15-251: Great Theoretical Ideas in Computer Science

Fall 2016 Lecture 1.5

August 31, 2016

Proofs

Bits of Wisdom on Solving Problems, Writing Proofs, and Enjoying the Process: How to Succeed in This Class

No specific topic covered today,
but we'll very briefly recap induction

2. What is a proof?

1. How do I find a proof?

3. How do I write a proof?

2. What is a proof?

1. How do I find a proof?

3. How do I write a proof?

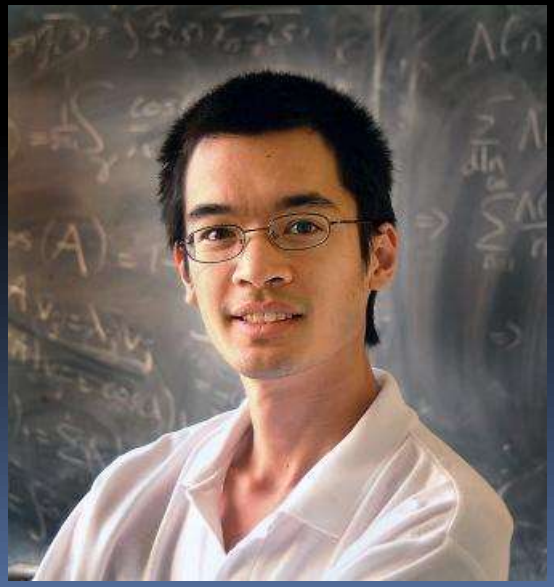

The “Aha!” Moment

Typical philosophy for working in math:

Small progress per day,
for many days.

251 HMWK version: 15% progress per day for 7 days.

I don't have any magical ability. I look at a problem, and it looks something like one I've done before; I think maybe the idea that worked before will work here. When I was a kid, I had a romanticized notion of mathematics, that hard problems were solved in 'Eureka' moments of inspiration. [But] with me, it's always, 'Let's try this. That gets me part of the way, or that doesn't work. Now let's try this. Oh, there's a little shortcut here…. It's not about being smart or even fast. It's like climbing a cliff: If you're very strong and quick and have a lot of rope, it helps, but you need to devise a good route to get up there. Doing calculations quickly and knowing a lot of facts are like a rock climber with strength, quickness and good tools. You still need a plan — that's the hard part — and you have to see the bigger picture.



Terence Tao

2006 Fields Medalist,
winner of 10+ international math
prizes worth over $5 million

10 tips for finding proofs

1. Read and understand the problem.
2. Try small or special cases.
3. Develop good notation.

4. Understand why the problem seems hard
   (Put yourself in the mind of the adversary)
5. Collaborate, bounce off ideas.

10 tips for finding proofs

6. Use blocks of ≥ 1 hour, or at least 30 minutes.
7. Take breaks.
8. Use plenty of paper (or whiteboard/tablet), and draw pictures if possible.
9. Clarify, abstract out, summarize pieces. Record partial progress.
10. A crisp write-up is important (both for scoring points, and checking that argument is airtight).

A 251 Homework Problem:

The kitchen for a cookie baking contest is arranged in an m by n grid of ovens. Each contestant is assigned an oven and told to make as many cookies as possible in three hours. Prizes are awarded in the following manner: in each row the p people who produced the most cookies receive a prize. Likewise, in each column the q people who produced the most cookies receive a prize. Assume $p \leq n$, $q \leq m$, and that no two people produced the same number of cookies. Prove that at least pq people received two prizes for their cookie-baking performance.

Solution write-up

Proof by induction on n+m.

P(k) = claim true when n+m=k for all (p,q) $\in$ {1,2..,n} x {1,2,…m}

P(2) is true (n=m=p=q=1)

Assume P(k) is true. Let's prove P(k+1). Suppose n+m=k+1.

If everyone who wins a prize wins two prizes, we are done, since at least (mp+nq)/2 ≥ pq people win prizes.

So there is someone who receives just one prize. Among those, pick the person, say X, who made the most cookies. Either X is not among top p in her row or not among the top q in her column. *Without loss of generality,* assume the latter. (Why's this okay?)

Remove X's column. By induction hypothesis, the remaining m x (n-1) grid has at least (p-1)q people receiving two prizes (since every row has at least (p-1) prize winners in new grid). Add to this set the q winners in X's column, who by choice of X, all win two prizes (otherwise X wouldn't have been the largest single prize winner). This gives pq two-prize winners in all. QED.

If you just read the solution, it's frustrating:

Writeup is short: 3 short paragraphs.

Seems to have some "aha!" moments (eg. choice of X)

Hides cognitive process behind discovery of "aha!"-like step(s).

But you need to set yourself up for making such a step.

For the write-up, you can step back and try for the clearest possible explanation (which often is also succinct, but some intuition is nice to include, especially in difficult proofs).

2. What is a proof?

1. How do I find a proof?

3. How do I write a proof?

What is a proof?

In math, there are agreed-upon rigorous rules of deduction. Proofs are right or wrong.

Nevertheless, what constitutes an acceptable proof is a social construction.

(But computer science can help.)

Proofs — prehistory

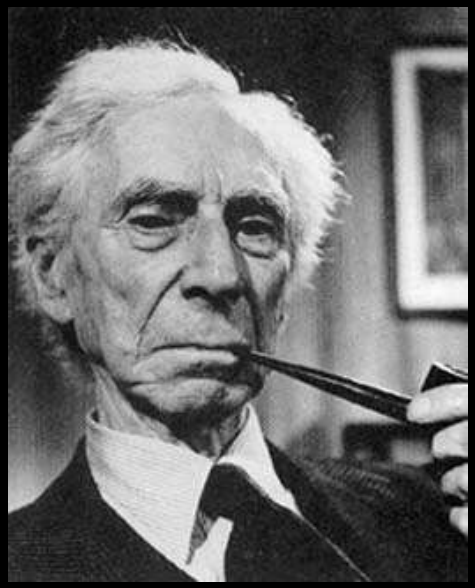
Euclid's *Elements*
(ca. 300 BCE)

Canonized the idea of giving
a rigorous, axiomatic deduction
for all theorems.

Proofs — 19th century

True rigor developed.

Culminated in the understanding that math proofs can be formalized with First Order Logic.



Bertrand Russell Alfred Whitehead

Principia Mathematica, ca. 1912

Developed set theory, number theory, some real analysis using **formal logic.**

page 379: “1+1=2”

It became generally agreed that you ***could*** rigorously formalize mathematical proofs.

But nobody wants to!

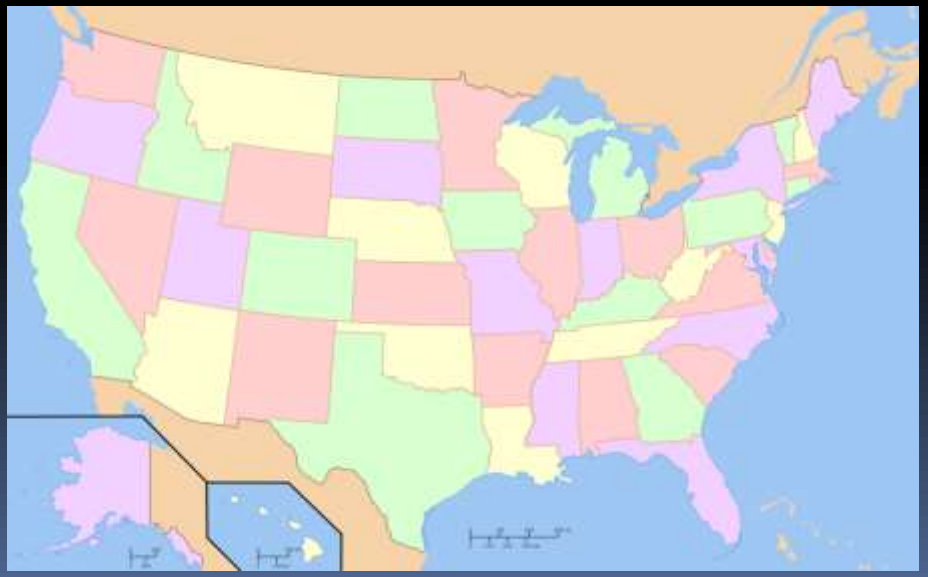
(by hand, at least)

But are English-language proofs sufficient?

Four Color Theorem

1852 conjecture:

Any 2-d map of regions can be colored with 4 colors so that no adjacent countries get the same color.



Four Color Theorem

1879: Proved by Kempe in *Amer. J. of Math*

1880: Alternate proof by Tait in
Trans. Roy. Soc. Edinburgh

1890: Heawood finds a bug in Kempe's proof.

1891: Petersen finds a bug in Tait's proof.

Kempe's "proof" was widely acclaimed.

Four Color Theorem

1969: Heesch showed that the theorem could in principle be reduced to checking a large number of cases.

1976:

Appel and Haken wrote a massive amount of code to compute and then check 1936 cases
(1200 hours of computer time).

Claimed this constituted a proof.

More anecdotes

1993: Wiles announces proof of Fermat's Last Thm.
Then a bug is found.

1994: Bug fixed, 100-page paper.

1994: Gaoyong Zhang, *Annals of Mathematics*:
disproves "n=4 case of Busemann-Petty".

1999: Gaoyong Zhang, *Annals of Mathematics*:
proves "n=4 case of Busemann-Petty".

Kepler Conjecture

Kepler, 1611: As a New Year's present (???) for his friend, wrote a paper with this conjecture:

The densest way to pack spheres is like this:

Kepler Conjecture

2005:

Our neighbor Tom Hales:
120 page proof in
Annals of Mathematics

Plus code to solve 100,000 distinct optimization problems, taking 2000 hours computer time.

Annals recruited a team of 20 referees.
They worked for 4 years.
Some quit. Some retired. One died.
In the end, they gave up.
But said they were “99% sure” it was a proof.

Kepler Conjecture

Hales: “We will code up a completely formal axiomatic proof, checkable by computer.”

Open source “Project Flyspeck”:
2004 --- August 10, 2014

Computer-assisted proof

Proof assistant software like
HOL Light, Mizar, Coq, Isabelle,
does two things:

1. Checks that a proof encoded
   in an axiomatic system for
   First Order Logic (or typed lambda calculus theory) is valid.

2. Helps user code up such proofs.

Developing proof assistants is an
active area of research, particularly at CMU!

Computer-formalized proofs

Fundamental Theorem of Calculus (Harrison)

Fundamental Theorem of Algebra (Milewski)

Prime Number Theorem (Avigad++ @ CMU)

Gödel's Incompleteness Theorem (Shankar)

Jordan Curve Theorem (Hales)

Brouwer Fixed Point Theorem (Harrison)

Four Color Theorem (Gonthier)

Proofs in 251

For theorems we will prove in 251, we won't need computer assistance. (Though you're welcome to program small test cases if it helps in formulating hypotheses & solving HW problems.)

Higher-level, but *precisely argued* proofs.

Appropriate level of detail in proof also depends on context and target audience:

- Your proofs need to convince TAs/instructors that you have a clearly articulated air-tight solution.

2. What is a proof?

1. How do I find a proof?

3. How do I write a proof?

So as to get full points
on the homework.

Your homework is not like
the Four Color Theorem.

The TAs can correctly decide
if you have written a valid proof.

Here is the mindset you must have.

Pretend that your TA is going to
code up a formalized proof of your solution

Your job is to write a complete
English-language **spec** for your TA.

You must give a spec to your TA
that they could implement
with no complaints or questions.

Equivalently, you must
convince your TA that you know
a complete, correct proof.

Alternate Perspective

You: must present an airtight case.

Your TA

Possible complaints/points off from your TA:

- A does not logically follow from B.
- You missed a case.
- This statement is true, but you haven't justified it.

But also:

- Your without loss of generality is *with* l.o.g.
- I don't understand your proof.
- This explanation is unclear.
- Your proof is very hard to read.

Problem: Prove $n^2 \geq n$ for all integers n.

Solution:

We prove F_n = "$n^2 \geq n$" by induction on n.
The base case is n = 0: indeed, $0^2 \geq 0$.
Assume F_n. Then
$(n+1)^2 = n^2+2n+1 \geq n^2+1 \geq n+1$ (by F_n).
This is F_{n+1}, so the induction is complete.

Read the question carefully.

Some common induction mistakes

"The base case F_0 is true because […].
For the induction step, assume F_k holds for all k.
We now show that F_{k+1} holds…"

You just assumed what you're trying to prove!

"The proof is by strong induction.
The base case F_0 is true because […]
For the induction, assume F_k holds for all $k \leq n$.
We will now show F_{k+1}: […]"

What is k? Where did n go?

Old homework problem:

How many ways to arrange c ≥ 0 ♣'s and d ≥ 0 ♦'s so that all ♣'s are consecutive?

Solution:

You can have any number between 0 and d ♦'s, then the string of ♣'s; then you must have the remainder of the ♦'s. Hence there are d+1 possibilities.

Fallacious if c = 0: there is only 1 possibility.

Handle all edge cases!
Don't have any missing parts in your spec.

Problem: Prove $2^n > n$ for all integers $n \geq 1$.

Solution:

F_n = "$2^n > n$"

F_1 = "$2 > 1$" ✓

$F_n \Rightarrow F_{n+1}$:

$2^{n+1} = 2 \cdot 2^n > 2 \cdot n$ (induction) $\geq n+1$

because $n \geq 1$

Therefore proved.

Is this a definition? A claim?

What does this check mark mean?

Is this a claim? An assumption?

Oh, apparently you're doing an induction? [sarcasm]

Is it? Why?

This is not a full sentence.

This is not written in English!

Another old homework question:

There is a circle of 15,251 chips, green on one side, red on the other. Initially all show the green side. In one step you may take any four consecutive chips and flip them. Is it possible to get all of the chips showing red?

Intended solution:

No. If g of the 4 flipped chips are green, then after flipping $4-g$ of them are green. Note that g and $4-g$ have the same parity; hence the parity of the number of green chips will always remain odd.

Solution:

No it is not possible. Let's assume for contradiction we converted all 15,251 chips to red. But this means in the very last step there must be 4 consecutive green chips and the remaining 15,247 must be red. Repeating this k times for
$1 \leq k \leq 3812$, we get three consecutive red chips, with the rest green. But we started from all green, contradiction.

If asked to show something is impossible, it does not suffice to show that one particular method does not work.

Spring ’11 homework 2, #3b:

There is a circle of 15,251 chips, green on one side, red on the other. Initially all show the green side. In one step you may take any seven consecutive chips and flip them. Is it possible to get all of the chips showing red?

Intended solution:

Yes. Number the chips 0…15,250. Flip the sequence [0,1,…,6], then [1,2,…,7], then [2,3, …,8], etc., up until [15,250,0,1, …,5]. Now each chip’s been flipped exactly 7 times, an odd number. Hence each chip is now red.

Solution:

At any given time, let g be the number of chips showing green and r the number of chips showing red. The possible remainders when a number is divided by 7 are 0, 1, 2, 3, 4, 5, 6, 7. A flip that involves 6 red and 1 green increments the current modular class of g by 5 while the move that involves 1 red and 6 green decrements the current modular class of g by 5. Originally, with the number 15,251, the modular class of g mod 7 is 5. Thus, it is possible to make all chips red.

In short: this proof does not make sense.
Do not just write a bunch of random facts.

Success in computer science requires:

- Content: An up to date grasp of fundamental concepts and problems
- Method: Principles and techniques to solve the vast array of *unfamiliar* problems that arise in a *rapidly changing field*

251 will surely have lot of content, but its overarching aim is to use the topics as a vehicle to prepare you to

(i) model/abstract the core features of a problem,
(ii) reason rigorously, without fooling yourself, towards a correct solution, and
(iii) express your solution in a cogent, convincing manner

Quick review:
Structural Induction

Induction Principle:

If F_0 and $\forall k, F_k \Rightarrow F_{k+1}$

then $\forall n, F_n$

Well Ordering Principle:

Every non-empty set of positive integers contains a least* element

*under the usual ordering “<”

Inductive Proofs

To Prove $\forall k \in \mathbf{N}$, S_k

1. Establish "Base Case": S_0
2. Establish that $\forall k$, $S_k \Rightarrow S_{k+1}$

To prove $\forall k$, $S_k \Rightarrow S_{k+1}$

Assume hypothetically that S_k for any particular k;

Conclude S_{k+1}

Theorem:

Every natural number n > 1 can be factored into primes

S_n = “n can be factored into primes”

Base case:
2 is prime $\Rightarrow$ S_2 is true

How do we use the fact:

S_{k-1} = “k-1 can be factored into primes”
to prove that:

S_k = “k can be factored into primes”

Use
“all previous induction”

a.k.a. **strong induction**

Theorem:

Every natural number > 1 can be factored into primes

A different approach:

Assume 2,3,…,k-1 ***all*** can be factored into primes

Then show that k can be factored into primes

Strong Induction

To Prove $\forall k, S_k$

Establish Base Case: S_0

Inductive step:

For $k > 0$, assume $\forall j<k, S_j$
use that to derive S_k

Invariant (n):

1. Not varying; constant.
2. *Mathematics.* Unaffected by a designated operation, as a transformation of coordinates.
3. *Programming.*
   A rule, such as the ordering of an ordered list, that applies throughout the life of a data structure or procedure. Each change to the data structure maintains the correctness of the invariant

Invariant Induction

Suppose we have a time varying world state: W_0, W_1, W_2, …

Each state change is assumed to come from a list of permissible operations. We seek to prove that statement S is true of all future worlds

Argue that S is true of the initial world W_0

Show that if S is true of some world – then S remains true after one permissible operation is performed

Odd/Even Handshaking Theorem

At any party at any point in time define a person's parity as ODD/EVEN according to the number of hands they have shaken

Statement:
The number of people of odd parity must be even

Statement: The number of people of odd parity must be even

Initial case: Zero hands have been shaken at the start of a party, so zero people have odd parity

Invariant Argument:

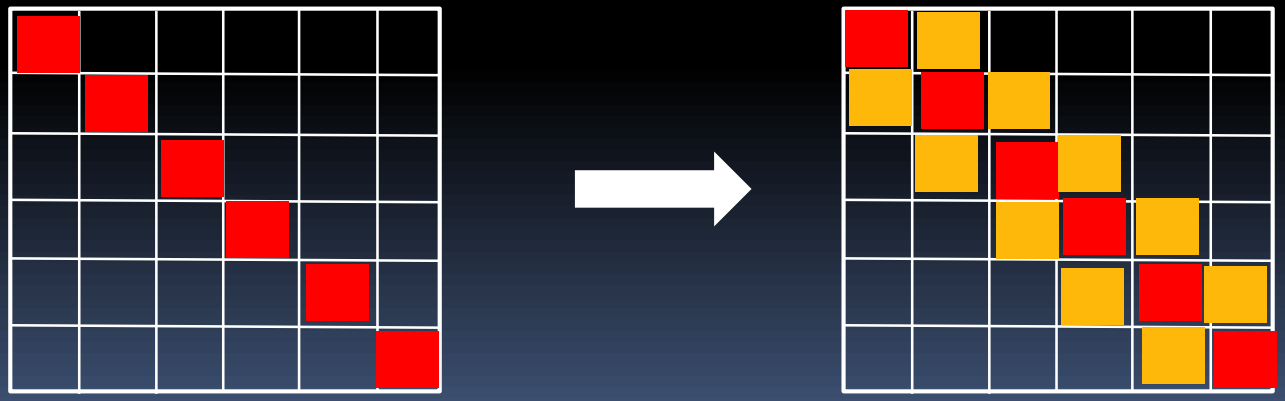
If 2 people of the same parity shake, they both change and hence the odd parity count changes by 2 – and remains even

If 2 people of different parities shake, then they both swap parities and the odd parity count is unchanged

Bored? Here's a "chessboard" puzzle

An infection spreads among the squares of an n x n chessboard in the following manner. If a square has two or more infected neighbors, then it becomes infected itself. (Neighbors are orthogonal only, so a square has at most 4 neighbors)

What's the minimum number of infected squares needed at the beginning to infect the whole board?



Structural Induction

Induction can also be used to define or construct objects of various kinds (sets, sequences, formulas, trees,…)

Structural induction is a convenient form to prove results about recursively-defined objects

Let's illustrate with a simple example.

Consider a set S defined by:
Base step: $3 \in S$
Recursive step: If $x \in S$ and $y \in S$, then $x + y \in S$

What's the set S?

All positive multiples of 3

Proof that $S = \{ 3n \mid n$ is a positive integer$\}$

Denote T $= \{ 3n \mid n$ is a positive integer$\}$

Two directions:

1. $T \subseteq S$: Proof by induction on n
2. $S \subseteq T$: **Structural induction**

- Base step: Check the claimed property for the base cases of the definition.
- Recursive step: Prove the claim holds for new objects created by the recursive combination rule, assuming that it is true for the old objects used in the recursive step.

In our example:

Base case: $3 \in T$

Recursive step: If x, y are multiples of 3, then so is $x + y$

Why is structural induction valid?

It follows from strong induction on the *number of applications of the recursive rule to create a particular object*

It is just a convenient packaging so we don't have to repeat saying "Let's induct on n,
the number of applications of recursive rule …"

(Rooted) Binary tree

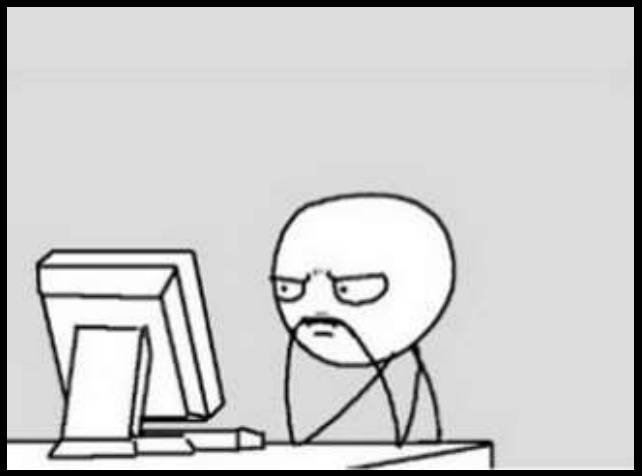
Base step: A single node r is a binary tree with root r

Recursive step: If T_1 and T_2 are binary trees with roots r_1 and r_2, then T which has a node r adjacent to r_1 and r_2 is a binary tree with root r (and r_1, r_2 are called children of r)

A *leaf* of a binary tree is a node with no children. Rest of nodes are called *internal*

Claim: In every binary tree, the number of leaves is one more than the number of internal nodes.

Easy exercise: Prove above by structural induction



Study Guide

Solving problems:

- Understand problem
- Try small cases
- Use enough time & paper
- put yourself in the mind of adversary

Writing proofs:

- like designing a complete, correct spec
- put yourself in the TA's shoes
- use good English!