Induction

1 Introduction

Induction is a powerful proof technique that is widely used in computer science and mathematics. It has many variations, and we shall look at some of them.

- Ordinary induction over $\mathbb{N}$.
- Course-of-values induction over $\mathbb{N}$.

2 Induction over $\mathbb{N}$

Imagine an infinite sequence of dominoes standing on an infinite table, with Domino $n+1$ standing just behind Domino n, and someone pushes Domino 0. Then Domino 0 falls, causing Domino 1 to fall, causing Domino 2 to fall, causing Domino 3 to fall... What about Domino $10^{10^{100}}$? It will eventually fall. Indeed it is obvious that *each domino will fall*.

This is the idea behind induction over $\mathbb{N}$. Let P be a property of natural numbers. Suppose that $P(0)$—this is called the *base case*. Suppose also that, for any natural number $\mathbb{N}$, the statement $P(n)$ implies $P(n+1)$—this is called the *inductive step*, and the hypothesis $P(n)$ is called the *inductive hypothesis*. From these two facts, we may conclude that *every* natural number satisfies P, even big ones like $10^{10^{100}}$.

Example. Let's prove $0 + \ldots + (n-1) = \frac{1}{2}(n-1)n$ by induction on $n \in \mathbb{N}$. Clearly this is true for $n = 0$, since the sum of no numbers is defined to be 0. Assuming it's true for n, let's show that it's true for $n+1$.

$$\begin{aligned}
0 + \cdots + ((n+1)-1) &= 0 + \cdots + (n-1) + n \\
&= \frac{1}{2}(n-1)n + n \quad \text{(by the inductive hypothesis)} \\
&= \frac{1}{2}n^2 - \frac{1}{2}n + n \\
&= \frac{1}{2}n^2 + \frac{1}{2}n \\
&= \frac{1}{2}n(n+1) \\
&= \frac{1}{2}((n+1)-1)(n+1) \quad \text{as required.}
\end{aligned}$$

Negative viewpoint This argument can be seen negatively. If the desired property doesn't hold for n, then $n > 0$, so it doesn't hold for $n - 1$, and $n - 1 > 0$, so it doesn't hold for $n - 2$, and so on. But these are natural numbers, so this can't continue forever. Contradiction!

3 Variations

Here are some variations. Let P be a property of natural numbers.

- Suppose we have proved that P holds for 0, 1 and 2, and also that, if it holds for n, $n+1$ and $n+2$, then it also holds for $n+3$. We now know that P holds for all natural numbers.
- Suppose we have proved that P holds for 1 and 3, and also that, if it holds for n and $n+2$, then it also holds for $n+4$. We now know that P holds for all odd natural numbers.
- Suppose we have proved that P holds for 1, and also that, if it holds for n, then it also holds for $2n$. We now know that P holds for every power of 2.

4 Course-of-values induction

When we give a proof by ordinary induction, the inductive step proves that $P(1)$ follows from $P(0)$, that $P(2)$ follows from $P(1)$, that $P(3)$ follows from $P(2)$, and so on. But surely, when proving $P(3)$, it should be acceptable to assume not just $P(2)$ but also $P(1)$ and $P(0)$. This thinking leads to *course-of-values induction* (also called "strong induction").

The principle is as follows. Let P be a property of natural numbers. Suppose that, for any natural number n, the statement $P(n)$ holds if P holds for all natural numbers less than n. (The latter assumption is called the *inductive hypothesis*.) This means that

- $P(0)$
- if $P(0)$, then $P(1)$
- if $P(0)$ and $P(1)$, then $P(2)$
- if $P(0)$ and $P(1)$ and $P(2)$, then $P(3)$
- etc.

From this fact, we may conclude that *every* natural number satisfies P, even big ones like $10^{10^{100}}$.

Example. The *merge sort* algorithm is the following recursively defined algorithm for sorting a list p.[1]

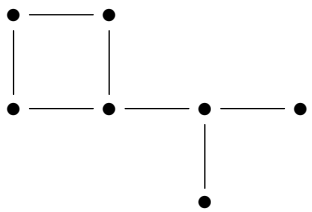
- If the length of p is 0 or 1, return p.

[1]The version given here is intended to return the sorted version of the list. The version for sorting an *array* with in-place update is slightly different, but the idea is the same.

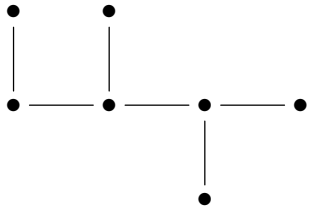
- If the length of p is $2k$ where $k > 0$, then sort the left part of length k, and sort the right part of length k, and merge the results.
- If the length of p is $2k + 1$ where $k > 0$, then sort the left part of length k, and sort the right part of length $k + 1$, and merge the results.

How do we know that this algorithm terminates, and returns a list that is a sorted version of p? By course-of-values induction on the length of the list. In each of the three cases, it is easy to see that the algorithm yields a sorted version of p, assuming that it works correctly on shorter lists.

Example. An undirected graph G is *connected* when it has at least one vertex and there is a path between any two vertices.[2] Show that, if G is connected, then $V(G) \leqslant E(G) + 1$, where $V(G)$ is the number of vertices and $E(G)$ the number of edges. For example, if G is



then $V(G) = E(G) = 7$, whereas if G is



then $V(G) = 7$ and $E(G) = 6$.

Our proof proceeds by induction on $E(G)$. If $E(G) = 0$, then there can only be one vertex, so the property holds. So suppose that $E(G) > 0$. Pick an edge e from x to y. Let H be G with the same vertices, but e removed.

- Suppose there's a path p from x to y in H. Then H is connected. (For any nodes z and w, take a path in G from z to w, then replace e in this path by p.) Since $E(H) = E(G) - 1$, we apply the inductive hypothesis to H giving $V(H) \leqslant E(H) + 1$. So

$$\begin{aligned} V(G) &= V(H) \\ &\leqslant E(H) + 1 \\ &= E(G) \\ &< E(G) + 1 \end{aligned}$$

[2] Some authors do not require the first condition. It makes no difference to this question.

- On the other hand, suppose there's no path from x to y in H. Then H consists of two connected components, the part H_x that's connected to x and the part H_y that's connected to y. (For any vertex z, there's a path in G from z to x with no cycles. It either lies entirely in H, in which case $z \in H_x$, or consists of a path in H from z to y followed by the edge e, in which case $z \in H_y$. We can't have both $z \in H_x$ and $z \in H_y$, as this would give a path from x to y.) Since H_x and H_y are connected and have fewer edges than G, we can apply the inductive hypothesis to them, giving $V(H_x) \leqslant E(H_x) + 1$, and $V(H_y) \leqslant E(H_y) + 1$. Thus

$$\begin{aligned} V(G) &= V(H_x) + V(H_y) \\ &\leqslant E(H_x) + E(H_y) + 2 \\ &= E(H) + 2 \\ &= E(G) + 1 \end{aligned}$$

  as required.

Notice that, in this example, we don't quote the inductive hypothesis at the start of the inductive step. Instead, we put some work into obtaining two smaller graphs, and only then do we apply the inductive hypothesis to each of them.

Negative viewpoint This argument can also be seen negatively. If the result doesn't hold for a connected graph G, then there's a smaller one G' that it also doesn't hold for, and therefore an even smaller one G'', and so on. But these are finite graphs, so this can't continue forever. Contradiction!