My Capstone Journey - A Paper on *Tree Data Structures and Algorithms*

•••

By Bill Lin

What is my Capstone about?

- Decided to write a paper on a specific topic in computer science: trees.
  - Specifically, algorithms and data structures related to trees
- Not too “formal,” meant to help the reader gain a better understanding of the topics described

An Analysis of Tree Algorithms and Data Structures

Bill Lin

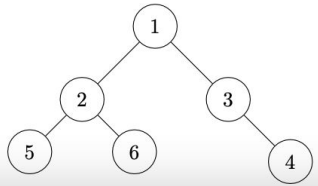
1 Introduction

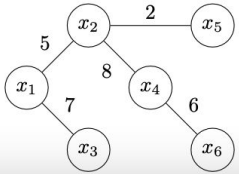
The purpose of this paper is to analyze a certain collection of tree algorithms and tree data structures, and to give an intuition as to why they work.

We will first define what a tree is and look at its properties. A tree is defined as a **connected acyclic undirected graph**. The **connected** part means that every node can be reached from every other node. **Acyclic** means that the tree does not have a sequence of edges that joins a sequence of vertices that starts and ends on the same vertex (basically, no cycles). **Undirected** means that the edges in the tree go in both directions. As a result of these properties, trees will always have $n - 1$ edges, where n is the number of nodes.

Why would trees always have $n - 1$ edges? Let’s start with the simplest case: a single node. With only a single node, there will be no edges, as the only edge we could add is from the node to itself, which would create a self-loop and would create a cycle. If we were to add a new node, we would need exactly one edge from the previous connected tree to the new node in order to make the graph connected. So, that means the tree starts with 1 node and no edges, and every new node added would require one extra edge. Thus, the number of nodes will always be one more than the number of edges in the tree.

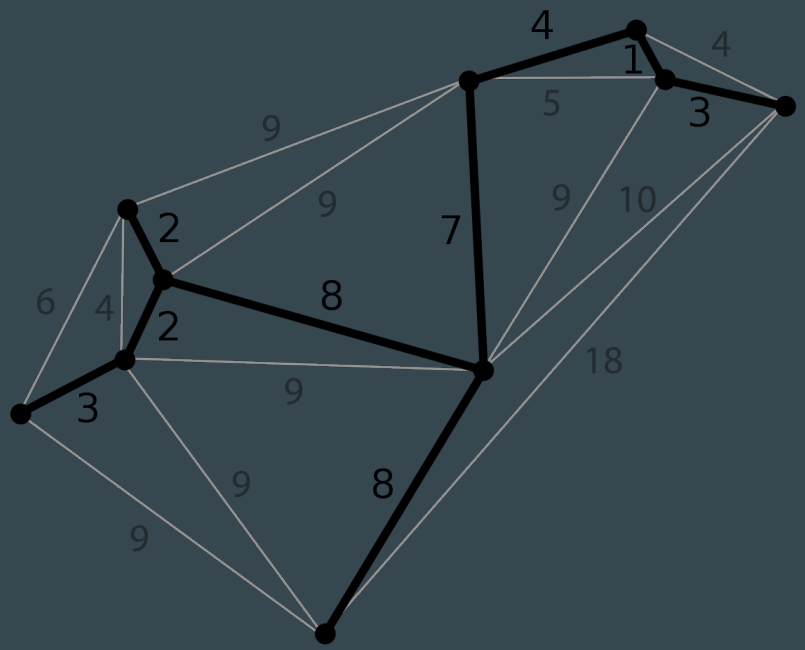
Here are a couple of examples of trees:





Why? - Motivation behind my Capstone

- Enjoy programming contests, want to study Computer Science in university
- Algorithms and data structures are required to solve problems in these contests
- However, I felt like I had a lack of understanding about these algorithms
    - Understood how to implement these algorithms but not really how it works
- Example: Kruskal's gives MST of a graph, but why is it optimal?

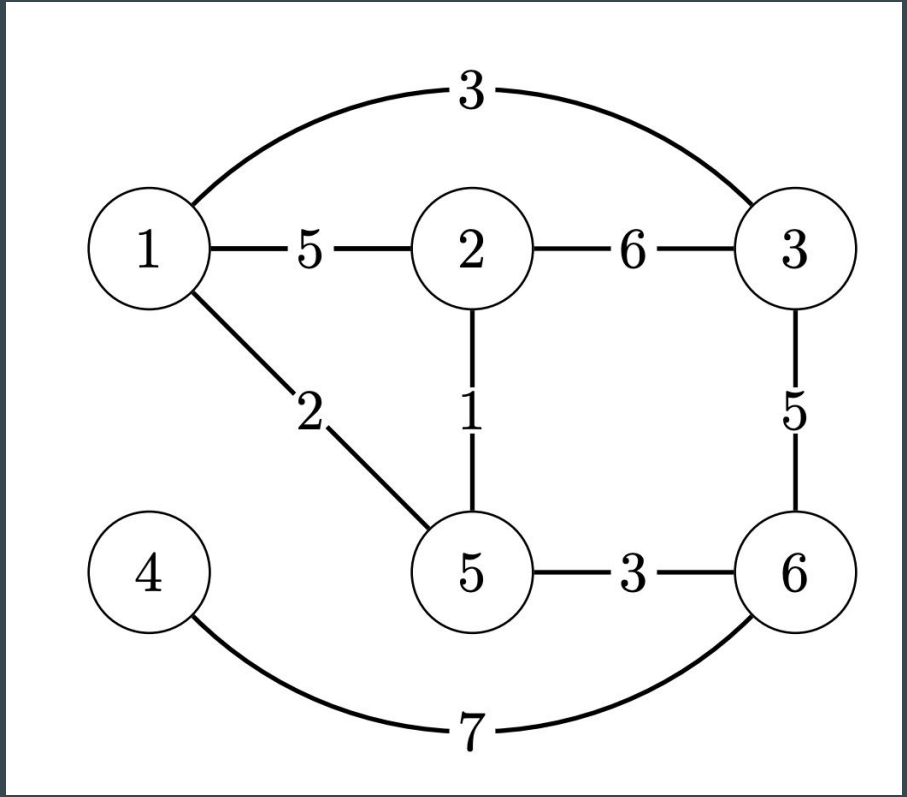


Why? - Motivation behind my Capstone

- Wasn't satisfied with just knowing what they did - I wanted to gain a deeper understanding of the algorithms - focus on the intuition behind them
- I was also always really impressed by papers written in LaTeX and I wanted to be able to do the same

Why? - Motivation behind my Capstone

- Learning to explain a topic -> deeper understanding, more than just reading about it
- So I wanted to write a paper focused on describing the intuition behind the algorithms.
    - I didn't want to focus too much on the formalities - it was just an opportunity for me to develop a better intuition
- Also just wanted to draw cool graphs



Who? Others involved

- Mr. Ubial was my mentor
  - First suggested that I use LaTeX to write my paper and suggested using the program Overleaf
  - Discussed my plan with him in early December and he gave me some advice on the topics that I chose
  - Helped me a bit with formatting
  - Read over my paper and gave suggested changes to some explanations to increase clarity

When? - Timeline

- Started in around Oct. 2021
- I'd separate the project into 3 phases:
  - 1. Planning (Oct to Nov 2021)
  - 2. Research/Solving Problems (Nov to Jan 2022)
  - 3. Writing the paper (Jan to April 2022)

The Learning Process - Planning

- Started by planning out which topics I wanted to learn about
- Got a lot of resources from online
  - cp-algorithms.com
  - codeforces.com
  - Tutorials on YouTube
- I had to make a lot of changes to this list
  - Used to be just general graph theory - changed it to just trees because there wasn't enough time
- Had to pick what I thought would be the most interesting topics

List & Order of Topics

1. BFS, DFS
2. Single source shortest path
   - Dijkstra
     - 0-1 BFS should be mentioned here
   - Bellman-Ford
3. Floyd-Warshall (All Pairs Shortest Path)

Trees

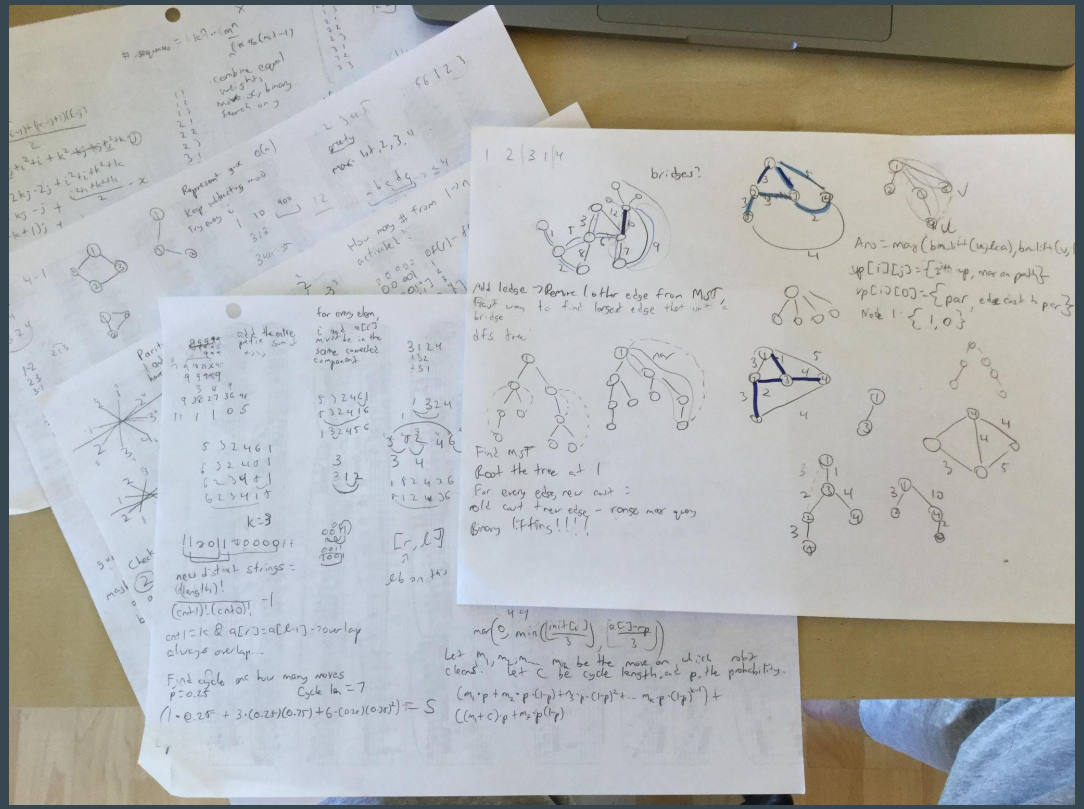
4. Spanning Trees
   - Prim's, Kruskal
5. Eulerian Paths, Tours & Range Queries on Subtrees
6. LCA
   - Binary Lifting, RMQ

(if I have time):

7. DFS Trees, finding bridges
8. Heavy-Light Decomposition optimization

The Learning Process - Research, doing practice problems

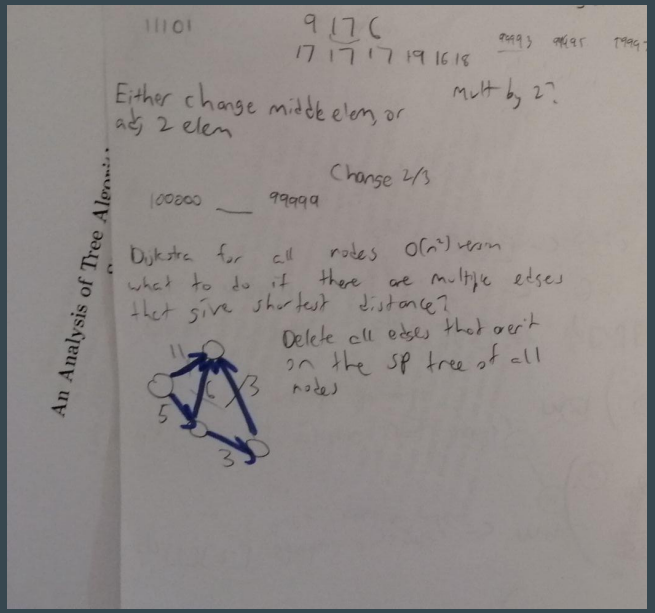
- Second part of my Capstone - getting a deep enough understanding in order to be able to explain it well
- Started doing problems on these topics on sites like codeforces
  - Learned a lot while solving these problems

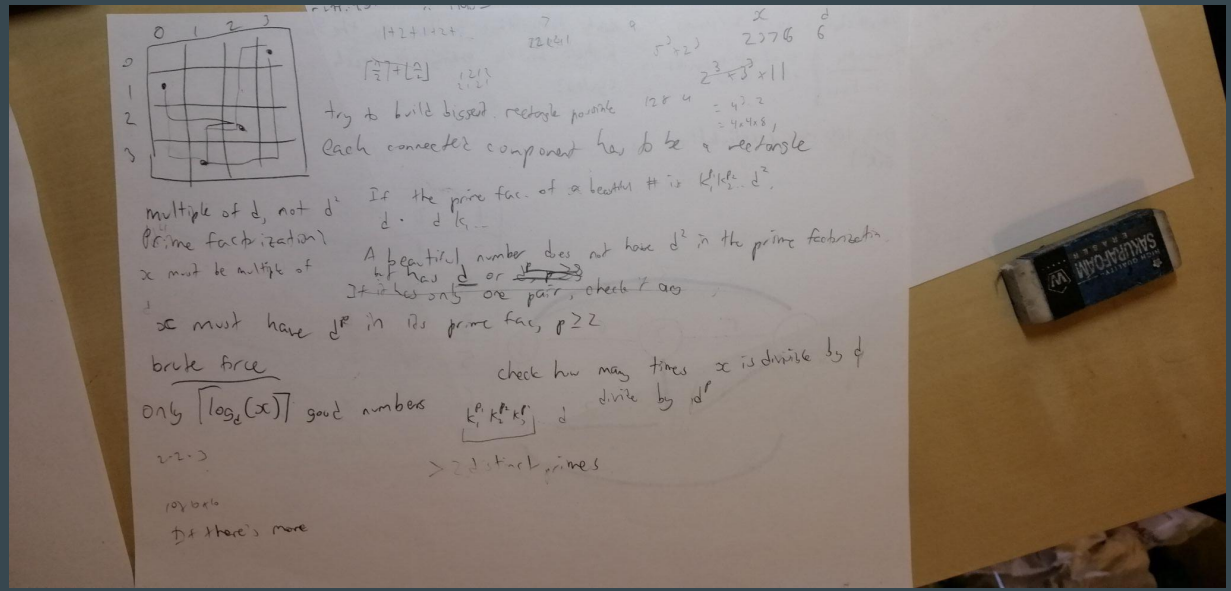


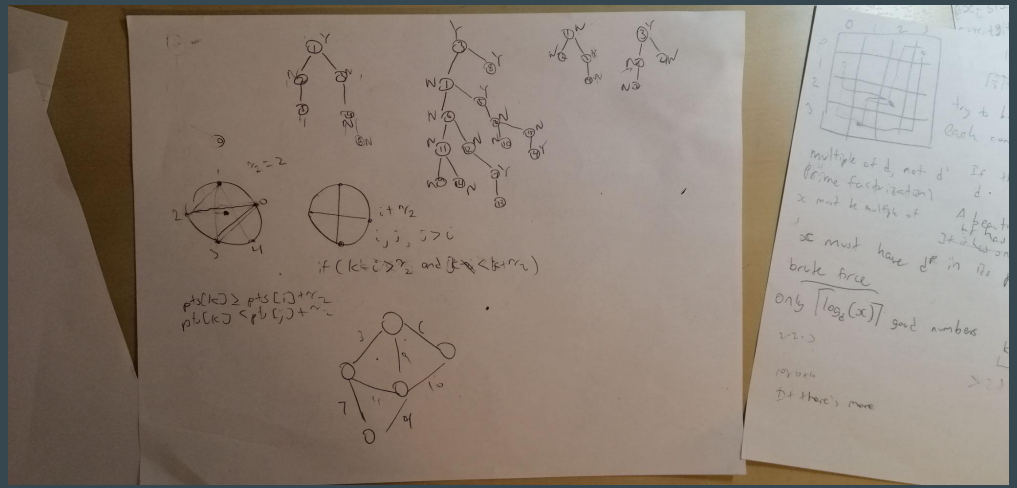
The Learning Process - Research, doing practice problems

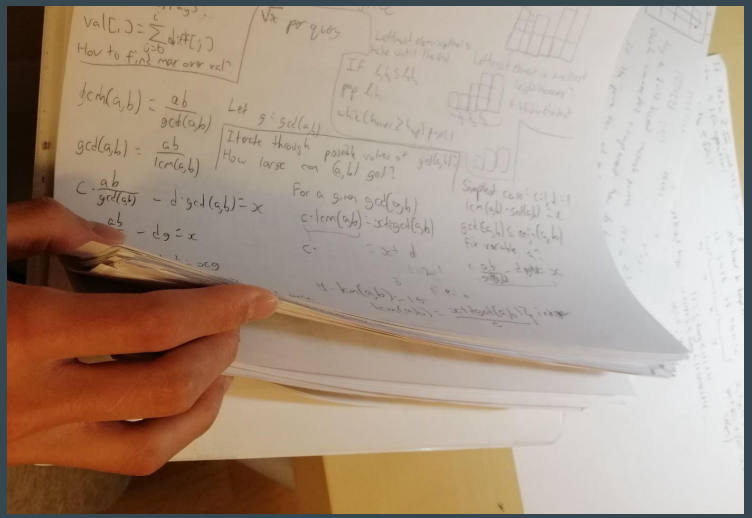
- Did over 20+ practice problems on these topics, mainly on codeforces
- Only focused on problems that were just above my skill level in order to learn from them - read solutions if I couldn't get the problem, but not before trying for at least a couple hours
- Did a ton of brainstorming for these problems - improved problem solving skills
  - have a binder full of scrap paper used for these problems
- I felt like these practice problems was the main part of my learning

The Learning Process - Research, doing practice problems






brute force
only $\lceil \log_2(x) \rceil$ good numbers
multiple of d, not d^2
x must be multiple of
x must have d^{th} in its
if there's more


How to find max over val?
$lcm(a,b) = \frac{ab}{gcd(a,b)}$
$gcd(a,b) = \frac{ab}{lcm(a,b)}$
Iterate through possible values of gcd(a,b)
How large can (a,b) go?
For a given gcd(a,b)

Aside on doing practice problems

- These practice problems also helped me in the programming contests that I did during this period
- In Feb. 2022, there was the Canadian Computing Competition
- I was practicing problems on the topic of dynamic programming on trees for the 3rd topic in my capstone
- The hardest problem was a problem on tree DP
- Ended up solving it and got top 25 out of over ~3200 contestants
- Shows proficiency

The Learning Process - Writing

- 3rd part - writing the actual paper
- Had a lot of trouble learning LaTeX because it has a pretty steep learning curve - used a lot of tutorials, Googling, and mainly trial and error to learn
- A huge obstacle was formatting and drawing - it took forever to make my diagrams look right
    - Alignment issues, overlapping edges/vertices, uneven gaps between nodes
- Another obstacle was that the online editor that I used - Overleaf, couldn't compile because there were too many diagrams, had to switch to TeXShop, an offline editor

The Learning Process - Writing

- While writing, I found that I had quite a lot of knowledge gaps and this was a big obstacle
- Example: I couldn't explain why Prim's works with negative edge weights, even though a very similar algorithm (Dijkstra), doesn't
- I went back and did more reading on sites like cp-algorithms.com and did more practice problems
- made an "intuition" section on my paper to explain further
- This was also a big reason why I wanted to write this paper - I read that the best way to truly understand something is to try to explain it
- Really showed my knowledge gaps and I'm glad I got rid of them

The Learning Process - Writing

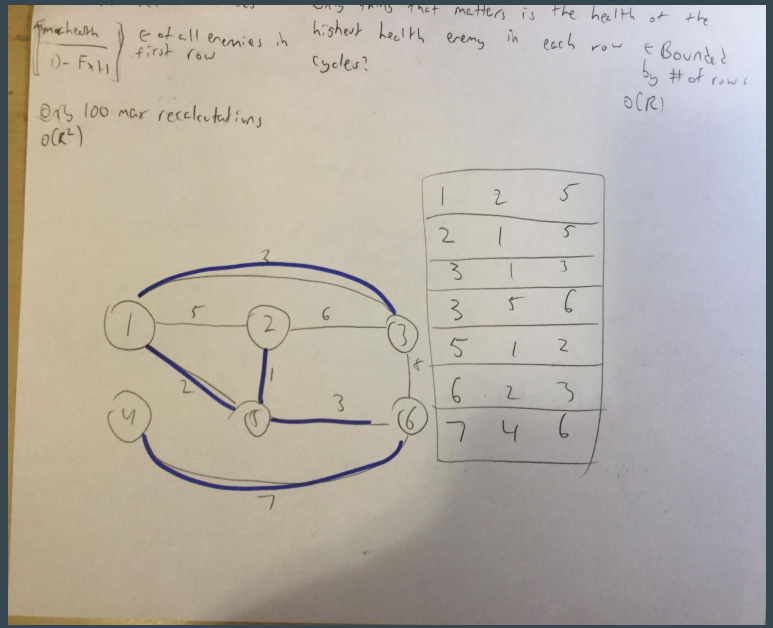
Diagrams - Drawing graphs

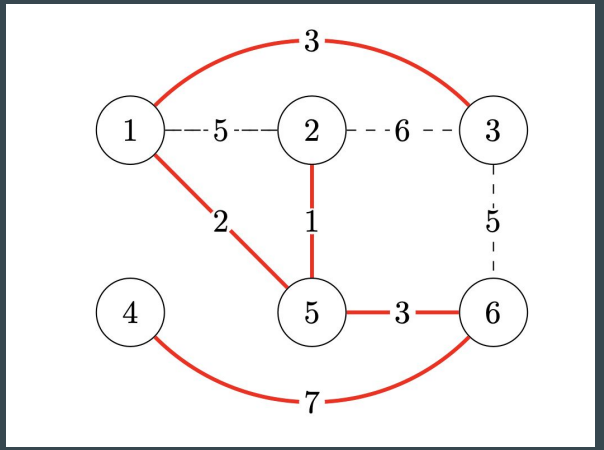
```
Consider the following tree, rooted at 1:
\begin{center}
  \begin{tikzpicture}[node distance={1.5cm}, main/.style = {draw, circle}, minimum size=0.75cm, every edge/.style = {draw = black}]
  \node[main] (1) at (0, 0) {1};
  \node[main] (2) at (-1, -1.5) {2};
  \node[main] (3) at (1, -1.5) {3};
  \node[main] (4) at (0, -3) {4};
  \node[main] (5) at (1, -3) {5};
  \node[main] (6) at (2, -3) {6};
  \node[main] (7) at (-0.5, -4.5) {7};
  \node[main] (8) at (0.5, -4.5) {8};

  \begin{scope}[>={Stealth[black]},
      every node/.style={fill=white, circle, inner sep = 0pt, minimum size = 0.1cm},
      every edge/.style={draw=black,thick}]
  \path
  (1) edge (2)
  (1) edge (3)
  (3) edge (4)
  (3) edge (5)
  (3) edge (6)
  (3) edge (6)
  (4) edge (7)
  (4) edge(8);
  \end{scope}
  \end{tikzpicture}
\end{center}
```

The Learning Process - Writing

Drawing diagrams is a very time consuming and tedious process





weight	first node	second node
1	2	5
2	1	5
3	1	3
3	5	6
5	1	2
6	2	3
7	4	6

What I learned

Skills that I've developed:

- Problem solving skills - brainstorming ideas to solve problems, using other resources, etc.
- Writing and drawing diagrams (Communication)
- Gained a much better intuition behind the algorithms
- Perseverance
    - many times a problem feels like it's too hard or annoying to deal with - had to force myself to continue to try to solve it
- Ability to adapt to time constraints

What I learned & Results

- Time management - there were a lot of really time consuming tasks that needed to be done

- The paper is 10 pages long, with 4200+ words
- 600+ lines of code in TeXShop
- I spent ~25 hours on writing and about ~50+ hours just doing practice problems alone
- Mostly satisfied with what I've done but I wished I had more time in order to do more research on the topics that I had to cut out

Reflection, Changes, and what I would've done differently

- I would've definitely started sooner - had to cut out a bunch of topics I was interested in simply because there wasn't enough time and because I severely underestimated the amount of time it would've taken
- I changed from a broad range of topics (general graph theory) to a specific one in the middle of writing my paper
    - I think I should've actually done a bit more research and dabbled in on general graph theory before honing in on one topic
- Asking for help - even though Mr. Ubial did help me a lot, I think I should've still asked for more guidance as I think I would've improved the paper a lot more with more guidance

Thank you for listening