EXPERIMENTAL METHODS IN PERMUTATION PATTERNS AND BIJECTIVE PROOFS

BY NATHANIEL SHAR

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Mathematics

Written under the direction of

Doctor Zeilberger

and approved by

New Brunswick, New Jersey

March, 2016

ABSTRACT OF THE DISSERTATION

Experimental Methods in Permutation Patterns and Bijective Proofs

by Nathaniel Shar

Dissertation Director: Doron Zeilberger

Acknowledgements

Chapter 2 consists of material adapted from [36], previously published in *Journal of Difference Equations and Applications*.

Chapter 3 consists of material adapted from [35], to be published in *Annals of Combinatorics*. The material in this chapter, with the exception of the Addendum, was coauthored by Doron Zeilberger. The final publication is available at Springer via `http://dx.doi.org/[insertDOI]`.

Section 4.1 features material adapted from [37]. The material in this section was coauthored by Doron Zeilberger.

Dedication

Table of Contents

Abstract ii

Acknowledgements iii

Dedication iv

1. Introduction 1

1.1. Experimental mathematics 1

1.2. The ansatz ansatz 1

1.3. Automatic proof of combinatorial identities 3

1.3.1. Bridging the gap between algebraic and bijective proofs 7

1.4. Enumeration schemes 8

1.5. Permutation patterns 10

1.5.1. Gessel's approach: Young tableaux and the Robinson-Schensted correspondence 11

1.5.2. Vatter, Wilf, and Zeilberger's approach: Prefix schemes 13

1.5.3. Other enumeration schemes 17

1.5.4. Patterns of low codimension 17

2. Automatic bijections 19

2.1. Introduction 19

2.2. The translation method 20

2.2.1. Building up bijections from simpler bijections 20

2.2.2. Bijectification 21

2.2.3. Substitution 22

2.2.4. Translation 22

2.3. A bijectifiable proof of Franel's identity 26

2.4. Further automation 29

3. 123-avoiding words 30

3.1. Introduction 30

3.2. Some Crucial Background and Zeilberger's Beautiful Snappy Proof that 123-Avoiding Words are Equinumerous with 132-Avoiding Words 33

3.3. Definitions 34

3.4. First Warm-Up: $r = 1$ 35

3.5. Second Warm-Up: $r = 2$ 35

3.6. The General Case 37

3.7. Guessing Linear Recurrences for our sequences 37

3.8. The Maple package Words123 38

3.9. The recurrences for $1 \leq r \leq 3$ 38

3.10. The Asymptotics for $1 \leq r \leq 5$ 39

3.11. Addendum 39

4. Repeating patterns of low codimension 41

4.1. The increasing pattern 41

4.1.1. Preface 41

4.1.2. Counting the "Bad Guys" 44

4.1.3. Examples for small values of r 45

4.1.4. Integer sequences 46

4.1.5. Efficient Computer-Algebra Implementation of Ira Gessel's AMAZING Determinant Formula 46

4.2. General repeating patterns 48

4.2.1. Definitions 49

4.2.2. The key lemmata 53

4.2.3. The main result 56

4.2.4. Independence of coefficients terms from choice of pattern 57

Appendices 63

A. Exact formula for $C_{r,d}$ 64

B. Excerpts of code for Chapter 2 65

C. Code for Chapter 4 87

C.1. main.cc 87

C.2. trie.h 94

C.3. trie.cc 95

References 100

Chapter 1

Introduction

1.1 Experimental mathematics

This thesis focuses on interesting results in enumerative combinatorics obtained through experimental techniques. In truth, it would be more correct to say that the focus is on experimental techniques, which incidentally obtain interesting results. For, as Tim Gowers noted, "the important ideas of combinatorics do not usually appear in the form of precisely stated theorems, but more often as general principles of wide applicability." [23]

The general principle at work here is that theorems become easier to prove when you let a computer do most of the work. Sometimes, the computer can produce a complete, self-contained proof of the theorem, as in the first chapter, in which we describe (with an example) how a computer can generate bijective proofs. Other times, as in the remaining chapters, a human first proves that a class of problems fit into a certain *ansatz*; then, based on that fact, a computer can rigorously solve the problems by *guessing* an answer and checking it for sufficiently many special cases.

Along the way, we will encounter interesting problems from the history of combinatorics and experimental mathematics.

1.2 The ansatz ansatz

One of the fundamental notions of experimental mathematics is that of the *ansatz*. Consider this old mathematical chestnut: What is the next term in the following sequence?

$$1, 2, 4, 8, 16, \dots \tag{1.1}$$

The victim of the question is supposed to answer "32," to which he is told "No, you idiot! It's 31!", after which much laughter is had at his expense. The sequence continues $31, 57, 99, 163, \ldots$ and counts the number of regions in 4-space formed by n hyperplanes (see A000127 in OEIS). Of course!

This joke is silly, but it makes an important point. If we know in advance that this sequence satisfies a linear recurrence, then the best answer is probably 32, because that causes the sequence to satisfy the simplest recurrence, $f(n) = 2f(n-1)$. On the other hand, if we know in advance that this sequence is a polynomial, then 31 is the best answer, because that allows f to have the lowest possible degree, 4. In short, the *kind* of sequence we are looking at determines how we should think about it and how we should guess the next term.

The fancy word for the "kind" of sequence we are looking at is *ansatz.* Typical ansatzen for integer sequences arising in enumerative problems include periodic, polynomial, quasipolynomial (a sequence consisting of several interlaced polynomials), C-recursive (a sequence that solves a linear recurrence with constant coefficients), algebraic (a sequence whose generating function is algebraic), and P-recursive (a sequence that solves a linear recurrence with polynomial coefficients). Of course, other disciplines of mathematics have their own ansatzen: for example, in number theory, the multiplicative ansatz is important, and in combinatorics on words, key ansatzen include Sturmian sequences and sequences that are fixed points of morphisms.

In general, if you know a sequence and an ansatz to which it belongs, it is relatively easy to guess a formula for a sequence. Also, if we know a sequence and its ansatz, then we reasonably believe that a simple formula from the ansatz is the true formula if it matches a sufficient amount of the data. For example, it may be difficult to directly count the regions in 4-space formed by n hyperplanes; most people's geometric intuition is limited to 2 or perhaps 3 dimensions. But if you can satisfy yourself that the answer must be a polynomial of at most 4th degree, then you can simply guess an answer of that form based on the first five terms of the sequence, and that (plus the proof that the sequence belongs to the "polynomial of degree at most 4" ansatz) constitutes a rigorous proof of the formula.

As another example of the power of the ansatz, consider Conway's famous "audioactive decay" sequence [11] [13], which proceeds

$$1, 11, 21, 1211, 111221, 312211, \ldots.$$

The nth term of this sequence contains a number of characters that is proportional to γ^n, where γ is a root of a 71st-degree polynomial. Proving this seems truly challenging (and indeed the theorem is almost unbelievable) – until you realize that the sequence a_n, where a_n is the number of characters in the nth term, belongs to the C-finite ansatz. Then, instead of solving an arbitrary problem, one is merely seeking a particular recurrence relation with constant coefficients, which can be guessed from finitely many examples.

1.3 Automatic proof of combinatorial identities

The field of combinatorial identities goes back almost as far as mathematics itself. One of the simplest combinatorial identities is the formula for the sum of an arithmetic series. For example, according to a dubious anecdote, Gauss is said to have discovered the following formula when a schoolteacher asked him to sum the numbers between 1 and 100:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{1.2}$$

The formula is, of course, far older, dating back to the Pythagoreans (see [6]), but as Gauss is said to have been five years old when he discovered it, he can be forgiven for failing to credit his predecessor.

Another source of combinatorial identities is the triangle of binomial coefficients, which, though usually referred to as "Pascal's triangle" in the West, is in fact far older than Pascal. In 1261 AD, Chinese mathematician Yang Hui published a method of finding square and cube roots using the binomial coefficients $\binom{n}{k}$, which he organized into a triangle. However, he credits the discovery to Jia Xian, who lived 200 years earlier. Even in the West, the binomial coefficients were known, for example to Levi Ben Gerson [3], long before Pascal. The key benefit of the triangular shape is that it allows binomial coefficients to be calculated rapidly using the identity

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$$

now known in the West as *Pascal's Identity.*

Once the binomial coefficients are written in a triangle, it is natural to notice other patterns. For example, we could sum the rows to find the identity

$$\sum_k \binom{n}{k} = 2^n, \tag{1.3}$$

or with a little more ingenuity, we might discover the following identity noted by Vandermonde (and, much earlier, by Zhu Shijie):

$$\binom{a+b}{n} = \sum_k \binom{a}{k}\binom{b}{n-k}. \tag{1.4}$$

Such patterns are combinatorial identities because they relate formulas involving quantities with a combinatorial interpretation.

Much ingenuity has gone into proving combinatorial identities, and many proof techniques were developed or adapted for this purpose; induction, bijective proof, generating functions, and hypergeometric identities being just a few examples. But mathematicians who encountered combinatorial identities in their work were not necessarily familiar with most of the techniques. Chapter 5 of [24] gives an example, in which

$$\sum_{k=0}^{n} k \frac{\binom{m-k-1}{m-n-1}}{\binom{m}{n}}$$

was not reduced to the much simpler equivalent form

$$\frac{n}{m-n+1}.$$

Reference works containing hundreds of identities have been published; among the most famous is Gould's table [22], which contains over 500 identities. But even such a monumental work does not remove the requirement for human ingenuity when a newly discovered identity is to be proved.

A systematic approach to the subject was eventually developed. This began with the work of Mary Celine Fasenmyer, who is better known as "Sister Celine" because she was, for most her life, a nun in the order of the Sisters of Mercy. She had always excelled at mathematics, and when she was 36 years old, the order sent her to the University of Michigan to pursue a doctorate. Her thesis, later summarized in two papers ([15], [16]), began the field of algorithmic proofs of combinatorial identities. The ideas were developed further by Gosper, Wilf, and Zeilberger, as explained beautifully in [28].

The method now known as "Sister Celine's method" is fundamentally experimental in nature. It is used to prove identities of the form

$$f(n) = \sum_k F(n,k) \tag{1.5}$$

where $F(n,k)$ is an expression involving binomial coefficients. (More technically, we will assume it is a doubly hypergeometric expression with compact support.) To do this, we first discover a recurrence relation of the form

$$\sum_{i=0}^{A}\sum_{j=0}^{B} a_{ij}(n)F(n-i,k-j) = 0,$$

where A and B are integers and each $a_{ij}(n)$ is a polynomial. Then this recurrence can be summed on k to obtain a formula for $f(n)$.

To discover the recurrence, we divide by $F(n,k)$; because $F(n,k)$ is hypergeometric, $F(n-i,k-j)/F(n,k)$ is a rational function. We then put everything over a common denominator, which will leave the numerator as a polynomial in k. We can then solve the system of equations that results from setting coefficients of k^j to zero for all j. If there is no nontrivial solution, then increase A and/or B until a trivial solution is found.

General theorems guarantee that for sufficiently large A and B, a nontrivial solution exists, and allow the required A and B to be estimated in advance.

As an example, we will show how Sister Celine's method 1.3 and 1.4. To prove 1.3, we first guess a recurrence for $F(n,k) = \binom{n}{k}$. Using the method described above, this can be done systematically by a computer, but in this case we have already noted such a recurrence; namely, 1.2, which can be rewritten

$$F(n,k) = F(n-1,k) + F(n-1,k-1). \tag{1.6}$$

To find a formula for $f(n)$, we simply sum both sides on k. Because $\binom{n}{k}$ is nonzero when $n < 0$ or $n > k$, we get

$$f(n) = f(n-1) + f(n-1). \tag{1.7}$$

Combined with the initial condition $f(0) = 1$, this yields $f(n) = 2^n$, and 1.3 is proved.

For a somewhat more elaborate example, we prove 1.4. We must first guess a recurrence for $F(n,k) = \binom{a}{k}\binom{b}{n-k}$. Here an answer is not immediately obvious. To obtain one, we may use an implementation of Sister Celine's method, such as the `celine` function from the Maple package `EKHAD` associated with [28]. In this way, we can obtain the recurrence

$$(n-a-b-2)F(n-2,k-1)+(n-a-1)F(n-1,k-1)+(n-b-1)F(n-1,k)+nF(n,k) = 0. \tag{1.8}$$

Summing on k yields

$$(n-a-b-2)f(n-2) + (2n-a-b-2)f(n-1) + nf(n) = 0. \tag{1.9}$$

We also have initial conditions $f(0) = 1$ and $f(1) = a + b$. We can now check that $f(n) = \binom{a+b}{n}$; alternatively, we could derive it using an algorithm such as Algorithm Hyper of [28].

While these experimentally produced proofs suffice to establish the truth of identities, some mathematicians find them unappealing. Connoisseurs of so-called "bijective proofs" seek to prove an identity $A = B$ by the following method:

1. Find sets S_A and S_B whose sizes are "obviously" equal to A and B, respectively

2. Find a bijection $f : S_A \to S_B$.

In some cases, the sets S_A and S_B are the same, in which case the second step may be omitted.

A bijective proof of Pascal's identity, for example, might go as follows. The left side $\binom{n}{k}$ counts the number of ways to choose a committee of k professors from the faculty of the Rutgers math department, which consists of n professors. The right side

$\binom{n-1}{k} + \binom{n-1}{k-1}$ also counts the number of ways to choose such a committee; the first term $\binom{n-1}{k}$ counts the number of ways to choose the committee such that Doron Zeilberger is not a member (so the k members must be chosen from the other $n-1$ professors), and the second term counts the number of ways to choose the committee so that Doron Zeilberger *is* a member (the other $k-1$ members being chosen from the other $n-1$ professors).

Here is a bijective proof of 1.3. The left side counts the ways to choose a subset of $[n]$ with k elements, then adds these up over all k. The right side counts the number of ways to choose a subset of $[n]$, regardless of the number of elements. Either way, both sides count the power set of $[n]$.

Finally, a bijective proof of 1.4 might go like this: The left side counts the n-subsets of $[a+b]$. The right side counts pairs (S,T), where S is a k-subset of $[a]$ and T is an $(n-k)$-subset of $[b]$. There is a bijection between the n-subsets of $[a+b]$ and the pairs (S,T). Namely, to a pair (S,T), we associate the set $S \cup (T+a)$. The reader may check that this is a bijection whose inverse associates S to the pair

$$(\{x \in S : x \leq a\}, \{x - a : x \in S, x > a\}).$$

This kind of bijective proof is simple and often seems to explain "why" an identity is true; thus, it is very different in character from the proofs generated by Sister Celine's method or the W-Z algorithm. The simplicity may be an illusion, though. One downside of bijective proofs is that papers presenting nontrivial (or even trivial) bijections are often long-winded; Chapter 2 of this thesis is no exception, and it doesn't even present the bijection explicitly. To avoid this, phrases like "The reader may observe" are often used; for example, see the previous paragraph.

1.3.1 Bridging the gap between algebraic and bijective proofs

In [49], Wood and Zeilberger show how certain inductive proofs can be transformed systematically into bijective proofs, thus, in some cases, reducing the need for human ingenuity in obtaining such proofs. Their method involves breaking the inductive proof into elementary algebraic steps, then using the sequence of steps to build a bijection in a mechanical way. The resulting bijection is not described in the cutesy terms of the bijections presented above; it is described as a mathematical function with a rather elaborate, recursive, and usually opaque definition. As a result, [49] suggests a methodology where the automatically generated bijection is implemented as a Maple function (or in some other computer language), and then a human explores the behavior of the function by hand until a simpler and more direct formula can be deduced, and then proved. The last step, of course, is optional! Once the function is implemented, that is already a bijection, and as we noted, the apparent simplicity of human-crafted bijections is somewhat illusory.

The idea behind this method can be applied in a broader context than inductive proofs. An example is provided in Chapter 2.

1.4 Enumeration schemes

To count a set S of combinatorial objects, we frequently partition S into subsets $S_1, \ldots, S_k$, and count the (now-smaller) sets. This technique is so basic to enumerative combinatorics that it has no name.

As a simple example, consider counting the subsets of $[n]$; let $S(n)$ be the collection of such subsets. If we do not see how to count $S(n)$ directly, we may define sets $S_1(n)$ and $S_2(n)$ so that $S_1(n)$ is the collection of subsets of $[n]$ that contain n, and $S_2(n)$ is the collection of subsets of $[n]$ that do not contain n. Then we have the formula

$$S(n) = S_1(n) + S_2(n).$$

Of course, this formula is useless, because it does not tell us how to count S_1 or S_2. But if we note that the elements of $S_2(n)$ are identical to the elements of $S(n-1)$, we have

$$S_2(n) = S(n-1).$$

Furthermore, the elements of $S_1(n)$ are *in bijection with* the elements of $S(n-1)$, where the bijection $f : S_1(n) \to S(n-1)$ is removing the element n. Instead of one formula for $S(n)$, we now have a system of formulae:

$$\begin{aligned} S(n) &= S_1(n) + S_2(n), \qquad n \geq 0 \\ S_1(n) &= S(n-1), \qquad n \geq 1 \\ S_2(n) &= S(n-1), \qquad n \geq 1. \end{aligned}$$

We can also calculate the *base cases* by direct enumeration; namely, $S(0) = S_1(0) = 1$, $S_2(0) = 0$. Thus we have found a system of recurrence relations for $S(n)$. This is called an enumeration scheme.

Of course, in this case, we may easily solve the system by substituting the latter two equations into the first, and obtaining $S(n) = 2S(n-1)$ for $n \geq 1$ and $S(0) = 1$. Thus $S(n) = 2^n$. Because this easy solution is availabile, every student of combinatorics is familiar with the formula for $S(n)$. The power of enumeration schemes is more evident when such an easy-to-derive solution does not exist. In some cases, the solution may be out of reach of a human, but within reach of a computer. To see how this is done, note the steps we followed in the process above:

1. Break S into disjoint sets $S_1, \ldots, S_k$.
2. For each piece S_i, do one of the following:
   (a) Count it directly
   (b) Find a simple relationship (e.g. bijection, identity) between S_i and other pieces
   (c) Break it into disjoint sets $S_{i1}, \ldots, S_{ij}$ and repeat the process.

Often, the "breaking" can be done in certain automatic ways. For example, when counting subsets, we broke a collection of subsets into pieces based on whether or not the largest allowed element was present. When counting permutations, we will frequently break into subsets based on the first or last element. So this step may be performed by a computer.

Finding relationships between $S_{ijk\ldots}$ and already enumerated sets may also be done automatically, if we are sufficiently clever in telling a computer how to search for such relationships. Knowledge of the problem is important. The classic (and original) example is [51], which was later extended in [43] by Vince Vatter's more clever method of searching for relationships.

Automation is beneficial because it allows the construction of enumeration schemes that would overwhelm a human with complexity. Even without automation, though, constructing an enumeration scheme to count a set can tell us something about the ansatz to which the enumeration problem belongs. This can allow formulas to be guessed and then proved, or in some cases, proved simply by verifying finitely many cases. In Chapter 3, we will show that the problem of enumerating 123-avoiding words with r occurrences of each letter belongs to the algebraic ansatz. And in Chapter 4, we will show that the problem of enumerating permutations of a given codimension avoiding "repeating" patterns belongs to the "eventually polynomial" ansatz.

1.5 Permutation patterns

The applications of enumeration schemes in this thesis are to problems involving permutation patterns. The field of permutation patterns is vast and rapidly developing; we will focus here on the most relevant background. A broader survey of the field may be found in [27]; the subject is also discussed in the highly accessible [5].

Two sequences of numbers $\pi_1, \ldots, \pi_n$ and $\sigma_1, \ldots, \sigma_n$ are *order-isomorphic* if for all $1 \leq i, j \leq n$, $\pi_i < \pi_j$ if and only if $\sigma_i < \sigma_j$, and $\pi_i = \pi_j$ if and only if $\sigma_i = \sigma_j$.

Given two permutations, $\pi \in S_n$ and $\sigma \in S_k$, we say that π contains the pattern σ if some subsequence of π is order-isomorphic to σ. (The subsequence need not be contiguous; for example, 13542 matches the pattern 132 because the subsequence 142 is order-isomorphic to 132.) If a permutation does not contain a pattern, we say it avoids the pattern.

More generally, if w is any finite *word* on the alphabet $[n]$, we say that w contains the pattern σ if some subsequence of w is order-isomorphic to σ.

A classical problem in the field of permutation patterns is counting the set $Av_n(\sigma)$, which is the set of permutations in S_n that avoid σ. This problem has proven to be extremely difficult in general. Formulas for the simplest cases are easily conjectured and proved:

$$|Av_n(1)| = \delta_{n0}$$
$$|Av_n(12)| = 1$$
$$|Av_n(123)| = |Av_n(132)| = C_n.$$

Here C_n is the nth Catalan number $\frac{1}{n+1}\binom{2n}{n}$. (See below for a proof of the latter equality.)

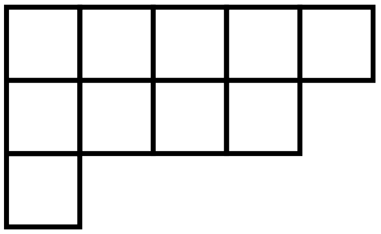
The classes $Av_n(12\dots k)$ were enumerated (for each k) by Gessel in [20]. The class $Av_n(2413)$ was proven to be equinumerous to $Av_n(1342)$ by Stankova in [39], and the latter was enumerated by Bona in [4]. For all other σ, the enumeration of $Av_n(\sigma)$ is open.

We will look in more detail at the enumeration of $Av_n(12\dots k)$; first, describing Gessel's original approach to the problem (which required much human cleverness); then, looking at an experimental approach by Zeilberger and Vatter that uses enumeration schemes.

1.5.1 Gessel's approach: Young tableaux and the Robinson-Schensted correspondence

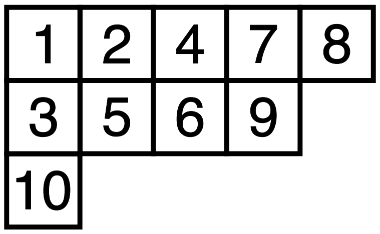
A Young diagram is a collection of finitely many boxes, arranged in left-justified rows of nonincreasing length (see Figure 1.5.1).

Figure 1.1: A Young diagram with shape $(5,4,1)$ [45].



Let $\mathcal{P}_n$ denote the set of partitions of n. If $\lambda \in \mathcal{P}_n$, a Young diagram is said to have shape λ if the lengths of the rows are given by λ.

Figure 1.2: A standard Young tableau with shape $(5, 4, 1)$ [44].



If numbers drawn from $[n]$ are written in the boxes of a Young diagram, such that the entries in each row are nondecreasing and the entries in each column are increasing, the result is called a semistandard Young tableau. To each semistandard Young tableau T, associate the monomial $x^T = \prod_{i \in T} x_i$, where i ranges over the numbers written in the boxes. The Schur function indexed by λ is defined to be $s_\lambda(x) = \sum_T x^T$, where $\lambda \in \mathcal{P}_n$ and the sum ranges over all semistandard Young tableau with shape λ.

A semistandard Young tableau is said to be standard if all the entries are distinct (see Figure 1.5.1). Thus, the coefficient of $x_1 \cdots x_n$ in s_λ is the number of standard Young tableaux of shape λ.

The Robinson-Schensted correspondence is a famous bijection between the sets S_n and

$$\bigcup_{\lambda \in \mathcal{P}_n} F_\lambda^2.$$

Among the many famous properties of the Robinson-Schensted correspondence is that permutations with a longest increasing subsequence of length k correspond to pairs of Young tableaux (P, Q) where the length of the first row of P (and Q) is k. Thus, to count $Av_n(12 \cdots (k+1))$, it suffices to find $\sum f_\lambda^2$, where λ ranges over the partitions with largest element less than or equal to k. Equivalently, we may have the sum range over the partitions with at most k parts.

In [20], Gessel proves a beautiful formula for

$$R_k(x, y) = \sum_\lambda s_\lambda(x) s_{lambda}(y),$$

where λ ranges over the partitions with at most k parts. (This is an infinite sum because there is no restriction on the size of the partition.) Specifically,

$$R_k(x, y) = \det(A), \tag{1.10}$$

where

$$A_{ij} = \sum_{r=0}^{\infty} h_{r+j-i}(x)h_r(y).$$

(Here $h_n(x)$ is the nth homogeneous symmetric polynomial.) Letting $x = y$ and extracting the homogeneous terms allows the number of permutations avoiding $12\cdots(k+1)$ to be enumerated explicitly.

1.5.2 Vatter, Wilf, and Zeilberger's approach: Prefix schemes

Let σ be a fixed permutation. Let

$$Av_n(\sigma;\tau) = \{\pi \in Av_n(\sigma) : red(\pi_1,\ldots,\pi_k) = \tau\};$$

that is, it is the set of permutations $\pi \in S_n$ such that π avoids σ and the first k elements of π form the pattern τ. As a refinement of this definition, let

$$Av_n(\sigma;i_1,\ldots,i_k) = \{\pi \in Av_n(\sigma) : \pi_1 = i_1, \pi_2 = i_2, \ldots, \pi_k = i_k\}.$$

That is, $Av_n(\sigma;i_1,\ldots,i_k)$ is the subset of permutations $\pi \in S_n$ such that π avoids σ and the first k elements of π are $i_1,\ldots,i_k$.

Let F_r be the map from S_n to S_{n-1} that deletes the rth element and then reduces. It is trivial that for $r \le k$,

$$F_r(Av_n(\sigma;i_1,\ldots,i_k)) \subseteq Av_n(\sigma;i'_1,\ldots,i'_{r-1},i'_{r+1},\ldots,i'_k),$$

where

$$i'_j = \begin{cases} i_j & \text{if } i_j < i_r \\ i_j - 1 & \text{if } i_j > i_r. \end{cases}$$

This inclusion is sometimes an equality. We focus on cases where this happens with the following definition:

Definition 1.1. *If $r \le k$ and $\tau \in S_k$ is such that, for all $n \ge k$ and for all $1 \le i_1,\ldots,i_k \le n$ with $red(i) = \tau$, 1.5.2 is an equality, then we say that place r is reversely deletable from τ.*

Examples are necessary. Let $\sigma = 1234$, and $\tau = 21$. Then place 1 is reversely deletable, but place 2 is not.

To show that place 1 is reversely deletable, it is necessary to prove that for all $i_2 < i_1$, all the permutations of length $n-1$ avoiding 1234 that start with i_2 (that is, i_2') continue to avoid 1234 when a i_1 is inserted at the beginning. To prove this, suppose π is a permutation of length $n-1$ such that $\pi(1) = i_2$ and such that π avoids 1234. Let π' be the permutation that results from inserting i_1 at the beginning of π. If the pattern 1234 occurs in π', then every occurrence must have that first element, i_1, as its first element. Suppose $1 < j < k < \ell$ and $\pi'(1), \pi'(j), \pi'(k), \pi'(\ell)$ is an occurrence of 1234. Because $\pi'(1) = i_1$ and $\pi'(2) = i_2 < i_1$, $j > 2$. Thus, $\pi'(2), \pi'(j), \pi'(k), \pi'(\ell)$ is a different occurrence of 1234, because $\pi'(2) < \pi'(1)$. But then $\pi(1), \pi(j-1), \pi(k-1), \pi(\ell-1)$ is an occurrence of 1234, which is a contradiction. So π' is 1234-avoiding.

To observe that place 2 is not reversely deletable, it suffices to show a counterexample. Let $i_1 = 3$ and $i_2 = 1$. Then 2673451 avoids 1234 and starts with $i_1' = 2$, but 31784562 does not avoid 1234

Critically, reverse deletability can be proved in a systematic way. We must check that inserting i_r is safe; therefore, we look at all the ways the inserted i_r could participate in an occurrence of σ. If each way in which i_r participates implies the existence of another occurrence of σ in which i_r does not participate, then place r is reversely deletable.

Consider our previous example. The argument may be summarized as follows. Any occurence of 1234 involving i_1 must look like one of the following:

1. $i_1 i_2 \pi_a \pi_b$

2. $i_1 \pi_a \pi_b \pi_c$

The first, however, is impossible, because $i_2 < i_1$. The second is possible, but it implies that $i_2 \pi_a \pi_b \pi_c$ is another occurrence of 1234, and i_1 does not particpate in that.

For a further example, let us show that place 2 is reversely deletable if $\sigma = 1234$ and $\tau = 2413$. The possible ways in which i_2 could participate in an occurrence of 1234 are:

1. $i_1 i_2 i_3 i_4$

2. $i_1 i_2 i_3 \pi_a$

3. $i_1 i_2 i_4 \pi_a$

4. $i_1 i_2 \pi_a \pi_b$

5. $i_2 i_3 i_4 \pi_a$

6. $i_2 i_3 \pi_a \pi_b$

7. $i_2 i_4 \pi_a \pi_b$

8. $i_2 \pi_a \pi_b \pi_c$

Scenarios 1, 2, 3, 5, 6, and 7 are impossible because of the order of i_1, i_2, i_3, i_4. If scenario 4 holds, then $i_1 i_4 \pi_a \pi_b$ is another occurrence of 1234, in which i_2 does not participate. If scenario 8 occurs, then $i_4 \pi_a \pi_b \pi_c$ is another occurrence of 1234, in which i_2 does not participate. So i_2 is reversely deletable.

We will now focus on the case where σ is an increasing permutation (that is, a permutation of the form $12 \dots k$).

Let us consider $\pi \in Av_n(1234; i_1, i_2, i_3)$, with $i_1 < i_2 < i_3$. If any element $\pi_a \in \pi$ is greater than i_3, then $i_1 i_2 i_3 \pi_a$ forms an occurrence of 1234. Therefore, there can be no such element, which means $i_3 = n$.

More generally, consider $\pi \in Av_n(123 \dots k; \tau; i_1, \dots, i_s)$. If $i_1, i_2, \dots, i_s$ contains an increasing subsequence of length $k-1$ that ends in i_s, then there can be no element after π_s that is greater than i_s. Thus, either $s = n$ or $i_s = n$.

Let $A(n; 123 \dots k; \tau; i_1, \dots, i_s) = |Av_n(123 \dots k; \tau; i_1, \dots, i_s)|$. Then we have the following facts. In all cases,

$$A(n; 123 \dots k; \tau; i_1, \dots, i_s) = \sum_{i_{s+1} \in [n] \setminus \{i_1, \dots, i_s\}} A(n; 123 \dots k; i_1, \dots, i_{s+1}). \tag{1.11}$$

If place r is reversely deletable in τ, then

$$A(n; 123 \dots k; \tau; i_1, \dots, i_s) = A(n-1; \sigma, i_1, \dots, i_{r-1}, i_{r+1}, \dots, i_s). \tag{1.12}$$

Finally, if τ contains a $123\dots k-1$ that includes the last element of τ,

$$A(n;123\dots k;\tau;i_1,\dots,i_s)=\begin{cases}A(n-1;\sigma,i_1,\dots,i_{s-1}) & \text{if } i_s=n\\ 1 & \text{if } s\geq n\\ 0 & \text{otherwise.}\end{cases} \tag{1.13}$$

In combination, 1.11, 1.12, and 1.13 can be used to form an enumeration scheme of size depending only on k that allows the computation of $A(n;123\dots k;)$. The simplest example is for $k=2$;

$$\begin{aligned}A(n;12;) &= \sum_{i=1}^{n} A(n;12;i)\\ A(n;12;n) &= A(n-1;12;)\\ A(n;12;i) &= 0 \qquad i\neq n.\end{aligned}$$

This system of recurrence relations quickly simplifies to

$$A(n;12;)=A(n-1;12;)$$

and with the initial condition $A(1;12;)=1$, we get the obvious formula $A(n;12;)=1$.

We get a more elaborate example when $k=3$. Here the resulting system of equations is

$$\begin{aligned}A(n;123;) &= \sum_{i=1}^{n} A(n;123;1;i)\\ A(n;123;1;i) &= \sum_{j=1}^{i-1} A(n;123;21;i,j)+\sum_{j=i+1}^{n} A(n;123;12;i,j)\\ A(n;123;12;i,n) &= A(n-1;123;1;i)\\ A(n;123;12;i,j) &= 0 \qquad (\text{if } j\neq n)\\ A(n;123;21;i,j) &= A(n-1;1;j)\end{aligned}$$

Here the solution $A(n;123;)=C_n$ is far less obvious, but once the first few terms have been generated, it can be guessed and then proved. This is the power of choosing an appropriate ansatz!

Even if we are not able to guess the solution, we can still generate the terms of the sequence $\{A(n; 123;)\}_{n=1}^{\infty}$ in polynomial time, making the enumeration scheme a solution in the sense of Wilf [47] to the problem of enumerating the 123-avoiding permutations.

Similar enumeration schemes can be produced for permutations avoiding $12 \dots k$, but the construction does not extend to permutations avoiding the more difficult patterns (such as 1324) because the enumeration schemes do not stop at a finite depth.

1.5.3 Other enumeration schemes

One may prove that $|Av_n(132)| = C_n$ through the use of a nonlinear enumeration scheme. Let $A(n) = |Av_n(132)|$ and let $B(n; i) = |\{\pi \in Av_n(132) : \pi_i = n\}|$. If $\pi \in B(n; i)$, then the elements before π_i must be greater than the elements after π_i. So $B(n; i) = A(i-1)A(n-i)$. While nonlinear recurrences (and nonlinear systems) are generally difficult to analyze, in this case, we can see that

$$A(n) = \sum_{i=1}^{n} A_{i-1}A_{n-i},$$

which is the recurrence for the Catalan numbers. It remains only to check that the initial conditions of the two sequences match, which they do because $A(0) = C_0 = 1$.

With sufficient cleverness, it may be that other permutation classes may be enumerated through nonlinear enumeration schemes. In Chapter 2, we analyze certain classes of words avoiding 132 using this kind of scheme.

1.5.4 Patterns of low codimension

In the classical problem, a pattern is fixed, and then the set of permutations avoiding that pattern is counted. Alternatively, we may fix a number r, and then, for each pattern σ, try to enumerate the set of permutations of length $|\sigma| + r$. The number r is called the codimension.

For codimension 1, the problem is simple, because the answer does not depend on σ. The origins of the following lemma are not completely clear. Vatter [42] attributes it to Pratt [29]; however, because the statement is so simple it could easily have been discovered earlier. The proof here is not Pratt's; he left it as an exercise to the reader, possibly because a correct proof is not quite as simple to state as one might hope! The author has not been able to find this proof in print before.

Lemma 1.2. *The number of permutations in S_{n+1} containing the pattern $\pi \in S_n$ is n^2+1.*

Proof. Consider all the ways to insert one element, j, in the ith position of π, where $1 \leq i, j \leq n+1$. Let $\pi[i,j]$ be the result, and say that two pairs (i,j) and (i',j') are equivalent if $\pi[i,j] = \pi[i',j']$. Clearly, if (i,j) is equivalent to (i',j'), then $i \neq i'$ and $j \neq j'$.

Call a pair (i,j) *redundant* if there is an equivalent pair (i',j') with $i' > i$.

Suppose $\pi(i) = j$. Then trivially, $\pi[i,j] = \pi[i+1, j+1]$, so (i,j) is redundant. Alternatively, suppose $\pi(i) = j-1$. Then $\pi[i,j] = \pi[i+1, j-1]$, so (i,j) is redundant.

Suppose $\pi(i)$ is neither j nor $j-1$. Suppose $i' > i$ and $j' > j$. Then $\pi[i',j']_i = \pi(i) \neq j = \pi[i,j]_i$, so (i,j) is not equivalent to (i',j'). On the other hand, if $i' > i$ and $j' < j$, then $\pi[i',j']_i = \pi(i)+1 \neq j = \pi[i,j]_i$, so (i,j) is not equivalent to (i',j'). We have shown that (i,j) is not redundant.

Finally, if $i = n+1$, then of course (i,j) is not redundant because there are no pairs (i',j') with $i' > n+1$.

Thus, there are exactly $2n$ redundant pairs. Each equivalence class of pairs contains exactly one non-redundant pair, so there are $(n+1)^2 - 2n = n^2+1$ such equivalence classes, as claimed. $\square$

Building on Pratt's result, Ray and West [31] have an almost-exact solution to the problem for codimension 2:

Theorem 1.3. *The number of permutations in S_{n+2} containing the pattern $\pi \in S_n$ is $n^4 + 2n^3 + n^2 + 4n + 4 - 2j$, where $0 \leq j \leq n-1$.*

No such result is known for codimension 3 or higher, though Ray and West prove a estimate with error $O(n^{2r-2})$, where r is the codimension. Experimentally, the error is much smaller.

Chapter 2

Automatic bijections

2.1 Introduction

Let

$$a_n^{(r)} = \sum_{k=0}^{n} \binom{n}{k}^r.$$

For fixed $r \geq 0$, the sequence $\left\{a_n^{(r)}\right\}_{n=1}^{\infty}$ satisfies a P-finite recurrence. The cases $r = 1$ and $r = 2$ are well-known:

$$a_n^{(1)} = 2a_{n-1}^{(1)}$$

$$na_n^{(2)} = (4n-2)a_{n-1}^{(2)}$$

These recurrences also have simple bijective proofs.

In the 1890s Franel [17, 18] proved the following recurrence for the cases $r = 3$:

$$n^2 a_n^{(3)} = (7n^2 - 7n + 2)a_{n-1}^{(3)} + 8(n-2)^2 a_{n-2}^{(3)}. \tag{2.1}$$

Nowadays Sister Celine's algorithm and Zeilberger's algorithm can be used to routinely find and prove such recurrences for larger r.

In this paper, we describe a method, extending that of Wood and Zeilberger [49], for translating algebraic proofs of recurrence relations into bijective proofs. As a proof of concept, we then apply the method to a carefully-crafted algebraic proof of Franel's recurrence, and give an explicit bijection that, although it may not be aesthetically pleasing, is provably correct.

2.2 The translation method

2.2.1 Building up bijections from simpler bijections

In order to develop a complicated enough bijection to prove the Franel recurrence, we define four operations on bijections, which previously appeared in [49]. Three of these operations, $+, \cdot$, and $\circ$, are binary. The fourth operation, $\hat{\cdot}$, is unary. In addition, only certain bijections can be subjected to the $\circ$ and $\hat{\cdot}$ operations.

Let finite sets A, B, C, D and bijections $f : A \to B$ and $g : C \to D$ be given. Also, let $A \dot{\cup} B$ denote the disjoint union of A and B. Then we can construct bijections that we denote $f + g$ and $f \cdot g$ as follows. The bijection $f + g : A \dot{\cup} C \to B \dot{\cup} D$ is defined by

$$(f+g)(x) = \begin{cases} f(x) & x \in A \\ g(x) & x \in C. \end{cases}$$

The bijection $f \cdot g : A \times C \to B \times D$ is defined by

$$f \cdot g(x, y) = (f(x), g(y)).$$

It is easy to see that these are in fact bijections; in fact, $(f+g)^{-1} = f^{-1} + g^{-1}$ and $(f \cdot g)^{-1} = f^{-1} \cdot g^{-1}$.

In the special case that $B = C$; that is, $f : A \to B$ and $g : B \to D$, then we can also construct a bijection $g \circ f : A \to D$ by composing f and g.

Finally, if $f : A \dot{\cup} B \to C \dot{\cup} B$ is a bijection, we can define $\hat{f} : A \to C$, which implements the Garsia-Milne involution principle [19], recursively as follows:

$$\hat{f}(x) = \begin{cases} f(x) & f(x) \in C \\ \hat{f}(f(x)) & f(x) \in B \end{cases}.$$

Although this initially appears to be a circular definition, this is not the case.

Lemma 2.1. *The function $\hat{f}(x)$ is well-defined and is a bijection.*

Proof. If, for every $x \in A$, there exists n such that $f^{(n)}(x) \in C$, then $\hat{f}$ is well-defined. Suppose for purposes of contradiction that, for some $x \in A$, there is no such n. Then $f^{(n)}(x) \in B$ for all $n \geq 1$. Let $m > |B|$, and let $S = \left\{f^{(k)}(x)\right\}_{k=1}^{m}$. Then $S \subseteq B$, so $|S| \leq |B| < m$. Therefore, there are two values $0 \leq k_1 < k_2 \leq m$ such that $f^{(k_1)}(x) = f^{(k_2)}(x)$. We may assume that k_1 and k_2 are the smallest two values with this property. If $k_1 = 0$, then $f^{(k_1)}(x) = x = f^{(k_2)}(x)$. But $x \notin B$, whereas $f^{(k_2)}(x) \in B$; this is a contradiction. So $k_1 > 0$. But now $f^{(k_1-1)}(x) \neq f^{(k_2-1)}(x)$, because k_1 and k_2 were minimal. This contradicts the fact that f is a bijection, so $\hat{f}$ is well-defined.

We now show that $\hat{f}$ is injective. Suppose $\hat{f}(x) = \hat{f}(y)$. This means there exist m and n such that $f^{(m)}(x) = f^{(n)}(y) \in C$, but for all $1 \leq i < m$ and $1 \leq j < n$, $f^{(i)}(x) \in B$ and $f^{(j)}(y) \in B$. If $m > n$ then we have the contradiction that $f^{(n-m)}(y) = f^{(m)}(x)$ must be in both B and C. So $m = n$, which means that $x = y$.

Finally, $\hat{f}$ is bijective because an injective function from a set to another set of the same size is bijective. And, because $|A \dot{\cup} B| = |C \dot{\cup} B|$, we have $|A| = |A \dot{\cup} B| - |B| = |C \dot{\cup} B| - |B| = |C|$. $\square$

2.2.2 Bijectification

A *bijectification* of an equation $a = b$, where a and b are algebraic expressions, is a bijection $f : A \to B$ such that $|A| = a$ and $|B| = b$ in a “natural” way. Exactly what “natural” means is a matter of taste and may vary depending on the problem at hand. However, we adopt the following convention when we deal with identities on binomial coefficients:

1. Expressions a and b must be written using only the operations $+$ and $\cdot$, and entities from $\mathbb{N}$ and $\{\binom{n}{k}\}_{0 \leq k \leq n}$.
2. The cardinality of $[n]$ is naturally equal to n, and the cardinality of $\binom{[n]}{k}$ is naturally equal to $\binom{n}{k}$.
3. If A_i is naturally equal to a_i for $1 \leq i \leq r$, then $\dot{\bigcup}_{i=1}^{r} A_i$ is naturally equal to $\sum_{i=1}^{r} a_i$, and $\prod_{i=1}^{r} A_i$ is naturally equal to $\prod_{i=1}^{r} a_i$.

For example, $[3] \dot{\cup} [1]$ is naturally equal to $3 + 1$, but $[3] \dot{\cup} [1]$ is not naturally equal to 4. We can, however, bijectify the equation $3 + 1 = 4$ by finding a bijection $f : [3] \dot{\cup} [1] \to [4]$ (which is an easy task).

For a slightly more complicated example, consider the equation $5 \cdot 5 + 2 \cdot 6 = 6 \cdot 6 + 1$. To bijectify this equation, we would need to find a bijection $f : ([5] \times [5]) \dot{\cup} ([2] \times [6]) \to ([6] \times [6]) \dot{\cup} [1]$.

We may also bijectify a family of equations. A bijectification of a one-parameter family $\{a_n = b_n\}_{n=n_0}^{\infty}$ is a sequence of bijections $\{f_n : A_n \to B_n\}_{n=n_0}^{\infty}$ such that $|A_n| = a_n$ and $|B_n| = b_n$ naturally. Here we allow the parameter n to appear in the expressions a_n and b_n. In addition, we will also allow expressions of the form $(an + b)$ to appear, and to be naturally in bijection with $[an + b]$.

For example, a bijectification of the equations $n \cdot n = (n-1) \cdot (n-1) + (2n-1)$ would consist of bijections $f_n : [n] \times [n] \to ([n-1] \times [n-1]) \dot{\cup} [2n-1]$ for $n \geq 2$.

2.2.3 Substitution

Suppose we have bijections $f : A \dot{\cup} D \to C$ and $g : B \to D$. Then there is a natural way to construct a bijection from $A \dot{\cup} B \to C$. First, let $\iota_A : A \to A$ be the identity bijection on A. Then let $g' = \iota_A + g$. Now, if $h = f \circ g'$, h is the desired bijection. We will write $h = f[g]$, which is pronounced "f substitute g". In more generality, if $\mathcal{S}, \mathcal{T}$, and $\mathcal{U}$ are expressions consisting of sets combined by the operations $\dot{\cup}$ and $\times$, $\mathcal{U}$ is a subexpression of $\mathcal{S}$, and $f : \mathcal{S} \to X$ and $g : \mathcal{T} \to \mathcal{U}$, let $\mathcal{S}'$ be $\mathcal{S}$ with $\mathcal{T}$ substituted for $\mathcal{U}$. then $f[g]$ is defined by $f[g] = f \circ g'$, where $g' : \mathcal{S} \to \mathcal{S}'$ is defined by

$$g'(x) = \begin{cases} g(x) & x \in \mathcal{T} \\ x & x \notin \mathcal{T} \end{cases}.$$

Similarly, if $\mathcal{S}, \mathcal{T}$, and $\mathcal{U}$ are expressions consisting of sets combined by the operations $\dot{\cup}$ and $\times$, $\mathcal{U}$ is a subexpression of $\mathcal{S}$, and $f : X \to \mathcal{S}$ and $g : \mathcal{U} \to \mathcal{T}$, then we define $[g]f$, pronounced "g substituted after f," by $[g]f = (f^{-1}[g^{-1}])^{-1}$.

2.2.4 Translation

The translation method is best illustrated by an example. We bijectify the identity

$$n^2 = (n-1)^2 + (2n-1), \qquad n \geq 2$$

by building up a family of bijections from our elementary operations, substitution, and simple bijections.

First, we prove the identity algebraically, avoiding the use of subtraction and division, as follows.

$$\begin{aligned}
n &= n \\
n \cdot n &= n \cdot n \\
n &= (n-1)+1 \\
((n-1)+1) \cdot n &= n \cdot n \\
((n-1)+1) \cdot ((n-1)+1) &= n \cdot n \\
((n-1)+1) \cdot (n-1) + ((n-1)+1) \cdot 1 &= n \cdot n \\
(n-1) \cdot ((n-1)+1) + 1 \cdot ((n-1)+1) &= n \cdot n \\
((n-1) \cdot (n-1) + (n-1) \cdot 1) + 1 \cdot ((n-1)+1) &= n \cdot n \\
((n-1) \cdot (n-1) + (n-1) \cdot 1) + (1 \cdot (n-1) + 1 \cdot 1) &= n \cdot n \\
(n-1) \cdot (n-1) + (n-1) \cdot 1 + 1 \cdot (n-1) + 1 \cdot 1 &= n \cdot n \\
(n-1) \cdot (n-1) + (n-1) + 1 \cdot (n-1) + 1 \cdot 1 &= n \cdot n \\
(n-1) \cdot (n-1) + (n-1) + (n-1) + 1 \cdot 1 &= n \cdot n \\
(n-1) \cdot (n-1) + (n-1) + (n-1) + (n-1) + 1 &= n \cdot n \\
(n-1) \cdot (n-1) + ((n-1) + (n-1) + 1) &= n \cdot n \\
(n-1) + (n-1) + 1 &= (2n-1) \\
(n-1) \cdot (n-1) + (2n-1) &= n \cdot n
\end{aligned}$$

To bijectify this proof, we first need to introduce some very simple kinds of bijections that can be constructed in a mechanical manner.

An *identity bijection* is a bijection $\iota : A \to A$.

A *sum bijection* is a bijection $\mathrm{Sum} : [m]\dot{\cup}[n] \to [m+n]$.

A *left distribution bijection* is a bijection $\mathrm{DisL} : A \times (B\dot{\cup}C) \to (A \times B)\dot{\cup}(A \times C)$.

A *right distribution bijection* is a bijection $\mathrm{DisR} : (A\dot{\cup}B) \times C \to (A \times C)\dot{\cup}(B \times C)$.

A *commutation bijection* is a bijection $\mathrm{Comm}^{+} : A\dot{\cup}B \to B\dot{\cup}A$ or $\mathrm{Comm}^{\times} : A\times B \to B \times A$.

A *one-eliminating bijection* is a bijection $\mathrm{Elim} : [1] \times A \to A$.

A *sum-associating bijection* is a bijection

$$\mathrm{Assoc}^{+} : \dot{\bigcup}_{i=1}^{n} A_i \to A_1\dot{\cup}\cdots\dot{\cup}A_{j-1}\dot{\cup}\left(\dot{\bigcup}_{i=j}^{k} A_i\right)\dot{\cup}A_{k+1}\dot{\cup}\cdots\dot{\cup}A_n.$$

We stress that these symbols do not represent particular bijections, but rather families of bijections that are all defined in very similar ways. The Maple package BijBuilder is available on the author's website [1] and excerpts can be seen in Appendix B contains Maple functions that can produce these bijections in particular cases. [2] For example, every choice of A, B, and C yields a different bijection of type DisL; the software can produce the correct bijection if it is given the sets A, B, and C.

We now reprint the proof above. This time, however, each line is followed by an annotation. The annotation defines a new bijection, using the bijection operators, in terms of the previous bijections and the "basic" bijections that were introduced above. Each bijection provides a bijectification of the equation printed on the same line. On the final line, the bijection B_{15} is defined; this is the desired bijectification of the identity. To simplify the presentation, the "basic" bijections are not named explicitly. Instead, the name of the family of basic bijections is given. For example, the sixth line states that $B_6 = B_5[\mathrm{DisL}]$. This means that we find the bijection D in the family DisL so that

$$D : ([n-1]\dot{\cup}[1]) \times ([n-1]\dot{\cup}[1]) \to (([n-1]\dot{\cup}[1]) \times [n-1])\dot{\cup}(([n-1]\dot{\cup}1) \times [1]),$$

and then put $B_6 = B_5[D]$.

[1] `http://math.rutgers.edu/~nbs48/BijBuilder`

[2] An earlier Maple package called "BijTools" containing some of these features was implemented by Wood and Zeilberger [49] The present package uses a different implementation that is better suited for bijections between very large sets, such as those that occur in the bijectification of Franel's identity for larger n.

$n = n$

$$B_1 = \iota_{[n]}$$

$n \cdot n = n \cdot n$

$$B_2 = B_1 \cdot B_1$$

$n = (n-1)+1$

$$B_3 \in \mathrm{Sum}^{-1}$$

$((n-1)+1) \cdot n = n \cdot n$

$$B_4 = B_2[B_3]$$

$((n-1)+1) \cdot ((n-1)+1) = n \cdot n$

$$B_5 = B_4[B_3]$$

$((n-1)+1) \cdot (n-1) + ((n-1)+1) \cdot 1 = n \cdot n$

$$B_6 = B_5[\mathrm{DisL}]$$

$(n-1) \cdot ((n-1)+1) \cdot 1 \cdot ((n-1)+1) = n \cdot n$

$$B_7 = B_6[\mathrm{Comm}^{\times}][\mathrm{Comm}^{\times}]$$

$((n-1) \cdot (n-1) + (n-1) \cdot 1) + 1 \cdot ((n-1)+1) = n \cdot n$

$$B_8 = B_7[\mathrm{DisL}]$$

$((n-1) \cdot (n-1) + (n-1) \cdot 1 + (1 \cdot (n-1) + 1 \cdot 1) = n \cdot n$

$$B_9 = B_8[\mathrm{DisL}]$$

$(n-1) \cdot (n-1) + (n-1) \cdot 1 + 1 \cdot (n-1) + 1 \cdot 1 = n \cdot n$

$$B_{10} = B_9[(\mathrm{Assoc}^{+})^{-1}][(\mathrm{Assoc}^{+})^{-1}]$$

$(n-1) \cdot (n-1) + (n-1) + 1 \cdot (n-1) + 1 \cdot 1 = n \cdot n$

$$B_{11} = B_{10}[\mathrm{Comm}^{\times}][\mathrm{Elim}]$$

$(n-1) \cdot (n-1) + (n-1) + (n-1) + 1 \cdot 1 = n \cdot n$

$$B_{12} = B_{11}[\mathrm{Elim}]$$

$(n-1) \cdot (n-1) + (n-1) + (n-1) + (n-1) + 1 = n \cdot n$

$$B_{13} = B_{12}[\mathrm{Elim}]$$

$(n-1) \cdot (n-1) + ((n-1) + (n-1) + 1) = n \cdot n$

$$B_{14} = B_{13}[\mathrm{Assoc}^{+}]$$

In this case, the final bijection $B_{15} : ([n-1] \times [n-1]) \dot{\cup} [2n-1] \to [n] \times [n]$ can be expressed succinctly as follows:

$$B_{15}(x) = \begin{cases} (b, a) & \text{if } x \in [n-1] \times [n-1] \text{ and } x = (a, b) \\ (n, x) & \text{if } x \in [2n-1] \text{ and } x \leq n-1 \\ (x-n+1, n) & \text{if } x \in [2n-1] \text{ and } x \geq n. \end{cases}$$

When this method is used to prove a more complex identity, the explicit form of the resulting bijection may be too unwieldy to write in this way.

2.3 A bijectifiable proof of Franel's identity

We now give a proof of Franel's recurrence (2.1) that can be bijectified with this method. The actual bijections themselves are too unwieldy to be printed here. However, they can be produced by the function bijFranel in the accompanying BijBuilder package.

In bijectifying Franel's identity it is convenient to use the following identity as a building block, which requires us to give a bijective proof for that identity.

Lemma 2.2.

$$(n+1)\binom{n}{k}\binom{n}{k-1} = n\binom{n-1}{k-1}\binom{n+1}{k}.$$

Proof. The House of Representatives has $(n+1)$ Federalists and n Whigs as members, and a budget committee must be formed with k Whigs and k Federalists, one of whom is designated chairperson of the committee. The left side counts the number of ways to do this if the chairperson is a Federalist (first choose the chair, then the Whigs, then the remaining Federalists). The right side counts the number of ways to do this if the chairperson is a Whig (first choose the chair, then the remaining Whigs, then the Federalists). A bijection between the two sides is as follows. If we have a Federalist chair whose name is rth in alphabetical order among the Federalists on the committee, replace him or her with the rth Whig from the committee, in alphabetical order, and vice versa. This is clearly a bijection, so the two sides are equal. $\square$

We will now give an algebraic proof for Franel's identity that can be bijectified by the method previously discussed, using as building blocks the basic bijections of the previous section, Pascal's identity $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$, the symmetry identity $\binom{n}{k} = \binom{n}{n-k}$, and the bijection of Lemma 2.2. This is done in the Maple package. In doing so, some care is required with the boundary conditions on the summations; for convenience, we have swept that under the rug in what follows.

Theorem 2.3. *(Franel) Let $A(n) = \sum_k \binom{n}{k}^3$. Then for all $n \geq 2$,*

$$\begin{aligned} &n^2A(n) + 2(5n^2 - 7n + 2)A(n-2) \\ &= 2n^2A(n-1) + 6(3n^2 - 5n + 2)A(n-2) + (5n^2 - 7n + 2)A(n-1). \end{aligned}$$

Proof. Let $B(n) = \sum_k \binom{n}{k}^2 \binom{n}{k-1}$. Applying Pascal's identity and symmetry, we get

$$\begin{aligned} A(n) &= \sum_k \left(\binom{n-1}{k} + \binom{n-1}{k-1} \right)^3 \\ &= \left[\sum_k \binom{n-1}{k}^3 + \sum_k \binom{n-1}{k-1}^3 \right] \\ &\quad + 3 \left[\sum_k \binom{n-1}{k}^2 \binom{n-1}{k-1} + \sum_k \binom{n-1}{k} \binom{n-1}{k-1}^2 \right] \\ &= 2A(n-1) + 6B(n-1) \qquad \text{(1a)} \end{aligned}$$

Applying Pascal's Identity, symmetry, and Lemma 2.2, we get

$$\begin{aligned}
(n+1)^2B(n) &= \sum_k (n+1)^2\binom{n}{k}^2\binom{n}{k-1} \\
&= \sum_k n(n+1)\binom{n}{k}\binom{n-1}{k-1}\binom{n+1}{k} \\
&= \sum_k n(n+1)\binom{n}{k}\binom{n-1}{k-1}\left(\binom{n}{k}+\binom{n}{k-1}\right) \\
&= \sum_k n(n+1)\binom{n}{k}^2\binom{n-1}{k-1} + \sum_k n(n+1)\binom{n}{k}\binom{n}{k-1}\binom{n-1}{k-1} \\
&= \sum_k n(n+1)\left(\binom{n-1}{k}^2+2\binom{n-1}{k}\binom{n-1}{k-1}+\binom{n-1}{k-1}^2\right)\binom{n-1}{k-1} \\
&\quad + \sum_k n^2\binom{n-1}{k-1}^2\binom{n+1}{k} \\
&= \sum_k n(n+1)\left(\binom{n-1}{k}^2\binom{n-1}{k-1}+2\binom{n-1}{k-1}^2\binom{n-1}{k}+\binom{n-1}{k-1}^3\right) \\
&\quad + \sum_k n^2\binom{n-1}{k-1}^2\left(\binom{n-1}{k}+2\binom{n-1}{k-1}+\binom{n-1}{k-2}\right) \\
&= 3n(n+1)B(n-1)+n(n+1)A(n-1) \\
&\quad + \sum_k n^2\left(\binom{n-1}{k-1}^2\binom{n-1}{k}+2\binom{n-1}{k-1}^3+\binom{n-1}{k-1}^2\binom{n-1}{k-2}\right) \\
&= 3n(n+1)B(n-1)+n(n+1)A(n-1)+2n^2B(n-1)+2n^2A(n-1) \\
&= (5n^2+3n)B(n-1)+(3n^2+n)A(n-1) \qquad \text{(2a)}
\end{aligned}$$

Similarly, we can prove the recurrences

$$A(n-1) = 2A(n-2)+6B(n-2) \qquad \text{(1b)}$$

and

$$n^2B(n-1) = (5n^2-7n+2)B(n-2)+(3n^2-5n+2)A(n-2). \qquad \text{(2b)}$$

Multiplying (1b) by $(5n^2-7n+2)$ and (2b) by 6, adding, and canceling the common term gives

$$6n^2B(n-1)+2(5n^2-7n+2)A(n-2) = 6(3n^2-5n+2)A(n-2)+(5n^2-7n+2)A(n-1). \qquad \text{(3b)}$$

Returning to (1a), we can multiply by n^2, and then add $2(5n^2-7n+2)A(n-2)$ to both sides to get

$$n^2A(n)+2(5n^2-7n+2)A(n-2) = 2n^2A(n-1)+6n^2B(n-1)+2(5n^2-7n+2)A(n-2).$$

Then, substituting for $6n^2B(n-1)+2(5n^2-7n+2)A(n-2)$ as in (3b), we have

$$\begin{aligned}&n^2A(n)+2(5n^2-7n+2)A(n-2)\\ &= 2n^2A(n-1)+6(3n^2-5n+2)A(n-2)+(5n^2-7n+2)A(n-1).\end{aligned}$$

This is the Franel recurrence, as we hoped. □

2.4 Further automation

Many identities involving hypergeometric terms can be proved algebraically using Sister Celine's method [16, 28]. Those proofs could be bijectified with this method in an almost automatic way. The only input required from a human would be bijective versions of the defining relations on the hypergeometric term.

For example, if $T(n,k) = \binom{n}{k}$, then Sister Celine's method produces an algebraic proof of Pascal's identity, $T(n,k) = T(n-1,k) + T(n-1,k-1)$, using only basic algebra and the relations

$$(n-k)T(n,k) = nT(n-1,k) \qquad \text{and} \qquad kT(n,k) = (n-k+1)T(n,k-1).$$

Given bijective proofs of these two equations, the entire proof can be bijectified in a mechanical way. However, at present, a human would be required to provide these fundamental bijections.

Chapter 3

123-avoiding words

3.1 Introduction

Recall that a word $w = w_1 \dots w_n$ in an ordered alphabet contains a *pattern* σ (a certain permutation of $\{1, ..., k\}$) if there exist

$$1 \leq i_1 < i_2 < \cdots < i_k \leq n$$

such that the subword $w_{i_1} \dots w_{i_k}$ is *order isomorphic* to σ; in other words $w_{i_1}, \dots, w_{i_k}$ are distinct, and if you replace the smallest entry by 1, the second smallest entry by 2, etc., you get σ. (If two entries are tied, they receive the same number; in particular, this means that no subword with a repeated entry is order isomorphic to σ.)

For example, the word *mathisfun* contains the pattern 132, since (inter alia) the subword *hsn* is order-isomorphic to 132 (under the usual lexicographic order).

In a remarkable PhD thesis, under the guidance of guru Herbert S. Wilf, Alexander Burstein [7] initiated the study of *forbidden patterns* (alias *Wilf classes*) in *words*, extending the very active and fruitful research on forbidden patterns in *permutations* initiated by Donald Knuth, Rodica Simion, Richard Stanley, Herbert Wilf, and others. For the current state of the art of the latter, see [46]. Burstein's pioneering thesis was extended by quite a few people, and the current knowledge is described in the lucid and insightful research monographs [25] and [27]. A systematic approach for computer-assisted enumeration of words avoiding a given set of patterns, extending the work of Zeilberger and Vatter for permutations (see [53] and its references), was initiated by Lara Pudwell ([30]). Some of the recent work (e.g. [21]) is phrased in the equivalent language of *ordered set partitions.* This equivalence is cleverly used in Anisse Kasraoui's ([26]) recent article.

Most of this work concerns the set of *all* words avoiding a pattern. In a very interesting recent paper by Godbole et al. (GGHP) [21], the authors consider (in the equivalent language of ordered set partitions), among other problems, the problem of enumerating 123-avoiding words of length $2n$ where each of the n letters $\{1, 2, \ldots, n\}$ occurs exactly twice, and conjectured a certain second-order linear recurrence equation with polynomial coefficients. They apparently did not realize that, in their case, it was possible to justify it by a (fully rigorous, or at least rigorizable) *hand-waving* argument. By general "holonomic nonsense" ([50]) it is known beforehand that there is *some* such linear recurrence, and it is possible to bound the order, thereby justifying, *a posteriori*, the guessed recurrence, provided that it is checked for sufficiently many initial values. A more direct proof was given by Chen, Dai, and Zhou [9], who proved the stronger statement that the generating function is algebraic, and even found the defining equation explicitly:

$$1 - (2x+1)\,F^2 + x\,(x+4)\,F^4 = 0. \tag{3.1}$$

Using Comtet's algorithm ([10], see also [40]) for deducing, out of the algebraic equation, a linear differential equation for the generating function, and hence a linear recurrence for the sequence itself, Chen, Dai and Zhou proved the GGHP conjecture directly.

We will generalize this and prove that, for **every** positive integer r, the ordinary generating function enumerating 123-avoiding words of length rn where each of the n letters of $\{1, 2, \ldots, n\}$ occurs exactly r times, is algebraic, and present an algorithm for finding the defining equation. Alas, since at the end it uses the memory-heavy, and exponential time, Buchberger's algorithm for finding Gröbner bases, our computer (running Maple) only agreed to explicitly find the next-in-line, the analogous equation for $r = 3$:

$$(4x+1)^2 + \left(64x^2+48x-1\right)F^2 - 2x\left(128x^2+108x+27\right)F^4 - 16x^2\,(32x+27)\,F^6$$
$$+\,x^2\,(32x+27)^2\,F^8 = 0.$$

This took less than a second, but the case $r = 4$ took about an hour. Here is the minimal algebraic equation satisfied by the generating function, let's call it F, whose coefficient of x^n is the number of 123-avoiding words with $4n$ letters with 4 occurrences of each i $(1 \le i \le n)$:

$$
\begin{aligned}
&x^3 (5x-256)^4 (4x+1)^4 F^{16} \\
&+ 4x^3 (85x+58)(5x-256)^3 (4x+1)^3 F^{14} \\
&+ 2x^2 \left(200x^4+11845x^3+8658x^2+6503x+256\right)(5x-256)^2 (4x+1)^2 F^{12} \\
&+ 4x^2 (5x-256)(4x+1)(25500x^5-977800x^4+15739435x^3+9911721x^2 \\
&+ 2082455x+138496)F^{10} \\
&+ x(60000x^8+2772000x^7-471787725x^6+11351360680x^5+15348867846x^4 \\
&+ 7091445146x^3+1387805641x^2+96468480x-458752)F^8 \\
&+ 4x(127500x^7-6439500x^6+28100475x^5+187145995x^4+58215739x^3 \\
&- 5955159x^2-2743199x-108800)F^6 \\
&+ (10000x^8+628250x^7-57924600x^6+1098116930x^5+827342646x^4 \\
&+ 223797652x^3+24970546x^2+842512x+1024)F^4 \\
&+ (42500x^7-1521500x^6-6516800x^5-7480160x^4-276672x^3+461716x^2 \\
&+ 49271x-1024)F^2 \\
&+ x(x+1)^2 \left(25x^2+65x+11\right)^2 = 0.
\end{aligned}
$$

We didn't even try the case $r = 5$.

However, since we know once again (now even without using Zeilberger's holonomic theory) that the generating function is D-finite, since it has the stronger property of being algebraic, it justifies **rigorously** guessing a linear recurrence equation with polynomial coefficients, which enables one to compute, in linear time, any term of the enumerating sequence. We succeeded, using our algorithm, to be described below (which in particular enables a very fast enumeration of many terms of the enumerating sequences), in discovering such recurrences for $1 \le r \le 5$, and using [14] we (or rather our beloved servant, Shalosh B. Ekhad, running Maple) found precise asymptotics for these cases. This enables us to formulate the following intriguing conjecture (which has now been proved; see the Addendum below):

Conjecture 3.1. *Let $w_r(n)$ be the number of 123-avoiding words of length rn with r occurrences of each of $\{1, \ldots, n\}$. Then*

$$\lim_{n\to\infty} \frac{w_r(n)}{w_r(n-1)} = (r+1)\,2^r \qquad .$$

More strongly, $w_r(n)$ is asymptotically $C_r \cdot ((r+1)2^r)^n \cdot n^{-3/2}$, where C_r is a 'nice' constant (probably $\frac{1}{\sqrt{\pi}}$ times the square-root of a rational number that depends 'nicely' on r).

Using the Maple package `Words123`, we proved it for $r \leq 5$ (but we were unable to guess an expression for C_r in terms of r from the five data points).

Currently the cases $1 \leq r \leq 4$ appear in the OEIS [38]. They are entries **A000108**, **A220097**, **A266736**, and **A266739**, respectively.

3.2 Some Crucial Background and Zeilberger's Beautiful Snappy Proof that 123-Avoiding Words are Equinumerous with 132-Avoiding Words

Burstein [7] proved that the number of all words in a given (ordered) alphabet of a given length n avoiding 123 is the same as the number of words avoiding 132, and hence, via trivial symmetry, all patterns of length 3 have the same enumeration. The stronger result that this is still true if one specifies the number of occurrences of each letter was first proved in [1], but the "proof from the book" appeared in the half-page gem [52]. We reproduce this proof here.

Proof. Define a mapping F on a word w in the alphabet $\{1, 2, \ldots, n\}$ recursively as follows. If w is empty, then $F(w) := w$. Otherwise, $i := w_1$; let W be the word obtained from w by first removing the first element, then replacing all letters larger than $i+1$ by $i+1$; and let s be the sub-sequence of w obtained by deleting the letters $\leq i$. Let $\bar{s}$ be the reverse of s. Let $V := F(W)$, and let U be the word obtained from V by replacing (in order) the letters that are $i+1$ by the members of $\bar{s}$. Finally let $F(w) := iU$.

F is an involution that sends 123-avoiding words to 132-avoiding ones, and vice versa. This follows from the fact that s above is non-increasing and non-decreasing respectively. Hence, for any vector of non-negative integers $(a_1, \ldots, a_n)$ amongst the $(a_1 + \cdots + a_n)!/(a_1! \cdots a_n!)$ words with a_1 1s, …, a_n ns, the number of those that avoid the pattern 123 equals the number of those that avoid 132,

It also follows that we have a quick recurrence that enables us to compute the number of such words, which we will call $A(a_1, \ldots, a_n)$:

$$A(a_1, \ldots, a_n) = \sum_{i=1}^{n} A(a_1, \ldots, a_{i-1}, a_i - 1, a_{i+1} + \cdots + a_n). \tag{3.2}$$

□

Another important consequence (which also follows from the Robinson-Schensted-Knuth algorithm) is that $A(a_1, \ldots, a_n)$ is *symmetric* in its arguments.

Because of the equinumeracy of all patterns of length 3, we can consider 231-avoiding words.

3.3 Definitions

Let $\mathcal{W}_r(n)$ be the set of 231-avoiding words in the alphabet $\{1, \ldots, n\}$ with exactly r occurrences of each letter.

Also, let $w_r(n)$ be the number of elements of $\mathcal{W}_r(n)$.

Define the "global set"

$$\mathcal{W}_r := \bigcup_{n=0}^{\infty} \mathcal{W}_r(n)$$

Let $g_r(x)$ be its *weight enumerator* with respect to the weight $w \to x^{length(w)}$. Note that $g_r(x) = f_r(x^r)$, where $f_r(x)$ is the generating function of the sequence $w_r(n)$,

$$f_r(x) := \sum_{n=0}^{\infty} w_r(n) x^n.$$

We will soon show how, for any *specific*, given, positive integer r, to obtain an algebraic equation (i.e. a polynomial $P_r(x, F)$ with integer coefficients such that $P_r(x, f_r(x)) = 0$), but let's srart with some warm-ups.

3.4 First Warm-Up: $r = 1$

$\mathcal{W}_1$ is the set of *all* permutations (of any length!) that avoid the pattern 231. Let the weight of a permutation π be $x^{length(\pi)}$. Consider any member π of that set. It may happen to be the empty permutation, of course (weight 1), or else it has a largest element; let's call that element n. All the entries to the left of n must be **smaller** than all the elements to the right of n (or else a 231 pattern would emerge), and each portion must be 231-avoiding in its own right. If the location of n is at the i-th place, then the portion to the left of n is a 231-avoiding permutation of $\{1, \dots, i-1\}$ and the portion to the right is a 231-avoiding permutation of $\{i, \dots, n-1\}$. Conversely, if π_1 and π_2 are 231-avoiding permutations of $\{1, \dots, i-1\}$ and $\{i, \dots, n-1\}$ respectively, then $\pi_1 n \pi_2$ is a 231-avoiding permutation of length n, since no trouble can arise by joining them. Hence,

$$f_1(x) = 1 + x f_1(x)^2 \quad ,$$

giving the good-old Catalan numbers.

The reader may note that we have seen this argument previously, in the section on nonlinear enumeration schemes.

3.5 Second Warm-Up: $r = 2$

The following argument is inspired by the beautiful proof in [9], but is phrased in such a way that will make it transparent how to generalize it for general r.

Let $g(x)$ be the weight-enumerator of $\mathcal{W}_2$. Recall that $\mathcal{W}_2$ is the set of all 231-avoiding words whose letters consist of $\{1, 1, \dots, n, n\}$ for some $n \geq 0$, and the weight is $x^{length(w)} = x^{2n}$).

(Note that $g(x) = f_2(x^2)$), so once we have $g(x)$ we will have $f_2(x)$ immediately.)

Consider a typical member of $\mathcal{W}_2$, and let n be its largest element (i.e. it is of length $2n$). Let i be the location of the **leftmost** occurrence of n. Notice, just as before, that the entries to the left of that first n must be less than or equal to the entries to the right of that n, and each portion is 231-avoiding in its own right, and conversely, if you place such 231-avoiding words with these entries to the left and right of that leftmost n, you will not cause any trouble, and get a 231-avoiding word whose entries are $\{1,1,2,2,\ldots,n,n\}$.

Case I: i is odd, i.e. $i=2j+1$.

Then the entries to the left of that first n are $\{1,1,\ldots,j,j\}$ and the entries to the right are $\{j+1,j+1,\ldots,n-1,n-1,n\}$. The generating function of the left part is our $g(x)$, but the entries to the right are a new combinatorial creature: a 231-avoiding word with all the letters occurring twice, except for one of them (which by symmetry can be taken to be '1') that only occurs once. So let's give the set $\mathcal{W}_2$ the new name $\mathcal{W}_2^{(0,0)}$, and let $\mathcal{W}_2^{(1,0)}$ be the union of the sets of 231-avoiding words on $\{1,2,2,3,3,\ldots,n,n\}$, for all $n\geq 0$. Let $g^{(1,0)}(x)$ be its weight-enumerator. Hence the total weight-enumerator of Case I is

$$xg^{(0,0)}(x)g^{(1,0)}(x).$$

(The x in front corresponds to the first n separating the two parts).

We will deal with $g_2^{(1,0)}(x)$ in due course, but now let's proceed to Case II.

Case II: i is even, i.e. $i=2j$.

Once again let its length be $2n$ (so the largest entry is n). The entries to the left of that first n are $\{1,1,\ldots,j-1,j-1,j\}$, and the entries to the right are $\{j,j+1,j+1,\ldots,n\}$. The generating function of the left part is the already familiar $g^{(1,0)}(x)$, but the right part is a new combinatorial creature; namely, a 231-avoiding word with all the letters occurring twice, **except** for *two* of them (that by symmetry may be taken to be the smallest and the largest) that only occur *once*. Let's call this set $\mathcal{W}_2^{(1,1)}$, and its weight-enumerator $g^{(1,1)}(x)$. Hence the total weight of Case II is $x\,g^{(1,0)}(x)\,g^{(1,1)}(x)$.

Combining the two cases, plus the empty permutation, leads to the following equation

$$g^{(0,0)}(x)=1+xg^{(0,0)}(x)g^{(1,0)}(x)+xg^{(1,0)}(x)g^{(1,1)}(x). \tag{3.3}$$

We have two new uninvited (and unenumerated) guests, $g^{(1,0)}(x)$ and $g^{(1,1)}(x)$. Using the same reasoning as above, the readers are welcome to convince themselves that

$$g^{(1,0)}(x)=xg^{(0,0)}(x)^2+xg^{(1,0)}(x)^2, \tag{3.4}$$

and

$$g^{(1,1)}(x) = xg^{(0,0)}(x)g^{(1,0)}(x) + xg^{(1,0)}(x)(1 + g^{(1,1)}(x)). \tag{3.5}$$

Solving this *algebraic scheme*, a system of three algebraic equations 3.3, 3.4, and 3.5 in the three "unknowns" $\{g^{(0,0)}(x), g^{(1,0)}(x)$, and $g^{(1,1)}(x)\}$, using Gröbner bases (though in this simple case it could be easily done by hand) gives an algebraic equation satisfied by $g^{(0,0)}(x)$, and hence, after replacing x^2 by x, the [9] equation for $f_2(x)$ mentioned above:

$$1 - (2\,x + 1)\,f_2(x)^2 + x\,(x + 4)\,f_2(x)^4 = 0.$$

3.6 The General Case

For $0 \le i \le j \le r - 1$ and $n \ge 0$, let $\mathcal{W}_r^{(i,j)}(n)$ be the set of 231-avoiding words of length $rn + i + j$, in the alphabet $\{1, 2, \ldots, n, n + 1, n + 2\}$, with i occurrences of the letter '1', j occurrences of '$n + 2$', and exactly r occurrences of the other n letters (i.e. $2, 3, \ldots, n + 1$), and let $\mathcal{W}_r^{(i,j)}$ be the union of $\mathcal{W}_r^{(i,j)}(n)$ over all $n \ge 0$.

By symmetry they have the same weight-enumerator as if *any* two letters have i and j occurrences respectively, and the remaining letters each occur exactly r times.

Using the same logic as above, we have the following $\binom{r+1}{2}$ equations, for $0 \le i \le j \le r - 1$, where below we make the convention that if $r > s$ then $g^{(r,s)} = g^{(s,r)}$.

$$g^{(i,j)}(x) = \delta_{i,0}\delta_{j,0} + x\sum_{t=0}^{r-1} g^{(i,t)}(x)g^{((r-t) \mod r,(j-1) \mod r)}(x) + \sum_{m=0}^{i-1} x^{m+1}g^{(i-m,j-1)}(x). \tag{3.6}$$

By eliminating $g^{(0,0)}(x)$, and replacing x^r by x, we get the equation of our object of desire $f_r(x)$. In fact, this equation typically has several solutions, and the right one is picked by plugging in the first few terms.

3.7 Guessing Linear Recurrences for our sequences

Now that we know, even without WZ-theory, that for every positive integer r, the generating function $f_r(x)$ is D-finite, since it has the much stronger property of being algebraic, we immediately know that the sequence itself, $\{w_r(n)\}$, is P-recursive in the sense of Stanley [40]; in other words, it satisfies *some* homogeneous linear recurrence equation with *polynomial* coefficients.

With a very large computer, one should be able to get the algebraic equation for quite a few r, and then use Comtet's algorithm (built-in in the Maple package `gfun`, procedure `algeqtodiffeq` followed by procedure `diffeqtorec`), to get a rigorously derived recurrence. Alas, because our system has $(r+1)r/2$ algebraic equations, and Gröbner bases are notoriously slow, we were only able to do two new cases explicitly, namely $r = 3$ and $r = 4$, mentioned above. But now that we know for sure that such recurrences exist, and it is easy to find a priori bounds for the order, it is easy to justify these empirically-derived recurrences, a posteriori.

But in order to guess complicated linear recurrences, one needs lots of data. Our algebraic scheme implies very fast *nonlinear* recurrences for the coefficients of $g^{(i,j)}(x)$, and in particular for $g^{(0,0)}(x)$, our primary interest. These turn out to be much faster than the "vanilla" linear recurrence for $A(a_1, \ldots, a_n)$ mentioned above.

3.8 The Maple package Words123

Everything (and more!) is implemented in the Maple package `Words123`, available directly from `http://www.math.rutgers.edu/\~{}zeilberg/tokhniot/Words123`, or via the home page of the article that was the genesis of this section, `http://www.math.rutgers.edu/\~{}zeilberg/mamarim/mamarimhtml/words123.html`, which also contains some sample input and output files.

3.9 The recurrences for $1 \leq r \leq 3$

For $r = 1$ we get the good-old Catalan numbers

$$-2\frac{(1+2n)\,w_1(n)}{n+2} + w_1(n+1) = 0.$$

For $r = 2$ we get a new proof of the GGHP [21] conjecture (first proved in [9])

$$-3\frac{(7n+12)(1+2n)(1+n)\,w_2(n)}{(2n+5)(7n+5)(n+2)} - \frac{\left(528+1426n+1215n^2+329n^3\right)w_2(n+1)}{2(2n+5)(7n+5)(n+2)} + w_2(n+2) = 0.$$

For $r = 3$ we get

$$-\frac{64}{3}\frac{(4n+1)(2n+3)(4n+3)(14n+25)(n+1)\,w_3(n)}{(3n+5)(1+2n)(3n+7)(14n+11)(n+2)}$$
$$-\frac{8}{3}\cdot\frac{\left(3975+20322n+39676n^2+37144n^3+16736n^4+2912n^5\right)w_3(n+1)}{(3n+5)(1+2n)(3n+7)(14n+11)(n+2)}+w_3(n+2)=0.$$

See the output file

`http://www.math.rutgers.edu/~zeilberg/tokhniot/oWords123c`

for the recurrences for $w_4(n)$ and $w_5(n)$.

3.10 The Asymptotics for $1 \le r \le 5$

$$w_1(n) = \frac{1}{\sqrt{\pi}}\cdot 4^n\cdot n^{-\frac{3}{2}}\left(1-\frac{9}{8}n^{-1}+\frac{145}{128}n^{-2}-\frac{1155}{1024}n^{-3}+O(n^{-4})\right),$$
$$w_2(n) = \frac{1}{\sqrt{\pi}}\cdot\frac{3\sqrt{3}}{7\sqrt{7}}\cdot 12^n\cdot n^{-\frac{3}{2}}\left(1-\frac{249}{392}n^{-1}+\frac{13255}{43904}n^{-2}-\frac{2674485}{17210368}n^{-3}+O(n^{-4})\right),$$
$$w_3(n) = \frac{1}{\sqrt{\pi}}\cdot\frac{1}{8}\cdot 32^n\cdot n^{-\frac{3}{2}}\left(1-\frac{33}{64}n^{-1}+\frac{1105}{8192}n^{-2}-\frac{27195}{524288}n^{-3}+O(n^{-4})\right),$$
$$w_4(n) = \frac{1}{\sqrt{\pi}}\cdot\frac{1}{6\sqrt{6}}\cdot 80^n\cdot n^{-\frac{3}{2}}\left(1-\frac{23}{48}n^{-1}+\frac{1621}{23040}n^{-2}-\frac{339199}{16588800}n^{-3}+O(n^{-4})\right),$$
$$w_5(n) = \frac{1}{\sqrt{\pi}}\cdot\frac{3\sqrt{3}}{125}\cdot 192^n\cdot n^{-\frac{3}{2}}\left(1-\frac{471}{1000}n^{-1}+\frac{389141}{10000000}n^{-2}-\frac{162387477}{50000000000}n^{-3}+O(n^{-4})\right).$$

3.11 Addendum

There were further developments after this article was first published on the ArXiv.

Robin Chapman kindly communicated to us the (at that time conjectured) expression $C_r = \frac{1}{\sqrt{\pi}}\cdot(6/(r^2+5r))^{3/2}$.

Zeilberger [12] proved that the generating functions enumerating words, avoiding $12\dots d$, with r copies of $1, 2, \dots, n$ are D-finite, and issued a challenge to prove the following:

Conjecture 3.2. *If $A_{d,r}(n)$ is the number of words of $1\dots d$-avoiding words with r copies of each of $1,\dots,n$, then*

$$A_{d,r}(n) \sim C_{r,d}\left(\binom{d+r-2}{d-2}(d-1)^r\right)^n\frac{1}{n^{((d-1)^2-1)/2}}, \qquad (3.7)$$

where $C_{r,d}$ is a constant.

Soon afterwards, the conjecture was proved by Guillaume Chapuy [8], who also provided exact formulas for $C_{r,d}$ (see the appendices), thus claiming Zeilberger's bounty of a \$125 donation to the OEIS, and proving Chapman's conjecture.

Subsequently, another one of Zeilberger's challenges in [12] was answered by Ferenc Balogh [2], who generalized Gessel's determinant formula 1.10 to the case of general r, and in this way re-derived the formulas of this chapter.

Chapter 4

Repeating patterns of low codimension

4.1 The increasing pattern

4.1.1 Preface

How many permutations are there of length googol+30 avoiding an increasing subsequence of length googol?

This number is way too big for our physical universe, but the number of permutations of length googol+30 that *contain* at least one increasing subsequence of length googol is

3 769 987 628 815 905 643 852 921 525 646 105 664 146 833 823 621 994 801 456 991 357 113 502 936 781 270 538 054 719 048 039 675 278 076 919 335 437 172 135 000 152 461 057 809 770 004 597 279 282 389 729 095 962 420 389 610 198 195 292 964 080 517 012 928 207 388 347 400 188 075 711 475 340 912 299 512 494 359 131 491 717 953 025 923 124 774 560 912 778 123 219 562 128 022 047 855 785 980 202 555 625 008 802 850 838 455 586 257 402 947 256 848 380 647 181 479 993 222 566 420 025 908 679 106 917 004 348 077 812 428 261 510 240 634 017 630 058 539 751 799 003 239 303 665 395 130 492 458 648 996 865 080 978 929 229 148 927 096 871 099 480 967 705 017 659 675 107 259 562 023 507 508 413 760 950 240 463 968 449 685 112 434 947 841 620 148 817 953 378 355 286 261 428 081 500 731 111 012 833 610 980 701 571 937 952 824 136 796 425 017 224 636 196 853 995 950 587 943 259 043 687 431 653 922 927 840 572 864 396 105 085 190 223 258 279 906 781 037 838 989 063 519 632 242 566 746 733 515 889 050 082 012 876 833 175 085 546 996 305 032 243 297 319 447 233 194 709 898 259 344 696 960 793 447 230 536 790 011 300 336 678 275 249 660 346 617 820 648 510 682 141 824 547 313 657 434 134 867 297 300 631 055 444 127 725 930 013 792 836 515 384 850 702

346 797 298 406 803 049 230 145 697 433 567 004 811 555 984 158 378 611 125 895 014

576 890 134 872 555 072 603 752 766 981 262 635 326 683 768 503 739 740 886 276 708

201 823 957 939 266 302 413 179 210 540 728 047 887 208 406 185 144 634 650 353 921

038 843 949 812 020 078 347 241 100 944 471 166 134 391 287 582 850 442 694 718 085

020 832 756 629 374 247 928 521 501 786 839 409 853 287 740 758 570 056 230 853 738

462 527 374 534 709 641 735 458 487 560 816 949 365 616 486 069 562 691 302 969 992

264 810 209 161 552 994 941 494 064 858 804 883 648 537 275 877 580 874 323 136 561

745 951 532 919 097 239 870 745 439 464 155 787 284 399 943 060 712 796 540 451 464

323 795 575 413 584 089 781 568 631 729 804 197 208 392 927 610 252 617 526 805 876

626 590 163 265 795 592 248 178 664 681 630 980 893 821 587 688 413 815 206 609 216

082 514 983 787 883 386 977 226 071 420 216 491 477 289 935 925 789 614 221 777 002

944 825 967 409 939 865 193 572 469 599 306 681 465 050 852 707 144 755 611 501 137

472 212 088 787 004 775 335 817 731 620 626 335 692 795 572 945 875 468 655 064 443

263 468 768 028 202 797 640 277 277 248 383 611 710 547 348 145 611 509 228 154 510

472 000 404 130 614 639 780 926 417 137 329 939 732 465 722 014 680 564 902 839 930

824 306 834 920 414 545 138 747 536 000 552 520 920 011 368 145 713 293 845 873 255

824 684 878 872 443 952 952 455 854 191 886 467 927 642 528 321 599 620 296 941 164

954 437 213 105 323 538 743 944 687 543 469 879 373 512 141 279 640 023 696 573 258

448 468 721 998 289 835 514 598 029 197 786 269 234 486 135 973 112 564 250 247 007

239 135 280 355 775 712 267 954 726 019 033 893 771 378 762 777 423 669 196 575 295

174 512 964 525 876 697 257 261 448 327 403 717 828 223 080 061 705 319 100 992 656

781 414 836 225 171 440 141 077 162 170 100 983 838 399 688 450 780 459 024 472 066

740 659 392 956 413 459 154 780 579 363 446 852 393 445 432 890 675 675 870 120 976

547 151 488 057 237 075 909 084 331 736 216 302 289 075 177 161 806 402 089 083 889

989 467 334 293 366 576 755 423 738 845 099 552 628 279 269 937 176 915 588 594 278

358 704 445 398 006 444 800 528 216 309 223 317 799 370 232 286 563 052 729 741 599

319 773 650 648 178 493 618 609 094 453 010 812 014 057 743 669 000 714 070 157 059

948 417 686 104 746 105 282 677 474 489 924 674 666 690 926 457 806 707 624 392 345

308 856 196 698 778 069 217 767 194 382 941 365 732 112 039 412 879 713 531 991 598

317 675 682 505 439 845 424 625 600 438 225 076 973 116 586 491 302 133 085 147 997

288 307 646 371 721 290 040 656 119 074 756 104 017 130 087 909 728 914 090 203 626

587 419 465 098 918 321 657 701 667 667 006 001 209 610 998 909 380 382 010 865 003

885 220 777 565 531 701 133 543 218 588 330 720 970 852 694 358 826 481 897 737 757

381 491 860 736 859 345 865 582 855 966 329 016 368 188 788 860 428 833 268 391 323

270 593 913 089 901 528 577 501 918 097 456 348 791 214 247 627 656 062 131 012 346

884 500 965 061 477 592 565 827 356 220 792 375 195 479 399 434 709 301 661 829 216

458 040 271 254 279 814 338 864 116 761 417 830 119 059 817 479 387 880 694 430 162

532 210 993 791 275 528 220 779 177 902 246 600 447 925 840 824 462 949 592 761 349

881 316 543 422 038 699 183 826 473 525 107 075 809 508 274 778 093 413 168 220 963

984 409 028 566 362 293 900 402 154 158 824 193 586 495 186 743 554 148 010 950 474

138 260 408 245 663 451 297 894 260 392 218 420 887 970 529 814 395 737 366 965 022

330 793 088 649 449 089 550 661 242 226 637 700 975 872 048 802 255 877 951 342 510

043 234 892 643 042 766 512 594 498 769 308 942 245 751 122 706 284 028 982 754 337

386 885 459 391 626 543 570 555 162 051 612 664 363 788 373 280 457 226 691 660 908

679 569 539 271 630 815 625 199 040 300 459 332 749 317 423 320 187 045 689 570 750

025 918 058 945 571 060 293 734 271 997 586 449 192 338 696 885 903 842 897 776 980

002 129 651 552 219 483 587 717 759 774 043 798 881 299 174 958 483 572 178 675 529

350 262 014 933 898 703 122 232 518 225 184 081 589 902 714 463 624 365 018 242 747

599 082 635 817 593 737 724 580 337 688 809 342 550 695 342 366 935 036 425 354 918

809 144 353 766 748 764 322 702 047 644 140 655 613 822 124 251 002 536 953 366 801

093 535 788 780 414 052 627 726 381 391 247 928 321 640 648 394 196 028 626 519 959

966 325 451 252 664 262 353 889 631 883 841 776 653 646 129 270 593 661 149 306 259

085 397 802 418 629 266 233 934 211 681 736 693 714 241 352 634 384 615 108 485 320

700 947 811 487 618 744 149 158 225 668 175 169 324 385 259 284 556 343 634 093 729

448 184 378 424 215 074 591 762 603 340 467 588 946 300 632 760 395 911 666 231 000

926 265 506 283 360 070 907 064 341 332 664 779 779 937 712 263 184 388 203 605 477

211 162 014 805 937 750 522 978 535 620 225 925 004 722 918 738 657 674 699 944 947

405 347 907 659 143 618 050 579 417 087 497 652 165 460 185 477 043 345 636 632 204

978 226 001 800 424 273 526 341 460 220 242 548 683 728 799 179 065 030 083 029 494

514 450 905 531 725 089 967 903 293 290 935 500 874 548 539 339 178 735 194 085 694

882 107 486 318 798 833 745 852 508 207 772 876 776 458 002 804 430 766 991 660 626 376 067 637 977 770 235 404 212 193 344 610 052 823 762 990 072 265 783 070 820 234 545 141 480 898 874 637 486 106 893 816 774 598 214 664 007 156 038 886 731 975 384 257 202 382.

Hence the number of permutations of length googol+30 *avoiding* an increasing subsequence of length googol is $(googol + 30)!$ *minus* the above small number.

4.1.2 Counting the "Bad Guys"

Recall that thanks to Robinson-Schensted [32], [34], the number of permutations of length n that do **not** contain an increasing subsequence of length d is given by

$$G_d(n) := \sum_{\substack{\lambda \vdash n \\ \#\mathrm{rows}(\lambda) < d}} f_\lambda^2 \quad , \tag{4.1}$$

where λ denotes a typical Young diagram, and f_λ is the number of standard young tableaux whose shape is λ.

Hence the number of permutations of length n that **do** contain an increasing subsequence of length d is

$$B_d(n) := \sum_{\substack{\lambda \vdash n \\ \#\mathrm{rows}(\lambda) \geq d}} f_\lambda^2 \quad .$$

Since the total number of permutations of length n is $n!$ ([3]), if we know how to find $B_d(n)$, we then know immediately $G_d(n) = n! - B_d(n)$, at least if we leave $n!$ alone as a factorial, rather than spell it out.

Recall that the *hooklength formula* (see, for example, theorem 6.5 of [5]) tells you that if λ is a Young diagram then

$$f_\lambda = \frac{n!}{\prod_{c \in \lambda} h(c)} \quad ,$$

where the product is over all the n cells of the Young diagram, and the *hook-length*, $h(c)$, of a cell $c = (i, j)$, is $(\lambda_i - i) + (\lambda'_j - j) + 1$, where λ' is the *conjugate* diagram, where the rows become columns and vice-versa.

Let r be a fixed integer, then for *symbolic* d, valid for $d \geq r - 1$, any Young diagram with at least d rows, and with $d + r$ cells, can be written, for some Young diagram

$\mu = (\mu_1, \ldots, \mu_r)$, with at most r cells, (where we add zeros to the end if the number of parts of μ is less than r) as

$$\lambda = (1 + \mu_1, \ldots, 1 + \mu_r, 1^{d-r+r'}),$$

where $r' = r - |\mu|$. For such a shape λ, with at least d rows,

$$\prod_{c\in\lambda} h(c) = \left(\prod_{c\in\mu} h(c)\right) \cdot ((d+r'+\mu_1)(d+r'-1+\mu_2)\cdots(d+r'-r+1+\mu_r)) \cdot (d-r+r')!. \tag{4.2}$$

Hence f_λ, that is $(d+r)!$ divided by the 4.2, is a certain specific number times a certain polynomial in d. Since, for a specific *numeric* r, there are only finitely many Young diagrams with at most r cells, the computer can find all of them, compute the polynomial corresponding to each of them, square it, and add-up all these terms, getting an explicit **polynomial** expression, in the variable d, for $B_d(d+r)$, the number of permutations of length $d+r$ that contain an increasing subsequence of length d. As we said above, from this we can find $G_d(d+r) = (d+r)! - B_d(d+r)$, valid for *symbolic* $d \geq r-1$.

4.1.3 Examples for small values of r

$$\begin{aligned}
B_d(d) &= 1\\
B_d(d+1) &= d^2 + 1\\
B_d(d+2) &= \frac{1}{2}\,d^4 + d^3 + \frac{1}{2}\,d^2 + d + 3\\
B_d(d+3) &= \frac{1}{6}\,d^6 + d^5 + \frac{5}{3}\,d^4 + \frac{2}{3}\,d^3 + \frac{19}{6}\,d^2 + \frac{31}{3}\,d + 11\\
B_d(d+4) &= \frac{1}{24}\,d^8 + \frac{1}{2}\,d^7 + \frac{25}{12}\,d^6 + \frac{19}{6}\,d^5 + \frac{29}{24}\,d^4 + 9\,d^3 + \frac{247}{6}\,d^2 + \frac{395}{6}\,d + 47\\
B_d(d+5) &= \frac{1}{120}\,d^{10} + \frac{1}{6}\,d^9 + \frac{31}{24}\,d^8 + \frac{14}{3}\,d^7 + \frac{823}{120}\,d^6 + \frac{67}{30}\,d^5 + \frac{653}{24}\,d^4 + \frac{959}{6}\,d^3\\
&\quad + \frac{10459}{30}\,d^2 + \frac{3981}{10}\,d + 239
\end{aligned}$$

For $B_d(d+r)$ for r from 6 up to 30, see

`http://www.math.rutgers.edu/~zeilberg/tokhniot/oGessel64a` .

4.1.4 Integer sequences

The sequence $G_3(n)$ is the greatest celeb in the kingdom of combinatorial sequences (the subject of an entire book([41]) by Ira Gessel's illustrious academic father, Richard Stanley), the super-famous **A000108**, the longest entry in Neil Sloane's legendary database [38]. $G_4(n)$, while not in the same league as the Catalan sequence, is still moderately famous, **A005802**. $G_5(n)$ is **A047889**, $G_6(n)$ is **A047890**, $G_7(n)$ is **A052399** , $G_8(n)$ is **A072131**, $G_9(n)$ is **A072132**, $G_{10}(n)$ is **A072133**, $G_{11}(n)$ is **A072167**, but $G_d(n)$ for $d \geq 12$ are absent (for a good reason, one must stop somewhere!). Also the flattened version of the triangle, $\{G_d(n)\}$ for $1 \leq d \leq n \leq 45$, is **A047887**. Using the polynomials $B_d(d+r)$, we computed the first $2d+1$ terms of $G_d(n)$ for $d \leq 30$. See `http://www.math.rutgers.edu/~zeilberg/tokhniot/oGessel64b` .

But this method can only go up to $2d+1$ terms of the sequence $G_d(n)$, and of course, the first $d-1$ terms are trivial, namely $d!$ (and the d-th term is $d!-1$). Can we find the first 100 terms (or whatever) for the sequences $G_d(n)$ for d up to 20, and beyond, **efficiently?**

4.1.5 Efficient Computer-Algebra Implementation of Ira Gessel's AMAZING Determinant Formula

Recall Ira Gessel's [20] famous expression for the generating function of $G_d(n)/n!^2$, canonized in Herb Wilf's epistle on experimental mathematics [48]. Here it is:

$$\sum_{n \geq 0} \frac{G_d(n)}{n!^2} x^{2n} = \det(I_{|i-j|}(2x))_{i,j=1,\ldots,d} \quad ,$$

in which $I_\nu(t)$ is (the modified Bessel function)

$$I_\nu(t) = \sum_{j=0}^{\infty} \frac{(\frac{1}{2}\,t)^{2j+\nu}}{j!(j+\nu)!} \quad .$$

Can we use this to compute the first 100 terms of, say, $G_{20}(n)$?

While computing *numerical* determinants is very fast, computing *symbolic* ones is a different story. First, do not get scared by the "infinite" power series. If we are only interested in the first N terms of $G_d(n)$, then it is safe to truncate the series up to

t^{2N}, and take the determinant of a $d \times d$ matrix with *polynomial entries*. If you used the vanilla determinant in a computer-algebra system such as Maple, it would be very inefficient, since the degree of the determinant is much larger than $2N$. But a little cleverness can make things more efficient. The Maple package `Gessel64`, available from

`http://www.math.rutgers.edu/~zeilberg/tokhniot/Gessel64` ,

accompanying this article, has a procedure `SeqIra(k,N)` that computes the first N terms of $G_k(n)$, using a division-free algorithm (see [33]) over an appropriate ring to compute the determinant in Gessel's famous formula.

```
SeqIra:=proc(k,N) local ira,t,i,j, R:
  R := table():
  R['0'] := 0:
  R['1'] := 1:
  R['+'] := '+':
  R['-'] := '-':
  R['*'] := proc(p, q): return add(coeff(p*q, t, i)*t**i, i=0..2*N): end:
  R['='] := proc(p, q): return evalb(p = q): end:
  ira:=expand(LinearAlgebra[Generic][Determinant][R](Matrix([seq([seq(Iv(abs(i-j),t,2*N)
                                                                              j=1..k-1)],
                                                                        i=1..k-1)]))):
  [seq(coeff(ira,t,2*i)*i!**2,i=1..N)]:
end:
```

In the above code, procedure `Iv(v,t,N)` computes the truncated modified Bessel function that shows up in Gessel's determinant, and it is short enough to reproduce here:

```
   Iv:=proc(v,t,N) local j:  add(t**(2*j+v)/j!/(j+v)!,j=0..trunc((N-v)/2)+1):
end:      .
```

Using this procedure, Shalosh B. Ekhad computed (in 4507 seconds) the first 100 terms of each of the sequences $G_d(n)$ for $3 \leq d \leq 20$, and could have gone much further.

See `http://www.math.rutgers.edu/~zeilberg/tokhniot/oGessel64c` .

4.2 General repeating patterns

The previous method successfully counted the permutations containing a long increasing pattern, where "long" is defined relative to the length of the permutation. Of course, there are many patterns other than the increasing pattern. What about other patterns that are "long" relative to the length of the permutation?

In order for this concept to make sense, we must have a whole family of patterns of successively increasing lengths, so that for an arbitrarily long permutation we have a pattern in the family that is almost as long. The following definition is perhaps the simplest way to construct such families.

Definition 4.1. *The family of repeating permutations generated by $\tau \in S_d$ is the sequence of permutations $\tau, \tau \oplus \tau, \tau \oplus \tau \oplus \tau, \ldots$. We will denote the kth permutation in this sequence as τ^k.*

The family of increasing patterns is a special case of this, where $\tau = 1$. (Of course, if $\tau = 12$, then we get the increasing even-length patterns; if $\tau = 123$, we get the increasing patterns of length divisible by 3, and so on. So sometimes the familly of repeating permutations generated by τ is contained in the family of repeating permutations generated by a different permutation τ'.)

Let $P_{\tau,r}(k)$ denote the set of permutations of length $dk + r$ that match the pattern τ^k. Let $p_{\tau,r}(k) = |P_{\tau,r}(k)|$. We will often omit τ when its value is understood.

Our main theorem states that $p_{\tau,r}(k)$ is eventually a polynomial of degree $2d$. This allows the polynomials $p_{\tau,r}$ to be guessed by computing only a finite number of terms. In addition, we will show that the first three coefficients of $p_{\tau,r}(kd)$ are independent of the choice of τ.

Theorem 4.2. *For sufficiently large k, $p_{\tau,r}(k)$ is given by a polynomial of degree $2d$.*

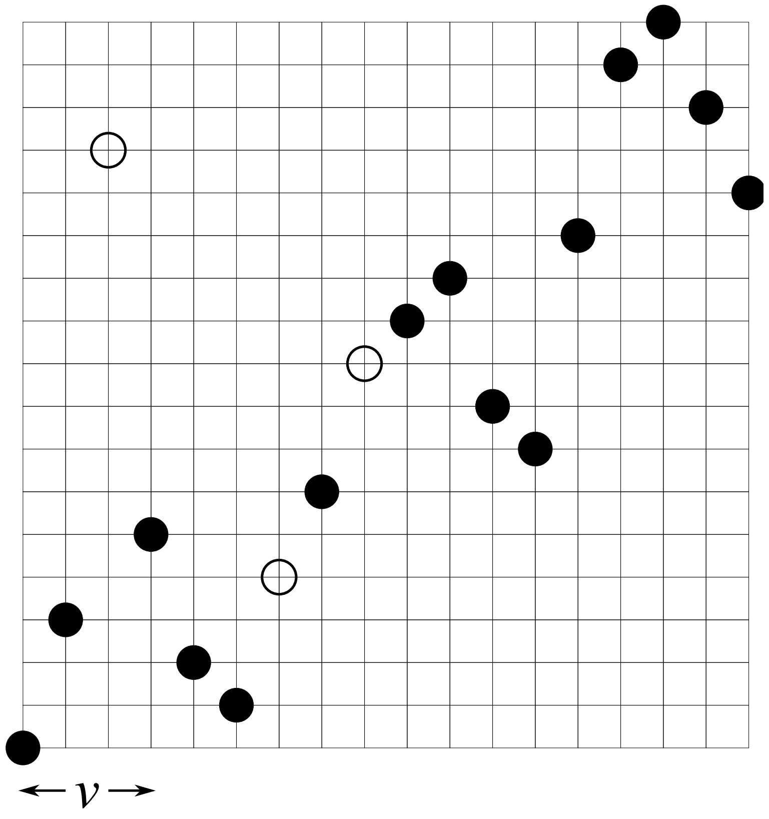
The proof of this theorem involves constructing a prefix-based enumeration scheme (see [51]) to count $P_{\tau,r}(k)$. We will then prove that the enumeration scheme is finite; after that, by examining the structure of the enumeration scheme, we will be able to deduce that $p_{\tau,r}(k)$ is eventually polynomial.

The proof itself requires some detailed constructions.

4.2.1 Definitions

Definition 4.3. *A (τ, r)-marking of π is a coloring of the elements of π so that r of them are white and the rest black, and such that the black elements form a τ^k pattern for some k. (See Figures 4.2.1 and 4.2.1.) For convenience, when τ and r are understood from context, we will simply refer to a marking of π.*

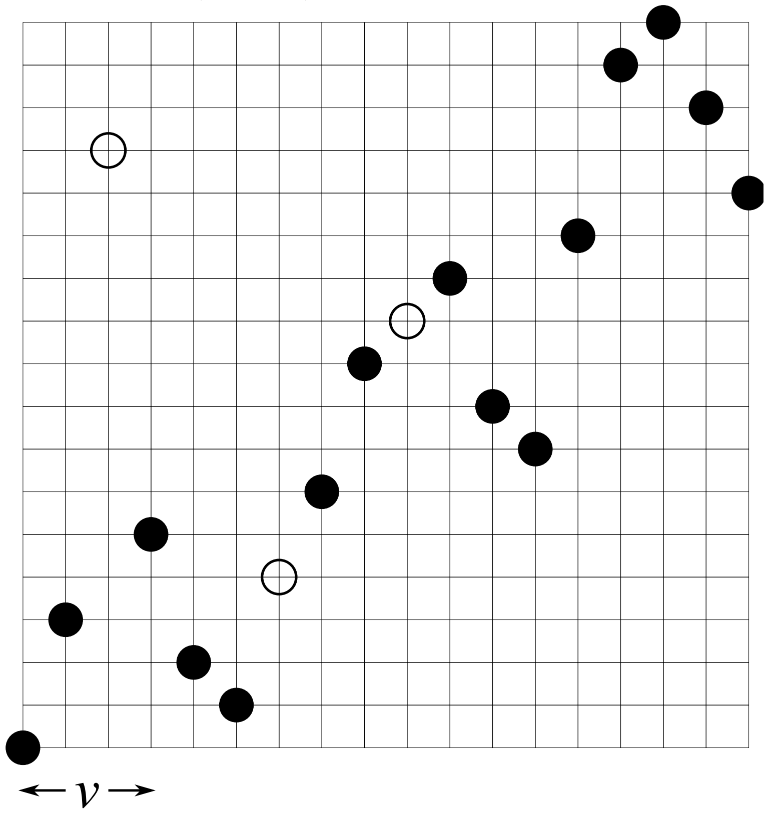
Figure 4.1: A $(14532, 3)$-marked permutation



Definition 4.4. *If $\tau \in S_d$, an r-insertion of τ^k is a (τ, r)-marking of some permutation π of length $kd + r$. Two r-insertions are said to be equivalent if they are markings of the same permutation.*

Definition 4.5. *A permutation affix is a vector of natural numbers that are all distinct. (In an appropriate context, we will refer to affixes as prefixes or suffixes for clarity.) For an affix v, let $h(v)$ be the largest element in v and let $\ell(v)$ be the number of elements*

Figure 4.2: A different $(14532, 3)$-marking of the permutation of Figure 4.2.1



in v. A complete extension of a prefix v is a permutation π whose first $\ell(v)$ elements are equal to v. A partial extension (or simply an extension) of a prefix v is a prefix w whose first $\ell(v)$ elements are equal to v. In these cases, we will say that w or π extends v.

The complement of a prefix v is $\mathbb{N}\backslash v$.

Definition 4.6. *If v and w are affixes, and $h(v) \leq \ell(v) + \ell(w)$, let $v|w$ denote the permutation that results from appending the $\ell(w)$ smallest elements from the complement of v to v in the same relative order as w. For example, $(1, 4, 3)|(1, 2, 3) = (1, 4, 3, 2, 5, 6)$.*

If v is a prefix and w is an extension of v, then $w - v$ denotes the vector of elements of w that are not in v.

Definition 4.7. *An extension w of v is s-diagonal if for $v + 1 \leq i \leq w$, $|w_i - i| \leq s$. An extension is s-subdiagonal if for $v + 1 \leq i \leq w$, $w_i - i \leq s$.*

Definition 4.8. *Given $\tau \in S_d$, let $Q_{\tau,r,v}(k)$ be the set of permutations of length $kd + r$*

that extend v and match τ^k. Let $q_{\tau,r,v}(k) = |Q_{\tau,r,v}(k)|$. As with P and p, we will often omit τ when it is understood.

In what follows, let $\tau \in S_d$ be given.

Lemma 4.9. *For two extensions w and w' of v that are not $d+r$-subdiagonal, $q_{r,w}(k) = q_{r,w'}(k)$.*

Proof. Let $\pi \in Q_{r,w}(k)$ and let x be such that $\pi = w|x$. Let $\pi' = w'|x$. We claim that $\pi' \in Q_{r,w'}(k)$. Let a marking of π be given. This restricts to markings of w and x. Note that the last element of w is colored white, because only elements within $d+r$ of the diagonal may be black.

Let w' be marked with the first $\ell(v)$ elements colored as in w, and the last element colored white. Let π' be colored with the first $\ell(w')$ elements colored as in w' and the rest as in x. We claim that this coloring places π' in $Q_{r,w'}(k)$. To see this, note that all the black elements in w' are still in the same relative order as in π, and all the black elements in x are still in the same relative order as in π. Furthermore, let i be the last element of w and j be the last element of w'. The only differences between π and π' are among elements at least $\min\{i, j\}$, and these are both larger than all the black elements in v. Hence all the black elements remain in the same order relative to each other, so $\pi' \in Q_{r,w'}(k)$.

The map from π to π' is clearly an injection, so $q_{r,w}(k) \leq q_{r,w'}(k)$. By symmetry, $q_{r,w}(k) = q_{r,w'}(k)$. This completes the proof. $\square$

Definition 4.10. *For any permutation $\pi \in S_n$, Let $a_{\tau,r,i}(\pi)$ be the minimum, over all markings of π, of the number of white elements in the first i elements of π. Let $b_{\tau,r,i,j}(\pi)$ be the minimum, over all markings of π, of the number of white elements that are either in the first i elements of π or are less than or equal to j.*

Definition 4.11. *For any prefix v, let $a_{\tau,r}(v)$ be the minimum, over all permutations $\pi \in S_n$ extending v, of $a_{\tau,r,i}(\pi)$, where $i = \ell(v)$. Let $A_{\tau,r}(v)$ be the set of permutations achieving this minimum.*

Let $b_{\tau,r}(v)$ be the minimum, over all permutations $\pi \in A_{\tau,r}(v)$, of $b_{\tau,r,i,j}(\pi)$, where $i = \ell(v)$ and $j = h(v)$. Let $B_{\tau,r}(v)$ be the set of permutations achieving this minimum.

For example, $a_{14532,3}(1,4,15,6) = 1$, because 15 must be colored white, but there exist extensions of $(1,4,15,6)$, such as that of Figure 4.2.1 , where all the other elements are colored black. Similarly, $b_{14532,3}(1,4,15,6) = 2$, because in any extension of $(1,4,15,6)$, 15 must be colored white, and at least one of the elements $\{1,2,3,4,5,6\}$ must also be colored white; but there exist extensions, such as that of Figure 4.2.1 , where the rest of $\{1,2,3,4,5,6\}$ is colored black.

Definition 4.12. *Let v be a prefix of a marked permutation π with $Kd + L$ black elements, where $L < d$. A number i is called stale if i is less than or equal to the Kdth black element in v.*

Definition 4.13. *Let $c_{\tau,r}(v)$ be the minimum, over all markings of permutations $\pi \in B_{\tau,r}(v)$, of the number of stale white elements in π.*

Definition 4.14. *If v and w are prefixes and w is an extension of v, we say that $v \prec w$ if one of the following is true:*

1. *1. $a(w) > a(v)$.*
2. *2. $a(w) = a(v)$ and $b(w) > b(v)$.*
3. *3. $a(w) = a(v)$ and $b(w) = b(v)$ and $c(w) > c(v)$.*

Definition 4.15. *Let τ be given. Let v be a prefix and let w be an extension of v by d elements. Let $x = w - v$; let x_j be the largest component of x and let x_j be the smallest. Any y between x_i and x_j such that y is not in $(x_1, \ldots, x_d)$ is called a gap. We call w a packed extension of v if:*

1. *1. $a(w) = a(v)$.*
2. *2. Every gap is either filled by an element of v, or is occupied by a black element in every extension x of w by d elements such that $a(x) = a(w)$.*

The key idea here is that a packed extension of v is a way to extend v by d elements that can all be black, in such a way that those elements are as close together as possible. In particular, if there are two packed extensions w and w' of v with $x = w - v$ and $x' = w' - v$, and $x'_j = x_j$, then $(x_1, \ldots, x_d)$ and $(x'_1, \ldots, x'_d)$ have the same relative order, and $x_i = x'_i$.

4.2.2 The key lemmata

Lemma 4.16. *There is at most one packed extension w of v with the property that $a(w) = a(v), b(w) = b(v)$, and $c(w) = c(v)$. (We all such an extension a perfect extension of v.)*

Proof. Suppose for purposes of contradiction that there are two such extensions, w and w'. Let $x = w - v$, and $x' = w' - v$. Because $a(w) = a(w') = a(v)$, all the elements $(x_1, \ldots, x_d)$ and $(x'_1, \ldots, x'_d)$ are black in every permutation $\pi \in B(w)$ or $\pi' \in B(w')$. Thus, $(x_1, \ldots, x_d)$ and $(x'_1, \ldots, x'_d)$ have the same relative order.

Because both w and w' are packed, either $x'_i > x_i$ for all i, or $x_i < x'_i$ for all i. Suppose, without loss of generality, that $x'_i > x_i$ for all i. Let x_j be the smallest of the x_i. Thus, x_j does not appear in w', because it is smaller than x'_j.

Since the last d elements of w are black, the stale white elements in π are those integers less than x_j that do not appear in w. Similarly, the stale white elements in π' are those integers less than x'_j that do not appear in w'. Because $x_j < x'_j$ and x_j does not appear in w', there are more stale white elements in π' than in π, contradicting $c(w) = c(w') = c(v)$. ☐

Lemma 4.17. *Let v be a prefix and let w be an extension of v by d elements. Then one of the following is true:*

1. 1. $v \prec w$

2. 2. w is a perfect extension of v

Furthermore, in the latter case, $q_{r,w}(k) = q_{r,v}(k-1)$.

Proof. Given v and w, suppose that $v \not\prec w$. Then we will show that 2 occurs and that $q_{r,w}(k) = q_{r,v}(k-1)$.

It can be easily seen that if $v \not\prec w$, then $a(w) = a(v)$, $b(w) = b(v)$, and $c(w) = c(v)$. So we only have to show that w is packed.

The idea here is relatively simple. If w is not packed, then it must contain a gap that is not filled by an element of v. This gap must eventually be filled by a white element. If the gap is in the upper portion of w, then that would result in $b(w) > b(v)$. If the gap is in the lower portion of w, that would result in $c(w) > c(v)$. These contradict $v \not\prec w$. Now we will look at the details.

Because $a(w) = a(v)$, all the added elements in w are colored black in all permutations $A(w)$. This proves that they must have the same relative order as $\tau^k(\ell(v)+1-a(v)), \ldots, \tau^k(\ell(v)+d-a(v))$, because that is the only way they can be black.

Now let $x = w - v$ and fix a marking of v. Under this marking, all of $x_1, \ldots, x_d$ are black, as previously noted. Also let $\ell(v) - a(v) = Kd + L$, where $L < d$. Let y be the smallest element of $(x_{L+1}, \ldots, x_d)$ and let z be the largest element of $(x_1, \ldots, x_L)$.

First, note that $(x_{L+1}, \ldots, x_{L+(d-L)})$ must be a translate of $(\tau^k(\ell(v) - a(v) + L + 1), \ldots, \tau^k(\ell(v) - a(v) + d))$. Otherwise, some element between y and x_i would be white in every marking of w. That element was not required to be white in every marking of v, though, because y is greater than all black elements in v, and no element greater than all black elements in v can be required to be white. Thus $b(w)$ would be greater than $b(v)$, and this is impossible.

Next, note that, because $c(w) = c(v)$, every element between x_1 and y that is required to be white in w must also have been required to be white in v. (Otherwise, that would be a new stale white element in w, making $c(w) > c(v)$.) Therefore, all the elements of $(x_1, \ldots, x_d)$ that are less than y must be as close together as possible, given x_j. Together with the previous paragraph, this establishes that w is packed.

Now consider a permutation $\pi \in Q_{r,w}(k)$ of the form $w|x$. Let $\pi' = v|x$. We claim $\pi' \in Q_{r,v}(k-1)$.

To do this, we will need some terminology. Fix a marking of π and let π' be marked by restricting the marking of w to v. Let $(v_1, \ldots, v_d)$ be the last d *black* elements of

v. Let $(x_1, x_2, \dots)$ be the elements of x. Let $(x'_1, x'_2, \dots)$ be the elements of x', where $x' = v|x - v$.

Note first that π' has the correct number of black elements. Also, the black elements in v are in the correct relative order, and the black elements in x are in the correct relative order. So it remains only to show that the black elements in v and x "mesh" correctly. That is, we will show that if x_i is black and is less than exactly s black elements of w, then x'_i is less than exactly s black elements of v.

Let $i \geq 1$ be given. Suppose x_i is less than exactly s elements from $w - v$. If $s \geq d$, then x_i cannot be black, because it is less than d black elements that appear prior to it. If $s = 0$, then x_i is greater than all elements of w, so x'_i is greater than all elements of $v_1, \dots, v_d$. Thus, x_i is greater than 0 black elements of w and x'_i is greater than 0 black elements of v. So the only remaining case is that $1 \leq s < d$. In this case, because w is packed, x_i is black. Suppose it is the mth smallest black element of x. It can easily be seen that x'_i is less than exactly s elements from $v_1, \dots, v_d$, because x'_i will occupy the slot of the mth smallest element of $w - v$. Thus $\pi' \in Q_{r,v}(k-1)$.

For the other direction, suppose $\pi' \in Q_{r,v}(k-1)$ is of the form $v|x$. Let $\pi = w|x$. We claim $\pi \in Q_{r,w}(k)$. Let $(v_1, \dots, v_d)$ be the last d *black* elements of v and let $(x_1, x_2, \dots)$ be the elements of x. Let $(x'_1, x'_2, \dots)$ be the elements of x', where $x' = w|x - w$.

Similarly to before, we need only establish that the black elements in w and x' "mesh" correctly; that is, we will show that if x_i is black and is less than exactly s elements of $v_1, \dots, v_d$, then x'_i is less than exactly s elements of $w - v$.

Let $i \geq 1$ be given. If x_i is stale, then x_i is white, so we may assume x_i is not stale.

Now, for each non-stale element y in the complement of v, $y \in w$. For otherwise, that element would be a stale element of the complement of w, which contradicts $c(w) = c(v)$.

Suppose x_i is the kth smallest non-stale element of the complement of v and is less than s elements from $v_1, \dots, v_d$. If $s > 0$, then x'_i fills the kth smallest gap of w; because w is packed, x'_i is less than s elements from $w - v$. This is what we claimed. On the other hand, if $s = 0$, then x'_i is greater than all elements of w, so x'_i is less than 0 black elements from $w - v$, which again is what we wanted. So $\pi' \in Q_{r,w}(k)$.

We have established a bijection between $Q_{r,v}(k-1)$ and $Q_{r,w}(k)$, so $q_{r,v}(k-1) = q_{r,w}(k)$. $\square$

Lemma 4.18. *Let v be a prefix with $a(v), b(v)$, and $c(v)$ all less than or equal to r. Then, for sufficiently large k,*

$$q_{\mathcal{T},r,v}(k) = \sum_{w \in S} q_{\mathcal{T},r,w}(k) + \alpha q_{\mathcal{T},r,v}(k-1),$$

where $\alpha \in \{0,1\}$ and where S is a set of extensions of v of length d such that for all $w \in S$, $v \prec w$. Furthermore, the number of values taken by $q_{r,w}(k)$ over S is bounded above by a constant that does not depend on k or v.

Proof. Let w range over the extensions of v with d additional elements. By Lemma 4.17 , each w either satisfies $v \prec w$ or is a perfect extension of v, and at most one prefix falls into the latter category. Grouping all the prefixes of the former category into S yields the formula.

For the second part, if w and w' differ only in $d+r$-superdiagonal elements, then $q_{r,w}(k) = q_{r,w'}(k)$ by Lemma 4.9. There are only $2(d+r)-1$ $(d+r)$-diagonal elements and, because $b(v) \le r$, at most r possibilities for subdiagonal elements in the additional elements of w and w'. So as w ranges over all the extensions of v by d elements, there are at most $(2d+3r)^d$ different values for $q_{r,w}(k)$. $\square$

4.2.3 The main result

Theorem 4.19. *The function $q_{\mathcal{T},r,v}(k)$ is eventually polynomial in k.*

Proof. The result is certainly true if $a(v)$, $b(v)$, or $c(v)$ is greater than r, for then $q_{\mathcal{T},r,v}(k) = 0$.

Now assume the result is true for all prefixes w such that $v \prec w$. By Lemma 4.18 , we have the following formula for sufficiently large k:

$$q_{\mathcal{T},r,v}(k) = \sum_{w \in S} q_{\mathcal{T},r,w}(k) + \alpha q_{\mathcal{T},r,v}(k-1).$$

Grouping together like terms in the sum, we have

$$q_{\mathcal{T},r,v}(k) = \alpha q_{\mathcal{T},r,v}(k-1) + \sum_{i=1}^{C} \beta_i(k) q_i(k),$$

where C is a constant, $q_i(k)$ is a polynomial by the assumption and β_i is a polynomial of degree at most 1 that counts the number of occurrences of $q_i(k)$ in the summation. Observe that $D(k) = \sum_{i=1}^{C} \beta_i(k)q_i(k)$ is eventually polynomial. Now, there are two cases:

1. 1. $\alpha = 0$. Then $q_{r,v}(k) = D(k)$, so $q_{r,v}(k)$ is eventually polynomial.
2. 2. $\alpha = 1$. Then $q_{r,v}(k) = q_{r,v}(k-1) + D(k)$. All solutions to this recurrence are eventually polynomial.

Either way, we have established the result for $q_{r,v}$.

If we have a chain of prefixes $v_0 \prec v_1 \prec \cdots \prec v_k$, where $a(v_k), b(v_k)$, and $c(v_k)$ are all less than or equal to $r+1$, then the chain has length at most $3r+3$ (because a, b, or c must be increased by at least 1 at each step). In particular, it is finite, so we have established the result by backward induction. □

Corollary 4.20. *The function $p_{\tau,r}(k)$ is eventually polynomial in k.*

Proof. $p_{\tau,r}(k) = q_{\tau,r,\varnothing}(k)$, where $\varnothing$ is the length 0 prefix. □

4.2.4 Independence of coefficients terms from choice of pattern

By convention, we will write $n = kd$ (recall that $\tau \in S_d$, and k is the number of repeats of τ). The functions $p_{\tau,r}(k)$ can be rewritten as polynomials in n by simply substituting $k = n/d$.

We will now show that the functions $p_{\tau,r}(n)$ and $p_{\tau',r}(n)$ only differ in the lower order terms for different permutations $\tau \in S_d$.

Theorem 4.21. *Two r-insertions of τ^k are equivalent if and only if the following are true:*

1. *1. The non-$(d+r)$-diagonal white elements are inserted at identical places*
2. *2. The permutations that result from inserting only the $(d+r)$-diagonal white elements elements are equivalent.*

Proof. Suppose the two insertions are equivalent. All the non-$(d+r)$-diagonal elements are white in both insertions, so they must be identical if the insertions are to be equivalent. Since they are identical, the permutations that result from deleting those elements are also equivalent.

Suppose 1 and 2 are both satisfied. Then insert the $(d+r)$-diagonal elements first. This results in equivalent permutations, by 2. Now insert the non-diagonal elements. Since they are identical, the permutations remain equivalent. □

Because of this result, we may count the extensions of τ^k by r elements as follows. as follows. First, insert s elements near the diagonal. Suppose this generates A different permutations. Then insert the remaining $r-s$ elements far from the diagonal. The number of ways to insert the elements far from the diagonal depends only on s, k, and d, and each way yields a different permutation, so there exists a number B such that each of the original A permutations results in B new permutations. Thus, the total number of permutations with s near-diagonal elements is AB.

As n becomes large, there are more ways to insert an element far from the diagonal than close to it. In particular, the number of ways to insert a new element near the diagonal is $\Theta(n)$, while the number of ways to insert a new element off the diagonal is $\Theta\left(n^2\right)$. So the number of extensions of τ^k by r elements, s or more of which are diagonal, is $O\left(n^{2r-s}\right)$.

The degree bound in the following theorem is not tight, but the proof is simpler than that for the tight bound.

Theorem 4.22. *If $\tau, \tau' \in S_d$, then $p_{\tau,r}(k) - p_{\tau',r}(k) = O\left(k^{2r-1}\right)$.*

Proof. Let us count $P_{\tau,r}(k)$. First, we count the permutations with no diagonal elements; as noted, this is a number that does not depend upon the permutation. Call it $C(n)$. Then we count the permutations with diagonal elements; this number is $O\left(n^{2r-1}\right)$. So

$$p_{\tau,r}(k) = C(n) + O\left(n^{2r-1}\right).$$

The same argument, of course, is true for $P_{\tau',r}(k)$. So

$$p_{\tau',r}(k) = C(n) + O\left(n^{2r-1}\right).$$

Adding gives

$$p_{\tau,r}(k) - p_{\tau',r}(k) = O\left(n^{2r-1}\right),$$

as desired. $\square$

To reduce $2r-1$ to $2r-2$, we repeat the same argument, except we also count those extensions with one diagonal white element separately. To do this, we need Lemma 1.2.

Theorem 4.23. *If $\tau, \tau' \in S_d$, then $p_{\tau,r}(k) - p_{\tau',r}(k) = O\left(k^{2r-2}\right)$.*

Proof. Let us count $P_{\tau,r}(k)$. First, we count the number of permutations with no diagonal elements; this is the $C(n)$ from the previous theorem. Then we count the permutations with one diagonal white element. The number of ways to insert the diagonal white element does not depend on τ. Some of these might be equivalent; but, by Lemma 1.2, the number of equivalences is $2n$, regardless of τ. So the number of permutations with one diagonal white element is some function $D(n)$ that does not depend on τ. Finally, we count the permutations with two or more diagonal white elements, which is $O\left(n^{2r-2}\right)$. So

$$p_{\tau,r}(k) = C(n) + D(n) + O\left(n^{2r-2}\right).$$

Of course, the same is true for $P_{\tau',r}(k)$, so

$$p_{\tau,r}(k) - p_{\tau',r}(k) = O\left(n^{2r-2}\right).$$

$\square$

We can reduce $2r-2$ to $2r-3$ with one final trick and the theorem of Ray and West.

Theorem 4.24. *If $\tau, \tau' \in S_d$, then $p_{\tau,r}(k) - p_{\tau',r}(k) = O\left(k^{2r-3}\right)$.*

Proof. This time we will count the permutations with exactly 2 diagonal elements separately. The number of ways to insert the diagonal white elements does not depend on τ. Some of these might be equivalent; but, by Theorem 1.3, the number of such equivalences is $2n^3 + 6n^2 + 4n + j(\tau)$, where $0 \leq j(\tau) \leq n-1$. Thus, the number of such permutations is $(E(n) + j(\tau))K(n)$, where $j(\tau) = O(n)$ and $K(n) = \Theta\left(n^{2r-4}\right)$. So

$$\begin{aligned} p_{\tau,r}(k) &= C(n) + D(n) + (E(n) + j(\tau))K(n) + O\left(k^{2r-3}\right) \\ &= C(n) + D(n) + (E(n) + O\left(n\right))K(n) + O\left(k^{2r-3}\right) \\ &= C(n) + D(n) + E(n)K(n) + O\left(n^{2r-3}\right). \end{aligned}$$

As before, the same is true of τ', so

$$p_{\tau,r}(k) - p_{\tau',r}(k) = O\left(n^{2r-3}\right).$$

□

Since we know that $p_{\tau,r}(k)$ is eventually polynomial, and have bounded the degree, a finite amount of empirical data provides a proof of a formula for $p_{\tau,r}(k)$. In fact, it appears that $p_{\tau,r}(k)$ becomes polynomial when $k = d-1$. Our proof is not quite sharp enough to prove this fact, so the following formulas are currently only semi-rigorous, because they are based on that assumption. However, with enough data, they could be rigorously proved. We also expect that the proof can be sharpened to establish once and for all that $p_{\tau,r}(k)$ is polynomial for $k \geq d-1$.

$$\begin{aligned} p_{21,2}(n) &= \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + n + 3 \\ p_{21,3}(n) &= \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + \frac{2}{3}n^3 + \frac{19}{6}n^2 + \frac{59}{6}n + 13 \\ p_{21,4}(n) &= \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{19}{6}n^5 + \frac{29}{24}n^4 + \frac{17}{2}n^3 + \frac{241}{6}n^2 + \frac{241}{3}n + 38 \end{aligned}$$

$$p_{132,2}(n) = \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + n + 3$$
$$p_{132,3}(n) = \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + \frac{2}{3}n^3 + \frac{19}{6}n^2 + 10n + 12$$
$$p_{132,4}(n) = \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{19}{6}n^5 + \frac{29}{24}n^4 + \frac{26}{3}n^3 + \frac{241}{6}n^2 + \frac{443}{6}n + 45$$

$$p_{231,2}(n) = \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + \frac{4}{3}n + 2$$
$$p_{231,3}(n) = \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + n^3 + \frac{7}{2}n^2 + 8n + 6$$
$$p_{231,4}(n) = \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{10}{3}n^5 + \frac{19}{8}n^4 + \frac{61}{6}n^3 + \frac{595}{18}n^2 - \frac{50}{3}n + 201$$

$$p_{321,2}(n) = \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + n + 3$$
$$p_{321,3}(n) = \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + \frac{2}{3}n^3 + \frac{19}{6}n^2 + 10n + 13$$
$$p_{321,4}(n) = \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{19}{6}n^5 + \frac{29}{24}n^4 + \frac{26}{3}n^3 + \frac{247}{6}n^2 + \frac{449}{6}n + 66$$

$$p_{1243,2}(n) = \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + n + 3$$
$$p_{1243,3}(n) = \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + \frac{2}{3}n^3 + \frac{19}{6}n^2 + \frac{121}{12}n + 12$$
$$p_{1243,4}(n) = \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{19}{6}n^5 + \frac{29}{24}n^4 + \frac{35}{4}n^3 + \frac{122}{3}n^2 + \frac{220}{3}n + 41$$

$$p_{1324,2}(n,2) = \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + n + 3)$$
$$p_{1324,3}(n,3) = \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + \frac{2}{3}n^3 + \frac{19}{6}n^2 + \frac{121}{12}n + 12$$
$$p_{1324,4}(n,4) = \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{19}{6}n^5 + \frac{29}{24}n^4 + \frac{35}{4}n^3 + \frac{119}{3}n^2 + \frac{202}{3}n + 55$$

$$p_{1342,2}(n) = \frac{1}{2}n^4 + n^3 + \frac{1}{2}n^2 + \frac{3}{2}n + 2$$
$$p_{1342,3}(n) = \frac{1}{6}n^6 + n^5 + \frac{5}{3}n^4 + \frac{7}{6}n^3 + \frac{25}{6}n^2 + \frac{11}{6}n + 18$$
$$p_{1342,4}(n) = \frac{1}{24}n^8 + \frac{1}{2}n^7 + \frac{25}{12}n^6 + \frac{41}{12}n^5 + \frac{77}{24}n^4 + \frac{25}{4}n^3 + \frac{199}{24}n^2 + \frac{400}{3}n - 7$$

Note that $p_{1342,4}(n) - p_{1243,4}(n) = \Omega(n^5)$. This demonstrates by example that the $O\left(n^{2r-3}\right)$ bound cannot be improved to $O\left(n^{2r-4}\right)$.

Appendices

Appendix A

Exact formula for $C_{r,d}$

The formula for $C_{r,d}$ (see 3.7), as proved by Chapuy, is

$$C_{r,d} = \frac{\sqrt{d-1}\prod_{i=1}^{d-2} i!}{(2\pi)^{\frac{d}{2}-1}} \left(\frac{d(d-1)}{r(2d+r-1)}\right)^{d(d-2)/2}. \tag{A.1}$$

Appendix B

Excerpts of code for Chapter 2

```
################################################################################
##                                 SECTION 2                                  ##
################################################################################

# If A(n) = n^2, this bijectifies the identity A(n) = A(n-1) + (2n-1)
# using the proof in the paper. The final bijection is tb17.

n := 6:

tb0  := bxIdentity(n):
tb1  := bijMuln([tb0, tb0]):
tb2  := bijInvert(bxSum([n-1, 1])):
tb3  := bijPreSubs(tb1, tb2, [1]):
tb4  := bijPreSubs(tb3, tb2, [2]):
tb5  := bijPreApplyFamily(tb4, ‘bxLeftDistribute‘, [], []):
tb6  := bijPreMulPermute(bijPreMulPermute(tb5, [2, 1], [1]), [2, 1], [2]):
tb7  := bijPreApplyFamily(tb6, ‘bxLeftDistribute‘, [], [1]):
tb8  := bijPreApplyFamily(tb7, ‘bxLeftDistribute‘, [], [2]):
tb9  := bijPreApplyFamily(tb8, ‘bxFlattenSum‘, [1, 2], []):
tb10 := bijPreApplyFamily(tb9, ‘bxFlattenSum‘, [3, 2], []):
tb11 := bijPreMulPermute(tb10, [2, 1], [2]):
tb12 := bijPreApplyFamily(tb11, ‘bxOneEliminate‘, [], [2]):
tb13 := bijPreApplyFamily(tb12, ‘bxOneEliminate‘, [], [3]):
```

```
tb14 := bijPreApplyFamily(tb13, ‘bxOneEliminate‘, [], [4]):

tb15 := bijPreApplyFamily(tb14, ‘bxPullOutSum‘, [2, 3], []):

tb16 := bxSum([n-1, n-1, 1]):

tb17 := bijPreSubs(tb15, tb16, [2]):


################################################################################

##                                  SECTION 3                                 ##

################################################################################


# The following implements the Franel identity. Bijections are stored

# frequently so that you can look at various stages in the process, if

# you are interested.


bxBCCubedSystems := proc(n)

    global Lc0, Lc1, tc2, tc3a, tc3, tc4a, tc4, tc5a, tc5, tc6a, tc6, tc7a,

    tc7, tc8a, tc8, tc9a, tc9, tc10a, tc10, Lc11, tc12a, tc12, tc13a, tc13,

    tc14a, tc14, tc15a, tc15, tc16a, tc16, tc17a, tc17, tc18a, tc18, tc19a, tc19,

    tc20a, tc20, tc21a, tc21, tc22a, tc22, tc23a, tc23:

    local g, j, k, tc5perm, tc17perm, tc20perm, tc22perm, tct:


    Lc0 := [seq(bxBCIdentity(n,k), k=0..n)]:

    Lc1 := map(x->bijMuln([x, x, x]), Lc0):

    tc2 := bijAddn(Lc1):


# Expand by pascal

    tc3a := tc2:

    for k from 0 to n do:

        tc3a := bijPostSubs(tc3a, bxPascal(n, k), [k+1, 1]):

        tc3a := bijPostSubs(tc3a, bxPascal(n, k), [k+1, 2]):

        tc3a := bijPostSubs(tc3a, bxPascal(n, k), [k+1, 3]):
```

```
    end:
    tc3 := tc3a:

# (n-1 choose 0)^3 + sum_{i=1}^{n-1}[((n-1 choose i) + (n-1 choose
# i-1))^3] + (n-1 choose n-1)^3



# Distribute
    tc4a := tc3:
    tct := [0,0,0,0,0,0,0,0,0,0]:
    for k from 1 to n-1 do:
        tc4a := bijPostApplyFamily(tc4a, `bxPullOutProd`, [2, 2], [k+1]):
        # term is (\binom{n-1}{k} + \binom{n-1}{k-1}) * ((\binom{n-1}{k} +
        # \binom{n-1}{k-1}) * (\binom{n-1}{k} + \binom{n-1}{k-1}))
        tct[k] := tc4a:
        tc4a := bijPostDistribute(tc4a, [k+1, 2]):
        # term is (\binom{n-1}{k} + \binom{n-1}{k-1}) * ((\binom{n-1}{k} +
        # \binom{n-1}{k-1}) * \binom{n-1}{k} + (\binom{n-1}{k} +
        # \binom{n-1}{k-1}) * \binom{n-1}{k-1})

        tc4a := bijPostApplyFamily(tc4a, `bxRightDistribute`, [], [k+1, 2, 1]):
        tc4a := bijPostApplyFamily(tc4a, `bxRightDistribute`, [], [k+1, 2, 2]):
        # term is (\binom{n-1}{k} + \binom{n-1}{k-1}) *
        # ((\binom{n-1}{k}*\binom{n-1}{k} +
        # \binom{n-1}{k-1}*\binom{n-1}{k}) +
        # (\binom{n-1}{k-1}*\binom{n-1}{k} +
        # \binom{n-1}{k-1}*\binom{n-1}{k-1}))

        tc4a := bijPostApplyFamily(tc4a, `bxFlattenSum`, [2, 2], [k+1, 2]):
        tc4a := bijPostApplyFamily(tc4a, `bxFlattenSum`, [1, 2], [k+1, 2]):
```

```
        # term is (\binom{n-1}{k} + \binom{n-1}{k-1})*
        # (\binom{n-1}{k}*\binom{n-1}{k} +
        # \binom{n-1}{k-1}*\binom{n-1}{k} +
        # \binom{n-1}{k-1}*\binom{n-1}{k} +
        # \binom{n-1}{k-1}*\binom{n-1}{k-1})

        tc4a := bijPostApplyFamily(tc4a, `bxLeftDistribute`, [], [k+1]):
        for j from 1 to 4 do:
            tc4a := bijPostApplyFamily(tc4a, `bxRightDistribute`, [], [k+1, j]):
        od:
        for j from 4 to 1 by -1 do:
            tc4a := bijPostApplyFamily(tc4a, `bxFlattenSum`, [j, 2], [k+1]):
        od:
        for j from 1 to 8 do:
            tc4a := bijPostApplyFamily(tc4a, `bxFlattenProd`, [2, 2], [k+1, j]):
        od:
    od:
    tc4 := tc4a:

# Flatten entire thing and collect like terms

    tc5a := tc4:
    for k from n-1 to 1 by -1 do:
        tc5a := bijPostApplyFamily(tc5a, `bxFlattenSum`, [k+1, 8], []):
    od:

# 8n-6 terms
# first and last are special; other than that every 8 terms must be brought
# together
```

```
    tc5perm := [1, seq(floor((i-1)/(n-1))+2+((i-1) mod (n-1))*8, i=1..8*n-8),
                8*n-6]:
    tc5a := bijPostAddPermute(tc5a, tc5perm, []):

# Pull out groups of n-1 or n

    for j from 8 to 1 by -1 do:
        if j = 8 then:
            tc5a := bijPostApplyFamily(tc5a, `bxPullOutSum`, [8*n-6 - n + 1, n],
                                        []):
        elif j = 1 then:
            tc5a := bijPostApplyFamily(tc5a, `bxPullOutSum`, [1, n], []):
        else:
            tc5a := bijPostApplyFamily(tc5a, `bxPullOutSum`, [(j-1)*(n-1)+2,
                                                               n-1], []):
        fi:
    od:

    tc5 := tc5a:

    tc6a := tc5:

# Rearrange groups 4, 6, and 7 (which are of the form (n choose k)(n
# choose k-1)^2) to look like groups 2, 3, and 5 (n choose k)^2(n
# choose k-1).

    for k from 1 to n-1 do:
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k-1), [4, k, 1]):
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k-1), [4, k, 2]):
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k), [4, k, 3]):
```

```
    od:
    tc6a := bijPostAddPermute(tc6a, Reverse([seq(i, i=1..n-1)]), [4]):


    for k from 1 to n-1 do:
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k-1), [6, k, 1]):
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k), [6, k, 2]):
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k-1), [6, k, 3]):
    od:
    tc6a := bijPostAddPermute(tc6a, Reverse([seq(i, i=1..n-1)]), [6]):


    for k from 1 to n-1 do:
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k), [7, k, 1]):
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k-1), [7, k, 2]):
        tc6a := bijPostSubs(tc6a, bxBCSymm(n-1,k-1), [7, k, 3]):
    od:
    tc6a := bijPostAddPermute(tc6a, Reverse([seq(i, i=1..n-1)]), [7]):

# Rearrange groups 2 and 7 (n choose k-1)(n choose k)^2 and groups 3
# and 6 (n choose k)(n choose k-1)(n choose k) to look like
# groups 4 and 5 (n choose k)^2(n choose k-1)

    for g in [2, 7] do:
        for k from 1 to n-1 do:
            tc6a := bijPostMulPermute(tc6a, [2, 3, 1], [g, k]):
        od:
    od:

    for g in [3, 6] do:
        for k from 1 to n-1 do:
            tc6a := bijPostMulPermute(tc6a, [1, 3, 2], [g, k]):
        od:
    od:

# Bring group 8 next to group 1

    tc6a := bijPostAddPermute(tc6a, [1, 8, 2, 3, 4, 5, 6, 7], []):

# Introduce factors of 1

    for g from 1 to 8 do:
        tc6a := bijPostApplyFamily(tc6a, `bxOneIntroduce`, [], [g]):
    od:

# Pull out like terms

    tc6a := bijPostApplyFamily(tc6a, `bxPullOutSum`, [3, 6], []):
    tc6a := bijPostApplyFamily(tc6a, `bxPullOutSum`, [1, 2], []):

# Factor out common factor

    tc6a := bijPostApplyFamily(tc6a, `bxRightFactor`, [], [1]):
    tc6a := bijPostApplyFamily(tc6a, `bxRightFactor`, [], [2]):


    tc6 := tc6a:

# Add up all the 1s
```

```
    tc7 := bxSum([1, 1]):
    tc8 := bxSum([1, 1, 1, 1, 1, 1]):


    tc9 := bijPostSubs(tc6, tc7, [1, 1]):
    tc10 := bijPostSubs(tc9, tc8, [2, 1]):

# That's item (2) in the paper.









# Now, on to item (3)

    Lc11 := map(i->bijMuln([Lc0[i+1], Lc0[i+1], Lc0[i]]), [seq(i, i=1..n)]):
    tc12 := bijMuln([bxIdentity(n+1), bxIdentity(n+1)]):
    tc13 := bijAddn(Lc11):
    tc14 := bijMuln([tc12, tc13]):

    tc14a := tc14:
    tc14a := bijPostApplyFamily(tc14a, `bxLeftDistribute`, [], []):
    for k from 1 to n do:
        # Flatten
        tc14a := bijPostApplyFamily(tc14a, `bxFlattenProd`, [2, 3], [k]):
        tc14a := bijPostApplyFamily(tc14a, `bxFlattenProd`, [1, 2], [k]):
    od:

    tc15 := tc14a:
```

```
tc15a := tc15:

for k from 1 to n do:
    # Rearrange and pull out to prepare for special identity
    tc15a := bijPostMulPermute(tc15a, [1, 3, 2, 4, 5], [k]):
    tc15a := bijPostApplyFamily(tc15a, ‘bxPullOutProd‘, [3, 3], [k]):

    # Apply special identity
    tc15a := bijPostSubs(tc15a, bxBCSpecialIdentity(n, k), [k, 3]):

    # Re-flatten
    tc15a := bijPostApplyFamily(tc15a, ‘bxFlattenProd‘, [3, 3], [k]):

    # Bring third term (n) to front as done in algebra in paper
    tc15a := bijPostMulPermute(tc15a, [3, 1, 2, 4, 5], [k]):
od:

tc16 := tc15a:
tc16a := tc16:

for k from 1 to n do:
    # Pascal’s identity on last factor (n+1 choose k)
    tc16a := bijPostSubs(tc16a, bxPascal(n+1, k), [k, 5]):
od:

tc17 := tc16a:
tc17a := tc17:

for k from 1 to n do:
    # Group terms for distribution:
        tc17a := bijPostApplyFamily(tc17a, ‘bxPullOutProd‘, [1, 4], [k]):
    od:

# Distribute inside the summation sign
    for k from 1 to n do:
        tc17a := bijPostApplyFamily(tc17a, ‘bxLeftDistribute‘, [], [k]):
    od:

# Flatten the summation
    for k from n to 1 by -1 do:
        tc17a := bijPostApplyFamily(tc17a, ‘bxFlattenSum‘, [k, 2], []):
    od:

# Bring together alternating terms
    tc17perm := [seq(floor((i-1)/n)+((i-1) mod n)*2+1, i=1..2*n)]:
    tc17a := bijPostAddPermute(tc17a, tc17perm, []):

# Break up sum into two parts
    tc17a := bijPostApplyFamily(tc17a, ‘bxPullOutSum‘, [n+1, n], []):
    tc17a := bijPostApplyFamily(tc17a, ‘bxPullOutSum‘, [1, n], []):

# Flatten products and put last factor in 4th position (in both sums) as per
# algebra
    for k from 1 to n do:
        tc17a := bijPostApplyFamily(tc17a, ‘bxFlattenProd‘, [1, 4], [1, k]):
        tc17a := bijPostMulPermute(tc17a, [1, 2, 3, 5, 4], [1, k]):
        tc17a := bijPostApplyFamily(tc17a, ‘bxFlattenProd‘, [1, 4], [2, k]):
        tc17a := bijPostMulPermute(tc17a, [1, 2, 3, 5, 4], [2, k]):
    od:
```

```
    tc18 := tc17a:

    tc18a := tc18:


#Working on the first sum, expand both (n choose k) with pascal and

# distribute. CAUTION: Attention requried if k = n.


    for k from 1 to n do:

        tc18a := bijPostApplyFamily(tc18a, ‘bxPullOutProd‘, [3, 2], [1, k]):

        tc18a := bijPostSubs(tc18a, bxPascal(n,k), [1, k, 3, 1]):

        tc18a := bijPostSubs(tc18a, bxPascal(n,k), [1, k, 3, 2]):

        if k < n then:

            tc18a := bijPostApplyFamily(tc18a, ‘bxLeftDistribute‘, [], [1, k, 3]):

            tc18a := bijPostApplyFamily(tc18a, ‘bxRightDistribute‘, [],

                                            [1, k, 3, 1]):

            tc18a := bijPostApplyFamily(tc18a, ‘bxRightDistribute‘, [],

                                            [1, k, 3, 2]):

            tc18a := bijPostApplyFamily(tc18a, ‘bxFlattenSum‘, [2, 2], [1, k, 3]):

            tc18a := bijPostApplyFamily(tc18a, ‘bxFlattenSum‘, [1, 2], [1, k, 3]):

        fi:

    od:


# Commute multiplication and combine like terms (the cross-terms)


    for k from 1 to n-1 do:

        tc18a := bijPostMulPermute(tc18a, [2, 1], [1, k, 3, 2]):

        tc18a := bijPostApplyFamily(tc18a, ‘bxPullOutSum‘, [2, 2], [1, k, 3]):


        # Introduce factors of 1

        tc18a := bijPostApplyFamily(tc18a, ‘bxOneIntroduce‘, [],

                                        [1, k, 3, 2, 1]):
```

```
        tc18a := bijPostApplyFamily(tc18a, ‘bxOneIntroduce‘, [],
                                    [1, k, 3, 2, 2]):

        # Factor
        tc18a := bijPostApplyFamily(tc18a, ‘bxRightFactor‘, [], [1, k, 3, 2]):

        # 1 + 1 = 2
        tc18a := bijPostSubs(tc18a, tc7, [1, k, 3, 2, 1]):

        # Flatten (not absolutely sure if this is desirable, but
        # doing it anyway)
        tc18a := bijPostApplyFamily(tc18a, ‘bxFlattenProd‘, [2, 2],
                                    [1, k, 3, 2]):
    od:

# Working on the second sum, use special identity, then rearrange factors.

    for k from 1 to n do:
        tc18a := bijPostApplyFamily(tc18a, ‘bxPullOutProd‘, [2, 3], [2, k]):
        tc18a := bijPostSubs(tc18a, bxBCSpecialIdentity(n, k), [2, k, 2]):
        tc18a := bijPostApplyFamily(tc18a, ‘bxFlattenProd‘, [2, 3], [2, k]):
        tc18a := bijPostMulPermute(tc18a, [1, 2, 3, 5, 4], [2, k]):
        # Opting not to combine n*n, at least for now.
    od:

    tc19 := tc18a:
    tc19a := tc19:

# In first sum, distribute fourth factor across third, then reorder
# factors in 2nd term of result.
```

```
    for k from 1 to n-1 do:
        tc19a := bijPostApplyFamily(tc19a, `bxPullOutProd`, [3, 2], [1, k]):
        tc19a := bijPostApplyFamily(tc19a, `bxRightDistribute`, [], [1, k, 3]):
        tc19a := bijPostApplyFamily(tc19a, `bxFlattenProd`, [1, 2],
                                    [1, k, 3, 1]):
        tc19a := bijPostApplyFamily(tc19a, `bxFlattenProd`, [1, 3],
                                    [1, k, 3, 2]):
        tc19a := bijPostApplyFamily(tc19a, `bxFlattenProd`, [1, 2],
                                    [1, k, 3, 3]):
        tc19a := bijPostMulPermute(tc19a, [1, 3, 4, 2], [1, k, 3, 2]):
    od:

# Handling last term specially
    tc19a := bijPostApplyFamily(tc19a, `bxFlattenProd`, [3, 2], [1, k]):
    tc19a := bijPostApplyFamily(tc19a, `bxPullOutProd`, [3, 3], [1, k]):

# In second sum, expand (n+1 choose k) with pascal twice (CAUTION:
# attention required if k = 1 or n.)

    for k from 1 to n do:
        tc19a := bijPostSubs(tc19a, bxPascal(n+1,k), [2, k, 5]):
        tc19a := bijPostSubs(tc19a, bxPascal(n, k), [2, k, 5, 1]):
        tc19a := bijPostSubs(tc19a, bxPascal(n, k-1), [2, k, 5, 2]):
        if k = 1 then:
            tc19a := bijPostApplyFamily(tc19a, `bxFlattenSum`, [1, 2], [2, k, 5]):
        elif k = n then:
            tc19a := bijPostApplyFamily(tc19a, `bxFlattenSum`, [2, 2], [2, k, 5]):
        else:
            tc19a := bijPostApplyFamily(tc19a, `bxFlattenSum`, [2, 2], [2, k, 5]):
```

```
            tc19a := bijPostApplyFamily(tc19a, ‘bxFlattenSum‘, [1, 2], [2, k, 5]):
        fi:
        if k < n then:
            tc19a := bijPostApplyFamily(tc19a, ‘bxPullOutSum‘, [2, 2], [2, k, 5]):
            tc19a := bijPostApplyFamily(tc19a, ‘bxOneIntroduce‘, [],
                                            [2, k, 5, 2, 1]):
            tc19a := bijPostApplyFamily(tc19a, ‘bxOneIntroduce‘, [],
                                            [2, k, 5, 2, 2]):
            tc19a := bijPostApplyFamily(tc19a, ‘bxRightFactor‘, [], [2, k, 5, 2]):
            tc19a := bijPostSubs(tc19a, tc7, [2, k, 5, 2, 1]):
        else:
            tc19a := bijPostApplyFamily(tc19a, ‘bxPullOutSum‘, [1, 2], [2, k, 5]):
            tc19a := bijPostApplyFamily(tc19a, ‘bxOneIntroduce‘, [],
                                            [2, k, 5, 1, 1]):
            tc19a := bijPostApplyFamily(tc19a, ‘bxOneIntroduce‘, [],
                                            [2, k, 5, 1, 2]):
            tc19a := bijPostApplyFamily(tc19a, ‘bxRightFactor‘, [], [2, k, 5, 1]):
            tc19a := bijPostSubs(tc19a, tc7, [2, k, 5, 1, 1]):
        fi:
    od:


    tc20 := tc19a:
    tc20a := tc20:


# Break up first sum into 3 pieces
    for k from 1 to n-1 do:
        tc20a := bijPostApplyFamily(tc20a, ‘bxPullOutProd‘, [1, 2], [1, k]):
        tc20a := bijPostApplyFamily(tc20a, ‘bxLeftDistribute‘, [], [1, k]):
    od:
```

```
# handle last term separately
    tc20a := bijPostApplyFamily(tc20a, `bxPullOutProd`, [1, 2], [1, n]):

    for k from n-1 to 1 by -1 do:
        tc20a := bijPostApplyFamily(tc20a, `bxFlattenSum`, [k, 3], [1]):
    od:

    tc20perm := [seq(floor((i-1)/(n-1))+((i-1) mod (n-1))*3+1, i=1..3*n-3), 3*n-2]:
    tc20a := bijPostAddPermute(tc20a, tc20perm, [1]):
    tc20a := bijPostApplyFamily(tc20a, `bxPullOutSum`, [2*n-1, n], [1]):
    tc20a := bijPostApplyFamily(tc20a, `bxPullOutSum`, [n, n-1], [1]):
    tc20a := bijPostApplyFamily(tc20a, `bxPullOutSum`, [1, n-1], [1]):
    tc20a := bijPostApplyFamily(tc20a, `bxLeftFactor`, [], [1, 1]):
    tc20a := bijPostApplyFamily(tc20a, `bxLeftFactor`, [], [1, 2]):
    tc20a := bijPostApplyFamily(tc20a, `bxLeftFactor`, [], [1, 3]):

# Use symmetry on second piece to make it look like first, but backwards
    for k from 1 to n-1 do:
        tc20a := bijPostSubs(tc20a, bxBCSymm(n-1,k-1), [1, 2, 2, k, 2]):
        tc20a := bijPostSubs(tc20a, bxBCSymm(n-1,k-1), [1, 2, 2, k, 3]):
        tc20a := bijPostSubs(tc20a, bxBCSymm(n-1,k), [1, 2, 2, k, 4]):
    od:

# Reverse second piece
    tc20a := bijPostAddPermute(tc20a, Reverse([seq(i, i=1..n-1)]), [1, 2, 2]):

# Factor out 2 from second piece and bring this factor to the front
    for k from 1 to n-1 do:
        tc20a := bijPostApplyFamily(tc20a, `bxPullOutProd`, [2, 3], [1, 2, 2, k]):
    od:
```

```
    tc20a := bijPostApplyFamily(tc20a, `bxLeftFactor`, [], [1, 2, 2]):
    tc20a := bijPostApplyFamily(tc20a, `bxFlattenProd`, [2, 2], [1, 2]):
    tc20a := bijPostMulPermute(tc20a, [2, 1, 3], [1, 2]):
    tc20a := bijPostApplyFamily(tc20a, `bxPullOutProd`, [2, 2], [1, 2]):

# Introduce a factor of 1 to the first piece

    tc20a := bijPostApplyFamily(tc20a, `bxOneIntroduce`, [], [1, 1]):

# Combine first two pieces and identify common factor

    tc20a := bijPostApplyFamily(tc20a, `bxPullOutSum`, [1, 2], [1]):
    tc20a := bijPostApplyFamily(tc20a, `bxRightFactor`, [], [1, 1]):
    tc21 := bxSum([1, 2]):
    tc20a := bijPostSubs(tc20a, tc21, [1, 1, 1]):

# Flatten
    tc20a := bijPostApplyFamily(tc20a, `bxFlattenProd`, [2, 2], [1, 1]):
    tc20a := bijPostApplyFamily(tc20a, `bxFlattenProd`, [2, 2], [1, 1]):
    tc20a := bijPostApplyFamily(tc20a, `bxPullOutProd`, [1, 3], [1, 1]):

# In second sum, distribute and rearrange

    for k from 1 to n do:
        tc20a := bijPostApplyFamily(tc20a, `bxPullOutProd`, [3, 3], [2, k]):
        tc20a := bijPostApplyFamily(tc20a, `bxPullOutProd`, [1, 2], [2, k, 3]):
        tc20a := bijPostApplyFamily(tc20a, `bxLeftDistribute`, [], [2, k, 3]):
        if k > 1 and k < n then:
            tc20a := bijPostApplyFamily(tc20a, `bxFlattenProd`, [1, 2],
                                        [2, k, 3, 3]):
```

```
        fi:
        tc20a := bijPostApplyFamily(tc20a, ‘bxFlattenProd‘, [1, 2], [2, k, 3, 2]):
        tc20a := bijPostApplyFamily(tc20a, ‘bxFlattenProd‘, [1, 2], [2, k, 3, 1]):
        if k = n then:
            tc20a := bijPostApplyFamily(tc20a, ‘bxFlattenProd‘, [3, 2],
                                            [2, k, 3, 1]):
            tc20a := bijPostMulPermute(tc20a, [3, 1, 2, 4], [2, k, 3, 1]):
        else:
            tc20a := bijPostApplyFamily(tc20a, ‘bxFlattenProd‘, [3, 2],
                                            [2, k, 3, 2]):
            tc20a := bijPostMulPermute(tc20a, [3, 1, 2, 4], [2, k, 3, 2]):
        fi:
    od:

    tc22 := tc20a:
    tc22a := tc22:

# Break up second sum into 3 pieces
    for k from 1 to n do:
        tc22a := bijPostApplyFamily(tc22a, ‘bxPullOutProd‘, [1, 2], [2, k]):
        tc22a := bijPostApplyFamily(tc22a, ‘bxLeftDistribute‘, [], [2, k]):
    od:

    tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenSum‘, [n, 2], [2]):
    for k from n-1 to 2 by -1 do:
        tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenSum‘, [k, 3], [2]):
    od:
    tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenSum‘, [1, 2], [2]):
```

```
    tc22perm := [1, seq(3*i, i=1..n-2), 2, seq(3*i+1, i=1..n-2), 3*n-3,
                 seq(3*i+2, i=1..n-2), 3*n-2]:
    tc22a := bijPostAddPermute(tc22a, tc22perm, [2]):
    tc22a := bijPostApplyFamily(tc22a, `bxPullOutSum`, [2*n, n-1], [2]):
    tc22a := bijPostApplyFamily(tc22a, `bxPullOutSum`, [n, n], [2]):
    tc22a := bijPostApplyFamily(tc22a, `bxPullOutSum`, [1, n-1], [2]):
    tc22a := bijPostApplyFamily(tc22a, `bxLeftFactor`, [], [2, 1]):
    tc22a := bijPostApplyFamily(tc22a, `bxLeftFactor`, [], [2, 2]):
    tc22a := bijPostApplyFamily(tc22a, `bxLeftFactor`, [], [2, 3]):

# Use symmetry on first piece to make it look like third, but backwards
    for k from 1 to n-1 do:
        tc22a := bijPostSubs(tc22a, bxBCSymm(n-1,k-1), [2, 1, 2, k, 1]):
        tc22a := bijPostSubs(tc22a, bxBCSymm(n-1,k-1), [2, 1, 2, k, 2]):
        tc22a := bijPostSubs(tc22a, bxBCSymm(n-1,k), [2, 1, 2, k, 3]):
    od:

# Reverse first piece
    tc22a := bijPostAddPermute(tc22a, Reverse([seq(i, i=1..n-1)]), [2, 1, 2]):

# Exchange second and third pieces
    tc22a := bijPostAddPermute(tc22a, [1, 3, 2], [2]):

# Introduce a factor of 1 to the first and second pieces

    tc22a := bijPostApplyFamily(tc22a, `bxOneIntroduce`, [], [2, 1]):
    tc22a := bijPostApplyFamily(tc22a, `bxOneIntroduce`, [], [2, 2]):

# Combine first two pieces and identify common factor
```

```
    tc22a := bijPostApplyFamily(tc22a, ‘bxPullOutSum‘, [1, 2], [2]):
    tc22a := bijPostApplyFamily(tc22a, ‘bxRightFactor‘, [], [2, 1]):
    tc22a := bijPostSubs(tc22a, tc7, [2, 1, 1]):

# Flatten

    tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenProd‘, [2, 2], [2, 1]):
    tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenProd‘, [2, 2], [2, 1]):
    tc22a := bijPostApplyFamily(tc22a, ‘bxPullOutProd‘, [1, 3], [2, 1]):

# Factor out 2 from third (now second) piece and bring this factor to the front
    for k from 1 to n do:
        tc22a := bijPostApplyFamily(tc22a, ‘bxPullOutProd‘, [2, 3], [2, 2, 2, k]):
    od:
    tc22a := bijPostApplyFamily(tc22a, ‘bxLeftFactor‘, [], [2, 2, 2]):
    tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenProd‘, [2, 2], [2, 2]):
    tc22a := bijPostMulPermute(tc22a, [2, 1, 3], [2, 2]):

# Combine 2 and n^2
    tc22a := bijPostApplyFamily(tc22a, ‘bxFlattenProd‘, [2, 2], [2, 2]):
    tc22a := bijPostApplyFamily(tc22a, ‘bxPullOutProd‘, [1, 3], [2, 2]):

    tc22 := tc22a:
    tc23a := tc22:

# Flatten the entire sum
    tc23a := bijPostApplyFamily(tc23a, ‘bxFlattenSum‘, [2, 2], []):
    tc23a := bijPostApplyFamily(tc23a, ‘bxFlattenSum‘, [1, 2], []):

# Bring like terms adjacent
```

```
    tc23a := bijPostAddPermute(tc23a, [1, 3, 2, 4], []):

# Extract common factor from second two
    tc23a := bijPostApplyFamily(tc23a, `bxPullOutSum`, [3, 2], []):
    tc23a := bijPostApplyFamily(tc23a, `bxRightFactor`, [], [3]):

# Extract common factor from first two
    tc23a := bijPostApplyFamily(tc23a, `bxPullOutSum`, [1, 2], []):
    tc23a := bijPostApplyFamily(tc23a, `bxRightFactor`, [], [1]):
    tc23 := tc23a:

    return [tc10, tc23]
end:



tc24, tc25 := op(bxBCCubedSystems(n)):
tc26, tc27 := op(bxBCCubedSystems(n-1)): # This handles equations (4) and (5),
                                          # which are analogous to (2) and (3).

# What follows handles the elimination procedure.

tc28 := bijAddn([bijMuln([bxIdentity(3), bxIdentity(n-1), bxIdentity(n)]),
                 bijMuln([bxIdentity(2), bxIdentity(n-1), bxIdentity(n-1)])]):
tc29 := bijMuln([tc28, tc26]):
tc30 := bijMuln([bxIdentity(6), tc27]):
tc31 := bijAddn([bijInvert(tc29), tc30]):
tc31a := tc31:
tc31a := bijPreApplyFamily(tc31a, `bxLeftDistribute`, [], [1]):
tc31a := bijPreApplyFamily(tc31a, `bxFlattenSum`, [1, 2], []):
tc31a := bijPostApplyFamily(tc31a, `bxLeftDistribute`, [], [2]):
```

```
tc31a := bijPostApplyFamily(tc31a, `bxFlattenSum`, [2, 2], []):
tc31a := bijPreAddPermute(tc31a, [3, 1, 2], []):
tc31a := bijPostAddPermute(tc31a, [3, 1, 2], []):
tc31a := bijPreApplyFamily(tc31a, `bxFlattenProd`, [2, 2], [3]):
tc31a := bijPreMulPermute(tc31a, [2, 1, 3], [3]):
tc31a := bijPreApplyFamily(tc31a, `bxPullOutProd`, [1, 2], [3]):
tc31a := bijPostApplyFamily(tc31a, `bxFlattenProd`, [2, 2], [3]):
tc31a := bijPostApplyFamily(tc31a, `bxPullOutProd`, [1, 2], [3]):
tc31a := bijLastTermCancel(tc31a):
tc31a := bijPreApplyFamily(tc31a, `bxFlattenProd`, [2, 2], [1]):
tc31a := bijPreApplyFamily(tc31a, `bxFlattenProd`, [2, 2], [2]):
tc31a := bijPreMulPermute(tc31a, [2, 1, 3], [2]):
tc32 := tc31a:

Lc33 := [seq(bxBCIdentity(n-2,k), k=0..n-2)]:
Lc34 := map(x->bijMuln([x, x, x]), Lc33):
tc35 := bijAddn(Lc34):

tc36 := bijMuln([bijMuln([bxIdentity(n), bxIdentity(n)]), tc24]):
tc36a := tc36:
tc36a := bijPostApplyFamily(tc36a, `bxLeftDistribute`, [], []):
tc36a := bijAddn([tc36a, bijMuln([bxIdentity(2),
                                  tc28,
                                  tc35])]):
tc36a := bijPostApplyFamily(tc36a, `bxFlattenSum`, [1, 2], []):
tc36a := bijPostApplyFamily(tc36a, `bxFlattenProd`, [2, 2], [1]):
tc36a := bijPostMulPermute(tc36a, [2, 1, 3], [1]):
tc36a := bijPostApplyFamily(tc36a, `bxFlattenProd`, [2, 2], [2]):
tc36a := bijPostMulPermute(tc36a, [2, 1, 3], [2]):
tc36a := bijPostApplyFamily(tc36a, `bxPullOutSum`, [2, 2], []):
```

```
tc37 := tc36a:

tc38a := bijPostSubs(tc37, tc32, [2]):

tc38a := bijPostApplyFamily(tc38a, `bxFlattenSum`, [2, 2], []):

tc38a := bijPostApplyFamily(tc38a, `bxFlattenProd`, [2, 2], [2]):

tc38 := tc38a:


# The bijection for the Franel recurrence is tc38.
```

Appendix C

Code for Chapter 4

This code generates the sequence $\{p_{\tau,r}(k)\}_{k=0}^{N}$. The author was too lazy to create a user interface, so both τ and N are hardcoded into the `main` function; in this example, $\tau = 1342$ and $N = 11$.

Compilation requires C++11.

C.1 main.cc

```
#include <cstdint>
#include <map>
#include <vector>
#include <string>
#include <sstream>
#include <memory>
#include <iostream>
#include "trie.h"



using namespace std;

int64_t factorial(int n);
void reduce(const vector<vector<int>>& forbidden_list_vec,
            int f, vector< vector<int> >* r);
void simplify2(const vector< vector<int> >& input,
               shared_ptr<vector< vector<int> > > output);
```

```
void printv(const vector<int>& word);
void printvv(const vector<vector<int> >& words);
string arg_to_str(int n, const vector< vector<int> >& flvec);
int64_t pattern_match(int n, const vector<int>& pat);
vector<int> repeating(int n, const vector<int>& unit);
vector<int64_t> repeating_matching_seq(int n, int codim,
                                       const vector<int>& unit);

vector<vector<int> > combinations(const int& n, const int& k);

vector<int64_t> repeating_matching_seq(int n, int codim,
                                       const vector<int>& unit) {
  int k = unit.size();
  vector<int64_t> seq;
  for (int i = 0; i <= n; ++i) {
    seq.push_back(pattern_match(codim+i*k, repeating(i, unit)));
    cout << i << endl;
  }
  return seq;
}

vector<int> repeating(int n, const vector<int>& unit) {
  if (n == 0) {
    return {};
  }
  else {
    vector<int> rv = repeating(n-1, unit);
    for (auto ch : unit) {
      rv.push_back(ch + (n-1)*unit.size());
    }
    return rv;
  }
}

int64_t pcount_flat(int n,
                    const vector< vector<int> >& forbidden_list_vec ) {
  static map<string,int64_t> cache;
  auto iter = cache.find(arg_to_str(n, forbidden_list_vec));
  if (iter != cache.end()) {
    return iter->second;
  }
  else {
    if (forbidden_list_vec.size() == 0) {
      return 0;
    }
    for (vector<int> v : forbidden_list_vec) {
      if (v.size() == 0) {
        return factorial(n);
      }
    }
    int64_t total = 0;
    for (int f = 1; f <= n; ++f) {
      vector< vector<int> > r;
      reduce(forbidden_list_vec, f, &r);
      shared_ptr<vector<vector<int> > > t(new vector<vector<int> >);
      simplify2(r, t);
      total += pcount_flat(n-1, *t);
    }
    cache[arg_to_str(n, forbidden_list_vec)] = total;
    return total;
  }
}

string arg_to_str(int n, const vector< vector<int> >& flvec) {
  stringstream ss;
  ss << n;
  ss << ": ";
  for (int i = 0; i < flvec.size(); ++i) {
    if (i != 0) {
      ss << "; ";
    }
    for (int j = 0; j < flvec[i].size(); ++j) {
      if (j != 0) {
        ss << ", ";
      }
      ss << flvec[i][j];
    }
  }
  return ss.str();
}

void reduce(const vector< vector<int> >& forbidden_list_vec, int f,
            vector< vector<int> >* r) {
  for (vector<int> v : forbidden_list_vec) {
    if (v.size() > 0 and v[0] == f) {
      vector<int> w;
      for (int i = 1; i < v.size(); ++i) {
        w.push_back(v[i]>f ? v[i]-1 : v[i]);
      }
      r->push_back(w);
    }
    else {
      bool found = false;
      for (int j : v) {
        if (j == f) {
          found = true;
        }
      }
      if (!found) {
        vector<int> w;
        for (int i = 0; i < v.size(); ++i) {
          w.push_back(v[i]>f ? v[i]-1 : v[i]);
        }
        r->push_back(w);
      }
    }
  }
}

void simplify2(const vector< vector<int> >& input,
               shared_ptr<vector< vector<int> > > output) {
  Trie t;
  for (auto word : input) {
    t.insertr(word);
  }
  t.read_all(output);
}

int64_t factorial(int n) {
  static map<int,int64_t> cache;
  auto iter = cache.find(n);
  if (iter != cache.end()) {
    return iter->second;
  }
  else {
    if (n == 0) {
      cache[n] = 1;
      return 1;
    }
    else {
      cache[n] = n*factorial(n-1);
      return cache[n];
    }
  }
}

int64_t pattern_match(int n, const vector<int>& pat) {
  int k = pat.size();
  auto combs = combinations(n, k);
  vector<vector<int> > flvec;
  for (vector<int> comb : combs) {
    vector<int> word;
    for (int entry : pat) {
      word.push_back(comb[entry-1]);
    }
    flvec.push_back(word);
  }
  return pcount_flat(n, flvec);
}
```

```
void printvv(const vector<vector<int> >& words) {
  for (auto word : words) {
    for (auto ch : word) {
      cout << ch;
    }
    cout << ", ";
  }
}

void printv(const vector<int>& word) {
  for (auto ch : word) {
    cout << ch;
  }
}

vector<vector<int> > combinations(const int& n, const int& k) {
  if (k > n) {
    return {};
  }
  else if (k == 0) {
    return {{}};
  }
  else {
    vector<vector<int> > rv;
    for (int last = k; last <= n; ++last) {
      for (vector<int> partial_comb : combinations(last-1, k-1)) {
        partial_comb.push_back(last);
        rv.push_back(partial_comb);
      }
    }
    return rv;
  }
}

int main(int argc, char** argv) {
  vector<vector<int>> v = {};
  for (auto val : repeating_matching_seq(11, 4, {1, 3, 4, 2})) {
    cout << val << endl;
  }
}
```

C.2 trie.h

```
#include <vector>
#include <map>
#include <memory>
#include <iostream>

using namespace std;

class Node {
public:
  Node();
  ~Node();
  bool has_child(int i);
  Node* get_child(int i);
  void insert_child(int i, Node* node);
  void make_end_node();
  vector<int> get_child_keys();
  bool is_end_node;
private:
  map<int,Node*> children;
};


class Trie {
 public:
  Trie();
  ~Trie();
  void insert(const vector<int>& word);
  void insertr(const vector<int>& word);
  void read_all(shared_ptr<vector< vector<int> > > words);
  void print_all();
 private:
  void descend_print(Node* node, vector<int> current_word);
  void descend(shared_ptr<vector< vector<int> > > words, Node* node,
               vector<int> current_word);
  Node* head;
};
```

C.3 trie.cc

```
#include "trie.h"


using namespace std;


Node::Node() {
  is_end_node = false;
}


Node::~Node() {
  for (const auto& i : children) {
    delete i.second;
  }
}

bool Node::has_child(int i) {
  return children.count(i) > 0;
}

Node* Node::get_child(int i) {
  return children[i];
}

void Node::insert_child(int i, Node* node) {
  children[i] = node;
}

void Node::make_end_node() {
  is_end_node = true;
}

vector<int> Node::get_child_keys() {
  vector<int> keys;
  for (const auto& i : children) {
    keys.push_back(i.first);
  }
  return keys;
}

Trie::Trie() {
  head = new Node;
}
```

```
Trie::~Trie() {
  delete head;
}

void Trie::insert(const vector<int>& word) {
  Node* current = head;
  Node* next;
  for (int i = 0; i < word.size(); ++i) {
    if (current->has_child(word[i])) {
      current = current->get_child(word[i]);
    }
    else {
      next = new Node;
      current->insert_child(word[i], next);
      current = next;
    }
  }
  current->make_end_node();
}

void Trie::insertr(const vector<int>& word) {
  Node* current = head;
  Node* next;
  for (int i = word.size() - 1; i >= 0; --i) {
    if (current->has_child(word[i])) {
      current = current->get_child(word[i]);
    }
    else {
      next = new Node;
      current->insert_child(word[i], next);
      current = next;
    }
  }
  current->make_end_node();
}

void Trie::print_all() {
  cout << "Trie: ";
  Trie::descend_print(head, {});
  cout << endl;
}

void Trie::descend_print(Node* node, vector<int> current_word) {
  if (node->is_end_node) {
    for (auto ch : current_word) {
      cout << ch;
    }
    cout << ", ";
  }
  for (int child_key : node->get_child_keys()) {
    current_word.push_back(child_key);
    Trie::descend_print(node->get_child(child_key), current_word);
    current_word.pop_back();
  }
}

void Trie::read_all(shared_ptr< vector< vector<int> > > words) {
  vector<int> current_word = {};
  Trie::descend(words, head, current_word);
```

```
}


void Trie::descend (shared_ptr<vector< vector<int> > > words, Node* node,
                    vector<int> current_word) {
  if (node->is_end_node) {
    words->push_back(current_word);
  }
  else {
    for (int child_key : node->get_child_keys()) {
      current_word.push_back(child_key);
      Trie::descend(words, node->get_child(child_key), current_word);
      current_word.pop_back();
    }
  }
}
```

References

[1] M. H. Albert, R. E. L. Aldred, M. D. Atkinson, C. Handley, and D. Holton. Permutations of a multiset avoiding permutations of length 3. *European J. Combin.*, 22(8):1021–1031, 2001.

[2] F. Balogh. The generating function enumerating words in n letters without increasing subsequences of length d and with each letter occurring r times. *ArXiv e-prints*, May 2015.

[3] Levi Ben Gerson. *Sefer Ma'aseh Hoshev.* Avignon, 1321.

[4] Miklós Bóna. Exact enumeration of 1342-avoiding permutations: a close link with labeled trees and planar maps. *J. Combin. Theory Ser. A*, 80(2):257–272, 1997.

[5] Miklós Bóna. *Combinatorics of permutations.* Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, second edition, 2012. With a foreword by Richard Stanley.

[6] Carl B. Boyer. *A history of mathematics.* John Wiley & Sons, Inc., New York, second edition, 1991. With a foreword by Isaac Asimov, Revised and with a preface by Uta C. Merzbach.

[7] Alexander Burstein. *Enumeration of words with forbidden patterns.* PhD thesis, University of Pennsylvania, 1998.

[8] G. Chapuy. The asymptotic number of 12..d-Avoiding Words with r occurrences of each letter $1, 2, ..., n$. *ArXiv e-prints*, December 2014.

[9] William Y. C. Chen, Alvin Y. L. Dai, and Robin D. P. Zhou. Ordered partitions avoiding a permutation pattern of length 3. *European J. Combin.*, 36:416–424, 2014.

[10] L. Comtet. Calcul pratique des coefficients de Taylor d'une fonction algébrique. *Enseignement Math. (2)*, 10:267–270, 1964.

[11] J. H. Conway. *Open Problems in Communication and Computation*, chapter The Weird and Wonderful Chemistry of Audioactive Decay, pages 173–188. Springer New York, New York, NY, 1987.

[12] S. B. Ekhad and D. Zeilberger. The Generating Functions Enumerating 12..d-Avoiding Words with r occurrences of each of 1,2, ... , n are D-finite for all d and all r. *ArXiv e-prints*, December 2014.

[13] Shalosh B. Ekhad and Doron Zeilberger. Proof of Conway's lost cosmological theorem. *Electron. Res. Announc. Amer. Math. Soc.*, 3:78–82 (electronic), 1997.

[14] Shalosh B. Ekhad and Doron Zeilberger. Asyrec: A maple package for computing the asymptotics of solutions of linear recurrence equations with polynomial coefficients. *Personal Journal of Shalosh B. Ekhad and Doron Zeilberger*, April 2008.

[15] Mary Celine Fasenmyer. Some generalized hypergeometric polynomials. *Bull. Amer. Math. Soc.*, 53:806–812, 1947.

[16] Mary Celine Fasenmyer. A note on pure recurrence relations. *Amer. Math. Monthly*, 56:14–17, 1949.

[17] Jérôme Franel. On a question of Laisant. *L'intermédiaire des mathématiciens*, 1(3):45–47, 1894.

[18] Jérôme Franel. On a question of J. Franel. *L'intermédiaire des mathématiciens*, 2:33–35, 1895.

[19] A. M. Garsia and S. C. Milne. Method for constructing bijections for classical partition identities. *Proc. Nat. Acad. Sci. U.S.A.*, 78(4, part 1):2026–2028, 1981.

[20] Ira M. Gessel. Symmetric functions and P-recursiveness. *J. Combin. Theory Ser. A*, 53(2):257–285, 1990.

[21] Anant Godbole, Adam Goyt, Jennifer Herdan, and Lara Pudwell. Pattern avoidance in ordered set partitions. *Ann. Comb.*, 18(3):429–445, 2014.

[22] Henry W. Gould. *Combinatorial identities.* Henry W. Gould, Morgantown, W.Va., 1972. A standardized set of tables listing 500 binomial coefficient summations.

[23] W. T. Gowers. The two cultures of mathematics. In *Mathematics: frontiers and perspectives*, pages 65–78. Amer. Math. Soc., Providence, RI, 2000.

[24] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete mathematics.* Addison-Wesley Publishing Company, Reading, MA, second edition, 1994. A foundation for computer science.

[25] Silvia Heubach and Toufik Mansour. *Combinatorics of Compositions and Words: Solutions Manual.* Chapman & Hall/CRC, 2009.

[26] Anisse Kasraoui. Pattern avoidance in ordered set partitions and words. *Adv. in Appl. Math.*, 61:85–101, 2014.

[27] Sergey Kitaev. *Patterns in permutations and words.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg, 2011. With a foreword by Jeffrey B. Remmel.

[28] Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger. *$A = B$.* A K Peters Ltd., Wellesley, MA, 1996. With a foreword by Donald E. Knuth, With a separately available computer disk.

[29] Vaughan R. Pratt. Computing permutations with double-ended queues. Parallel stacks and parallel queues. In *Fifth Annual ACM Symposium on Theory of Computing (Austin, Tex., 1973)*, pages 268–277. Assoc. Comput. Mach., New York, 1973.

[30] Lara Pudwell. Enumeration schemes for words avoiding permutations. In *Permutation patterns*, volume 376 of *London Math. Soc. Lecture Note Ser.*, pages 193–211. Cambridge Univ. Press, Cambridge, 2010.

[31] Nigel Ray and Julian West. Posets of matrices and permutations with forbidden subsequences. *Ann. Comb.*, 7(1):55–88, 2003.

[32] G. de B. Robinson. On the Representations of the Symmetric Group. *Amer. J. Math.*, 60(3):745–760, 1938.

[33] Günter Rote. Division-free algorithms for the determinant and the Pfaffian: algebraic and combinatorial approaches. In *Computational discrete mathematics*, volume 2122 of *Lecture Notes in Comput. Sci.*, pages 119–135. Springer, Berlin, 2001.

[34] C. Schensted. Longest increasing and decreasing subsequences. *Canad. J. Math.*, 13:179–191, 1961.

[35] N. Shar and D. Zeilberger. The (Ordinary) Generating Functions Enumerating 123-Avoiding Words with r occurrences of each of 1,2, ..., n are Always Algebraic. *ArXiv e-prints*, November 2014.

[36] Nathaniel Shar. Computer-assisted bijectification of franel's recurrence. *Journal of Difference Equations and Applications*, 20(12):1583–1591, 2014.

[37] Nathaniel Shar and Doron Zeilberger. The number of 1...d-avoiding permutations of length d+r for symbolic d but numeric r. *Personal Journal of Shalosh B. Ekhad and Doron Zeilberger*, April 2015.

[38] N.J.A. Sloane et al. The online encyclopedia of integer sequences.

[39] Zvezdelina E. Stankova. Forbidden subsequences. *Discrete Math.*, 132(1-3):291–316, 1994.

[40] R. P. Stanley. Differentiably finite power series. *European J. Combin.*, 1(2):175–188, 1980.

[41] Richard P. Stanley. *Catalan numbers.* Cambridge University Press, 2015.

[42] Vince Vatter. Problems and conjectures presented at the fifth international conference on permutation patterns.

[43] Vincent Vatter. Enumeration schemes for restricted permutations. *Combin. Probab. Comput.*, 17(1):137–159, 2008.

[44] Wikipedia. Young tableaux for 541 partition, 2007-2008. By users Arichnad, Kilom691, and RobHar. Permission for use granted under Creative Commons Attribution-Share Alike license v3.0. See `https://creativecommons.org/licenses/by-sa/3.0/legalcode`.

[45] Wikipedia. Young tableaux for 541 partition, 2007-2009. By users Arichnad and Kilom691. Permission for use granted under Creative Commons Attribution-Share Alike license v3.0. See `https://creativecommons.org/licenses/by-sa/3.0/legalcode`.

[46] Wikipedia. Permutation pattern — Wikipedia, the free encyclopedia, 2016. [Online; accessed 26-February-2016].

[47] Herbert S. Wilf. What is an answer? *Amer. Math. Monthly*, 89(5):289–292, 1982.

[48] Herbert S. Wilf. Mathematics: An experimental science, 2008.

[49] Philip Matchett Wood and Doron Zeilberger. A translation method for finding combinatorial bijections. *Annals of Combinatorics*, 13(3):383–402, 2009.

[50] Doron Zeilberger. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.*, 32(3):321–368, 1990.

[51] Doron Zeilberger. Enumeration schemes and, more importantly, their automatic generation. *Ann. Comb.*, 2(2):185–195, 1998.

[52] Doron Zeilberger. A snappy proof that 123-avoiding words are equinumerous with 132-avoiding words. *Personal Journal of Shalosh B. Ekhad and Doron Zeilberger*, April 2005.

[53] Doron Zeilberger. On Vince Vatter's brilliant extension of Doron Zeilberger's enumeration schemes for counting Herb Wilf's classes. *Personal Journal of Shalosh B. Ekhad and Doron Zeilberger*, December 2006.