Induction Proofs

for cmps 102, spring 2004

1 Introduction to Induction

This handout describes why an induction proof works, provides the correct layout for a proof by induction, and gives several examples of correct inductions. In addition, Section 4 gives several common errors in induction proofs and describes why they are wrong.

To understand why induction works, it may be useful to consider the difference between a "straight-line program" and a "recursive program". For example, a straight-line program to compute 5! might work as follows:

```
total := 1;
total := total * 2;
total := total * 3;
total := total * 4;
total := total * 5;
```

while a recursive program might look more like:

```
int factorial(int n)
{
  int total;

  if (n==1) then
    {
      total := 1;
      return(total);
    }
  else
    {
      total := factorial(n-1) * n;
      return(total);
    }
}
```

where 5! is computed by calling `factorial(5)`. Although the recursive program is actually longer for $n = 5$, you can easily see that it saves a lot of writing when $n = 100,000$, or even when $n = 50$. Note that both the straight-line program and the recursive program make *exactly* the same assignments to total, the recursive program is just a way of describing these assignments more generally. In fact, we can describe the recursive algorithm pictorially as follows.
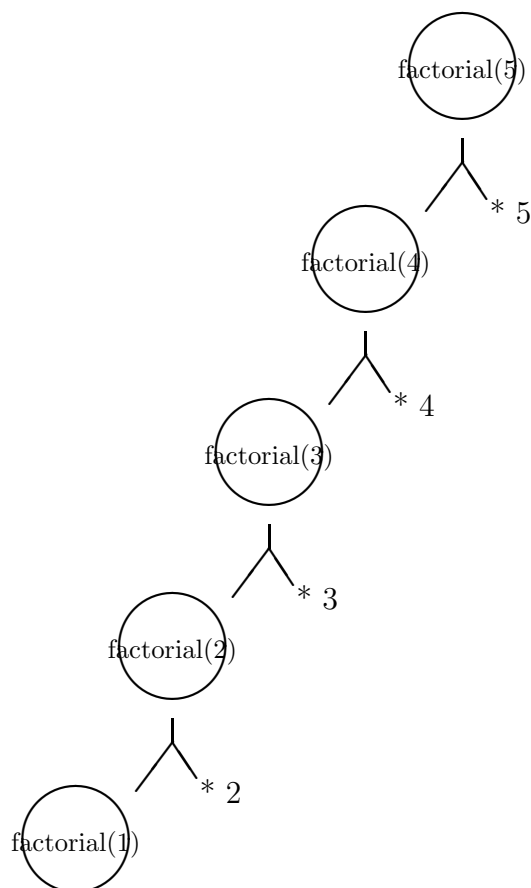
Induction works in almost the same way. Many proofs can be viewed as "straight-line proofs" where the chain of reasoning starts with a fact and proceeds using other facts to reach the desired conclusion. However, just like a single straight-line program cannot compute different factorials, proving many statements that start with "for all positive integers ..." or "for all integers greater than k, ..." can be very difficult with only a "straight-line proof." Just like the recursive factorial program represents a collection of different straight-line programs (one for each positive value of n), an inductive proof represents an infinite number of "straight-line proofs," one for each suitable value of n. Thus one is allowed to use a correct inductive proof to conclude that that the desired property holds for all suitable values of n.

In the factorial programs we moved from one value of n to the next by multiplying `factorial(n-1)` by n. In inductive proofs, the connection between the different values of n can be much more subtle. Therefore, the key to any inductive proof is the various *inductive hypotheses*, the "things" that get done for each value of n. Just like $n!$ has a different value for different n, we use a different inductive hypothesis for each value of n. I use "$P(n)$" to denote the particular inductive hypothesis for the value n. In fact, each $P(n)$ is a statement that could be either true or false, such as "All complete binary trees of height n have $2^{n+1} - 1$ nodes", or "If a binary tree has n degree-2 nodes then it has $n + 1$ leaves," or "$\sum_{i=1}^{n} i = n(n + 1)/2$."

In this class we do induction over *structures*, like graphs and trees. This is related to induction over the integers, but requires a "size" measure on the structures. Basically, you

stratify the set of all structures into subsets based on the size of the structures. Typical size measures are number of nodes or height for trees, and number of nodes (or edges) for graphs. Usually the definition of the kind of structure gives a hint as to what to use as the size measure, "large" structures often incorporate "smaller" structures.

The inductive proof includes a *base case* to prove that for some concrete value k (usually $k = 0$ or $k = 1$), the statement $P(k)$ is true. It is certainly permissible (and sometimes necessary) to include several base cases – proofs of $P(k)$ for several different values of k (see Section 4). A base case is like the `total := 1;` assignments in the factorial problem, they establish the needed fact to "base" the induction on a firm foundation. The recursive part of the factorial program is like the inductive step. Just like the recursive part generates many assignments to `total`, the inductive step generates many sub-proofs. Just like the assignments to `total` are spliced together to compute whatever $n!$ we want from the `total := 1;` assignment, the inductive step allows us to splice together a proof of $P(n)$ from the base cases for whatever large value of n we want by replicating and specializing the inductive step the appropriate number of times. Although the inductive proof contains explicit proofs of $P(n)$ only when n is (one of) the base case(s), it can be viewed as a "template" for "generating" proofs of $P(n)$ for whatever n-values are desired. That is why inductive proofs allow us to conclude that $P(n)$ is true for all large values of n. You don't need to write down all these proofs, just say enough so that anyone (with enough paper) can write down whichever $P(n)$ proof they need.

Important note: Both the inductive step and the base cases are *proofs* and must adhere to the proof rules. Although each of these should start the with a description of what is to be shown (which is *not* an established fact), the proof part **must** start with a fact, and continue with facts, until the desired conclusion is reached. Each fact should be easily justified and verified, either by referencing a previously established fact, a stated assumption, a well-known theorem, or simple algebra. If you want to use a complicated fact that doesn't fit into one of these categories, call the fact a *claim*, write a separate proof of the claim, and then continue with your induction.

2 Format of an Induction Proof

First, identify the statement (theorem) to be proven, and then identify what the family of properties $P(n)$ are. The theorem should be equivalent to "for all $n \geq n_0$, $P(n)$ is true" for some particular value n_0 (usually 0 or 1).

The actual proof consists of two steps: the base case(s) and the inductive step.

- The base case is a (usually straight-line) proof proving that $P(n_0)$ is true.

- The inductive step is a proof that for all $n \geq n_0$, the property $P(n)$ implies $P(n+1)$ (i.e. that if $P(n)$ is true, then so is $P(n+1)$). To show this, pick an arbitrary $k \geq n_0$, assume that $P(k)$ is true for this particular value of k, and show that $P(k+1)$ is also true. The statement $P(k)$ is the *induction hypothesis*.

Note that I changed notation and use k in the inductive step instead of n. I find that this change emphasizes that k stands for a single (but arbitrary) value whereas the n's are usually

quantified by a "for all". I have found that the change to k in the inductive step helps many students avoid mistakes, but if you wish to use n's in the inductive step, that is O.K. too.

Since the inductive step is done for an arbitrary k (nothing can be assumed about k except that it is an integer at least n_0) the inductive step shows that $P(n)$ implies $P(n+1)$ is true for all integers $n \geq n_0$. Now, like an infinite line of dominos, we have that $P(n_0)$ is true, and thus $P(n_0+1)$ is true, and thus $P(n_0+2)$ is true, and so on forever. That is why an inductive proof allows us to conclude that for all $n \geq n_0$, $P(n)$ is true.

There is an alternative form for the inductive step,

- Pick an arbitrary $k > n_0$, assume that $P(k-1)$ is true, and show that $P(k)$ is also true. In this form the statement $P(k-1)$ is the *induction hypothesis*.

The two differ only by a reparameterization of k. Both of these two forms are called "the first principle of mathematical induction" or "weak induction".

There is also a third form for the inductive step, for "strong induction":

- Pick an arbitrary $k \geq n_0$, assume that $P(i)$ is true for all $n_0 \leq i \leq k$, and show that $P(k+1)$ is also true.

This form can also be reparameterized as

- Pick an arbitrary $k > n_0$, assume that $P(i)$ is true for all $n_0 \leq i < k$, and show that $P(k)$ is also true.

The last two are called strong induction because you are making a stronger assumption (assuming that more properties are true in the inductive step).

3 Examples of Induction Proof

1. For the first example we will prove the geometric formula:
   Theorem: For all integers $n \geq 0$ and all real $x \neq 1$,

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1} \ .$$

For this theorem, we let $P(n)$ be the statement:

$$\text{for all real } x \neq 1, \ \sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1} \ .$$

Note that the theorem is equivalent to the compound statement:

for all $n \geq 0$, statement $P(n)$ is true.

This indicates that 0 is the appropriate base case value.

Proof: By induction on n using the $P(n)$'s defined above.

4

Base Case: Prove that $\sum_{i=0}^{0} x^i = \frac{x^1-1}{x-1}$ (i.e. that $P(0)$ is true).

$$\sum_{i=0}^{0} x^i = x^0 = 1 = \frac{x-1}{x-1}$$

as $x \neq 1$, completing the proof of $P(0)$.

Inductive Step: Let $k > 0$, assume that $\sum_{i=0}^{k-1} x^i = \frac{x^k-1}{x-1}$ (i.e. $P(k-1)$ is true), to prove that $\sum_{i=0}^{k} x^i = \frac{x^{k+1}-1}{x-1}$ (i.e. that $P(k)$ is true).

$$
\begin{aligned}
\sum_{i=0}^{k} x^i \quad &= x^k + \sum_{i=0}^{k-1} x^i \quad &&\text{algebra} \\
&= x^k + \frac{x^k-1}{x-1} \quad &&\text{ind. hypothesis} \\
&= \frac{x^k(x-1)+x^k-1}{x-1} \quad &&\text{algebra} \\
&= \frac{x^{k+1}-1}{x-1} \quad &&\text{algebra}
\end{aligned}
$$

showing $P(k)$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ $\square$

Note that both steps started with a statement of what was going to happen. These are comments, not facts that can be used in the proof, however the inductive hypothesis $P(k-1)$ can be used as a fact in the inductive step. Another important idea is that the inductive step starts with something large (the summation in $P(k)$ in this case), finds something smaller (the summation in $P(k-1)$), uses the inductive hypothesis on the smaller thing, and combine back up to show $P(k)$. Most inductive steps work in this way.

2. Recall that an extended binary tree contains (A) just a single external node, or (B) is an internal node (the root) connected to two disjoint subtrees, which are themselves extended binary trees.

Note the recursion in the definition, this should be exploited in your proof as well. Also, the number of internal nodes is a natural "size measure" for extended binary trees. (The total number of nodes can be used as well).

Finally, the proof exploits the following fact about trees: the set of nodes in any tree is the root (if any) plus the disjoint union of the nodes in the subtrees. Thus we can count nodes in a tree by counting up the nodes in the subtrees and adding one for the root. Listing the important

definitions and/or less obvious facts needed in your proofs before the theorem is generally a good idea.

Theorem: In any extended binary tree, the number of external nodes is one greater than the number of internal nodes.

Note that there is no obvious choice for the statements $P(n)$, so our proof begins by identifying the ones we will use.

Proof: By induction on number of internal nodes.

For all $n \geq 0$, let $P(n)$ be the statement: "all extended binary trees having exactly n internal nodes have $n + 1$ external nodes."

Note: since every extended binary tree has 0 or more internal nodes, the theorem is equivalent to: for all $n \geq 0, P(n)$ is true. For this proof we will need to use strong induction.

Base Case: Show $P(0)$. Every extended binary tree with zero internal nodes is formed by case (A) of the definition, and thus consists of just a single external node. Therefore, every extended binary tree with 0 internal nodes has exactly 1 external node, and $P(0)$ is true.

Inductive Step: Assume $k > 0$ and $P(j)$ holds for all $0 \leq j < k$ in order to show that $P(k)$ also holds.

Let T be an arbitrary extended binary tree with k internal nodes. Since $k > 0$, T's root is an internal node connected to two extended binary tree T_l and T_r (case (B) of the definition). Since the root is in neither sub tree (and all nodes in the subtrees are in T), each of T_l and T_r has less than k internal nodes. Let n_l and n_r be the number of internal nodes in T_l and T_r respectively, so $0 \leq n_l < k$ and $0 \leq n_r < k$. Using the inductive hypotheses $P(n_l)$ and $P(n_r)$, T_l and T_r have $n_l + 1$ and $n_r + 1$ external nodes, respectively. The root is an internal node, so T has $k = n_l + n_r + 1$ internal nodes, similarly, adding up the external nodes in the subtrees we get that T has $n_l + n_r + 2 = k + 1$ external nodes. Therefore, arbitrary trees with k internal nodes have one more external node than internal nodes, and $P(k)$ is shown. $\square$

4 Common Errors in Induction Proof

1. Base Case

   You must prove the base case because this is the foundation of the induction proof. Without a base case, the following inductive step has no meaning. One common error is to forget to prove the base case. Another one is proving an insufficient number of base cases. Although one base case is usually enough, many problems, like the Fibonacci numbers, $F_n = F_{n-1} + F_{n-2}$, require two base case values in order to use the recurrence equation. E.g, if you only prove the base case for $i = 0$, you might wind up using $F_1 = F_0 + F_{-1}$ in the inductive step, but this is not part of the definition of the Fibbonacci sequence. Similarly, if you have proven the base case only for $i = 1$, then you can use $F_2 = F_1 + F_0$, but you have not shown that F_0 meets the required form. A classic example of this is the "all horses" proof below:

   Theorem?: In any set of horses, all horses in the set are the same color.

   Proof??: By induction on the number of horses in the set, for each $n \geq 1$, the property $P(n)$ is "in any set of n horses, all horses in the set are the same color."

   Base Case: $n = 1$. If a set contains only one horse, then all horses in the set are the same color.

   Inductive Step: Let $n \geq 1$ and show that $P(n)$ implies $P(n + 1)$. consider any set S of $n + 1$ horses, so $S = \{h_1, h_2, \ldots, h_{n+1}\}$. Consider $S' = S - \{h_1\} = \{h_2, \ldots, h_{n+1}\}$. and $S'' = S - \{h_{n+1}\} = \{h_1, h_2, \ldots, h_n\}$. Since S' is a set of n horses, by the inductive hypothesis all n horses in S' are the same color, so h_{n+1} is the same color as h_2. Similary, all n horses in S'' are the same color Since $S = S'' + \{h_{n+1}\}$, all horses in S'' are the same color, and h_{n+1} is the same color as h_2, all horses in S are the same color. $\square$??

2. Conditions in $P(n)$.

   Don't forget the premise or preconditions of the $P(n)$'s. he following is an example that uses a flawed induction to get a wrong conclusion.

   Theorem: If n is an even number and $n \geq 2$, then n is a power of two.

   Proof?: By induction on the even number n. Let $P(k)$ be the statement "if k is an even number and $k > 2$, then $k = 2^i$, where i is a natural number."

   Base Case Show $P(2)$. $2 = 2^1$, so 2 is a power of 2.

   Inductive Step??: Assume k is a number greater than 2, that $P(j)$ holds for every $2 \leq j < k$, and show that $P(k)$ also holds.

Case 1: k is odd, and there is nothing to show.

Case 2: k is even, so $k \geq 4$. Since $k \geq 4$ is an even number, $k = 2\ell$ with $2 \leq \ell < k$. Therefore we can use the inductive hypothesis $P(\ell)$, showing that $\ell = 2^i$ for some integer i. Since $k = 2 * \ell$, $k = 2^{i+1}$ and is also a power of 2, so $P(k)$ holds. □??

The above theorem and proof are incorrect since, for example, we can not write 6 as a power of 2 although 6 is a even number larger than 2. The error comes from the inductive step when $P(\ell)$ is used. $P(\ell)$ only states that $\ell = 2^i$ if ℓ is even, and that was not verified in the proof.

3. Constructing a large object

When you do the inductive step, always go from (an arbitrary) large thing back to a smaller thing in order to apply the inductive hypothesis. If a proof starts with a small size thing and then constructs a large thing, it is likely to be flawed. E.g, if you already have a binary tree with n-1 nodes, when you want to prove property of the binary tree with n nodes, One way is to assume you have a binary tree with n nodes, cut the root you can get two sub trees which has nodes less than n. This will cover all possibilities. If you directly add one node to the n-1 nodes trees, it is quite difficult to say how you add the node. If you can not clearly state all the cases, your proof is invalid. Here is a concrete example of how a proof can be flawed in this way.

Theorem?: For all n, all n-node binary trees have height $n - 1$.

Proof??: By induction on n. For each $n \geq 1$, let $P(n)$ be the statement "all n node binary trees have height $n - 1$".

Base Case: $n = 1$. Every 1-node binary tree consists of just a root, and has height 0. Therefore $P(1)$ is true.

Inductive Step: Let $n \geq 1$ and show that $P(n)$ implies $P(n + 1)$. Let T be an arbitrary n-node binary tree. Create the $n+1$-node binary tree T' by adding a new leaf to T hanging off the bottom level. The height of T' is thus one greater than the height of T. Furthermore, by the inductive hypothesis, the height of T is $n - 1$. Therefore T' has $n + 1$ nodes and

height $n - 1 + 1 = n$, showing $P(n + 1)$. $\qquad\qquad\qquad\qquad$ □??

The flaw in the above proof is not easy to find. The problem is that $P(n + 1)$ says that *all* $n + 1$-node binary trees have height n. The inductive step constructs only one $n + 1$-node binary tree. The correct approach for the inductive step is to start with an arbitrary $n + 1$-node binary tree, find a smaller tree inside of it, and apply the inductive hypothesis on the smaller tree.

4. Only examples, no proof.

   Some students give particular examples for the problem, and examples (generally) do not make a proof. For example, you can not specify the exact number of nodes for a tree to finish the proof because for induction proof, the problem size ordinarily is infinite. Whatever examples you gave, it is only a portion of the possible problem. You must use the inductive step to construct the infinite loop to finish the proof.

5   Formatting tips

When writing a proof by induction always make sure the following are clear:

- What the $P(n)$'s are

- What is being assumed (the inductive assumption) and what is being proven in the inductive step.

In any proof, make sure your substitutions are easy to follow, and any notation you use is clearly defined.

Most of the sample proofs in this document were written "paragraph style". This style emphasizes the reasoning and rational for each algebraic step. Any long sequence of algebra should be written "equation style" with one equation per line.