Graph Algorithms: Part 2

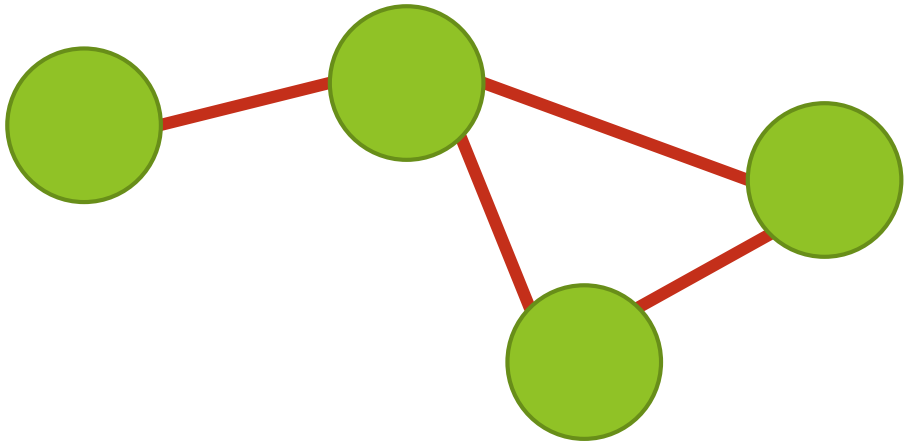
Dr. Baldassano

chrisb@princeton.edu

Yu's Elite Education

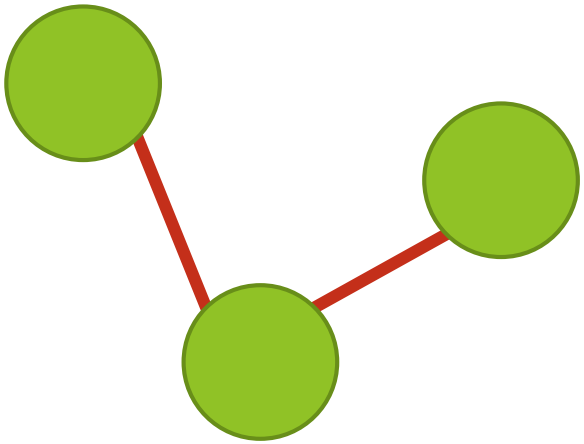
Graphs

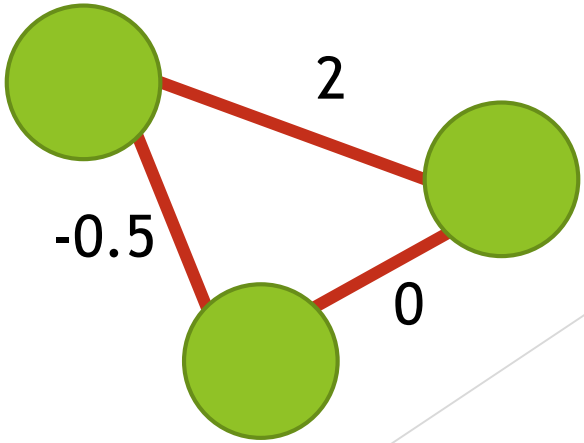
- In Computer Science we describe pairwise relationships as a “graph”
- Graphs are made up of two types of things:
  - Nodes (or vertices), which represent items
  - Edges, which represent relationships



Types of graphs

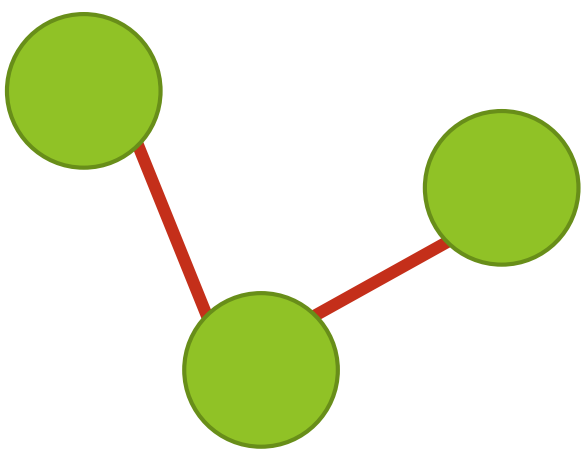
- Weighted vs. unweighted
  - Unweighted graphs have edges that either exist or don't exist, between each pair of nodes
  - Weighted graphs have a edges with a real-valued strength

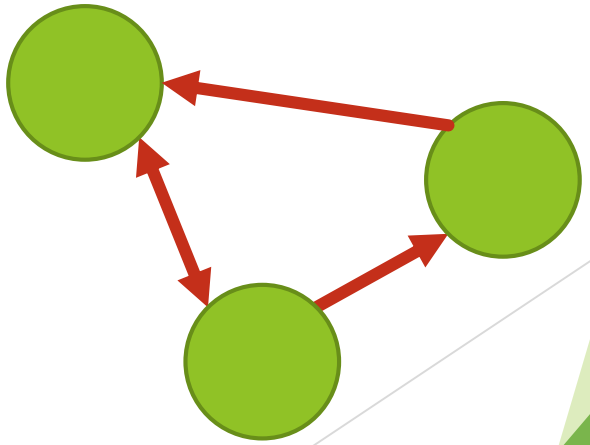




Types of graphs

- 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

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

Pagerank

- How do we determine the importance of individual nodes in a graph?
- Google became famous by coming up with a solution to this problem called “Pagerank”
- Basic idea: if a lot of important pages link to me, then I’m important!

Pagerank model

- Assume that people randomly click on links in webpages
- Where will most people end up?
- Example:
  - Page A has 4 links, 1 of which is to C
  - Page B has 2 links, 1 of which is to C
  - PR(C) = PR(A)/4 + PR(B)/2

Pagerank model

- One other tweak: assume that with probability 1-d, people just jump to a random website
  - This avoids issues with pages that don't have links, and makes solving easier
- Final equation:

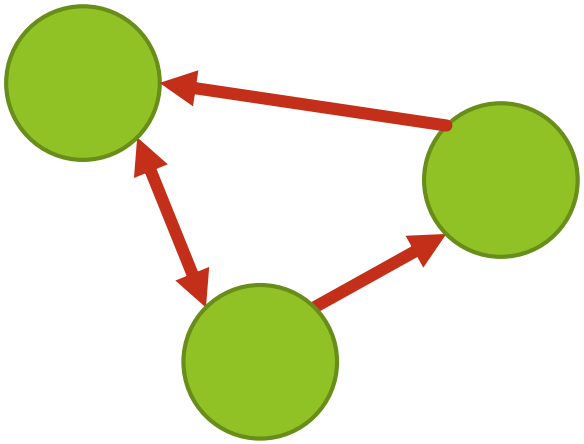
$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right).$$

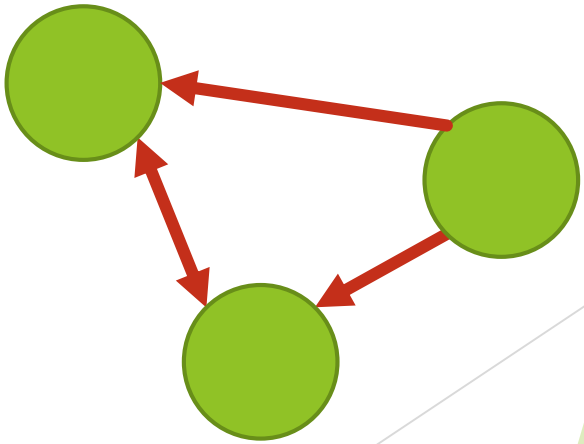
Calculating Pagerank

- Pagerank has a circular definition, so it is hard to calculate
- Can start with uniform Pageranks on all pages, then iterate Pagerank equations
- Let's code Pagerank for wikipedia

Strongly connected nodes

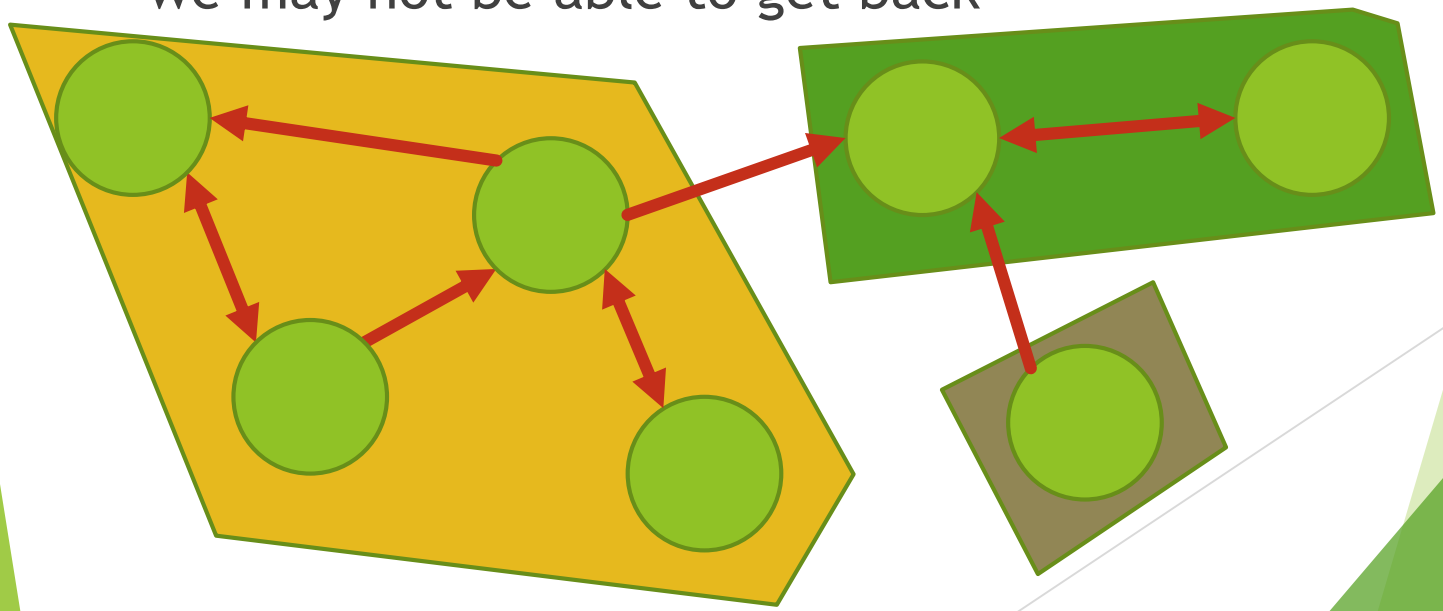
- In some directed graphs, we can move from every node to every other node
- We call this a “strongly connected” graph
- Which of these graphs are strongly connected?





Strongly connected components

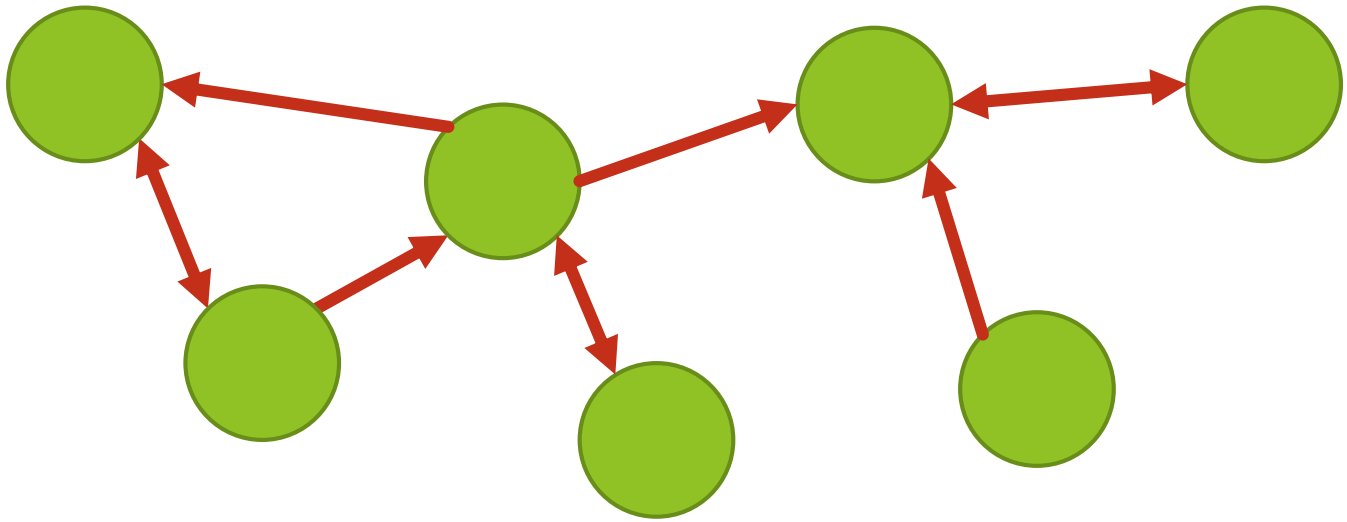
- We can divide any graph into components that are strongly connected
- We can “move freely” within these components, but once we leave a component we may not be able to get back



Tarjan's algorithm for SCC

- Calculate 2 numbers at each node:
  - An index
  - Lowest index in my SCC
- Perform DFS, update lowest index when:
  - We find a new node
  - We find an old node in our SCC
- If index == lowest index after DFS, then create new SCC with this root

Tarjan's algorithm: Example



Tarjan's algorithm: Big O

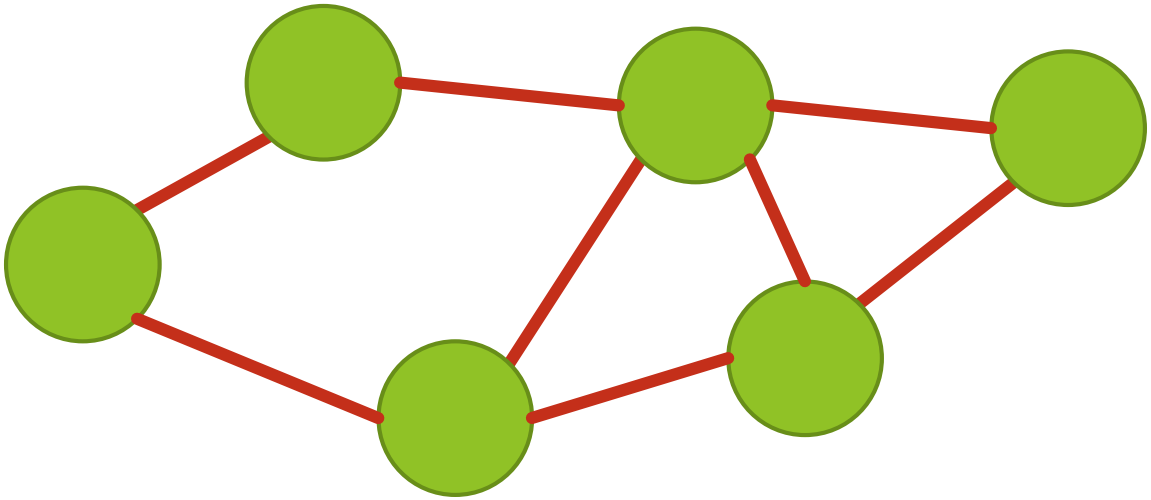
- Equivalent to two depth-first traversals (one forward, one backward)
- DFS visits every node and every edge
- O(N+E)

Olympiad Example

- Grass Cownoisseur: http://usaco.org/index.php?page=viewproblem2&cpid=516

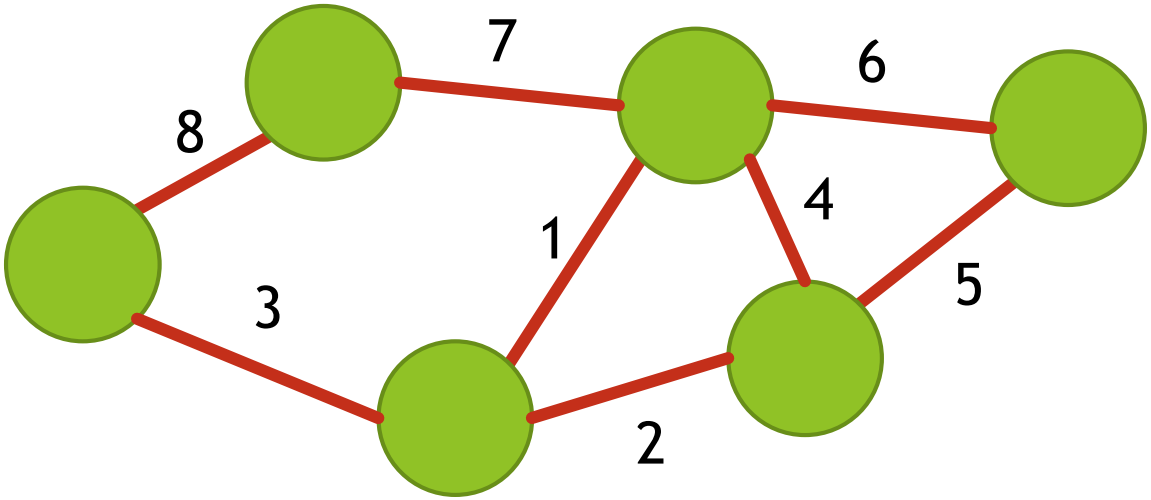
Spanning tree

- For a connected, undirected graph, how many edges can we remove and still have a connected graph?



Minimum spanning tree

- If edges have weights, we want to select the spanning tree with the smallest total edge weight



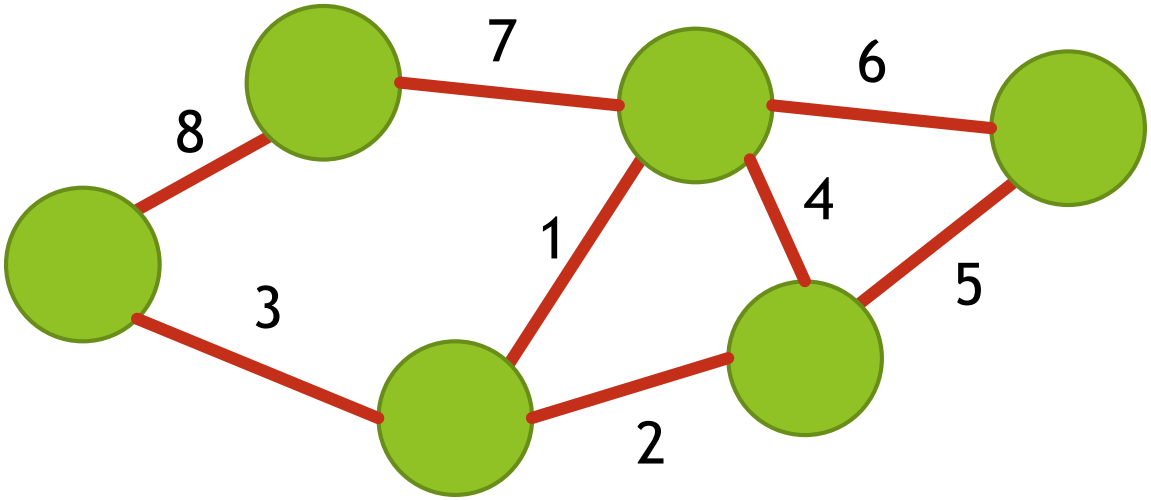
Applications of minimum spanning tree

- Broadcast paths in the internet
- Planning a road system
- Hierarchical clustering
- Taxonomy and evolutionary biology

Prim's algorithm

- Surprisingly simple!
- Pick a random starting node
- Add minimum edge
- Keep adding smallest edge that connects to an unconnected node

Minimum spanning tree



Prim's Big O

- Depends on how we search for the next edge
- If we keep a heap of nodes with keys equal to their minimum edge, we will perform a decrease-key operation for each edge on a size N heap
- O(E log N)

Homework

- Superbull: http://usaco.org/index.php?page=viewproblem2&cpid=531
- Hint: if teams are nodes, tournament can be described by a spanning tree