Induction Essentials

CSCI 136: Fall 2018
Handout 4
24 Sept

What is Mathematical Induction?

Consider the statement

$$\text{For every integer } n \geq 0, 0 + 1 + \ldots + n = \frac{n(n+1)}{2}$$

or, using *summation notation*

$$\text{For every integer } n \geq 0, \sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$

Here's a short proof* of this statement. Note that—ignoring the 0 term, which makes no contribution to the sum

$$2 * (0 + \ldots + n) = (0 + \ldots + n) + (n + \ldots + 0) = (n + \ldots + n)) = n(n+1)$$

Dividing both sides by 2 gives the desired result.

This single formula above represents an infinite collection of simpler propositions:

Proposition P_0: When $n = 0$, $0 = \frac{0(0+1)}{2}$

Proposition P_1: When $n = 1$, $0 + 1 = \frac{1(1+1)}{2}$

Proposition P_2: When $n = 2$, $0 + 1 + 2 = \frac{2(2+1)}{2}$

Proposition P_3: When $n = 3$, $0 + 1 + 2 + 3 = \frac{3(3+1)}{2}$

...and so on....

Mathematical induction is a proof technique that can be used to simultaneously establish the truth of infinitely many propositions. The idea is really quite simple. Given a sequence $P_1, \ldots, P_n, \ldots$ of propositions, we do the following:

Base Case Somehow prove that P_0 is true.

Induction Step Prove that, for all $n \geq 0$, *if* proposition P_n is true *then* P_{n+1} is true.

Why is this sufficient to establish the truth of all of the propositions? Well, the Base Case establishes the truth of proposition P_0. The Induction step establishes the proof of all of the remaining propositions: Since P_0 is true, P_1 must be true. But since P_1 is true, then P_2 is true. And so on....

Let's try this on the propositions above: For each $n \geq 0$, P_n is the proposition $0 + 1 + \ldots + n = \frac{n(n+1)}{2}$.

Base Case: Is P_0 true? Well if $n = 0$, the left hand summation is merely 0. The right hand quantity $n(n+1)/2$ is $0(0+1)/2 = 0$. Thus the base case is established.

Induction Step: Now let n be any integer greater than or equal to 0 and assume that for this n, P_n is true; that is, $0 + 1 + \ldots + n = \frac{n(n+1)}{2}$. Is it the case then that P_{n+1} must be true? Well, P_{n+1} says that $0 + 1 + \ldots + (n+1) = \frac{(n+1)(n+2)}{2}$. But

$$0 + 1 + \ldots + (n+1) = [0 + 1 + \ldots + n] + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)((n+1)+1)}{2}$$

because the truth of P_n allows us to replace, for this specific value n, the sum $0+1+\ldots+n$ with the quantity $\frac{n(n+1)}{2}$. So, yes, if P_n is true, then P_{n+1} is also true!

*Attributed to a 7-year-old Carl Friedrich Gauss (1777-1855)

This proof technique can require some getting used to but it is an incredibly powerful tool making the investment of time well worth the effort. Let's state the principle more formally:

Theorem 1 (The First Principle of Mathematical Induction)**.** *Let $P_0, P_1, \ldots, P_n, \ldots$ be a sequence of propositions, one for each integer $n \geq 0$. Suppose that*

- *P_0 is true, and that*
- *For every $n \geq 0$, if P_n is true, then P_{n+1} is true.*

Then all of the propositions $P_0, P_1, \ldots, P_n, \ldots$ are true.

Let's try using this technique on some additional problems....

Example
Prove that for every $n \geq 0, 1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$; that is,

$$\sum_{k=0}^{n} 2^k = 2^{n+1} - 1.$$

Solution. Here the propositions are $P_n : \sum_{k=0}^{n} 2^k = 2^{n+1} - 1$.

Base Case: Prove that P_0 is true. Just check the left- and right-hand sides of the equation:

LHS: $\sum_{k=0}^{0} 2^k = 1$.
RHS: $2^{0+1} - 1 = 2 - 1 = 1$. ✓

Induction Step: Prove that, for all $n \geq 0$, *if* proposition P_n is true *then* P_{n+1} is true.

Proceed as in the previous example

$$1 + 2 + 2^2 + \ldots + 2^{n+1} = [1 + 2 + 2^2 + \ldots + 2^n] + 2^{n+1} = (2^{n+1} - 1) + 2^{n+1} = 2^{n+2} - 1 = 2^{(n+1)+1} - 1,$$

the second equality made possible by the assumed truth of P_n. And that's all there is to it! □

We can make the proof even more compact (and, arguably, clearer) by using summation notation in the induction step:

$$\sum_{k=0}^{n+1} 2^k = (\sum_{k=0}^{n} 2^k) + 2^{n+1} = (2^{n+1} - 1) + 2^{n+1} = 2^{n+2} - 1 = 2^{(n+1)+1} - 1.$$

Let's do one more example. We'll also sneak in a few additional details about induction, the first being that the base case does not need to be $n = 0$. In fact, let's restate the principle of induction to formalize this point:

Theorem 2 (The First Principle of Mathematical Induction (Version 2.0))**.** *Let b be any integer and let $P_b, P_{b+1}, \ldots, P_n, \ldots$ be a sequence of propositions, one for each integer $n \geq b$. Suppose that*

- *P_b is true, and that*
- *For every $n \geq b$, if P_n is true, then P_{n+1} is true.*

Then all of the propositions $P_b, P_{b+1}, \ldots, P_n, \ldots$ are true.

As with the first version of the principle, we can see that this version is valid by noting that if P_b is true, then the second property implies that P_{b+1} is true; but the second property then allows us to conclude that P_{b+2} is true since P_{b+1} is true, and so on....

Example

Prove that, for each $n \geq 1, 1^3 + \ldots + n^3 = (1 + \ldots + n)^2$; that is:

$$\sum_{k=1}^{n} k^3 = (\sum_{k=1}^{n} k)^2.$$

Note that, using the result of our first example, $(\sum_{k=1}^{n} k)^2 = (n(n+1)/2)^2$, so we'll prove that

$$\sum_{k=1}^{n} k^3 = \left(\frac{n(n+1)}{2}\right)^2.$$

Solution. We'll now stop formally using the explicit "P_n" notation and we'll also more explicitly emphasize a particular part of the induction step.

Base Case ($n = 1$): The left hand side is $1^3 = 1$ and the right hand side is $1^2 = 1$. ✓

We present the induction step somewhat differently. We make the following *induction hypothesis*:

Induction Hypothesis: For some $n \geq 1, \sum_{k=1}^{n} k^3 = \left(\frac{n(n+1)}{2}\right)^2$.

We now use the assumed truth of the induction hypothesis to establish the *induction step*:

Induction Step: Assuming the induction hypothesis holds for some $n \geq 1$, show that it holds for $n + 1$; that is, show that

$$\sum_{k=1}^{n+1} k^3 = \left(\frac{(n+1)((n+1)+1)}{2}\right)^2.$$

Ok, let's do it. **Warning:** Slightly messy algebra ahead. Proceed with caution....

$$\begin{aligned}\sum_{k=1}^{n+1} k^3 = (\sum_{k=1}^{n} k^3) + (n+1)^3 = (\sum_{k=1}^{n} k)^2 + (n+1)^3 = \left(\frac{n(n+1)}{2}\right)^2 + (n+1)^3 = (n+1)^2 \frac{n^2 + 4(n+1)}{4} \\ = \frac{(n+1)^2(n+2)^2}{4} = \left(\frac{(n+1)((n+1)+1)}{2}\right)^2. \quad (1)\end{aligned}$$

□

To summarize, the (First) Principle of Mathematical Reduction (2.0) provides a method for establishing the truth of a sequence of propositions. To use the principle, we

- First establish the truth of the first proposition directly (the base case: proposition P_b)
- State the induction hypothesis; that is, the fact that we are going to assume that proposition P_n holds for *some arbitrary* $n \geq b$.
- Use the induction hypothesis to establish the truth of the $(n+1)^{st}$ proposition.

So far, we've only used induction on summation problems; this is just the tip of the iceberg of problems on which this method can be used. Here are some further examples.
Example Consider the sequence of integers defined as follows:

- $s_1 = 1$
- For each $n > 1, s_n = 2s_{n-1} + 1$

So, $s_1 = 1, s_2 = 2s_1 + 1 = 3, s_3 = 2s_2 + 1 = 7, \ldots$. Prove that, for all $n \geq 1, s_n = 2^n - 1$.

Solution. We'll use induction. The base case is $n = 1$: $s_1 = 1$ and $2^1 - 1 = 1$ ✓

The induction hypothesis is that, for some $n \geq 1$, $s_n = 2^n - 1$. We'll use it to show that $s_{n+1} = 2^{n+1} - 1$ (the induction step). Here's how:

$$s_{n+1} = 2s_n + 1 = \text{(by induction) } 2(2^n - 1) + 1 = 2^{n+1} - 2 + 1 = 2^{n+1} - 1.$$

Done! □

Let's do a more interesting example. First we need to present another variant on the principle of mathematical induction.

Theorem 3 (The First Principle of Mathematical Induction (Version 3.0))**.** *Let b be any integer and let $P_b, P_{b+1}, \ldots, P_n, \ldots$ be a sequence of propositions, one for each integer $n \geq b$. Suppose that*

- *P_b and P_{b+1} are true, and that*
- *For every $n \geq b + 1$, if P_{n-1} and P_n are true, then P_{n+1} is true.*

Then all of the propositions $P_b, P_{b+1}, \ldots, P_n, \ldots$ are true.

As with the first version of the principle, we can see that this version is valid by noting that if P_b and P_{b+1} are true, then the second property implies that P_{b+2} is true; but now since P_{b+1} and P_{b+2} are true, the second property then allows us to conclude that P_{b+3} is true, and so on....
Let's try to use this.

Example

The *Fibonacci Numbers* are defined as follows

- $F_0 = F_1 = 1$,
- For all $n \geq 2, F_n = F_{n-1} + F_{n-2}$

Here's a recursive method to compute the n^{th} Fibonacci number

```
        public static int fib(int n) {
                if (n == 0 || n == 1) return 1;
                else return fib(n  1) + fib(n  2);
}
```

Prove that in computing the n^{th} Fibonacci number F_n, the method `fib` is invoked at least F_n times.

Solution. We use Version 3.0 of the First Principle of Mathematical Induction. The base cases are F_0 and F_1. Using `fib(0)` to compute F_0 involves 1 call to `fib` and $F_0 = 1$. Similarly, using `fib(1)` to compute F_1 involves 1 call to `fib` and $F_1 = 1$. Thus `fib` is invoked at most F_0 times to compute F_0 and at least F_1 times to computer F_1.

Our induction hypothesis is that, for some $n \geq 1$, the property holds for $n - 1$ and for n; that is, `fib` is called at least F_{n-1} times in computing F_{n-1} and at least F_n ties in computing F_n. Now we need to show that given this assumption, Using `fib(n+1)` to compute F_{n+1} will require at least F_{n+1} calls to `fib`.

Since $n \geq 1$, we have $n+1 \geq 2$, so when `fib(n+1)` is invoked (there's 1 call to `fib`!), the else block is entered and `fib(n)` and `fib(n-1)` are invoked. But, by the induction hypothesis, `fib(n)` results in at least F_n invocations of `fib` and `fib(n-1)` results in at least F_{n-1} invocations of `fib`. This gives at least $1+F_n+F_{n-1} = 1+F_{n+1} > F_{n+1}$ invocations of `fib`, which is what we wanted to show.

□

This is cool—we used induction to show that a particular algorithm for computing F_n has to take at least F_n steps! By the way, one can also show by induction that F_n grows exponentially quickly, which implies that the recursive method for computing Fibonacci numbers is very inefficient (but elegant!).

As you may now suspect, there is nothing special about having two base cases (P_b and P_{b+1}); there are variants of this version of the P.M.I. for any fixed number of base cases.

Theorem 4 (The First Principle of Mathematical Induction (Version 4.0))**.** *Let b be any integer and let $P_b, P_{b+1}, \ldots, P_n, \ldots$ be a sequence of propositions, one for each integer $n \geq b$. Suppose that, for some fixed $k \geq 0$*

- *$P_b, P_{b+1}, \ldots, P_{b+k}$ are all true*†*, and that*
- *For every $n \geq b+k$, if $P_n, P_{n-1}, \ldots, P_{n-k}$ are true*‡*, then P_{n+1} is true.*

Then all of the propositions $P_b, P_{b+1}, \ldots, P_n, \ldots$ are true.

There are two small differences here

- We allow multiple consecutive base cases: $P_b, \ldots, P_{b+k}$ for some fixed k, and
- To prove the truth of proposition P_{n+1}, our induction hypothesis is that the $k+1$ previous propositions are true.

The justification of the validity of this variant of the principle of induction is similar to those for the earlier versions. We directly establish the truth of $P_b, P_{b+1}, \ldots, P_{b+k}$ somehow. Then, using the second property above, we can conclude that P_{b+k+1} is true. But now $P_{b+1}, P_{b+2}, \ldots, P_{b+k+1}$ are true, so P_{b+k+2} must be true, and so on.

We'll introduce other important variants of the P.M.I. later in the semester, but for now these variants should be sufficient for establishing the validity of a wide range of properties for some of the data structures and algorithms we'll be discussing.

†$k+1$ base cases!

‡The previous $k+1$ propositions.