Lecture 22

*Digraph = directed graph*

Shortest Path

Shortest ‘weighted’ path - Google Maps, etc

Cliques - Find set of nodes such that there exists a path connecting each node in the set

Minimum Cut - Given a graph and given 2 sets of nodes, find the minimum number of edges such that if those edges are removed the 2 sets are disconnected, (i.e. you can’t get from a member of one set to a member of another set.)

Find index patient (patient Zero) - does there exist a node such that Node has TB (or tested positive) and is connected to every node with TB. If there is no such node then you know there is no single source of the disease in the community

graph theory problem for solving the best way to distribute my limited supply of vaccine?

- Minimum Cut -
  - Infected is one set of nodes
  - uninfected is another set of nodes
  - find the edges that need to cut to separate them
  - then vaccinate somebody on one side of the edge so they don’t contract the disease

Social Networks

- whats the shortest path to Donald Trump? Sex Degrees of Separation

Recursive Depth-First Search (DFS)

- Recursion ends when start == end
- recursion part starts by choosing one child
  - continues until it reaches a **node with no children**, **reaches destination**, or **reaches a visited node**
- Avoid cycles
- Uses backtracking

**Memoization**

- use to solve problems where you remember what the answer was, and instead

of recalculating, you look it up
    - **table lookup**
- This technique is at the heart of **Dynamic Programming**
- **Why is it important?**
    - It helps us avoid redoing work that we have already done, thereby making programs more efficient. It is the key idea behind the programming techniques known as **dynamic programming**

**Dynamic Programming**

- one of the most important algorithms that we know today
- used over and over again to provide practical, efficient solutions to optimization problems that appear intractable, exponential
- most of the times, takes exponential problems and solves it really quickly
- can use it (not all the time) on problems that exhibit 2 properties:
    1. optimal substructure:
        - can find globally optimal solutions by combining locally optimal solutions
    2. overlapping sub-problems -
        - gives an indication of how much of a speedup we can expect to achieve (i.e., how many problems will we not have to solve because we can look up the solution?)