

PROJECT PROPOSALS: MATHCAMP 2006

THE 290 THEOREM

Supervised by: David Roe.

Description: In “The Fifteen Theorem”, I covered the proof of the fact that any integer-valued, positive definite quadratic form that represents nine specific numbers between 1 and 15 in fact represents all integers. A more difficult theorem, “The 290 Theorem”, states that any integer-valued, positive definite quadratic form that represents twenty-nine specific numbers between 1 and 290 in fact represents all integers. Proved only last year, the proof involves modular forms as well as the techniques used on the 15 Theorem. In this project, you’ll read and try to understand the proof of the 290 Theorem.

Expected project output: Greater knowledge. Perhaps a writeup or poster about what you’ve learned.

Difficulty level: ****

Mathematical prerequisites: Knowledge of groups, rings and fields. Elementary number theory. Complex analysis.

Math you’ll get to learn along the way: Modular forms, the 290 Theorem.

Computer use: To look up papers online. No programming.

Preferred number of project participants: 1–3.

Degree of supervision: Medium: I’m happy to help answer questions, talk to you, and guide you toward references, but I won’t be teaching you everything you need to know.

CIRCUITS AND THE NATURAL NUMBERS

Supervised by: Dan.

Description: A circuit is a computational object that takes inputs and passes them through *gates* that perform operations on the inputs. For example, many circuits in common use take in boolean values (true or false) and put them through AND gates, OR gates, and NOT gates. The circuit outputs a boolean value that is a result of passing the inputs through the gates. Computation can actually be defined with circuits instead of Turing machines.

Now suppose a circuit takes as input sets of natural numbers, and has operations

$$\cup, \cap, \text{ complement}, +, \times.$$

This kind of circuit will output a set of natural numbers. Here's a question: is the outputted set decidable by some algorithm, given the circuit C and input to the circuit w ?

This problem is still open, although there are many intermediate results. In particular, simpler versions of this problem are (variously) NP-complete, PSPACE-complete, and NEXPTIME-complete (nondeterministic exponential time complete). We'll learn about attempts to solve this problem and then attempt to solve it ourselves.

By the way, if this turns out to be decidable, then you can use this to decide Golbach's conjecture! I personally suspect the problem isn't decidable, but if it turns out to be decidable and an algorithm appears, we can run a Turing machine/computer that will output the solution to Goldbach's conjecture!

Expected project output: Maybe a theorem.

Difficulty level: Hard.

Mathematical prerequisites: Theoretical CS is highly recommended to understand the paper. Number theorists are also welcome, but it may be hard to deal with all the surrounding computer science.

Math you'll get to learn along the way: Lots of stuff about complexity theory.

Computer use: None.

Preferred number of project participants: 3.

Degree of supervision: A fair amount, but you'll still need a good work ethic to make progress!

CROSSING NUMBERS

Supervised by: Marisa.

Description: Interested in working on an open problem? One of my favorite things about graph theory is that neat problems are often simple to pose. Here's one: our goal is to draw the complete graph K_n with as few edge crossings as possible. What is the minimum crossing number? Surprisingly, that question hasn't been satisfactorily answered yet. Graph theorists are interested in knowing the answers to that and lots of related questions, and we'd be quite grateful if you answered some for us!

Expected project output: You'll create a survey of the state of the art in crossing numbers and try applying some of the proof techniques you learn to unexplored families of graphs. You might even have a publishable research paper with shiny new results!

Difficulty level: Learning about the existing results will be accessible if you put in the time. Making progress on open questions will be pretty tough, but rewarding.

Mathematical prerequisites: Basic graph theory, including Kuratowski's Theorem (we'll do this at the beginning of Topological Graph Theory, so that could be a co-requisite).

Math you'll get to learn along the way: Modern proof techniques in graph theory.

Computer use: You'll be looking up papers online and finding them in the library, but we won't be doing math on computers.

Preferred number of project participants: 1–5.

Degree of supervision: I'll give you quick primer and then I'll point you to materials to read, you'll start reading and come back with questions, we'll discuss. This is a reasonably self-motivated project: you'll need to spend some time in the library and be committed to making progress. Equally appropriate for a small group of students working together.

DIRICHLET'S THEOREM ON ARITHMETIC PROGRESSIONS

Supervised by: Miljan.

Description: In number theory, Dirichlet's theorem states that for any two positive coprime integers a and d , there are infinitely many primes of the form $a + nd$, where $n > 0$, or in other words: there are infinitely many primes which are congruent to a modulo d .

Euler stated that every arithmetic progression beginning with 1 contains an infinite number of primes. The theorem in the above form was first conjectured by Gauss and proved by Dirichlet in 1837 with Dirichlet L-series. The proof is modeled on Euler's earlier work relating the Riemann zeta function to the distribution of primes. The theorem represents the beginning of rigorous analytic number theory.

Expected project output: Mastering techniques of the proof, understanding underlying ideas and concepts, writing a survey and giving a talk.

Difficulty level: ****

Mathematical prerequisites: Elementary number theory, basic algebra and basic complex analysis

Math you'll get to learn along the way: Riemann's Zeta Function, Dirichlet's characters and L-functions, Gauss sums, a lot of analytic number theory, ...

Computer use: No

Preferred number of project participants: 2.

Degree of supervision: Miljan will be available for help, but the most of work is expected to be done by you!

THE HAPPY END THEOREM

Supervised by: M@.

Description: The Happy End Theorem is a beautiful result of Erdős and Szekeres that says that there exists a function f with the following property. Given any $f(n)$ points in the plane in general position, i.e. with no 3 on a line, there exists some subset of n of them that form the vertices of a convex n -gon. (A historical note: In proving this, they actually rediscovered Ramsey theory.) Let $g(n)$ be the minimum possible value for $f(n)$ so that the above holds. It was conjectured in 1935 that $g(n) = 2^{n-2} + 1$ but over 70 years later this is only known to be true for $n \leq 5$. Now this conjecture's reputation has grown notorious.

In this project, you will read a survey article on this theorem and attacks people have made on the conjecture. The article I have in mind also discusses various generalizations, including higher dimensional analogues.

Ideally you will try yourself to prove something new here, just using the article as reference. Although it is unlikely to prove the notorious conjecture, there are many possible directions you could go, and I want you to get a small taste of math research. For example, can you improve the bounds for $n = 6$? Not much seems written here. I think it would be a publishable result if you could give a short proof that 20 points in general position force a convex hexagon. (Probably this is true with 17.)

If nothing else, you will learn some cool math and get a feeling for the history of this problem. But I'm most interested in working with participants who will try things out for themselves, obsessively drawing pictures and thinking about this through meals and in their sleep. Okay, maybe that's a bit much, but this really should be an open ended creative project for self-motivated participants.

Expected project output: A new theorem optimistically. Otherwise a poster summarizing the survey article for the Project Fair.

Difficulty level: Proving the full conjecture? Probably totally impossible. Making a cool poster? Easy.

Mathematical prerequisites: None.

Math you'll get to learn along the way: A little convex geometry, graph theory, and Ramsey theory.

Computer use: If participants feel so inclined, but I will warn you ahead of time that computers aren't as useful here as might appear at first glance.

Preferred number of project participants: 1–5.

Degree of supervision: Meeting once or twice a week. I will help explain any math in the survey article, or else you can tell me what problems you've been trying to solve, and how you're attacking them.

KNIGHT'S TOURS

Description: The diagram shows a “closed knight’s tour” on a 5×8 “chessboard”.

A knight moving (as usual in chess) as indicated will visit every square exactly once and return to its starting square. The goal of the project is to discover, and prove, for exactly which values of m and n a knight’s tour exists on an $m \times n$ “chessboard”. This can be investigated both for closed and for “open” tours (which do not return to the starting square).

Supervised by: Mark.

Expected output: Poster, probably.

Difficulty level: **–***

Math prerequisites: Induction (and patience).

Math you’ll get to learn: Practice in constructing a somewhat complicated argument.

Computer use: Probably none.

Preferred number of participants: 2 or 3.

Degree of supervision: Light to moderate.

KOLMOGOROV COMPLEXITY

Supervised by: Dan.

Description: Can one finite string be more random than another? You might say no, but here's a description of randomness you might not be expecting: strings get more "random" as the size of the smallest Turing machine that outputs them gets larger. (So a string of all zeros is not very random, since you can just loop and output a zero at each point in the loop; the digits of π are slightly more random; really complicated strings get much more random). We can also extend this idea to infinite-length strings and ask if a Turing machine can predict the next character given that it knows the preceding few characters.

Kolmogorov complexity has applications in computer science, but also physics and entropy. With some work, we can definitely get to these topics, so there's a lot of great stuff we can do; the topics we explore will be up to you guys.

This will be a "reading course" on Kolmogorov complexity. I won't be teaching for the class: instead, you'll all read sections of a book or articles and then present and discuss those sections. That means it'll be a bit of work to keep up, and you'll need to really make sure you do the readings. On the other hand, you'll learn a lot about reading math on your own and presenting it, in addition to the cool math itself.

Expected project output: A lot of learning, maybe a poster or some notes.

Difficulty level: Medium.

Mathematical prerequisites: The definition of a Turing machine.

Math you'll get to learn along the way: Kolmogorov complexity.

Computer use: None.

Preferred number of project participants: 3–5.

Degree of supervision: Moderate. I actually don't know much about this stuff, so I want to learn, too!

MAGIC CONFIGURATIONS

Supervised by: Ellen

Description: A few weeks ago, I was at a discrete and computational geometry conference, and this problem was stated as an open problem. It was the topic of much discussion, because it's a beautiful problem and it is easy to state. The problem is as follows: we have m points x_1, x_2, \dots, x_m in \mathbb{R}^2 . Each point is labeled with a number $n(x_i)$ so that along any line L in \mathbb{R}^2 with at least two of our points on it, the sum of $n(x_i)$ over x_i on L is constant.

There are 4 different known families of such arrangements, which are known as “magic configurations” (can you find all 4?). The question is: are there any more? Your goal would be to either find another configuration or prove that there are no more.

Expected project output: A publishable paper, if you can either find another arrangement or prove that there are no more.

Difficulty level: my guess: very hard.

Mathematical prerequisites: I don't know. My guess: metric geometry, combinatorics, ... maybe unexpected things will be helpful.

Math you'll get to learn along the way: Anything you decide you need to know to help you solve the problem.

Computer use: I'm guessing none, but you never know.

Preferred number of participants: 1–3; if there are several, you probably should be willing to work together.

Degree of supervision: quite small.

MATHEMATICS OF SET

Supervised by: M@.

Description: We will investigate some math behind the matching card game SET. Many of you already know how to play this, and you already had some vague intuition that there was some math behind it. We aim to make this precise.

It turns out that what you're really doing when you play set is finding lines in 4-dimensional (affine) space over the finite field of 3 elements. I will explain what this means, and then you will use this knowledge to attack one or more of the following questions:

- (1) Why is 20 the maximum number of cards not containing a set? There is only a totally lame pseudoproof on the SET webpage, so I'm sure we can do better. This would involve two things: giving an example of a set of 20 cards with no set, and proving that 21 cards always contain a set. I don't know myself how to do the second part yet, but I expect that it will yield much faster to cool mathematics than to a brute force computer search.
- (2) If we assume that sets get grabbed randomly, what should we expect the ending configuration is like? In my experience, the whole board gets cleared sometimes, but only rarely. Perhaps there's a nice explanation of this. If nothing else, we could simulate it with a computer program and tell you how likely it is to end with $n \leq 18$ cards for every $n \equiv 0 \pmod{3}$. Assuming the SET people are right, you never end with 21 cards, but I've certainly never seen a game end with 18, so it seems pretty unlikely.
- (3) I could act as a consultant for whatever else is on your mind mathematically about this game.

Expected project output: Solid answers to the above questions!

Difficulty level: Not sure about the first problem, but the second problem should be easy for someone good with coding.

Mathematical prerequisites: None.

Math you'll get to learn along the way: Linear algebra over finite fields.

Computer use: Highly recommended, especially if you're interested in the second problem.

Preferred number of project participants: 1–10.

Degree of supervision: Minimal. Perhaps meeting once a week.

MORE ON THE RAINBOW GAME

Supervised by: Mira

Description: Do you like the Rainbow Game from this year's Qualifying Quiz? So do I: it's the only nice problem that I have ever invented, and I'm very proud of it!

I think there's more interesting math there besides finding a strategy that works. For example, there are actually many non-isomorphic strategies and one can try to count and describe them. Two years ago, I had a student of mine at Wellesley look at the problem and we found a characterization of all possible strategies that connects to Latin squares. And this year, Justus Matthiesen, on his Qualifying Quiz, found a really cool connection to Hall's Marriage Theorem (which may also be related to what my Wellesley student and I were doing – I haven't had a chance to think about that yet).

I would really love it if one of you guys tried to find some more interesting math in the Rainbow Game! I can't guarantee that there's any to be found, but I think it's worth trying. And if you do find any, it could well be publishable.

Expected project output: Hopefully, some brand new math. At the very least, you'll get to "build" all possible solutions for the 3-person game out of Zometool!

Difficulty level: I have no idea. Mostly, I think, it's just about being clever.

Mathematical prerequisites: None.

Math you'll get to learn along the way: Certainly Latin squares and Hall's Marriage Theorem if you don't already know about them. Who knows what else!

Computer use: Not sure, but certainly not required.

Preferred number of project participants: Any.

Degree of supervision: I'll tell you everything I know and then you're basically on your own. I think we should try to meet regularly for progress reports, but I'm flexible.

NONORIENTABLE KNITTING

Supervised by: Anti and Ellen

Description: A Möbius strip is a surface with only one side and one boundary.

A Klein bottle is like a Möbius strip, but it has no boundary. When immersed in 3-space, it intersects itself. When knitted in 3-space, it can be folded neatly into itself and placed on the head so as to keep one's head and ears warm. No previous knitting experience required (but if you have some, that's okay too). We'll play with Möbius strips and Klein bottles made out of paper while we learn the basics of knitting, and then I'll show you a special trick for knitting things like Möbius strips and Klein bottles. It's so easy that even people who know practically nothing about knitting can do it.

Expected project output: You should end up with a knitted Möbius strip, and possibly a start on a knitted Klein bottle or other wonderful things.

Difficulty level: Easy.

Mathematical prerequisites: None.

Math you'll get to learn along the way: What Möbius strips and Klein bottles are and how to make them by gluing things together.

Computer use: None.

Preferred number of project participants: $1 - \infty$.

Degree of supervision: Low.

NUMBER FIELD SIEVE

Supervised by: David Roe

Description: The number field sieve is the fastest known factoring algorithm for very large integers. It is also the most difficult to understand and implement. In this project, you'll learn the background math necessary to understand the algorithm, then design and write your own implementation of it.

Expected project output: Either pseudocode or actual code implementing the number field sieve.

Difficulty level: ****

Mathematical prerequisites: Knowledge of groups, rings and fields. Elementary number theory. Programming skills will be useful, but not required.

Math you'll get to learn along the way: Algebraic number theory, factoring algorithms.

Computer use: At the end, if desired.

Preferred number of project participants: 1-3

Degree of supervision: Medium: I'm happy to help answer questions, talk to you, and guide you toward references, but I won't be teaching you everything you need to know.

OPEN PROBLEMS IN COMBINATORIAL GAME THEORY

Supervised by: Alfonso.

Description: Here is a collection of unsolved problem in combinatorial game theory. Do not despair, they are hard, but not untractable. (See also “Projects in combinatorial game theory”).

(1) *Chomp*

Let us fix a positive integer N . Two players take turn in naming positive divisors of N . No multiple of a number already mentioned can be mentioned. The player who says 1 loses.

The simplest form of this game is $N = 2^a \cdot 3^b$ (and it is then equivalent to a game played with chocolate you may be familiar with). We can prove there is a winning strategy for the first player, but we do not know what this strategy is. Can you find it?

Computer programming may or may not be helpful.

(2) *More Wyt Queens*

We know the solution for Wyt Queens with one queen. In class, Rachel proved a formula that gives all the P positions, and Keith derived a recursive expression to obtain them. However, if we want to play with several queens, we need the nim values of all the positions. This is still an unsolved problem.

A related question: what do you do with three dimensional Wyt Queens?

(3) *Period of subtraction games*

We fix a finite set of positive integers S . We start with a pile of N beans, and we take turns in subtracting beans. On your turn, you may subtract any number of beans, as long as that number is in S . If you calculate the nim value $n(k)$ of a pile with k beans, then $n(k)$ is eventually periodic as a function of k . Can you find a formula for the (minimum) period in terms of S ?

Heavy programming knowledge may be very useful for this problem.

(4) *The princess and the roses*

Princess Alice has two suitors: Laura and Renata. They alternate days in trying to gain her heart by bringing her roses. They pick up the roses from the same garden. Each day, the corresponding suitor will bring her one or two roses, but never two roses of the same colour. The suitor to pick the last rose will bring Alice’s heart. Assuming that initially there are 3 blue roses, 4 red roses, 5 yellow roses, and 6 white roses, who will win?

Can you solve the same problem if there are roses of k colours, with initial values (n_1, n_2, \dots, n_k) . Despite the very simple rules, we only know the answer to these problem for small values of n_i or for $k = 5$, but not in general.

(5) *Antonim*

Antonim is played with the same rules as nim, but we are not allowed to leave two piles with the same number of beans. We have a nice pattern for 3 pile antonim, but we have no idea for 4 pile antonim.

(6) *Games born on Day n*

In the beginning there was nothing.

On Day 0, 0 was created with nothing as left moves, and nothing as right moves.

On Day 1, three more games were created with nothing or 0 as left moves or right moves: 1, -1 , and \star .

On Day 2, sixteen new games were created.

How many games were created on Day 3? How many on Day n ?

In MC2004 this problem was solved *by hand* for $n = 3$. I strongly suggest you attempt it with a computer this time. You will need to learn how to write the canonical form of a game.

Expected project output: A solution to any of the problems would be really cool! But a partial result will be interesting, too.

Difficulty level: ****

Mathematical prerequisites: Enough knowledge of combinatorial game theory to understand the statement of the problem (or willingness to learn it).

Math you'll get to learn along the way: More game theory.

Computer use: Depends on the problem.

Preferred number of project participants: 1 or a small group.

Degree of supervision: Depends on you. I will be available to discuss your ideas, although I expect you to take the initiative once we get started.

OPTIMAL CAR RACING

Supervised by: Rob.

Description: You've got a race car that follows some simple physics on a curvy track, and you want to determine the fastest way to drive the car from the start to the finish line. It's probably infeasible to find the exact optimal solution, but how fast can you make it?

This problem was the goal of the ICFP 2003 programming contest. Competitors used clever tricks to reduce the size of the problem, effective approximations, and even solutions found by a human and computer working together, in the span of three days.

You have more time to think, and you can see what worked well for others. Can you do better than those solutions?

Expected project output: Some efficient routes around the contest tracks, and a computer program to help find them.

Difficulty level: ***

Mathematical prerequisites: Some programming experience.

Math you'll get to learn along the way: Computational geometry and algorithms.

Computer use: High

Preferred number of project participants: 2 or 3.

Degree of supervision: Moderate (Rob will be available for programming help).

PLAYING TWENTY QUESTIONS WITH A LIAR

Supervised by: Mira

Description: I'm thinking of a number from 1 to 16. You need to guess it by asking me yes/no questions. The twist is that once during the game, I'm allowed to lie. How many questions will it take for you to figure out my number? What if I *have* to lie? What if you have to decide on the questions ahead of time? What if instead of 1 to 16 it's 1 to 2048? And what does all this have to do with Nim and your CD player?

Expected project output: A poster and a chance to perform an impressive trick at the end-of-camp project fair: someone will think of a number, you'll ask them a few random-sounding yes/no questions, and then say, "I can tell that you lied on question number 5; your number is 13!"

Difficulty level: One star! This project is very elementary; if you've already seen any coding theory, stay away.

Mathematical prerequisites: None.

Math you'll get to learn along the way: This is related to lots of stuff: linear algebra, codes, games, finite projective planes – you name it. It just depends on where you take it.

Computer use: Probably none.

Preferred number of project participants: 1–2

Degree of supervision: You'll be discovering everything for yourself, but I'll be there to guide you when you need it. Meeting every few days sounds about right. I'll be gone in Week 4, but there are plenty of people who can take over from me during that time.

PRIMES IS IN P

Supervised by: Mira.

Description: PRIMES is the problem of determining whether a given integer is prime. For years, people had been wondering whether “PRIMES is in P” – i.e. whether it is possible to test for primality in polynomial time. Then, in 2002, an Indian mathematician and two of his students came up with a beautifully simple proof that it can be done! Simplicity, of course, is relative: on the Mathcamp scale their proof is ****, but it’s definitely accessible to Mathcampers. (In fact, David Roe and I taught a **** class on it in 2003.) In this project, you will read the original paper (only about 7 pages long) and work out some of the details that the authors leave out (which turn out to be quite interesting).

Expected project output: A poster, a talk, or just the sense of accomplishment at reading and understanding an important research paper.

Difficulty level: ****

Mathematical prerequisites: Basic number theory, preferably some familiarity with rings and fields. If you want to do this project, you should definitely take Dave Savitt’s class on finite fields in Week 3. Theoretical Computer Science is **not** a prerequisite: the background that you need is very minimal and easily learned.

Math you’ll get to learn along the way: A wide assortment of topics, ranging from primitive polynomials (which made a brief appearance in the card trick I showed in Week 1) to the Fast Fourier Transform.

Computer use: None. (You don’t actually want to implement this algorithm. For one thing, despite being in P, it is *really* slow and never used in practice.)

Preferred number of project participants: 2–3.

Degree of supervision: I’ll be happy to meet with you every two or three days to answer questions and give guidance. I’ll be gone in Week 4, but there are plenty of people who can take over from me during that time.

PROJECTS IN COMBINATORIAL GAME THEORY

Supervised by: Alfonso.

Description: Here are some problems with nice solutions in Combinatorial Game Theory. (See also “Open problems in combinatorial game theory”).

(1) *Simonim* (***)

This game is played like Nim with an additional feature. We have various piles of beans. In your turn, you may:

- remove one or more beans from the same pile
- remove the exact same amount of beans from various piles, as long as those piles have the same amount of beans to begin with.

Notice that you cannot analyze the various piles independently. This game is harder than Nim, but is still has a pattern. Can you find it?

(2) *Almonds* (**)

Two players play this game with various piles of almonds (not beans). In his turn, a player can divide a heap into two or more equal-sized heaps or unite any two heaps of different sizes. The first player who cannot move loses. When can you find a winning strategy in a finite amount of time? Careful, as this game may be loopy!

(3) *The mentor and the campers* (***)

This game is played in a $N \times N$ chessboard. The two players take turns moving. Player L starts with a mentor in the center of the board. In her turn, player L can move the mentor like a chess king. In his turn, player R can place a camper (from an infinite supply) in any square not occupied by the mentor. Campers do not move and the mentor cannot move into a square with a camper. Player L wins if she manages to move the mentor to the side of the board. Player R wins if he manages to surround the mentor entirely with campers.

If N is small, then the mentor always win. If N is large , then the campers always win. There is a critical value of N , for which the first player to move wins. What is this critical value and what is the winning strategy?

(4) *Exponential nim* (*)

We play with one pile of beans and take turns removing beans. The first player may take any amount he wants, but not the whole pile. After that, a player may take at most twice as many beans as the previous player took. The player to remove the last bean wins. Can you find a winning strategy?

Careful on how you formulate this game, because a position depends on history!

- (5) *Berlekamp's rule for hackenbush* (**)
You know that in hackenbush

Find a formula for the value of a general, blue–red hackenbush string of arbitrary (finite) length.

Expected project output: A solution to one of the questions above.

Difficulty level: See each one of them.

Mathematical prerequisites: You need to have taken my CGT class, or be familiar with game theory.

Math you'll get to learn along the way: More game theory.

Computer use: See each one of them.

Preferred number of project participants: 1 or small group.

Degree of supervision: I expect you to have fun solving the problem yourselves, and I will be available to hear your conjectures, catch mistakes, and give you hints if needed (hopefully not!).

A RICOCHET ROBOT SOLVER

Supervised by: Dan.

Description: The game Ricochet Robot has an exponential tree when trying to solve each puzzle posed. When humans play, they usually find suboptimal solutions — human solutions are nice, but the “optimal” solutions are amazingly beautiful. We’ll construct a program to play Ricochet Robot, and we’ll do our best to make that program able to solve any position at all. We’ll also try to come up with useful heuristics to make its solutions faster. In the end, perhaps we’ll try to find out what positions are reachable at all, and also to generalize to larger boards.

Expected project output: A program that solves Ricochet Robot.

Difficulty level: Easy, but programming skill required.

Mathematical prerequisites: Some algorithms, programming, and data structures.

Math you’ll get to learn along the way: First-hand knowledge of algorithmic (lack of) efficiency. :)

Computer use: Very heavy.

Preferred number of project participants: 2–3

Degree of supervision: I’ll be around to help and join in on brainstorming meetings, but I’ll expect you to code and think about it lots on your own.

SOLVING TSP

Supervised by: Ellen, Rob, and Dan.

Description: The Travelling Salesman Problem (TSP) is one of the most famous NP-complete problems. It asks: given a graph with weighted edges, what is the minimal cost of visiting every node of the graph?

We're going to try to make a program to solve some instances of TSP and look into algorithms for approximating solutions. If we manage to come up with an interesting algorithm then we'll code it up and try to solve some previously unsolved instances of TSP.

Expected project output: A program that solves or approximates TSP.

Difficulty level: Depends on what you want to do; probably Medium or Easy.

Mathematical prerequisites: None, but theoretical CS will help you understand why the problem is interesting.

Math you'll get to learn along the way: Some graph theory, algorithms.

Computer use: Heavy.

Preferred number of project participants: 2–3.

Degree of supervision: Medium; we'll be around to help a fair bit.

THE UNIQUE GAMES CONJECTURE

Supervised by: Dan.

Description: The Unique Games Conjecture is a computer science problem about coloring a graph. The graph has certain rules for coloring. Coloring a graph by these rules is actually quite easy, and can be done in polynomial time.

Now suppose you adjust the problem so that you're allowed to occasionally disobey the rules for coloring some small percentage of the time. The problem suddenly becomes much, much harder. So hard that we think it's NP-hard, in fact.

This project will be an attempt to do some work on the unique games conjecture — possibly to try to prove it (unlikely!) or to do some special cases, or at least to understand how it fits into the global scheme of complexity theory, including possibly talking about probabilistically checkable proofs (a major result from the mid 90s).

Expected project output: A proof or partial work on a proof, understanding of some crazy complexity theory.

Difficulty level: Hard.

Mathematical prerequisites: Theoretical CS — as much as possible. Some experience with algorithms helpful.

Math you'll get to learn along the way: Lots of complexity theory and/or graph theory.

Computer use: Probably none.

Preferred number of project participants: 3.

Degree of supervision: Some, but participants should be willing and motivated to work independently even when the direction isn't clear.