straightforward application of Bayes’ formula.

**Lemma 7.** Suppose $X$ is the random element of $A$ with distribution

$$L(X) = L(A \mid h(A, B) = 1),$$  
(4.5)

and $Y$ is the random element of $B$ with conditional distribution

$$L(Y \mid X = a) = L(B \mid h(a, B) = 1).$$  
(4.6)

Then $(X, Y) =^d S$, i.e. the pair $(X, Y)$ has the same distribution as $S$, given by (4.4).

**Proof.** A restatement of (4.5) is

$$\mathbb{P}(X \in da) = \frac{\mathbb{P}(A \in da) Eh(a, B)}{p},$$  
(4.7)

and relation (4.6) is equivalent to

$$L((X, Y) \mid X = a) = L((a, B) \mid h(a, B) = 1).$$  
(4.8)
Hence, for any bounded measurable $g : \mathcal{A} \times \mathcal{B} \to \mathbb{R}$,

$$
\mathbb{E} g(X, Y) = \mathbb{E} (\mathbb{E} (g(X, Y) | X)) \\
= \int_\mathcal{A} \mathbb{P}(X \in da) \mathbb{E} (g(X, Y) | X = a) \\
= \int_\mathcal{A} \frac{\mathbb{P}(A \in da) \mathbb{E} h(a, B) \mathbb{E} (g(a, B) h(a, B))}{\mathbb{E} h(a, B)} \\
= \frac{1}{p} \int_\mathcal{A} \mathbb{P}(A \in da) \mathbb{E} (g(a, B) h(a, B)) \\
= \mathbb{E} (\mathbb{E} (g(A, B) | h(A, B) = 1)) = \mathbb{E} g(S).
$$

We used (4.8) for the middle line in the display above; on the set $\mathcal{A}_0 := \{a \in \mathcal{A} : \mathbb{E} h(a, B) = 0\}$, which contributes 0 to the integral, we took the usual liberty of dividing by 0, rather than writing out separate expressions for the integrals over $\mathcal{A}_0$ and $\mathcal{A} \setminus \mathcal{A}_0$.

\square

4.2.1 Algorithmic implications of the basic lemma

Assume that one wants a sample of fixed size $m$ from the distribution of $S$. That is, one wants to carry out a simulation that provides $S_1, S_2, \ldots$, with $S_1, S_2, \ldots, S_m$ being mutually independent, with each equal in distribution to $S$. According to Lemma 7, this can be done by providing $m$ independent pairs $(X_i, Y_i)$, $i = 1$ to $m$, each equal in distribution to $S$.

A reasonable choice of how to carry this out, not yet using mix-and-match, involves the following:
Outline of an algorithm to gather sample of size $m$.

**Stage 1.** Sample $X_1, X_2, \ldots, X_m$ from the distribution of $X$, i.e., from

$$L(X) = L(A \mid h(A, B) = 1).$$

**Stage 2.** Conditional, on the result of stage 1 producing

$$(X_1, \ldots, X_m) = (a_1, \ldots, a_m), \text{ find } Y_1, Y_2, \ldots, Y_m, \text{ mutually independent, with distributions } L(Y_i) = L(B \mid h(a_i, B) = 1).$$

Note that in general, conditional on the result of stage 1, the $Y_i$ in stage 2 are *not* identically distributed. Furthermore, the trials undertaken to find these $Y_i$ need not be independent of each other.

### 4.2.2 Use of acceptance/rejection sampling

Assume that we know how to simulate $A$ — this is under the distribution in (4.1), where $A, B$ are independent. But, we need instead to sample from an alternate distribution, denoted above as that of $X \in A$. The rejection method recipe, for using (4.7), may be viewed as having 4 steps, as follows.

1. Find a threshold function $t : A \to [0, 1]$, with $t(a)$ proportional to $\mathbb{E} h(a, B)/p$, i.e.,

   $t$ of the form $t(a) = C \times \mathbb{E} h(a, B)/p$ for some positive constant $C$.

2. Propose iid samples $A_1, A_2, \ldots$,

3. Independently generate uniform $(0,1)$ random variables $U_1, U_2, \ldots$,

4. If $U_i \leq t(A_i)$, then accept $A_i$ as the $X$ value; otherwise reject it.
The cost of using acceptance/rejection.

Naturally, one wants to take the constant $C$ for the threshold function $t$ in step (1) to be as large as possible. This is subject to the constraint $t(a) \leq 1$ for all $a$, i.e., $C \times \mathbb{E} h(a, B)/p \leq 1$. The expected fraction of proposed samples $A_i$ to be accepted will be the average of $t(a)$ with respect to the distribution of $A$, i.e.,

$$p_{\text{acc}} := \mathbb{P}(U \leq t(A)) = \mathbb{E} t(A) = \mathbb{E} \left( C \times \frac{h(A, B)}{p} \right) = C,$$

and the expected number of proposals needed to get each acceptance is the reciprocal of this, so we define

$$\text{Acceptance cost} := \frac{1}{p_{\text{acc}}} = \frac{1}{C}. \quad (4.9)$$

Assuming that we can find an $a^*$ where $\mathbb{E} h(a, B)$ achieves its maximum value, this simplifies to

$$\text{Acceptance cost} = \frac{1}{C} = \frac{p}{\mathbb{E} h(a^*, B)}, \quad (4.10)$$

For comparison, if we were not using divide-and-conquer, but instead proposing pairs $(A_i, B_i)$ and hoping to get lucky, i.e., hoping that $h(A_i, B_i) = 1$, with success probability $p$ and expected number of proposals to get one success equal to $1/p$, then, ignoring the cost of proposing the $B_i$, the speedup involved in (4.10) is a factor of $1/\mathbb{E} h(a^*, B)$.

Is the threshold function $t$ computable? In step (4), for each proposed value $a = A_i$, we need to be able to compute $t(a)$; this can be either a minor cost, a major cost, or an absolute impediment, making probabilistic divide-and-conquer infeasible. All of these
variations occur, in the context of integer partitions, and will be discussed in Sections 4.4.3 – 4.4.5, and again in Section 4.5.2.

4.2.3 Caution: mix-and-match not yet enabled

In Stage 2 of the 2-stage algorithm described at the beginning of Section 4.2.1, there is a subtle issue involved in the phrase “independently find $Y_1, Y_2, \ldots, Y_m$”. Suppose, for example, that one plans to propose iid copies of $B$, say $B_1, B_2, \ldots$, waiting for samples that will match one of the $a_1, a_2, \ldots, a_m$ obtained in the first stage.

A correct way to carry out stage 2 is to consider stopping times $1 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_m$, with

$$\alpha_1 := \min\{j \geq 1: h(a_1, B_j) = 1\}, \quad (4.11)$$

and then recursively, for $\ell = 2$ to $m$, $\alpha_\ell := \min\{j \geq \alpha_{\ell-1}: h(a_\ell, B_j) = 1\}$. Here, stage 2 is completed after proposing $\alpha_m$ copies of $B$, and the $m$ samples of $S$ are $(a_\ell, B_{\alpha_\ell})$ for $\ell = 1$ to $m$.

In general, it would be incorrect to try to speed up stage 2 by taking stopping times $1 \leq \beta_1 < \beta_2 < \cdots < \beta_m$, with

$$\beta_1 := \min\{j \geq 1: h(a_i, B_j) = 1 \text{ for some } i \in \{1, 2, \ldots, m\}\}, \quad (4.12)$$

and then $\beta_2 := \min\{j \geq \beta_1: h(a_i, B_j) = 1 \text{ for one of the } m-1 \text{ values } i \text{ not already matched}\}$, and so on. [In case the $B$ at time $\beta_1$ matches $a_i$ for more than one index $i$, we might choose, for example, to declare the smallest available $i$ to serve as the index matched, for the sake of specifying which $m-1$ indices are available in the definition of $\beta_2$.]
Other choices as to which \( i \), even those involving auxiliary randomization, will have the same effect.] Say that \( I(\ell) \) is the index matched at time \( \beta_\ell \). With the random permutation \( \pi \) defined to be the inverse of the permutation \( I(\cdot) \), the sample consists of \((a_\ell, B_{\beta_\ell(\ell)})\) for \( \ell = 1 \) to \( m \). Not only there may be dependence in the sequence \( B_{\beta_n(1)}, B_{\beta_n(2)}, \ldots, B_{\beta_n(m)} \), it may also be the case\(^7\) that \( B_{\beta_n(\ell)} \) fails to have the desired marginal distribution, i.e. that of \( B \) conditional on \( h(a_\ell, B) = 1 \). (Perhaps this \( B \) was matched to \( a_\ell \) when a higher priority index \( i \) was available; the fact of not satisfying \( h(a_i, B) = 1 \) biases the distribution.)

### 4.3 Simple matching enables mix-and-match

Lemma 7 was basic, and so is the following lemma, but the pair serves nicely to clarify the logical structure of what is needed to enable probabilistic divide-and-conquer, versus what is needed to further enable mix-and-match.

**Lemma 8.** Given \( h : A \times B \to \{0,1\} \), the following two conditions are equivalent:

**Condition 1:** \( \forall a,a' \in A, b,b' \in B, \)

\[
1 = h(a,b) = h(a',b) = h(a,b') \text{ implies } h(a',b') = 1,  \tag{4.13}
\]

**Condition 2:** \( \exists C, \) and functions \( c_A : A \to C, c_B : B \to C, \) so that \( \forall (a,b) \in A \times B, \)

\[
h(a,b) = 1(c_A(a) = c_B(b)). \tag{4.14}
\]

\(^7\)We thank Sheldon Ross for first observing this.
We think of $\mathcal{C}$ as a set of colors, so that condition (4.14) says that $a$ and $b$ match if and only if they have the same color.

Proof. That (4.14) implies (4.13) is trivial. In the other direction, it is easy to check that (4.13) implies that the relation $\sim_A$ on $A$ given by $a \sim_A a'$ iff $\exists b \in B, 1 = h(a, b) = h(a', b)$ is an equivalence relation. Likewise for the relation $\sim_B$ on $B$ given by $b \sim_B b'$ iff $\exists a \in A, 1 = h(a, b) = h(a, b')$. For the set of colors, $\mathcal{C}$, we might take either the set of equivalence classes of $A$ modulo $\sim_A$, or the set of equivalence classes of $B$ modulo $\sim_B$, and then (4.13) also provides a bijection between these two sets of equivalence classes, to induce (4.14).

Remark 1. The proof of Lemma 8, with equivalence classes of $A/\sim_A$ and $B/\sim_B$, shows that the pair of coloring functions satisfying (4.14) is essentially unique. Specifically, unique apart from relabeling and padding, i.e., an arbitrary permutation on the names of the colors used, and enlarging the range, $\mathcal{C}$, to an arbitrary superset of the image.

Remark 2. The statement of Lemma 8 shows that coloring is not essentially an issue of sufficient statistics. After all, hypothesis (4.13) only concerns the logical structure of the matching function $h$ appearing in (4.3), and doesn’t involve the distributions on $A$ and $B$ appearing in (4.1).
Remark 3. When (4.14) holds, we can write the event that $A$ matches $B$ as a union indexed by the color involved:

$$\{h(A, B) = 1\} = \bigcup_{k \in \mathcal{C}} \{c_A(A) = k, c_B(B) = k\},$$

so that $p = \sum_{k \in \mathcal{C}} \mathbb{P}(c_A(A) = k, c_B(B) = k)$, and we see that at most a countable set of colors $k$ contribute a strictly positive amount to $p$. As a notational convenience, we take $\mathbb{N} \subset \mathcal{C}$, and use positive integers $k$ for the names of colors that have

$$\mathbb{P}(c_A(A) = k, c_B(B) = k) > 0. \quad (4.15)$$

[The fact that $A, B$ are independent, hence $\mathbb{P}(c_A(A) = k, c_B(B) = k) = \mathbb{P}(c_A(A) = k)\mathbb{P}(c_B(B) = k)$, is irrelevant to main idea behind (4.15). However, a technique we refer to as ‘roaming $x$’ in Section 4.4.3.2 is an example of how to take advantage of the independence of $A$ and $B$.]

The intent of the following lemma is to show that if $h$ satisfies (4.14), then mix-and-match strategies can can be used in stage 2 of the broad outline of Section 4.2.1.

Lemma 9. Assume that $h$ satisfies (4.14). Consider a procedure which proposes a sequence $D_1, D_2, \ldots$ of elements of $\mathcal{B}$ with the following properties:

There is a sequence of $\sigma$–algebras $\mathcal{F}_0 \subset \mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots$. [We think of $\mathcal{F}_0$ as carrying the information from stage 1 of an algorithm along the lines described in Section 4.2.1, carrying information such as “which demands $a_1, a_2, \ldots$ must be met — or reduced information, such as the colors $c_A(a_1), \ldots, c_A(a_m)$.]
For every \( n \geq 1 \) and \( k \) satisfying (4.15), conditional on \( F_{n-1} \) together with \( c_B(D_n) = k \), the distribution of \( D_n \) is equal to \( \mathcal{L}(B|c_B(B) = k) \).

For every \( k \) satisfying (4.15),

\[ \text{with probability 1, infinitely many } n \text{ have } c_B(D_n) = k. \tag{4.16} \]

Define stopping times \( \tau_{i}^{(k)} \), the “time of the \( i \)-th instance of \( c_B(D_n) = k \), by \( \tau_{0}^{(k)} = 0 \) and for \( i \geq 1 \), \( \tau_{i}^{(k)} = \inf\{n > \tau_{i-1}^{(k)} : c_B(D_n) = k\} \). We write \( D(n) \equiv D_n \), to avoid multi-level subscripting, and define \( B_{i}^{(k)} := D(\tau_{i}^{(k)}) \) for \( i = 1, 2, \ldots \).

Then, for each \( k \), the \( B_{1}^{(k)}, B_{2}^{(k)}, \ldots \) are independent, with the distribution \( \mathcal{L}(B|c_B(B) = k) \), and as \( k \) varies, these sequences are mutually independent.

**Proof.** The proof is a routine exercise; it suffices to check the independence claim for an arbitrary finite number of choices of \( k \), restricting to the \( B_{i}^{(k)} \) for \( 1 \leq i \leq i_0 \) with an arbitrary finite \( i_0 \), and this can be done, along with checking for the specified marginal distributions, by summing over all possible values for the random times \( \tau_{i}^{(k)} \). Writing out the full argument would be notationally messy, and not interesting. \( \square \)

### 4.4 Algorithms for simulating random partitions of \( n \)

In this section we focus on the generation of partitions under the uniform distribution. Other distributions on partitions such as the Plancherel measure, while important in many applications, for example [14, 18], are not addressed in the current paper, but it is highly plausible that these techniques can be adapted to those measures. We note that the phrase “generating partitions” is usually taken, as in [28], to mean the systematic
listing of all partitions, perhaps subject to additional constraints; this is very different from simulating a random instance.

The computational analysis that follows uses an informal adaptation of uniform costing; see Section 4.5.2. Some elements of the analysis, specifically asymptotics for the acceptance rate, will be given rigorously, for example in Theorems 10, 11, and 12.

4.4.1 For baseline comparison: table methods

A natural simulation method is to find the largest part first, then the second largest part, and so on according to a distribution of largest part\(^8\). The main cost associated with this method is the storage of all the distributions needed. Some details follow.

Let \(p(\leq k, n)\) denote the number of partitions of \(n\) with each part of size \(\leq k\), so that \(p(n, n) = p_n\). These can be quickly calculated from the recurrence

\[
p(\leq k, n) = p(\leq k - 1, n) + p(\leq k, n - k),
\]

where the right hand side counts the number of partitions without any \(k\)’s plus the number of partitions with at least one \(k\).

Let \(X_i\) denote the \(i\)-th largest part of a randomly generated partition, so \(\lambda = (X_1, X_2, \ldots)\). We have

\[
\mathbb{P}(X_1 \leq j) = p(\leq j, n)/p_n,
\]

\[
\mathbb{P}(X_2 \leq s | X_1 = j) = p(\leq s, n - j)/p(\leq j, n - j),
\]

\(^8\)For the largest part, we are dealing with unrestricted partitions of \(n\), but for subsequent parts, the problem involves the largest part of a partition of an integer \(m\) into parts of size at most \(k\).
and in general for $i \geq 2$

$$P(X_i \leq j_i | X_1 = j_1, \ldots, X_{i-1} = j_{i-1}) = \frac{p(\leq j_i, n - \sum j_i)/p(\leq j_{i-1}, n - \sum j_i)}{p(\leq j_i, n - \sum j_i)}/p(\leq j_{i-1}, n - \sum j_i).$$

Rather than computing each quantity on the right hand side as it appears, an $n$ by $n$ table, where the $j$-th column consists of the numbers $p(\leq 1, j), p(\leq 2, j), \ldots, p(\leq n, j)$, can be computed and stored, hence generating partitions is extremely fast once this table has been created. For one uniformly distributed random number, we can look up the value of the largest part of a random partition; this implies order of $\sqrt{n} \log n$ lookups. An easy variation also finds the multiplicity of the largest part, implying order of $\sqrt{n}$ lookups.

The memory conditions are on the order of $n^2$ for the entire table, a severe constraint, but once it has been constructed the generation of random partitions is rapid. Another variation on this method, based on Euler’s identity $np_n = \sum_{d,j \geq 1} dp_{n-dj}$, is given in [38, 39], and cited as “the recursive method.” We haven’t found a clearcut complexity analysis in the literature, although [13] comes close. But we believe that this algorithm is less useful than the $p(\leq k, n)$ table method — sampling from the distribution on $(d, j)$ implicit in $np_n = \sum_{d,j \geq 1} dp_{n-dj}$ requires computation of partial sums; if all the partial sums for $\sum_{d,j \geq 1, dj \leq m} dp_{m-dj}$ with $m \leq n$ are stored, the total storage requirement is of

---

9Since the largest part of a random partition is extremely unlikely to exceed $O(\sqrt{n} \log n)$, one can get away using a table of size $O(n^{3/2} \log n)$, which will only rarely need to be augmented. Specifically, writing $\lambda'_1$ for the largest part of a partition, with $c = \pi/\sqrt{6}$, for fixed $A > 0$, with $i_0 = i_0(n, A) := \sqrt{n} \log(A \sqrt{n}/c), P_n(\lambda_1' \geq i_0) \to 1 - \exp(-1/A)$. For example, take $A = 1000$, so that the $i_0$ by $n$ table will need to be augmented with probability approximately .001. With $M$ words of memory available for the table, we solve $i_0(n, A) \times n = M$; for example, with $M = 2^{37}$ and $A = 1000$ we have $n = 15000 \approx 2^{17}$ and $i_0 = 1700$, and increasing $M$ to $2^{37}$ gets us up to $n = 9 \times 10^6 \approx 2^{21}$, $i_0 = 15000$. Instead of actually augmenting the table, one could treat the roughly one out of every $A$ computationally difficult cases as deliberately missing data, as in Section 4.4.3.3.
order $n^2 \log n$, and if they aren’t stored, computing the values needed, on the fly, becomes a bottleneck.

4.4.2 Waiting to get lucky

Hardy and Ramanujan [22] proved the asymptotic formula, as $n \to \infty$,

$$p_n \sim \exp(2\sqrt{n})/(4\sqrt{3}n), \quad \text{where } c = \pi/\sqrt{6} \approx 1.282550.$$  \hspace{1cm} (4.18)

Fristedt [17] observed that, for any choice $x \in (0,1)$, if $Z_i \equiv Z_i(x)$ has the geometric distribution given by

$$P(Z_i = k) \equiv P_x(Z_i = k) = (1 - x^i)(x^i)^k, \quad k = 0, 1, 2, \ldots.$$  \hspace{1cm} (4.19)

with $Z_1, Z_2, \ldots$ mutually independent, and $T$ is defined by

$$T = Z_1 + 2Z_2 + \cdots.$$  \hspace{1cm} (4.20)

then, conditional on the event ($T = n$), the partition $\lambda$ having $Z_i$ parts of size $i$, for $i = 1, 2, \ldots$, is uniformly distributed over the $p_n$ possible partitions of $n$.\footnote{We write $P_x$ or $P$, or $Z_i$ or $Z_i(x)$ interchangeably, depending on whether the choice of $x \in (0,1)$ needs to be emphasized, or left understood.}

This extremely useful observation is easily seen to be true, since for any nonnegative integers $(c(1), c(2), \ldots)$ with $c(1) + 2c(2) + \cdots = n$, specifying a partition $\lambda$ of the integer $n$,

$$P(Z_i = c(i), i = 1, 2, \ldots) = \prod P(Z_i = c(i)) = \prod (1 - x^i)(x^i)^{c(i)}$$
\[ = x^{c(1)+2c(2)+\cdots} \prod (1-x^i) = x^n \prod (1-x^i), \tag{4.21} \]

which does not vary with the partition \( \lambda \) of \( n \).

The event \( (T = n) \) is the disjoint union, over all partitions \( \lambda \) of \( n \), of the events whose probabilities are given in (4.21), showing that

\[ \mathbb{P}_x(T = n) = p_n \ x^n \ \prod (1-x^i). \tag{4.22} \]

If we are interested in random partitions of \( n \), an especially effective choice for \( x \), used by [17, 42], is

\[ x(n) = \exp(-c/\sqrt{n}), \text{ where } c = \pi/\sqrt{6}. \tag{4.23} \]

Under this choice, we have, as \( n \to \infty \),

\[ \frac{1}{n} \mathbb{E}_x(n)T \to 1, \quad \frac{1}{n^{3/2}} \text{Var}_x(n)T \to \frac{2}{c}; \tag{4.24} \]

this is essentially a pair of Riemann sums, see [7], page 106. If we write \( \sigma(x) \) for the standard deviation of \( T \), then the second part of (4.24) may be paraphrased as, with \( x = x(n) \), as \( n \to \infty \),

\[ \sigma(x) \sim \sqrt{2/c} \ n^{3/4}. \tag{4.25} \]

The local central limit heuristic would thus suggest asymptotics for \( \mathbb{P}_x(T = n) \), and these simplify, using (4.23) and (4.24), as follows:

\[ \mathbb{P}_x(T = n) \sim \frac{1}{\sqrt{2\pi} \ \sigma(x)} \sim \frac{1}{2\sqrt{6} \ n^{3/4}}. \tag{4.26} \]
The Hardy-Ramanujan asymptotics (4.18) and the exact formula (4.22) combine to show that (4.26) does hold.

**Theorem 10. Analysis of Waiting-to-get-lucky.**

Consider the following algorithm to generate a random partition of \(n\), chosen uniformly from the \(p_n \sim \exp(2c\sqrt{n})/(4\sqrt{3}n)\) possibilities. Use the distributions in (4.19), with parameter \(x\) given by (4.23).

1. Propose a sample, \(Z_1, Z_2, \ldots, Z_n\); compute \(T_n := Z_1 + 2Z_2 + \cdots + nZ_n\).
2. In case \((T_n = n)\), we have got lucky. Report the partition \(\lambda\) with \(Z_i\) parts of size \(i\), for \(i = 1\) to \(n\), and stop. Otherwise, repeat from the beginning.

This algorithm does produce one sample from the desired distribution, and the expected number of proposals until we get lucky is asymptotic to \(2^{4/6}n^{3/4}\).

**Proof.** It is easily seen that \(\mathbb{P}_x(Z_i = 0 \text{ for all } i > n) \to 1\). [In detail, \(\mathbb{P}(\text{not } (Z_i = 0 \text{ for all } i > n)) \leq \sum_{i>n} \mathbb{P}(Z_i \neq 0) = \sum_{i>n} x^i < \sum_{i\geq n} x^i = x^n/(1-x) \sim x^n/(c/\sqrt{n}) = \exp(-c\sqrt{n})/\sqrt{n}/c \to 0\]. Hence the asymptotics (4.26), given for the infinite sum \(T\), also serve for the finite sum \(T_n\), in which the number of summands, along with the parameter \(x = x(n)\), varies with \(n\). \(\Box\)

**Remark 4.** We are not claiming that the running time of the algorithm grows like \(n^{3/4}\), but only that the number of proposals needed to get one acceptable sample grows like \(n^{3/4}\).

The time to propose a sample also grows with \(n\). Assigning cost 1 to each call to the random number generator, with all other operations being free, the cost to propose one sample grows like \(\sqrt{n}\) rather than \(n\); details in Section 4.5.2. Combining with Theorem 10,
the cost of the waiting-to-get-lucky algorithm grows like $n^{5/4}$. A simple Matlab® program
to carry out waiting-to-get lucky is presented in Section 4.5.1.

4.4.3 Divide-and-conquer, by small versus large

The waiting-to-get-lucky strategy is limited primarily by the probability that the target
is hit, which diminishes like $n^{-3/4}$. Already at $n = 10^8$, the probability is one in a million.
Instead of trying to hit a hole in one, we allow approach shots.

Recall that to sample partitions uniformly, based on (4.19) – (4.26), our goal is to
sample from the distribution of $(Z_1, Z_2, \ldots, Z_n)$ conditional on $T = n$. Using $x = x(n)$
from (4.23), and with $b \in \{1, 2, \ldots, n - 1\}$, we let

$$A = (Z_{b+1}, Z_{b+2}, \ldots, Z_n), \quad B = (Z_1, \ldots, Z_b). \quad (4.27)$$

Motivated by the standard paradigm for deterministic divide-and-conquer, that the two
tasks should be roughly equal in difficulty, we will take $b = O(\sqrt{n})$, having observed that
the expected largest part of a random partition grows like $\sqrt{n} \log n$. With

$$T_A = \sum_{i=b+1}^{n} iZ_i, \quad T_B = \sum_{i=1}^{b} iZ_i, \quad (4.28)$$

we want to sample from $(A, B)$ conditional on $T_A + T_B = n$. We have $A, B$ independent,
and we use $h(A, B) = 1(T_A + T_B = n)$. The divide-and-conquer strategy, according
to Lemma 7, is to sample $X$ from the distribution of $A$ biased by the probability that
an independently chosen $B$ will will make a match, and then, having observed $A = a$,
sampling $Y$ from the distribution of $B$ conditional on $h(a, B) = 1$. 

68
In order to simulate $X$, we will use rejection sampling, as reviewed in Section 4.2.2.

To find the optimal rejection probabilities, we want the largest $C$ such that

$$C \max_j \frac{\mathbb{P}(T_B = n - j)}{\mathbb{P}(T_n = n)} \leq 1,$$

or equivalently,

$$C = \frac{\mathbb{P}(T_n = n)}{\max_j \mathbb{P}(T_B = j)}. \quad (4.29)$$

The values $\mathbb{P}(T_B = k)$ for $k = 0, 1, \ldots, n$ can be simultaneously computed, using the recursion (4.17); what we really have is a variant of the $n$ by $n$ table method of Section 4.4.1, in which the table is $b$ by $n$, so the computation time is $bn$; furthermore, one only needs to store the current and previous row, (or with overwriting, only the current row), so the storage is $n$. Once we have the last row of the $b$ by $n$ table, we can easily find $C$ and indeed the entire threshold function $t$. The time to compute the table, $bn$, would have been the much larger $(n - b)n$ had we interchanged the roles of $A$ and $B$, i.e., taken $B := (Z_{b+1}, Z_{b+2}, \ldots, Z_n)$ instead of $B := (Z_1, \ldots, Z_b)$.

In order to simulate $Y = (B|Z_1 + 2Z_2 + \cdots + bZ_b = n - k)$, in situations where $b$ is too large to store a $b$ by $n$ table, we resort to waiting-to-get-lucky.\(^{11}\) Our goal for the next section is to explore mix-and-match using integer partitions as an example, regardless of whether there exists a better competing algorithm for integer partitions.

\(^{11}\)Of course, instead of trying to get lucky all at once, one might apply divide-and-conquer to this $B$ task. But we do not pursue this particular iteration, in light of a better iteration scheme, presented in Section 4.4.5.
4.4.3.1 Divide-and-conquer with mix-and-match

When a sample of size \( m > 1 \) is desired, Lemmas 7 – 9 can be used to generate unbiased samples. Observe that the matching function with \( h(A, B) = 1(T_A + T_B = n) \), for \((A, B) \in A \times B\), satisfies condition 2 of Lemma 8, with \( c_A(A) = T_A \) and \( c_B(B) = n - T_B \). Hence mix-and-match is enabled. The first phase of the algorithm, phase A, generates samples \( A_1, A_2, \ldots, A_m \) from \( X \), according to Lemma 7. This creates a multiset of \( m \) colors, \( \{c_1, \ldots, c_m\} \), where \( c_i = c_A(A_i) \). We think of these as \( m \) demands that must be met by phase B of the algorithm. One strategy for phase B is to generate an iid sequence of samples of \( B \); initially, for each sample, we compute its color \( c = c_B(B) \) and check whether \( c \) is in the multiset of demands \( \{c_1, \ldots, c_m\} \); when we find a match, we pair \( B \) with one of the \( A_i \) of the matching color, to produce our first sample, which we set as \( S_i = (A_i, B) \). Then we reduce the multiset of demands by one element, and iterate, until all \( m \) demands have been met. Lemma 9 implies that the resulting list \( (S_1, S_2, \ldots, S_m) \) is an iid sample of \( m \) values of \( S \), as desired.

Remark 5. Note that in the above, if the first match found, \((A_i, B)\), is labelled as \( S_1 \), and the second matching pair is labelled \( S_2 \), and so on, then the resulting list \( (S_1, S_2, \ldots, S_m) \) is not necessarily an iid sample of \( S \); the colors \( c \) with \( \mathbb{P}(c = c_B(B)) \) large would tend to show up earlier in the list.

4.4.3.2 Roaming \( x \)

Consider again a sample of size \( m = 1 \). Having accepted \( X = (Z_{b+1}, \ldots, Z_n) \) with color \( k = T_A \), in the notation of (4.28), we now need \( Y \), which is \( B = (Z_1, \ldots, Z_b) \) conditional
on having color \( k \), which simplifies to having \( n - k = T_B := \sum_{i=1}^{b} iZ_i \). One obvious strategy is to sample \( B \) repeatedly, until getting lucky. The distribution of \( B \) is specified by (4.19) and (4.23) — with a choice of parameter, \( x = x(n) \), not taking into account the values of \( b \) and \( n - k \). A computation similar to (4.21) shows that the distribution of \( (Z_1(y), \ldots, Z_b(y)) \) conditional on \( \sum_{i=1}^{b} iZ_i(y) = n-k \) is the same, for all choices \( y \in (0,1) \).

As observed in [7], the \( y \) which maximizes \( \mathbb{P}_y T_B = n - k \), i.e., gives us the best chance of getting lucky, is the solution of \( n - k = \sum_{i=1}^{b} \mathbb{E} iZ_i(y) \). Thus, in the case \( m = 1 \), the optimal choice of \( y \) is easily prescribed. However, for large \( m \) we also recommend using mix-and-match, which brings into play a complicated coupon collector’s situation. With a multiset of demands \( \{c_1, \ldots, c_m\} \) from Section 4.4.3.1, the algorithm designer has many choices of global strategy; it is not obvious whether or not a greedy strategy — picking \( y \) for the next proposed \( B \), to maximize the chance that \( B \) satisfied at least one of the demands — is optimal.

### 4.4.3.3 Deliberately missing data

For motivation: in the B phase of mix-and-match, one situation would have all \( m \) demands \( c_1, c_2, \ldots, c_m \) distinct, and in addition, \( \mathbb{P}(c_B(B) = c_i) \) relatively constant as \( i \) varies. In this situation, we have the classic coupon collector’s problem; with time \( m \log m \) to collect all \( m \) coupons. There is a harmonic slowdown: for example, the first match is found 100 times as quickly as the match after ninetynine percent of the demands have been met; the last \( v \) matches are expected to take about \( (1 + 1/2 + \cdots + 1/v)/\log m \) as long as the first \( m - v \) matches combined. In the situation with the \( \mathbb{P}(c_B(B) = c_i) \) relatively constant, but with large multiplicities in the multiset \( \{c_1, \ldots, c_m\} \), there is an even more dramatic
slowdown near the end, based on the size of the set underlying the multiset of remaining demands. Note however, the values $\mathbb{P}(c_B(B) = c_i)$ might not be relatively constant, even if we adjust the sampling distributions of $B$ to fit the particular colors remaining to be found, as described in Section 4.4.3.2.

Suppose we stop before completing the B phase, with some $v$ of demands remaining to be met. There is usable information in the partially completed sample, which has the $m - v$ partitions found so far. This list of $m - v$ is not a sample of size $m - v$, since sample requires iid from the original target distribution. But, had we run the B phase to completion, we would have had an honest sample of size $m$, so there is information in knowing all but $v$ of the $m$ values. Think of this sample of size $m$ as the sample, with some $v$ of its elements being unknown. For estimates based on the sample proportion, an error of size at most $v/m$ is introduced by the unknown elements. Since the standard deviation, due to sample variability, decays roughly like $1/\sqrt{m}$, it makes sense to allow $v$ comparable to $\sqrt{m}$.

For example, Pittel [43] proves that, as $n \to \infty$, given two partitions of $n$, choosing $(\lambda, \mu)$ uniformly over the $(p_n)^2$ possible pairs, the probability $\pi_n$ that $\lambda$ dominates $\mu$ satisfies $\pi_n \to 0$. Also, he proves that, choosing a single partition $\lambda$ uniformly over the $p_n$ possibilities, the probability $\pi^*_n$ that $\lambda$ dominates its dual $\lambda'$ satisfies $\pi^*_n \to 0$. It is natural to guess that $\lim \pi_n / \pi^*_n$ exists, with a value in $[0, \infty]$; one can use simulations to suggest whether 0, 1, or $\infty$ is the most plausible value. The harder task is to simulate for $\pi_n$.

If one has an honest sample of $2m$ partitions of $n$, with $v$ missing items due to terminating the B phase early, then one would have an honest sample of $m - V$ pairs
$(\lambda, \mu)$, with random variable $V \leq v$.\textsuperscript{12} If we had $v = 0$, and $H$ of the pairs have $\lambda$ dominates $\mu$, then the point estimate for $p := \pi_n$ is $H/m$, and the standard $(1 - \alpha)\%$ confidence interval is

$$\left[ \frac{H}{m} - \frac{z_{\alpha/2}}{m} \sqrt{\frac{p(1-p)}{m}}, \frac{H}{m} + \frac{z_{\alpha/2}}{m} \sqrt{\frac{p(1-p)}{m}} \right].$$

With $V$ missing pairs, consider the count $K$, how many of the $m - V$ completed pairs had $\lambda$ dominates $\mu$. We can do a worst-case analysis by assuming, on one side, that all $V$ of the missing pairs have domination, and on the other side, that none of the $V$ missing pairs have domination; more succinctly, $H \in [K, K + V]$. Hence the confidence interval

$$\left[ \frac{K}{m} - \frac{z_{\alpha/2}}{m} \sqrt{\frac{p(1-p)}{m}}, \frac{K + V}{m} + \frac{z_{\alpha/2}}{m} \sqrt{\frac{p(1-p)}{m}} \right].$$

is at least a $(1 - \alpha)\%$ confidence interval, for the procedure with deliberately missing data\textsuperscript{13}.

### 4.4.4 Divide-and-conquer with a trivial second half

Ignoring the paradigm that divide-and-conquer should balance its tasks, in (4.27) a very useful choice is $b = 1$. Loosely speaking, it reduces the cost of waiting-to-get lucky from order $n^{3/4}$ to order $n^{1/4}$.

\textsuperscript{12}The pairing on $\{1, 2, \ldots, 2m\}$ must be assigned before observing which $v$ items are missing. Pairing up the missing partitions, in order to get $\lceil v/2 \rceil$ missing pairs, in not valid; see Remark 5.

\textsuperscript{13}We feel compelled to speculate that many users of “confidence intervals” don’t really care about the confidence, nor the width of the interval, but really rely on the center of the interval, which in the standard case is an unbiased estimator. Our interval has center $(K + V/2)/m$, and this value is not an unbiased estimator; indeed, anything based on $K$, including the natural $K/(m - V)$, is biased in an unknowable way.
The analysis of the speedup, relative to waiting-to-get-lucky, is easy.

**Theorem 11.** *The speedup of the \( b = 1 \) procedure above, relative to the waiting-to-get-lucky algorithm described in Theorem 10, is asymptotically \( \sqrt{n}/c \), with \( c = \pi/\sqrt{6} \). Equivalently, the acceptance cost is asymptotically \( 2n^{1/4}6^{3/4}/\pi \).*

**Proof.** Recall that \( x = e^{-c/\sqrt{n}} \). From (4.10) and (4.29), the acceptance cost \( 1/C \) is given by \( C = \mathbb{P}(T = n)/\max_k \mathbb{P}(Z_1 = k) = \mathbb{P}(T = n)/\mathbb{P}(Z_1 = 0) = \mathbb{P}(T = n)/(1 - x) \). The comparison algorithm, waiting-to-get-lucky, has acceptance cost \( 1/\mathbb{P}(T = n) \). The ratio simplifies to \( 1/(1 - x) \sim \sqrt{n}/c \). \( \square \)

To review, stage 1 is to simulate \((Z_2, Z_3, \ldots, Z_n)\), and accept it with probability proportional to the chance that \( Z_1 = n - (2Z_2 + \cdots + nZ_n) \); the speedup comes from the brilliant idea in [50]. In contrast, waiting-to-get-lucky can be viewed as simulating \((Z_2, Z_3, \ldots, Z_n)\) and then simulating \( Z_1 \) to see whether or not \( Z_1 = n - (2Z_2 + \cdots + nZ_n) \).

### 4.4.5 Self-similar iterative divide-and-conquer: \( p(z) = d(z)p(z^2) \)

The methods of Sections 4.4.2 – 4.4.4 have acceptance costs that go to infinity with \( n \). We now demonstrate an iterative divide-and-conquer that has an asymptotically constant acceptance cost.

A well-known result in partition theory is

\[
p(z) = \prod_i (1 - z^i)^{-1} = \left( \prod_i 1 + z^i \right) \left( \prod_i \frac{1}{1 - z^{2i}} \right) = d(z)p(z^2),
\]

where \( d(z) = \prod_i 1 + z^i \) is the generating function for the number of partitions with distinct parts, and \( p(z^2) \) is the generating function for the number of partitions where each part has
an even multiplicity. This can of course be iterated to, for example, $p(z) = d(z)d(z^2)p(z^4)$, etc., and this forms the basis for a recursive algorithm.

Recall from (4.19) in Section 4.4.2, that each $Z_i \equiv Z_i(x)$ is geometrically distributed with $P(Z_i \geq k) = x^k$. The parity bit of $Z_i$, defined by

$$
\epsilon_i = 1(Z_i \text{ is odd}),
$$

is a Bernoulli random variable $\epsilon_i \equiv \epsilon_i(x)$, with

$$
P(\epsilon_i(x) = 1) = \frac{x^i}{1 + x^i}, \quad P(\epsilon_i(x) = 0) = \frac{1}{1 + x^i}.
$$

Furthermore, $(Z_i(x) - \epsilon_i)/2$ is geometrically distributed, as $Z_i(x^2)$, again in the notation (4.19), and $(Z_i(x) - \epsilon_i)/2$ is independent of $\epsilon_i$. What we really use is the converse: with $\epsilon_i(x)$ as above, independent of $Z_i(x^2)$, the $Z_i(x)$ constructed as

$$
Z_i(x) := \epsilon_i(x) + 2Z_i(x^2), \quad i = 1, 2, \ldots
$$

indeed has the desired geometric distribution.

**Theorem 12.** The asymptotic acceptance cost for one step of the iterative divide-and-conquer algorithm using $A = (\epsilon_1(x), \epsilon_2(x), \ldots)$, $B = ((Z_1(x) - \epsilon_1)/2, (Z_2(x) - \epsilon_2)/2, \ldots)$, is $\sqrt{8}$. 

75
Proof. The acceptance cost $1/C$ can be computed via (4.29) and (4.26), with

$$C = \frac{P_x(T = n)}{\max_k P_x(T = \frac{n-k}{2})} = \frac{P_x(T = n)}{\max_k P_x(T = k)} \sim \frac{1}{\sqrt{2\pi \sigma(x)}} \frac{1}{\sqrt{2\pi \sigma(x^2)}} \sim n^{-3/4} \frac{(n/4)^{-3/4}}{4^{-3/4}} = 8^{-1/2}.$$

Here is an informal discussion of the full algorithm. First, propose $A$ until getting acceptance, then, since the B task is to find a uniformly chosen partition of a smaller integer, iterate to finish up. In effect, the iterative algorithm is to determine the $(Z_1, Z_2, \ldots, Z_n)$ conditional on $Z_1 + 2Z_2 + \ldots = n$, by finding the binary expansions: first the 1s bits of all the $Z_i$s, then the 2s bits, then the 4s bits, and so on.

With a little more detail: to start, with $A = (\epsilon_1(x), \epsilon_2(x), \ldots)$ and $T_A = \sum_i i \epsilon_i$, we have $\mathbb{E} T_A = \sum_{i=1}^n \frac{i \epsilon_i}{1+x^i} \sim n/2$, and it can be shown that, even after conditioning on acceptance, the distribution of $T_A$ is concentrated around $n/2$. Since $B = ((Z_1(x) - \epsilon_1)/2, (Z_2(x) - \epsilon_2)/2, \ldots)$ is equal in distribution to $(Z_1(x^2), Z_2(x^2), \ldots)$, and target $n' = (n - T_A)/2$, we see that the B phase is to find a partition of the integer $n'$, uniform over the $p_{n'}$ possibilities. In carrying out the B task we simply use $x(n')$ as the parameter, but the choice $(x(n))^2$ would also work.

4.4.5.1 Exploiting a parity constraint

Theorem 12 states that the asymptotic acceptance cost for proposals of $A = (\epsilon_1(x), \epsilon_2(x), \ldots)$ is $2\sqrt{2}$, and this already takes into account an obvious lower bound of 2, since the parity of $T_A = \epsilon_1 + \epsilon_2 + \cdots + \epsilon_n$ is nearly equally distributed over \{odd,even\},
and rejection is guaranteed if $T_A$ does not have the same parity as $n$. An additional speedup is attainable by moving $\epsilon_1$ from the A side to the B side: instead of simulating $\epsilon_1$, there now will be a trivial task, just as there was $Z_1$ in the “$b = 1$” procedure of Section 4.4.4. That is, we switch to $A = (\epsilon_2(x), \epsilon_3(x), \ldots)$ and $B = (\epsilon_1(x), (Z_1(x) - \epsilon_1)/2, (Z_2(x) - \epsilon_2)/2, \ldots)$; the parity of the new $T_A$ dictates, deterministically, the value of the first component of $B$, under the conditioning on $h(a, B) = 1$. The rejection probabilities for a proposed $A$ are like those in Theorem 12, but with an additional factor of $1/(1+x^1)$ or $x/(1+x^1)$, depending on the parity of $n + \epsilon_2 + \cdots + \epsilon_n$. Since $x = x(n) \to 1$ as $n \to \infty$, these two factors both tend to $1/2$, so the constant $C$ as determined by (4.29) becomes, asymptotically, twice as large.

**Theorem 13.** The asymptotic acceptance cost for one step of the iterative divide-and-conquer algorithm using $A = (\epsilon_2(x), \epsilon_3(x), \ldots)$ and $B = (\epsilon_1(x), (Z_1(x) - \epsilon_1)/2, (Z_2(x) - \epsilon_2)/2, \ldots)$ is $\sqrt{2}$.

**Proof.** The acceptance cost $1/C$ can be computed, as in the proof of Theorem 12, with the only change being that in the display for computing $C$, the expression under the $\max_k$, which was $\mathbb{P}(T(x^2) = (n-k)/2)$ changes to

$$
\mathbb{P}\left(2|n-k+\epsilon_1(x)\text{ and } T(x^2) = \frac{n-k}{2}\right)
$$

$$
= \mathbb{P}(2|n-k+\epsilon_1(x)) \times \mathbb{P}\left(T(x^2) = \frac{n-k}{2}\right) \sim \frac{1}{2} \mathbb{P}\left(T(x^2) = \frac{n-k}{2}\right).
$$

$\square$
The overall cost of the main problem and all its subproblems

Informally, for the algorithm in the previous section, the main problem has size $n$ and acceptance cost $\sqrt{2}$, applied to a proposal cost asymptotic to $c_0 \sqrt{n}$, for a net cost $\sqrt{2} c_0 \sqrt{n}$.

The first subproblem has random size, concentrated around $n/4$, and hence half the cost of the main problem. The sum of a geometric series with ratio $1/2$ is twice the first term, so the net cost of the main problem and all subproblems combined is $2\sqrt{2} c_0 \sqrt{n}$.

In framing a Theorem to describe this, we try to allow for a variety of costing schemes. We believe that the first sentence in the hypotheses of Theorem 14 is valid, with $\theta = 1/2$, for the scheme of Remark 4. The second sentence, about costs of tasks other than proposals, is trivially true for the scheme of Remark 4, but may indeed be false, in costing schemes which assign a cost to memory allocation, and communication.

Theorem 14. Assume that the cost $C(n)$ to propose the $A = (\epsilon_2(x),\epsilon_3(x),\ldots)$ in the first step of the algorithm of Section 4.4.5.1, is given by a deterministic function with $C(n) \sim c_0 n^\theta$ for some positive constant $c_0$ and constant $\theta \geq 1/2$, or even more generally, $C(n) = n^\theta$ times a slowly varying function of $n$. Assume that the cost of all steps of the algorithm, other than making proposals, is relatively negligible, i.e., $o(C(n))$. Then, the asymptotic cost of the entire algorithm is

$$\frac{1}{1 - 1/4^\theta} \sqrt{2} C(n) \leq 2 \sqrt{2} C(n).$$

such as the arithmetic to compute acceptance/rejection thresholds, the generation of the random numbers used in those acceptance/rejection steps, and merging the accepted proposals into a single partition.
Proof. The key place to be careful is in the distinction between the distribution of a proposed $A = (\epsilon_2(x), \epsilon_3(x), \ldots)$, and the distribution after rejection/acceptance. For proposals, in which the $\epsilon_i$ are mutually independent, with $T_A := \sum_2^n i \epsilon_i (x)$, with $x = x(n)$ from (4.23), calculation gives $\mathbb{E} T_A \sim n/2$ and $\text{Var}(T_A) \sim (1/c)n^{3/2}$. Chebyshev’s inequality for being at least $k$ standard deviations away from the mean, to be used with $k = k(n) = o(n^{1/4})$, and $k \to \infty$, gives $\mathbb{P}(|T_A - \mathbb{E} T_A| > k \text{SD}(T_A)) \leq 1/k^2$.

Now consider the good event $G$ that a proposed $A$ is accepted; conditional on $G$, the $\epsilon_i$ are no longer mutually independent. But the upper bound from Chebyshev is robust, with $\mathbb{P}(|T_A - \mathbb{E} T_A| > k \text{SD}(T_A)|G) \leq 1/(k^2 \mathbb{P}(G))$. Since $\mathbb{P}(G)$ is bounded away from zero, by Theorem 13, we still have an upper bound which tends to zero, and shows that $(n - T_A)/2$, divided by $n$, converges in probability to 1/4.

Write $N_i \equiv N_i(n)$ for the random size of the subproblem at stage $i$, starting from $N_0(n) = n$. The previous paragraph showed that for $i = 0$, $N_{i+1}(n)/N_i(n) \to 1/4$, where the convergence is convergence in probability, and the result extends automatically to each fixed $i = 0, 1, 2, \ldots$. We have deterministically that $N_{i+1}/N_i \leq 1/2$, so in particular $N_i > 0$ implies $N_{i+1} < N_i$. Set $C(0) = 0$, redefining this value if needed, so that the costs of all proposals is exactly the random

$$S(n) := \sum_{i \geq 0} C(N_i(n)).$$
It is then routine analysis to use the hypothesis that $C(n)$ is regularly varying, to conclude that $S(n)/C(n) \to 1/(1 - 4^{-\theta})$, where again, the convergence is \textit{convergence in probability}. The deterministic bound $N_{i+1}(n)/N_i(n) \leq 1/2$ implies that the random variables $S(n)/C(n)$ are bounded, so it also follows that $\mathbb{E}S(n)/C(n) \to 1/(1 - 4^{-\theta})$.

\subsection*{4.4.5.3 A variation based on $p(z) = p_{\text{odd}}(z) p(z^2)$}

With

$$p_{\text{odd}}(z) := \prod_{i \text{ odd}} (1 - z^i)^{-1},$$

Euler’s identity $d(z) = p_{\text{odd}}(z)$ suggests a variation on the algorithm of section 4.4.5. It is arguable, whether the original algorithm, based on $p(z) = d(z) p(z^2)$, and the variant, based on $p(z) = p_{\text{odd}}(z) p(z^2)$, are genuinely different.

Arguing the variant algorithm is different: the initial proposal is

$A = (Z_1(x), Z_3(x), Z_5(x), \ldots)$. Upon acceptance, we have determined

$(C_1(\lambda), C_3(\lambda), C_5(\lambda), \ldots)$, where $\lambda$ is the partition of $n$ that the full iterative algorithm will determine, and $C_i(\lambda)$ is the number of parts of size $i$ in $\lambda$. The $B$ task will find

$(C_2(\lambda), C_4(\lambda), C_6(\lambda), \ldots)$ by iterating the divide-and-conquer idea, so that the second time through the $A$ procedure determines $(C_2(\lambda), C_6(\lambda), C_{10}(\lambda), \ldots)$, and the third time through the $A$ procedure determines $(C_4(\lambda), C_{12}(\lambda), C_{20}(\lambda), \ldots)$, and so on.

Arguing that the variant algorithm is \textit{essentially} the same: just as in Euler’s bijective proof that $p_{\text{odd}}(z) = d(z)$, the original algorithm had a proposal $A = (e_1(x), e_2(x), \ldots)$, which can be used to construct the proposal

$(Z_1(x), Z_3(x), Z_5(x), \ldots)$ for the variant algorithm. That is, one can check that starting
with independent $\epsilon(i, x) \equiv \epsilon_i(x)$ given by (4.31), for $j = 1, 3, 5, \ldots$, $Z_j := \sum_{m \geq 0} \epsilon(j 2^m, x) 2^m$

indeed has the distribution of $Z_j(x)$ specified by (4.19), with $Z_1, Z_3, \ldots$ independent. And conversely, one can check that starting with the independent geometrically distributed $Z_1(x), Z_3(x), \ldots$, taking base 2 expansions yields mutually independent $\epsilon_1(x), \epsilon_2(x), \ldots$

with the Bernoulli distributions specified by (4.31). Hence one could program the two algorithms so that they are coupled: starting with the same seed, they would produce the same sequence of colors $T_A$ for the initial proposal, and the same count of rejections before the acceptance for the first time through the A procedure, with same $T_A$ for that first acceptance, and so on, including the same number of iterations before finishing. Under this coupling, the original algorithm produces a partition $\mu$ of $n$, the variant algorithm produces a partition $\lambda$ of $n$ — and we have implicitly defined the deterministic bijection $f$ with $\lambda = f(\mu)$.

Back to arguing that the algorithms are different: we believe that the coupling described in the preceding paragraph supplies rigorous proofs for the analogs of Theorems 12 and 13. For Theorem 14 however, one should also consider the computational cost of Euler’s bijection, for various costing schemes, and we propose the following analog, for the variant based on $p(z) = p_{\text{odd}}(z) p(z^2)$, combined with the trick of moving $\epsilon_1(x)$ from the A side to the B side, as in Section 4.4.5.1:

**Theorem 15.** Assume that the cost $D(n)$ to propose $(Z_1(x), Z_2(x), \ldots, Z_n(x))$, with $x = x(n)$, satisfies $D(n) = n^\theta$ times a slowly varying function of $n$. Assume also that the cost $D_A(n)$ to propose $A = (Z_1(x) - \epsilon_1(x), Z_3(x), Z_5(x), \ldots)$ satisfies $D_A(n) \sim D(n)/2$. 

81
Then, the asymptotic cost of the entire algorithm is

\[
\frac{1}{1 - 1/4^\theta} \sqrt{2} D(n)/2 \leq \sqrt{2} D(n).
\]

Proof. Essentially the same as the proof of Theorem 14.

It is plausible that the cost function \( C(n) \) from Theorem 14 and the the cost function \( D(n) \) from Theorem 15 are related by \( C(n) \sim D(n) \); note that this depends on the choice of costing scheme, essentially asking whether or not the algorithmic cost of carrying out Euler’s odd-distinct bijection is negligible.

Another argument that the two algorithms are different arises from the completely artificial example at the end of Section 4.5.4.

4.5 For integer partitions: review, method of choice

In Section 4.4, five methods were presented for the simulation of partitions of the integer \( n \). Now we review the costs of running these algorithms, taking into account the size of \( n \), the number \( m \) of samples to be delivered, and so on. We also consider alternate simulation tasks involving integer partitions with restrictions on the parts to see how the methods adapt.

4.5.1 Method of choice for unrestricted partitions

If one is interested in generating just a few partitions of a moderately sized \( n \), then the waiting-to-get-lucky method, with a dumb “time \( n \)” method of proposal for \((Z_1, \ldots, Z_n)\),
is very easy to program. The overall runtime is order of $n^{7/4}$ — a factor of $n$ to make a proposal, and a factor of $n^{3/4}$ for waiting to get lucky.\footnote{A smarter “time $\sqrt{n}$” method of proposal for $(Z_1, \ldots, Z_n)$ is described in Secton 4.5.2. It is harder to program, but gets the overall runtime down to order of $n^{5/4}$.} For example, in Matlab\textsuperscript{®} [36]

\begin{verbatim}
 n=100; logx=-sqrt(6*n)/pi; s=0;
 while s~=n,
     Z=floor(log(rand(n,1))./(1:n)'.*logx);
     s=(1:n)*Z;
 end
\end{verbatim}

runs on a common desktop computer\footnote{Macintosh iMac, 3.06 Ghz, 4 GB RAM} at around 600 partitions of $n = 100$ per second; with $n = 1000$ the same runs at about 20 partitions per second, and at $n = 10,000$ takes about 2 seconds per partition.

The table method is by far the fastest method, if one is interested in generating many samples, and the table of size $n^2$ floating point numbers fits in random access memory. For example, for $n = 10,000$ the same computer as above takes 5 seconds to generate the table — a one time cost, and then finds 40 partitions per second. At $n = 15,000$, the same computer takes 28 seconds to generate the table, and then finds 25 partitions per second. But at $n = 19,000$, the computer freezes, as too much memory was requested.

The divide-and-conquer methods of Sections 4.4.3 and 4.4.4, using the small versus large division of (4.27), offer a large speedup over waiting-to-get-lucky, but only for case
\( b = 1 \), with its trivial second half, can we analyze the speedup — the \( \sqrt{n}/c \) factor in Theorem 11.

The divide-and-conquer method based on \( p(z) = d(z)p(z^2) \) is unbeatable for large \( n \). Regardless of the manner of costing, be it only counting random bits used, or uniform costing, or logarithmic (bitop) costing, the cost to find a random partition of \( n \) must be asymptotically at least as large as the time to propose \( (Z_1, \ldots, Z_n) \) for a random partition of a random number around \( n \). The entire divide-and-conquer algorithm of Theorem 15, compared with just proposing \( (Z_1, \ldots, Z_n) \), has asymptotically an extra cost factor of \( \sqrt{2} \). So the claim of unbeatable at the start of this paragraph really means: asymptotically unbeatable by anything more than a factor of \( \sqrt{2} \).

### 4.5.2 Complexity considerations

At the end of Section 4.2.2 we note that in the general view of probabilistic divide-and-conquer algorithms, a key consideration is computability of the acceptance threshold \( t(a) \). The case of integer partitions, using any of the divide-and-conquer algorithms of Section 4.4.5, is perhaps exceptionally easy, in that computing the acceptance threshold is essentially the same as evaluating \( p_m \), an extremely well-studied task. For \( m > 10^4 \) a single term of the Hardy-Ramanujan asymptotic series suffices to evaluate \( p_m \) with relative error less than \( 10^{-16} \); see Lehmer [32, 34].\(^\text{17}\) This single term is

\[
hr_1(n) := \frac{\exp(y)}{4\sqrt{3}(n - \frac{1}{24})} \left( 1 - \frac{1}{y} \right), \text{ where } y = 2c \sqrt{n - \frac{1}{24}},
\]

\(^\text{17}\) We thank David Moews for bringing these papers to our attention.
and numerical tabulation\textsuperscript{18}, together with Lehmer’s guarantee, shows that

\[ |\ln p_n - \ln hr_1(n)| < 10^{-16} \text{ for all } n \geq 810. \]

Is floating point accuracy sufficient, in the context of computing an acceptance threshold \( t(a) \)? There is a very concrete answer, based on [29]. First, as in Lemma 7.14 in [4], given \( p \in (0,1) \), a \( p \)-coin can be tossed using a random number of fair coin tosses; the expected number is exactly 2, unless \( p \) is a \( k \)th level dyadic rational, i.e., \( p = i/2^k \) with odd \( i \), in which case the expected number is \( 2 - 2^{1-k} \). The proof is by consideration of say \( B, B_1, B_2, \ldots \) iid with \( \mathbb{P}(B = 0) = \mathbb{P}(B = 1) = 1/2 \); after \( r \) tosses we have determined the first \( r \) bits of the binary expansion of a random number \( U \) which is uniformly distributed in \((0,1)\), and the usual simulation recipe is that a \( p \)-coin is the indicator \( 1(U < p) \). Unless \( \lfloor 2^r p \rfloor = \lfloor 2^r U \rfloor \), the first \( r \) fair coin tosses will have determined the value of the \( p \)-coin. Exchanging the roles of \( U \) and \( p \), we see that number of bits of precision read off of \( p \) is, on average, 2, and exceeds \( r \) with probability \( 2^{-r} \). If a floating point number delivers 50 bits of precision, the chance of needing more precision is \( 2^{-50} \), per evaluation of an indicator of the form \( 1(U < p) \). Our divide-and-conquer doesn’t require very many acceptance/rejection decisions; for example, with \( n = 2^{60} \), there are about 30 iterations of the algorithm in Theorem 13, each involving on average about \( \sqrt{2} \) acceptance/rejection decisions, according to Theorem 12. So one might program the algorithm to deliver exact results; most of the time determining acceptance thresholds \( p = t(a) \) in floating point arithmetic, but keeping track of whether more bits of \( p \) are needed. On the unlikely

\textsuperscript{18}done in Mathematica\textsuperscript{®} 8.
event, of probability around $30 \times \sqrt{2}/2^{50} < 4 \times 10^{-14}$, that more precision is needed, the program demands a more accurate calculation of $t(a)$. This would be far more efficient than using extended integer arithmetic to calculate values of $p_n$ exactly.

Another place to consider the use of floating point arithmetic is in proposing the vector $(Z_1(x), \ldots, Z_n(x))$. If one call to the random number generator suffices to find the next arrival in a rate 1 Poisson process, we have an algorithm using $O(\sqrt{n})$ calls, which can propose the entire vector $(Z_1, Z_2, \ldots)$, using $x = x(n)$ from (4.23). The proposal algorithm is based on a compound Poisson representation of geometric distributions, and is similar to a coupling used in [5], Section 3.4.1. The supporting calculation here is that, with $x = x(n) = \exp(-c/\sqrt{n})$ and $c = \pi/\sqrt{6}$,

$$s(n) := \sum_{i,j \geq 1} x_{ij} = \sum_j \frac{1}{j^2} \cdot x_{ij} \leq \frac{\pi^2}{6} \cdot \frac{x}{1-x} \sim c\sqrt{n}.$$  

The algorithm constructs the rate 1 Poisson process on $(0, s(n))$, with the full interval partitioned into subintervals of lengths $x_{ij}/j$. With $Y_{i,j}$ equal to the number of arrivals in the subinterval of length $x_{ij}/j$, the $Y_{i,j}$ are mutually independent, Poisson distributed, and $Z_i := \sum jY_{i,j}$ constructs the desired mutually independent geometrically distributed $Z_1(x), Z_2(x), \ldots$.

Once again, suppose we want to guarantee exact simulation of a proposal $(Z_1(x), \ldots, Z_n(x))$. In the compound Poisson process with $s(n)$ arrivals expected, we need to assign exactly, for each arrival, say at a random time $R$, the corresponding index $(i, j)$, such the partial sum for $s(n)$ up to, but excluding the $ij$ term, is less than $R$, but the partial sum,
including the \( ij \) term, is greater than or equal to \( R \). Based on an entropy result from Knuth-Yao [29], a crucial quantity is

\[
h(n) := \sum_{i,j \geq 1} x^{ij}_{j} \log \frac{j}{x^{ij}_{j}} \leq (c - \zeta'(2)/c) \sqrt{n}.
\]

An exact simulation of the Poisson process, assigning \( ij \) labels to each arrival, can be done\(^{19}\) with \( O(s(n) + h(n)) \) genuine random bits, and the bounds for \( s(n) \) and \( h(n) \) show that this is \( O(\sqrt{n}) \).

The costing scheme which counts only the expected number of random bits needed is clearly inadequate. Consider the impractical algorithm: list all \( p_n \) partitions, for example in lexicographic order. Use \( \lfloor \log_2 p_n \rfloor \) random bits to choose an integer \( I \) uniformly from \( \{1, 2, \ldots, p_n\} \). Report the \( I \)th partition of \( n \). If one costs only by the number of random bits needed, the algorithm just described is \textit{strictly} unbeatable!

\subsection*{4.5.3 Partitions with restrictions}

As with unrestricted partitions, if \( n \) is moderate and a recursive formula exists, analogous to that of Section 4.4.1, then the table method is the most rapid, and divide-and-conquer is not needed. However, the requirement of random access storage of size \( n^2 \) is a severe limitation.

The self-similar iterative divide-and-conquer method of Section 4.4.5 is nearly unbeatable for large \( n \), for ordinary partitions. There are many classes of partitions with

\(^{19}\) on each interval \((m-1, m] \) for \( m = 1 \) to \( \lfloor s(n) \rfloor \), perform an exact simulation of the number of arrivals, which is distributed according to the Poisson distribution with mean 1. For each arrival on \((m-1, m] \), there is a discrete distribution described by those partition points lying in \((m-1, m] \), together with the endpoints \( m-1 \) and \( m \); calling the corresponding random variable \( X_m \), the sum of the base 2 entropies satisfies \( \sum h(X_m) \leq h(n) + s(n) \), since each extra subdivision of one of the original subintervals of length \( x^{ij}/j \) adds at most one bit of entropy.
restrictions that iterate nicely, and should be susceptible to a corresponding iterative divide-and-conquer algorithm. Some of these classes, with their self-similar divisions, are

1. distinct parts, \( d(z) = d_{\text{odd}}(z)d(z^2); \)

2. odd parts, \( p_{\text{odd}}(z) = d_{\text{odd}}(z)p_{\text{odd}}(z^2); \)

3. distinct odd parts, \( d_{\text{odd}}(z) = d_*(z)d_{\text{odd}}(z^3). \)

Here \( d_*(z) = (1 + z)(1 + z^5)(1 + z^7)(1 + z^{11}) \cdots \) represents distinct parts \( \equiv \pm 1 \mod 6. \)

Other recurrences are discussed in [27, 41, 46], and the standard text on partitions [2].

It is not easy to come up with examples where the optimal divide-and-conquer is like that in Section 4.4.3, based on small parts versus large parts. One suggestion is partitions with all parts prime; there should be a large range of \( n \) for which table methods are ruled out by the memory requirement, while the \( n \) memory, \( b \times n \) computational time to calculate rejection probabilities is not prohibitive. Another suggestion is partitions with a restriction on the multiplicity, for example, a part of size \( i \) can occur at most \( f(i) \) times — with \( f \) sufficiently complicated as to rule out iterative formulas such as those in the preceding paragraph. Another family is, for \( d = 2, 3, \ldots \), partitions where all parts differ by at least \( d \).

4.5.4 An eye for gathering statistics

The underlying motivation for sampling a random element \( S, m \) times under some given distribution \( \mu \), might be to produce a statistic \( h(S_1, S_2, \ldots, S_m) \) based on that sample. Conceivably, the deterministic function \( h \) might only look at some part of the data, so that there is a divide-and-conquer scheme in which \( S = (X, Y) \), using the notation of
Lemma 7, and \( h(S_1, S_2, \ldots, S_m) = g(X_1, X_2, \ldots, X_m) \) for a deterministic \( g \). In this case, in the notation of Section 4.2.1, one need only carry out Stage 1.

Here is a completely artificial example, motivated by the question of dominance, discussed in Section 4.4.3.3. For a partition \( \lambda \) of \( n \), define \( f(\lambda) \) to be the partition, having distinct parts, in which \( i \) is a part of \( f(\lambda) \) if and only if \( i \) has odd multiplicity as a part of \( \lambda \). Now suppose that one needs to approximate, for say \( n = 10^9 \), \( \pi_n^\text{parity} = \mathbb{P}(f(\lambda) \text{ dominates } f(\mu)) \), choosing \( (\lambda, \mu) \) uniformly over the \( (p_n)^2 \) possible pairs of partitions of \( n \).

4.6 For comparison: opportunistic divide-and-conquer

with mix-and-match

Recall the setup of (4.1) – (4.4). In the third and fourth paragraphs of Section 4.1.2, we describe our starting point, a motivation stemming from mix-and-match, and admit that we can find no way to carry out “opportunistic” mix-and-match, to get a perfect sample. Here, we point out that nonetheless, the natural opportunistic procedure does supply a consistent estimator.

Take a deterministic design: for integers \( m_1, m_2 \geq 1 \), let \( A_1, \ldots, A_{m_1} \) be distributed according to the distribution of \( A \) given in (4.1), and let \( B_1, \ldots, B_{m_2} \) be distributed according to the distribution of \( B \), with these \( m_1 + m_2 \) sample values being mutually independent. The opportunistic observations, under a take-all-you-can-get strategy,
are all the pairs \((A_i, B_j)\) for which \(h(A_i, B_j) = 1\). To use these in an estimator, one would naturally count the available pairs, via

\[
W = W(m_1, m_2) = \sum_{1 \leq i \leq m_1, 1 \leq j \leq m_2} h(A_i, B_j), \tag{4.32}
\]

and for a deterministic function \(g : \text{support } h \subset A \times B \rightarrow \mathbb{R}\), form the total score from these pairs, say

\[
G \equiv G(m_1, m_2) = \sum_{1 \leq i \leq m_1, 1 \leq j \leq m_2} g(A_i, B_j) h(A_i, B_j). \tag{4.33}
\]

The natural estimator is the observed average score per pair, \(G/W\). Unfortunately, this is not unbiased. However, it is consistent, i.e., as \(m_1, m_2 \to \infty\), \(G/W \to \mathbb{E} g(S)\) in probability.

**Theorem 16.** For bounded \(g\), we get an unbiased estimate of \(p \mathbb{E} g(S)\) from \(G/(m_1 m_2)\). Furthermore, as \(m_1, m_2 \to \infty\) \(G(m_1, m_2)/(m_1 m_2) \rightarrow p \mathbb{E} g(S)\), where the convergence is convergence in probability.

**Proof.** To show unbiased: for each \(i, j\) we have, since \(h\) is an indicator, with \(\mathbb{E} h(A_i, B_j) = p\),

\[
\mathbb{E} g(A_i, B_j) h(A_i, B_j) = \mathbb{E} (g(A_i, B_j) | h(A_i, B_j) = 1) \times p.
\]

According to (4.4), the conditional distribution of \((A_i, B_j)\) given \(h = 1\) is equal to the distribution of \(S\), so the right side of the display above equals \(p \mathbb{E} g(S)\).

For consistency: write \(X_{ij} = g(A_i, B_j) h(A_i, B_j)\). In the variance-covariance expansion of \(\text{Var}(G)\), there are \(m_1 m_2\) terms \(\text{cov}(X_{ij}, X_{i'j'})\) where both \(i = i'\) and \(j = j'\),
\(m_1m_2(m_2 - 1)\) terms where only \(i = i'\), \(m_1(m_1 - 1)m_2\) terms where only \(j = j'\), and all other terms — involving four independent random variables \(A_i, A_{i'}, B_j, B_{j'}\) — are zero. Hence \(Var(G/(m_1m_2)) \to 0\), and Chebyshev’s inequality implies the desired convergence in probability.

Observe that in the special case \(g = 1\), the random variable \(G\) is the count \(W\), so Theorem 16 implies that \(W/(m_1m_2) \to p\). This shows that \(G/W\) is a consistent estimator of \(\mathbb{E} g(S)\).
Chapter 5

Opportunistic Probabilistic Divide-And-Conquer

Don’t only practice your art,

but force your way into its secrets,

for it and knowledge can raise Men to the Divine.

– Beethoven

Finally I’m becoming stupider no more.

– Erdos

The original intention of PDC was to provide exact samples, but in order to achieve this we must have access to the conditional distributions. In Section 4.2.2 the quantity $t(a)$ was defined as the threshold function necessary to carry out von Neumann’s rejection recipe in order to obtain the appropriate conditional distribution of $(A|h(A, B) = 1)$. Unfortunately, it is not always possible to calculate, or to calculate efficiently, the conditional distributions required, and yet we may still be interested in the information contained in
a sample from a related distribution, i.e., a sample that comes from a distribution “close” to the target.

There are different ways to quantify the distance between two distributions, for example total variation or Wasserstein distance, and this information can be used in various contexts like confidence intervals. A good starting reference on probability metrics is [21].

Section 4.6 introduced a take-all-you-can-get approach to parameter estimation, which produced consistent but not unbiased estimators. We are in the process of developing various opportunistic approaches to PDC, with the goal of obtaining upper bounds on the total variation and Wasserstein distance between the opportunistic quasi-samples and the desired but unobtainable exact samples. The results are preliminary at present, and will not be discussed in this thesis.
References


