Counting Matrices that are Squares

Victor S. Miller

Center for Communications Research - Princeton

28 January, 2021

A request from Neil Sloane on 7 May 2013

“Count 0/1 matrices which are squares of such matrices”

What I answered

- Count squares in $\mathrm{Mat}_n(\mathbb{F}_q)$ and $\mathrm{GL}_n(\mathbb{F}_q)$, where $q = 2^m$.

What I answered

- Count squares in $\mathrm{Mat}_n(\mathbb{F}_q)$ and $\mathrm{GL}_n(\mathbb{F}_q)$, where $q = 2^m$.
- $a(n) = \#$ squares in $\mathrm{Mat}_n(\mathbb{F}_2)$
- $b(n) = \#$ squares in $\mathrm{GL}_n(\mathbb{F}_2)$

What I answered

- Count squares in $\mathrm{Mat}_n(\mathbb{F}_q)$ and $\mathrm{GL}_n(\mathbb{F}_q)$, where $q = 2^m$.
- $a(n) = \#$ squares in $\mathrm{Mat}_n(\mathbb{F}_2)$
- $b(n) = \#$ squares in $\mathrm{GL}_n(\mathbb{F}_2)$
- $a(n) =$
  $2, 10, 260, 31096, 13711952, 28275659056, 224402782202048, \ldots$
- $b(n) =$
  $1, 3, 126, 11340, 5940840, 12076523928, 95052257647200, \ldots.$

A good problem

- A good problem is one in which the solution gets you to learn about other things.

A good problem

- A good problem is one in which the solution gets you to learn about other things.
- Word maps in groups.

A good problem

- A good problem is one in which the solution gets you to learn about other things.
- Word maps in groups.
- The cycle index for matrix algebras.

A good problem

- A good problem is one in which the solution gets you to learn about other things.
- Word maps in groups.
- The cycle index for matrix algebras.
- Dickson's formula for the size of conjugacy classes.

A good problem

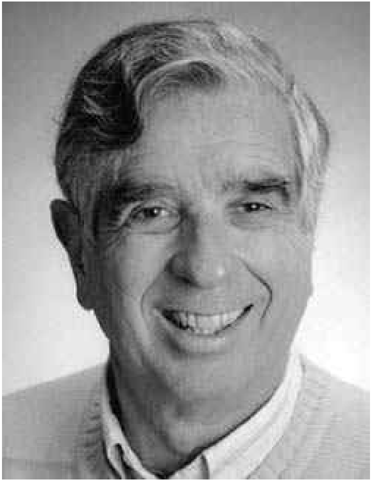
- A good problem is one in which the solution gets you to learn about other things.
- Word maps in groups.
- The cycle index for matrix algebras.
- Dickson's formula for the size of conjugacy classes.
- Meinardus' theorem for the asymptotics of classes of partitions.

A good problem

- A good problem is one in which the solution gets you to learn about other things.
- Word maps in groups.
- The cycle index for matrix algebras.
- Dickson's formula for the size of conjugacy classes.
- Meinardus' theorem for the asymptotics of classes of partitions.
- Using logarithmic derivatives to get faster calculation for products of generating functions.

What is an Answer?

We have a class of combinatorial objects C_n. An *good answer* is an algorithm to calculate $|C_n|$ in time $o(|C_n|)$. Of course, the faster the better.



WHAT IS AN ANSWER?

HERBERT S. WILF
Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104

In many branches of pure mathematics it can be surprisingly hard to recognize when a question has, in fact, been answered. A clearcut proof of a theorem or the discovery of a counterexample leaves no doubt in the reader's mind that a solution has been found. But when an "explicit solution" to a problem is given, it may happen that more work is needed to evaluate that "solution," in a particular case, than exhaustively to examine all of the possibilities directly from the original formulation of the problem. In such a situation, other things being equal, we may justifiably question whether the problem has in fact been solved.

Examples of this sort can turn up anywhere, but here we will concentrate on problems in combinatorial mathematics, specifically those of the type "how many—are there?" Such enumeration problems lie at the heart of the subject, and it is important to be able to recognize solutions when they appear. The point, of course, is that sometimes the "answer" is presented as a formula that is so messy and long, and so full of factorials and sign alternations and whatnot, that we may feel that the disease was preferable to the cure.

An answer to such an enumeration question may be given by means of a generating function, a recurrence relation, or by an explicit formula. Each of these is, in essence, just an algorithm for the computation of the counting sequence that is to be determined.

How do we judge the usefulness of such answers? Obviously we might be able to do many things with the answer, such as to make asymptotic estimates, to discover congruence relations, to delight in its elegance, and so forth. We're going to restrict attention here to the appraisal of solutions from the point of view of how easily they allow us to calculate the number of objects in the set that is being studied.

The quality of such formulas should therefore be judged by the usual combination of esthetic and quantitative benchmarks that are used on algorithms. In particular, the quantitative criterion is the computational complexity: the amount of work required to get an answer. We suggest here that the same criterion should be applied to enumeration formulas. We will see that a corollary of this attitude is that our decision as to what constitutes an answer may be time-dependent: as faster algorithms for listing the objects become available, a proposed formula for counting the objects will have to be comparably faster to evaluate.

For concreteness, suppose that for each integer $n > 0$ there is a set S_n that we want to count. Let $f(n) = |S_n|$ (the cardinality of S_n), for each n.

Suppose further that a certain formula has been found, say

$$f(n) = \text{Formula}(n) \qquad (n = 1, 2, \ldots) \tag{1}$$

in which Formula(n) may involve various multiple summation signs extending over various sets and various complicated summands, etc.

In order to evaluate the "answer" (1), let's look at the competition. To insure my own immortality in the subject, I am now going to show you a simple formula that "answers" all such questions at once. If you're ready, then, here it is:

$$f(n) = \sum_{S_n} 1. \tag{2}$$

Well, anyway, the summand is elegant, even if the range of summation is a bit untidy.

Now (2) is unacceptable in polite society as an answer, despite its elegant appearance, because it is just a restatement of the question, and it does not give us a tool for calculating $f(n)$ that we

In addition to his other accomplishments (see this volume, p. 4) the author is currently one of the associate editors of this MONTHLY.

Herb Wilf: American Mathematical Monthly, May 1982

Exhaustion for the original problem

- Generate all $A \in \mathrm{Mat}_n(\mathbb{F}_2)$, interpret A as a bit string of length n^2 and set bit corresponding to A^2.

Exhaustion for the original problem

- Generate all $A \in \mathrm{Mat}_n(\mathbb{F}_2)$, interpret A as a bit string of length n^2 and set bit corresponding to A^2.
- *Gray Code*: enumerate all N-bit strings by flipping one bit at a time.

Exhaustion for the original problem

- Generate all $A \in \mathrm{Mat}_n(\mathbb{F}_2)$, interpret A as a bit string of length n^2 and set bit corresponding to A^2.
- *Gray Code*: enumerate all N-bit strings by flipping one bit at a time.
- Use Gray code: $B := A^2$, If $A \leftarrow A + E$, then $B \leftarrow B + EA + AE + E^2$, where E has only 1 bit set.
- EA and AE select a row/column from A, $E^2 = E$ or 0.

Exhaustion for the original problem

- Generate all $A \in \mathrm{Mat}_n(\mathbb{F}_2)$, interpret A as a bit string of length n^2 and set bit corresponding to A^2.
- *Gray Code*: enumerate all N-bit strings by flipping one bit at a time.
- Use Gray code: $B := A^2$, If $A \leftarrow A + E$, then $B \leftarrow B + EA + AE + E^2$, where E has only 1 bit set.
- EA and AE select a row/column from A, $E^2 = E$ or 0.
- $a(n), n = 1, \ldots, 5$ almost instantaneous, $a(6)$ takes about 1 hour on my iMac.

Exhaustion for the original problem

- Generate all $A \in \mathrm{Mat}_n(\mathbb{F}_2)$, interpret A as a bit string of length n^2 and set bit corresponding to A^2.
- *Gray Code*: enumerate all N-bit strings by flipping one bit at a time.
- Use Gray code: $B := A^2$, If $A \leftarrow A + E$, then $B \leftarrow B + EA + AE + E^2$, where E has only 1 bit set.
- EA and AE select a row/column from A, $E^2 = E$ or 0.
- $a(n), n = 1, \ldots, 5$ almost instantaneous, $a(6)$ takes about 1 hour on my iMac.
- Can't go much further since time is $\approx 2^{n^2}$.

A good strategy, that sometimes works

- Find a "nice" set $X \supset C_n$, where $C_n = \{x \in X : P(x)\}$, and P is some predicate.

A good strategy, that sometimes works

- Find a "nice" set $X \supset C_n$, where $C_n = \{x \in X : P(x)\}$, and P is some predicate.
- Find a large group G acting on the right on X, *compatibly* with P: $\mathcal{P}(xg) = P(x)$ for all $x \in X, g \in G$.

A good strategy, that sometimes works

- Find a "nice" set $X \supset C_n$, where $C_n = \{x \in X : P(x)\}$, and P is some predicate.
- Find a large group G acting on the right on X, *compatibly* with P: $\mathcal{P}(xg) = P(x)$ for all $x \in X, g \in G$.
- Enumerate all *orbits* aG such that $P(aG)$ is true.
- When G acts by conjugation, the orbits are called *conjugacy classes*, and action is written $x \mapsto x^g$.
- $G_a := \{g \in G : ag = a\}$ the *stabilizer* of a. Called *centralizer* for conjugacy classes.
- X/G = representatives of the orbits.

A good strategy, that sometimes works

- Find a "nice" set $X \supset C_n$, where $C_n = \{x \in X : P(x)\}$, and P is some predicate.
- Find a large group G acting on the right on X, *compatibly* with P: $\mathcal{P}(xg) = P(x)$ for all $x \in X, g \in G$.
- Enumerate all *orbits* aG such that $P(aG)$ is true.
- When G acts by conjugation, the orbits are called *conjugacy classes*, and action is written $x \mapsto x^g$.
- $G_a := \{g \in G : ag = a\}$ the *stabilizer* of a. Called *centralizer* for conjugacy classes.
- X/G = representatives of the orbits.
- Answer $|C_n| = \sum_{a \in X/G, P(a)} \frac{|G|}{|G_a|}$.

A good strategy, that sometimes works

- Find a "nice" set $X \supset C_n$, where $C_n = \{x \in X : P(x)\}$, and P is some predicate.
- Find a large group G acting on the right on X, *compatibly* with P: $\mathcal{P}(xg) = P(x)$ for all $x \in X, g \in G$.
- Enumerate all *orbits* aG such that $P(aG)$ is true.
- When G acts by conjugation, the orbits are called *conjugacy classes*, and action is written $x \mapsto x^g$.
- $G_a := \{g \in G : ag = a\}$ the *stabilizer* of a. Called *centralizer* for conjugacy classes.
- X/G = representatives of the orbits.
- Answer $|C_n| = \sum_{a \in X/G, P(a)} \frac{|G|}{|G_a|}$.
- Sometimes can count all orbits aG with the same value of $|G_a|$ to get a shorter sum, or use generating functions if orbits decompose nicely.

Word Maps

- G: group, w a word in the free group on r generators.

Word Maps

- G: group, w a word in the free group on r generators.
- $w : G^r \to G$ given by plugging in elements of G.

Word Maps

- G: group, w a word in the free group on r generators.
- $w : G^r \to G$ given by plugging in elements of G.
- Questions: What is the image of w?

Word Maps

- G: group, w a word in the free group on r generators.
- $w : G^r \to G$ given by plugging in elements of G.
- Questions: What is the image of w?
- w induces a measure on G. What are its properties?

Word Maps

- G: group, w a word in the free group on r generators.
- $w : G^r \to G$ given by plugging in elements of G.
- Questions: What is the image of w?
- w induces a measure on G. What are its properties?

Theorem (Michael Larsen, 2004)

Let G_n be a sequence of simple groups, $|G_n| \to \infty$, and w a non-trivial word. Then

$$\lim_{n\to\infty} \frac{\log |w(G_n)|}{\log |G_n|} = 1.$$

Word Maps

- G: group, w a word in the free group on r generators.
- $w : G^r \to G$ given by plugging in elements of G.
- Questions: What is the image of w?
- w induces a measure on G. What are its properties?

Theorem (Michael Larsen, 2004)

Let G_n be a sequence of simple groups, $|G_n| \to \infty$, and w a non-trivial word. Then

$$\lim_{n\to\infty} \frac{\log |w(G_n)|}{\log |G_n|} = 1.$$

Shows that the image $w(G)$ is "big". So $a(n)$ grows approximately like 2^{n^2}.

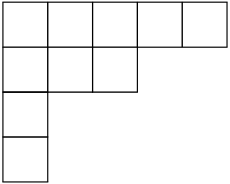
Partitions

We need *partitions* to describe the conjugacy classes.

- *Partition*: A non-decreasing sequence of nonnegative integers, all but a finite number are 0: $\lambda := \lambda_1 \geq \lambda_2 \geq \dots$, $\mathcal{P}$: set of all partitions.
- If $|\lambda| := \sum_i \lambda_i = n$, we write $\lambda \vdash n$: λ is a partition of n. The $0 \neq \lambda_i$ are the *parts* of λ.

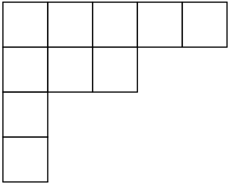
Partitions

We need *partitions* to describe the conjugacy classes.

- *Partition*: A non-decreasing sequence of nonnegative integers, all but a finite number are 0: $\lambda := \lambda_1 \geq \lambda_2 \geq \dots$, $\mathcal{P}$: set of all partitions.
- If $|\lambda| := \sum_i \lambda_i = n$, we write $\lambda \vdash n$: λ is a partition of n. The $0 \neq \lambda_i$ are the *parts* of λ.
- *Young diagram*: $\lambda = (5, 3, 1, 1, 0, \dots) \rightarrow$ .

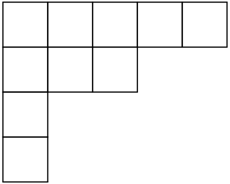
Partitions

We need *partitions* to describe the conjugacy classes.

- *Partition*: A non-decreasing sequence of nonnegative integers, all but a finite number are 0: $\lambda := \lambda_1 \geq \lambda_2 \geq \dots$, $\mathcal{P}$: set of all partitions.
- If $|\lambda| := \sum_i \lambda_i = n$, we write $\lambda \vdash n$: λ is a partition of n. The $0 \neq \lambda_i$ are the *parts* of λ.
- *Young diagram*: $\lambda = (5, 3, 1, 1, 0, \dots) \to$ .
- *Conjugate*: $\lambda'_i := \#\{j : \lambda_j \geq i\}$. Flip the diagram.

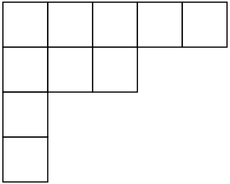
Partitions

We need *partitions* to describe the conjugacy classes.

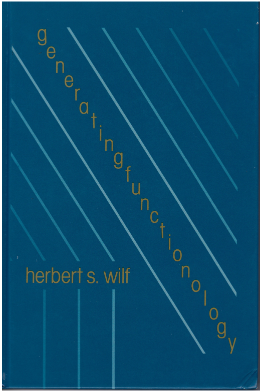
- *Partition*: A non-decreasing sequence of nonnegative integers, all but a finite number are 0: $\lambda := \lambda_1 \geq \lambda_2 \geq \dots$, $\mathcal{P}$: set of all partitions.
- If $|\lambda| := \sum_i \lambda_i = n$, we write $\lambda \vdash n$: λ is a partition of n. The $0 \neq \lambda_i$ are the *parts* of λ.
- *Young diagram*: $\lambda = (5, 3, 1, 1, 0, \dots) \to$  .
- *Conjugate*: $\lambda'_i := \#\{j : \lambda_j \geq i\}$. Flip the diagram.
- *Multiplicity*: If $\lambda \vdash n$, and $i > 0$, multiplicity of i: $m_i(\lambda) := \#\{j : \lambda_j = i\}$

Partitions

We need *partitions* to describe the conjugacy classes.

- *Partition*: A non-decreasing sequence of nonnegative integers, all but a finite number are 0: $\lambda := \lambda_1 \geq \lambda_2 \geq \ldots$, $\mathcal{P}$: set of all partitions.
- If $|\lambda| := \sum_i \lambda_i = n$, we write $\lambda \vdash n$: λ is a partition of n. The $0 \neq \lambda_i$ are the *parts* of λ.
- *Young diagram*: $\lambda = (5, 3, 1, 1, 0, \ldots) \to$  .
- *Conjugate*: $\lambda'_i := \#\{j : \lambda_j \geq i\}$. Flip the diagram.
- *Multiplicity*: If $\lambda \vdash n$, and $i > 0$, multiplicity of i: $m_i(\lambda) := \#\{j : \lambda_j = i\}$
- Asymptotics: $p(n) = \#\{\lambda \vdash n\} \sim \frac{1}{4\sqrt{3}n} \exp(\pi\sqrt{2n/3})$.

A simpler, related problem

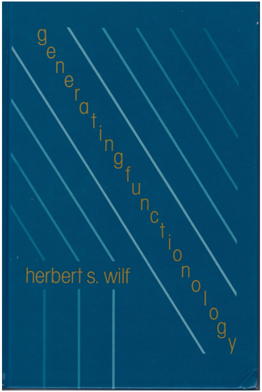


Square permutations

How many permutations are squares of other permutations?

- Conjugacy classes $\leftrightarrow$ partitions.

A simpler, related problem

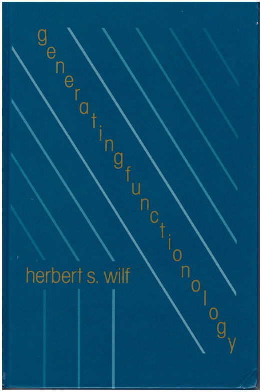


Square permutations

How many permutations are squares of other permutations?

- Conjugacy classes $\leftrightarrow$ partitions.
- $a(n) = \#$ squares in S_n.

A simpler, related problem

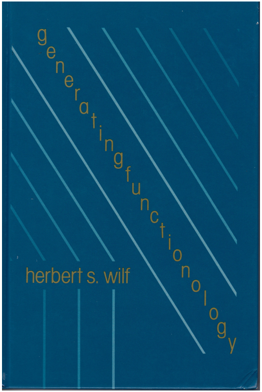


Square permutations

How many permutations are squares of other permutations?

- Conjugacy classes $\leftrightarrow$ partitions.
- $a(n) = \#$ squares in S_n.
- Squares invariant under conjugation.

A simpler, related problem

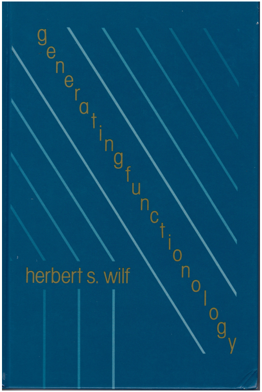


Square permutations

How many permutations are squares of other permutations?

- Conjugacy classes $\leftrightarrow$ partitions.
- $a(n) = \#$ squares in S_n.
- Squares invariant under conjugation.
- Stabilizer size for λ: $\prod_j m_j(\lambda)! j^{m_j(\lambda)}$.

A simpler, related problem



Square permutations

How many permutations are squares of other permutations?

- Conjugacy classes $\leftrightarrow$ partitions.
- $a(n) = \#$ squares in S_n.
- Squares invariant under conjugation.
- Stabilizer size for λ: $\prod_j m_j(\lambda)! j^{m_j(\lambda)}$.
- Conjugacy classes of squares: $m_{2j}(\lambda)$ even.

A product of generating functions

- $\lambda \vdash n$ if and only if $\sum_j j m_j(\lambda) = n$.
- The centralizer size is a product of independent factors $f_j(x)$: the generating function of m_j.

$$j \text{ even:} \quad \sum_{m \geq 0,\ \text{even}} \frac{x^{jm}}{j^m m!} = \cosh\left(\frac{x^j}{j}\right)$$

$$j \text{ odd:} \quad \sum_{m \geq 0} \frac{x^{jm}}{j^m m!} = \exp\left(\frac{x^j}{j}\right).$$

- Their product

$$\sum_{n \geq 1} \frac{a(n) x^n}{n!} = \prod_j f_j(x) = \sqrt{\frac{1+x}{1-x}} \prod_{j \geq 1} \cosh\left(\frac{x^{2j}}{2j}\right).$$

- Bender: the probability that a permutation is a square $\sim \frac{2}{\sqrt{\pi n}} \prod_{k \geq 1} \cosh\left(\frac{1}{2k}\right)$.

Application to matrices

- Let $X = \mathrm{Mat}_n(F)$ all $n \times n$ matrices, $P(A)$ true if and only if A is a square.
- Let $G = \mathrm{GL}_n(F)$, invertible matrices. Acts on X by conjugation $A^U := UAU^{-1}$ and is compatible wth P. Orbits are *conjugacy classes*. Usually write $\mathcal{C}(a) = G_a$.
- If A is conjugate to B we write $A \sim B$.

Polynomials

- $\mathcal{I}(q)$: monic polynomials irreducible over $\mathbb{F}_q$.
- $\mathcal{I}(q)_d$: members of $\mathcal{I}(q)$ of degree d.
- $|\mathcal{I}(q)_d| = \frac{1}{d}\sum_{e|d} \mu(d/e)q^e$, where μ is the Möbius function.
- ϕ is monic and r a positive integer: $\phi^{(r)}(x)$ the polynomial whose roots are the r-th powers (with multiplicity) of the roots of ϕ.

Generalized companion matrices

- If ϕ is a monic polynomial denote by $M(\phi)$ its companion matrix.

Generalized companion matrices

- If ϕ is a monic polynomial denote by $M(\phi)$ its companion matrix.
- If $\lambda \in \mathcal{P}$ denote by $M(\lambda, \phi) := \bigoplus_j M(\phi^{\lambda_j})$.

Generalized companion matrices

- If ϕ is a monic polynomial denote by $M(\phi)$ its companion matrix.
- If $\lambda \in \mathcal{P}$ denote by $M(\lambda, \phi) := \bigoplus_j M(\phi^{\lambda_j})$.
- Call the elements conjugate to $M(\lambda, \phi)$, for $\phi \in \mathcal{I}(q)$, a *primitive* conjugacy classes.
- If $\phi, \psi \in \mathcal{I}(q)$, and $\lambda, \nu \in \mathcal{P}$ then $M(\lambda, \phi) \sim M(\nu, \psi)$ if and only if $(\lambda, \phi) = (\nu, \psi)$.

Frobenius normal form

Georg Ferdinand Frobenius.

Every conjugacy class in $\mathrm{Mat}_n(F)$ is the conjugacy class of a direct sum of distinct primitive conjugacy classes.

Generating functions from the primitive classes

- We work in $\mathbb{F}_q$ for $q = 2^m$ for some m.
- It suffices to find squares of the primitive conjugacy classes in $M(\lambda, \phi)$.
- By direct calculation $M(\lambda, \phi)^2 \sim M(\Psi(\lambda), \phi^{(2)})$, for a particular partition $\Psi(\lambda)$, independent of ϕ.
- Let $\mathcal{S} := \{\Psi(\lambda) : \lambda \in \mathcal{P}\}$.
- Let $F_\phi(X) := \sum_{\lambda \in \mathcal{S}} \frac{X^{\deg(\phi)|\lambda|}}{|\mathcal{C}(M(\lambda,\phi))|}$, the "local" generating function for ϕ.
- Then $\sum_n \frac{a(n)}{|\operatorname{GL}_n(\mathbb{F}_q)|} X^n = \prod_\phi F_\phi(X)$.
- $|\mathcal{C}(M(\lambda, \phi))|$: only dependence on ϕ is by $\deg(\phi)$.

Characterizing the conjugacy classes of squares

- If k is a part of λ it yield parts $\lfloor k/2 \rfloor, \lceil k/2 \rceil$ in $\Psi(\lambda)$.
- $\Psi(\lambda)$ is characterized by $m_i(\Psi(\lambda)) = 2m_{2i}(\lambda) + m_{2i-1}(\lambda) + m_{2i+1}(\lambda)$ for all i.
- In a field of characterstic 2, $\phi \mapsto \phi^{(2)}$ is a permutation on $\mathcal{I}(q)$.

Aha!?

- For each n exhaust over $\lambda \vdash n$ to find all $\Psi(\lambda)$.
- Get the sequence $1, 1, 2, 3, 4, 5, 7, 10,$ $13, 16, 21, 28, 35, 43, 55, 70, \ldots$ which is A006950 in OEIS.
- "Number of partitions of n in which each even part occurs with even multiplicity. There is no restriction on the odd parts.": same as for squares in S_n.
- "Also the number of partitions of n in which all odd parts occur with multiplicity 1. There is no restriction on the even parts."
- The $\Psi(\lambda)$ don't have either property!
- But their conjugates do!
- Constructive theorem: $\nu = \Psi(\lambda)$ for some λ if and only if $m_{2i-1}(\nu') \leq 1$ for all i.

A generating function for the partitions

- Let $a'(n)$ = number of $\lambda \vdash n$, such that $m_{2j-1}(\lambda') \leq 1$.
- Algorithm exhausts over these so we need to estimate their number.
- Generating function in A006950 is Ramanujan's mock theta function

$$\vartheta(X) := \sum_{n \geq 1} a'(n) X^n = \prod_{k \geq 1} \frac{1 + X^{2k-1}}{1 - X^{2k}} = \prod_{n \geq 1, n \not\equiv 2 \bmod 4} (1 - X^n)^{-1}.$$

Counting Conjugacy Classes of Squares

- Each $\phi \in \mathcal{I}(q)$ has a set S_ϕ of allowed $\lambda \in \mathcal{P}$.

Counting Conjugacy Classes of Squares

- Each $\phi \in \mathcal{I}(q)$ has a set S_ϕ of allowed $\lambda \in \mathcal{P}$.
- Generating function for number of classes is $\prod_{\phi \in \mathcal{I}(q)} \sum_{\lambda \in S_\phi} X^{|\lambda| \deg \phi}$.

Counting Conjugacy Classes of Squares

- Each $\phi \in \mathcal{I}(q)$ has a set S_ϕ of allowed $\lambda \in \mathcal{P}$.
- Generating function for number of classes is $\prod_{\phi \in \mathcal{I}(q)} \sum_{\lambda \in S_\phi} X^{|\lambda| \deg \phi}$.
- In our case all S_ϕ are the same.
- So generating function for the number of conjugacy classes is $\prod_d \vartheta(X^d)^{|\mathcal{I}(q)_d|}$.

Counting Conjugacy Classes of Squares

- Each $\phi \in \mathcal{I}(q)$ has a set S_ϕ of allowed $\lambda \in \mathcal{P}$.
- Generating function for number of classes is $\prod_{\phi \in \mathcal{I}(q)} \sum_{\lambda \in S_\phi} X^{|\lambda| \deg \phi}$.
- In our case all S_ϕ are the same.
- So generating function for the number of conjugacy classes is $\prod_d \vartheta(X^d)^{|\mathcal{I}(q)_d|}$.
- Useful trick

$$1 - qX = \prod_{d=1}^{\infty} (1 - X^d)^{|\mathcal{I}(q)_d|}.$$

- Yields generating function for number of classes

$$\prod_{n \geq 1, n \not\equiv 2 \bmod 4} (1 - qX^n)^{-1}.$$

Counting Conjugacy Classes of Squares

- Each $\phi \in \mathcal{I}(q)$ has a set S_ϕ of allowed $\lambda \in \mathcal{P}$.
- Generating function for number of classes is $\prod_{\phi \in \mathcal{I}(q)} \sum_{\lambda \in S_\phi} X^{|\lambda| \deg \phi}$.
- In our case all S_ϕ are the same.
- So generating function for the number of conjugacy classes is $\prod_d \vartheta(X^d)^{|\mathcal{I}(q)_d|}$.
- Useful trick
$$1 - qX = \prod_{d=1}^{\infty} (1 - X^d)^{|\mathcal{I}(q)_d|}.$$
- Yields generating function for number of classes
$$\prod_{n \geq 1, n \not\equiv 2 \bmod 4} (1 - qX^n)^{-1}.$$
- Converges for $|X| < 1/q$ with a simple pole at $X = 1/q$. Thus # classes for n is $\sim cq^n$ for some $c > 0$.

Generating Function for the number of squares

- Local function for $\phi \in \mathcal{I}(q)$:

$$F_\phi(X) = \sum_{\lambda \in \mathcal{S}_\phi} \frac{X^{\deg(f)|\lambda|}}{|\mathcal{C}(M(\lambda,\phi))|}.$$

- As above, all $\mathcal{S}_\phi$ are the same.
- $|\mathcal{C}(M(\lambda,\phi))|$ only dependence on ϕ is by $\deg(\phi)$ (see next slide).
- So

$$\sum_n \frac{a(n)}{|\mathrm{GL}_n(\mathbb{F}_q)|} X^n = \prod_{d \geq 1} F_d(X)^{|\mathcal{I}(q)_d|}.$$

Size of the centralizer

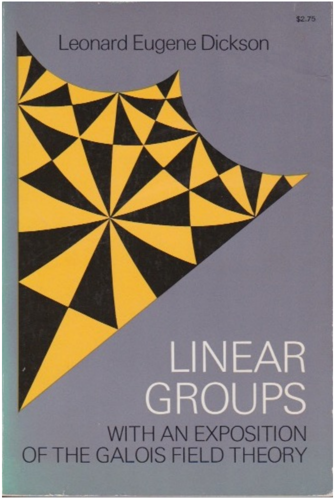
- Frobenius showed:

$$\dim_{\mathbb{F}_q}\{U \in \mathrm{Mat}_n(\mathbb{F}_q) : UM(\lambda,\phi) = M(\lambda,\phi)U\} = \deg(\phi)\sum_i {\lambda_i'}^2.$$

- Yields $q^{\deg(\phi)\sum_i \lambda'^2}$ matrices U.
- We need a correction factor since U must be invertible.
- Let $r_n(q) = \frac{|\mathrm{GL}_n(\mathbb{F}_q)|}{|\mathrm{Mat}_n(\mathbb{F}_q)|}$: the probability that a matrix is invertible.
- Dickson: Multiply by $\prod_i r_{m_i(\lambda)}(q^{\deg(\phi)})$.
- Note: $r_\infty(q) = \lim_{n\to\infty} r_n(q) > 0$, $r_\infty(2) \approx 0.28878809508660242$.

A forgotten theorem

Leonard Eugene Dickson



Dickson proved the above in 1900, but many subsequent authors appeared to be unaware of this!

Asymptotics

- From calculated values it appears that $a(n) \sim c_1 2^{n^2}, b(n) \sim c_2 2^{n^2}$ for some $c_1, c_2 > 0$.
- In other words the probability that a matrix is a square has a nonzero limit as $n \to \infty$.
- Wall proved a result like this for counting semisimple classes.
- An analysis of his method shows that it applies more generally, in particular to our problem.
- A bit different than for S_n, where the probability goes to 0.

Bounding the running time

- We iterate over all partitions $\lambda \vdash n$ for $n \leq N$, and $m_{2i-1}(\lambda') \leq 1$.

Bounding the running time

- We iterate over all partitions $\lambda \vdash n$ for $n \leq N$, and $m_{2i-1}(\lambda') \leq 1$.
- Meinardus' Theorem gives asymptotics for the number of restricted partitions

$$a'(n) \sim \frac{1}{4\sqrt{2n}} \exp(\pi\sqrt{n/2}).$$

Still super-polynomial.

A useful speedup

- We need to calculate things like $F(X) = \prod_{d=1}^{n} f_d(X)^{n_d}$, where $f_d(X)$ are power series with constant term 1.
- Take logarithmic derivatives

$$\frac{F'(X)}{F(X)} = \sum_{d=1}^{n} n_d \frac{f_d'(X)}{f_d(X)},$$

- Want first $n+1$ terms. Treat those as unknowns, and 0-th term is 1.
- Get a lower triangular linear system.
- Using this trick, sped up calculation for $n = 14$ from 318 seconds to under 1 second.

Other powers

- Counting squares in characteristic 2 is easier because $\phi \mapsto \phi^{(2)}$ is one-to-one.
- In odd characteristic one needs to break up the polynomials in $\mathcal{I}(q)$ into different classes.
- Some $\phi^{(2)}$ are squares of irreducibles, so the partition changes.
- Need to use counting results of Stephen Cohen on decomposition of $\phi(x^r)$, for ϕ irreducible.
- Similar but more complicated formulas.

Further Questions

- Right now we exhaust over restricted partitions. Is there a polynomial time algorithm in n?
- Finer asymptotics for $a(n), b(n)$.
- Analogous results for powers that are relatively prime to the characteristic of the field.
- Faster algorithm for square roots.
- The sequence of the the maximum number of square roots of a matrix. Related to counting integer points in a polytope.

Acknowledgements

- Neil Sloane, for posing the problem and for the OEIS!
- Bob Guralnick and Jason Fulman for helpful correspondence.
- Richard Stong for explaining Wall's analytic method.

Doron and Herb