CDM

Number Theory II

K. Sutner

Carnegie Mellon University

12-numberth 2017/12/15 23:15

Density

For any word $w \in \mathbf{2}^k$ we write

$$\Delta(w) = \frac{\#_1 w}{k}$$

for the density of w, and $\Delta(W)$ for the average density of a set of words $W \subseteq \mathbf{2}^k$.

Clearly $\Delta(\mathbf{2}^k) = 1/2$.

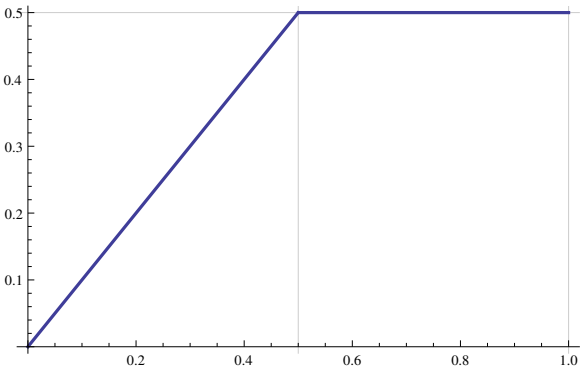
Let $0 \leq \alpha \leq 1$ and write

$$\mathbf{2}^{k,\alpha} = \{\, w \in \mathbf{2}^k \mid \Delta(w) \leq \alpha \,\}$$

How about all words of density up to α?

$$\lim_{k\to\infty} \Delta(\mathbf{2}^{k,\alpha}) = ???$$

A Surprise



Exercise

Exercise

Find an intuitive reason why the last picture makes sense.

Exercise

Give an actual proof of the corresponding claim.

Extra credit, give your solution directly to me.

Recall: Little Fermat

Theorem (Fermat's Little Theorem)

If p is prime and coprime to a, then $a^{p-1} = 1 \pmod{p}$.

Last time we gave a somewhat pedestrian proof by multiplying out the modular numbers in $\mathbb{Z}_p^*$.

We will now start to develop more powerful machinery that makes it possible to generalize this result.

Exponentiation in Modular Numbers

Inspired by FLT, let's take a closer look at exponentiation modulo m.

First recall that we can compute a^k in $O(\log k)$ modular multiplications using the standard squaring trick.

```
z = 1;
while( k > 0 ){
    if( k odd ) z = z * a mod m;
    a = a * a mod m;
    k = k/2;
}
return z;
```

This uses $ds(k) + dc(k)$ multiplications where

$ds(k)$ is the digit sum of k (number of 1's in binary expansion), and

$dc(k)$ is the total number of digits in the binary expansion.

Warning

Is there any other way we could speed up exponentiation modulo m?

It is in general **not** the case that

$$a^k = a^{k \bmod m} \pmod{m}$$

But by FLT we get

$$a^k = a^{k \bmod p-1} \pmod{p}$$

whenever p is prime. Could this be coincidence? Nah ...

Lemma

$$a^k = a^{k \bmod \varphi(m)} \pmod{m}$$

provided a and m are coprime.

A Brilliant Idea: Groups

Definition

A group $G = \langle G, *, 1 \rangle$ is a structure with a binary operation $*$ and a special element 1 such that

Associativity $*$ is associative $x * (y * z) = (x * y) * z$.

Neutral Element 1 is a neutral element $1 * x = x * 1 = x$.

Inverse Element For every element x there is an inverse x' such that

$$x * x' = x' * x = 1.$$

The order of G is its cardinality.

Claim

The inverse is uniquely determined.

Proof. Suppose $x * x' = 1 = x * x''$. Multiplying by x' we get $x' * (x * x') = x' * (x * x'')$. By associativity we get $1 * x' = 1 * x''$ and thus $x' = x''$. □

Abelian Groups

Definition

A group is commutative or Abelian if $x * y = y * x$ for all x and y.

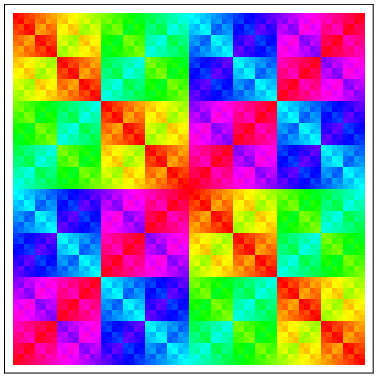
Notation:

It is customary to write Abelian groups additively as $\langle G, +, 0 \rangle$.

General groups are written multiplicatively as $\langle G, *, 1 \rangle$ or $\langle G, \cdot, 1 \rangle$. As usual, the multiplication operator is often omitted: xy instead of $x * y$.

The inverse is correspondingly written $-x$ in additive notation and x^{-1} in multiplicative notation.

Cayley Tables



For finite groups we can simply write down a multiplication table. Above the table for bit-wise xor on on 6-bit numbers.

Classical Examples

Example

Integers (rationals, reals, complexes) with addition form a group; neutral element is 0.

Example

Non-zero rationals (reals, complexes) with multiplication form a group; neutral element is 1.

Example

Modular numbers with addition form a group; neutral element is 0.

Example

Modular numbers coprime with modulus m with multiplication form a group; neutral element is 1.

More Classical Examples

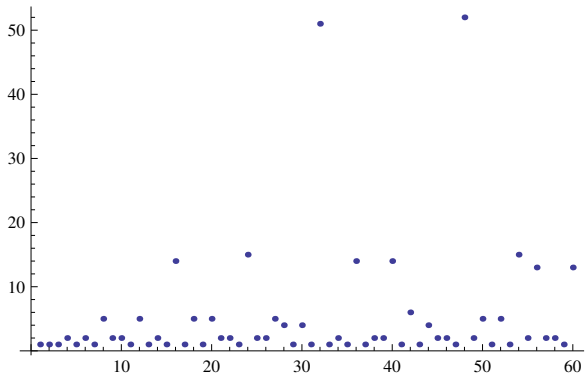
Example

The set of all permutations $f : A \to A$ for some arbitrary set A forms a group with functional composition as operation. The neutral element is the identity function. This is the symmetric group on A.

Example

The set of all regular n by n matrices of reals, with matrix multiplications forms a group; the neutral element is the identity matrix.

Counting Finite Groups



Why Bother?

Why bother with abstract algebra? Why not simply continue with the kind of basic arithmetic we've done so far?

Laziness: any property, of, say, groups derived from only the axioms automatically holds in all groups. You only check three simple properties, and all results apply.

Psychology: it is sometimes easier to argue abstractly than in a concrete situation (you can't see the forest because of all the trees).

Abstract is Easy

The second point may be hard to swallow, but it's true.

E.g., consider non-singular matrices of reals. We have

$$(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$$

One could try to use the properties of matrix multiplication and, say, Gaussian elimination, to prove this. This would be very difficult at best.

Claim

In any group, $(xy)^{-1} = y^{-1}x^{-1}$.

Proof. By the last claim it suffices to show that $y^{-1}x^{-1}$ behaves like an inverse. □

Subgroups

Let $\langle G, *, 1 \rangle$ be a group.

Definition

$H \subseteq G$ is a subgroup of G if $\langle H, *, 1 \rangle$ is a group.

Strictly speaking, the second $*$ should be the restriction to H–no one ever bothers to distinguish between the two.

Lemma

Let G be a group and $\emptyset \neq H \subseteq G$.

Then H is a subgroup of G if, and only if, $x, y \in H$ implies $x^{-1} * y \in H$.

If the group is finite than it suffices that $x, y \in H$ implies $x * y \in H$.

Cyclic Groups

Definition

Let G be a group and $a \in G$. The subgroup generated by a is

$$\langle a \rangle = \{ a^i \mid i \in \mathbb{Z} \} \subseteq G.$$

The order of a is the order of $\langle a \rangle$.

A group G is cyclic if there is some element $a \in G$ such that $\langle a \rangle = G$. In this case a is a generator for G.

If G is a finite cyclic group we have

$$G = \{ a^i \mid 0 \leq i < \mathsf{ord}(a) \}$$

In other words, there is an element whose order is the same as the order of G.

Isomorphisms

Definition

Suppose $G = \langle G, *, 1_G \rangle$ and $H = \langle H, \cdot, 1_H \rangle$ are groups. A function $f : G \to H$ is a (group) homomorphism if

$$f(x * y) = f(x) \cdot f(y)$$

If the function f is in addition bijective then it is an isomorphism.

Up to isomorphism there is only one cyclic group of order k, and it is isomorphic to $\langle \mathbb{Z}_k, +, 0 \rangle$. A generator is 1.

Note that there are other generators, though: ℓ is a generator iff $\gcd(\ell, k) = 1$.

All cyclic groups are commutative: $a^i * a^j = a^{i+j} = a^j * a^i$.

Logarithms

Here is the classic historical example of such a map:

$$\log : \mathbb{R}^+ \to \mathbb{R}$$

which translates multiplication into addition (more precisely: from $\langle \mathbb{R}^+, \cdot, 1 \rangle$ to $\langle \mathbb{R}, +, 0 \rangle$).

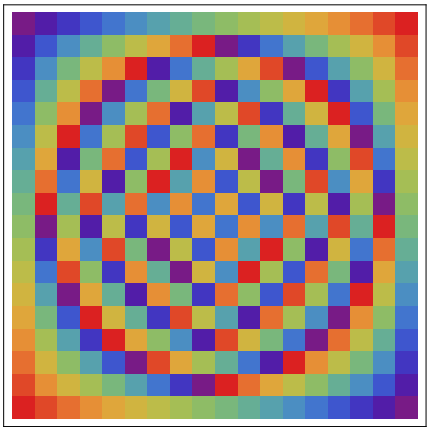
We can compute products of (positive) reals by

$$x \cdot y = e^{\ln x + \ln y}$$

Makes a huge difference: $O(k)$ plus table lookup rather than $O(k^2)$ where k is the number of decimal digits.

Of course, we have to compute a logarithm table first–but only once. It took John Napier some 20 years to construct such a table in the early 1600s.

Example $\mathbb{Z}_{19}^*$



This group is cyclic: generators are $2, 3, 10, 13, 14, 15$.

Counterexample $\mathbb{Z}_{20}^*$

Here is the Cayley table for $\mathbb{Z}_{20}^*$.

1	3	7	9	11	13	17	19
3	9	1	7	13	19	11	17
7	1	9	3	17	11	19	13
9	7	3	1	19	17	13	11
11	13	17	19	1	3	7	9
13	19	11	17	3	9	1	7
17	11	19	13	7	1	9	3
19	17	13	11	9	7	3	1

Note the subgroup $\{1, 3, 7, 9\}$ in the top-left corner.

No element generates a subgroup of size larger than 4.

Subgroups of $\mathbb{Z}_{19}^*$

Clearly $|\mathbb{Z}_{19}^*| = \varphi(19) = 18$.

Here are all the subgroups and their generators:

1	$\{1\}$
18	$\{1, 18\}$
$7, 11$	$\{1, 7, 11\}$
$8, 12$	$\{1, 7, 8, 11, 12, 18\}$
$4, 5, 6, 9, 16, 17$	$\{1, 4, 5, 6, 7, 9, 11, 16, 17\}$
$2, 3, 10, 13, 14, 15$	$\mathbb{Z}_{19}^*$

The order of all group elements:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	18	18	9	9	9	3	6	9	18	3	6	18	18	18	9	9	2

- Digression: A Binomial Problem
- From Modular Arithmetic to Algebra
3. The Fibonacci Monoid

Fibonacci Numbers

The simplest second-order recurrence is the famous Fibonacci recurrence (homogeneous):

$$\begin{aligned} F_0 &= 0, \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

The obvious approach to computing Fibonacci numbers is linear in n (assuming the arithmetic is constant time, which it really isn't).

Is there are faster way? Something sublinear?

A Fast Recurrence

$$F_{2n} = F_n^2 + 2F_nF_{n-1}$$

$$F_{2n+1} = F_n^2 + F_{n+1}^2.$$

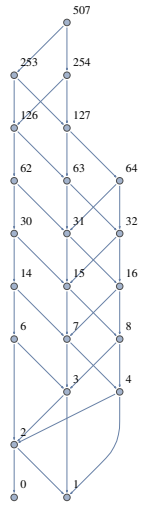
Note that each call results in 2 subcalls.

Still, with memoization one can evaluate this recurrence in $O(\log n)$ calls.

Call Tree

This is the call DAG for $n = 507$.

Can you can read off a proof that the number of calls is $O(\log n)$?



A Wild Structure

The last approach to computing F_n has a distinctly combinatorial flavor. Might algebra also help?

Definition (Fibonacci Algebra)

Define the Fibonacci product $*$ on pairs of natural numbers by

$$(x, y) * (x', y') = (xx' + yy', yy' + xy' + x'y)$$

It is straightforward to check:

Claim

The Fibonacci product on $\mathbb{N} \times \mathbb{N}$ is associative and has neutral element $(1, 0)$.

Who Cares?

The key fact that makes this algebra useful for us is this:

$$(x, y) * (0, 1) = (y, x + y)$$

A straightforward induction shows that

Claim

In the Fibonacci algebra, $(0, 1)^n = (F_{n-1}, F_n)$.

Example

Here are some powers of $(0,1)$ in the Fibonacci monoid:

k	$(0,1)^{2^k}$	F_{2^k}
0	$(0,1)$	1
1	$(1,1)$	1
2	$(2,3)$	3
3	$(13,21)$	21
4	$(610,987)$	987
5	$(1346269,2178309)$	2178309

So we can compute F_{32} in just 5 Fibonacci operations.

Of course, these operations involve several multiplications and additions of naturals.

Computing F_{1022}

So we can compute F_{1024} and F_{1023} very quickly: 11 Fibonacci multiplications suffice.

But how about F_{1022}?

If we use the standard algorithm we need 19 operations.

Wild Idea: Perhaps we can work backwards from F_{1024} as in

$$F_{1022} = (0,1)^{2^{10}} * (0,1)^{-2}$$

using just 13 operations?

Well, for this we need a group, not just some arbitrary associative multiplication.

Any chance the Fibonacci algebra might be a group?

Towards an Inverse

Given x and y we have to solve

$$(xx' + yy', yy' + xy' + x'y) = (1,0)$$

for x' and y'. No problem:

$$x' = \frac{x+y}{x^2+xy-y^2}$$
$$y' = \frac{-y}{x^2+xy-y^2}$$

. . . except that we are now dealing with rationals rather than naturals: we need $\mathbb{Q} \times \mathbb{Q}$ as carrier set.

The Fibonacci Group

There is another little glitch: $(0,0)$ has no inverse since $(0,0)$ is a multiplicative annihilator:

$$(0,0) * (x',y') = (0,0)$$

The good news is that this is the only problem: the denominators x^2+xy-y^2 have no other integral roots since the solution in reals is

$$y = \frac{x}{2}\left(1 \pm \sqrt{5}\right)$$

So we have a group $\mathcal{G}$ with carrier set $\mathbb{Q} \times \mathbb{Q} - \{(0,0)\}$.

The Right Subgroup

In fact, since we are only interested in computing Fibonacci numbers and since the inverse of $(0,1)$ is integral:

$$(0,1)^{-1} = (-1,1)$$

we can just work in the subgroup $\mathcal{F}$ generated by $(0,1)$.

All the elements in $\mathcal{F}$ are integral, no problem.

The question arises how many group operations are needed to compute F_n.

Straight Line Programs

We can describe a computation like the one above for F_{1022} as a straight line program (SLP), a sequence of instructions

$$v_k = v_i * v_j$$
$$v_k = v_i / v_j = v_i * v_j^{-1}$$

where k is the line number and $i,j < k$. We always start with

$$v_0 = (0,1)$$

The result is the second component of the value of the variable v_k in the last instruction.

We call that k the length of the SLP. Optimal then simply means: minimal length.

So, we want an algorithm that, on input n, determines the optimal SLP with output F_n. This turns out to be difficult even for fairly small values of n.

Examples

Here is a SLP for F_{31} using only multiplication (no division).

$$\begin{array}{ll} v_0 = (0,1) & F_1 \\ v_1 = v_0 * v_0 & F_2 \\ v_2 = v_1 * v_1 & F_4 \\ v_3 = v_2 * v_2 & F_8 \\ v_4 = v_3 * v_3 & F_{16} \\ v_5 = v_4 * v_3 & F_{24} \\ v_6 = v_5 * v_2 & F_{28} \\ v_7 = v_6 * v_1 & F_{30} \\ v_8 = v_7 * v_0 & F_{31} \end{array}$$

So length 8 suffices.

F_{31}

A length 6 SLP for F_{31} that uses division.

$$\begin{array}{ll} v_0 = (0,1) & F_1 \\ v_1 = v_0 * v_0 & F_2 \\ v_2 = v_1 * v_1 & F_4 \\ v_3 = v_2 * v_2 & F_8 \\ v_4 = v_3 * v_3 & F_{16} \\ v_5 = v_4 * v_4 & F_{32} \\ v_6 = v_5 / v_0 & F_{31} \end{array}$$

Is length 6 perhaps optimal?

F_{31}

Yes, but that's not so easy to show.

Nor is the solution unique.

$$\begin{array}{ll} v_0 = (0,1) & F_1 \\ v_1 = v_0 * v_0 & F_2 \\ v_2 = v_1 * v_1 & F_4 \\ v_3 = v_2 * v_2 & F_8 \\ v_4 = v_3 * v_3 & F_{16} \\ v_5 = v_4 / v_0 & F_{15} \\ v_6 = v_5 * v_4 & F_{31} \end{array}$$

Exercise

Show that length 6 is optimal for F_{31}: there is no shorter SLP that computes F_{31}.

A Little Help

Looking at these examples, notice that we only worry about the n in F_n. We could have written

$$\begin{array}{ll} v_0 = 1 & 1 \\ v_1 = v_0 + v_0 & 2 \\ v_2 = v_1 + v_1 & 4 \\ v_3 = v_2 + v_2 & 8 \\ v_4 = v_3 + v_3 & 16 \\ v_5 = v_4 - v_0 & 15 \\ v_6 = v_5 + v_4 & 31 \end{array}$$

The same simplification works for all our SLPs.

What's going on?

First note that in order to execute an SLP we need two things:

- an arbitrary group G, and
- a special element $a \in G$.

We can then initialize $v_0 = a$ in line 0, and run through the other instructions interpreting multiplication and division in the group G.

The same program can be run over any group G, and with any starting value $a \in G$.

So, given an SLP P, a group G, and $a \in G$, we can define

$$P(G, a) \in G$$

as the result of executing P over G with initial value a.

The Infinite Cyclic Group

Our computations use the Fibonacci group $\mathcal{F}$ and $a = (01,)$ (plus projection on the second component in the end).

Now consider the two groups

$$\mathbb{Z} = \langle \mathbb{Z}, +, 0 \rangle$$
$$\mathcal{F} = \langle F, *, (1,0) \rangle$$

These two groups look very different, but they are isomorphic via $f : \mathbb{Z} \to \mathcal{F}$:

$$f(n) = (0,1)^n = (F_{n-1}, F_n)$$

So, in a sense, instead of computing $P(\mathcal{F}, (0,1))$ we can just as well compute $P(\mathbb{Z}, 1)$:

$$f(P(\mathbb{Z}, 1)) = P(\mathcal{F}, (0,1))$$

Representation is Crucial

This is a perfect example where an isomorphism makes life so much easier.

Everybody understands $\langle \mathbb{Z}, +, 0 \rangle$ very well intuitively.
But $\langle F, *, (1,0) \rangle$ is a bit mysterious.

It doesn't matter, they are isomorphic, so we can argue in either one.

Efficiency?

We need to find short SLPs for our computation. In the literature, these are referred to as addition chains and addition-subtraction chains since one thinks about $\mathbb{Z}$.

Unfortunately, finding minimal addition chains is $\mathbb{NP}$-hard.

The problem is open for addition-subtraction chains but clearly not easy.

Incidentally, this is important for elliptic curve cryptography where the group operation involves an elliptic curve and is quite expensive computationally.

Heuristic: Non-Adjacent Form

Base $B = 2$, digit set $D = \{-1, 0, 1\}$.

Claim

For every natural number n there are digits $d_i \in D$ such that $n = \sum d_i 2^i$ and $d_i d_{i+1} = 0$.

```
i = 0;
while( n > 0 )
{
     if( n odd )
         d[i] = 2 - (n mod 4);
     else
         d[i] = 0;
     n = (n - d[i])/2;
     i++;
}
```

Examples

n	binary	NAF
8191	$(1,1,1,1,1,1,1,1,1,1,1,1,1)$	$(1,0,0,0,0,0,0,0,0,0,0,0,0,-1)$
8182	$(1,1,1,1,1,1,1,1,1,1,0,1,1,0)$	$(1,0,0,0,0,0,0,0,0,0,-1,0,-1,0)$
12345	$(1,1,0,0,0,0,0,0,1,1,1,0,0,1)$	$(1,0,-1,0,0,0,0,0,1,0,0,-1,0,0,1)$

Exercise

Show that the conversion algorithm is correct.

OEIS

Unsurprisingly, there is a sequence on OEIS for addition-subtraction chains:

http://oeis.org/A128998

Summary