Topics for Today

- Introduction to graphs
- Basic operations and problems
- NetworkX
- Drawing graphs

Reading

- [https://networkx.lanl.gov/wiki/WikiStart](https://networkx.lanl.gov/wiki/WikiStart) - tutorial
- [http://people.hofstra.edu/geotrans/eng/ch2en/meth2en/ch2m1en.html](http://people.hofstra.edu/geotrans/eng/ch2en/meth2en/ch2m1en.html) - gives more formal definitions
- [http://en.wikipedia.org/wiki/Graph_theory](http://en.wikipedia.org/wiki/Graph_theory) - history and numerous links
Graphs

- Mathematical model for studying pair-wise connectivity among entities
- A graph is a discrete structure consisting of
  - nodes (also called vertices, sites) and
  - edges between nodes (also called links, connectors, arcs)
- Graphs are used to model problems in almost every discipline
Two graphs
<table>
<thead>
<tr>
<th></th>
<th><strong>NODES</strong></th>
<th><strong>EDGES</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication network</td>
<td>computers</td>
<td>fiber optic cable</td>
</tr>
<tr>
<td>Financial stocks</td>
<td>currency</td>
<td>transactions</td>
</tr>
<tr>
<td>Transportation</td>
<td>street intersections</td>
<td>highways</td>
</tr>
<tr>
<td>Protein interaction</td>
<td>proteins</td>
<td>protein-protein</td>
</tr>
<tr>
<td>networks</td>
<td></td>
<td>interactions</td>
</tr>
<tr>
<td>Internet</td>
<td>web pages</td>
<td>hyperlinks</td>
</tr>
<tr>
<td>Social networks</td>
<td>people</td>
<td>friendships</td>
</tr>
<tr>
<td>Software systems</td>
<td>functions</td>
<td>function calls</td>
</tr>
<tr>
<td>Games board</td>
<td>positions</td>
<td>legal moves</td>
</tr>
<tr>
<td>Chemical compounds</td>
<td>molecules</td>
<td>chemical bonds</td>
</tr>
<tr>
<td>NODES</td>
<td>EDGES</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------</td>
<td></td>
</tr>
<tr>
<td>Communication network</td>
<td>computers</td>
<td>fiber optic cable</td>
</tr>
<tr>
<td>Financial stocks</td>
<td>currency</td>
<td>transactions</td>
</tr>
<tr>
<td>Transportation</td>
<td>street intersections</td>
<td>highways</td>
</tr>
<tr>
<td>Protein interaction networks</td>
<td>proteins</td>
<td>protein-protein interactions</td>
</tr>
<tr>
<td>Internet</td>
<td>web pages</td>
<td>hyperlinks</td>
</tr>
<tr>
<td>Social networks</td>
<td>people</td>
<td>friendships</td>
</tr>
<tr>
<td>Software systems</td>
<td>functions</td>
<td>function calls</td>
</tr>
<tr>
<td>Games board</td>
<td>positions</td>
<td>legal moves</td>
</tr>
<tr>
<td>Chemical compounds</td>
<td>molecules</td>
<td>chemical bonds</td>
</tr>
</tbody>
</table>
Graph consisting of 6 nodes and 9 edges
Edges have no directions,
Edges have directions
Not a tree, but it almost looks like one
Directed graph with cycles and self-loops (edges have labels, nodes have labels)
Which of the following is not naturally modeled by a graph?

A. Facebook users and their friends
B. Spread of a disease
C. Weather prediction
D. Course requirements and prerequisites
E. Electric power distribution system
Some terminology

- Graphs have a finite number of nodes, but they can grow and shrink
- Undirected and directed graphs
- Weights on edges: weighted and unweighted graphs
- Nodes generally have names
- At most one edge between a pair of nodes: simple graph (otherwise a multi-graph)
- Self-loops are often excluded as possible edges
More terminology

- A tree is a special type of graph
- Undirected graph
  - a node is “adjacent” to other nodes
  - these nodes are also called its neighbors
- Directed graph
  - a node has edges going out and edges coming in (in-degree and out-degree of a node)
  - adjacency is often the number of edges leaving the node
- Path between two nodes – sequence of edges connecting them
Undirected, simple graph
Undirected, simple graph
How many nodes have degree 3?

A. 3  
B. 4  
C. 5  
D. 6
Basic functionality we need

- Create a graph
  - read edges from a file
  - add and delete edges and nodes
  - Perform the union graphs
  - Pull out subgraphs
- Iterate through all the nodes of a graph
- For a given node u, iterate through all its the adjacent nodes
- Print /visualize a graph
Representing Graphs

- We will not discuss the internal representation of graphs
- NetworkX will do it for you

Otherwise …

- As a 2-dimensional matrix (adjacency matrix)
  - $A[a][b] = 1$ if there exists an edge from node a to b (can also store the edge weight)
- As a list of lists (adjacency lists)
  - i-th list contains all the nodes node i is adjacent to
Problems on undirected, simple graphs

- Is graph G connected?
- If it is not connected, identify its connected components
- Is the graph a tree?
- Create a subgraph from a graph

- Explore all nodes in a specified order (depth-first-search, breadth first search)
- Find paths of shortest length between nodes
What is the minimum number of edges that have to be added to make the graph connected?

A. 1 edge
B. 2 edges
C. 3 edges
D. 4 edges
Undirected, simple with colored connected components
What is the minimum number of nodes that have to be removed to create at least two connected components of size \( >1 \)?

A. 1
B. 2
C. 3
NetworkX

https://networkx.lanl.gov/wiki/WikiStart

Read Tutorial and Quick Reference sections

- Easy to get started and intuitive to use
- Poor visualization of graphs; complete reference material overwhelming

from networkx import *
import pylab as P
NetworkX Basics

```python
G = Graph()

G.add_edges_from((tuple(s.split()) for s in open('graph1.txt')))

print G.edges()
print G.nodes()
CC = component.connected_components(G)

draw(G)
P.show()
```
Drawing graphs generated by NetworkX

```python
G_com = complete_graph(5)
G_star = star_graph(n)
G_lad = circular_ladder_graph(n)
G_hyp = hypercube_graph(4)
G_ran1 = erdos_renyi_graph(2*n, p, seed=None)

P.figure(1)
draw(G_star)

P.figure(2)
draw(G_lad)

P.figure(3)
draw_spectral(G_ran1)

P.figure(4)
draw_circular(G_com)
```
Some graph operations

G1 = complete_graph(5)
G2 = star_graph(5)
G3 = ladder_graph(4)
H = Graph()
H = union(G1, G2, rename=('G-', 'H-'))
print "len(H): ", len(H)

H = union(H, G3)
HN = convert_node_labels_to_integers(H)

list_ccHN = connected_components(HN)
print list_ccHN

sub_HN = subgraph(HN, list_ccHN[0])
Directed, simple graphs

- What does “connected” mean?
- A directed graph is strongly connected if there is a (directed) path between any two nodes
- If it is not strongly connected, identify its strongly connected components
Directed graph has 8 nodes

3 strongly connected components

\[ [[a,b,e], [f, g], [c,d, h]] \]

Connected components represented as a list of list
Operations on directed graphs

G = DiGraph()
G.add_edges_from([tuple(s.split()) for s in open('EBI.txt')])

print 'Number of nodes:', G.order()

C1 = component.strongly_connected_components(G)

print 'Number of strongly connected components:', len(C1)
print 'Strongly connected component sizes > 3: ' 
for t in C1:
    if len(t) > 3:
        print "a strongly connected component of size: ", len(t)

print