Introduction to Mathematics: Week 3

September 25, 2020

1 Induction

There are two types of induction arguments: strong and weak induction. You will most often use induction when proving statements which are true for all natural numbers, so if you see a formula which depends on some $n \in \mathbb{N}$, think "induction" first.

Theorem 1 (Weak Induction)**.** Let $P(n)$ be a statement defined for each $n \geq n_0 \in \mathbb{N}$ such that

1. If n_0 is when the base case occurs, $P(n_0)$ is true.
2. for each $n \in \mathbb{N}$, if $P(n)$ is true then $P(n+1)$ is true.

Then $\forall n \in \mathbb{N} : P(n)$.

Theorem 2 (Strong Induction)**.** Let $P(n)$ be a statement defined for each $n \geq n_0 \in \mathbb{N}$ such that

1. For the base case n_0 we have $P(n_0)$ is true.
2. for each $n \in \mathbb{N}$, if $P(n_0), P(n_0+1), \ldots, P(n)$ are true, then $P(n+1)$ is true.

Then $\forall n \in \mathbb{N} : P(n)$.

Remark. The difference between strong and weak induction is that in weak induction, we only assume that $P(n_0)$ is true, not all of the $P(k)$ for $k < n$ as we do in strong induction. If a proof requires more information, then that is when strong induction would be preferred over weak induction.

Remark. The base case may not always be 1, so be careful!

Exercise 1. Prove that $\sum\limits_{j=1}^{n} j = \frac{n(n+1)}{2}$.

Exercise 2. Show that $n! \leq n^n$, $\forall n \in \mathbb{N}$.

Theorem 3 (Fundamental Theorem of Arithmetic)**.** Every integer $n \geq 2$ can be uniquely written as the product of prime numbers.

Exercise 3. Prove that for all natural numbers $n \geq 2$, $n^2 \geq n+2$.
Hint: When trying to prove an inequality $a \leq b$, it can be helpful to find some c s.t. $a \leq c \leq b$, so that $a \leq b$ follows by transitivity of inequalities.

Exercise 4. Prove that $n! > 2^n$ for $n \in \mathbb{N}$, $n \geq 4$.

2 Solutions

Exercise 1. We will proceed via induction.

Base case: First, we see that $\sum_{j=1}^{1} j = 1 = \frac{1 \cdot 2}{2}$, so the statement is true for $n = 1$.

Induction step: assume the formula holds for n, i.e. $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$. We want to show that it also holds for n + 1, i.e. that

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

Indeed,

$$\sum_{j=1}^{n+1} j = \sum_{j=1}^{n} j + (n+1)$$

By our induction assumption that $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$. Therefore,

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

And thus we have shown that the formula holds for all n. □

Exercise 2. Base case: When $n = 1$,

$$1! = 1 \leq 1^1 = 1,$$

so it holds for the base case.

Induction step: Assume that $n! \leq n^n$. We want to show $(n+1)! \leq (n+1)^{n+1}$. Note that $(n+1)! = (n+1)n!$ By our induction assumption, we then have

$$(n+1)n! \leq (n+1)n^n \leq (n+1)(n+1)^n = (n+1)^{n+1}.$$

□

Theorem 3. We will proceed by strong induction.

Base Case: For $n = 2$, we have that 2 is prime so this is clear.

Inductive Step: Suppose the statement is true for $k = 2, 3, 4, \dots n$. We want to show it is also true for $n + 1$.

If $n+1$ is prime, then we are done. Otherwise, $n+1$ has a smallest prime factor, which we will denote p. Let $n + 1 = p \cdot N$. Since $N < n$, by our induction hypothesis, N can be written as the product of prime numbers. But this means that $n + 1 = p \cdot N$ can be written as a product of primes.

But why is this factorization unique? Try to see if you can figure it out using proof by contradiction. □

Exercise 3. Base case: For $n = 2$ the inequality becomes

$$2^2 = 4 \geq 2 + 2 = 4$$

which is true.

Induction step: Assume $n^2 \geq n + 2$ for some $n \in \mathbb{N}$. We want to show that this implies $(n + 1)^2 \geq (n + 1) + 2 = n + 3$. We compute:

$$(n + 1)^2 = n^2 + 2n + 1 \geq n + 2 + 2n + 1 = 3n + 3 \geq n + 3$$

where the last inequality holds since $n \geq 0$. □

Exercise 4. Base case: Note that here our base case is not n=1, but rather $n = 4$. When $n = 4$, the left hand side is $4! = 24$ and the right hand side is $2^4 = 16$ and indeed we have that $24 > 16$ so the inequality holds.

Induction step: Assume the inequality holds for a fixed $n \geq 4$. Observe that $(n + 1)! = n!(n + 1)$. By induction hypothesis, $n > 2^n$, and since $n \geq 4 \Rightarrow n + 1 > 2$, we have

$$2^n(n + 1) > 2^n(2) = 2^{n+1},$$

and so the inequality holds for $n+1$. Therefore, $(n+1)! > 2^{n+1}$ for all $n \geq 4$ by induction. □

3 Latex and Presentation

Most classes have weekly problem sets for you to hand in, and they‘re arguably the biggest way you learn the material. They’re also part of your grade, so some TA will be marking whatever you hand in. Making your homework easy to read will be good practice for you, and will avoid making the grader annoyed with you. Some easy guidelines to follow are:

- If you can, type up your homework in LaTeX, and if not make sure your handwriting is clean and legible. Please, please, please don’t use Word. If you use handwriting, rewrite your answers on a new sheet of paper, especially if you’ve erased and rewritten sections.

- Try to get used to using LaTeX as soon as possible, since the earlier you start the more natural it will be.
- Make sure your logic is clearly organized. Your proof should read like a paragraph, rather than putting different ideas as you think of them on the page and using arrows to point to what you want to reference (this happens more often than you would think, and it‘s very hard for a grader to read).
- Staple your problem sets! Loose pages will get separated in the drop box.
- Remember to write your name and the number of the problem set on the page (again, you'd be surprised how often people don't do this)

Here we shall introduce you to the world of LaTeX, pronounced as "lah-tech" or "lay-tech", is a document formatting system for high quality type-setting. In other words, we will be able to construct pdfs with our ideas in however way we want our ideas to be expressed. It is namely, not:

- a word processor
- a calculator
- a place to do your homework

The idea is to present your homeworks, exams, papers, articles, in a concise and professional manner. The question of which design is up to the author. That's the beauty of LaTeX.

3.1 Importance of Notation

Now, typesetting is good for a number of reasons. One, is that you can copy and paste on a computer and you can't by hand. Another, is that you can create a nice, professional file that looks good (especially the new generation's tendency to have bad handwriting). The biggest reason, and the reason why we add this into our lecture series, is that **presentation is everything.**
I do not mean that one should add a nice fancy cover page and a bibliography and hyperlink your homework. I mean that it allows one to disassociate the act of doing a proof with the presentation of a proof, the latter of which is essential in making a proof. This is important because even when one understands how to do a proof, if one were to present the proof incoherently, imprecisely, or unprofessionally, the proof would not be communicated. **An uncommunicative proof is no proof at all.** It doesn't matter if you solved the Millenium problems in your head if you can't tell the world about it. That's the principle at least.

3.2 Using Latex

In the end, we are creating code and compiling it to create pdfs. Therefore, we need a way to edit our code and compile it (which in our case means that we need to convert our code into a pdf document). Like all code, the language was not written perfect the first time around. There are numerous packages that are very useful (and essential!) that require downloading or at least telling the machine to look for these packages.

Creating the file

Command Line

For those familiar with the command-line, the standard command is “pdflatex”. This may however change if you have a particular need for “LuaLatex” or “Bibtex”, which may be useful if one needed complicated diagrams or bibliographies. So, for example, if our latex file has the file name “filename.tex", then we can make a pdf by going to the directory (folder) that our tex file lives in, and entering,

```
pdflatex filename.tex
```

For a guide to using the terminal, no matter what operating system you are running, see this crash course. For those who don’t have time for that, know that “cd foldername” lets you travel to folders, “cd .." lets you leave a folder, and “ls" gives you a list of the files in a folder.

Overleaf

Overleaf is the free online server that allows you to compile latex files and creates your pdf as you make. It is also great for sharing with friends to collaborate on things. In particular, their guide to making latex documents are very essential to optimizing your efficiency. One of the great things to note is that Overleaf stays mostly up to date with their packages! So most fairly known packages can be called without any kind of downloading. We’ll talk about that later.

Latex Compilers

For those who are not so familiar with computers, you may find it easier to use a Tex distribution and compile yor code. The generally good ones are, “MiKTeX” for Windows, “TeX Live” for linux, “MacTeX” for macOS. There are also editors: AUCTEX, GNU TeXmacs, Gummi, Kile, LaTeXila, MeWa, TeXShop, TeXnicCenter, Texmaker, TeXstudio, TeXworks Freeware: LEd, WinShell , Inlage, Scientific WorkPlace, WinEdt, with the last three being proprietary.

An Example

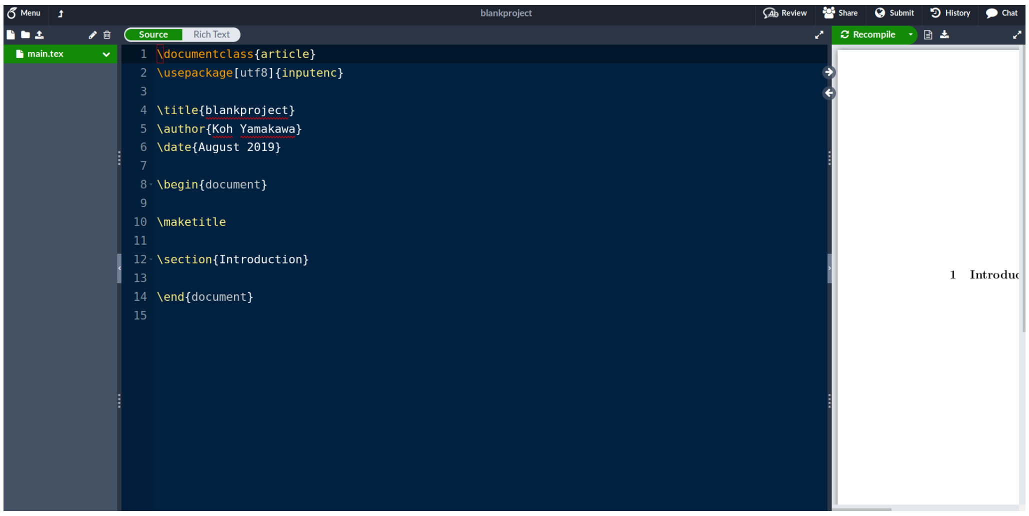
Now, let's begin by making your first latex code. We will be using Overleaf for our example because then we do not have to waste anytime downloading packages. Now, everyone should get on their computers and go to,

$$www.overleaf.com$$

and make an account. Using your university email may be good on the off chance that Columbia becomes a member and we can get free professional memberships.

1. Click new project. Let's name it "blankproject"
2. Return arrow in top left will bring you back to the main page.
3. Top left "Menu" are settings for the source code. You can also upload your project to your github (if its been connected), look up shortcuts (Hotkeys), and other nice management settings.
4. Left panel is to organize your files, in case you want to refer to other bibliography files, jpegs, etc. These can be uploaded with the upwards arrow narrow bar.
5. Right panel is the compiled pdf. Note the Recompile options of Autocompiling, Fast compiling, and the options to see the logged errors (overleaf has code to ignore many errors, other compilers are not so nice), and the download pdf button.
6. Top right options will help in collaborations.

Now, what you should be getting is something that looks like:



```latex
\documentclass{article}
\usepackage[utf8]{inputenc}

\title{blankproject}
\author{Koh Yamakawa}
\date{August 2019}

\begin{document}

\maketitle

\section{Introduction}

\end{document}

```

Figure 1: Caption

Let's break this down:

- \documentclass{article}
  \usepackage[utf8]{inputenc}
  This will tell us exactly what formatting our document will be. Any alterations can be googled.
- \title{blankproject}
  \author{Koh Yamakawa}
  \date{August 2019}
  All used to construct the \maketitle command used later.
- \begin{document} The beginning of an environment. Latex is based around these and this will determine what is shown in your pdf.
- \maketitle This is one of the ways to construct a title. The default setting for what this command does can also be changed. For example, the title of this document is done in this manner.
- \section{Introduction} This is one of many ways to *section* off your document in chapters. These will be useful when we make our *table of contents.*
- \end{document} This is the end of our environment. Anything typed after this will not "directly affect" the document. This space is usually used for bibliographies.

3.3 Syntax

Newline

New lines in paragraph mode or "normal" mode can be created through \\\\or through \newline. However, there may be errors using these commands as the end of environments always create new lines and adding an additional \\\\after an environment may tell latex to add a new line to a new line, which it does not like.

Packages

Latex was not made perfectly when it came out. There are many versions of Latex[1] and many packages. These packages can be used by adding,

\usepackagepackagenamehere

In the case of online compilers, you usually do not need to do anything else. In the case of compilers on your personal computers, you may need to download and install from the web.

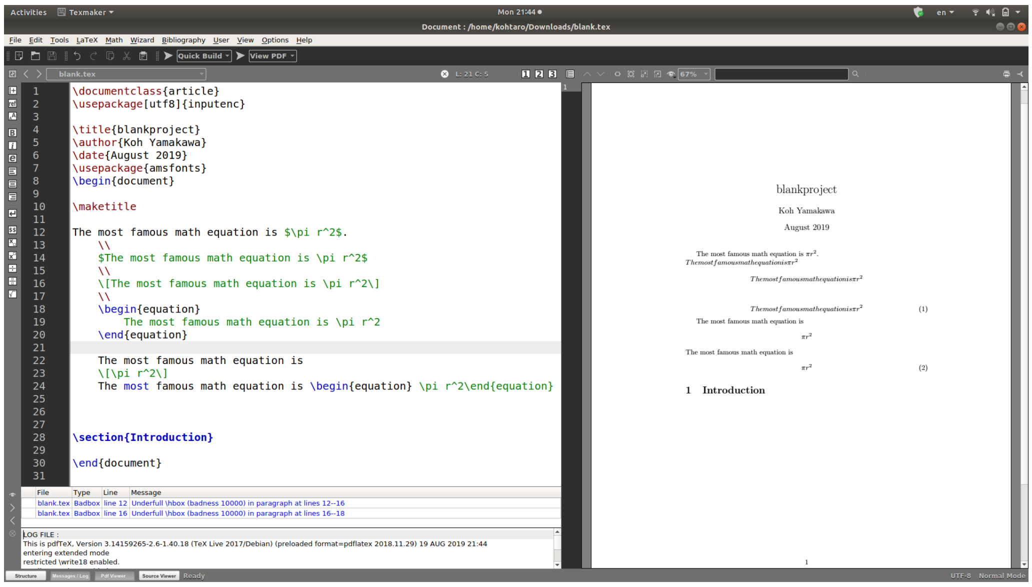
[1]Notably pdfLatex, LuaLatex, and BibTex, but we will always assume pdfLatex

Special characters

Some keys don't work as they are special characters. Some of them are: # $ % ˆ{ } & _ { } ∼ One must write them with a backslash \to properly write them out, save for the tilde, \sim, and the actual backslash in which case we use \textbackslash. There are others such as <> and more that end up looking like <> when compiled. Therefore, if you find question marks in your pdf when you wanted another symbol, odds are, its a special character.

Math Mode

Now, there are two operating modes. Paragraph or "normal mode" and math mode. The former respects spaces in between words and is like typing in a word document. The second was designed to type complex mathematical formulae. Now, while normally typing will result in paragraph mode, one requires the following environments. We will illustrate their usages via typing out the following:



```latex
\documentclass{article}
\usepackage[utf8]{inputenc}

\title{blankproject}
\author{Koh Yamakawa}
\date{August 2019}
\usepackage{amsfonts}
\begin{document}

\maketitle

The most famous math equation is $\pi r^2$.
    \\
    $The most famous math equation is \pi r^2$
    \\
    \[The most famous math equation is \pi r^2\]
    \\
    \begin{equation}
        The most famous math equation is \pi r^2
    \end{equation}

    The most famous math equation is
    \[\pi r^2\]
    The most famous math equation is \begin{equation} \pi r^2\end{equation}



\section{Introduction}

\end{document}
```

File	Type	Line	Message
blank.tex	Badbox	line 12	Underfull \hbox (badness 10000) in paragraph at lines 12--16
blank.tex	Badbox	line 16	Underfull \hbox (badness 10000) in paragraph at lines 16--18

```
LOG FILE :
This is pdfTeX, Version 3.14159265-2.6-1.40.18 (TeX Live 2017/Debian) (preloaded format=pdflatex 2018.11.29) 19 AUG 2019 21:44
entering extended mode
restricted \write18 enabled.
```

blankproject

Koh Yamakawa

August 2019

The most famous math equation is πr^2.
$The most famous math equation is \pi r^2$

$$The most famous math equation is \pi r^2$$

$$The most famous math equation is \pi r^2 \qquad (1)$$

The most famous math equation is

$$\pi r^2$$

The most famous math equation is

$$\pi r^2 \qquad (2)$$

1 Introduction

1

Figure 2: Caption

- \[· · · \ ]
  This is math mode on a new line, centered in the center of the page.
- $· · · $
  This is math mode in the same line.
- \being{equation}· · · \end{equation} This is math mode on a new line, BUT you can label and reference this type of equation.

We'll just note that the environments *align, align**[2] can supplant the *equation* environment. Therefore, we have summarized everything you need to know to get started in latex. What we have here, plus a internet search engine will be sufficient to providing you the freedom to make any homeworks that you desire.

3.4 Organizational Thinking

Now, we shall dive deeper into making our latex formatting elegant and organized well.

Environments

Environments are used to format blocks of text in Latex. They can be personalized with something in the preamble (before the \begin{document} command) with the command:

> \newenvironment{newenvironmentnamehere}
> {before block of text}{after block of text}

For example, one could construct an alternative proof element where we begin every proof with the lines "Here is my proof and ending with the line, "QED" with:

> \newenvironment{myproof}
> {\begin{proof} Here is my proof}{ Q.E.d.\end{proof}}

Commands

Commands are just shortcuts in mathmode. They are usually constructed in the preamble and can be created via,

> \newcommand {\newcommandnamehere} [number of arguments]
> {Shortcut with 1,2, . . . as the argument}

Then, we can call this command by $ newcommandnamehere{#1}{#2} etc where the brackets are not necessary when the next character is clear. For example, let's create a function shorcut,

> \newcommand {\expAdd} [3] {#1\exp#2+#3}

which we can call by, for example, \expAdd 143 or by \expAdd 143 to get,

$$1e^4 + 3$$

If the command already exists, you'll need to use "renewcommand" instead to redefine the command. Or change the title of your command.

[2]The asterisk tasks away the explicit labeling and referencing

Sections

As I said before, the sections are very useful in that we can partition out document quite easily to make chapters. There are sections, subsections, and subsubsections, and they can be called via the command

\{section}, \{subsection}, \{subsubsection}

The end result, is exactly what we have been using to partition our code. Note that adding an asterisk between the backslash creates a subsection that does not show up in the table of contents and does not come with an indexing.

Titles, Table of contents

The title is mostly a thing that people google to personalize. I suggest taking a look at this wikibooks site for more in formation. The important part is that in the document (and by that I mean in your begin and end document environment), you need to add

\maketitle

The table of contents looks like the one on this document. It also may be personalized but, most importantly we need to add the following in the document.

\tableofcontents

Itemize and Enumerate

One very important way to organize one's thoughts is to list things. Thats where the environments "itemize" and "enumerate" come in. The itemize environment, will look like

\begin{itemize} \item ··· \item ··· \end{itemize}

which looks like:

- ···
- ···

while the enumerate environment will look like:

\begin{enumerate} \item ··· \item ··· \end{enumerate}

which looks like:

1. ···

2. $\cdots$

The table of contents looks like the one on this document. It also may be personalized but, most importantly we need to add the following in the document.

```
\tableofcontents
```

Itemize and Enumerate

One very important way to organize one's thoughts is to list things. Thats where the environments “itemize” and “enumerate” come in. The itemize environment, will look like

```
\begin{itemize} \item ··· \item ··· \end{itemize}
```

which looks like:

- $\cdots$
- $\cdots$

while the enumerate environment will look like:

```
\begin{enumerate} \item ··· \item ··· \end{enumerate}
```

which looks like:

1. $\cdots$
2. $\cdots$

The way you enumerate can also be personalized and googled.

Hyperlink and References

In general, you need to label something to reference something. Let's label an equation:

```
\begin{equation}
\pi r^2
\label{circleEq}
\end{equation}
Equation $\ref {circleEq}$ is the most important equation.
```

which should come up as:

$$\pi r^2$$

Equation 3.4 is the most important equation.

The way things are labeled and the way things are referenced can, as always, be googled for more details.

Fonts,Themes, and Personalization

Google is our best friend. I will only note that many of these personalizations will require installations if compiling on a local computer. For the most part, one does not have to on Overleaf.

3.5 Becoming Efficient

Now that we have a grasp on how to create a file, let's learn a little on how to make this more efficient. In general, to be efficient, the method in which one Latexs makes a big difference. Doing work while you Latex has, in my opinion, is uneffective. Unless there is a lot of copy and pasting, the lines between work and formatting is blurred and we do not concentrate enough on one task. The point of this section is to accentuate on the principle that the process of making a proof and communicating a proof are distinct processes. That being said, we would like to focus on the essences of the proof and shorten the time it takes to format the proof. To do so, requires efficiency. In general, here are a few basic principles.

- Create commands, environments, hotkeys, and templates for formats that you use often.
- Commands can be made to shorten key strokes. For example, I use \pd for \partial.
- Read back over your written formatted work. Sometimes reading back over formatted work may make you think of better notation and make your communication more clear.
- Add comments to organize your work

In the next few sections, I will expand more on the value of some of the above principles.

Organized code

In line with the entire point of this section, it is good to organize your code so that it is easier to edit. This may be indenting your sections, environments, paragraphs etc. So, in general, the newline command ca nbe its own line, comment with the % symbol often, creating \todo[3] to organize your thoughts

Typing better

While typing is fast, it is not the fastest. Thats why VIM and Emacs is great. These create shorcuts to help you navigate without extensive, or any,

[3]using package todo

use of your mouse and once you get used to it, will make you very efficient. Overleaf has a setting where you can edit with vim hotkeys.

3.6 The Principle

Now we have learned how to make code that is good, clean, and efficient. We need to remember that this does not give you the excuse to write out the problem and then a general overview of the proof you've worked out, and say Q.E.D. Instead, this herculean effort to learn how to present our proofs allows us to organize the way that we think and therefore methodically present our ideas to our readers.
What this also means is that despite the very human tendency to make errors in our code, all of our effort would be supplanted by bad presentation. Whether that be typos, <> bad text, misplaced periods, or more, in the process of a rigorous and succinct presentation, in our context, a typo can make a rigorous proof ill-defined and therefore counteract all of our efforts.
Latexing our work allows us to exploit the Rubber Ducky principle-a debugging technique in which coders debug by explaining how their code works to a rubber duck. In the process of formulating a presentation to explain our logical thoughts, we are forced to understand our own thoughts and logic (proofs) and (perhaps) reveal their flaws. The act of latexing therefore while tiresome at first, will lead to a re-examination of our proofs and make us more efficient in the long run.
Latex is a typesetting code that is essential to anyone who wants to work in STEM. Papers, lecture notes, talks, homeworks, etc. use Latex and create an environment for efficient, precise notation and presentation of our logical thoughts. It will quite definitely help you organize your thoughts and give your TAs a break from terribly written homeworks.