You will see many examples in this course and upcoming courses of algorithms for solving various problems. It many cases, it will be obvious that the algorithms do what you think. However, it many cases it will not be obvious, and one needs to argue very convincingly (or "prove") why the algorithms are correct. To do so, and to understand the arguments of others, we need to learn about how such proofs work.

In particular, the next core topic in the course is *recursion*. We will look at a number of recursive algorithm and analyze how long they take to run. Recursion can be a bit confusing when one first learns about it. One way to understand recursion is to relate it to a proof technique in mathematics called *mathematical induction*. In a nutshell, you can use mathematical induction to prove that recursive algorithms are correct. Today I will introduce the basics of mathematical induction, and in the following few lectures we'll look at some recursive algorithms.

Before I introduce induction, let me give an example of a convincing proof of a statement you have seen many times before:.

$$1 + 2 + 3 + \cdots + (n-1) + n = \frac{n(n+1)}{2}.$$

One trick for showing this statement is true is to add up two copies of the left hand side, one forward and one backward:

$$1 + 2 + 3 + \cdots + (n-1) + n$$
$$n + (n-1) + ... + 3 + 2 + 1.$$

Then pair up terms:

$$1 + n, 2 + (n-1), 3 + (n-2), ...(n-1) + 2, n + 1$$

and note that each pair sums to $n+1$ and there are n pairs, which gives $(n+1) * n$. We then divide by 2 because we added up two copies. This proves the statement above, namely that: for all $n \geq 0$

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}.$$

Mathematical induction

Mathematical induction is a different type of proof technique than the one I just showed. Mathematical induction requires that you know in advance what you are trying to prove. Suppose, for example, I give the above equation

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

and I ask you to prove that it is true. How would you do that (other than using the construction I gave above) ?

Mathematical induction is a technique for proving statements about positive integers. We have some proposition (or "predicate") $P(n)$ which is either true or false for each n. We want to prove that: "for all $n \geq n_0$, $P(n)$ is true". Here n_0 is some constant that we state explicitly. In the above example, this constant is 1, but sometimes we have some other constant that is greater than 1.

A proof by *mathematical induction* has two parts, and one needs to prove both parts.

1. a *base case*: the statement $P(n)$ is true for $n = n_0$.
2. *induction step*: for any $k \geq n_0$, *if* $P(k)$ is true, *then* $P(k+1)$ must also be true.

   When we talk about $P(k)$ in step 2, we refer to it as the "induction hypothesis". Note that $P(k)$ is just $P(n)$ with $n = k$, *i.e.* it is the same proposition, but we are using parameter k instead of n to emphasize that we're in the context of proving the induction step.

The logic of a proof by mathematical induction goes like this. Let's say we can show that the base case and induction step both hold. Then, $P(n)$ is true for the base case $n = n_0$, and the induction step implies that $P(n)$ is true for $n = n_0 + 1$, and applying the induction step again implies that the the statement is true for $n = n_0 + 2$, and so on forever for all $n \geq n_0$.

Example 1

Statement: for all $n \geq 1$,

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

Proof: The base case, $n_0 = 1$, is true, since

$$1 = \frac{1 \cdot (1+1)}{2}.$$

We next prove the induction step. For any $k \geq 1$, we assume $P(k)$ is true and show that $P(k+1)$ must therefore also be true.

$$\begin{aligned}
\sum_{i=1}^{k+1} i &= (\sum_{i=1}^{k} i) + (k+1) \\
&= \frac{k(k+1)}{2} + (k+1), \text{ by induction hypothesis that P(k) is true} \\
&= (k+1)(\frac{k}{2} + 1) \\
&= \frac{1}{2}(k+1)(k+2)
\end{aligned}$$

which proves the induction step. The proof is complete, since we have proven the base case and the induction step.

Example 2

The next example is also not useful in itself, but I will use it in the following example. (In mathematics, this is called a lemma.) I am giving you this example because we will see something similar when we get to the formal definition of $O(\ )$ in a few weeks.

Statement: for all $n \geq 3$,

$$2n + 1 < 2^n.$$

Proof: The base case $n_0 = 3$ is easy to prove, i.e. $7 < 8$. (Note that the base case $n_0 = 2$ would not be correct, nor would $n_0 = 1$. That's why we chose $n_0 = 3$.)

Next we prove the induction step. Let k be any integer such that $k \geq 3$. We hypothesize that $P(k)$ is true and show that it would follow that $P(k+1)$ is also true. Note that $P(k)$ is the inequality $2k+1 < 2^k$.

$$\begin{aligned} 2(k+1)+1 &= 2k+3 \\ &= (2k+1)+2 \\ &< 2^k+2, \quad \text{by induction hypothesis that } P(k) \text{ is true} \\ &< 2^k+2^k, \quad \text{since } 2<2^k, \text{when } k \geq 2 \\ &= 2^{k+1}\ . \end{aligned}$$

Note that the induction step holds for $k \geq 2$, whereas the base case did not hold for $k = 2(or 1)$. This is why the statement used $n_0 = 3$.

Another point: You might be asking yourself, how did I know to use the inequality $2 < 2^k$? The answer is that I knew what inequality I eventually wanted to have, namely $P(k+1 < 2^{k+1})$ and so as long as I didn't exceed that inequality, I was ok.

Example 3

Statement: For all $n \geq 5$, $n^2 < 2^n$.

Proof: The base case $n_0 = 5$ is easy to prove, i.e. $25 < 32$.
Next we prove the induction step. The induction hypothesis $P(k)$ is the statement "$k^2 < 2^k$." We show that if $P(k)$ is true, then $P(k+1)$ must also be true.

$$\begin{aligned} P(k) &= (k+1)^2 \\ &= k^2+2k+1 \\ &< 2^k+2k+1, \quad \text{by induction hypothesis} \\ &< 2^k+2^k, \text{ if } k \geq 3, \quad \text{from Example 3} \\ &= 2^{k+1} \end{aligned}$$

which proves the induction step.

Note that the base case choice is crucial here. The statement $P(b)$ is just not true for $n = 0, 1, 2, 3, 4$. Also, the induction step is value for a larger range of n, namely $n \geq 3$, which is no problem.

Example 4: upper bound on Fibonacci numbers

Consider the Fibonnacci[1] sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... where

$$F(0) = 0, \quad F(1) = 1,$$

and, for all $n > 2$, we define $F(n)$ by

$$F(n) \equiv F(n-1) + F(n-2).$$

[1] `http://en.wikipedia.org/wiki/Fibonacci_number`

Statement:

$$\text{for all } n > 0, \quad F(n) < 2^n.$$

Proof:

The statement $P(k)$ is "F(k) $< 2^k$". We take the base case to be two values of n, namely $n_0 = 0, 1$. By definition, $F(0) = 0, F(1) = 1$ and so $P(k)$ is true for both base cases since $F(0) = 0 < 2^0$ and $F(1) = 1 < 2^1$.

The induction hypothesis $P(k)$ is that $F(k) < 2^k$. For the induction step, we assume that $P(k)$ is true and we show this implies $P(k+1)$ must also be true.

$$\begin{aligned} F(k+1) &\equiv F(k) + F(k-1) \\ &< 2^k + 2^{k-1} \quad \text{by induction hypothesis} \\ &< 2^k + 2^k \\ &= 2^{k+1} \end{aligned}$$

and so the induction step is proven, and we are done.

Insertion sort algorithm is correct

Mathematical induction also can be used to prove that certain algorithms are correct. (The parameter n might be the size of the problem.) Here I'll discuss an algorithm, namely insertion sort, which you should already believe is correct and you should have a good intuition of why it is correct. I'll sketch out how the proof by induction roughly goes.[2]]

Recall the main idea of insertion sort: at some intermediate step of the algorithm, we assume that the k elements (indices 0 to $k-1$) at the front of the list are sorted. We then take the element at index k and insert it into its proper place with respect to the k sorted elements. By definition, we now have $k+1$ elements at the front of the list and these elements are sorted.

This uses an induction type argument, since a list with one element is obviously sorted (base case). The induction hypothesis would be that the first k elements are sorted. The induction step would be to show that the algorithm then yields a list of $k+1$ sorted elements.

```
insertion_sort( list ){
   for k = 1 to n - 1 {
      elementK =  list[k]
      i = k
      while (i > 0) and (list[i - 1] > elementK ){
         list[i] =  list[i - 1]
         i  = i -1
      }
      list[i] = elementK
   }
}
```

[2] There is a more formal way to prove such algorithms are correct, using something called *loop invariants*. But this topic is too subtle and advanced to be presented in COMP 250, in my opinion.

To formalize it a bit more, we would need to state $P(n)$, e.g. "At the start of the `for` loop with index `k`, the elements `list[0],... list[k-1]` contains the same set of elements as the original list did (in this positions), and these elements are now sorted.

The base case is `k=1` and simply says that when the first pass starts, the `list[0]` is the first element of the original list. We can consider it sorted since a single element is obviously sorted.

To prove the induction step, we need to show that if $P(k)$ is true, then $P(k+1)$ is true. For this, we need to ensure that the body of the `for` loop when index is k ensures that $P(k+1)$ holds at the start of pass `k+1` (or at the very end of pass `k`). Proving the induction step requires verifying that the body of the `for loop` does what I just wrote.[3]

Once the base case and induction step are proven, we can conclude that $P(n)$ is true, when n is the size of the size. Thus, the algorithm exits the loop and terminates with a sorted list containing the original elements.

My goal in this example was to give you a flavour of how mathematical induction could be used to prove formally that an algorithm is correct. (I believe that you already used induction intuitively to understand why the algorithm is correct. If not, then what *was* your intuition of why that algorithm was correct?) In the next few lectures, we'll look at a different type of algorithm, called a recursive algorithm. Again, we'll use an induction intuition (and sometimes formal argument) to see why the algorithms are correct.

[3]Verifying that the body of a loop does what it is supposed to is sometimes non-trivial, and there are formal proof methods for doing so. I am not providing such details here, since this is a separate topic. My goal rather is to point out what needs to be proven.