THE MATHILY ROM

THE RECORD OF MATHEMATICS

Issue Three
July 28, 2013
Authors: Andy, Ben, Cecilia, Crystal, Emi, Ethan, Gideon, Ina, Josh, Joshua, Lydia, Riley, William
Editors: Joshua, Emi

Contents

1 Class Summaries **1**
1.1 Objects of Algebra: Josh 1
1.2 Loops on Loops on Loops: Gideon 2
1.3 A Cure for the Common Absolute Value: Ethan 3
1.4 Generatingfunctionology: William 3
1.5 Class of Chaos: Ina 4
1.6 Random Walks on Money: Andy . 6
1.7 Cryptography: Lydia 7
1.8 Functional Functional Programming: Crystal 7
1.9 Problems of Tuvalu: Riley 8
1.10 Lebesgue Integration: Emi 9
1.11 Purely Theoretical Computers: Joshua 10

2 Daily Gathers **11**
2.1 Kidney Transplant Mathematics: Ben 11
2.2 Poker Combinatorics: Josh 12

3 Fun Stuff **13**
3.1 Detective Josh and the Case of the Unsinkable Ship, Part 3 13

1 Class Summaries

1.1 Objects of Algebra: Josh

On Monday, we began by defining the three types of algebraic objects we would be dealing with over the course of the week. We reviewed the Snarfu, which is simply a set with a binary operation that is associative, has an identity, and has at least one inverse for each element in the set. We defined a new algebraic object: a sniglar. A sniglar is a set with two binary operations (denoted $+$ and $\cdot$) where the set is a commutative snarfu under $+$, and $\cdot$ is associative and distributes over $+$. Using this, we gave a shorter definition of a Magicloc: a set M with a sniglar of scalars S such that M is a snarfu under $+$, $\cdot$ distributes over $+$, and $s_1 \cdot s_2(\mathbf{m}) = s_1(s_2\mathbf{m})$.

After this, we proved a few theorems on these objects, such as: if $\varphi : A_1 \to A_2$ is a homomorphism, then $ker(\varphi)$ is a subobject of A_1. We also defined a generating set: a set of objects in an *Algebraic Object* such that you can obtain every element in the set by applying the operations of the *Algebraic Object* to the generators and their inverses repeatedly.

Tuesday

On Tuesday, we began by proving some small results, such as: a subset of a generating set for an object A generates a subobject of A. We were asked a few more questions by sarah-marie, for example: can *any* object be the kernel of a homomorphism? We also reviewed a few old questions, such as: can any snarfu be reduced to being expressed as a set of relations and a generating set? (This is known as a presentation) Gideon also wanted to know if every finite snarfu had a presentation that allowed writing any element of a snarfu as a product of generators in order. Many such questions were asked.

Wednesday

On Wednesday, Emi began by conjecturing that given a finite snarfu G and a subsnarfu SG, $|SG|$ divides $|G|$. We proved this conjecture by showing that such a snarfu could be expressed as a union of multiples of the subsnarfu, and such multiples were either disjoint or equal. This result is known as Lagrange's Theorem for finite snarfus. Gideon extended this claim to infinite snarfus: Given an infinite snarfu G and a subsnarfu SG, for each $a \in SG$, there exists n corresponding elements in G for some n. We also showed that the size of a minimal generating set for a finite snarfu with n objects must be less than or equal to $\frac{n}{2}$.

In addition, Josh was observed to have said "I don't know how my hands have survived this long" after hitting them repeatedly against the chalk shelf.

Thursday

On Thursday, Gideon began by conjecturing that any subobject of a commutative snarfu could be the kernel of a homomorphism. In a commutative snarfu, we begin by partitioning the generating set into a generator for the subobject and a generator for a second subobject disjoint except for the identity (This was not proven). We then define our homomorphism as sending generators of the first subobject to the identity, and sending generators of the second object to themselves. Because the snarfu is commutative, this is indeed a homomorphism.

Josh conjectured that any submagicloc could be the kernel of a homomorphism. He proposed an algorithm of first constructing a basis for the submagicloc SB, and then constructing a basis B

for the magicloc that contained SB. Then one can define a homomorphism $\varphi : M \to M$ that sends anything in SB to $\mathbf{0}$ and sends everything else in B to itself. This homomorphism is well-defined for arbitrary $\mathbf{m}$ since basis representations are unique.

We also proved Lagrange's Theorem for finite sniglars (essentially the same proof) and sarah-marie gave us a tool to extend it to infinite objects as well. Unfortunately, time ran out before proving Lagrange's Theorem for infinite snarfus.

Friday

On Friday, Gideon found a counterexample to the claim that any subobject could be the kernel of a homomorphism. Consider the snarfu D_3 and the subsnarfu $\{e, b\}$. Since $a^2b = ba$, $\varphi(a^2b) = \varphi(ba)$ and so $a^2 = a$, meaning that $a = e$. Thus, $\{e, b\}$ is not the kernel. Emi also proved Lagrange's Theorem for infinite snarfus: Given a subsnarfu of an infinite snarfu, there is a natural (or infinite) number of elements in the snarfu for each element in the subsnarfu. This is denoted $[G : SG] = n$.

In addition, Josh proved it was possible to construct a basis for an object that contained a basis for an arbitrary subobject, thus proving his conjecture from Thursday.

1.2 Loops on Loops on Loops: Gideon

The class "Loops on Loops on Loops" began with Emil's departure, and the end of "Functional Functional Programming." Max used the analogy of paths between buildings on a college campus, such as Bryn Mawr, to introduce the concept of maxotopy, which describes paths that can be "dragged" into each other. We established rules for maxotopy, namely that maxotopic paths must start and end at the same points, and that the paths cannot be dragged through buildings. Then we considered loops, the main focus of the class, making distinctions between the directions of loops and noting that the building at which the loop begins is irrelevant. We eventually proved that the set of loops for any campus is a "Snuper Important" snarfu (denoted SnIS) under loop concatenation. In order to demonstrate certain properties of maxotopy, we created a specific college campus, "No Horse College," consisting only of two buildings. The loops we considered here were given maxotopy numbers, describing the number of times they passed a line around one of the buildings, based on direction. We then showed that equivalent maxotopy numbers implies maxotopy between two loops. Based on this, we conjectured that the SnIS of maxotopy classes (sets of maxotopic loops) of "No Horse College" was isomorphic to the integers. We questioned whether all SnIS's might be commutative, but there was a simple counterexample. It was noted that the orientation of buildings on the campus is also irrelevant, because they can be "dragged" in the same manner as loops. We called a space "Cecilian" if its area is connected such that any point can be reached from any other point. We then turned to SnIS's in different contexts, such as a triangular graph, the SnIS of which is identical to that of "No Horse Campus." Josh asked whether any space (campus) might be represented as a graph, and Brandon conjectured that any graph could be redrawn as a campus by transforming each face into a building. It was suggested that "No Horse College" could be represented as a loop from a point to itself, which Max extended to an infinite "slinky" of loops, with the base loop as a projection of the slinky onto the space. We then established that after traveling up and down the slinky, the number of loops traversed was equivalent to the maxotopy number of the projected loop. We then attempted to demonstrate that the the set of classes of equal-length paths along the slinky was isomorphic to the set of maxotopy classes on the projection. Once this was complete, Max proved Brouwer's Fixed Point Theorem, namely that for any continuous mapping of a space onto itself, there exists a point unchanged by the mapping.

Max finally demonstrated that there exists a finite nontrivial SnIS, that the SnIS of a Sphere is the identity, and that the SnIS of a torus is isomorphic to $\mathbb{Z} \times \mathbb{Z}$.

1.3 A Cure for the Common Absolute Value: Ethan

The equation $Vp(x)$ gives the exponent of the prime p when it is a factor of x. A p-adic number $|X|p = p^{-Vp(x)}$. P-adic numbers are equivalent to absolute values in that they have the same properties. They map from the rationals to the positive real numbers, $|x| = 0$ only if $x = 0, |ab| = |a|*|b|, |-a| = |a|$ and $|a+b| \leq |a|+|b|$ In order to find a p-adic equation that converged to 32, we set Xn equal to $32 + 2^n$. Because $|Xn - 32|2 = |2^n|2$ which converges to 0 as n increases, $|Xn|2$ converges to 32 as n increases. Balls are sets defined as $Br(a) = xinQs.t.|x - a|p < r$ Closed balls use a $\leq$ rather than a $<$. We proved that if a ball of radius r shares a point with a ball of radius r, then they contain exactly the same set of points.

1.4 Generatingfunctionology: William

In this class, we learned about generating functions. This is a method where we create a polynomial in order to solve counting problems without thinking or to find explicit formulas for recursive definitions.

For example, consider the following counting problem: At a body part convention, we want 15 parts (torsos, arms, or heads) in total, but we need at least 2 of each, an even number of arms, and no more than 5 torsos. We create a separate polynomial for each of the three body parts and multiply them together: $(x^2 + x^3 + x^4 + x^5)(x^2 + x^3 + x^4 + ...)(x^2 + x^4 + x^6 + x^8 + ...)$. Notice that the exponents in the first polynomial corresponds to the number of possible torsos we can have, the exponents in the second corresponds to the number of heads, and the exponents in the third corresponds to the number of arms. If we multiply out the polynomial, the coefficient of the x^{15} term corresponds to the number of different combinations of torsos, heads, and arms for a total of 15 body parts. This method works because every time we generate an x^{15} term, we are really adding exponents from each of the 3 polynomials, each of which represents the number of a distinct body part.

If instead we were to find an explicit formula for $a_{n+1} = 2a_n + 1$, where $a_1 = 0$, then we can let $G(x) = \sum_{n=1}^{\infty} a_n x^n$. In this sum, each a_n is the coefficient of its corresponding x_n. Hence, it suffices to find this coefficient. By our recursive definition, we can create the equation $\sum_{n=1}^{\infty} a_{n+1} x^n = \sum_{n=1}^{\infty} 2a_n x^n + \sum_{n=1}^{\infty} x^n$. We can substitute and solve this equation in terms of $G(x)$ by using partial fraction decomposition and the power series $\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$. In the end, we get $a_n = 2^{n-1} - 1$.

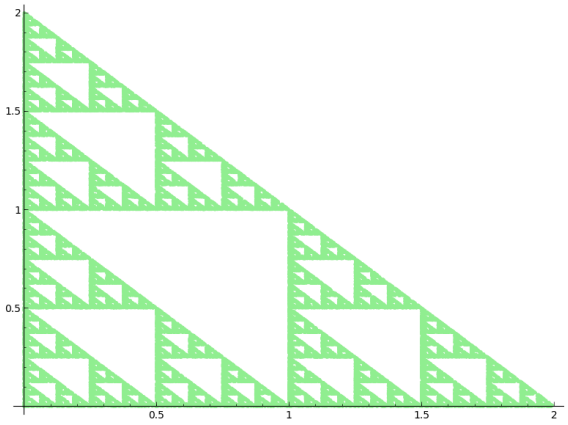
A technique used to simplify combinatorial sums is the Snake Oil Method. In this method, we first identify the free (independent) variable, say n. Next, we multiply our sum by x^n and sum over n to get a generating function expressed as a double sum. Then, we move the inner sum out and simplify the inner part. Finally, we identify the coefficient of x^n. For example, we can

use the Snaks Oil Method to find an explicit formula for $\sum_{k=0}^{\infty} \binom{k}{n-k}$ for any nonnegative integer n: $\sum_{n=0}^{\infty} x^n \sum_{k=0}^{\infty} \binom{k}{n-k} = \sum_{k=0}^{\infty}\sum_{n=0}^{\infty} \binom{k}{n-k} x^n = \sum_{k=0}^{\infty} x^k \sum_{n=0}^{\infty} \binom{k}{n-k} x^{n-k} = \sum_{k=0}^{\infty} x^k \sum_{r=-k}^{\infty} \binom{k}{r} x^r = \sum_{k=0}^{\infty} x^k (1+x)^k$, and we can simplify from here by using former methods.

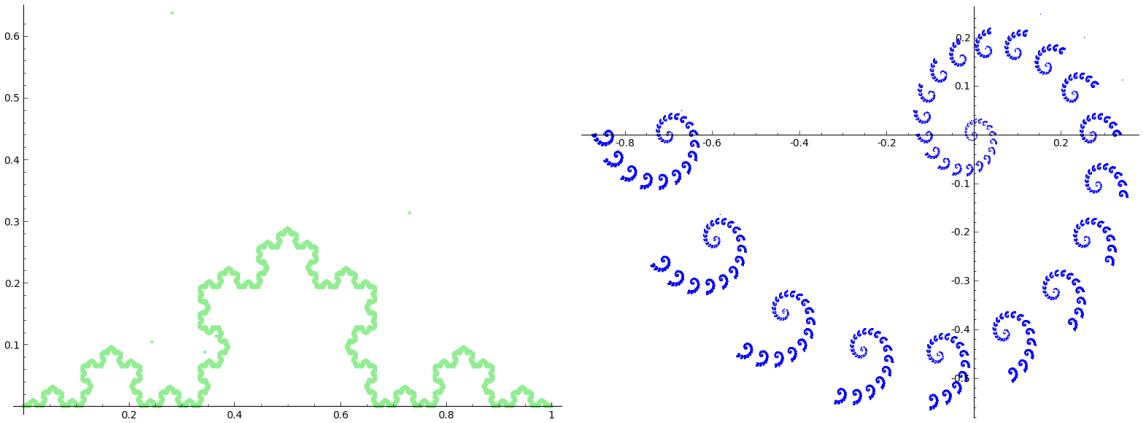
All of the above examples can be categorized as ordinary power series generating functions (opsgfs). This technique is good for solving counting problems where order does not matter. If instead we want to solve problems where order does matter, we can use exponential generating functions (egfs). Instead of creating a function similar to $G(x) = \sum_{n=0}^{\infty} a_n x^n$ and finding the coefficient to x^k for some k, we instead create a function similar to $G(x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$ and find the coefficient to $\frac{x^k}{k!}$ for some k. Moreover, we can use the power series $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ to help simplify. For example, if we want to find the number of distinct ways to make four letter "words" by using the letters a, b, c, with at least 2 a's, then we can set up the polynomial $(\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+...)(1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+...)^2$ (the first term represents the number of a's, and the second term raised to the second power represents the number of b's and c's) and find the coefficient of $\frac{x^4}{4!}$. The reason why this method works is because since order matters, we have to count each case a total of 4! times, but we are over counting by a factor of $k!$ if we have k a's (similarly for b's and c's). Another application of exponential generating function (by using the e^x power series) is to find a closed formula of a sum with $n!$ in the denominator, such as $\sum_{n=0}^{\infty} \frac{n^2+4n+5}{n!}$.

1.5 Class of Chaos: Ina

During the first few days of The Class of Chaos and Fractals we learned how to program fractals into sage. To make fractals, we used the chaos algorithm. The chaos algorithm starts with a random point and uses midpoints and repetition to create fractals. On Monday we graphed the Sierpinski triangle:



On Tuesday and Wednesday, we used rotations and translations to turn the algorithm we used for the Sierpinski triangle into algorithms that created The Koch Snowflake Curve, and Swirls!



We also investigated the Logistic Function f(x)=L(1-x). This equation models the growth of animal populations. We iterated the function on sage, testing different values of L and x. For some values of L the results of the function eventually hit a cycle, while for others the result was chaos (there was no pattern). We created a graph on sage to model iterations of the function: