Jack's Own Introduction to Mathematical Induction

The Marbles

Imagine an infinite row of marbles, each marble being numbered: marble #1, marble #2, marble #3, etc. I am going to tell you two things about this row of marbles, and then I am going to ask you a question.

I will tell you this:

- Every marble other than the first is the same color as the preceding marble
- The first marble is green

Now here is the question:

What color is marble number 1,346,285?

No fair reading on until you have your answer!

(Jeopardy theme tune in the background)

If you answered green, you are correct. In fact, you probably realized that all of the marbles are green. (How did you know that?)

The Postage Stamp Problem

I claim that every amount of postage of 8 cents or larger can be built from 3-cent and 5-cent stamps. (We will not worry about how to fit the required stamps on the envelope.)

Let's start with 8 cents. That's easy: $8 = 5 + 3$.
Take a minute and write down solutions for 9, 10, 11, and 12 cents. Your solutions may be nice, but they don't tell us a thing about what to do for postage of $81.24. We need a more sophisticated strategy.

I claim that, if we have a solution for N cents (N being bigger than 8), we can modify that solution to get N + 1 cents! Here's how. Our solution for N cents has to contain *either* at least three 3-cent stamps *or* at least one 5-cent stamp. (Why?)

> Case 1: There are three 3-cent stamps in the solution. Remove three 3-cent stamps and replace them with two 5-cent stamps. We have just increased the postage by 1 cent!
>
> Case 2: There is at least one 5-cent stamp in the solution. Remove it and replace it with two 3-cent stamps. We have just increased the postage by 1 cent!

What to do about $81.24? Easy: start with our solution for eight cents, and then put the above "Case" structure in a "for" loop that executes 8,116 times.

```
// Program to solve the stamp problem for $81.24
// Let num3 be the number of 3-cent stamps
// Let num5 be the number of 5-cent stamps
int j, num3 = 1, num5 = 1; // That makes 8 cents
for (j = 8; j < 8124; j++) {
   // Here is the "case" structure (just a conditional)
   if (num3 >= 3) {num3 = num3 - 3; num5 = num5 + 2;}
   else           {num3 = num3 + 2; num5 = num5 - 1;}
   // End of the "case" structure
   System.out.println("You get " + (3*num3 + 5*num5) +
      " cents by using " + num3 + " 3-cent stamps and " +
      num5 + " 5-cent stamps.");
 }  // End of "for" loop
```

Of course, in this way, we can find a solution for any amount of postage by changing the upper limit on the "for" loop – at least until the program bombs on an integer overflow error.

What do the Marbles and the Postage Stamps have in common?

In each case, we have four steps:

(a) We identified a Boolean function P defined on the positive integers
(b) We identified a starting point s.
(c) We showed that P(s) is true. (This is often called the *base case*).
(d) We showed that if $n >= s$, then if P(n) is true, then P($n + 1$) is true.

> This step is often called the *inductive step*. P(n) is called *the inductive hypothesis*. P($n + 1$) is called the *inductive target*. The hard work in this step is finding a way to get from the inductive hypothesis to the inductive target.
>
> (*Inductive hypothesis* is a standard term. *Inductive target* is a non-standard term of my own invention. There is no standard term, but I find it helpful to have a term for use in this class.)

We then concluded that P is true from the starting point s onwards.

In the marbles case:

(a) P(n) is the statement "Marble number n is green".
(b) The starting point is 1.
(c) The first marble is green.
(d) The inductive hypothesis is
    "Marble number n is green".
and the inductive target is
    "Marble number $n + 1$ is green".

If marble number n is green, so is marble number $n + 1$, because the color of the each marble is the same as the color of its predecessor.

We concluded that all marbles are green.

In the postage stamps case:

(a) P(n) is the statement, “Postage of n cents can be built up from 3-cent stamps and 5-cent stamps”.

(b) The starting point is 8.

(c) We can get 8 cents by using one 3-cent stamp and one 5-cent stamp.

(d) The inductive hypothesis is

“Postage of n cents can be built up from 3-cent stamps and 5-cent stamps”.

and the inductive target is

“Postage of $n + 1$ cents can be built up from 3-cent stamps and 5-cent stamps”.

We saw that if we can get n cents postage out of 3's and 5's, then we can get $n + 1$ cents postage out of 3's and 5's using the procedure described above.

We concluded that we can get any postage from 8 cents onward by using only 3's and 5's.

That's all there is to it! All that remains is more examples. In any example, typically the only hard part is the fourth bullet: proving that $P(n) \rightarrow P(n + 1)$.

Another Example: Show that if $n >= 1$, then $5^n - 1$ is divisible by 4

(a) What is the Boolean function?

P(n) is the statement: “$5^n - 1$ is divisible by 4”

(b) What is the starting point?

The starting point is $s = 1$.

(c) (Base case) Is P(1) true?

Yes, because P(1) is the statement “$5^1 - 1$ is divisible by 4”.

This is true because $5^1 - 1 = 4$, and 4 is divisible by 4

(d) (Inductive step)

The inductive hypothesis is: $5^n - 1$ is divisible by 4.

The inductive target is: $5^{n+1} - 1$ is divisible by 4

So we have to show that, if $5^n - 1$ is divisible by 4, then $5^{n+1} - 1$ is divisible by 4.

This will take a little work. We have to refer to the definition of "divisible by". The statement “$5^n - 1$ is divisible by 4” means that there exists an integer x such that $5^n - 1 = 4x$.

From this, we have to deduce that $5^{n+1} - 1$ is divisible by 4.

$$\begin{aligned} 5^{n+1} - 1 &= 5^n * 5 - 1 \\ &= (4x + 1) * 5 - 1 \quad \text{(Why?)} \\ &= 20x + 5 - 1 \\ &= 20x + 4 \\ &= 4(5x + 1) \end{aligned}$$

and we have shown that $5^{n+1} - 1$ is divisible by 4, as required.

Conclusion: If $n >= 1$, then $5^n - 1$ is divisible by 4

One last example, that’s a bit different in strategy

You have known for a long time that every integer greater than 1 is either prime, or can be written as a product of primes. But have you ever seen a formal proof of this fact? Probably not.

Let’s start by checking out several cases, to see if we can discern a pattern:

2 is prime.
3 is prime.
4 is not prime, but it is a product of primes because it is equal to 2*2.
5 is prime.
6 is not prime, but it is a product of primes because it is equal to 2*3.
7 is prime.
8 is not prime, but it is equal to 2*4, and 2 is a prime, and 4 is a product of primes, therefore 8 is a product of primes.
9 is not prime, but it is a product of primes because it is equal to 3*3.
…etc…
24 is not prime, but it is equal to 6*4, and 6 and 4 are products of primes, so 24 is a product of primes.
…etc…etc…

You get the picture. You can fill in a few more of these cases yourself. You could even write a computer program to find prime factorizations of all integers up until your program crashes on integer overflow.

In general, suppose n is some big integer, and we want to show that n is a prime or a product of primes, and suppose that we have carried out the above drill all the way up to $n - 1$. n is either prime or it is not prime. If n is prime, we are done. If n is not prime, then it can be written in the form $n = x*y$, where x and y are smaller than n. But since x and y are smaller than n, we already know that x and y are each either prime or a product of primes, therefore n is a product of primes. This completes our proof.

In this example, we had:

(a) The Boolean function is: P(n) = "n is prime or can be written as a product of primes"
(b) The starting point is 2.
(c) The base case P(2) is true because 2 is prime.
(d) The inductive step is: Show that if $n - 1$ is a prime or a product of primes, then n is prime or a product of primes.

What's different here?

In our first three examples, to prove that P(n) is true, we relied on our having already proven P($n - 1$). In the last example, we had to rely on our having already proven our statement for *all* smaller numbers from 2 to $n - 1$, because we don't know how n will factor. In this case, the inductive hypothesis is: *All* of the integers from 2 up to and including $n - 1$ are either primes or products of primes.

This is a slightly more sophisticated strategy than we used in our first three examples. The strategy in our first three examples is called *weak induction*, and the strategy in the last example is called *strong induction*.

Cool, you say, but what does that have to do with computer science?

Mathematical induction is a basic tool for estimating the time and space requirements for a program or system as the workload increases. You will encounter this in Analysis of Algorithms and other advanced CMSC courses.

In fact, most applications of induction you will encounter in advanced courses will use strong induction. The reason is that if you break a problem down into smaller problems, you can't predict how much smaller the smaller problems will be.