Induction and Correctness Proofs

Mathematical induction plays a prominent role in the analysis of algorithms. There are various reasons for this, but in our setting we in particular use mathematical induction to prove the *correctness of recursive algorithms*. In this setting, commonly a *simple induction* is not sufficient, and we need to use *strong induction*.

We will, nonetheless, use simple induction as a starting point. This also gives us the opportunity to discuss sums that are relevant to the analysis of algorithms. We then move on to strong induction and correctness proofs for recursive algorithms.

1. The principle of induction

Induction is most commonly used to prove a statement about natural numbers. Lets consider as example the statement $P(n)$:

$$\sum_{i=0}^{n} 1/2^i = 2 - 1/2^i .$$

We can easily check whether this statement is true for a couple of values n. For instance, $P(0)$ states

$$\sum_{i=0}^{0} 1/2^i = 1/2^0 = 1 = 2 - 1 = 2 - 1/2^0,$$

which is true. But also, for instance, $P(3)$,

$$\sum_{i=0}^{3} 1/2^i = 1/2^0 + 1/2^1 + 1/2^2 + 1/2^3 = 1 + 0.5 + 0.25 + 0.125 = 2 - 0.125 = 2 - 1/2^3$$

is true.

Knowing that $P(3)$ holds, we can add $1/2^4$ to $P(3)$ to prove that $P(4)$ holds:

$$\sum_{i=0}^{4} 1/2^i = \sum_{i=0}^{3} 1/2^i + 0.0625 = (2 - 0.125) + 0.0625 = 2 - 0.0625 = 2 - 1/2^4.$$

More generally, if we know that $P(n)$ holds, we can prove $P(n+1)$:

$$\sum_{i=0}^{n+1} 1/2^i = \sum_{i=0}^{n} 1/2^i + 1/2^{n+1} = (2 - 1/2^n) + 1/2^{n+1} = 2 - (1/2^n - 1/2^{n+1}) = 2 - (2/2^{n+1} - 1/2^{n+1}) = 2 - 1/2^{n+1}.$$

Since we know that $P(0)$ holds, we can use this to conclude that $P(n)$ also holds for $P(1)$, then also $P(2), P(3), \ldots$ Actually we can conclude that $P(n)$ holds for all n. The principle we are implicitely applying here is induction.

To summarize, given a statement $P(n)$ about a natural number n, if we can prove

1.) $P(0)$ and

2.) $P(n)$ implies $P(n+1)$,

then by the principle of induction we can conclude that $P(n)$ holds for all n.

It seems natural to ask why induction constitutes a valid proof. But this is actually a tricky question, since it brings us to deeper questions like: what is proof, and what are the natural numbers? The short answer is, that we simply assume that induction constitutes a valid proof.

An alternative answer is the following: Assume induction would not work, that is we have a statement $P(n)$ for which 1.) and 2.) holds but there is an n for which $P(n)$ does not hold. Now suppose n is the smallest natural number for which $P(n)$ does not hold. Because of 1.) we have $n>0$, which in particular means that $n-1$ is also a natural number. Since $n-1<n$, we know that $P(n-1)$ holds. But because of 2.) we can conclude that $P(n)$ also holds. A contradiction!. This answer is valid, but we are making some fundamental assumptions about the natural numbers in this argument. In short: induction makes sense and is useful, and therefore we are going to use it in our proofs.

2. Simple Induction

Induction in its basic form always uses the two ingredients 1.) and 2.) from above. It therefore makes sense to structure our induction proofs always in the same way. Sticking to the same structure also helps us to easily see that we didn't forget some important ingredient. Below is a possible structure. Note that the parts in **bold** depend on the specific statement and need to be filled in appropriately. In the following, "I.H." is the abbreviation for "induction hypothesis".

Proof: By induction on n we prove the following statement for all n:

$P(n)$: **blabla** n **blabla**.

Basis ($n=0$): **Show here that P(0) holds.** Thus, the statement holds for $n=0$.

Step ($n \rightarrow n+1$): Assume the statement $P(n)$ holds for n (I.H.). **Show that $P(n+1)$ holds (assuming that P(n) holds. You should point out the place where you use the induction hypothesis by "by induction hypothesis".** Thus, the statement also holds for $n+1$.

By induction we can conclude that the statement holds for all n. ▫

Note that ▫ is used here to mark the end of the proof. Let us try this structure on our previous $P(n)$. However, to make the statement more useful, we prove a generalization. Before reading the next proof you may want to try to come up with it yourself.

Note on choosing the basis: So far we always assumed that the statement that we want to prove holds for all $n \geq 0$. However, often enough we may want to prove a statement that only holds for $n \geq 1$. This can be easily achieved: We simple use $P(1)$ as basis. More generally, if a statement holds for $n \geq k$, we probably want to use $P(k)$ as basis. However, sometimes we have to be careful: some inductions, in particular some strong inductions, require several base cases; since this is a problem that is more common in strong inductions, we ignore this here for now, and discuss it later, when looking at strong inductions.

2.1. Geometric Series

Claim: $\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q}$ for all n and $q \neq 1$.

Proof: By induction on n we prove the following statement for all n:

$P(n)$: $\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q}$ for $q \neq 1$.

Basis ($n=0$): $\sum_{i=0}^{0} q^i = 1 = \frac{1-q^{0+1}}{1-q}$. Thus, the statement holds for $n=0$.

Step ($n \rightarrow n+1$): Assume the statement $P(n)$ holds for n (I.H.). Now, $\sum_{i=0}^{n+1} q^i = \left(\sum_{i=0}^{n} q^i\right) + q^{n+1} = \frac{1-q^{n+1}}{1-q} + q^{n+1}$ by induction hypothesis. Further, $\frac{1-q^{n+1}}{1-q} + q^{n+1} = \frac{1-q^{n+1}}{1-q} + \frac{q^{n+1}-q^{n+2}}{1-q} = \frac{1-q^{n+2}}{1-q}$. Thus, the statement also holds for $n+1$.

By induction we can conclude that the statement holds for all n. ▫

Note that the series $\sum_{i=0}^{n} q^i$ is called *geometric series*. It shows up, for instance, in the proof of the Master Theorem. In terms of its asympotics, for us it is relevant that

1.) for $0<q<1$, $\sum_{i=0}^{n} q^i = \Theta(1)$, and

2.) for $q>1$, $\sum_{i=0}^{n} q^i = \Theta(q^n)$.

Both of this follows from the claim above.

Induction provides us with a simple and clean way to prove the claim stated above. It is worth noting, that a non-inductive proof can be more insightful: by nature induction uses "small steps" and as such does not provide us with the "big picture". A more intuitive proof of the statement above for instance could be

\begin{align} & (1 + q^1 + q^2 + q^3 \ldots + q^n) (1 - q)\ = & (1-q) + q^1 (1 -q) + q^2 (1 - q) + q^3 (1-q) \ldots + q^n (1-q)\ = & 1 - q + q - q^2 + q^2 - q^3 \ldots + q^n - q^{n+1}\ = & 1 - q^{n+1}:. \end{align}

When q is the base of our favourite number system (e.g., $q=10$), we actually take the claim above as granted, as it is the same as saying, e.g. for n = 4, $ 1000 - 1 = 999$.

2.2. Arithmetic series

A series that shows up frequently when analyzing loops is the arithmetic series:

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

To prove this statement by induction we use the identical proof structure as before (try it yourself before reading on). There are two natural choices as a basis: $n=0$ or $n=1$, and both work equally well. When starting with $n=0$, note that the "empty" sum $\sum_{i=1}^{0} i$ evaluates to 0.

Proof: By induction on n we prove the following statement for all n:

$$P(n): \sum_{i=1}^{n} i = n(n+1)/2.$$

Basis ($n=0$): $\sum_{i=1}^{0} i = 0 = 0(0+1)/2$. Thus, the statement holds for $n=0$.

Step ($n \rightarrow n+1$): Assume the statement $P(n)$ holds for n (I.H.). Now, $\sum_{i=0}^{n+1} i = \left(\sum_{i=0}^{n} i\right) + n + 1 = n(n+1)/2 + n + 1$ by induction hypothesis. Further, $n(n+1)/2 + n + 1 = (n+1)(n/2+1) = (n+1)(n+2)/2$. Thus, the statement also holds for $n+1$.

By induction we can conclude that the statement holds for all n. ▫

As for the geometric series we additionally consider a non-inductive argument for the statement. In the sum $\sum_{i=1}^{n} i$ adding the first (1) and last summand gives $n+1$, the second and next-to-last again sum to $n+1$, and so on. If n is even, we get $n/2$ such pairs and the statement follows for this case. If n is odd, we get $n-1$ such pairs, but also have an un-paired $(n+1)/2$ left; again the statement follows.

For the purpose of this course we actually don't need the exact value of the sum. Essentially we only make use of the asymptotics:

$$\sum_{i=1}^{n} i = n(n+1)/2 = n^2/2 + n/2 = \Theta(n^2).$$

This sum occurs in an algorithm with a loop when the work done in iteration i is proportional to i. Examples of this are selection sort and, in the worst case, insertion sort.

2.3. Induction proofs & asymptotic notation

If in contrast the work per iteration would be proportional to i^2 we get a running time proportional to $\sum_{i=1}^{n} i^2$. Then we would want to prove $\sum_{i=1}^{n} i^2 = \Theta(n^3)$. Of course we can, as above, prove an exact bound. But let us assume we don't know the exact bound. If we would want to prove the statement by induction. We need to prove an upper bound of $O(n^3)$ and a lower bound of $\Omega(n^3)$. Let us do one of the two, that is, we prove $\sum_{i=1}^{n} i^2 = O(n^3)$.

Now first a word of warning: we should not use asymptotic notation in an induction. Why? Consider the following (wrong) induction step proving $\sum_{i=1}^{n} i^2 = O(n^2)$:

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

Of course the statement cannot be true if $\sum_{i=1}^{n} i^2 = \Theta(n^3)$. But what went wrong? If we want to prove a statement involving an $O()$ then we will have to find a *fixed* constant $c > 0$ such that the statement holds for all $n \geq n_0$. But if we use asymptotic notation in the induction step, then we allow us to pick a different constant for every n, which then no longer is a constant.

Instead of using asymptotic notation in the proof, we should formulate the statement $P(n)$ with a suitable constant c. Typically we don't know the constant upfront; we then simply start the proof with a generic c, and from the proof infer which constant works. Let us do this by example.

Proof: By induction on n we prove the following statement for all n:

$P(n)$: $\sum_{i=1}^{n} i^2 \leq c n^3$ for $c = \ldots$.

Basis ($n=0$): $\sum_{i=1}^{0} i^2 = 0 \leq c\, 0^3$ for any c.

Step ($n \rightarrow n+1$): Assume the statement $P(n)$ holds for n (I.H.). Now,
$\sum_{i=1}^{n+1} i^2 = \left(\sum_{i=1}^{n} i^2\right) + (n+1)^2 \leq c n^3 + (n+1)^2$ by induction hypothesis. Further, we have $c(n+1)^3 = c n^3 + 3 c n^2 + 3 c n + 1$. This is larger than $c n^3 + (n+1)^2 = c n^3 + n^2 + 2n + 1$ for $c \geq 1$ (*Note: We could have picked an even smaller c here, but we don't need to.*). Thus, the statement also holds for $n+1$.

By induction we can conclude that the statement holds for all n. ▫

After writing the proof we see that we have one condition on c, namely $c \geq 1$. We should pick one c that fulfils this condition, for instance, $c=1$. Thus, we fill in 1 for the $\ldots$ in the proof above, and are done.

Two remarks: Firstly, while the statement in this section serves us well to explain how to handle asymptotic notation in induction proofs, it is not actually a statement that we would want to prove by induction. Instead, we can easily get an upper bound by observing $\sum_{i=1}^{n} i^2 \leq \sum_{i=1}^{n} n^2 = n^3$. For the lower bound we have to argue slightly more carefully, $\sum_{i=1}^{n} i^2 \geq \sum_{i=\lceil n/2 \rceil}^{n} i^2 \geq \sum_{i=\lceil n/2 \rceil}^{n} (n/2)^2 \geq (n - \lceil n/2 \rceil + 1)(n/2)^2 \geq (n/2)^3 = n^3/8$. (*Notation:* $\lceil n/2 \rceil$ *means* $n/2$ *rounded up.*)

Secondly, the exact bound is $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$. Proving this bound might be a useful exercise. More examples to practice simple induction can, for instance, be found at https://en.wikibooks.org/wiki/Mathematical_Proof/Methods_of_Proof/Proof_by_Induction

2. Strong Induction

Let us consider the running time $T(n)$ of MergeSort on an array of size n. As reminder, here is code for MergeSort.

```
def merge_sort(A):
    n = len(A)
    if n>1:
        mid = n//2
        A1 = A[:mid]
        A2 = A[mid:]
        # recursive calls:
        merge_sort(A1)
        merge_sort(A2)
        # merge solutions
        i=0
        j=0
        while i<mid and mid+j<n:
            if A1[i]<A2[j]:
                A[i+j]=A1[i]
                i+=1
            else:
                A[i+j]=A2[j]
                j+=1

        while i<mid:
            A[i+j]=A1[i]
            i+=1
        while mid+j<n:
            A[i+j]=A2[j]
            j+=1

A = [5,2,8,6]
merge_sort(A)
print(A)
```

```
[2, 5, 6, 8]
```

The recurrence for the running time is $T(1)=d$ for some constant $d>0$; and for $n>1$: $T(n)=2T(n/2)+c$ for some constant $c>0$. We would like to argue $T(n)=O(n\log n)$.

Now with the Master Theorem at hand this is of course easy. But lets try to use the *substitution method*, which is essentially a fancy way of saying that we want to prove the statement by induction. We cannot (easily) use simple induction, since $T(n+1)$ is not based on $T(n)$. But coming back to the principle of induction, we can observe that when proving the statement $P(n+1)$, we in some sense did not make use all that we knew by then. When proving $P(n+1)$ we assumed that we already knew $P(n)$, which we knew because of $P(n-1)$, which we knew because of $P(n-2)$, which we knew because of… $P(0)$. So actually, when proving $P(n+1)$ we can just as well assume that $P(k)$ holds for all $k<n+1$. Thus, in the induction step, we can assume as induction hypothesis $P(k)$ for all $k<n+1$, and then prove $P(n+1)$ under this assumption. It is more convenient (and works just as well) to prove $P(n)$ based on the

induction hypothesis: $P(k)$ holds for all $k<n$.

This is called strong induction. We can re-use the structure from before with small changes:

Proof: By strong induction on n we prove the following statement for all n:

$P(n)$: **blabla** n **blabla**.

Basis ($n=0$): **Show here that P(0) holds.** Thus, the statement holds for $n=0$. **You may need more base cases.**

Step ($k<n\to n$): Assume the statement $P(k)$ holds for $k<n$ (I.H.). **Show that** $P(n)$ **holds (assuming that** $P(k)$ **holds for** $k<n$**. You should point out the place where you use the induction hypothesis by "by induction hypothesis".** Thus, the statement also holds for n.

By induction we can conclude that the statement holds for all n. ▫

Before using this to analyze the recurrence of MergeSort, let us take a closer look at the induction basis. In our example, we can read of which bases we should use from the recurrence: we had a special case for $n=1$, so this will be our base case. However, note that often recurrences are stated without base/special case. So then, we might simply get the recurrence stated as $T(n)=2T(n/2)+\Theta(n)$. **Nonetheless, we need to handle the base case.** We can read it off from the algorithm as such ($n=1$ is a special case in the algorithm).

But if we really only have the recurrence, we will simply need to assume a suitable base case. How can we see that we don't need additional base cases? For $n=2$ we base $T(n)$ on $T(n/2)=T(1)$. So with the basis $n=1$ this is handled. Also for $n>2$ we base $T(n)$ on $T(n/2)$ with $1\leq n/2<n$, so we always end up with a case that we have previously handled. In contrast, if we would analyze the recurrence $T(n)=T(n/6)+\Theta(n)$, the base case $n=1$ is not enough: For $n=2$ we have $n/6=0$, so this is not handled by the case case $n=1$. We should have as basis $n=1, n=2, \ldots, n=5$.

Now let us use the structure above to bound the running time of merge sort. This serves also as an additional example of proving an asymptotic bound.

Proof: By strong induction on n we prove the following statement for all n:

$P(n)$: $T(n)\leq c' n\log n+d$ with the constant d given in the recurrence, and with $c'=c+d$.

Basis ($n=1$): $T(1)=d\leq c' 1\log 1+d$. Thus, the statement holds for $n=1$.

Step ($k<n\rightarrow n$): Assume the statement $P(k)$ holds for $1\leq k<n$ (I.H.). Now, $T(n)=2T(n/2)+cn\leq 2(c'n/2\log(n/2)+d)+cn$ by I.H.. Further, $2(c'n/2\log(n/2)+d)+cn=c'n\log(n/2)+cn+2d=c'n\log n-c'n\log 2+cn+2d=c'n\log n+d+(d+cn-c'n)\leq c$
.

Thus, the statement also holds for n.

By induction we can conclude that the statement holds for all n. ▫

In the proof above we could avoid the "$+d$" term. However, then the base case $n=1$ would no longer work since $1\log 1=0$. In this case we would need to use two base cases, $n=2$ and $n=3$.

To analyze the running time of recursive algorithms we can, as in this case, use the Master theorem. So we can often actually avoid going to the trouble of an induction. But we do need strong induction to prove correctness of recursive algorithms.

Now in the setting of MergeSort we would first need to consider the while-loops. Assuming $A1$ and $A2$ are sorted, after the while-loops the array A stores the values of $A1$ and $A2$ sorted. This we would actually need to prove using a loop invariant proof. Then the induction as such is quite simple.

Proof: By strong induction on n we prove the following statement for all n:

$P(n)$: MergeSort correctly sorts arrays A of length n.

Basis ($n=1$): An array of size one is already sorted. And indeed the algorithm does nothing.

Step ($k<n\rightarrow n$): Assume the statement $P(k)$ holds for $1\leq k<n$ (I.H.). The elements of A are disjointly stored in $A1$ and $A2$. Now, $len(A1)<len(A)$ and $len(A2)<len(A)$, so by I.H. the recursive calls sort $A1$ and $A2$ correcty. Since $A1$ and $A2$ are now sorted, after the while-loops the elements of $A1$ and $A2$ (thus, the elements originally in A), are stored sorted in A.

Thus, the statement also holds for n.

By induction we can conclude that the statement holds for all n. ▫

So in short, in most cases induction is not difficult to use for proving the correctness of recursive algorithms: essentially it is a matter of *(a)* using the structure of induction correctly and *(b)* making use of the fact that (by induction hypothesis) the recursive calls work correctly.

3. A Final Warning

An induction requires a base case and and induction step. It may (or may not) seem a reasonable approach to simply check a handful of "base cases", and then conclude that the pattern will continue without providing an induction step. A peculiar example for which this approach would go wrong is based on the *Borwein integral*. See https://johncarlosbaez.wordpress.com/2018/09/20/patterns-that-eventually-fail/. There you find a sequence of integrals (depending on n), which evaluates to $\pi/2$ for all $n<9.8\cdot 10^{42}$, but eventually evaluates to different values. So only checking a few small cases would lead to a wrong conclusion.