Massachusetts Institute of Technology
6.042J/18.062J: Mathematics for Computer Science
Professor David Karger

Lecture 5
February 20, 1997

Class Notes for Lecture 5

8 Strong Induction

8.1 Definition

A variant of induction. To prove $(\forall n \in N)P(n)$

- prove $P(0)$
- prove $((\forall k < n)P(k))$ implies $P(n)$.

Actually, don't even need the base case.
That's because the second case, when specialized to $n = 0$, says to prove $((\forall k < 0)P(k))$ implies $P(0)$.
But the hypothesis is *vacuously true*, since there's nothing < 0.
So this means you're really forced just to prove the conclusion, $P(0)$, on its own.

But never mind if this is confusing – we'll follow the book and check base cases – they're generally easy anyway.
Another reason for base cases:
Often the first case or first few cases need a different argument for the inductivve step, show explicitly.

Again, can modify this to work from any k instead of 0.

Follows from standard (weak) induction on the proposition $Q(n) = (\forall k < n)P(k)$ (base case at $n = 1$)

8.2 Nim

The game of Nim is defined as follows. A bunch of sticks are placed on the ground. Two players take turns removing one, two, or three sticks. The player to remove the last stick loses.

Theorem 8.1 First player loses if the number of sticks n is $4k+1$ for some k, and wins (if he plays optimally) otherwise.

We can formalize this if we define a strategy as a rule for how many sticks to remove when there are m left. If $n = 4k+1$, then player 2 has a strategy that will force a win for him. Otherwise, player 1 has a strategy that will force a win for him.

Proof. By strong induction. Suppose theorem true for numbers less than n, show for n. Want to show that if $n = 4k+1$ then first player loses, and otherwise ($n = 4k, 4k+2, 4k+3$) the first player wins.
(We'll see later why this exhausts all possible cases for n) There are four cases:

- $n = 4k+1$: show the first player loses. Case $n = 1$, first player has no choice but to remove 1 stick and lose.
  Case $n \neq 1$. So have $n \geq 5$.

  Consider what the first player might do to win.
  Can only choose to remove 1, 2 or 3 sticks.

  If removes one stick, the remainder is $n - 1 = 4k$.
  By strong induction, the player who plays at this point should win.
  It follows that the player who played first will lose.

  Similarly, if the first player removes two sticks, the remainder is $4(k-1)+3$, and he loses by the same reasoning.

  Similarly again for removing 3 sticks.

  So however the first player moves, he loses.
- $n = 4k$. show first player can win. Have first player remove 3 sticks.
  the second player sees $4(k-1)+1$ sticks.
  This means he loses, by stong inductive hypothesis.
- $n = 4k+2$, show first player can win.

  Similar to the previous case.
- $n = 4k+3$, show first player can win.

  Again similar.

From the division algorithm that we'll se later, we know these are all the possible cases. ∎

8.3 Structural Induction on a Recursive Definition

Induction is often used to prove properties for all elements of a set defined recursively, like well-formed arithmetic formulas or binary trees.
This is usually formulated so it looks like a different kind of inductive method:
Two steps:

- Prove for "base cases" of definition
- Prove for the result of any combination rule, assuming it's true for parts.

But this isn't really different. It's an instance of strong induction on the number of formation steps.
Just say that the base cases correspond to 0 applications of combination operations.

Give two examples here.

8.3.1 Well Formed Formulas

A WFF is:

- a number
- $(f+g)$, $(f-g)$, $(f*g)$, or (f/g) where f and g are WFFs.

Theorem 8.2 Well formed formulas always have the same number of left and right parentheses.

Proof. The proof is by strong induction on n, the number of applications of the "combination rules".
Let $P(n)$ be the proposition that a well formed formula formed by n applications of the combination rules has the same number of left and right parentheses.

BASE CASE: $P(0)$:
If the formula is just a number then the above proposition is obviously true since there are 0 left and right parentheses. Hence, $P(0)$ is true.

INDUCTIVE STEP: Assume $P(k)$ to be true, for all $k < n$. Consider a well formed formula formed by n applications of the combination rules. There are four cases.

1. The formula is of the form $(f + g)$, where f and g are well formed. Since f and g are well-formed and are formed by fewer than n applications of the combination rules, by the strong inductive hypothesis, they have an equal number of left and right parentheses and hence so does $(f + g)$.
2. The formula is of the form $(f - g)$. This case is similar to the first.
3. The formula is of the form $(f * g)$. This case is similar to the first.
4. The formula is of the form (f/g). This case is similar to the first.

This completes the inductive step and hence $P(n)$ is true for all nonnegative integers n. ∎

8.3.2 Binary Trees

A *binary tree* is one of

- A *node* with no children.
- A node with an edge to a left child that is a binary tree, and no right child.
- A node with a right child that is a binary tree, and no left child.
- A node with left and right binary-tree children.

Theorem 8.3 The number of edges in any binary tree is exactly one fewer than the number of nodes.

Proof. The proof is by strong induction on n, the number of applications of the "combination rules".
Let $P(n)$ be the proposition that a binary tree formed by n applications of the combination rules has 1 fewer edge than nodes.

BASE CASE: $P(0)$ is true since a binary tree that is a node with no children has 0 edges.

INDUCTIVE STEP: Assume $P(k)$ to be true, for all $k < n$. Consider a binary tree formed by n applications of the combination rules. There are three cases.

1. The tree is a node with a left child that is a binary tree.

   Since the left child is a binary tree formed by $n - 1$ applications of the combination rules, by the inductive hypothesis, it has 1 fewer edge than nodes. Hence the whole tree has 1 fewer edge than nodes, too.
   (Adding one edge and one node.)

2. The tree is a node with a right child that is a binary tree.

   This case is identical to the previous case.

3. The tree is a node with left and right binary-tree children.

   Then each of the two subtrees is formed by fewer than n applications of the combination rules. By inductive hypothesis (using strong induction here, since it needn't be $n-1$), each subtree has exactly 1 fewer edge than nodes. The whole tree adds 1 node and 2 edges, so also has exactly 1 fewer edge than nodes.

This completes the inductive step and hence $P(n)$ is true for all nonnegative integers n. This implies the result. ∎

9 Well Ordering Principle

9.1 Definition

The *well ordering principle* says:
Every nonempty set of natural numbers contains a least element.
This should be "obvious", as a property of natural numbers as commonly understood.

This may not look exactly like an induction principle, but it's similar.
Formally, it's equivalent (in the presence of basic axioms) to induction (weak or strong).
Come back to this later.

Well ordering not true for integers, positive reals, positive rationals.

An example:
We left a slight hole in our proof that $\sqrt{2}$ is irrational – we assumed that we could write a fraction in lowest terms,
that it, in such a way that the numerator and denominator have no common divisors.
Here is how to actually prove that fact:

Theorem 9.1 Every rational number can be written in lowest terms.

Proof. Fix rational q.
Let $S = \{b \in \mathbb{N} \mid (\exists a) q = a/b\}$.
(The set of possible denominators.)
S is nonempty, by the definition of q being rational.
Therefore S has a least element b and a corresponding a.

Then a/b is a lowest-terms representation of q (we still haven't proven it is unique, but we do know that b is the smallest possible denominator). ■

10 Number Theory

Having introduced the basic ideas of proofs, especially induction, we now turn to introducing the basic discrete structures used in computer science.

These are:
Natural numbers and integers.
Sets, relations, functions.
Equivalence relations, posets, trees.
Graphs.
State machines.

Then on to analysis techniques (counting, recurrences, etc.).

Natural numbers are $0, 1, \cdots$.
Integers also include negatives.
Operations are usual arithmetic operations $+$, $\times$, etc.
Ordering is the usual order.

We will review some basic concepts: divisibility, primes, composite numbers, modular arithmetic, etc.
Give basic theorems about these notions,
and some simple algorithims based on these theorems.

Fundamental theorem of arithmetic – every number has unique factorization into primes.
Algorithm for finding GCD, the greatest common divisor of two numbers.
RSA algorithm for encrypting/decrypting data.

Subject matter usually called *number theory*.
Used to be just for fun, but recently has become very important in practice, mainly because of applications to security.

We give some number theory examples of the use of well ordering and strong induction.

10.1 Factorization

Theorem 10.1 Every natural number ≥ 2 can be factored as a product of primes.

Proof. By well ordering.
Let S be the set of all natural numbers ≥ 2 that cannot be factored as a product of primes.
We assume $S \neq \emptyset$ and derive a contradiction.

Since S is not empty, well ordering implies that it has a least element n.

This element n cannot be a prime, because then it would be a (trivial) product of primes.
So n is composite – write $n = ab$ where $1 < a, b < n$.

Because $a, b < n$, and n is the smallest element of S, it follows that $a, b \notin S$.
This implies that a can be written as a product of primes, say $p_1 p_2 \cdots p_k$,
and b as a product of primes, say $q_1 \cdots q_l$.

Therefore, n can be written as $p_1 \cdots p_k q_1 \cdots q_l$,
a product of primes, contradicting the claim that $n \in S$.

Our assumption that $S \neq \emptyset$ must therefore be false (proof by contradiction). ∎

It is also possible to prove this result using strong induction.
The argument is very similar, in fact. (Check this.)
This similarity should give you the idea that the well ordering principle is very similar to strong induction.
In fact, it's equivalent (see later).

A fallacious argument:
Try to show that every number can be factored *uniquely* into primes.

Apply the same (well ordering) proof as above, modifying the definition of S:
Let S be the set of all natural numbers ≥ 2 that cannot be factored *in exactly one way* as a product of primes.

The fallacy is that even if $n = ab$ and a and b have unique factorizations,
it is still possible that $n = cd$ for different c and d,
producing a different factorization of n.

The argument is false, but the claim is true and is known as the *fundamental theorem of arithmetic* (proof soon).

10.2 The Division Algorithm

(An old name, not really an algorithm).

Theorem 10.2 (The division algorithm.) Let a, b be nonzero naturals. Then $\exists! q, r$ such that $0 \leq r < b$ and $a = qb + r$.

Proof. To prove this theorem, we need to prove both that some r exists and that this r is unique (this is typical of unique-existence proofs).
Fix a, b.

Existence: We use well ordering.
Let $S = \{r \in \mathbb{N} \mid (\exists q \in \mathbb{N}) a = qb + r\}$.

Example:
If $a = 13$ and $b = 3$, then $S = \{1, 4, 7, 10, 13\}$.

S is nonempty because $a = 0b + a$ implies $a \in S$.
It follows by well ordering that S contains a least element r_0 such that $a = q_0 b + r_0$.
We claim that this (q_0, r_0) pair has all the properties we need.

The only interesting thing to check is that $r_0 < b$.
We argue by contradiction: suppose $r_0 \geq b$.
Then $a = (q_0 + 1)b + (r_0 - b)$ and $r_0 - b \geq 0$,
which implies that $r_0 - b \in S$.
But this contradicts the assumption that r_0 is the least element of S.
Thus $r_0 < b$.

Uniqueness: Suppose $a = q_1 b + r_1 = q_2 b + r_2$.
Without loss of generality, suppose $r_1 > r_2$.
(What we mean is, if $r_1 < r_2$, then we can swap (q_1, r_1) with (q_2, r_2) to make the assumption true. It's a proof by cases, but doing both cases at once. Of course, we cannot have $r_1 = r_2$.)
Then subtract:
$(q_1 - q_2)b = (r_1 - r_2)$.
This means that $b \mid r_1 - r_2$.
By assumption, $0 \leq r_1, r_2 < b$.

So $0 \leq r_1 - r_2 < b$.
Therefore, $r_1 - r_2 = 0$.
So $q_1 - q_2 = 0$.

■

Definition 10.3 In the theorem above, q is the *quotient* and r is the *remainder*.

One other useful def:

Definition 10.4 If x and y are integers, where $y > 0$, then $x \bmod y$ is the remainder on division of x by y.

$x \bmod y$ is well defined by the division algorithm.
$0 \leq x \bmod y < y$.

10.3 Divisibility and Primes

Definition 10.5 If a and b are integers, $a \neq 0$, then a *divides* b (written $a \mid b$) if $(\exists c)$, c an integer, such that $b = ac$.
We say a *goes into* b (in elementary school), or b is *divisible* by a.

This means remainder 0 in division algorithm.

Makes sense even for negative integers, but we'll mainly think of positive integers.

Definition 10.6 If a is a natural number ≥ 2, then a is *prime* if the only natural numbers dividing a are 1 and a.

Definition 10.7 If a is a natural number ≥ 2, then a is *composite* if $a = bc$ for natural numbers $b, c \neq 1, a$.

Basic divisibility properties:

Theorem 10.8

1. If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$ and $a \mid (b - c)$.
2. If $a \mid b$ then $a \mid bc$ for any c.
3. If $a \mid b$ and $b \mid c$ then $a \mid c$.

Proof. (Of part 1) By definition of divisibility, $(\exists t)b = at$ and $(\exists s)c = as$.
Therefore, $b + c = at + as = a(t + s)$.
Therefore, by definition, $a \mid b + c$.

Parts 2 and 3 are similar (use 2 to prove 3). ∎

Already did some results about these notions – e.g., the division algorithm, and half of the Fundamental Theorem of Arithmetic:

Theorem 10.9 (Fundamental Theorem of Arithmetic)
Every natural number ≥ 1 has a factorization into primes.
Moreover, the factorization is unique (except for reordering the factors)

We proved existence using well ordering.
Later today, we will also prove uniqueness.

10.4 GCDs and LCMs

Remember Die Hard?
Why did 5 and 3 buckets work but 6 and 3 not?
What if buckets are 21 and 133, and goal is 13 gallons?

Definition 10.10 Let a, b be integers, not both 0. The *greatest common divisor* of a and b, written $\gcd ab$, is the largest natural number g such that $g \mid a$ and $g \mid b$.

We can prove GCD is well defined:
Can use well-ordering to prove that any nonempty finite set has a *largest* element.
The set of common divisors includes 1, and is finite because it is bounded above by either number.

Note GCD is always positive.

GCD of 24 and 36? 21 and 133?

It turns out that in Die Hard, buckets of size a, b always produce numbers divisible by $\gcd ab$.
Q: Can all such numbers in range $0, \ldots, \max(a, b)$ be produced?

Here is an interesting theorem that shows that a GCD is always expressible as a sum.

This theorem will be used for many purposes, e.g., to help in proving the uniqueness part of the Fundamental Theorem of Arithmetic.

Theorem 10.11 Let x and y be integers, not both 0.
Then $\gcd xy$ can be expressed in the form $ax + by$ for some integers a and b.

Examples: $x = 22$, $y = 16$
Then GCD is 2, express as $2 = 3(22) + (-4)(16)$. Note can use negative integers.

$x = -22$, $y = 16$
Again, GCD is 2, express as $2 = (-3)(-22) + (-4)(16)$. (The GCD is always nonnegative.)

Proof. This is a nontrivial proof. Until now, we've been looking at mechanics of poving things. This proof also requires some creativity.

Assume wlog that $x, y \geq 0$. (Otherwise, use negative coefficients as above.)
If either is 0, then the GCD is defined to be the other – unwind the definition, which is certainly expressible in this way.
So we're left with the case where x and y are both positive.

Let S be the set of positive integers expressible in the form $ax + by$, where a and b are integers.
Fix l to be the smallest element in S – exists by well-ordering, S is nonempty since $x \in S$.
Fix a and b such that $l = ax + by$.

As an example, let $x = 22$, and $y = 16$. S includes all numbers $ax + by$, for all integers x and y. It includes:

$$\begin{aligned} 0x + 1y &= 16 \\ 1x + (-1)y &= 8 \\ (-1)x + 2y &= 10 \\ 2x + (-2)y &= 12 \\ 3x + (-4)y &= 2 \end{aligned}$$

and many others. $l = 2$ turns out to be the smallest element in S. We claim l is what we want, *i.e.*, $\gcd xy$. We need to show:

1. l is a common divisor of x and y.

2. Any common divisor of x and y also divides l (so l is the greatest – if a number c goes into a positive number l, then $c \leq l$).

1. l is a common divisor of x and y.

   First show $l \mid x$.
   Write $x = ql + r$ by division algorithm, where $0 \leq r < l$.
   Then rewrite:

$$\begin{aligned} r &= x - ql \\ &= x - q(ax + by) \\ &= (1 - qa)x + (-qb)y \end{aligned}$$

   So r is also expressible in the sum form.
   We know $0 \leq r < l$.
   If r were positive, it would be in S, and would be smaller than l, which would contradict our choice of l as the smallest element of S.
   So it must be that $r = 0$.
   So $x = ql$, which says that $l \mid x$.

   Similarly, $l \mid y$.

2. Any common divisor of x and y also divides l.

   Easy: $c \mid x$ and $c \mid y$ imply $c \mid ax + by = l$.

■

The symmetric concept to GCD is LCM:

Definition 10.12 Let a, b be positive integers. The *least common multiple* of a and b, written $\operatorname{lcm} ab$, is the smallest natural number l such that $a \mid l$ and $b \mid l$.

Why do we know this is well defined?

10.5 Fundamental Theorem of Arithmetic

With the addition characterization we just proved for the GCD, we are finally ready to prove the Fundamental Theorem.

Recall statement. Already proved existence. So remains to prove uniqueness.
For this, use basic number theory lemmas:

Lemma 10.13 Let p be a prime, and a and b any integers. If $p \mid ab$, then $p \mid a$ or $p \mid b$.

Proof. If $p \mid a$, we are done, so assume not.
Then $\gcd pa = 1$.
(Because the only positive divisors of p are p and 1, and p doesn't divide a.)
So by the addition form above, can write 1 as a sum of the form $1 = sa + tp$, where s and t are integers.
Multiplying by b:
$b = sab + tpb$.
We have $p \mid sab$, since $p \mid ab$.
Also, $p \mid tpb$.
So $p \mid b$ (by Theorem 10.8). ∎

Corollary 10.14 Let p be a prime, and $a_1, \cdots, a_k$ any integers.
If $p \mid a_1 a_2 \cdots a_k$, then $(\exists i) p \mid a_i$.

Now prove uniqueness, in Fundamental Theorem:

Proof. (Uniqueness of prime decomposition) We argue by (strong) induction. The base case $n = 1$ is trivial – the product of no factors is defined to be 1.
INDUCTIVE STEP:
Suppose the claim true for all numbers less than n.
Now show for n.
Suppose for contradiction that there are two different factorizations, $n = p_1 p_2 \cdots p_k$ and $n = q_1 q_2 \cdots q_l$.
Assume that the primes in each factorization are written in nondecreasing order.
Then $p_1 \mid q_1 \cdots q_l$.
By corollary, $(\exists i) p_1 \mid q_i$.
But since q_i is prime, $p_1 = q_i$.
Divide both sides, and get two factorizations of n/p_1.
This contradicts our inductive hypothesis, so the theorem is proved. ∎

Unique factorization gives a method for computing GCD:
Factor both numbers, and then multiply out all their common factors (watch multiplicities).

Example: 100 vs. 120:
$100 = 2 \times 2 \times 5 \times 5$
$120 = 2 \times 2 \times 2 \times 3 \times 5$
Common factors (including multiplicities): 2, 2, 5.
Product $2 \times 2 \times 5 = 20$ is GCD.

Taking min number of occurrences of each prime.

10.6 Why do Proofs?

- Convince yourself that your reasoning is correct – that you haven't made mistakes.
- Convince others.
- A good proof has considerable structure, like a good program. Much of it can be reused for other related results (like a well-written program).
- A proof pins down *why* something is true, so you can know whether it will stay true when things change.
- A failure to prove something often reveals the reason it is false.

Program verification: proof that a program does the right thing.

- Pentium
- Airbus A300 (earily version of control software had a bug that caused plane to head straight down if a certain configuration was reached)
- Therac 25

In programming, the current state of the art is that people do not actually prove correctness of running programs. There has been a lot of research directed toward making this possible, but it hasn't yet gotten far enough to be practical. The problem is that the concepts that have to be reasoned about seem too low-level, complicated in their details.

In contrast, people do know how to model programs/systems abstractly, and prove correctness of the abstract versions. E.g., the *algorithm* used in a communication protocol can be described formally, and its correctness proved. But there is still a gap between what is verified and the actual code that runs on real systems – programmers implement algorithms in a rather ad hoc manner, and errors can creep in there. There isn't much available now to help programmers bridge this gap.

The high-level algorithm proofs are usually done as sketches. Some are very complicated, and some errors have been found in published proofs. There is also considerable research in actually doing complete algorithm proofs axiomatically, and having a computer tool (an "automatic theorem-prover") help in checking or even filling in the low-level steps. This sort of work is becoming practical.

Whether done by hand or with computer assistance, algorithm proofs are of great help in getting your programs right (and understanding why). I think this will be done much more in the future. It will become an important part of quality assurance for programs. It takes a lot of work to do it, and you have to understand formal logic proofs before you do it.

Other examples:

- Kaashoek's Orca system, for which Alan Fekete and I helped to verify the main algorithm (and found a significant logical error in the process of doing the proof).
- Current chip designs – companies like Intel, HP are now investing in verification people and computer tools.
- Communication protocols, AT&T.
- Practical verification efforts are also beginning for real-time process control systems, e.g., those systems that control nuclear power plants and airplanes. Still preliminary.

Program verification is a nice complement to testing – testing can often detect bugs quickly. Verification is harder, but it does help to find deeper logical mistakes. And when you're done, you know there are *no more* errors (which testing alone can never guarantee).

10.7 Tiling

Problem: What 2-dimensional planes can be tiled using only 1×3 and 2×2 tiles?
Rotations allowed.

Easy case: What about $1 \times n$ and $n \times 1$?
If and only if n is a multiple of 3.
Easy, LTTR to prove carefully.

Next case: $2 \times n$ (and $n \times 2$)
Claim any, if $n \geq 2$.
Show by strong induction:

BASE: 2×2, one tile.
BASE: 2×3, two tiles.

INDUCTIVE: $n \geq 4$.
If n even, then use one 2×2, what's left is $2 \times (n-2)$, where $n-2 \geq 2$. So by strong inductive hypothesis, can tile the rest.

If n odd, then $n \geq 5$. Then use two 1×3's to reduce to $2 \times (n-3)$, where $n-3 \geq 2$. Again, strong inductive hypothesis gives the rest.
Yet another case: $3 \times n$ (and ...)
Then all, obviously.

Finally: What about $m \times n$, $m, n \geq 4$?
Claim all.
Proof by strong induction on m (where $P(m)$ says "for all $n \geq 4$, can tile $m \times n$".

Cases:
m even, $m \geq 4$:
Then first two rows are $2 \times n$, $n \geq 4$, can be tiled by special case lemma.
Remove, get $(m-2) \times n$.
If $m-2=2$ then can do rest by special case lemma.
Else $m-2 \geq 4$, can do rest by inductive hypothesis.

m odd, $m \geq 4$:
So $m \geq 5$.
Then first three rows can be done, by special case lemma.
Remove, get $(m-3) \times n$.
If $m-3=2$ then can do rest by special case lemma.
Else $m-3 \geq 4$, can do rest by inductive hypothesis.