Theory and Applications of Graphs

EN 500.111

Outline

- Introductions
- Course logistics
- Graph terminology

Outline

- → Introductions
- → Course logistics
- → Graph terminology

About Me

Melanie Kirsche (she/her/hers)

- ➔ 4th-year Ph.D. student in Computer Science (Schatz lab)
- ➔ Previously studied computer science and math with a focus on algorithms and theory
- ➔ Research focus
  - ◆ Algorithms and data structures for computational genomics
  - ◆ Developing methods for analyzing structural variation in DNA sequences

Introductions

Please introduce yourself with the following:

- ➔ Name
- ➔ Major
- ➔ Where you’re from
- ➔ Either a fun fact about yourself or something positive that happened to you recently

Outline

- Introductions
- Course logistics
- Graph terminology

Course Goals

1. Introduce graph theory through a practical lens
2. Teach techniques for evaluating algorithms
3. Provide a (virtual) forum for students to interact and build community

Grading and Attendance

- Grade is pass/fail and based on two things
  - Attendance is **required**
    - First absence will be excused regardless of the reason
    - I will be understanding if special circumstances come up, but let me know!
  - Group presentations the final day of class
    - Details will be given about halfway through the semester

Other Notes

- Use Zoom’s hand raise feature if you have questions at any time
- The lectures are being recorded and will be available on Piazza
- Keeping your camera on is encouraged but not required

Useful Links

- ➔ Website: https://github.com/mkirsche/TAG2020
- ➔ Piazza: http://piazza.com/jhu/fall2020/en500111
- ➔ Welcome Survey: https://forms.gle/cYieAp8yH6125Lph8

Any questions about course logistics?

Outline

- → Introductions
- → Course logistics
- → Graph terminology

Data Structures

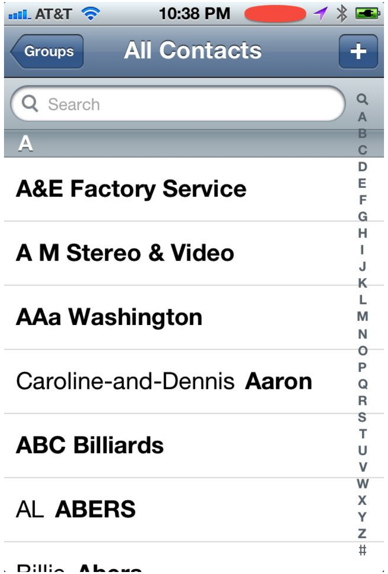
A data structure (according to Wikipedia) is “a data organization, management, and storage format that enables efficient access and modification”.

Two main practical components:

- ➔ The type of data being stored
- ➔ How it gets accessed

Example - Smart Phone Contact List

- ➔ **Type of data being stored**: everyone you know and their phone number
- ➔ **How it gets accessed**: given a person or company name, you want to know their phone number

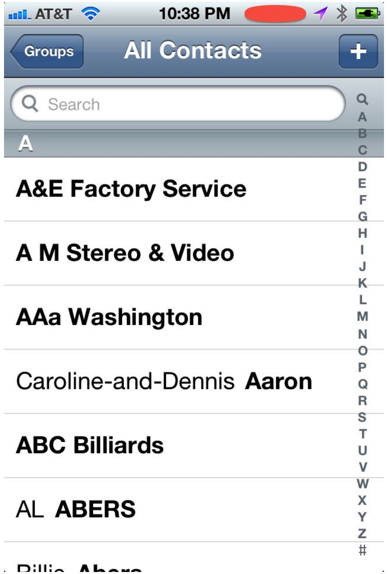


Example - Smart Phone Contact List

Given the specific access pattern, the data structure helps by:

- ➔ Sorting the names
- ➔ Letting you jump through the list by first letter

What are some other examples of data structures you interact with every day?

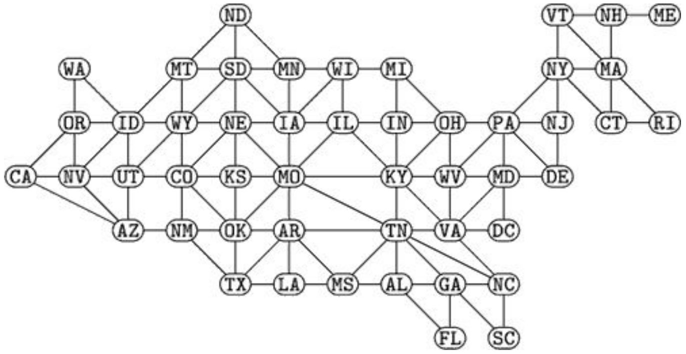


Graphs

A **graph** is a type of data structure which stores relationships between pairs of items

➔ "Item" and "relationships" can be many different things depending on the application
➔ The way we want to access this data also varies - we will discuss this more as we talk about using graphs

What are some other examples of graphs you could make,and what items and relationships are they made up of?



A graph of the continental US showing which states share borders (Credit: Wolfram Alpha)

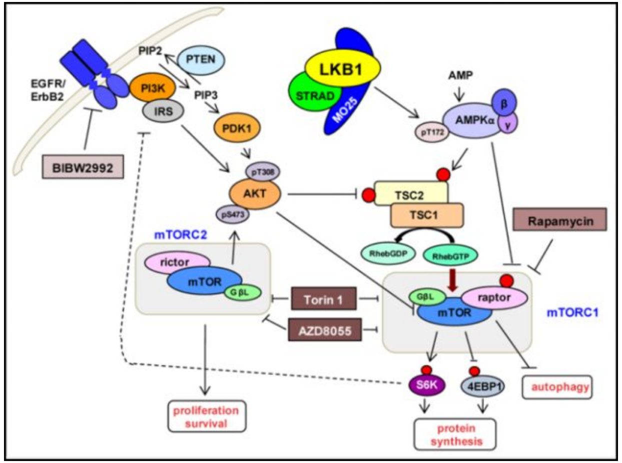
Nodes and Edges

- A **node**, or **vertex**, is one of the items we are studying the relationships between
- An **edge** is a representation of a relationship between two nodes/vertices
- We say an edge is **incident to** those two vertices, or that the two vertices it connects are **adjacent**.

Types of Edges

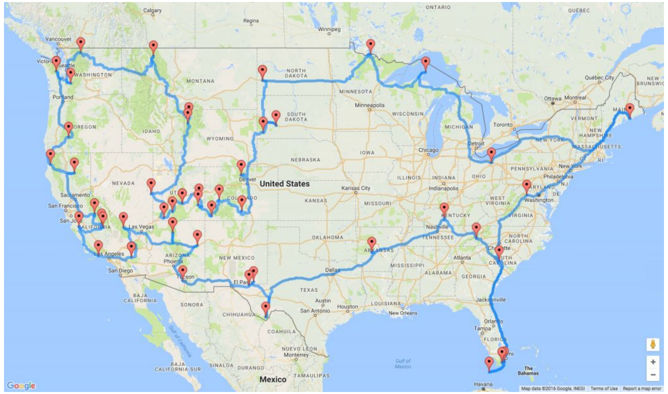
Edges can be:

- ➔ **Weighted** with a number that quantifies the relationship between a pair of items (e.g., distance)
- ➔ **Directed** from one node to another to show directional relationships (e.g., steps in a process where one happens before the other)



Example of a biological signaling pathway (Andrade-Vieira et. al. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0056567)

Weighted Graphs for Optimal National Park Hopping



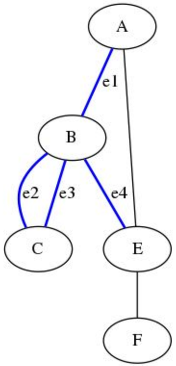
This map show a graph with national parks as nodes and routes between them as weighted edges which was used, along with an algorithm for solving the Traveling Salesman Problem, a classical graph problem, to find the shortest road trip which visits all the national parks in the continental US.

http://www.randalolson.com/2016/07/30/the-optimal-u-s-national-parks-centennial-road-trip/

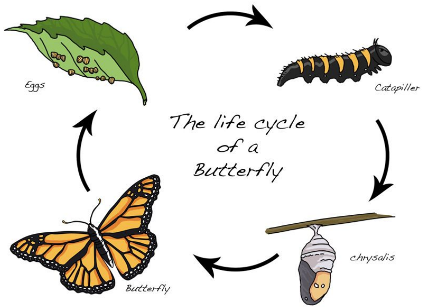
Paths and Cycles

➔ A **walk** is a sequence of edges which join a sequence of vertices (it can reuse edges and vertices)
➔ A **path** is a walk which does not revisit any vertices
➔ A **cycle** is a walk whose only repeated vertices are the first and last one being the same

A walk which is not a path because it revisits vertex B (Wikipedia)



An example of a cycle (https://quizlet.com/227960189/life-cycle-of-a-butterfly-diagram/)



Cycles in Classifying Graphs

Graph classes defined by cycles [edit]

Several important classes of graphs can be defined by or characterized by their cycles. These include:

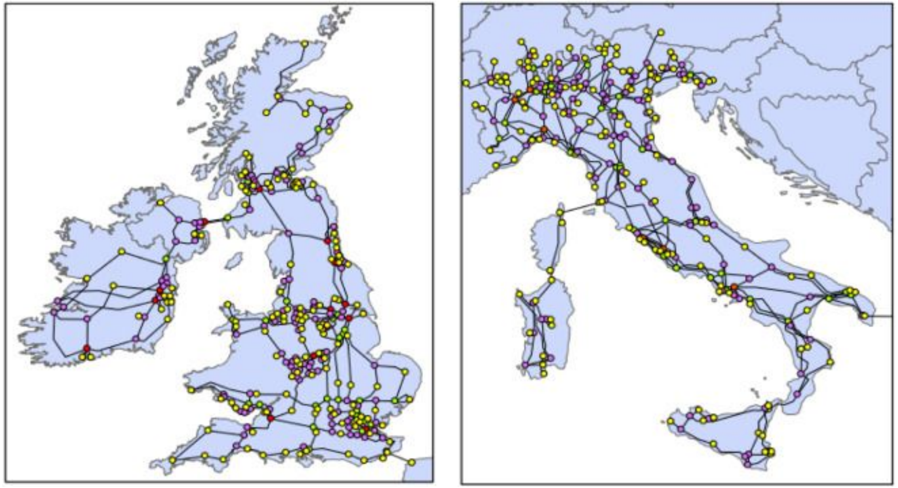
- Bipartite graph, a graph without odd cycles (cycles with an odd number of vertices).
- Cactus graph, a graph in which every nontrivial biconnected component is a cycle
- Cycle graph, a graph that consists of a single cycle.
- Chordal graph, a graph in which every induced cycle is a triangle
- Directed acyclic graph, a directed graph with no cycles
- Line perfect graph, a graph in which every odd cycle is a triangle
- Perfect graph, a graph with no induced cycles or their complements of odd length greater than three
- Pseudoforest, a graph in which each connected component has at most one cycle
- Strangulated graph, a graph in which every peripheral cycle is a triangle
- Strongly connected graph, a directed graph in which every edge is part of a cycle
- Triangle-free graph, a graph without three-vertex cycles

A list of some types of graphs which are defined by their presence or absence of cycles (Wikipedia)

Cycles are a big part of how we classify different types of graphs. We will talk about a few of these later in the semester.

Graph Connectivity

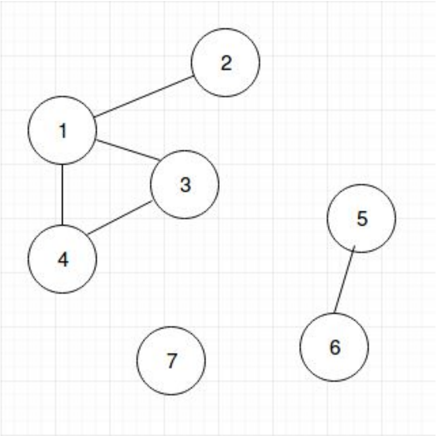
- ➔ A graph is called **connected** if there is a path between every pair of vertices
- ➔ There are also stronger versions such as bi-connected or two-connected which refer to whether a graph will remain connected if any one edge or vertex is removed
- ➔ Important in internet/utility/road networks to ensure robustness to blockages or outages



A graph of electrical networks in Europe, which was used as a case study to evaluate new stability metrics (Shahpari et. al., 2019, https://www.sciencedirect.com/science/article/abs/pii/S0378437118309993)

Connected Components

➔ If a graph is disconnected, it can be partitioned into **connected components**, or connected subgraphs with no edge incident to two vertices in different groups



An example of a graph with three different connected components:

➔ {1, 2, 3, 4}

➔ {5, 6}

➔ {7}

(http://sleepincode.blogspot.com/2017/07/finding-connected-components-using-dfs.html)

Connected Components in Image Processing

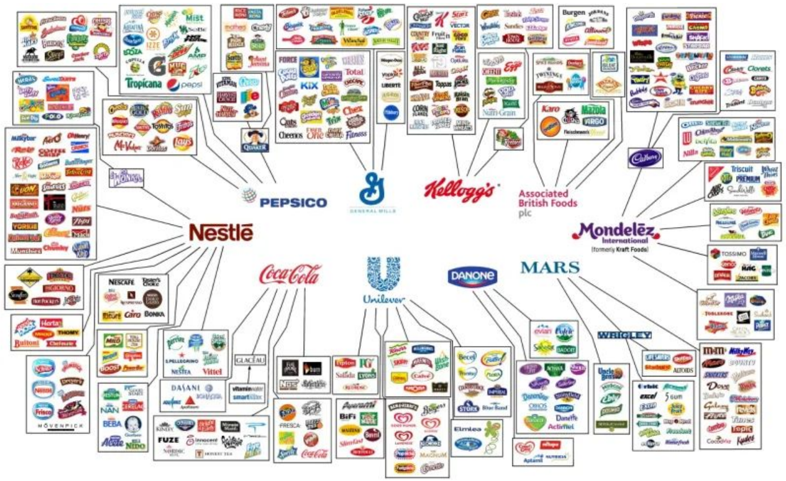
Using connected component finding to distinguish different objects in an image.

1. Pixels from original image are thresholded by intensity to get a binary image.
2. Form graph where the vertices are white pixels and there are edges between pixels which are adjacent.
3. Pixels are colored by connected component.

(https://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm)

Trees - Connected Acyclic Graphs

- A **tree** is defined as a graph which is connected and has no cycles
- If a tree has *n* vertices, it always has *n-1* edges - why?
- A **forest** is a (possibly disconnected) graph in which every connected component is a tree



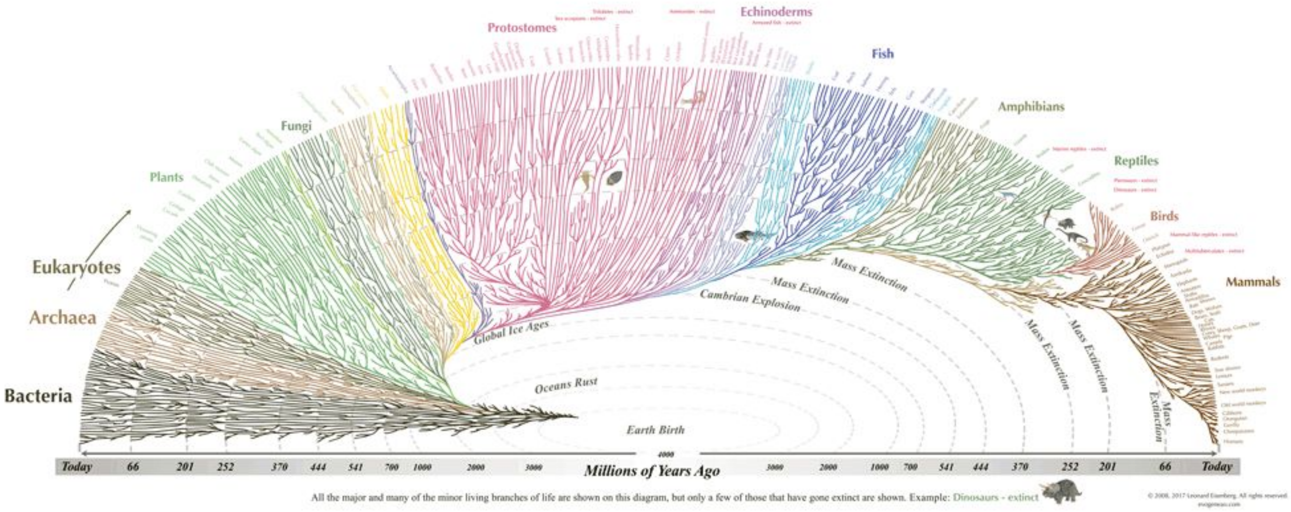
A forest of company ownership for consumer goods. There is an edge between companies if one is a parent company of the other. (https://gizmodo.com/fascinating-graphic-shows-who-owns-all-the-major-brands-1599537576)

Why do trees have n-1 edges?

Consider adding the edges one at a time

1. Initially you have *n* disconnected nodes
2. First edge groups two of those together so you have n-1 connected components
3. Second edge merges two of those components together so you have n-2 components
4. After n-1 steps you have a single component so the graph is connected

Another Tree Example - Biological Evolution

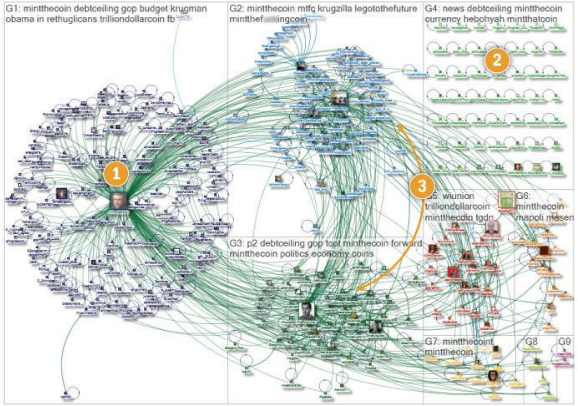


A tree representing the evolution of different biological species (credit evogeneao.com)

Node Degree

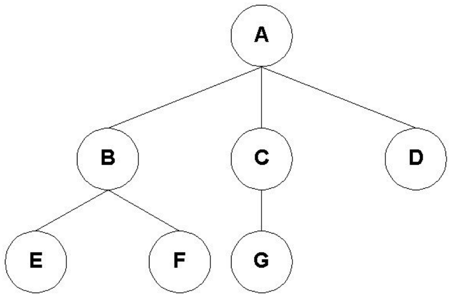
- ➔ The **degree** of a node is the number of edges which are incident to it
- ➔ In directed graphs this is often split up into **in-degree** (number of edges directed into the node) and **out-degree**.

Nodes with high degree are often of interest in social networks. Shown is a graph of users of tweeted the link to a specific news article, with edges representing retweet and follows relationships. A few high-degree vertices are “hubs” such as news network accounts who interact with many users who are otherwise disconnected (https://www.pewresearch.org/internet/2014/02/20/part-2-conversational-archetypes-six-conversation-and-group-network-structures-in-twitter/#network-type-5-broadcast-networks)



Questions - Discuss in Groups of 2-3

- Given an undirected graph, how would you check whether or not it has a cycle?
- How would you check whether a particular edge is part of any cycles?
- In the tree below, suppose you wanted to take a walk that starts at A, visits every node, and returns to A in as few moves (edges) as possible. What walk would you take?
- In the previous question, are there multiple answers? How many?



Any other questions?

Upcoming

- ➔ Algorithms for the shortest paths in weighted and unweighted graphs
- ➔ Using graphs to navigate between real-world locations
- ➔ Readings for next week on course website