# Introduction to LaTeX

Mitchell Jones
University of Sydney

April 23, 2015

# Outline

- The Basics
- Maths
- Referencing and Labels
- Algorithmic
- Figures
- Customising LaTeX
- BibTeX

# What is LaTeX?

- It is a typesetting system and markup language
- No GUI interface
- You concentrate on the content, it does the rest

# Example Document

A Short Introduction to the Depth First Search Algorithm

Author

University of Sydney

## 1 Introduction

Imagine you're a robot navigating through a maze. You're given a start point and would like to explore the entire place. What's the easiest way to do this? In computer science, this is known as a search algorithm. There are a lot of search algorithms known (for example, A*, BFS and DFS), but today we'll focus on one of the most simple algorithms: Depth first search (DFS) [1].

## 2 The Algorithm

Formally, we're given a graph $G = (V, E)$ where $V$ are the set of *vertices* in a graph and $E$ is the set of *edges*. Each edge is connected to two vertices. Given a starting vertex $v \in V$, we may like to determine if there is another vertex in the graph. To find this vertex, we need to *search* $G$ using a depth first search. The algorithm is described below.

**Algorithm 1** Depth first search on a graph.

```
procedure DEPTHFIRSTSEARCH(V, E, x)          ▷ x is our start vertex.
    Create an empty stack S and push x onto S
    Mark each vertex v ∈ V and not visited
    while S is not empty do
        u ← S.pop()
        if u has not been visited then
            Mark u as visited
            for every unseen neighbour w of u do
                S.push(w)
            end for
        end if
    end while
end procedure
```

## 3 Analysis

**Lemma 1.** *The running time of Algorithm 1 is $O(n + m)$, where $|V| = n$ and $|E| = m$.*

*Proof.* Notice that we first need to mark each vertex as visited. Since there are $n$ vertices, this will take $O(n)$ time.

For the `while` loop, notice that each vertex is pushed onto the stack **at most** once. Therefore there are $n$ iterations. Now, at each iteration for a particular vertex $u \in V$, we go through each neighbour $w$ of $u$ and push it onto $S$ if we haven't yet seen $w$. Let $\deg(v)$ denote the number of neighbours of a vertex $v \in V$. Then at each iteration, we perform $\deg(v)$ computations. In total, the total time is

$$\begin{aligned}\deg(u_1) + \deg(u_2) + \cdots + \deg u_n &= \sum_{u \in V} \deg(u) \\ &= 2|m| = O(m)\end{aligned}$$

where the second equality follows by the Handshaking lemma. This means the total running time of DFS is $O(n + m)$[1], completing the proof. □

## 4 Example

Here we provide a brief example on how the algorithm works. See Figure 1.

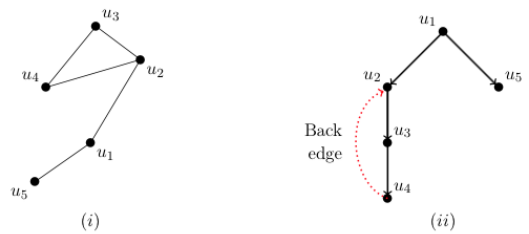



Figure 1: (*i*) A sample graph with five vertices. (*ii*) The tree generated by the DFS algorithm. Notice there is a back edge from $u_4$ to $u_2$ since $(u_2, u_4) \in E$, however we have already visited $u_2$ previously.

Let $V = \{u_1, u_2, u_3, u_4, u_5\}$ and $E = \{(u_1, u_5), (u_1, u_2), (u_2, u_3), (u_3, u_4), (u_4, u_2)\}$. We call the procedure `depthFirstSearch`$(V, E, u_1)$. Since the search starts at $u_1$, we mark it as visited and push $u_5$ and $u_2$ onto the stack $S$. Say we visit $u_5$ next, but nothing is pushed onto $S$ since $u_5$ has no other unvisited neighbours. Next we visit $u_2$, mark it as visited and push $u_3$ and $u_4$ onto $S$. Now we visit $u_3$ and mark it as visited, but $u_3$ has no unvisited neighbours, so we next pop $u_4$. The vertex $u_4$ has two neighbours, $u_2$ and $u_3$, which have already been visited, so we do nothing. $S$ is now empty, and we have explored the entire graph.

## References

[1] A. Reference, "An example reference," *Reference Journal*, vol. 314, no. 7, pp. 1477–1484, 2013.

[1] We can also show the DFS algorithm uses only $O(n)$ additional space.

# Downloading LaTeX

- See latex-project.org/ftp.html.
- Mac: Download MacTeX, includes LaTeX itself and an editor
- Windows: MiKTeX and TexWorks
- Linux: Your package manager
- On Unix machines, run `pdflatex my-document.tex`

# Creating a Document

```
\documentclass{article}
\title{My First Document}
\author{Author name}
\date{\today}
\begin{document}
  \maketitle
  Hello, world!
\end{document}
```

# Formatting

- Creating new sections

```
\section{Introduction}
...
\subsection{Contribution}
...
\subsubsection{Discussion}
```

- Paragraphs are separated by newlines. To enforce a new line, end it with a double slash: \\.
- Create a bullet list using `itemize`

```
\begin{itemize}
  \item Item 1
  \item Item 2
  ...
\end{itemize}
```

# Inline and Block

- Inline maths:

```
Euler's identity states that $e^{i\pi} + 1 = 0$.
```

- Produces: "Euler's identity states that $e^{i\pi} + 1 = 0$."
- Block equations

```
Euler's identity states that
$$e^{i\pi} + 1 = 0.$$
```

- Produces: "Euler's identity states that

$$e^{i\pi} + 1 = 0.''$$

- Detexify: detexify.kirelabs.org/classify.html.

# Alignment

```
\begin{equation}
  \label{eq:sum-to-n}
  \begin{aligned}
    1 + 2 + 3 + \ldots + n &= \sum_{i = 1}^n i \\
                           &= \frac{n(n + 1)}{2}
  \end{aligned}
\end{equation}
```

Produces..

$$\begin{aligned} 1 + 2 + 3 + \ldots + n &= \sum_{i=1}^{n} i \\ &= \frac{n(n+1)}{2} \end{aligned} \tag{1}$$

# Labels and Referencing

- Use labelling to refer back to previous equation, figure or sections using `label`

```
\section{Introduction}
\label{sec:intro}
...
```

- Convention: `\label{<label type>:<name of label>}`
- Then reference it anywhere by using `ref`
- In our example `\ref{eq:sum-to-n}`

# An Example

**Algorithm 1** Counting up to $n$.

```
function COUNT(n)
    total ← 0
    for i = 1 to n do
        total ← total + i
    end for
    return total
end function
```

# The Code

```
\begin{algorithm}
  \caption{Counting up to $n$.}
  \begin{algorithmic}
    \Function{count}{$n$}
      \State total $\gets$ 0
      \For{$i = 1$ to $n$}
        \State total $\gets$ total + $i$
      \EndFor
      \State\Return total
    \EndFunction
  \end{algorithmic}
\end{algorithm}
```

# Syntax

- Similar to a programming language
- `\State` and `\Comment`
- `\If{condition} ... [\Else] ... \EndIf`
- `\For{condition} ... \EndFor`
- `\While{condition} ... \EndWhile`
- `\Function{<name>}{<params>} ... \EndFunction`

# Graphics

- Graphics are also simple

```
\begin{figure}[H]
  \begin{center} % Center the graphic.
    \includegraphics[scale=<value>]{<my file>}
  \end{center}
  \caption{A caption.}
  \label{fig:my-label}
\end{figure}
```

# Positioning

- The `[H]` tells LaTeX to place the figure *right where you entered it*
- To float a figure around text, use `wrapfigure`
- E.g. Float it to the left, with padding $\frac{1}{4}\times$textwidth (can also use pt)

```
\begin{wrapfigure}{l}{0.25\textwidth}
  \vspace{-10pt}
  \begin{center}
    % Figure..
  \end{center}
  \vspace{-10pt}
  % Caption and label..
\end{wrapfigure}
```

# TikZ

- See cremeronline.com/LaTeX/minimaltikz.pdf for a short introduction
- All TikZ code is inside a figure environment

```
\begin{figure}[H]
  \begin{center}
    \begin{tikzpicture}
      % TikZ code..
    \end{tikzpicture}
  \end{center}
\end{figure}
```

# Lines

- Draw a line from point (0,0) to (1,2)

```
\begin{tikzpicture}
  \draw [<parameters>] (0,0) -- (1,2);
\end{tikzpicture}
```

- Some parameters include:
  - Directed line. `[->]`
  - Change line width. `[line width=<size>]`
  - Make it dashed or dotted. `[dashed]` or `[dotted]`
  - Change the colour to red. `[red]`
- E.g. Draw a directed, dashed, red line of width 2pt from (1,1) to (2,2):

```
\draw [red,dashed,line width=2pt,->] (1,1) -- (2,2);
```

# Drawing Nodes

- Can also add `nodes`, which can display text

```
\draw [fill] (1,1) circle [radius=0.1];
\node [below] at (1,1) {Area is $\pi r^2$};
```

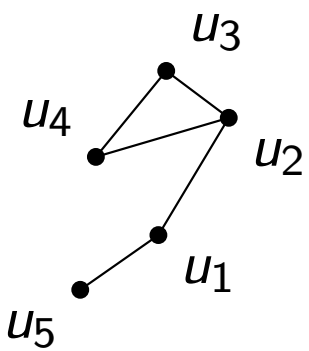

- Can also use different positions: `above`, `below`, `left` and `right` (or a combination)

# Shapes

```
\draw [fill=red,ultra thick] (0,0) rectangle (1,1);
\draw [fill=blue,ultra thick,blue] (2,0) rectangle (3,1);
\draw [green, fill=green] (4,0) -- (5,1) --
                          (4.75,0.15) -- (4,0);
\draw [fill] (7,0.5) circle [radius=0.1];
```

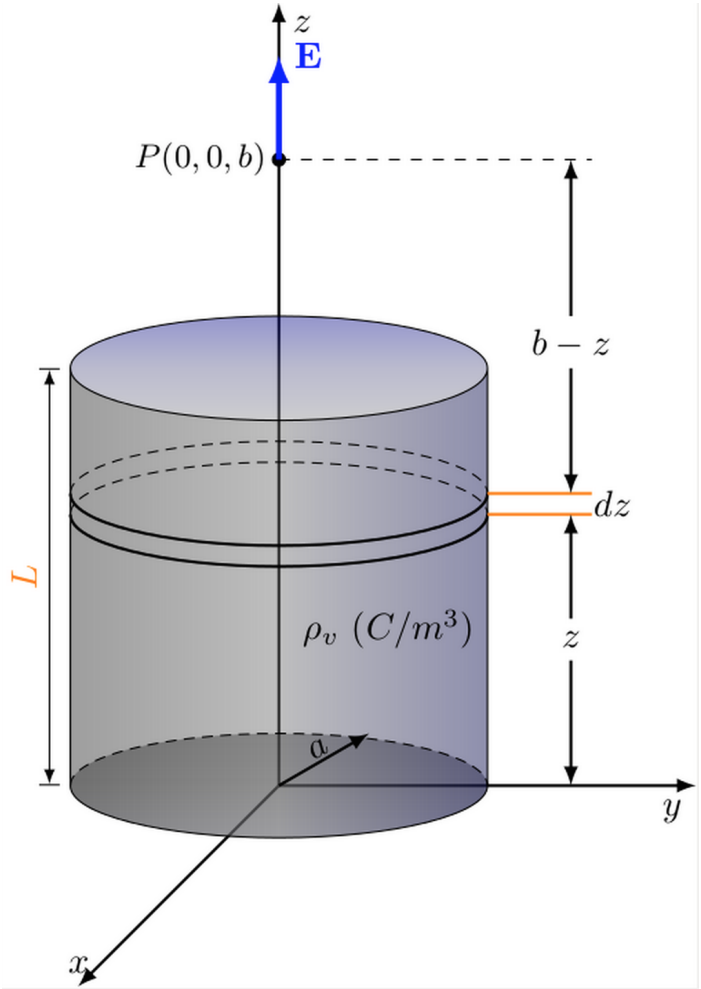
# Example

```
\foreach \Point in {(2,2), (1,1.3), (2.9,3.5),
                    (1.2,3), (2.1,4.1)} {
  \draw [fill] \Point circle [radius=0.1];
}
\draw [-] (1,1.3) -- (2,2) -- (2.9,3.5) --
          (2.1, 4.1) -- (1.2, 3) -- (2.9, 3.5);
\node at (2,2) [below right] {$u_1$};
\node at (2.9,3.5) [below right] {$u_2$};
\node at (2.1,4.1) [above right] {$u_3$};
\node at (1.2,3) [above left] {$u_4$};
\node at (1,1.3) [below left] {$u_5$};
```

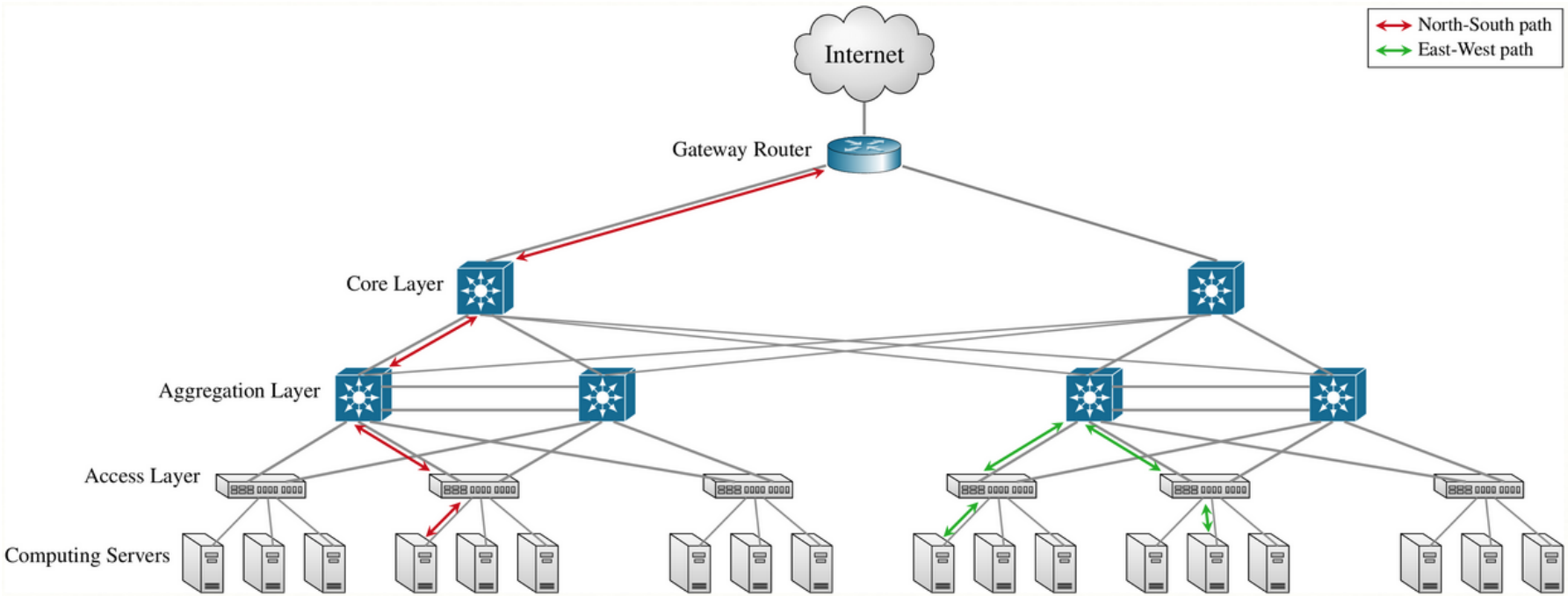


# The Sky is the Limit..

About $\approx$ 60 lines of code. (Source)



About $\approx$ 500 lines of code! (Source)



# Minted

- Minted is a LaTeX package that displays code, see github.com/gpoore/minted

```
\begin{minted}[mathescape,linenos,numbersep=5pt,
                frame=lines,framesep=2mm]{python}
# Returns the sum $1+2+\dots +n = \frac{n(n+1)}{2}$.
def sum_to_n(n):
  return (n*(n+1))/2
\end{ minted}
```

- Produces..

```
1 # Returns the sum 1 + 2 + ... + n = n(n+1)/2.
2 def sum_to_n(n):
3   return (n*(n+1))/2
```

# Maths Packages

- There are *a lot* of maths packages for fancy symbols
- Just to name a few `amsmath`, `amssymb`, `upgreek`, `eucal`, ...
- E.g. `mathcal` – For example `Let $\mathcal{O}$ be the set of numbers...` produces "Let $\mathcal{O}$ be the set of numbers..."
- E.g. `mathbb` – `$\mathbb{R}$ is the set of real numbers` produces "$\mathbb{R}$ is the set of real numbers"

# Hyperlinks

- Another common feature is adding hyperlinks
- Use the `hyperref` package
- When importing the package, can set various parameters for link colours such as

```
\usepackage[colorlinks=true,citecolor=RedViolet,
            linkcolor=RoyalBlue,
            urlcolor=ForestGreen]{hyperref}
```

- Then use `\href{<URL>}{<link name>}`

# Lemma & Theorems

- Often you will want numbered lemmas and theorems, use the `amsthm` package

```
\theoremstyle{plain}
\newtheorem{thm}{Theorem}
\newtheorem{lem}{Lemma}[chapter]
```

- This means we create a macro called `lem` that restarts numbering at the beginning of each chapter

```
\begin{thm}
The sum $1 + 2 + \dots + n = \frac{n(n+1)}{2}$.
\end{thm}
```

**Theorem**

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

# Macros

- Maybe you use `$\mathbf{A}$` a lot to represent a vector
- Format:
  `\newcommand{\<name>}[<num-args>]{<definition>}`
- For example
  `\newcommand{\mat}[1]{\ensuremath\mathbf{#1}}`
- Then using `$\mat{A}$` produces $\mathbf{A}$

# Adding a Reference

- BibTeX allows you to add citations to your report
- See en.wikibooks.org/wiki/LaTeX/Bibliography_Management
- Store these references in a *separate* `.bib` file

```
@article{example-ref,
  title = {An example reference},
  author = {Reference, A.},
  journal = {Reference Journal},
  volume = {314},
  number = {7},
  pages = {1477--1484},
  year = {2013}
}
```

# Citing the References

- Set the bibliography style: `\bibliographystyle{ieeetr}`
- Tell LaTeX to display our references just before the end of our document: `\bibliography{references}`
- Can then cite a reference in your document using `\cite{example-ref}`

# Compiling with BibTeX

- After adding the `.bib` file to your document, run the following:

```
$ pdflatex my-doc.tex # Compile the document first
$ bibtex my-doc # Generate the references
$ pdflatex my-doc.tex # Link references to document
```

# What we've covered

- How to make a basic document
- Referencing, labels, citations
- Drawing your own diagrams, including graphics
- Adding your own lemmas, theorems and definitions
- Defining your own macros
- Including code and algorithms

# Resources

- Example document and template: cl.ly/ahj0
- Detexify: detexify.kirelabs.org/classify.html
- Templates: latextemplates.com

Where to go from here..

- pgfplots
- beamer

# Thanks for listening!