Graph Algorithms: Part 1

Dr. Baldassano
chrisb@princeton.edu
Yu's Elite Education

Last week recap: Machine Learning

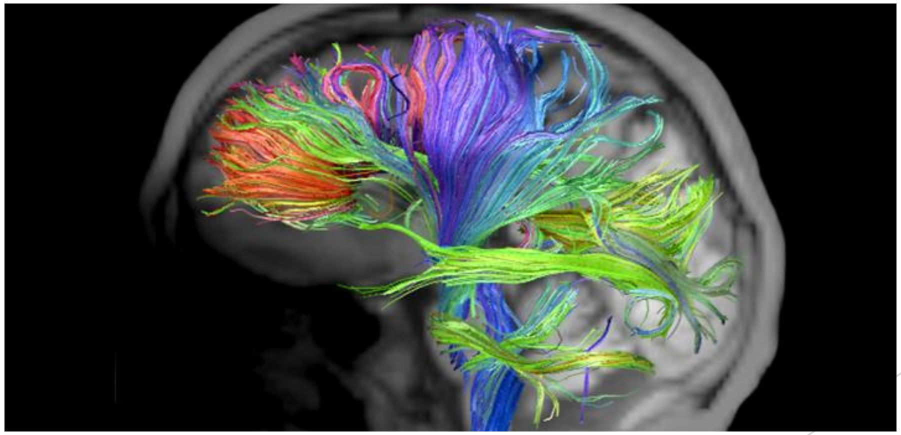
- Unsupervised learning: cluster datapoints without labels
  - K-means clustering
  - Hierarchical clustering
- Supervised learning: learn to predict labels of datapoints
  - Classification
  - Regression

Homework: Titanic ages

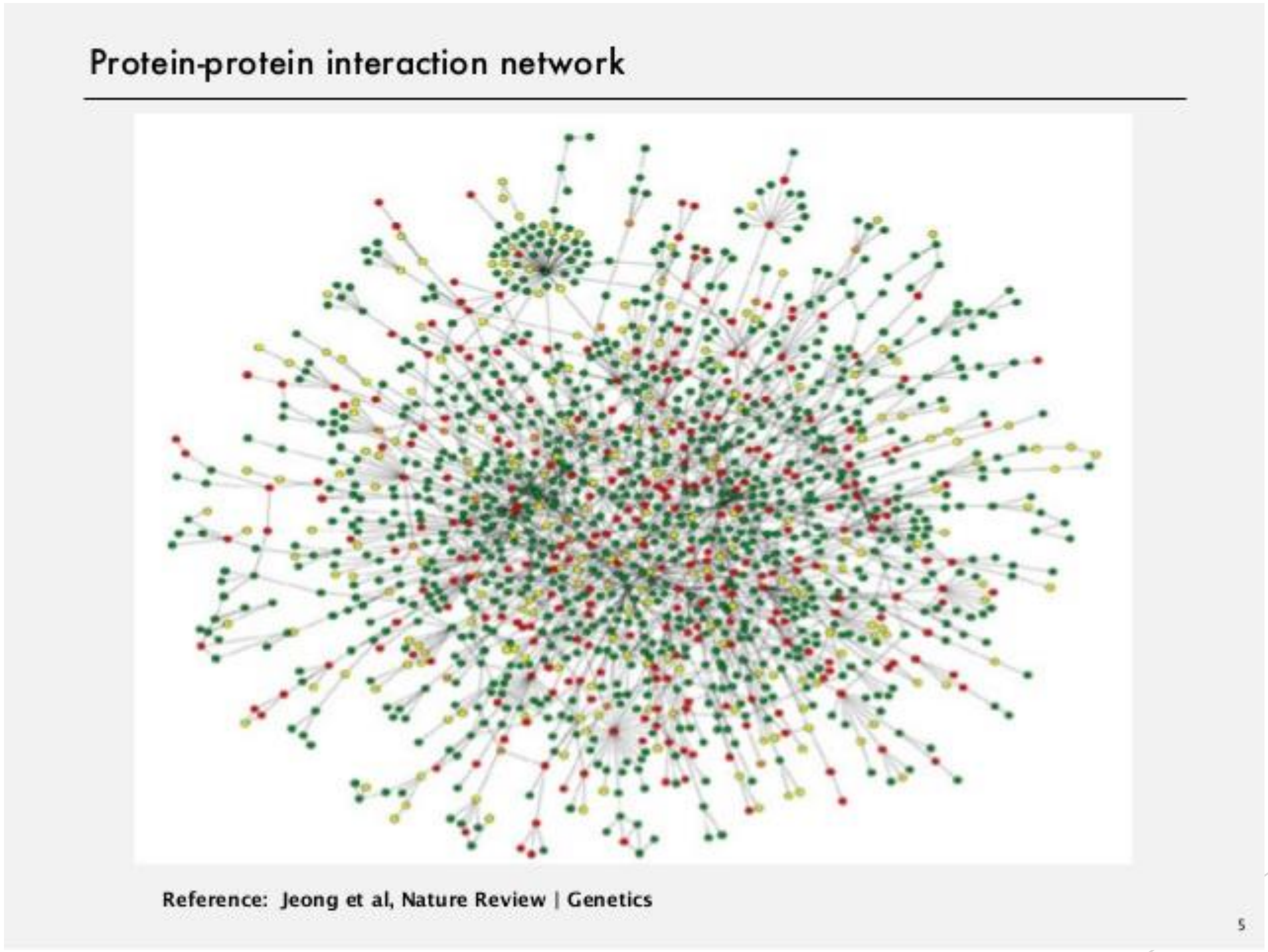
- Download www.chrisbaldassano.com/class/titanic.txt which gives age of each passenger and whether they survived (1) or died(0)
- Generate the best (one-layer) decision tree on this data that gives the highest accuracy
- E.g. if age < 20 guess survived, else guess died
- I can get ~62% accuracy

Modeling relationships

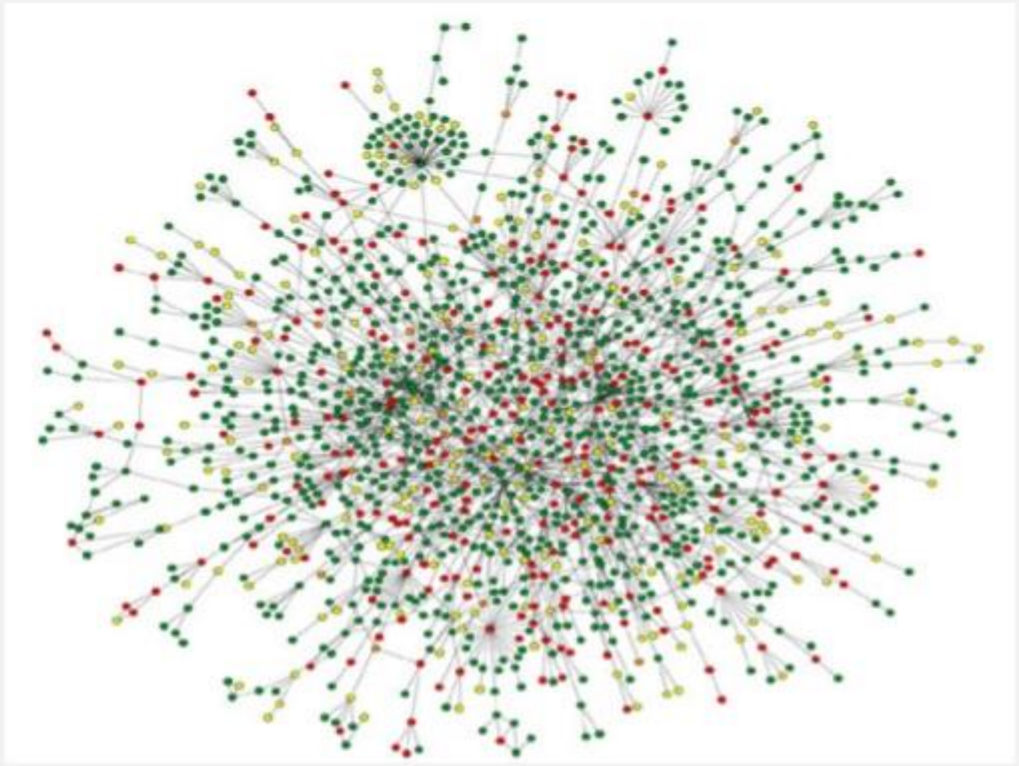
- Many real-world datasets involve relationships between pairs of items
  - Distances between locations
  - Social networks
  - Moves between game states
  - Neuron fibers connecting brain regions



Protein-protein interactions



Protein-protein interaction network



Reference: Jeong et al, Nature Review | Genetics

5

Social networks

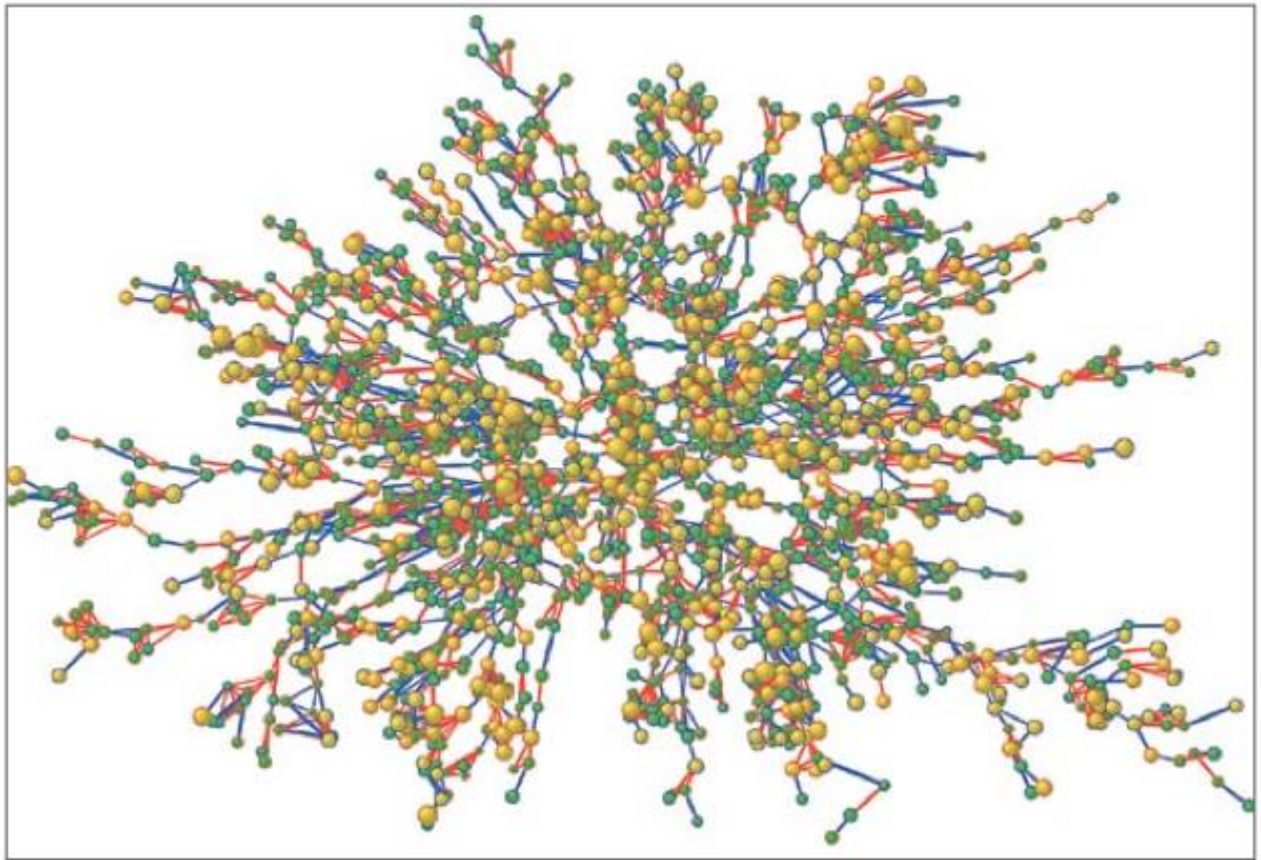
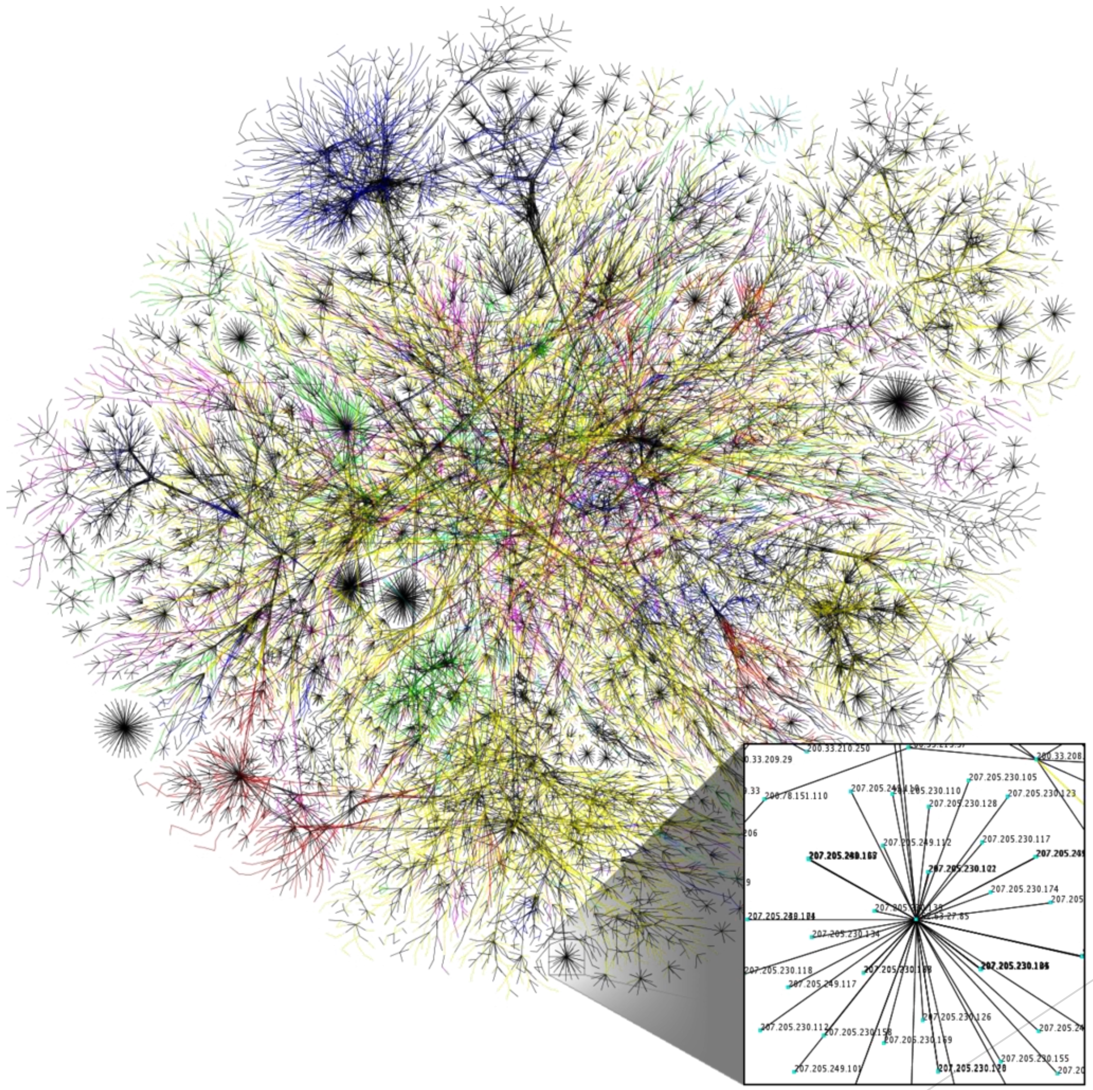


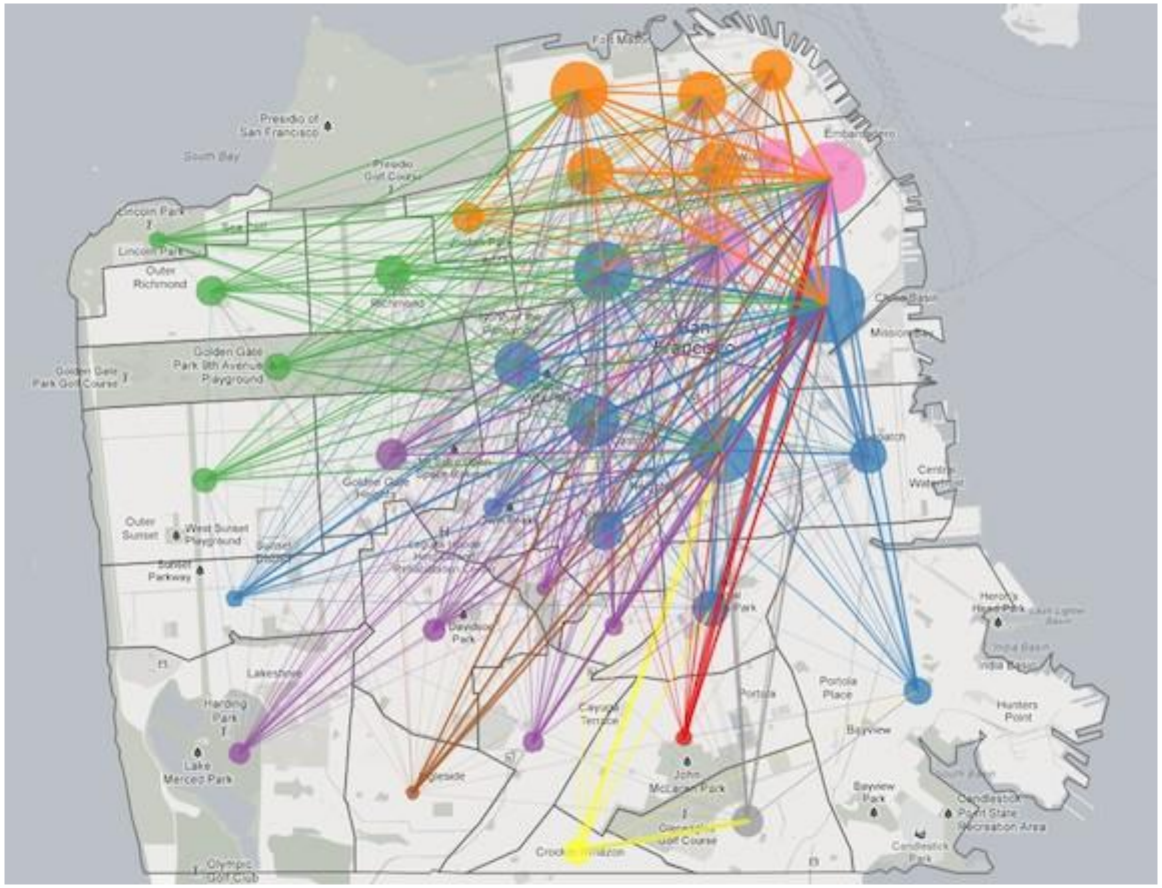
Figure 1. Largest Connected Subcomponent of the Social Network in the Framingham Heart Study in the Year 2000. Each circle (node) represents one person in the data set. There are 2200 persons in this subcomponent of the social network. Circles with red borders denote women and circles with blue borders denote men. The interior color of the circles indicates the person's obesity status: yellow denotes an obese person (body-mass index, >30) and green denotes a nonobese person. The size of each circle is proportional to the person's body-mass index. The colors of the ties between the nodes indicate the relationship between them: purple denotes a friendship or marital tie and orange denotes a familial tie.

Facebook

The Internet

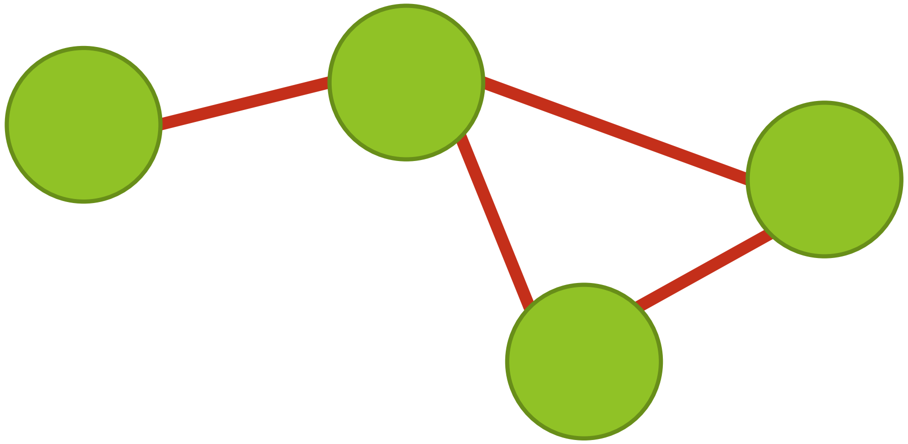


Uber rides



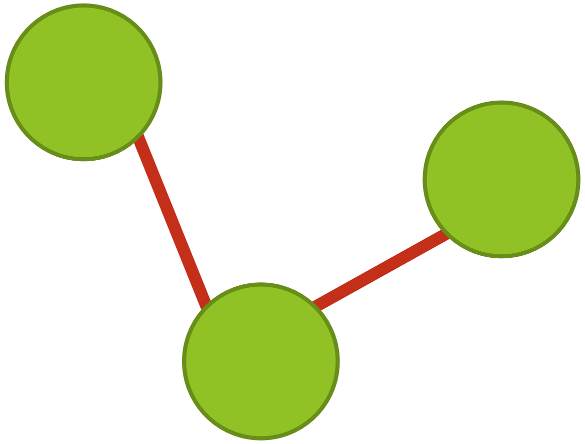
Graphs

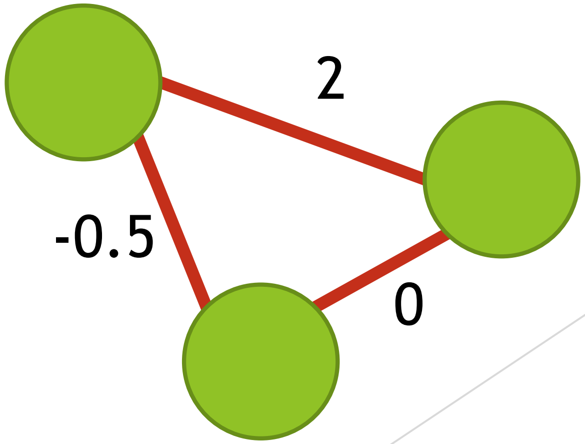
- In Computer Science we describe pairwise relationships as a “graph”
- Graphs are made up of two types of things:
  - Nodes (or vertices), which represent items
  - Edges, which represent relationships



Types of graphs

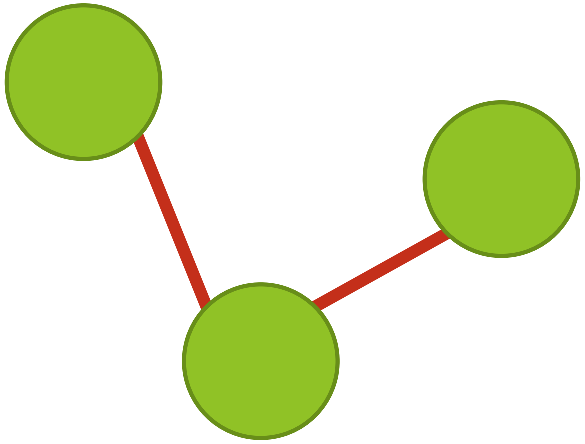
- Weighted vs. unweighted
  - Unweighted graphs have edges that either exist or don’t exist, between each pair of nodes
  - Weighted graphs have a edges with a real-valued strength

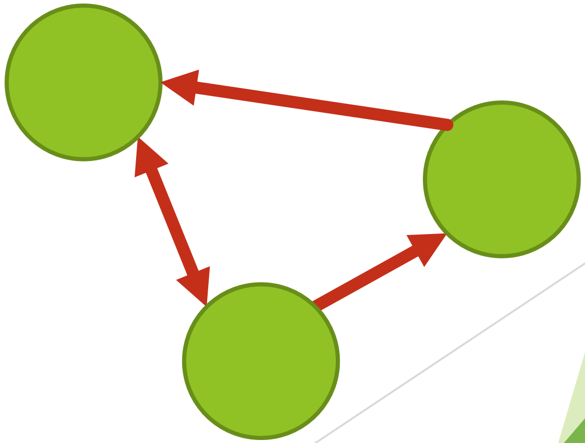




Types of graphs

- Directed vs. undirected
  - Undirected graphs have edges that are symmetrical – edge(a,b) = edge(b,a)
  - Directed graphs have edges with different strengths in each direction





Representing graphs

- Two ways to store graphs in a computer:
  - Adjacency matrix: adj[a][b] = edge between nodes a and b
    - If unweighted, adj[a][b] = 0 or 1
    - If undirected, adj[a][b] = adj[b][a]
  - Adjacency list: adj[a] = set of a’s neighbors (nodes with edges connected to a)
    - For weighted graph, also need to keep track of weights

Algorithms on Graphs

- Kinds of algorithms on graphs:
  - Finding shortest paths between nodes
  - Finding clusters
  - Finding spanning trees
  - Finding important nodes
- We'll focus on shortest paths today, look at other problems next week

Shortest paths

- Why do we care about finding the shortest path between nodes?
  - Distance graph: shortest path is literally the shortest distance to travel (e.g. airplane flights)
  - Graph of robot states: shortest path is most efficient sequence of actions to reach goal
  - Social network: find how closely two people know each other (e.g. Bacon number)
  - Computer network: shortest path is best route for sending information between two servers

Starting simple

- Let's take the simplest case: in undirected, unweighted graph, find shortest path from node a to node b

Adding weights

- What happens if edges are weighted?
- Need to replace queue with priority queue (often implemented as heap)
- This is Dijkstra’s algorithm

Adding hints

- So far we’ve looked at “blind” searches that don’t know anything about what nodes are likely to be on the solution path
- In some cases, like physical distances, we have a good guess about which way to explore
  - https://qiao.github.io/PathFinding.js/visual/

Informed search: A*

- Dijkstra: expand node with lowest distance from source, g(n)
- A*: expand node with lowest estimated distance from source to destination
  - f(n) = g(n) + h(n)
- New term h(n) is a *heuristic* estimate of distance from node n to destination
- For algorithm to work, heuristic must always be optimistic (underestimate distance to destination)

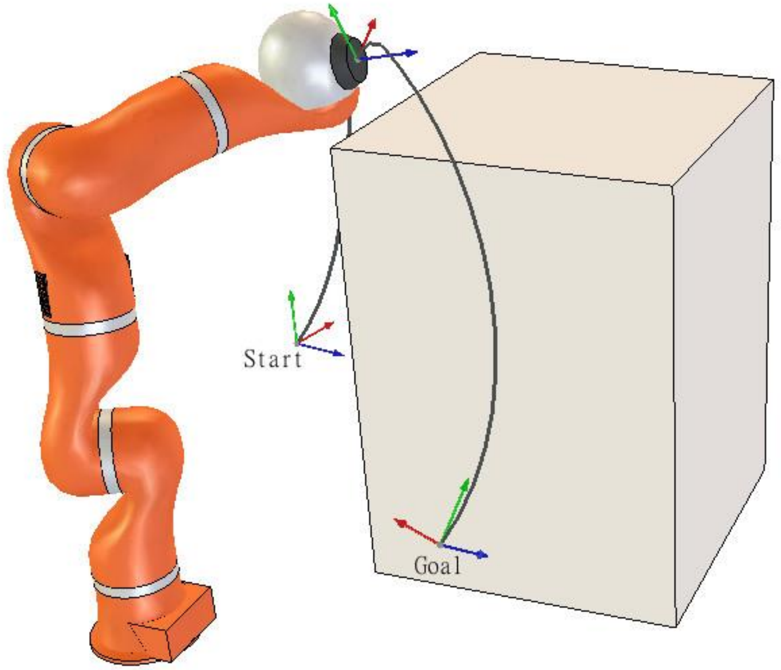
A* with distances

- For physical distances, we usually define h(n) = straight-line distance from n to goal
- This will always be optimistic, since it ignores obstacles
- https://qiao.github.io/PathFinding.js/visual/
- Note that Dijkstra is A* with the very optimistic heuristic h(n)=0

Example: NJ cities

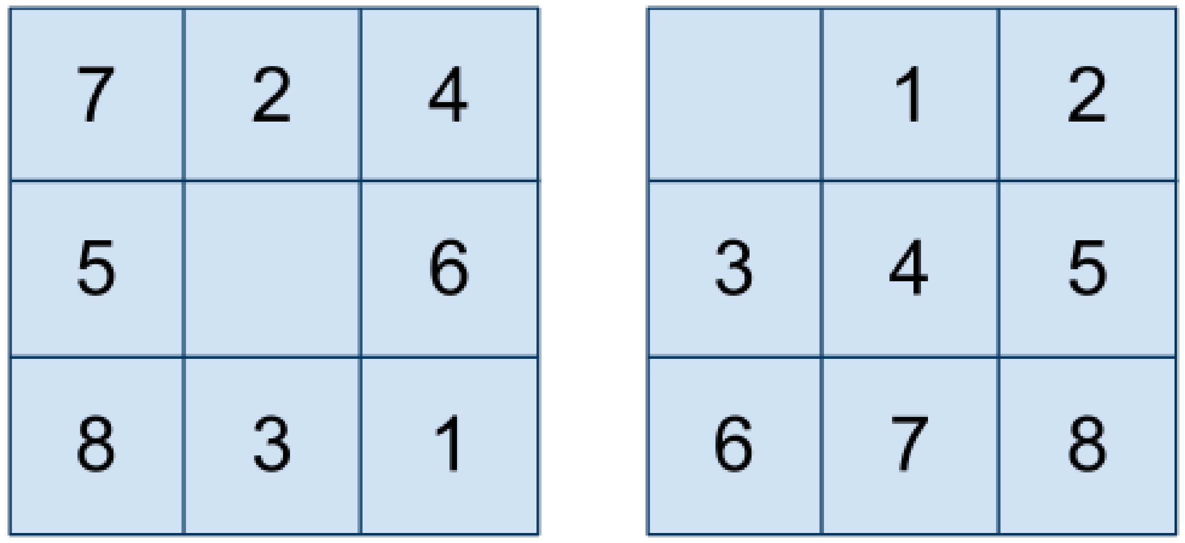
Heuristics in other cases

- Motion planning: distance to goal state, ignoring obstacles



Heuristics in other cases

- Game playing: number of misplaced items



7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Learning heuristics

- Can use machine learning methods from last week to learn heuristics for complicated situations
- For example, given some features about current chess position (pieces still on the board, number of pieces currently threatened…) can learn to predict number of moves to checkmate

Olympiad Problems

- Piggyback: http://www.usaco.org/index.php?page=viewproblem2&cpid=491
- Cow route: http://www.usaco.org/index.php?page=viewproblem2&cpid=51

Homework: Flight routes

- Download the time-table of flights:

  [Departing] [Depart HHMM] [Arriving] [Arrive HHMM]

- Given a starting and ending city, compute the fastest set of flights to get from one to the other, assuming you need 60 minutes between flights
  - For example, Paris to Houston