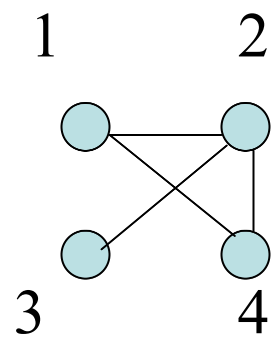
Basic Network Properties and How to Calculate Them

- Data input
- Number of nodes
- Number of edges
- Symmetry
- Degree sequence
- Average nodal degree
- Degree distribution
- Clustering coefficient
- Average hop distance (simplified average distance)
- Connectedness
- Finding and characterizing connected clusters

Matlab Preliminaries

- In Matlab everything is an array
- So
  - a = [2 5 28 777]
  - for i = a
  - do something
  - end
- Will "do something" for i = the values in array a
- [a;b;c] yields a column containing a b c
- [a b c] yields a row containing a b c

Matrix Representation of Networks



»A=[0 1 0 1;1 0 1 1;0 1 0 0;1 1 0 0]

A =

0	1	0	1
1	0	1	1
0	1	0	0
1	1	0	0

»

A is called the adjacency matrix

$A(i,j)=1$ if there is an arc from i to j

$A(j,i)=1$ if there is also an arc from j to i

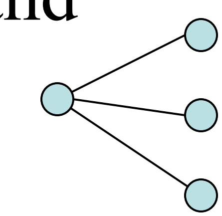
In this case A is symmetric and the graph is undirected

Directed, Undirected, Simple

- Directed graphs have one-way *arcs* or links
- Undirected graphs have two-way *edges* or links
- Mixed graphs have some of each
- Simple graphs have no multiple links between nodes and no self-loops

Basic Facts About Undirected Graphs

- Let n be the number of nodes and m be the number of edges
- Then *average nodal degree* is $<k>=2m/n$
- The *Degree sequence* is a list of the nodes and their respective degrees
- The sum of these degrees is $\sum_{i=1}^{n} d_i = 2m$
- D=sum(A) in Matlab
- sum(sum(A)) = 2m
- Valid degree sequences add to an even number



$D = [3\ \ 1\ \ 1\ \ 1]$

More Definitions and Calculations

- Geodesic - shortest path between two nodes
- Average path length = avg of all geodesics
- Graph diameter = longest geodesic

Data Input

- Edge list: each row is an arc, the numbers of the nodes it connects are in columns 1 and 2, with the first node being the source and the second being the destination of the arc
  – Ni,Nj
  – Nk,Nl
  – An edge = 2 arcs, one in each direction, requiring two entries; often only one arc is listed so you have to know if the matrix is intended to be symmetric or not
- Node list: Node #1 in column 1, the node numbers of nodes it links to by outgoing edges in the next columns, however many are necessary
  – N1,Ni,Nj,Nk
  – N2,Nl,Nm
- The easy way to generate these is to use Excel, save as ".csv" and load into Matlab using dlmread('filename.csv');

Example Node and Edge Lists

	A	B	C	D	E
1	1	23			
2	2	8			
3	3	4			
4	4	3	5	7	
5	5	4	6	11	
6	6	5			
7	7	4	9	11	
8	8	2	9	25	26
9	9	7	8	20	
10	10	72			
11	11	5	7	13	20

Part of a nodelist.
It is symmetric.

1	2
1	8
1	43
1	49
1	50
1	51
1	52
1	53
1	54
1	55
2	3
2	9
2	44
2	56
3	4
3	10
3	45
3	57
3	58
4	5
4	11
4	46
4	59
4	60

Part of an edgelist.
It is intended to be symmetric but each edge appears only once so you have to make the resulting matrix symmetric yourself!

Matlab Data Formats

- Comma delimited, space delimited, tab delimited
  – All can be read using dlmread and written using dlmwrite
- Lotus 1-2-3 format wk1 can be exchanged between Excel, Matlab, and UCINET
  – Matlab uses wk1read and wk1write
  – UCINET uses excel matrix input or small matrices can be pasted into UCINET's matrix
  – Matlab also uses xlsread and xlswrite if you don't need UCINET compatibility

Data Input and Misc Ops

- adjbuilde builds adjacency matrix from edge list
- adjbuildn builds adjacency matrix from node list
- diagnoseMatrix tests for power law
- Miscellaneous data conversion
  – adj2str adjacency matrix to Matlab data structure
  – adj2pajek for input to Pajek graph software
  – adj2inc adjacency matrix to incidence matrix
  – str2adj reverses adj2str

More Matlab Preliminaries

- Logical expressions:
- B=A > 1 returns a matrix B with entries = 1 everywhere matrix A has an entry > 1
- B is a logical matrix and can't be operated on like numerical matrices can (can't multiply, invert, etc)
- B=B+0 converts B to a numerical matrix
- Unitize(A) makes all non-zero entries in B = 1
  - function unitize=unitize(A)
  - % unitizes a matrix, makes all non zero entries = 1
  - unitize=A>0;
  - unitize=unitize+0;

Still More Matlab Preliminaries

- A’ = the transpose of A
- sum(A) adds up each column and stores the result as a row vector
- sum(A’) adds up each row and stores the result as a row vector
- sum(sum(A)) adds up all the entries in A
- length(x) counts the number of entries in vector x
- find(x logical expr) returns the subscripts of entries in x that satisfy the logical expression using linear indexing
  - Linear indexing gives every entry one subscript
- length(find(x logical expr)) tells how many entries in x satisfy the logical expression
- [i,j]= find(x logical expr) returns the i and j subscripts of entries in matrix x that satisfy the logical expression

More Preliminaries

- idx=[a b c]
- B=A(idx,idx)
- B(i,j)=A(one of the combinations of a,b,c in pairs)
  - A = 1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
  - idx = [1 2 5]
  - B= 1 2 5
6 7 10
21 22 25

Number of nodes

- Normally this is just the size of the network measured by the number of rows in the adjacency matrix: *size(A,1)* in Matlab
  - function numnodes=numnodes(A)
  - %finds number of nodes in A including isolates
  - numnodes=size(A,1);
- But if there are isolated nodes, it's useful to count only the non-isolated ones
- So numnonisonodes works differently
  - function nodes = numnonisonodes(A)
  - % counts non-isolated nodes in a matrix
  - A=unitize(A+A');
  - nodes=min(length(find(sum(A')~=0)),length(find(sum(A)~=0)));

Number of Links or Edges

- If A is symmetric (the network is undirected) then links are called edges and the number of edges is the number of entries in the adjacency matrix/2
- Max = n(n-1)/2; min = n-1
- If A is asymmetric (the network is directed or mixed) then directed links are called arcs and the number of edges+number of arcs = the same thing as above
- So numedges symmetrizes the matrix before calculating

Numedges

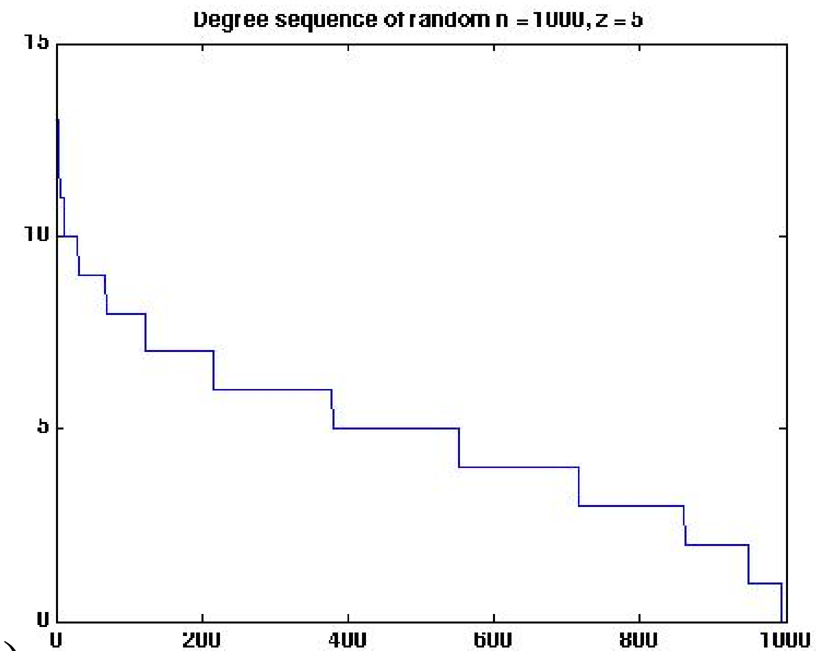
```
function numedges = numedges(A)
%counts edges in matrix A
%works when not all nodes have edges
AT=A+A';
numedges=sum(sum(AT~=0))/2;
```


Symmetry

- If A = A' (transpose) then the network is symmetric (undirected)
- In Matlab just ask *isequal(A,A')*
- The answer = 1 if yes, 0 if no
- To find non-symmetric entries, use
  - [i,j]=find(A~=A')

Degree Sequence

- kvec(A) finds the degrees of the nodes
  - function kvec = kvec(A)
  - %returns the k vector of a symmetric matrix
  - kvec=sum(A);
- Directed graphs have k_{in} and k_{out}
  - kin=sum(A)
  - kout=sum(A’)
- Routine **pds** plots degree seq vs percent of nodes
- Plot to right made with
  - plot(kvec(A))
  - after sorting kvec(A) in descending order using
  - kvsorted=sort(kvec,’descend’)



Average Nodal Degree

- Variously denoted z or $<k>$ (“k-hat”)
- Standard khat:

```
function khat = khat(A)
%function to find average nodal degree of a symmetric matrix
%includes isolated nodes, that is, nodes with k = 0
%to find <k> excluding isolates, use khatnoniso
numedges2=numedges(A)*2;
nodes=size(A,1);
khat=numedges2/nodes;
```

- khatnoniso:

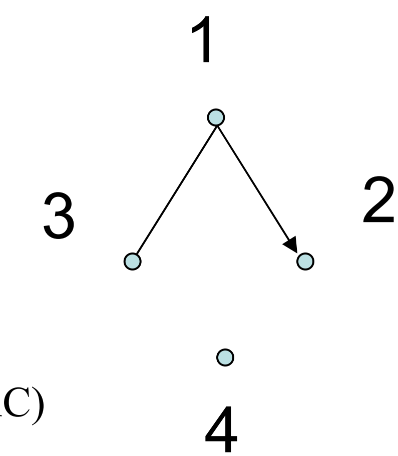
```
function khatnoniso = khat(A)
%function to find average nodal degree of a symmetric matrix
%does not include isolated nodes, that is, nodes with k = 0
%to find <k> including isolates, use khat
numedges2=numedges(A)*2;
nodes=numnonisonodes(A);
khatnoniso=numedges2/nodes;
```

Average Hop Distance

- Simplified average distance between two randomly selected nodes, assuming all node-node distances = 1
- Found by taking powers of A
- A^2 contains (i,j) entries = 1 where node *i* links to node *j* via a 2-hop path; A^3 contains (i,j) entries = 1 where node *i* links to node *j* via a 3-hop path, etc.
- If $A^{power}(i,j)$ flips from 0 to 1 for the first time when $power=k$ then there is a path of length k from *i* to *j*.
- When all possible paths have been found, the network's diameter $d = k$.   distmat.m works this way.
- Can probably be generalized to non-unity distances if they obey the triangle inequality
- Otherwise actual shortest paths must be found, computationally intensive

Hop Distance Example Using distmat.m

- >> AC=[0 1 1 0;0 0 0 0;1 0 0 0;0 0 0 0]
- AC =
- 0 1 1 0
- 0 0 0 0
- 1 0 0 0
- 0 0 0 0
- >> [distance_matrix, avg_dist, diameter]=distmat(AC)
- distance_matrix =
- 0 1 1 0
- 0 0 0 0
- 1 2 0 0
- 0 0 0 0
- avg_dist =
- 1.2500 (5 total length of 4 paths)
- diameter =
- 2



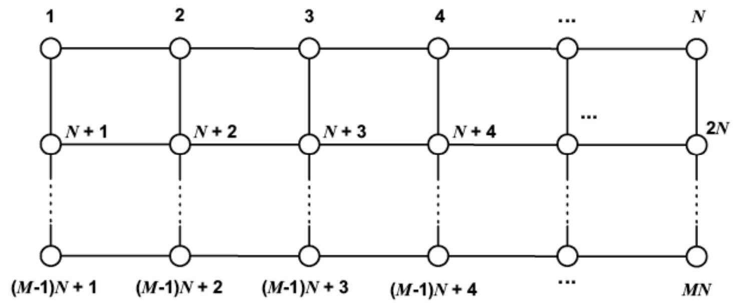
row:col = from:to

Average Hop Distance for Regular Network Structures

- Grids, stars, stars with concentric circles, etc
- Of interest to analysts of transportation networks and cellphone networks
  - Transport: nodes are transfer stations
  - Cellphones: nodes are transmission towers
- Lots of work done by Leonard Miller, NIST
- Evaluates the average hop distance formula from a few slides back in closed form for several such networks
- http://w3.antd.nist.gov/wctg/netanal/netanal_netmodels.html

CONNECTIVITY PROPERTIES OF MESH AND RING/MESH NETWORKS

L. E. Miller 2 April 2001



the average hop distance for a mesh network is

$$\overline{m} = \frac{N+M}{3}$$

$$Connectivity = \frac{M \times 2(N-1) + 2(M-1) \times N}{NM(NM-1)} = \frac{2[M(2N-1)-N]}{NM(NM-1)}$$

$$Connectivity = \frac{actual\_links}{\max\_poss\_links} = \frac{m}{n(n-1)/2} = \frac{<k>}{n-1} = clust_{random}$$

$$numnodes = MN$$

$$numedges = M(N-1) + N(M-1)$$

$$<k> = 2\frac{M(N-1)+N(M-1)}{MN} = 4 - \frac{2}{M} - \frac{2}{N}$$

Adjacency Matrix for Grids

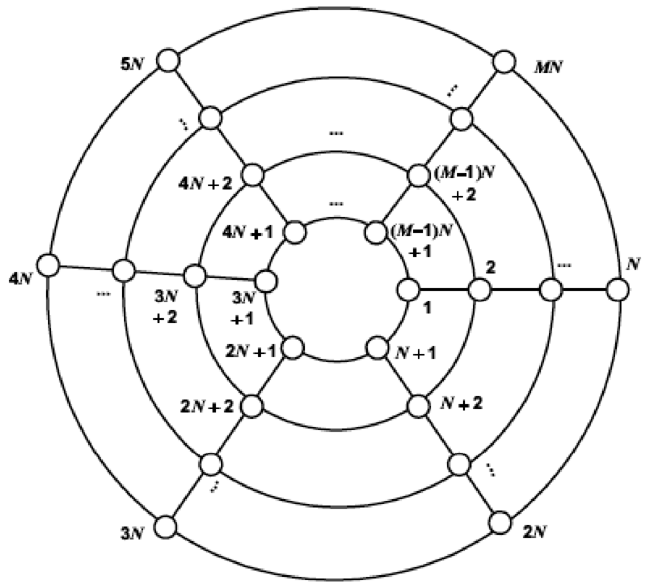
The adjacency (one-hop connectivity) matrix for such a network, in which a 1 entry at (i, j) indicates a connection from node i to node j and a 0 entry at (i, j) indicates no connection from node i to node j, is an $NM \times NM$ matrix with the form given by

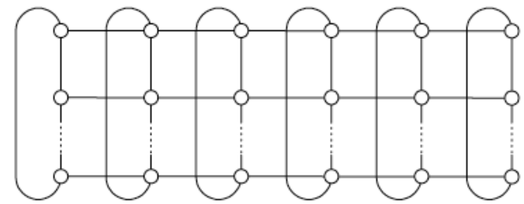
Should read A_{NMxMN}

$$A_{M\times N} = \begin{bmatrix} A_N & I_N & 0 & \cdots & 0 \\ I_N & A_N & I_N & \cdots & 0 \\ 0 & I_N & A_N & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_N \end{bmatrix} \tag{1}$$

where I_N is the $N \times N$ identity matrix and A_N is the $N \times N$ adjacency matrix for a single row of N nodes connected in tandem. The structure of A_N is easily determined to be a matrix of 0s, except for $N-1$ 1s on the first upper diagonal and $N-1$ 1s on the first lower diagonal, for example,

$$A_6 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$





the average hop distance for a ring-mesh network is

$$\overline{m} = \frac{1}{12(NM-1)} \times \begin{cases} 4M(N^2-1) + 3N(M^2-1), & M \text{ odd} \\ 4M(N^2-1) + 3NM^2, & M \text{ even} \end{cases}$$

$$Connectivity = \frac{M \times 2(N-1) + 2M \times N}{NM(NM-1)} = \frac{2(2N-1)}{N(NM-1)}$$

$$numnodes = MN$$

$$numedges = M(N-1) + N(M-1) + N$$

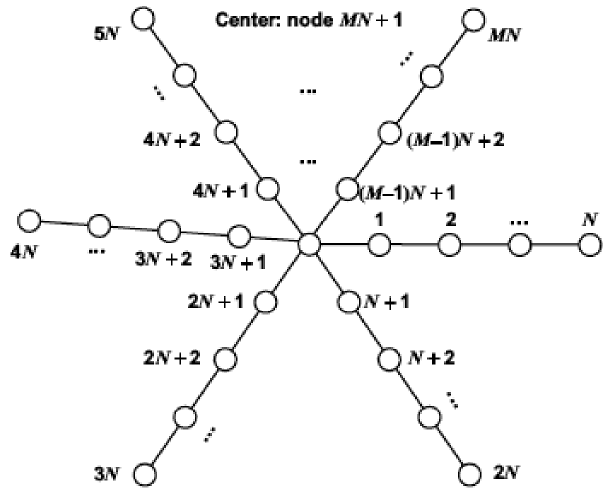
$$< k >= 2\frac{M(N-1) + N(M-1) + N}{MN} = 2\frac{2MN - M}{MN} = 4 - \frac{2}{N}$$

CONNECTIVITY PROPERTIES OF STAR AND STAR/MESH NETWORKS

L. E. Miller 2 April 2001

1. STAR NETWORK

A star network connected by bidirectional links can be modeled as M "rays" of N nodes plus a center node, as illustrated in the following figure:



$$numnodes = MN + 1$$

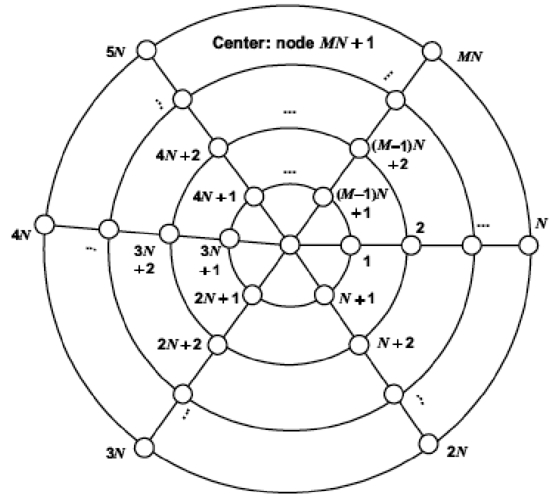
$$numedges = MN$$

$$< k >= 2\frac{MN}{MN+1}$$

the average hop distance for a star network is

$$\overline{m} = \frac{(N+1)(2MN+M-N+1)}{3(MN+1)}$$

$$Connectivity = \frac{M \times 2(N-1) + 2M \times 1}{NM(NM+1)} = \frac{2}{NM+1}$$



the average hop distance for a star-mesh network is

$$\overline{m} = \frac{MN^2 + 6MN - M - 9N + 3}{3(MN + 1)}$$

$$Connectivity = \frac{M \times 2(N-1) + 2M \times N + 2M \times 1}{NM(NM+1)} = \frac{4}{NM+1}$$

$$numnodes = MN + 1$$

$$numedges = staredges + ringedges = MN + MN = 2MN$$

$$< k >= 4\frac{MN}{MN+1}$$

Clustering Coefficient

- Asks if two neighbors of a node are neighbors
- This can be answered for every node and then the average taken as the network's overall C
- Normalize by k(k-1)/2 = the max number of triangles for a node whose degree = k
- Calculated by finding A^3 and looking at the diagonal, whose entries count the number of paths of length 3 that return to the start node
- A^n (i,j)= number of paths from i to j of length n
- Paths of length 3 that return to their start are triangles
- If the network is undirected then there will be 2 for each triangle since it counts both directions

Clustering Coeff Code

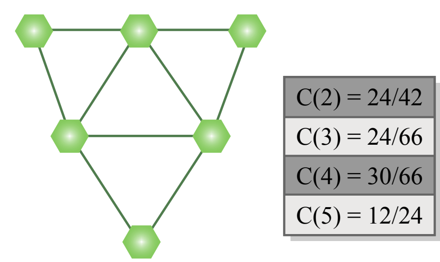
```
function [clust,cc] = clust(A)
%  the clustering coefficient modified from code by Ed Schneiderman, Johns Hopkins U
A=sortbyk(A); % sorts A so first node has biggest k, etc
% we will calculate the average clustering coefficient ONLY for those
% vertices where the number of neighbors is >1. We can calculate it
% for all vertices as well, by defining that if the vertex has zero
% neighbors, its clustering coeff is zero, and if it has one neighbor,
% its clustering coeff is one.
s=size(A,1);
T=A*A*A;
triangles=0.5*diag(T);
% a vector of s elements, each element being the # of triangles around that vertex
% note that we will use the *undirected* network here

k=sum(A,2); % a vector with the number of edges for each vertex

c_avg=0;
cc=zeros(s,1);
N_k_morethanone=0;

for i=1:s
if k(i)>1
    cc(i)=2*triangles(i)/(k(i)*(k(i)-1)); % each node's clustering coeff
c_avg=c_avg+cc(i); N_k_morethanone=N_k_morethanone+1;
end
end
c_avg=c_avg/N_k_morethanone;
clust=c_avg;
```

Two Ways to Define Clustering Coeff

- Newman Equation (3)
- Newman Equation (5)
- These are called clustEq3 and clustEq5

Generalized Clustering Coefficient

- Asks for loops of any length, not just 3
- This is a lot harder to do
- Paper by Huang discusses this



An example illustrating generalized clustering coefficients.

Image by MIT OpenCourseWare.

Link Prediction Based on Graph Topology: The Predictive Value of the Generalized Clustering Coefficient

Zan Huang
Department of Supply Chain and Information Systems
Pennsylvania State University
419 Business Building
University Park, PA, 16802
zanhuang@psu.edu

LinkKDD'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

Connectedness

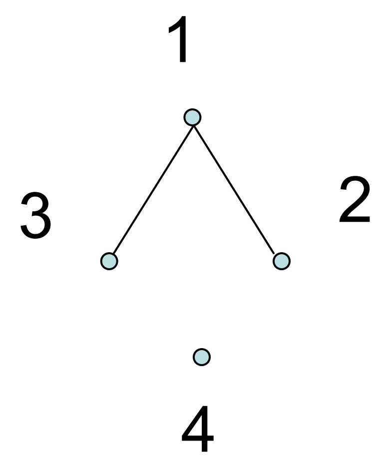
```
function yn = isconnected(g)
% isconnected(g) -- determine if g is a connected graph
% written by Ed Schneiderman, Johns Hopkins U, comments added by Whitney
% Gives the right answer only if g is undirected
% Works by asking if node 1 can be reached from every node
% If we set x(j)=1 before starting the while loop, it asks if node j can be
% reached from every node
% By testing each node this way, we can see if a directed graph is
% connected.  This is how isconnectedasym works.
% modified to work when not all nodes have edges 9-14-06
% g = g+0.;
n = length(g);
x = zeros(n,1);
x(1)=1;% by definition, node 1 can be reached from node 1
while(1)
    y = x; % remembers previous x
    x = g*x + x;% g*x(j) = 1 when node j has been reached from node 1.
                % Adding in x just remembers previously reached nodes.
    x = x>0;% unitize x
    if (x==y) ;% no new nodes were reached
        break
    end
end
if (sum(x)<numnodes(g))
    yn = 0;% we did not reach every node
else
    yn = 1;% we reached every node, so it's connected
end
```

Example of Connectedness Calculation

AC1 =

$$\begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

length(AC1)
=
4



x =

$$\begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix}$$

Start at node 1

AC1*x =	x=AC1*x+x =	x=AC1*x+x =	unitize(x) =
0	1	3	1
1	1	2	1
1	1	2	1
0	0	0	0

So we're done. Length(x) < 4 so it's disconnected

We reach nodes 2 and 3 right away and get no further

Network Components

- If a network is disconnected, it consists of isolated clusters and isolated nodes
- Calling sequence on next page for code written by Mo-Han Hsieh finds all these clusters

componentCount calling sequence only

```
% [componentCount] is used to generate the component partition of a matrix.
% Its input is the adjacency matrix, A.
% Its outputs are partition, componentList, mainNum, and singletonNum.
% /partition/ tells which component each node is in.
% /componentList/ is a list of components and the number of nodes in each
% component.  Its format is: [component ID, number of nodes in it].
% /mainNum/ is the number of components which have at least two members, and
% /singletonNum/ is the number of singletons.

function [partition,componentList,mainNum,singletonNum]=componentCount(A)
```


componentCount Example

AC =

0	1	1	0
0	0	0	0
1	0	0	0
0	0	0	0

```
[aa,bb,cc,dd]=componentCount(AC)
aa =
     1
     1
     1
     2
```

Nodes 1 - 3 are in comp 1
Node 4 is in comp 2

```
bb =
     1     3
     2     1
```

Comp 1 has 3 nodes
Comp 2 has 1 node

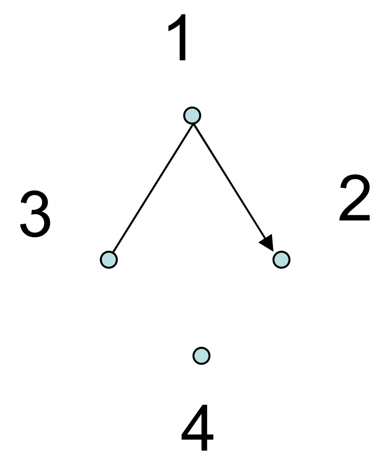
```
cc =
     1
```

comps with >1 node = 1

```
dd =
     1
```

singletons = 1

size(bb,1) = number of components = 2



List of Some Useful Routines

- Adjbuilde, adjbuildn for making adjacency matrices from edge lists and node lists
- Khat for finding average nodal degree
- Numnodes for counting nodes
- Numedges for counting edges
- Is connected to see if the network is whole or separated into isolated network components
- Distmat for finding shortest paths if node pairs are separated by one hop (don’t use ave_path_length or shortest_path)
- Randmatrix for building a Poisson random network (don’t use random_graph unless you use the default that gives a Poisson network)
- ClustEq3 or clustEq5 for finding clustering coefficient
- Componentcount for finding separate components

More (later)

- Centrality
  - Node centrality
  - Edge centrality
  - Betweenness centrality
- Calculations
- Degree correlation
    - Joint degree distribution
    - K-nearest neighbors
    - Pearson degree correlation
- Rich club metric
- Degree-preserving rewiring
- Generating a graph that has a specified degree sequence
- Finding Pearson degree correlation
- Finding communities

ESD.342 Network Representations of Complex Engineering Systems
Spring 2010