COMS 3251 (Fall 2022) Recitation 1: Basic Math and Proof Review

Samuel Deng

September 2022

Intro. In this recitation, we will try to overview, in about an hour, the basic mathematical preliminaries needed to understand the material in COMS 3251 Computational Linear Algebra. Most, if not all, of this material should be review from either Calculus I (MATH 1101) or Discrete Mathematics (COMS 3203). If there are any topics we outlined today that you need further assistance on, don't hesitate to ask me (my email is `samdeng@cs.columbia.edu`) or any of the other TAs, via Ed! These notes will also be distributed.

1 Set Theory

1.1 Sets

To begin, we need a formal definition for a collection of objects. We refer to a collection of objects as a *set*. The objects in a set are referred to as *elements* of the set. Given a set A, we write $x \in A$ to denote that x is an element of A. If x is *not* an element of A, we write $x \notin A$. Typically, in this course, the elements of sets will be numbers or vectors. Sets, unlike lists (covered below), are *unordered* and *do not contain duplicates.*

Set operations. Just as we can combine numbers, we can combine sets using various set operations. Given two sets A and B, their *union* is written $A \cup B$ and is defined by asserting: $x \in A \cup B$ if $x \in A$ *or* $x \in B$ (or potentially both). Their *intersection* is written $A \cap B$ and is defined by asserting: $x \in A$ *and* $x \in B$. Union and intersection are symmetric. That is, $A \cup B$ is the same as $B \cup A$ (same with intersection). A non-symmetric set operation is the *set difference*, denoted $A \setminus B$ or $B \setminus A$. We assert that $x \in A \setminus B$ if $x \in A$ but $x \notin B$.

Defining sets. There are multiple accepted ways to state the contents of a set. We typically use curly braces (i.e. '{' and '}') to denote a set. One way to define a set is to simply list its elements in curly braces. For example, a set that just includes the numbers $1, 2$, and 3 can be defined as:

$$A = \{1, 2, 3\}.$$

Sometimes, if there are too many elements to explicitly write down, we use ellipses (i.e. '...') when it is clear what elements are omitted. For instance, if we want the numbers from 1

through 100 to be in A, we may write: $A = \{1, 2, \ldots, 99, 100\}$. Sometimes, it is more efficient and clear to provide a rule that determines elements of a set. You may have used the term *set builder notation* in COMS 3203 to refer to defining a set this way. For example, if we take $\mathbb{N} = \{1, 2, 3, \ldots\}$, then we can define:

$$B = \{x \in \mathbb{N} : x = 2k + 1, \text{where } k \in \mathbb{N}\}.$$

The set B is then the set of all odd natural numbers greater than or equal to 3. We will also see this notation used another way, where we write: {expression with free variables : condition on the free variables}. For example, $\mathbb{Z}_O = \{\ldots, -5, -3, -1, 1, 3, 5, \ldots\}$ is the set of odd numbers and $\mathbb{Z}_E = \{\ldots, -4, -2, 0, 2, 4 \ldots\}$ is the set of all even numbers then:

$$C = \{a + b : a, b \in \mathbb{Z}_O\} = \mathbb{Z}_E.$$

Above, C is the set of all sums of two odd integers. Because every even number can be written as a sum of two odd numbers, this is the same as $\mathbb{Z}_E$.

There are a couple of infinite sets of numbers that we'll be using very frequently throughout this course that you've likely seen before. First are the *natural numbers*, $\mathbb{N} = \{1, 2, 3, \ldots\}$. On top of that, if we want to be able to perform subtraction and have an identity element for addition, we need the *integers*, $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$. If we'd like to multiply and divide, we need to extend to the *rational numbers*, $\mathbb{Q} = \{\frac{p}{q}$ where $p, q \in \mathbb{Z}$ with $q \neq 0\}$. Finally, we patch up all the 'holes' in $\mathbb{Q}$ with the real numbers, $\mathbb{R}$, which includes irrational numbers such as $\sqrt{2}$. We will not go into how to formally define $\mathbb{R}$, as that is outside the scope of this course (take MATH 4061 Modern Analysis I for that).

One more set that we have a shorthand for is the *empty set*, which is the set that contains no elements. We denote the empty set as $\emptyset$.

Subsets and set equality. We also need a way to say that a set is fully contained in another set. We write $A \subseteq B$ to indicate that A is a *subset* of B. Specifically, this means that every element of A is an element of B. Formally, this asserts: if $x \in A$ then $x \in B$. Using this, we can define set equality: $A = B$ means that $A \subseteq B$ *and* $B \subseteq A$. That is, we say that two sets are equal if and only if they are subsets of each other.

Because sets are *unordered,* the sets $A = \{1, 2, 3\}$ and $B = \{3, 2, 1\}$ are the exact same, i.e. $A = B$. Also, because sets *do not contain duplicates*, the sets $C = \{1, 1, 2, 3\}$ and $D = \{1, 2, 3\}$ are also the exact same. When we write sets explicitly with curly braces, writing out the duplicate elements is generally bad form. Note, however, that sometimes the "no duplicates" rule might be a bit pedantic, and this feature of a set is often overlooked or ignored for convenience (e.g. when we refer to a "data set" in data science contexts, we do not usually throw away the extra copies). In this course, it'll be clear if this is the case.

Python. An important part of this course is using Python and NumPy to apply and learn the linear algebra concepts we'll discuss. We will go over Python basics in future recitations, but, for the time being, we will just note that Python uses similar syntax to define a set: "curly braces." We can also use the `set` keyword as a constructor. Sets work similarly in Python: they are unordered collections of elements. In the code below, the `#` symbol denotes a comment.

```
example_set = {1, 2, 3} # example set 1
example_list = [1, 2, 3, 4, 4]
example_set2 = set(example_list)   # example set 2
```

1.2 Lists and Tuples

A *list* is an *ordered* collection of objects (unlike a set, which is unordered by definition). When we *do* care about the order of elements in a collection, we use a list. Sometimes, we know beforehand the length of a finite list and would like to specify it. An *n-tuple* is an ordered list of exactly n elements. For example, ordered pairs in the Cartesian plane we know and love are just 2-tuples of real numbers.

Defining lists. To define a list, we use parentheses ("round braces") to enclose our elements. For instance, the list of length 4 containing the first four natural numbers is simply:

$$A = (1, 2, 3, 4).$$

Because order matters, A is *not* the same as the list $B = (4, 3, 2, 1)$.

Concatenation. Sometimes, we want to extend a list by adding all the elements of another list to the end of it, in order. This operation is known as *concatenation.* Although notation for this operation tends to vary from resource to resource, in this course, we will typically use the following. For two lists $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots b_n)$ the concatenation of B to A is denoted $(A, B) = (a_1, \ldots, a_m, b_1, \ldots, b_n)$.

Cartesian product. Now that we know what a list and a tuple is, we can define another set operation, the *Cartesian product.* For two sets A and B, the Cartesian product is the set

$$A \times B = \{(a, b) : a \in A \text{ and } b \in B\}.$$

Notice that we are using our second version of set builder notation here. This is the set of all 2-tuples we get by ranging over all the elements of A in the first entry and all the elements of B in the second entry. Intuitively, think of forming a table where all rows are elements of A and all columns are elements of B. The Cartesian product contains all row-by-column entries of the table.

Note that the "and" in the condition for the Cartesian product is often omitted but implied. In general, a comma in a condition can be read off as a logical "and."

Python. Although for our purposes in math, lists and tuples are pretty much interchangeable, they are not in Python. We will go over this in more detail in future recitations, but the main difference is that a `list` is extendable and mutable, while a `tuple` is not. In analogy to C-based languages like Java, a `list` in Python is essentially an `array`. They are fundamentally different datatypes and their declaration syntax is different as well (a `list` is declared with square brackets and a `tuple` is declared with parentheses). As a small sneak peak, here's some code.

```
example_list = [1, 2, 3]
example_list2 = [4, 5, 6]
```

```
concatenated_list = example_list.extend(example_list2)
print(concatenated_list)    # prints [1, 2, 3, 4, 5, 6]
print(concatenated_list[4]) # prints 5

example_tup = (1, 2)
print(example_tup[0])       # prints 1
```

Also, like the vast majority of programming languages, Python is zero-indexed, meaning that the first element of an ordered collection is accessed by using `0`. Math, however, is one-indexed: when we refer to "the first element" or "element one" of a list, we are really referring to the first element. This will likely be a source of bugs in the future, so be on the lookout!

2 Functions

You are also likely familiar with functions from previous math classes. We'll quickly go over the relevant terminology for functions. First, a definition. Given two sets, A and B, a *function* from A to B is a rule or *mapping* that associates *each* element $x \in A$ with a *single* element in B. In this case, we write $f : A \to B$. Given an element $x \in A$, the expression $f(x)$ is used to represent the element of B associated to x by f.

There are a couple of things to note in this definition, as review. First, notice that we state that a function is a "rule or mapping." This liberates functions from just being "formulas" with rules that are algebraically well-defined, such as $f(x) = x^2 + 5$ or $f(x) = \sqrt{x^3 + 8}$. The word "mapping" suggests a broader range of possible functions, including ones we don't readily have a formula for.

Second, we note that *each* element of the set A must be associated with some element in B. There cannot be any elements of A that are "unassigned" by f, by definition.

Third, each element of A is assigned to a *single* element in B. This means that functions do not, by definition, take elements in A to multiple elements of B. So when we say $f(x)$, we unambiguously mean one, and only one, element of the set B.

Function terminology. There are a couple of important terms to recall in the context of functions. For a function $f : A \to B$ mapping A to B, the set A is the *input space* of f (in other courses, this may have been called the *domain*). The set B is the *output space* of f (in other courses, the *codomain*). The *image* (also referred to as the *range*) of f is not necessarily equal to B, but refers to all the elements of B that are "hit" by f: $\{y \in B : y = f(x) \text{ for some } x \in A\}$. The image is a subset of B. On the other hand, for a given subset $S \subseteq B$, the *preimage* of S is the set of all elements of A (possibly empty) that get mapped into S. Formally, the preimage of S is $\{x \in A : f(x) = y, \text{ where } y \in S\}$. Any preimage of f is a subset of A. Sometimes, we will write the image as $f(A)$ and the preimage as $f^{-1}(S)$ for a set $S \subseteq B$.

Function declarations. As with sets, there are multiple ways to declare a function. First, when we declare a function, we should make the function's input space and output space

explicit, usually by writing the declaration $f : A \to B$, where A and B are the input space and output space, respectively. Then, if we have a rule or formula for the function, we can just write $f(x) = rule$, e.g. $f(x) = x^2 + 4$. Sometimes, we write this all in one chunk, like this:

$$\begin{aligned} f : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto x^2 + 4 \end{aligned}$$

But, as discussed above, not all functions have a neat rule that tells us how to map inputs to outputs. In some cases, our sets are small enough to manually write out what maps to what. For example, we can exhaustively define a function as follows.

$$\begin{aligned} f : \{1, 2, 3\} &\to \{1, 2, 3\} \\ 1 &\mapsto 1 \\ 2 &\mapsto 1 \\ 3 &\mapsto 2 \end{aligned}$$

But it'll be rare that we'll map from finite sets to finite sets, so this quickly becomes an unwieldy way to declare a function. Oftentimes, it'll be most convenient to define functions piecewise. This notation is likely familiar from other classes. For instance, the absolute value function $f : \mathbb{R} \to \mathbb{R}$ can be defined piecewise as follows.

$$f(x) = |x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

3 Logic and Proofs

Finally, we'll cover the basics of logic and proof writing that should be review from COMS 3203 Discrete Mathematics. Note that we'll be omitting much of the basic propositional logic material from COMS 3203 with the assumption that you're already comfortable with it (i.e. truth tables, logical operators, etc.). The focus will be on reviewing how to read and write proofs that we'll commonly see in this course: direct proofs, proofs by contradiction, and proofs by induction.

3.1 Terminology

First, we go over some terminology that might help with parsing and writing proofs in this course. A *proposition* is an atomic statement that is either true or false. For instance, "it is 11:05 PM right now" is a proposition that is true exactly one minute each day and false otherwise. In mathematics, we typically deal with propositions that state that abstract objects have certain properties, like x is an even number.

For two propositions p and q, we say that p *implies* q if it is the case that, whenever p is true, q is also true. Shorthand, we write $p \implies q$, but typically we write this in English: "if

p, then q." Even stronger are *if-and-only-if* statements, denoted $p \iff q$. An if-and-only-if statement being true means that $p \implies q$ *and* $q \implies p$. To show some implication $p \implies q$, we need to show that, in all the cases where p is true, q is also true. We need not concern ourselves with the cases when p is false, as an implication doesn't "cover" those cases.

To build up linear algebra (and any other mathematical theory), we typically go from *definitions* of mathematical objects to *theorems* and their associated *proofs.* A mathematical *definition* is a set of conditions for an object. By itself, a definition is *not* a proposition. But saying an object "x is d", where d is the object we're defining, *is* a proposition because it can either be true or false. For example, an *even number* is defined to be an integer $x \in \mathbb{Z}$ such that $x = 2k$ for some $k \in \mathbb{Z}$. By itself, this is not a proposition (what does it mean to say "even number is true?"), but "5 is an even number" is a proposition (that is false). Throughout this note and the class lecture notes, definitions are *italicized and underlined*. When you see a definition, a good thing to first do is to test out a couple of example objects and see if the definition applies to them (i.e. "Is 2 an even number?").

A *theorem* is a true proposition, usually of some interest to the overarching mathematical theory. Theorems come with *proofs*, which are just arguments that show that a statement is, without a doubt, true. Proofs are typically written in grammatically correct English with mathematical vocabulary, and they should be written clearly and concisely. Remember that a proof is meant for a reader, at the end of the day! We will review the three most common types of proofs that will come up in this course in Section 3.3.

3.2 Quantifiers

On top of propositional logic, we will be using logical quantifiers. There are two logical quantifiers: *for all* (symbolically, $\forall$) and *there exists* (symbolically, $\exists$). Many statements in this course will use one of these quantifiers, and some statements will use both.

The *for all* quantifier (also known as the *universal quantifier*) asserts that some property (also called a *predicate*) is true of *every* object in a certain domain. In math, when we say *every*, we really mean it – for all quantifiers allow no exceptions. Mathematically, the "certain domain" will be a set, so we will always be referring to every object in a set when we use the for all quantifier.

For example, take the statement: "for all natural numbers n, $2n$ is also a natural number." The domain of quantification is $\mathbb{N}$, the natural numbers, and the predicate is the property: "is (also) a natural number." This statement happens to be true. No matter what natural number $n \in \mathbb{N}$ we try, the predicate holds. On the other hand, the statement "for all natural numbers n, $2n > 2 + n$" is false. Consider the natural number $1 \in \mathbb{N}$ and plug it in – immediately, we see that $2 > 3$ is false. Because one of the natural numbers doesn't work, this statement is false. So, when we need to check if a "for all" quantified statement is true, we need to check for every single object in the domain. To show that it's false, we just need to find one counterexample in the domain.

The *there exists* quantifier (also known as the *existential quantifier*) asserts that *at least* one object in a certain domain obtains some property (the *predicate*). As opposed to the "for

all" quantifier, the "there exists" quantifier only needs a single object in the domain to have some property.

For example, take the statement "there exists a real number x such that $x^2 = 2$." Here, the domain of quantification is $\mathbb{R}$ and the predicate is the property for x: $x^2 = 2$. This statement happens to be true because we can find a real number that satisfies $x^2 = 2$, namely the number $\sqrt{2} \in \mathbb{R}$ (the number $-\sqrt{2}$ also works). However, the statement "there exists a natural number x such that $x^2 = 2$" is false. This is because it is impossible to find a natural number that satisfies $x^2 = 2$ ($1^2 = 1$ and $2^2 = 4$, and the natural numbers don't include anything in between 1 and 2). Thus, to show that a "there exists" quantified statement is true, we need to come up with just a single object in our domain of quantification (though there may be more than one) that satisfies our predicate. To show it's false, however, we need to show that the predicate doesn't hold for *any* of the objects in our domain.

Negating quantifiers. From our previous discussion, you may have noticed that showing that a "for all" quantified statement is true requires the same work as showing a "there exists" quantified statement is false (and vice versa). This is because, logically, negating a "for all" quantifier means we "flip" the "for all" into a "there exists" and negate the predicate. You may have seen this as:

$$\neg(\forall x \in S, P(x)) \iff \exists x \in S, \neg P(x).$$

Likewise, negating a "there exists" quantifier means we "flip" the "there exists" into a "for all" and negate the predicate:

$$\neg(\exists x \in S, P(x)) \iff \forall x \in S, \neg P(x).$$

Nesting quantifiers. In certain statements, we will *nest* quantifiers to make assertions that are more logically complex. In this course, we'll likely only see two quantifiers nested together, typically in the form "for all $x \in S$, there exists..." or "there exists $x \in S$ such that, for all..." Whenever we see quantifiers nested like this, the order matters. In the "for all-there exists" case, we are stating something universal about a domain, and the task is to construct something that depends on every object of that domain. In the "there exists-for all" case, we are constructing something such that it has a property applying to all objects of some domain.

For example, consider the statement "for all positive real numbers, there exists a smaller positive real number." A bit more formally: for all $x \in \mathbb{R}$ such that $x > 0$, there exists $y \in \mathbb{R}$ such that $x > y > 0$. To prove that this is true, think of being handed an arbitrary real number x. Your job is to produce some y that is both smaller than x and greater than 0. Because dividing any real number in half still gives us a real number, we can choose $y = x/2$. $x > x/2 > 0$ if $x > 0$, so we have proven the statement is true.

For the other case, consider the statement "there exists a real number x such that, for all real numbers y, $x + y = y$." To show that this is true, we just need to find a single real number x, but we need to show that $x + y = y$ for *every* real number y. Choosing $x = 0$,

the additive identity of the reals, will suffice. We know that $0 + y = y$ for any $y \in \mathbb{R}$, so the statement is true.

3.3 Proof Techniques

Finally, we'll go over the proof techniques needed to understand and write the proofs in this course. This note assumes that you are already somewhat acquainted with these proof techniques from Discrete Mathematics (COMS 3203), so the following paragraphs should just serve as reminders. However, if you feel you need more help, don't hesitate to use the TAs as resources. You may not be proficient at reading and writing proofs, even after taking Discrete Math, but that is okay – one of the main goals of this course is to develop your mathematical fluency.

A mathematical *proof* is the bread and butter of building up solid mathematical theory. Most (if not all) major results in this course will have an accompanying proof that verifies the truth of the result. Good proofs tend to also elucidate *why* a certain result is true, so understanding proofs will, at times, also help you understand the intuition and importance of certain results more deeply.

Direct proof. The most basic form of proof is the *direct proof*, which is used to prove statements of the form "if p, then q" or "p implies q" (symbolically, $p \implies q$). Here, p can stand in for multiple propositions, also called the statement's *hypotheses* or *premises*. The same goes for q, the statement's *conclusions.* A direct proof starts by assuming that the hypotheses are true. From those true propositions, a direct proof proceeds through a series of logical deductions until the conclusions are shown to be true.

In a good chunk of the direct proofs you will see and write in this course, a common strategy is to "unravel" the definitions in the premises and the conclusions so there are more concrete symbols to manipulate and examine. Then, by applying already known results, algebraic techniques, or logically valid observations, you can typically arrive at the definitions provided by the conclusions. For example, the definition of "even number" is $x \in \mathbb{Z}$ such that $x = 2k$ for some $k \in \mathbb{Z}$. Unravelling "even number" into a more algebra-friendly quantity $2k$ is a first step towards proving something about x.

Direct proofs can also apply to the logically quantified statements mentioned in Section 3.2. With "for all $x \in S$" statements, a direct proof starts by considering an arbitrary x that has all the properties enjoyed by objects in the set S. Then, as before, logical deductions should entail that x satisfies the predicate. For example, a statement that starts with "for all $x \in \mathbb{R}$" allows us to use any and all properties of the real numbers, $\mathbb{R}$, to prove what we need about x. With "there exists $x \in S$" statements, we need to first construct some object $x \in S$ (sometimes it takes some work to show that x indeed satisfies the properties that put it in S). Then, we proceed to use whatever is true about the object we constructed (we have control over this!) to reach the predicate.

Also, recall that to prove a statement of the form "p if and only if q," it suffices to provide a direct proof for both directions: "if p, then q" and "if q, then p."

Example (direct proof). Let us try to prove the statement:

The square of any odd number is also odd.

This is the same as the statement: *for all odd $n \in \mathbb{Z}$, n^2 is odd* or *if $n \in \mathbb{Z}$ is odd, then n^2 is also odd.* Note that we can already assume that $n \in \mathbb{Z}$ because implicit in "odd number" is that the number is an integer. Begin by "unraveling" the definition in the premise. Recall from Discrete Math or previous classes that an odd number is defined as an $n \in \mathbb{Z}$ such that $n = 2k + 1$ for some $k \in \mathbb{Z}$. We aim to show that n^2 is odd. Using the same definition of odd number, this is the same as stating: $n^2 = 2\ell + 1$ for some $\ell \in \mathbb{Z}$. Thus, our goal is to find some $\ell \in \mathbb{Z}$ that satisfies this.

We know how to square numbers from basic algebra, so let's try to do that to n :

$$\begin{aligned} n^2 &= (2k+1)^2 \\ &= (2k+1)(2k+1) \\ &= 4k^2 + 4k + 1 \\ &= 2(2k^2 + 2k) + 1. \end{aligned}$$

All the equalities here are logical deductions that just follow from rules we learned in basic algebra that are likely second nature by now. To finish the proof, we let $\ell = 2k^2 + 2k$, so $n^2 = 2\ell + 1$. We know that $2k^2 + 2k \in \mathbb{Z}$ as well because $k \in \mathbb{Z}$, and $\mathbb{Z}$ is closed under the algebraic operations of squaring, adding, and multiplying. Therefore, $\ell \in \mathbb{Z}$, so $n^2 = 2\ell + 1$ for an $\ell \in \mathbb{Z}$, i.e. it is odd.

This proof was a bit pedantic just for the purposes of review, and some of the more obvious steps could be omitted (e.g. "$\mathbb{Z}$ is closed under the algebraic operations of squaring, adding, and multiplying" to justify that ℓ is indeed an integer). Knowing what can and cannot be omitted mostly comes from experience and knowing the audience for the proof you're writing.

Proof by contradiction. When direct proof fails or the way forward is unclear, we may try other proof techniques. *Proof by contradiction* is a proof technique that proves a statement is true by showing that assuming the statement is false leads to a logical contradiction. Assuming a statement is false involves negating the statement through the usual rules of logic. For instance, if our statement is an implication, $p \implies q$, the negation is $p \wedge \neg q$. To prove an implication through contradiction, we assume $p \wedge \neg q$ is true, see what logical deductions come from that, and eventually arrive at some contradiction.

The same general strategies apply to proof by contradiction. Typically, when a statement is ripe for a proof by contradiction, the contradicted statement contains a definition that is easier and more concrete to work with. We will see this in the upcoming example, where we prove that $\sqrt{2}$ is irrational. "Irrational" just means that $\sqrt{2} \notin \mathbb{Q}$, which doesn't give us much to work with. However, we know a clear definition and many properties for rational numbers, so it is desirable to work with the (false, but assumed true for sake of contradiction) statement $\sqrt{2} \in \mathbb{Q}$.

Example (proof by contradiction). We prove the classic statement:

The square root of 2 is irrational.

We need to show that $\sqrt{2} \notin \mathbb{Q}$. We prove this by contradiction. As a first step, we assume that the proposition is false, so assume $\sqrt{2} \in \mathbb{Q}$. By definition of $\mathbb{Q}$, we can express $\sqrt{2}$ as the ratio of two integers, a and b, such that a and b are in lowest terms.

$$\sqrt{2} = \frac{a}{b}, \text{ s.t. } a, b \in \mathbb{Z} \text{ and the GCD of } a \text{ and } b \text{ is } 1$$

If a and b were not in lowest terms, we could divide each number by their greatest common divisor to reduce $\frac{a}{b}$. This also implies that at least one of a or b is odd, because if both a and b were even, then $\frac{a}{b}$ could be reduced further by dividing by 2.

Simple algebra gives us $2b^2 = a^2$, so a^2 is even. Because any even number squared is even (if you don't believe this, try to write a direct proof for this similar to the one above!), a is also even. This implies b is odd, from the previous paragraph.

Because a is even, we can write $a = 2k$ for some $k \in \mathbb{Z}$ from definition. Plugging this in to our above equality and doing a bit of algebra,

$$b^2 = 2k^2.$$

But then b is also even! We previously established that b was odd, so b is both odd and even. This is a contradiction, as no number can be both odd and even. Arriving at this contradiction finishes the proof, and it follows that $\sqrt{2} \in \mathbb{Q}$ is false, and thus $\sqrt{2} \notin \mathbb{Q}$, as we originally desired.

Proof by induction. The final proof technique that will commonly appear in this course is *proof by induction.* A proof by induction proves some statement $P(n)$ for every natural number $n \in \mathbb{N}$. Statements we prove by induction are typically of some form where the proposition can be "indexed" in a way that we can put in correspondence to the natural numbers. Because proving some statement $P(n)$ holds for every natural number $n \in \mathbb{N}$ involves infinite cases $(P(1), P(2), P(3) \dots)$, induction leverages the fact that if $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$. This is usually introduced as a metaphor to tipping dominoes over because the hard work of a proof by induction is in proving that if $P(n)$ holds, then $P(n + 1)$ holds (i.e. if we topple any one domino, the next one will also topple).

A proof by induction can be seen as having two parts. First, the *base case* of a proof by induction involves showing that $P(1)$ is true (sometimes we may start at a different index – the base case always involves that starting index). This is usually a straightforward check. Second, the *inductive step* involves proving that *if* $P(n)$ is true, then $P(n + 1)$ is also true. Any proof by induction will heavily leverage the fact that $P(n)$ is true, so the brunt of the work is making deductions that come from assuming that $P(n)$ is true.

Note that, sometimes, proof by induction is presented as three steps, splitting the inductive step into an *inductive hypothesis* and an *inductive step*. In this case, the inductive hypothesis just involves assuming $P(n)$ is true, and the inductive step is showing that $P(n + 1)$ is true as a consequence. This is just a matter of semantics.

Example (proof by induction). We prove the following statement, allegedly proven by Gauss in elementary school:

The sum of n consecutive natural numbers is $\frac{n(n+1)}{2}$.

Formally stated, the proposition is:

$$P(n) : 1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

First, we prove the *base case*, which is $P(1)$. This is trivial to check: $1 = \frac{1(2)}{2}$. Then, assume that $P(n)$ is true. That is, we assume that $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$. We aim to show that $P(n+1)$ is true as a consequence:

$$P(n+1) : 1 + 2 + \cdots + n + (n+1) = \frac{(n+1)(n+2)}{2}.$$

Typically, proof by induction involves seeing the "pattern" laid out by $P(n)$ that is part of $P(n+1)$. In this case, we see that we can group the first n summands and use our formula from $P(n)$ on them, as follows:

$$(1 + 2 + \cdots + n) + (n+1) = \frac{n(n+1)}{2} + (n+1).$$

With a little bit of algebra,

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

That last equality is exactly what we wanted, so $P(n+1)$ is true. Therefore, by induction, the claim is proved for all $n \in \mathbb{N}$. We can state that, for any $n \in \mathbb{N}$, the sum $1+2+\cdots+n = \frac{n(n+1)}{2}$.