CSE 102, Introduction to Analysis of Algorithms
Notes on Proof by Induction

Zack Jorquera

April 8, 2023

1 Introduction

This document will serve as a reference for how to use proof by induction and what we expect in your own proofs on the homework. It is not required that you use the formatting seen in this document, but if you are still familiarizing yourself with the proof by induction technique, then it is highly recommended that you set up your proofs as we do in this document to help structure your proofs.

2 Proof By Induction

It is often useful to prove that a statement holds for any natural number. For example, we may want to prove that for any $n \in \mathbb{N}$, the following holds (note, we are using $\mathbb{N} = \{0, 1, 2, \dots\}$)

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

This sort of statement is impossible to verify for every $n \in \mathbb{N}$ due to the infinite nature of the natural numbers. We could alternatively try to prove this by considering any arbitrary $n \in \mathbb{N}$ and proving that it is true directly. However, this can be very challenging to prove and ignores the statement's structure. This structure is one of the smaller or earlier cases implying similar properties for subsequent cases. Namely, we have that $\sum_{i=0}^{n+1} i = (n+1) + \sum_{i=1}^{n} i$. Instead, we can show that some initial cases satisfy the desired statement, which serves as a starting point for a chain of implications that will, in turn, prove the statement for the whole domain.

Intuitively, we can think of this as a sequence of dominos. We have to knock over the first domino (proving the initial/base cases), and then if the dominos are close enough together, each domino will knock over the next in the sequence (the chain of implications).

More specifically, an inductive proof has three components: The base cases, the inductive hypothesis, and the inductive step.

1. **Bases Cases -** We first verify that the statement holds for the minimal cases (the ones that start the chain of implications).
2. **Inductive Hypothesis -** We want to show that *if* some earlier cases satisfy the statement, *then* so do the subsequent cases. The inductive hypothesis is the *if* part of this if-then statement. We do this by assuming that the statement holds for some or all earlier cases.
3. **Inductive Step -** We use the inductive hypothesis to prove that the subsequent cases also hold. This is the *then* part of the if-then statement.

As an example, we can then use this proof technique to prove the following proposition.

Proposition 1. *For all $n \in \mathbb{N}$, we have that:*

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof. We prove this by induction over $n \in \mathbb{N}$.

Base Case: We verify that the proposition holds for $n = 0$. We have that $\sum_{i=0}^{0} i = 0$ which is equal to $\frac{0\cdot(0+1)}{2} = 0$. And thus, the proposition holds for $n = 0$.

Inductive Hypothesis: Assume there exists a $k \geq 0$ such that the proposition holds:

$$\sum_{i=0}^{n} i = \frac{k(k+1)}{2}$$

Inductive Step: Consider the sum of integers from 0 to $k + 1$, for which we have the following:

$$\begin{aligned}
\sum_{i=0}^{k+1} i &= (k+1) + \sum_{i=0}^{k} i \\
&= (k+1) + \frac{k(k+1)}{2} \qquad \text{(by inductive hypothesis)} \\
&= \frac{2(k+2) + k(k+1)}{2} \\
&= \frac{k^2 + 3k + 2}{2} \\
&= \frac{(k+1)(k+2)}{2}
\end{aligned}$$

And thus, by induction, the proposition holds for all $n \in \mathbb{N}$. □

A brief note on writing the inductive hypothesis: A common mistake is to beg the question. That is, if we were to assume that the entire proposition is true. As an incorrect example, I could have said, "Suppose that for all $k \geq 0$, the proposition is true." This, however, is incorrect. The goal is to prove the proposition, and so assuming that it is already true defeats the purpose. Instead, we want to fix a k that is at least as large as our largest base case (in this case $n = 0$) and assume that the proposition holds for this specific fixed k.

The above proof by induction is an example of *weak induction*, the most basic form of induction. In short, weak induction is when we only have a single base case, and the inductive hypothesis only assumes the statement is true for some fixed k. Another form of induction that is widely used is *strong induction*. In short, strong induction is more relaxed. For the inductive hypothesis, we can assume the statement holds for all cases $m \leq k$, for some fixed k, and not just for k alone. And in turn, we may need to use more base cases.

Remark 1. Somewhat counterintuitively, strong induction and weak induction are equally as powerful. That is, any proof using strong induction can be converted into a proof using weak induction and vice versa. However, in practice, it may be easier to use one over the other.

As a side note, many of the loop invariant proofs we will see in this class use weak induction. That is, they assume only that the loop invariant is true at the start of the loop and then prove that it is true at the end of the loop (or at the start of the next iteration, depending on how you phrase it). In comparison, many proofs for recursive algorithms use strong induction. That is, they assume that the algorithm works for subproblems of smaller sizes.

We can then give some examples of using strong induction (you might have seen these in week 1 discussion section).

Proposition 2. *Let F_n for $n \geq 0$ denote the nth Fibonacci number, which is defined as $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$ for $n \geq 1$. And let L_n for $n \geq 0$ denote the nth Lucas number, which is defined as $L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2}$ for $n \geq 1$. We have that for all $n \geq 1$,*

$$L_n = F_{n-1} + F_{n+1}$$

For convenience, we write the first few numbers for both sequences:

n	0	1	2	3	4	5	6	7
F_n	0	1	1	2	3	5	8	13
L_n	2	1	3	4	7	11	18	29

Proof. We prove this by strong induction over $n \geq 1$.

Base Case: We verify that the proposition holds for $n = 1$ and $n = 2$. First, we can verify that $L_1 = 1$ and $F_0 + F_2 = 0 + 1 = 1$. Next, we can verify that $L_2 = 3$ and $F_1 + F_3 = 1 + 2 = 3$. And thus, the proposition holds for $n = 1$ and $n = 2$.

Inductive Hypothesis: Assume there exists a $k \geq 2$ such that the proposition holds for k and $k - 1$:

$$L_k = F_{k-1} + F_{k+1}$$

$$L_{k-1} = F_{k-2} + F_k$$

Inductive Step: Consider the $(k + 1)$th Lucas number, for which we have the following:

$$\begin{aligned} L_{k+1} &= L_k + L_{k-1} && \text{(by definition of the Lucas numbers)} \\ &= F_{k-1} + F_{k+1} + F_{k-2} + F_k && \text{(by inductive hypothesis)} \\ &= F_k + F_{k+2} && \text{(by definition of the Fibonacci numbers)} \end{aligned}$$

And thus, by induction, the proposition holds for all $n \geq 1$. □

Remark 2. In this proof, we used two base cases and assumed the proposition held for both $n = k$ and $n = k - 1$, but why is this? You will notice that in the inductive step, we applied the inductive hypothesis to both L_k and L_{k-1}. If we only assumed it held for $n = k$, then we would be left with $L_{k+1} = F_{k-1} + F_{k+1} + L_{k-1}$, which wouldn't help us prove the proposition for $n = k + 1$. Next, for the base cases, if we were only to have one base case, for $n = 1$, then the inductive hypothesis would have to consider some fixed $k \geq 1$. For the smallest value of $k = 1$, this would have us assume that $L_0 = F_{-1} + F_1$, which doesn't make sense as negative Fibonacci numbers haven't been defined in this proposition. As a general rule of thumb, the number of required base cases equals the number of assumptions you make in the inductive hypothesis (this isn't always true, but it often is).

Next, we will look at the tournament ranking problem. So far, all the proofs in this document have only concerned numerical equalities. However, this is an algorithms class, and so we want to prove things from the perspective of algorithms. The tournament ranking is a nice introduction to this.

Proposition 3 (The Tournament Ranking Problem)**.** *Consider a tennis tournament of $n \geq 1$ players, denoted by $\{P_1, P_2, \ldots, P_n\}$. For each pair of players, say $\{P_i, P_j\}$, they play a match where one player wins* *(there are no ties). We say $P_i \prec P_j$ if player P_j beats P_i in their match. Then for any outcome of the $\binom{n}{2}$ matches, there is always an ordering of the players:*

$$P_{i_1} \prec P_{i_2} \prec \cdots \prec P_{i_n}$$

Here, $i_1, i_2, \ldots, i_n$ denotes a permutation of $1, 2, \ldots, n$. Note, an ordering is not *transitive. That is for players $\{P_1, P_2, P_3\}$, $P_1 \prec P_2 \prec P_3$ does* not *imply $P_1 \prec P_3$, rather only that $P_1 \prec P_2$ and $P_2 \prec P_3$.*

Before we prove this, it is useful to give the idea of the proof. That is, for our tournament of n players, which, for notation we will call PLAYERS $= \{P_1, P_2, \ldots, P_n\}$, we fix player P_1 and consider all the players that lost to player P_1 and all the players that beat P_1. That is, we consider all the players that can go where the ??s are:

$$?? \prec P_1 \prec ??$$

Furthermore, the number of players that can go where the ??s are is strictly less than n as they can't include P_1. This defines a recursive structure of the problem. Namely, if we can find an ordering of these subsets of players and put them where the question marks are, we would get an ordering for all n players.

To make the proof easier, we will consider the base case to be when there are no players in the tournament. Even though we only need to prove this proposition for $n \geq 1$, we will find it to be much easier to prove it for $n \geq 0$. This may seem weird at first, but everything works out as far as the induction step is concerned.

Proof. We prove this by strong induction over the number of players in the tournament, $n \geq 0$.

Base Case: We verify that there exists an ordering for no player, $\emptyset$. The ordering is nothing.

Inductive Hypothesis: Assume there exists a $k \geq 0$ such that for any tournament of $0 \leq m \leq k$ players there is an ordering of the players.

Inductive Step: Consider a tournament of $k+1$ players and let PLAYERS $= \{P_1, P_2, \ldots, P_k, P_{k+1}\}$ be the set of the players. We then fix P_1 and consider the set of players that lost to P_1, which we will call L, and the set of players that beat P_1, which we will call W. These sets can be defined by

$$L = \{P \in \text{PLAYERS} \mid P \prec P_1\}$$

$$W = \{P \in \text{PLAYERS} \mid P_1 \prec P\}$$

Because $P_1 \notin L$ and $P_1 \notin W$ we have that $s = |L| \leq k$ and $t = |W| \leq k$. Furthermore, because no player can both lose and win to P_1 and every player had a match with P_1, they are disjoint, and their union is all players minus P_1. We can then use the induction hypothesis to get an ordering of the players in L and W (that is, we consider "sub-tournaments" of only the matches between the players in L and W). Let the orderings for L and W be given by the following, respectively:

$$P_{\ell_1} \prec P_{\ell_2} \prec \cdots \prec P_{\ell_s}$$

$$P_{w_1} \prec P_{w_2} \prec \cdots \prec P_{w_s}$$

Finally, we can construct the final ordering as follows. Note, if either L or W are the empty set, then we ignore their orderings and let P_1 be at one or both of the ends of the ordering.

$$P_{\ell_1} \prec \cdots \prec P_{\ell_s} \prec P_1 \prec P_{w_1} \prec \cdots \prec P_{w_s}$$

All that is left to check is that $P_{\ell_s} \prec P_1$ and $P_1 \prec P_{w_1}$. These follow from the fact that $P_{\ell_s} \in L$ and thus lost to P_1 and that $P_{w_1} \in W$ and thus beat P_1.

And thus, by induction, there is always an ordering for any tournament of $n \geq 1$ players. $\square$

Remark 3. In many ways, the inductive step can be seen as proving the correctness of a recursive algorithm for finding an ordering. This algorithm is a divide-and-conquer style strategy that can be roughly written in the following way: We first divide the problem into the losing and winning sets, then we find an ordering for them recursively, and finally, we combine them together to get an ordering for the whole set. Then proving the proposition is equivalent to proving the correctness of this algorithm (which is written in pseudo-code below). Note we store this ordering in a list data structure that supports list concatenation with the + operator.

```
procedure TournamentRanking(PLAYERS)
    if PLAYERS = $\emptyset$ then return [ ]
    else
        fix $P_1 \in$ PLAYERS
        $L \leftarrow \{P \in \text{PLAYERS} \mid P \prec P_1\}$
        $W \leftarrow \{P \in \text{PLAYERS} \mid P_1 \prec P\}$
        $L_{\text{ord}} \leftarrow$ TournamentRanking($L$)
        $W_{\text{ord}} \leftarrow$ TournamentRanking($W$)
    return $L_{\text{ord}} + [P_1] + W_{\text{ord}}$
```

3 Additional Reading

For a more in-depth resource, we recommend Richard Hammacks's *Book of Proof* (Chapter 10), which can be found here: `https://www.people.vcu.edu/~rhammack/BookOfProof/`.