# Induction and Recursion

Notes for 24.241, Fall 2022

Written by Prof. Jamie Tappenden; with edits by Josh Hunt

## Contents

## 1 Induction/Recursion: Some basic facts

### 1.1 What are recursion and induction?

*Mathematical induction* is a simple pattern of reasoning that is indispensible to computer programming and analysis of numbers. Rudimentary forms even show up occasionally in everyday life. Typically the uses of this reasoning are abstract, but the core pattern is simple and occurs in concrete cases as well as abstract ones. As we'll often find in this course, the pattern we are trying to learn is just a common-sense pattern of thinking, rendered more rigorous and explicitly.

Before explaining mathematical induction as a proof technique, I'll need to lay out a related pattern of definition called *definition by recursion / recursive definition* (or sometimes: *definition by induction / inductive definition*). Definitions of this type are central to computer programming and to the understanding of natural languages like English.

The basic idea of recursion is this. Say you have some collection of things and you want to define a subset of those things. Say you have:

**1) a fixed starting point and**
**2) an operation you can apply over and over again to produce new things**

And say that when you apply the operation to a given argument, you get something new that retains what you had before you applied the operation. Then you can apply this operation to this "something new" and get a further collection of things. And you can keep going, applying the operation again and again, and getting new things again and again.

This gives a way to define a collection: start at the starting point and take what you get by applying the operation over and over again as often as possible. So for example:

Problem: Specify all the numbers greater to or equal to 0: $\{0, 1, 2, ...\}$ .

Here is a rough statement of a way to define this set:
1) start with the set $\{0\}$,
2) put into the set everything you get by repeatedly adding one to the things you already have.
3) The "completed" new set will be the collection of all the things you get by following this process.

This needs to be formulated a bit more precisely. The idea is that we can define bigger and bigger sets in stages:

Stage 0: $S_0 = \{0\}$
Stage 1: $S_1 = \{0, 0+1(=1)\}$
Stage 2: $S_2 = \{0, 1, 1+1(=2)\}$
Stage 3: $S_3 = \{0, 1, 2, 2+1(=3)\}$
$\vdots$
Stage n: $S_n = \{0, 1, 2, \ldots (n-1), [(n-1)+1](=n)\}$
$\vdots$

At each stage $S_n$, you take the set formed at stage $S_{(n-1)}$ and put in $(n-1)+1$. That is, you put in $n$.

The set of all numbers greater than or equal to 0 is just the set you get by taking *all the stages together*.

There is a specific, canonical way you present a recursive defintion. **NB: If you are asked to give a *recursive definition* on a test, or problem set, you will need to give something set up this way, with a base case, recursion clause and closure clause:**

**Base Case**: 0 is a number.
**Recursion Clause**: If $n$ is a number, then $n + 1$ is a number.
**Closure Clause**: Nothing is a number except what is obtained by beginning with the Base Case and repeatedly applying the Recursion Clause.

The Closure Clause is sometimes left implicit, but you ought to *make it explicit!*

You can get different sets by keeping the Base Case fixed, and changing the Recursion Clause. For example, the non-negative even numbers $\{0, 2, 4, 6, ...\}$ result from beginning with $\{0\}$ and putting into the set everything you get by adding 2.

In the canonical form, this definition of the even numbers would be:

**Base Case**: 0 is a number.
**Recursion Clause**: If $n$ is a number, then $n + 2$ is a number.
**Closure Clause**: Nothing is a number except what is obtained by beginning with the Base Case and repeatedly applying the Recursion Clause.

You can also get different sets by keeping the Recursion Clause fixed, and changing the Base Case. For example, the non-zero natural numbers $\{1, 2, 3, 4, ...\}$ result from beginning with $\{1\}$ and putting into the set everything you get by adding 1.

In the canonical form, this definition of the non-zero natural numbers would be:

**Base Case**: 1 is a number.
**Recursion Clause**: If $n$ is a number, then $n + 1$ is a number.
**Closure Clause**: Nothing is a number except what is obtained by beginning with the Base Case and repeatedly applying the Recursion Clause.

Typically when recursion is used, it is applied to infinite sets of abstract things. But the basic reasoning can apply also to finite sets of concrete things, as we'll see in section 2. Among the most prominent topics that invite inductive arguments and support recursive definitions are the natural numbers $\{0, 1, 2, \ldots n, \ldots\}$ and languages that are generated by repeatedly applying rules of grammar to basic symbolic vocabulary.
We will be principally concerned with the structured languages, but inductive arguments in arithmetic are simpler, so I'll look at illustrations from arithmetic as well as from structured languages.

## 2 Proof By Induction: Examples with no Numbers

### 2.1 The Towers of Hanoi Puzzle

(See also the posted "towers-of-hanoi.pdf" slides.)

The most logically straightforward inductive arguments involve numbers, but since this isn't a math course, I'll begin with a couple of simple concrete cases.

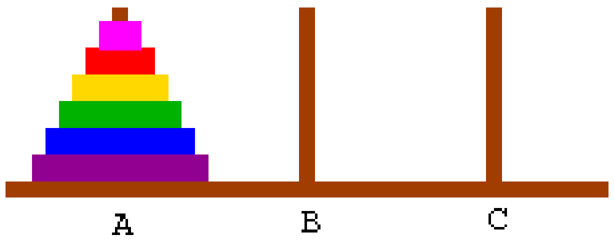
Here's a puzzle that looks at first as if it might be very hard, but it becomes simple and straightforward if we think about it recursively.

This is a tabletop puzzle called the *Towers of Hanoi*.

There are three pegs and $n$ discs of different sizes, with holes that allow the discs to be slipped onto any of the pegs.

At the beginning of the game, all $n$ discs are all on the leftmost pole, with the largest disc on the bottom, and then the remaining discs in decreasing order of size piled on top of the first, with the smallest on top.

Here is a drawing of the opening setup in the case of 6 discs, with all the discs on the leftmost peg A:



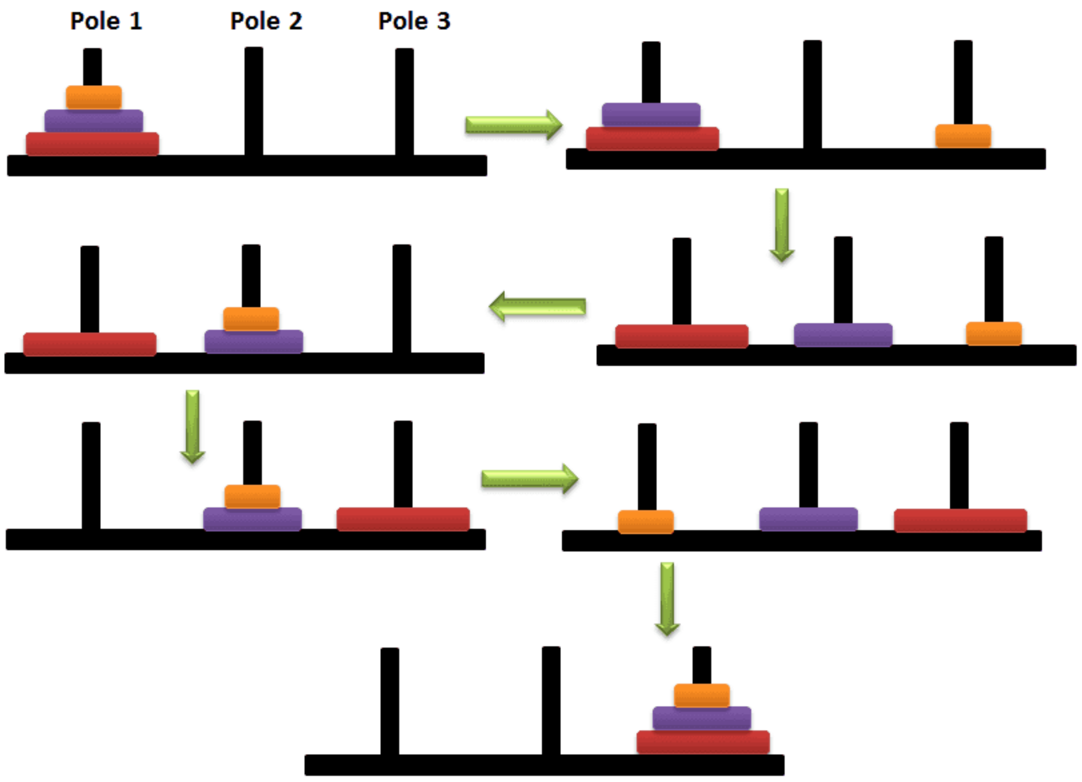
The goal of the game is to end up with all discs on the right peg C, in the same order, that is, smallest on top, and increasing in size towards the bottom.

There are restrictions on how the discs are moved:

1. The only allowed type of move is to take **one** disc from the **top** of a pile and drop it on another peg. Moving several discs at once is not allowed.
2. A larger disc can never lie above a smaller disc, on any peg.

Before you read on, try to solve a couple of cases for yourself. Try the case of 3 pegs and 4 pegs.

To get a better sense of what this involves, it's easiest if we use some visuals. I will go through an animated version in lecture, but for orientation here is a picture of a solution for 3 discs:



Why should we expect the problem to be solveable for arbitrary number $n$ of discs? The core insight is to think in terms of smaller sub-problems. Can you use the solution for case $n$ to help solve the case $n + 1$?

Let's see if the case of n=3 can give us a clue. Look at the first three moves. In those moves you shift two discs from pole 1 to pole 2, in size order. That is: **you carry out a solution for the case of 2 discs!** (Except the ending pole is the middle one instead of the rightmost one.)
The fourth move: Shift the biggest disc from the left pole to the right one.
The final moves: In those moves you shift two discs from pole 2 to pole 3, in size order. That is: **you carry out a solution for the case of 2 discs again!** (Except the beginning pole is the middle one and the auxiliary pole is the rightmost one.)
The bottom line: if you can solve the case for 2 discs, you can turn that into a solution for the case of 3 discs!

Now try this yourself with the case of 4 discs. You will find it easy to solve it this way: First, use the solution for 3 discs to move the smallest 3 discs to the middle pole. Move the biggest disc to the right pole. Then use the solution for 3 discs to move the smallest 3 discs to the right pole, on top of the biggest one.

If you review the slides, you can see that in each case except the base case $n = 1$, you can give a description of what you do for a pile of $n + 1$ discs, if you have a routine for solving the $n$ disc case.

a) Use the $n$ routine (with different pegs, but that doesn't matter) to move the top $n$ discs to peg 2, leaving the largest disc alone on peg 1.

b) Move the largest disc to peg 3.

c) Use the $n$ routine to move the discs from peg 2 to peg 3, and you are done!

This can be framed in terms of an inductive argument to prove: For every $n$, a pile of $n$ discs in increasing order of size from bottom to top can be moved from peg 1 to peg 3 using legal moves.

Proof:
Base Case: $n = 1$. A single disc can be moved from 1 to 3.

Induction step: $n = k$ for some arbitrary number $k$. Assume (Induction Hypothesis) that there is a routine to move a legal pile with $k$ discs from 1 peg to another with legal moves (if at the beginning the other pegs are empty).

Now we want to show that there is a routine to move a pile with $k + 1$ discs from 1 peg to another with legal moves.

Proof (Induction Step): (basically repeating what we've just said):

a) Use the $k$ routine (with different pegs) to move the top $k$ discs to peg 2, leaving the largest disc alone on peg 1.

b) Move the largest disc to peg 3.

c) Use the $k$ routine to move the rest of the discs from peg 2 to peg 3, (you can act as if peg 3 is empty for the sake of applying the routine) so that all of the discs on peg 2 rest on top of the largest disc, with all the discs on peg 3.

Do this and you are done! The problem is solved for the case $n+1$, on the assumption that you knew how to solve the case $n = k$. But $k$ was an arbitrary number, so this procedure will work for any number.

This completes the induction step, and hence the proof.

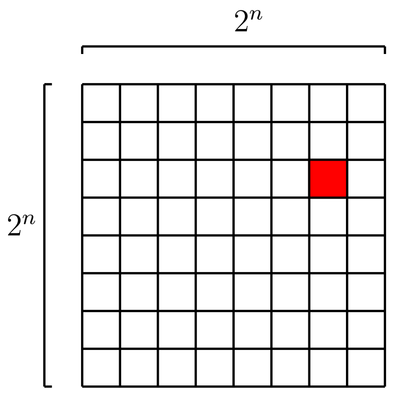
## 2.2 Tiling a Chessboard

Here is another argument by mathematical induction that doesn't involve any symbols or numbers. Like the Tower of Hanoi example, the point of the solution is to find a way of reducing the problem for a given case to a problem for a simpler case, and show that if you have a solution for the simpler case(s) you can turn that into a solution for the given case. This is the essence of much human problem-solving!

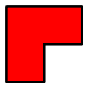
I'll consider the subset of chessboards that are $2^n$ squares wide and $2^n$ squares high for some number $n > 0$. That is, the chessboards that are $2\times2, 4\times4, 8\times8, 16\times16, \ldots$

Also, it will be useful to remind yourself of a simple fact: $2^{n+1} = 2 \times 2^n$. (According to Platonic epistemology of math, we all knew this fact already—we just needed to be *reminded* of it :p)

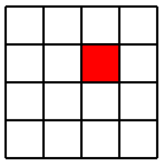
**Prove**: Any square chessboard that is $2^n$ squares wide and $2^n$ squares high, **with one square covered**, can be completely covered (with no overlap) by a collection of tiles, where each tile is three squares in an L-shape. In pictures:
Say you have a $2^n$ x $2^n$ chessboard, with one square removed (removed square marked in red). Example:



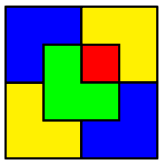
(The chessboard in the diagram is 8 x 8, but it is meant to represent an arbitrary $2^n \times 2^n$ board.) If you want to solve this yourself, don't read on yet! No spoilers!

The white squares can all be covered (with no overlap) by tiles in this shape:  .

**Clarifying Remark 1**: To give you an idea of what we are trying to prove, here is an example of a 4 x 4 chessboard (i.e. $2^2$ x $2^2$) with one square covered (in red).

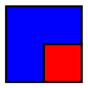


The above chessboard can be L-tiled this way. (The L-shaped tiles are displayed in different colors, for easier viewing.) You want to show that this kind of a tiling will be possible for any $2^n$ x $2^n$ board with one square removed.

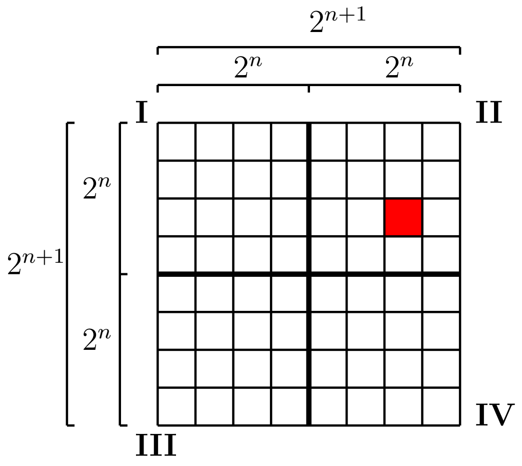


**Clarifying Remark 2**: The solution proceeds by induction, with the help of one small trick. First I'll layout out the situation with an eye to the inductive argument, then I'll show you the trick.

The base case, for a 2 x 2 board, is automatic. Whatever square is chosen, there will be an L-shape left over, as in this case (chosen square in red, remainder in blue):



Now we need to tackle the inductive step. This will require us to reduce the problem of tiling a $2^{n+1}$ x $2^{n+1}$ board to some question or family of questions about tiling $2^n$ x $2^n$ boards, since $2^n$ x $2^n$ boards are covered by the induction hypothesis. In a picture (okay, so now the solution is basically about to be given!):



By dividing the $2^{n+1}$ x $2^{n+1}$ board into 4 quadrants, each of size $2^n$ x $2^n$ we get a situation where the inductive hypothesis can help us part of the way. The red square will be in one of the quadrants - in the diagram above, it is in the upper right hand quadrant, labelled as quadrant II. By the induction hypothesis we know that that quadrant II can be covered in L-tiles, leaving the red square uncovered.

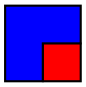
**But** this leaves us with one more problem to solve before we can conclude that every $2^{n+1}$ x $2^{n+1}$ board with one square missing can be covered in L-tiles: How can we show that the **rest** of the board (quadrants I, III, and IV in the above diagram) can be tiled with L-tiles? The induction hypothesis just addresses chessboards with one square missing, not chessboards with **all** the squares uncovered. Here we need a bit of inspiration to figure out how to produce a situation where we can use the induction hypothesis on the other three quadrants. Once that is done, the solution falls into place.

**This is where the trick is needed: Place an L-tile in the center of the board, so that each of the quadrants has a square covered. This reduces the problem of tiling the $2^{n+1}$ x $2^{n+1}$ board to 4 copies of the problem of tiling $2^n$ x $2^n$ boards. (In each case, with one square covered.)**

**The Solution to the Problem, Spelled Out as an Inductive Argument**:

Problem: Show by induction that a $2^n$ x $2^n$ chessboard with one square covered can be covered by L-tiles without overlap.

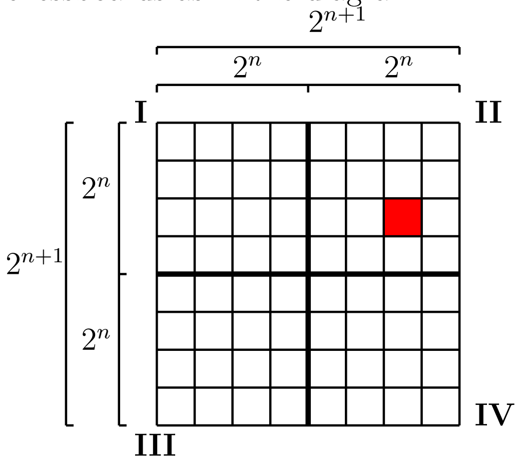
Proof:
**Base case**: Say that n=2. This is immediate, since whatever square is omitted, there will be an L-shape left over, as in this case (chosen square in red, remainder in blue):



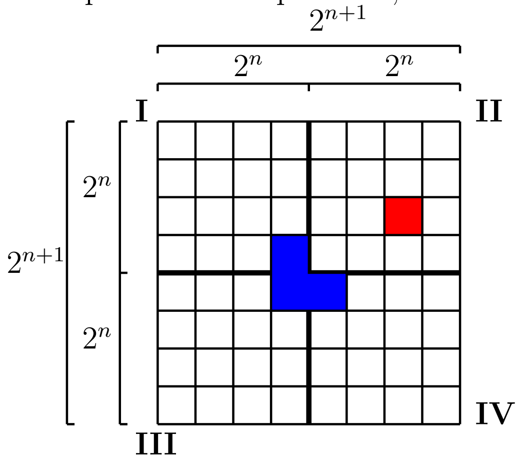
**Induction step**: **Assume** (induction hypothesis) that any $2^n$ x $2^n$ board with one square removed can be covered without overlap with L-tiles. We want to show that any $2^{n+1}$ x $2^{n+1}$ board with one square removed can be covered without overlap with L-tiles.

Proof of induction step:

Given a $2^{n+1}$ x $2^{n+1}$ board with one square removed, break it into four $2^n$ x $2^n$ chessboards as in the diagram:



By dividing the $2^{n+1}$ x $2^{n+1}$ board into 4 quadrants, each of size $2^n$ x $2^n$ we get a situation where the inductive hypothesis can help us. The red square will be in one of the quadrants - in the diagram above, it is in the upper right hand quadrant, labelled II. By the induction hypothesis we know that that quadrant II can be covered in L-tiles, leaving the red square uncovered. Place one L-tile in the center so it covers one square of each quadrant, as in this diagram:



**Now** we have a situation where we can apply the induction hypothesis to the other 3 quadrants: Each of them is a $2^n$ x $2^n$ chessboard with one square missing. Hence we can cover each of the remaining quadrants with L-tiles. The tiling of all four quadrants, plus the tile placed in the center, cover the whole $2^{n+1}$ x $2^{n+1}$ board except for the missing square. This proves the induction hypothesis.

From the base case and the induction step, we conclude that the thesis is true in general: every $2^{n+1}$ x $2^{n+1}$ board with one square missing can be covered without overlap with L-tiles.

## 3 Recursion on numbers: A simple example

A simple example of a recursive definition is the definition of multiplication by some number (say: 17) in terms of adding 17:

**Basis clause**: $17 \cdot 0 = 0$
**Recursion clause**: $17 \cdot (n+1) = (17 \cdot n) + 17$

(More generally, we can define "$x \cdot n$" recursively by replacing "17" above with "$x$".) Say that we are asked to calculate $17 \cdot 23$. This definition fixes a value for $17 \cdot 23$, though to get it we need to unpack the definition by applying the recursion clause repeatedly until we hit the bedrock of the basis:

$17 \cdot 23 = (17 \cdot 22) + 17$

$= (17 \cdot 21) + 17 + 17$

$= (17 \cdot 20) + 17 + 17 + 17$

$\vdots$
$= (17 \cdot 1) + \underbrace{17 + 17 + 17 \ldots + 17}_{22\, times}$
$= (17 \cdot 0) + \underbrace{17 + 17 + 17 \ldots + 17}_{23\, times}$
$= 0 + \underbrace{17 + 17 + 17 \ldots + 17}_{23\, times}$

At each step, until the last one, you get the value of a product of 17 with some number in terms of the product of 17 with a smaller number. At the bottom of the chain, you finally get an expression that is i) unpacked in terms of the basis clause alone, and ii) several "+ 17"'s and doesn't contain any reference to the "$\cdot 17$" function we are trying to define.

### 3.1 Mathematical Induction

Mathematical induction is a form of argument that applies to collections generated by recursion. (The natural numbers are a particularly simple example of such a collection.) Say you have some claim $C(x)$ that can be made about any element $x$ of the natural numbers $\mathbb{N}$. The following pattern of argument will give you a proof. **If**

**you are asked to give a proof by mathematical induction, the solution will need to contain something fitting this pattern:**

**Base Case**: The property or claim $C(x)$ holds of 0. [In some cases the first element will be 1.]
**Induction Step**: If $C(x)$ holds of $n$, then $C(x)$ holds of $n + 1$.

If you can prove the Base Case and the Induction Step, you can conclude: $C(x)$ is true for every $x \in \mathbb{N}$. In the next section I'll look at some examples.

[Note: There is a variation of this pattern called Complete Induction that I discuss in section 4.3]

## 4 Induction on Numbers

We have already seen in section 2 that induction applies to non-numerical cases. We'll consider later some concrete examples involving strings of letters (very relevant for proving things about our logical systems, whose sentences are, after-all, strings of symbols). These examples hopefully cure the misimpression that induction is only of use when dealing with the natural numbers $\mathbb{N}$. So with that, we'll now do a few examples involving numbers, since they display the structure of inductive arguments in a particularly simple way.

### 4.1 Induction on Numbers: Adding the first $n$ Numbers

Here is a well-known example of the proof that the sum of the first $n$ natural numbers (beginning at 1 and ignoring 0, because it contributes nothing to the sum) is $\frac{1}{2}n(n+1)$ for arbitrary $n \in \mathbb{N}$. (You may have already seen this in high school algebra.)

I'll discuss the problem a bit, and then give the proof, before dismantling it to study the moving parts.

First a bit of notation: if you want to write a sum of a list of numbers, then you can label the numbers with $1, 2, \ldots n$ (aka "index them"), yielding a list "$m_1, m_2, \ldots m_n$." For short, we write this list as:

$\displaystyle\sum_{i=1}^{n} m_i$ (using a capital greek Sigma).

It is particularly easy to write the sum of the first $n$ natural numbers, because you don't need to index them to anything - they are in order already!

So just write: $\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n$

Before we consider the problem in general, let's look at particular cases to orient ourselves.

Note that:

$1 = \frac{1}{2}(1)(1+1) = \frac{2}{2} = 1$
$1 + 2 = \frac{1}{2}(2)(2+1) = 3$
$1 + 2 + 3 = \frac{1}{2}(3)(3+1) = (3)(2) = 6$
$1 + 2 + 3 + 4 = \frac{1}{2}(4)(4+1) = (2)(5) = 10$
$\vdots$

Each of these is an instance of the formula $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

So: When you try the formula in simple cases, it works. You might conjecture that it holds generally - but jumping to a conclusion from a pattern you see in a few specific cases isn't a *proof*.

Note a crucial foothold: there is a way to break down each case so that the given case depends straightforwardly on the immediately previous one.

Say that I asked you to compute $1 + 2 + 3 + 4 + 5$ and you already knew that $1 + 2 + 3 + 4 = 10$

Note that $1 + 2 + 3 + 4 + 5 = \underbrace{[1 + 2 + 3 + 4]}_{\text{hmm.. where have I seen this before?}} + 5$

You know that $1+2+3+4 = 10$ so you can simplify this to: $10 + 5 = 15$, given what you already know. Not a big deal here, but it could be a real time-saver if you're dealing with 50 digit numbers!

It might take some time to add up the first 1,000,000 numbers, but if I already know the sum of the first 999,999, my job is easy: Just take *that* sum and add 1,000,000 to it!
First note that two pages ago we already covered the base case:

**Base Case:** $1 = \frac{1}{2}(1)(1+1) = \frac{2}{2} = 1$

For the Inductive Step, we first **assume** that we already know that for a given $n$, we know that $\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$

We call the assumption "The Induction Hypothesis".

[At this point there may be howls from the bleachers: Isn't this "Induction Hypothesis" just the assmption of what we have to prove???? Isn't that circular?

Good question! It shows you're paying attention. But the answer is: No, it isn't. Be patient, all will be revealed.]

Given the induction hypothesis, $\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$ we want to prove this equation, which I have labeled (*) for easy reference:

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

[**Note:** We assume the hypothesis is true for some number $n$, and we prove, *given this assumption*, that it must be true for $n+1$ as well.]

OK, let's remember what we noted above, namely that:

$$1 + 2 + \ldots + (n+1) = \underbrace{[1 + 2 + 3 + \ldots + n]}_{\text{hmm.. where have I seen this before?}} + (n+1)$$

Using the induction hypothesis, we can simplify: $1 + 2 + 3 + \ldots + (n+1) = [\frac{1}{2}n(n+1)] + (n+1)$

Some algebra: $[\frac{1}{2}n(n+1)] + (n+1) = [\frac{1}{2}(n^2+n)] + [\frac{2n}{2} + \frac{2}{2}]$

$= \frac{n^2+3n+2}{2} = \frac{(n+1)(n+2)}{2}$

Now look up at the equation we marked (*). How about that! We've produced what we were looking for!

So we conclude that, for every $n$, this formula holds: $\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$

Review what we did.

1) We proved that $\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$ is true when $n = 1$

2) We showed that for any $k$, **if** $\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$ is true when $n = k$ **then** $\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$ is true when $n = k+1$.

These two bits of information establish that
$\displaystyle\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$ when $n =$ **any** $k \in \mathbb{N}$.

It's easy to see how that works - say that I give you some specific number. To be concrete, say that I give you 325. How do you know that the formula holds when $n = 325$?

Well, the formula holds for $n = 1$ (base case) and so by applying the induction step to the base case you know that the formula holds for $n = 2$. Applying the induction step to $n = 2$ tells you the equation holds for $n = 3$ ...

Applying the induction step 324 times in this fashion gives you that the equation holds for $n = 325$.

### 4.2 A sketch of another example, to reinforce the proof pattern

Here's another standard example with the same flavor; I probably won't cover this one in lecture, and it is a bit more complicated, so if you don't think you'll find it illuminating, just cruise on to the next one...

$$\sum_{i=1}^{n} i^2 = \frac{1}{6}n(n+1)(2n+1)$$

Here is the general strategy for attacking this problem inductively. [This isn't a class on algebra, so I'll just skip over some computations, which are a distraction.]

First (**Base Case**) prove $\sum_{i=1}^{1} i^2 = \frac{1}{6}(1)(1+1)(2(1)+1)$.

That's just elementary arithmetic. (99% of the time, the base case is pretty trivial. Occasionally the conceptual realm throws us a curveball and the base case is harder than the induction step, but that is very rare. All hail the conceptual realm!)

To set up the **Induction Step**, make the assumption that for a given k, $\sum_{i=1}^{k} i^2 = \frac{1}{6}k(k+1)(2k+1)$.

Then set out to show that

$$\sum_{i=1}^{k+1} i^2 = \frac{1}{6}(k+1)(k+2)(2(k+1)+1).$$

The trick will be to find a way to *use* the information you are given in the induction hypothesis. Break down the $n = k+1$ case so it consists of the $n = k$ case plus some comparatively simple other stuff. This is usually one of the details of finding the solution that requires the most thought.

Happily in this case, when we are dealing with sums, it is easy to simplify the $n = k+1$ case by appealing to the $n = k$ case:

$$\sum_{i=1}^{k+1} i^2 = \underbrace{1+2+3+\ldots k}_{\text{This is } \sum_{i=1}^{k} i^2} + (k+1)^2$$

$$= [\textstyle\sum_{i=1}^{k} i^2] + (k+1)^2$$

$$= \underbrace{\frac{1}{6}k(k+1)(2k+1)}_{\text{Substituting } \frac{1}{6}k(k+1)(2k+1) \text{ for } \sum_{i=1}^{k+1} i^2} + (k+1)^2.$$

Now we have an equation that doesn't have the sum operator in it at all. We just have a *much* simpler problem in front of us: find out if it is possible to do the algebra that will reduce $\frac{1}{6}k(k+1)(2k+1) + (k+1)^2$ to

$\frac{1}{6}(k+1)(k+2)(2(k+1)+1)$. This can be done, but I'll spare you the algebra. You're welcome to do it if it suits your fancy!

## 4.3 Ordinary Induction and Complete Induction

Proofs by mathematical induction fit into two slightly different schemes, depending on how much you assume about the numbers less than $n+1$. These argument patterns aren't different in principle, but they are different in their superficial structure, so it is worth pointing out the difference in form explicitly.

In the variation we have just considered, we prove some property is true of 0 (or whatever is the first in the series - sometimes it is 1), and then we prove that if we assume that property is true of an arbitrary $n$, it will also be true of $n+1$. These two supports allow us to conclude that the property holds of every number.

The variation is sometimes called the method of *complete induction* (or: *strong induction*). In complete induction, we prove the base clause as before, and then we assume that the property holds of *every* number less than $n+1$. The difference is that instead of assuming the thesis *just* for the number preceding the given one, we assume it for *every* number less than the given one. Neither of these methods is any less correct than the other: it just happens that one form is convenient for some problems, and the other form is convenient for others. When dealing with strings of symbols in a language, the method of complete induction tends to be a little more useful.

In terms of an argument template, here are the two forms:

Ordinary mathematical induction (aka 'Weak Induction') :

**Base Case**: $C(x)$ holds of 0. [Or 1, or whatever the first element is in the given case.]

**Induction Step**: Take any $n$. If $C(x)$ holds of $n$, then $C(x)$ holds of $n+1$.

Complete induction (aka 'Strong Induction') :

**Base Case**: $C(x)$ holds of 0. [Or 1, or whatever the first element is in the given case.]

**Induction Step**: Take any $n > 0$ [Or $n > 1$, or whatever the first element is in the given case]. If $C(x)$ holds for every $x < n+1$, then $C(x)$ holds of $n+1$.

In both cases, if you can prove the Base Case and the Induction Step, you can conclude: $C(x)$ is true for every $x$

## 4.4 Fibonacci numbers

Here is an example to illustrate that sometimes the base case of a recursion / induction can involve more than one special case. Here I will use the style of "Complete Induction" described in section 4.3. A sequence of numbers that shows up in a surprising number of places in nature is the Fibonacci sequence:

1,1,2,3,5,8,13,21,34,...

Each number in the sequence is the sum of its two predecessors. The inductive definition of this one is a bit different from the ones we are used to, since we need to consider both $f(k)$ and $f(k-1)$ to define $f(k+1)$, and we need to define two starting points. But these are minor adjustments. Here's the definition:

$f(0) = 1$
$f(1) = 1$
$f(n+1) = f(n-1) + f(n)$ for $n > 1$

As usual, the inductive structure makes it possible to prove things. Here's a question: is the Fibonacci function $f$ we've just defined dominated by the exponential function $g(x) = 2^x$? That is, is it always the case that $f(x) < g(x)$ (except for $f(0) \leq g(0)$)? Think about how you might prove this. It's easy to confirm this for the base cases:

$f(0) = 1 = 2^0 = g(0)$ and
$f(1) = 1 < 2 = 2^1 = g(1)$

But once you've taken care of these specific cases, how can you go on? The basic observation is simple: use the elementary fact that if $a \leq c$ and $b < c$ then $a + b < 2c$. This tells us that repeatedly taking fibonacci numbers is not going to pull ahead of repeatedly multiplying by two. Try it for the first seven or so values of each sequence:

1,1,2,3,5,8,13...

1,2,4,8,16,32,64,...

No doubt by now you get the idea: the inductive argument is the way to make the idea rigorous.

We have already proven the base case for $f(0)$ and $f(1)$. We now **Assume** that for every n such that $1 < n < k$, we have $f(n) < 2^n$ (this is our *Induction Hypothesis*'). We want to show that $f(k) < 2^k$.

This follows immediately from the facts that (by the induction hypothesis) $f(k-1) < 2^{k-1}$ and $f(k-2) < 2^{k-2} (< 2^{k-1})$ and this instance of the "elementary fact" from above: $f(k) = f(k-2) + f(k-1) < 2^{k-1} + 2^{k-1} = 2 \times 2^{k-1} = 2^k$.

This proves the induction step, so we're done. Holllllla!!!

### 4.5 Something to try yourself

Here is the inductive definition of n! (read "n factorial"):
$1! = 1$
$(n+1)! = (n+1) \times n!$

That is, $n! = \underbrace{(n \times (n-1) \times \ldots 3 \times 2 \times 1)}_{n\ times}$

Exercise:

**Prove by induction:** For every n greater than 3, $2^n < n!$
We'll do this in class, as an analogue of a homework problem!

## 5 Proof By Induction: More Examples with no Numbers

### 5.1 Recursive definition of "descendent"

It will be good to consider one more concrete, down-to-earth example to dispel any air of mystery. A familiar example of a recursive definition is the definition of "descendent of x", where someone's descendents are their children, their children's children, etc.

Example: Not too long ago, there was a dispute about whether or not a particular currently living person was a descendent of Thomas Jefferson. But what does it mean to say that someone is a descendent of Thomas Jefferson? How do we define that set? (It'll be easier for our exposition if we tweak the sense of "descendent" a bit so that we count Thomas Jefferson as one of his own descendents.)

Informally, we can say: "Thomas Jefferson is one of Jefferson's descendents, and all the children of Jefferson are among Jefferson's descendents, and all the children of each of the children of Jefferson are among Jefferson's descendents, and so on..." Or we might put it a little differently: "You know. You take Jefferson, then his children, and then the children's children, and then the children's children's children, and keep on going like that."

To the logician, the crucial unclarity is in the meaning of "and so on", or "keep on going like that". Imagine we had to give explicit instructions to a computer to list the Jefferson descendents. How would we make the instructions adequately precise?[1]

One way to get oriented is to work backwards. Take some specific person Amy:

Under what conditions is Amy one of Jefferson's descendents?
Well, if at least one of her parents is a descendent of Jefferson.

Under what conditions is one of Amy's parents a descendent of Jefferson?
Well, if at least one of *their* parents is a descendent of Jefferson.

Under what conditions is one of Amy's parents parents a descendent of Jefferson?

$$\vdots$$

[1]To a philosopher, even after we provide precise instructions, there remain many questions about this "keep on going like that." How is it that a precisely stated rule is like rails laid to infinity?

This could in principle generate an infinite regress, except that there is one point where it stops: if we begin with a descendent of Jefferson, eventually we arrive at *Jefferson.* This "tracing back through the descendents" involves one "stipulate-the-base-case" principle—i.e. "x is a descendent of Jefferson if x is Jefferson"—and a further condition that gets repeated over and over again: "x is a descendent of Jefferson if x is the child of a descendent of Jefferson".

We can sum up the definition this way: "x is a descendent of Jefferson if and only if either i) x is Jefferson or ii) x is the child of a descendent of Jefferson." This may appear circular, but it isn't. I'll explain why it isn't circular in a moment. First let's recall the format that we use to express recursive definitions like this:

**Basis clause**: Jefferson is a decendent of Jefferson.
**Recursion clause**: If $x$ is the child of $y$, and $y$ is a descendent of Jefferson, then $x$ is a decendent of Jefferson.
**Closure clause**: nothing else is a descendent of Jefferson.

I'll say more about the closure clause later. I'm including it now "just for the record". It is often just left implicit.

It might seem as if the recursive definition has a basic, inherent defect. One way for a definition to be defective is for it to be *circular*. Say that I want to define "poefool" like this: "$x$ is a poefool if and only if $x$ is a bird and $x$ has feathers, and $x$ is a poefool." This is defective because the term I'm trying to define occurs in the clause that is supposed to define it. If I find a peacock and ask myself if it is a poefool, I can't use this definition to answer the question, because I would have to *already* know that it is a poefool to apply the definition.

At first glance, the definition of descendent might seem to have this defect, because the term being defined occurs in the definition itself. In fact, this isn't a problem, but you need to understand the structure of recursive definitions to see why not: note that each time you apply the definition, you get a bit closer to the base stipulation that Jefferson is a descendent of Jefferson, and that base stipulation doesn't presuppose any prior grasp of the definition of "descendent of Jefferson". (It does presuppose a prior grasp of *personal identity*—a thorny philosophical topic if there ever were one!)

Recursive definitions support the special kind of reasoning we've mentioned, called "Mathematical Induction". The "Jefferson's descendents" example gives a deceptively simple example of this. One test we would regard as conclusive for demonstrating that someone is a Jefferson descendent is a DNA test. We check a sample of Jefferson's DNA and a sample of the prospective descendent's DNA, and see if it matches in the relevant ways. Why should this tell us anything? Answer: because certain properties of DNA are passed on from parents to children. The reasoning involved is absolutely familiar, but logically it is a bit intricate, so I'll spell it out more ruthlessly than we usually do in daily life. I will label the parts of this bit of reasoning so that we can refer to them easily:

**Basis**: Jefferson has DNA property $\alpha$.
**Induction Step**: If a person has DNA property $\alpha$ then their children have DNA property $\alpha$.
Therefore:
Every descendent of Jefferson has DNA property $\alpha$.[2]

In most of the inductive arguments we'll study, we actually *prove* the two premises using mathematics or logic. In this imagined case the two premises would be established by empirical scientific investigation in microbiology. But the structure of the reasoning from the premises isn't affected by the way that the premises are justified.

## 5.2 Recursion and induction on languages with structure

If you speak English (or any other natural language) fluently, you are able to produce long and complicated sentences like:

"Incredibly, my neighbor was as irritated as a nest of disturbed bees when I asked reluctantly but insistently for the prompt return of the hedge trimmer I needed to tidy up the back yard before my son's already delayed birthday party."

We can understand long sentences like this, even when we have never heard them before. Of course, when they are unusually long, or the grammar is intricate, it may take some concentration to figure out what they say, but we can usually puzzle them out in no more than a few seconds. It's reasonable to conjecture that this ability to understand sentences we've never heard before is grounded in the fact that complex sentences are generated from simpler components by the application (possibly the repeated application) of rules. Here's an entertaining example.

Rule 1: Given two nouns (noun1) and (noun2) and a verb that takes two arguments,

[2]Of course, it matters both that Jefferson's descendents have a certain property and that people who aren't Jefferson descendents *don't* have the property. This inductive argument is only relevant to establishing the former. But that doesn't matter to the example.

we can form a sentence (noun1)(verb)(noun2). Example: given “bit”, “the dog” and “the cat” we can form “The dog bit the cat.”

Rule 2: Say we have a sentence of the form “(noun1)(verb)(noun2)”. We can form a noun phrase “The (noun2) the (noun1) (verb)”. Example: Given “The dog bit the cat.” we can get the noun phrase “the cat the dog bit”.

Similarly, we can form “The cat chased the rat” from “The cat”, “chased” and “the rat”, and then from “The cat chased the rat” we can form the noun phrase “The rat the cat chased”.

Notice that we can iterate these operations over and over: I can form “The dog bit the cat”, then form “The cat the dog bit”, then taking this as (noun1) I can form “The cat the dog bit chased the rat.” And we don't have to stop there. Further iterations give us some real prizes:

The cheese the rat the cat the dog bit chased ate had spoiled in the fridge.

The fridge the cheese the rat the cat the dog bit chased ate had spoiled in sat next to the stove.

$$\vdots$$

A digression: For entertainment purposes note that some fish, like angler fish, feed themselves by fishing for other fish. We can put this fact this way: Fish fish fish. And of course, following roughly the above pattern, we can note that there are some fish that are the targets of the fish who fish for fish. These are the fish fish fish. So understood, the sentence “Fish fish fish” and the noun phrase “Fish fish fish” support further iterations using the above rules, to get (meaningful) sentences like:

Fish fish fish fish fish fish fish .

The examples we have just spun out are all meaningful English sentences, though it may take some concentration to puzzle out what they mean if this is the first time you've seen them. The way we can understand them is that i) we trace back to the basic meaningful constituents (“cat”, “cheese” etc.) and ii) figure out the sentence structure by applying and re-applying the rules.

This is recursion in action. As we'll see as the course goes on, this makes it possible learn things about logic and languages by reasoning inductively.

### 5.3 Closure clause

I should say explicitly what the closure clause does in a recursive definition, since otherwise it won't get much attention. The closure clause is necessary because there will typically be many sets that contain the basis set and are closed under the conditions given by the recursion clause. We are interested in the *smallest* one. For example, if we assume (at least) for the sake of an example that there was a first pair of human beings, called Adam and Eve, and that every living person is descended from that pair, we can define the set of human beings who have lived or are currently living by this recursive definition:

**Basis clause**: Adam and Eve are human beings.
**Recursion clause**: If x is the child of a human being, then x is a human being.

But there are other sets containing Adam and Eve and closed under "x is a child of". For example, the set containing all the human beings who have ever lived, plus the Empire State Building also contains Adam and Eve and is also closed under "x is a child of". But the Empire State Building isn't a human being. The point of the closure clause is to exclude everything that isn't explicitly forced into the set we define by either the basis clause or the recursion clause, e.g. like the Empire State Building.

### 5.4 Palindromes

According to the most common definition, a *palindrome* is a string of letters that reads the same way backwards and forwards (ignoring spaces and punctuation). One simple example is "race car". More elaborate examples tend to sound a bit stilted, but they can be found. Some instances are: "Madam, I'm Adam", "A Man, A Plan, A Canal: Panama" and (thought of as spoken by Napoleon before his downfall and exile to the island of Elba): "Able was I, ere I saw Elba." For more, you can search on youtube for a truly odd video by the parodist Weird Al Yankovic making fun of a truly odd Bob Dylan video from the 1960's (Subterranean Homesick Blues).[3] The lyrics of Yankovic's song are a series of palindromes. ("Nurse, I spy gypsies: run!; "Do geese see God?" "Go hang a salami, I'm a lasagna hog." ...)

[3]For Yankovic's video, a youtube search on "Yankovic palindrome" will most likely turn it up. For full effect, you might want to watch part of the original Dylan video: A search on "Dylan Subterranean" should do.

FYI: Here is a link to the original (no palindromes):
https://www.youtube.com/watch?v=MGxjIBEZvx0

Here is a link to the parody:
https://www.youtube.com/watch?v=JUQDzj6R3p4

There are some trivial cases too: "I" is an English language palindrome, and so is "a". For bookkeeping purposes, we may want to speak of the empty string (a string with no letters at all, which I will name "$\epsilon$"). $\epsilon$ is a palindrome as well.

Notation: I will use "*" for the operation of concatenation (i.e. just writing one string then another). For example, aaba*aa is just aabaaa, the result of joining aaba and aa.

Rather than restricting attention to words that are meaningful in English, we can study the simpler case of palindromes in "language" consisting of *all* the strings in the alphabet $\{a, b, c\}$. As far as the structure of palindromes is concerned, one particular property is especially interesting: if you remove the first and last letter of a palindrome, you get another palindrome, two letters shorter. This fact gives us a way to inductively define "palindrome":

Call a string over $\{a, b, c\}$ an *inductive palindrome* if it satisfies the conditions:

**Base clause:** $\epsilon$, 'a', 'b', and 'c' are inductive palindromes.
**Recursion clause:** If $\sigma$ is an inductive palindrome, then $a * \sigma * a$, $b * \sigma * b$, $c * \sigma * c$ are inductive palindromes.
[Closure: Nothing else is an inductive palindrome in $\{a, b, c\}$].

What we want to do now is **prove** that every palindrome is an inductive palindrome. The proof exploits the recursive structure of "inductive palindrome" to give an inductive proof.

**Proof**
The induction will be on the length of the string $\sigma$.
**Base case:** Say that $\sigma$ is 0 symbols long. Then it is $\epsilon$, which is both a palindrome and an inductive palindrome.
**Inductive step**:
*Assume* (Induction hypothesis) that every palindrome of length less than k symbols is an inductive palindrome. Say that $\sigma$ is a palindrome with exactly k symbols.

If $\sigma$ has exactly one symbol, then it is one of ‘a’,‘b’, or ‘c’, which are inductive palindromes by definition. If $\sigma$ has two or more symbols, then we can remove the first and last letter, leaving a string $\sigma'$. Since $\sigma$ reads the same backwards and forwards, and $\sigma'$ comes from $\sigma$ by removing the first and last letter, $\sigma'$ reads the same backwards as forwards. That is, $\sigma'$ is a palindrome. Since $\sigma'$ is a palindrome, and it has fewer than k letters, then it is an inductive palindrome by the induction hypothesis.
Since $\sigma$ is a palindrome, the first and last letter must be the same, so it is either ‘a’, ‘b’, or ‘c’. So $\sigma = a * \sigma' * a$, or $\sigma = b * \sigma' * b$, or $\sigma = c * \sigma' * c$, where as we have shown $\sigma'$ is an inductive palindrome. So, by the inductive clause of the definition of inductive palindrome, $\sigma$ is an inductive palindrome.