Representing Our World:
An Introduction to Graph Theory

The Study of Related Objects

Ethan Mathieu

April 6, 2024

Table of Contents

- Introductions
- What does CS even have to do with math?
- What even is a graph?
- Lab : Network Engineering
- Interactive : Understanding the Algorithm behind Google Maps
- Conclusions + More Stuff

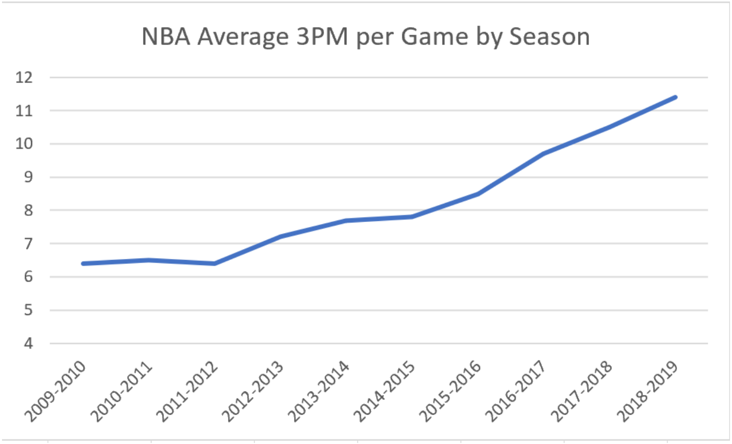
Introductions

- My name is Ethan, I'm studying Computer & Data Science.
- Aspiring software engineer, previously at Procter & Gamble + (soon) Netflix.
- Most of the engineering problems I've worked on/will work on are fundamentally just graph problems, so I'm excited to talk about this subject with you guys!
- **Icebreaker:** Anyways, let's get to know each other! Say where you're coming from (School + State) as well as your favorite food!

What does all this math have to do with CS?

- Computer Science has math involved. It's not all about programming!
- That's not to say programming/software creation is not important, its just not all of CS.
- Today, we focus on one of the tools in our mathematical kit that we use to bring solutions to life: Graph Theory!

This is a graph - but not the type of graph we care about!



Beginning to Motivate Mathematical Graphs

Imagine I want to express a **relationship** between a set of things.

Sports : The **Lakers** are set to play to **Celtics**.
Social Media : **I (Ethan)** "follow" my friend **Carter** on Linkedin. My friend Carter follows both me and my friend **Tony**.
Maps : How do I get from **Arby's** (ew!) to **Friendly's** (yum!) in a car if they are down the street from each other?

Graphs in Math

Definition

A graph G is a collection of **vertices** (a.k.a nodes) and **edges** connecting pairs of vertices. We define this concisely as $G = (V, E)$.

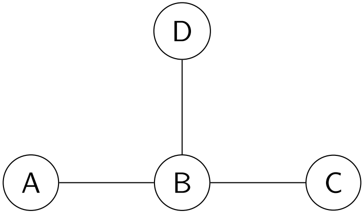
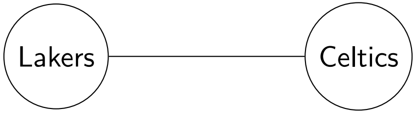


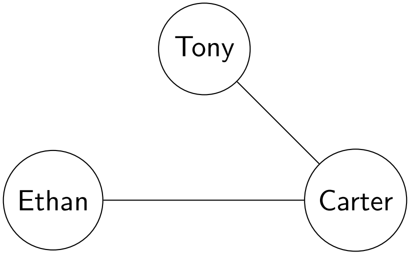
Figure: Example of a simple graph with four vertices (A, B, C, D) and three edges. $V = \{A, B, C, D\}$, $E = \{AB, DB, BC\}$

Going Back to Our Motivation - The Graph Representation

Sports : The **Lakers** are set to play to **Celtics**.



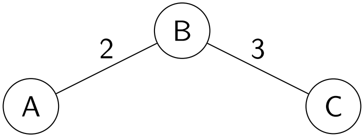
Social Media : **I (Ethan)** "follow" my friend **Carter** on Linkedin. My friend Carter follows both me and my friend **Tony**. I don't follow Tony.

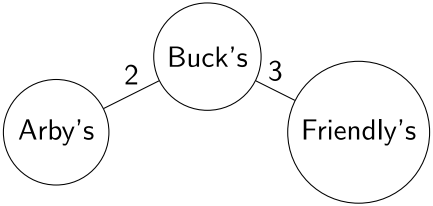


Graph Terminology

Definition: Weighted Graph

A graph $G = (V, E)$ where each edge has a weight. A weight is the cost to travel along that edge. All other graphs implicitly have a weight of 1 on all edges.





Graph Terminology

Definition

A **path** is a unique walk from one vertex to another.

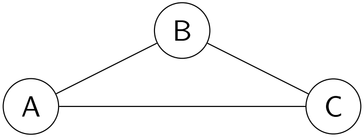


Figure: There are two paths from from A to C

Graph Terminology

Definition: Graph Cycles

A **cycle** is a closed path or circuit that starts and ends at the same vertex. A graph without cycles is called **acyclic**.

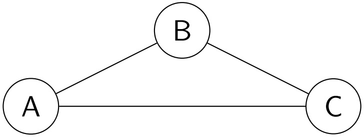


Figure: A cyclic graph

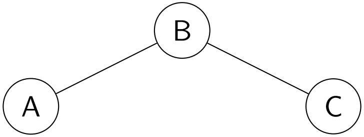


Figure: An acyclic graph

Graph Terminology

Definition: Disconnected & Connected Graphs

A **disconnected** graph is a graph where at least one pair of vertices have no path connecting them.

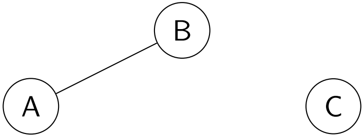


Figure: A disconnected graph

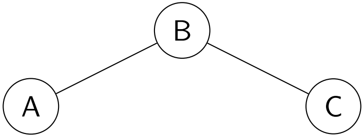
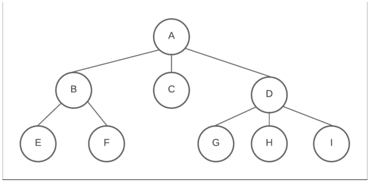


Figure: An connected graph

Graph Terminology

Definition

A **tree** is a special type of graph that is connected and acyclic.



- **Root:** A designated starting vertex.
- **Connected & Acyclic:** Every pair of nodes is connected by exactly one path, and there are no cycles.

Okay, but how do we actually use these graphs?

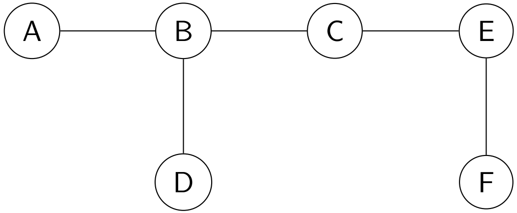
- We define **algorithms!** Algorithms are series of instructions that are executed (in our case by computers) to achieve a task.
- We can define algorithms to **traverse** the graphs that we create.
- In practice, graphs are stored in **databases**. There are many common representations used to encapsulate what the graph looks like, which we will abstract away for the sake of time.
- The results of exploring the graph, as well as some of the information used to execute the algorithm, are kept in (abstract) **storage devices**.

How do I actually use these graphs : Depth First Search (DFS) Algorithm

1. Choose a starting node and mark it as visited.
2. Explore as far as possible along each branch before backtracking in order.
3. Repeat until all nodes are visited.

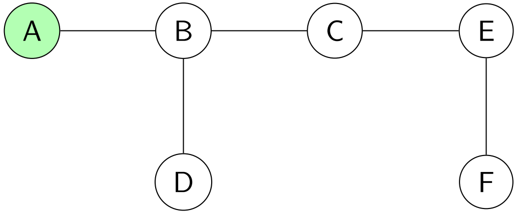
DFS will give you an idea of how computers **algorithmically** explore graphs. It is also the building block to a lot of graph algorithms.

Example Graph

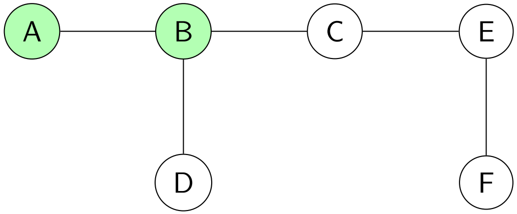


Question: Use some words to describe this graph!

Step 1: Choose Starting Node



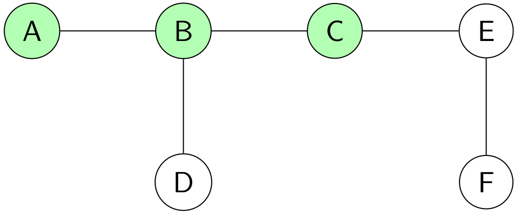
Step 2: Explore as Far as Possible (B)



We just grew the **cut**! The cut are the edges you can cross to reach vertices you haven't already reached. They have one vertex that has already been visited and one vertex that has not.

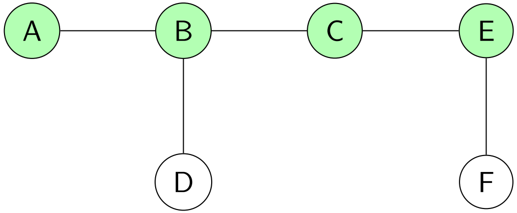
So in this instance, traversing *BC* and *BD* would grow our cut. Traversing *AB* would not.

Step 3: Explore as Far as Possible (C)

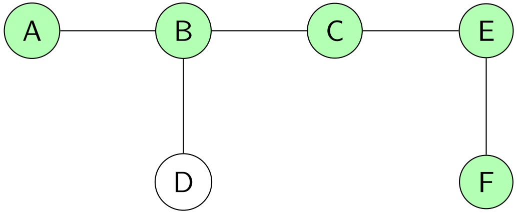


Continue exploring along the right branch!

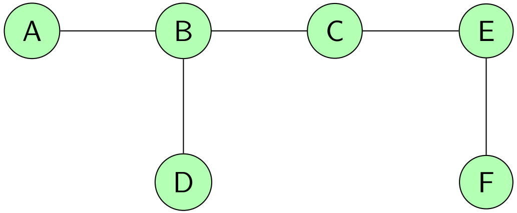
Step 4: Explore as Far as Possible (E)



Step 5: Explore as Far as Possible (F)



Step 6: Explore as Far as Possible (D)



Now try the left branch, if it exists!

Depth First Search (DFS) Algorithm Reminder

1. **Initialization:** Choose a starting node and mark it as visited.
2. **Step:** Explore as far as possible along each branch before backtracking.
3. **End Condition:** Repeat until all nodes are visited. Do not visit repeat nodes.

You can imagine how DFS could be used to obtain a list of people person A is associated with in a social media context!

Removing Cycles Optimally

Network Connectivity: Imagine you are a network engineer tasked with renovating a consumer phone network. The company that hired you has an old network with a lot of repeat connections. Some wires are longer than others.

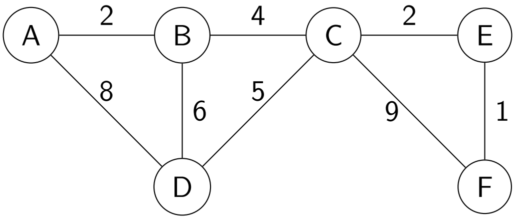
They want you to redesign the network in a way that minimizes the total amount of wire they need to maintain. These involves picking the shortest (and thus cheapest) wires to keep as well as removing repeat connections.

Removing Cycles Optimally!

Break into groups, and think of an algorithm to remove the cycles from this graph in a way that minimizes the total edge weight of the graph. You have 10 minutes.

You can use extra abstract storage devices.

Hint: It's similar in thinking to DFS. How should you use the weight of each edge in deciding which one to traverse?

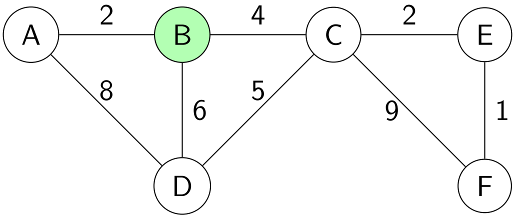


The Algorithm

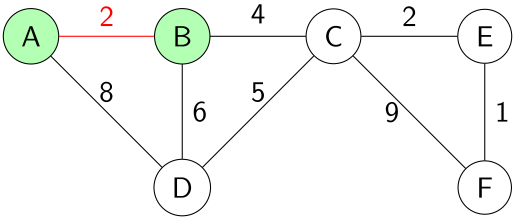
1. **Initialization:** Choose a starting node and mark it as visited.
2. **Step:** Given all the distances that grow the cut, pick the cheapest.
3. **End Condition:** Repeat until all nodes are visited.

This is known as **Prim's Algorithm**. It's a **greedy** algorithm because it makes the best choice at each step.

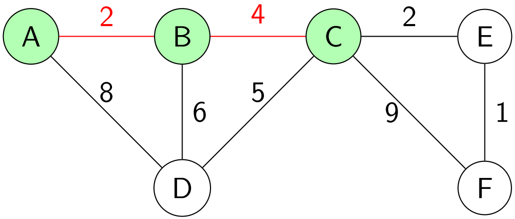
Prim's Algorithm: Step 1



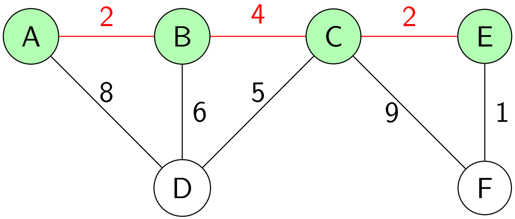
Prim's Algorithm: Step 2



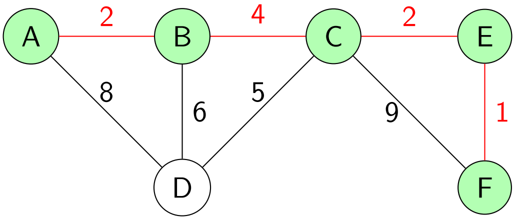
Prim's Algorithm: Step 3



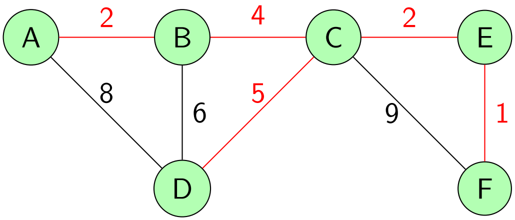
Prim's Algorithm: Step 4



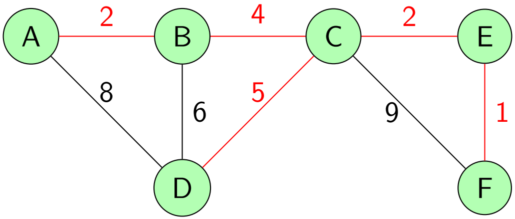
Prim's Algorithm: Step 5



Prim's Algorithm: Step 6

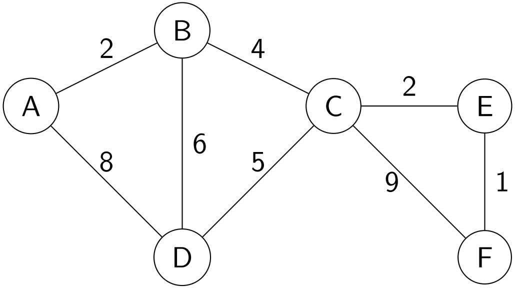


Prim's Algorithm: Result



Congrats, you just made a **Minimum Spanning Tree**. Given a cyclic graph, we converted it into a tree that costs the least amount to send information over. In other words, we spanned (aka reached) all the vertices of the graph using the minimal total edge weight.

Example Graph



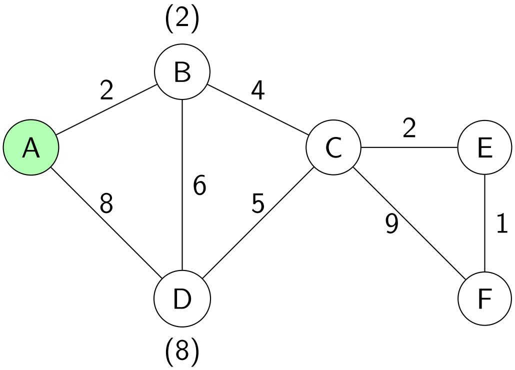
Arby's to Friendly's: I have a massive disdain for Arby's. Let's say I want to get from vertex A to vertex F as quicky as possible! You can imagine how the edge distances are miles and the other vertices are intersections. Design an algorithm to obtain the shortest path from vertex A to F.

The Algorithm

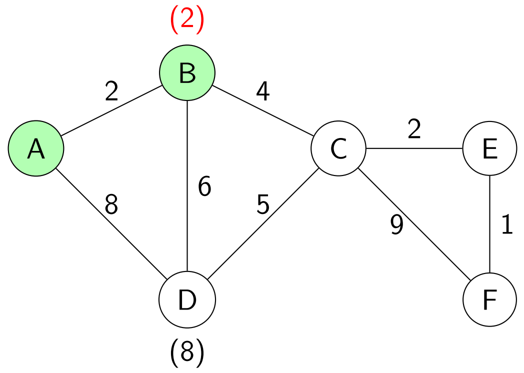
- Similar to Prim's Algorithm from before.
- Instead, our condition to grow the cut will be smallest cumulative edge distance rather than smallest edge distance!

1. **Initialization:** Choose a starting node and mark it as visited.
2. **Step:** Given all the cumulative distances to reach a new node (grow the cut), visit the cheapest one.
3. **End Condition:** Repeat until the target node is visited.

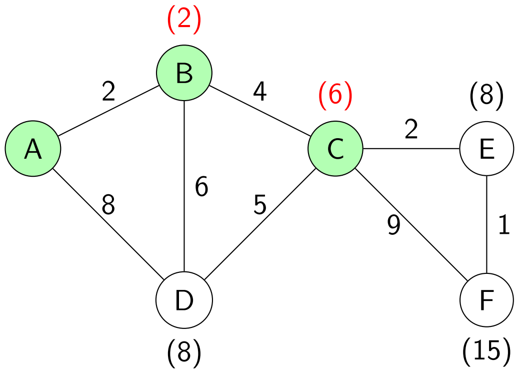
Initialization



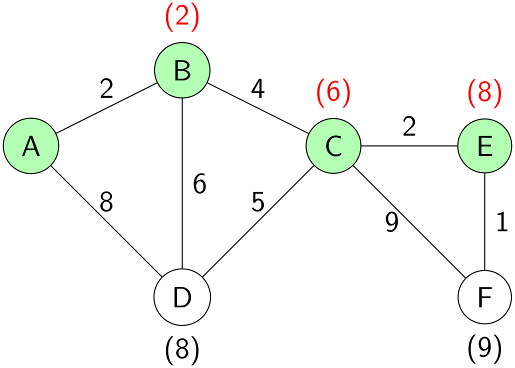
Iteration 1



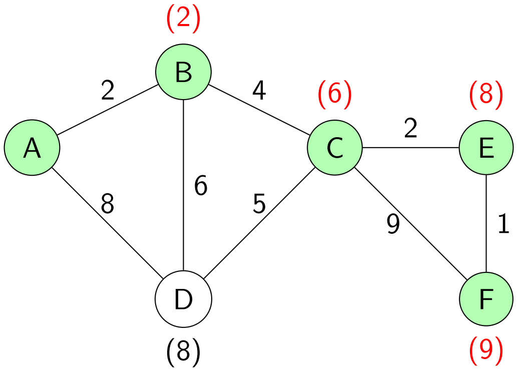
Iteration 2



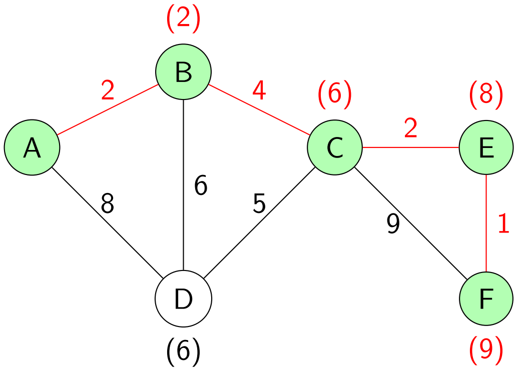
Iteration 3



Iteration 4



Result



This is **Dijkstra's Algorithm**, which is at the heart of the **A* Algorithm** used by route mapping applications like Google Maps.

What's a word I can use to describe this algorithm?

Conclusion

- Congrats, you just got a crash course in graph theory.
- You learned some basic graph terminology and some of the algorithms behind the applications you use every day!
- You even put yourselves in the shoes of a computer scientist trying to solve a theoretical problem.
- There are a lot of questions I didn't have time to answer
    - How do databases represent graphs to store? (e.g. with Matricies)
    - How do we actually maintain these conditions the algorithms need. Which structures are used to determine the cheapest cut-growing edge for instance?
    - What do these algorithms actually look like in code?
    - Why are these algorithms correct? (proofs)
- There are also tons of other graphs that solve interesting problems that we didn't have time to explore!

What do I do with this knowledge?

Keep exploring these types of problems!

- Algorithms Illuminated (Tim Roughgarden, Stanford)
  Free videos at algorithmsilluminated.org

See if CS interests you

- CS50 online, from Harvard - pretty good bitsized look at lots of CS fundamentals.

Get comfortable programming, start bringing these ideas to life!

- 100 Days of Python (Goes on sale on Udemy for reasonable prices, don't pay full price lol).
- Neetcode.io (Start easy, work to medium)

Enjoy the rest of Splash!