Lecture 15 — Mathematical Induction and Recursive Definitions

Table of contents

Learning goals 1
Worked: what is wrong with this 'proof' that all numbers are equal? 1
Recursively defined sequences 4
A recursively defined set 6
Worked: $F_n < 2^n$ for the Fibonacci sequence 7
See also 8

Pre-work: Video 019 (Recursive Definitions). See recursive-definitions[1] for the reference material — what a recursive definition of a sequence or a set consists of, and the worked examples — and mathematical-induction[2] / strong-induction-and-the-well-ordering-principle[3] for the proof machinery used below.

Learning goals

1. Continue to become more comfortable with **mathematical induction**.
2. Become more familiar with **recursive definitions of sequences and sets**.

Worked: what is wrong with this 'proof' that all numbers are equal?

Proof. Let $P(n)$ be the predicate "for any collection of n numbers, $x_1, x_2, \ldots, x_n$, it is always the case that $x_1 = x_2 = \cdots = x_n$."

Basis step: $P(1)$ is true, since the collection is just x_1.

[1] courses/math1061/recursive-definitions.pdf
[2] courses/math1061/mathematical-induction.pdf
[3] courses/math1061/strong-induction-and-the-well-ordering-principle.pdf

Inductive step: Suppose that k is a positive integer and that $P(k)$ is true. This means that in each collection of k numbers, all the numbers are equal.

Now consider a collection of $k+1$ numbers, say $x_1, x_2, \ldots, x_k, x_{k+1}$.

The sub-collection $x_1, x_2, \ldots, x_k$ contains k numbers, and so $x_1 = x_2 = \cdots = x_k$ by the inductive hypothesis.

The sub-collection $x_2, x_3, \ldots, x_{k+1}$ also contains k numbers, and so $x_2 = x_3 = \cdots = x_{k+1}$ by the inductive hypothesis.

Therefore $x_1 = x_2 = \cdots = x_k = x_{k+1}$ and so $P(k+1)$ is true.

By mathematical induction, $P(n)$ is true for each integer $n \geq 1$, and so we have proven that all numbers are equal.

The conclusion is **false**, so something in the argument must fail.

The basis step is fine. So are "now consider a collection of $k+1$ numbers" and the first sub-collection's conclusion. The trouble is with the *second* sub-collection, $x_2, \ldots, x_{k+1}$. There are two readings of where it goes wrong; both turn on that sub-collection and they differ only over which line is at fault.

One reading puts the fault at the **second use of the inductive hypothesis**: if $P(k)$ is a statement about $x_1, \ldots, x_k$, i.e. $x_1 = x_2 = \cdots = x_k$, then concluding $x_2 = x_3 = \cdots = x_{k+1}$ carries $P(k)$ over to a *different* k numbers.

The other puts it one line later. $P(n)$ as stated is about *any* collection of n numbers, and both sub-collections genuinely do contain k of them, so each conclusion is defensible taken alone; what cannot be defended is the **"Therefore" line, at** $k = 1$. Gluing $x_1 = \cdots = x_k$ to $x_2 = \cdots = x_{k+1}$ into one chain requires the two sub-collections to **overlap** — you need some number that appears in both, to carry the equality across. For $k \geq 2$ they share $x_2, \ldots, x_k$ and the step is sound. For $k = 1$ the collection is just x_1, x_2, the two sub-collections are $\{x_1\}$ and $\{x_2\}$, and they share nothing: each says only that a single number equals itself, and there is no reason at all that $x_1 = x_2$.

Put the two sub-collections on one row and the overlap is visible: $x_1, \ldots, x_k$ (orange in the figure) and $x_2, \ldots, x_{k+1}$ (yellow), the second leaving out x_1. What matters is what the two share.

So $P(1) \to P(2)$ is false. The chain of implications never gets off the basis step, and the "proof" proves nothing beyond $P(1)$.

The lesson: the inductive step has to hold for *every* k from the basis value on — including the very first one. An inductive step that is valid for all large k but fails at the bottom leaves a gap the induction cannot cross.

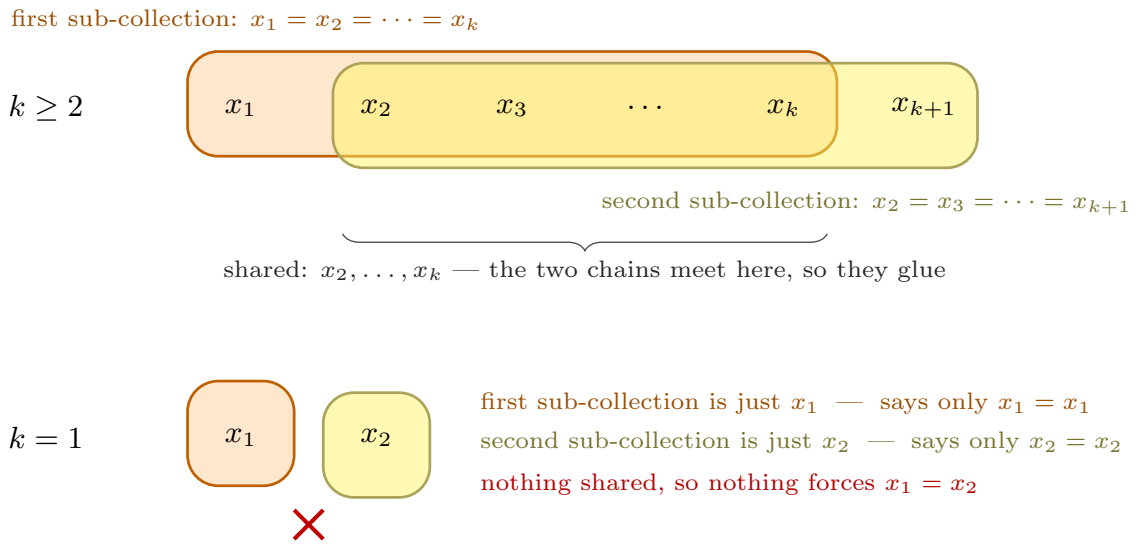


Figure 1: Why the inductive step fails at $k = 1$. The two sub-collections of the $k + 1$ numbers are drawn as overlapping rings on a single row, the first in orange and the second in yellow, so the shared elements can be seen at once. For $k \geq 2$ they overlap on $x_2, \ldots, x_k$, and that shared block is exactly what lets the two equality chains be joined into one. For $k = 1$ the rings shrink to x_1 and x_2, they share nothing, and nothing in the argument forces $x_1 = x_2$.

Recursively defined sequences

A recursive definition needs **initial conditions** and a **recurrence relation** — and the same recurrence relation with different initial conditions can give a completely different sequence ($a_1 = 2$, $a_k = 3a_{k-1}$ gives $2, 6, 18, 54, ...$; $b_1 = 7$ with the same recurrence gives $7, 21, 63, 189, ...$). See recursive-definitions[4].

Number the two parts in the order you *use* them: **(i) initial conditions, (ii) recurrence relation**. The numbering comes back in 2026-08-31-solving-recurrence-relations[5], where the two things Method 1 checks are labelled (i) and (ii) because they are exactly these two halves.

Worked: which recurrence does $b_n = 4n + 3$ satisfy? (pre-class Q2)

Which of the following recurrence relations is satisfied by the sequence $b_n = 4n + 3$ for each integer $n \geq 1$?

b. $b_k = 4 + b_{k-1}$ for each integer $k \geq 2$.

The method: **assume $b_{k-1} = 4(k-1)+3$, check what b_k equals** — that is, $P(k-1) \Rightarrow P(k)$ — for each candidate in turn. The target is $b_k = 4k + 3$.

Option	Substituting $b_{k-1} = 4(k-1) + 3$
a. $b_k = 4b_{k-1}$	$4(4(k-1)+3) =$ $16k - 16 + 3 = 16k - 13$
b. $b_k = 4 + b_{k-1}$	$4 + 4(k-1) + 3 = 4k + 3$
c. $b_k = 3 + b_{k-1}$	$3 + 4(k-1) + 3 = 4k + 2$
d. $b_k = 3b_{k-1}$	$3(4(k-1)+3) = 12k - 3$

Worked: the recursive definition of $c_n = 3^n - 2^n$ (pre-class Q3)

The sequence $c_n = 3^n - 2^n$ for each integer $n \geq 1$ satisfies the recurrence relation $c_k = 5c_{k-1} - 6c_{k-2}$ for each integer $k \geq 3$. What is the recursive definition for this sequence?

c. $c_1 = 1$, $c_2 = 5$, $c_k = 5c_{k-1} - 6c_{k-2}$ for each integer $k \geq 3$.

Two things pin it down: the values of the initial conditions, and the indexing.

The **initial conditions** come from evaluating the explicit formula:

[4] courses/math1061/recursive-definitions.pdf
[5] courses/math1061/2026-08-31-solving-recurrence-relations.pdf

$$c_1 = 3^1 - 2^1 = 3 - 2 = 1 \qquad c_2 = 3^2 - 2^2 = 9 - 4 = 5$$

Option c has those two values. Option **b** has the clashing $c_1 = 0$ — so b goes out on values. Options **a and d** go out on *indexing* instead: they define c_0 and start the recurrence at $k \geq 2$, whereas the sequence was given as starting at $n = 1$, so c_0 lies outside its domain. Option d fails on values too — it puts $c_1 = 5$.

Option a is the subtle one. Its arithmetic is entirely correct — $c_0 = 3^0 - 2^0 = 0$, $c_1 = 1$, and $c_2 = 5(1) - 6(0) = 5$, $c_3 = 5(5) - 6(1) = 19 = 3^3 - 2^3$ — so it does generate the right sequence for every $n \geq 1$. It is wrong only because it defines a term the given sequence does not have. That is the difference between a wrong sequence and a right sequence with the wrong domain.

Worked: a recursive definition for $a_n = (-1)^n/n!$

Consider the sequence defined by the explicit formula $a_n = \dfrac{(-1)^n}{n!}$ for each integer $n \geq 0$. (Recall that for $n \in \mathbb{Z}^+$, $n! = n(n-1)\cdots(2)(1)$, and that $0! = 1$ **by definition**. Memorise that clause: it is what makes a formula built out of $n!$ work at $n = 0$ with no special case, which is exactly what this sequence needs.)

(a) Determine the first five terms of this sequence and guess a suitable recurrence relation.

$$a_0 = \frac{(-1)^0}{0!} = \frac{1}{1} = 1 \qquad a_1 = \frac{(-1)^1}{1!} = \frac{-1}{1} = -1 \qquad a_2 = \frac{(-1)^2}{2!} = \frac{1}{2}$$
$$a_3 = \frac{(-1)^3}{3!} = \frac{-1}{3 \times 2} = -\frac{1}{6} \qquad a_4 = \frac{(-1)^4}{4!} = \frac{1}{4 \times 3 \times 2} = \frac{1}{24}$$

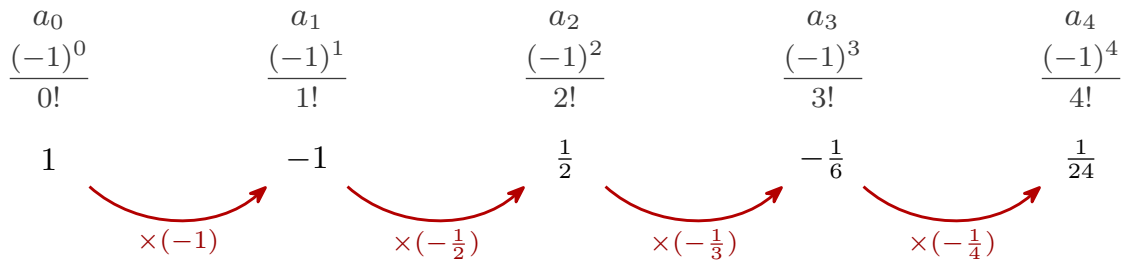
The guess does not come from staring at the five values; it comes from asking **what multiplies each one to give the next**. Write those multipliers in as curved arrows between consecutive terms:

$$a_n = -\frac{1}{n} a_{n-1}$$

(b) Show that this sequence satisfies your recurrence relation.

Suppose $a_{n-1} = \dfrac{(-1)^{n-1}}{(n-1)!}$. Then

$$a_n = -\frac{1}{n} \cdot a_{n-1} = \frac{(-1)(-1)^{n-1}}{n \cdot (n-1)!} = \frac{(-1)^n}{n!} \qquad \square$$



$$\text{the multiplier at step } n \text{ is } -\frac{1}{n}, \quad \text{so } a_n = -\frac{1}{n}\, a_{n-1}$$

Figure 2: Reading the recurrence off the terms. Each curved arrow carries the factor taking one term to the next, and the four factors $-1, -\frac{1}{2}, -\frac{1}{3}, -\frac{1}{4}$ are read as one pattern, $-\frac{1}{n}$ at step n. The pattern is the guess; part (b) is where it gets proved.

(c) Write down a recursive definition of this sequence.

- *initial condition*: $a_0 = 1$
- *recurrence relation*: $a_n = -\dfrac{1}{n} a_{n-1}$ for all $n \geq 1$

A recursively defined set

Three parts — **Base**, **Recursion** and **Restriction** — spelled out in recursive-definitions[6] along with the set of all binary strings B, in which the string 011 is built up as $\varepsilon \to 0 \to 01 \to 011$. The worked question below is practice with step II, the Recursion: what its rules do and do not generate.

Worked: which strings are in S?

Let the subset S of the set of all binary strings be defined recursively as follows. Base: the empty string is in S. Recursion: if $s \in S$, then $s1 \in S$ and $s01 \in S$. Restriction: nothing is in S other than objects defined by the Base and the Recursion.

So there are two rules, named by what they append: **rule `1`** (append `1`) and **rule `01`** (append `01`). Naming them matters, because a membership argument is a chain of rule applications and each link has to say which rule was used. The rule's bare name goes above the arrow, and nothing else:

[6] courses/math1061/recursive-definitions.pdf

(a) Is the string 1101 $\in S$?

Yes — it has a build history:

$$\varepsilon \xrightarrow{1} 1 \xrightarrow{1} 11 \xrightarrow{01} 1101 \in S$$

(b) Is the string 0110 $\in S$?

No — S contains no strings ending in 0. Both rules append a string ending in `1`, so every non-empty member of S ends in `1`, and by the Restriction there is nothing else in S.

(c) Is the string 1001 $\in S$?

No — S contains no strings with two or more consecutive 0s. A `0` can only ever enter as the first character of an appended `01`, so every `0` is immediately followed by a `1`.

(d) State a property that all strings in S must have and explain why.

The number of 1s is greater than or equal to the number of 0s.

The empty string has zero of each, so the property holds at the Base. Each rule preserves it: appending `1` adds one 1 and no 0s, and appending `01` adds one of each. So no application of the Recursion can ever make the 0s outnumber the 1s, and by the Restriction every member of S is reached by some finite chain of these rules.

Worked: $F_n < 2^n$ for the Fibonacci sequence

Recall the Fibonacci sequence: $F_0 = 1$, $F_1 = 1$, $F_k = F_{k-1} + F_{k-2}$ for every integer $k \geq 2$. Use (strong) mathematical induction to prove that $F_n < 2^n$ for every integer $n \geq 1$.

The first thing to settle is the number of base cases. The recurrence $F_k = F_{k-1} + F_{k-2}$ points to **2 base cases**; the "$n \geq 1$" in the claim points to **1 base case**. The recurrence settles it. $F_k = F_{k-1} + F_{k-2}$ reaches back **two** terms, so producing $P(k+1)$ needs $P(k)$ *and* $P(k-1)$: that is strong induction, and it needs two base cases to start from.

The same reasoning fixes the range of the inductive step: it is $\forall k \geq 2$, not $\forall k \geq 1$. At $k = 1$ the step would reach for $P(0)$, which is not one of the base cases. This is not merely an indexing nicety, because with $F_0 = 1$ the statement $P(0)$ is **false** — it would say $F_0 < 2^0$, i.e. $1 < 1$. So the claim could not have been extended down to $n = 0$ in any case.

Let $P(n)$ be the predicate $F_n < 2^n$.

Basis step — $P(1)$ and $P(2)$.

LHS of $P(1)$: $F_1 = 1$. RHS of $P(1)$: $2^1 = 2$. Since $1 < 2$, $P(1)$ is true.

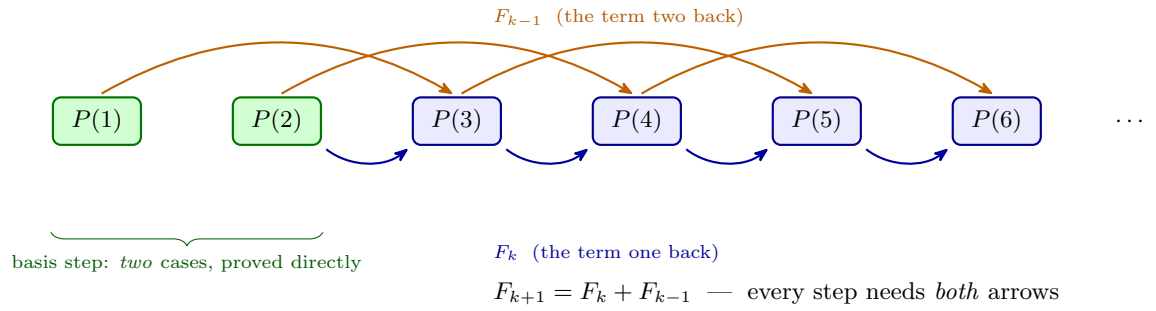


Figure 3: The shape of the strong induction, drawn out. Each $P(k+1)$ is reached from the term one back (blue) and the term two back (orange), because the recurrence $F_{k+1} = F_k + F_{k-1}$ names both; that is why the inductive hypothesis has to be the strong one, and why two cases have to be proved outright before the chain can start.

LHS of $P(2)$: $F_2 = F_1 + F_0 = 1 + 1 = 2$. RHS of $P(2)$: $2^2 = 4$. Since $2 < 4$, $P(2)$ holds.

Inductive step — $(P(1), \dots, P(k)) \to P(k+1)$ for all $k \geq 2$.

LHS of $P(k+1)$:

$$\begin{aligned} F_{k+1} = F_k + F_{k-1} &< 2^k + 2^{k-1} && \text{(by the inductive hypothesis } P(k) \text{ and } P(k-1)) \\ &= 2(2^{k-1}) + 2^{k-1} = 3(2^{k-1}) < 2^2(2^{k-1}) && (3 < 4 = 2^2) \\ &= 2^{k+1} \end{aligned}$$

which is the RHS of $P(k+1)$. Therefore $(P(1), \dots, P(k)) \to P(k+1)$. $\square$

The trick in the middle is rewriting 2^k as $2(2^{k-1})$ so that both terms carry the same power and can be collected — turning $2^k + 2^{k-1}$ into $3(2^{k-1})$, which is then bounded by $4(2^{k-1}) = 2^{k+1}$.

See also

- recursive-definitions[7] · mathematical-induction[8] · strong-induction-and-the-well-ordering-principle[9] · sequences-and-summations[10]

[7] courses/math1061/recursive-definitions.pdf
[8] courses/math1061/mathematical-induction.pdf
[9] courses/math1061/strong-induction-and-the-well-ordering-principle.pdf
[10] courses/math1061/sequences-and-summations.pdf

- 2026-08-27-strong-induction-and-well-ordering[11] — Lecture 14
- practice-problems[12] — §19 Recursive Definitions, with full worked solutions

[11] courses/math1061/2026-08-27-strong-induction-and-well-ordering.pdf
[12] courses/math1061/practice-problems.pdf