

Induction Proofs

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 3 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:

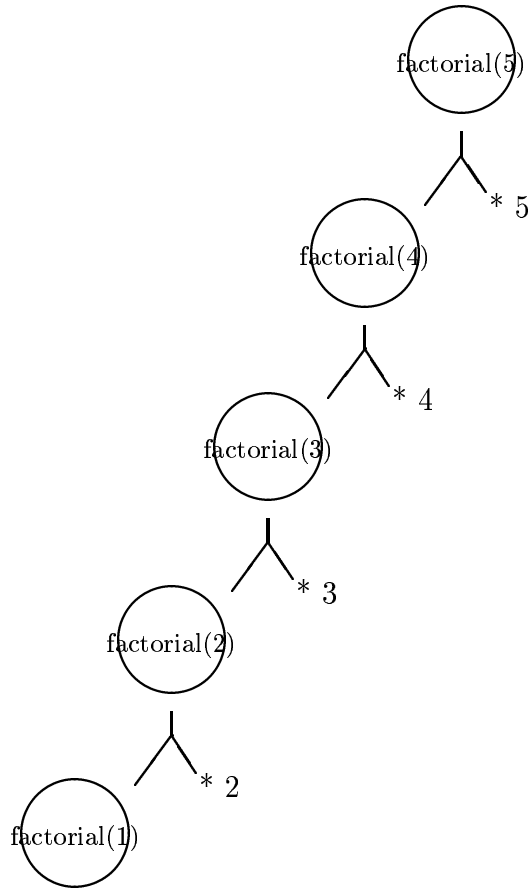
```
total := 1;
total := total * 2;
total := total * 3;
total := total * 4;
total := total * 5;
```

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 a statement like “for all positive integers ...” or “for all integers greater than k , ...” can be very difficult when only a “straight-line proof” is allowed. 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 “ $IH(n)$ ” to denote the particular inductive hypothesis for the value n . In fact, each $IH(n)$ 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 “for all n , the $\sum_{i=1}^n i = n(n + 1)/2$.”

In this class we do induction over *structures*, like graphs and trees. This is related to

induction over the integers (as in the summation example above), 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 $IH(k)$ is true. It is certainly permissible (and sometimes necessary) to include several base cases – proofs of $IH(k)$ for several different values of k . We will see an example of this in Section 2. 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 $IH(n)$ from the base cases for whatever large value of n we want. Although the inductive proof contains explicit proofs of $IH(n)$ only when n is (one of) the base case(s), it can be viewed as a “template” for “generating” proofs of $IH(n)$ for whatever n -values are desired. That is why inductive proofs allow us to conclude that $IH(n)$ 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 $IH(n)$ 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 with a description of what is to be shown, 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 Examples of Induction Proof

1. The first example is the final problem in the midterm. For this problem you need only one base case.

Recall from the midterm that an extended binary tree contains just a single external node, or 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: the set of nodes in any tree is the root (if any) plus the union of the nodes in the subtrees. Thus we can count nodes by counting up the nodes in the subtrees and adding in the root.

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

Proof: By induction on number of internal nodes.

IH(k): All extended binary trees with k internal nodes have $k + 1$ external nodes.

Base Case: Show $IH(0)$. Extended binary trees with zero internal nodes are just a single external node, so the number of internal nodes is one less than number of external nodes in this case.

Inductive Step: Assume $J > 0$, and $IH(k)$ holds for $0 \leq k < J$, we need to show $IH(J)$ also holds here.

Let T be an arbitrary extended binary tree with J internal nodes. Since $J > 0$, T 's root is an internal node connected to two extended binary tree T_l and T_r . Since the root is in neither sub tree (and all nodes in the subtrees are in T), T_l and T_r each has less than J internal nodes. Let n_l and n_r be the number of internal nodes in T_l and T_r . Using $IH(n_l)$ and $IH(n_r)$, 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 $J = n_l + n_r + 1$ internal nodes, similarly, adding up the external nodes in the subtrees we get that T has $n_l + n_r + 2 = J + 1$ external nodes. Therefore, arbitrary trees with j internal nodes have one more external node than internal nodes, and $IH(J)$ is shown.

QED

2. The second example is about Fibonacci number. The definition is follows: $F_0 = 0$, $F_1 = 1$, $F_i = F_{i-1} + F_{i-2}$, for $i \geq 2$. The general form is $F_i = (\phi^i - \psi^i)/\sqrt{5}$, where $\phi = (1 + \sqrt{5})/2$, $\psi = (1 - \sqrt{5})/2$. This induction will require two base cases.

Theorem: The i -th Fibonacci number satisfies $F_i = (\phi^i - \psi^i)/\sqrt{5}$.

Proof: By induction on integer number $n \geq 0$.

IH(k): The theorem holds for $k \geq 0$, i.e. $F_k = (\phi^k - \psi^k)/\sqrt{5}$.

Base Case: Show $IH(0)$ and $IH(1)$.

For $IH(0)$, $F_0 = 0, (\phi^0 - \psi^0)/\sqrt{5} = (1 - 1)/\sqrt{5} = 0$. $IH(0)$ holds here.

For $IH(1)$, $F_1 = 1, (\phi^1 - \psi^1)/\sqrt{5} = ((1 + \sqrt{5})/2 - (1 - \sqrt{5})/2)/\sqrt{5} = 1$, $IH(1)$ holds here.

Inductive Step: Assume $n > 1$, and $IH(k)$ holds for $0 \leq k < n$, we need to show $IH(n)$ also holds here.

By the definition of Fibonacci number, $F_n = F_{n-1} + F_{n-2}$. Since both $n - 1$ and $n - 2$ less than n (and at least 0), we can use $IH(n - 1)$ and $IH(n - 2)$.

$$\begin{aligned} F_n &= \frac{\phi^{n-1} - \psi^{n-1}}{\sqrt{5}} + \frac{\phi^{n-2} - \psi^{n-2}}{\sqrt{5}} \\ &= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}}{\sqrt{5}} + \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-2}}{\sqrt{5}} \end{aligned}$$

$$\begin{aligned}
&= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-2}\left(\frac{1+\sqrt{5}}{2} + 1\right) - \left(\frac{1-\sqrt{5}}{2}\right)^{n-2}\left(\frac{1-\sqrt{5}}{2} + 1\right)}{\sqrt{5}} \\
&= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-2}\left(\frac{1+\sqrt{5}}{2}\right)^2 - \left(\frac{1-\sqrt{5}}{2}\right)^{n-2}\left(\frac{1-\sqrt{5}}{2}\right)^2}{\sqrt{5}} \\
&= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} \\
&= \frac{\phi^n - \psi^n}{\sqrt{5}}
\end{aligned}$$

Therefore, $IH(n)$ holds.

QED

3 Common Errors in Induction Proof

1. Base Case

For induction proof, you must prove the base case because this is the foundation of the induction proof. Without a base case, the following proof 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 number example, $F_n = F_{n-1} + F_{n-2}$, 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 Fibonacci 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.

2. Premise

Whenever to prove the theorem, don't forget the premise of the theorem. The following is an example that uses the induction to get a wrong conclusion.

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

Proof: By induction on the even number n

$IH(k)$: If k is an even number and $k > 2$, then $k = 2^i$, where i is a natural number

Base Case Since 4 is the smallest even number larger than 2, we need to show $IH(4)$. $4 = 2^2$, $IH(4)$ holds here.

Inductive Step: Assume n is a even number and $n > 2$, $IH(k)$ holds for any even number k and $k < n$, we need to show $IH(n)$ holds here. Since n is a even number, so $n = 2k$, where $k < n$. Using the hypothesis we can write $k = 2^i$, so $2k = 2 \times 2^i = 2^{i+1}$, and $IH(n)$ holds.

QED

The above induction proof is incorrect since, for example, we can not write 6 as a power of 2 although 6 is an even number larger than 2. The error comes from the inductive step when we reduce $n = 2 \times k$, we directly use hypothesis on k but forget to verify that k is an even number larger than 2.

3. Assumption

In the inductive step, don't forget to state the assumptions and what you are showing in the inductive step.

4. Split problem size

When you do the inductive step, always you need to go from large problem size back to smaller problem size plus some answers you already knew. Splitting to smaller size is because you need to use the assumption. When you go back from smaller size to larger one, be careful. You need to make this step general. 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.

5. Only examples, no proof.

Some students give exact examples for the problem. This is not considered as the 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.