Social Network Analysis

Chen Avin
Communication Systems Engineering

Unit 5
Centrality

(Based on Networks, Crowds, and Markets: Reasoning about a Highly Connected World. By David Easley and Jon Kleinberg. Chapter 3,14 and Networks: An Introduction. By M.E.J Newman. Chapter 7)

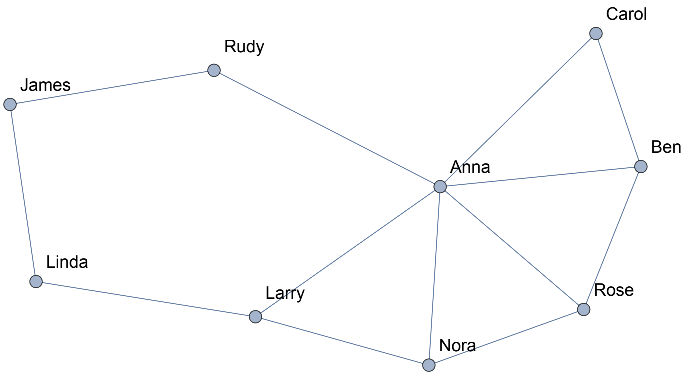
Node / Edge Centrality

- What is “Centrality” ???
- Degree Centrality
- Eigenvectors Centrality
- PageRank Centrality
- (Hubs and Authorities)
- Closeness Centrality
- Betweenness Centrality

Degree Centrality

In[1]:=
```
g = Graph[{"Rose", "Anna", "James", "Carol", "Nora"},
  EdgeList[ExampleData[{"NetworkGraph", "Friendship"}]], VertexLabels → "Name"]
```

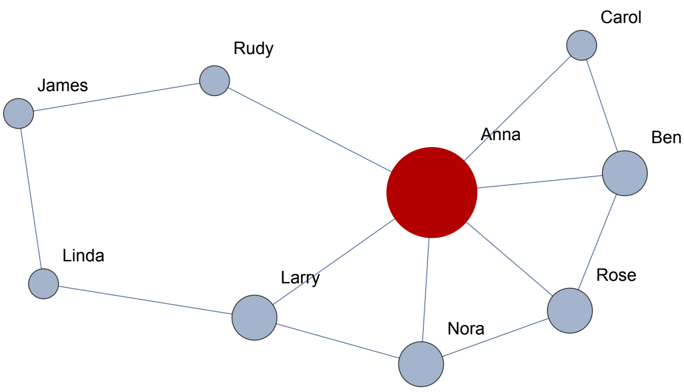
Out[1]=



- $\tilde{x}$ our centrality vector
- x_i = Degree(i)
- Most basic measure
- Social Network - Many friends → More influential, access to information, etc.
- Citation Network
  - Many Citation → Important Paper
  - Cite many papers → Important review
- Web
  - Many out going links → An important Directory (??)
  - Many in coming links → Important web-page

In[2]:=
```
HighlightGraph[g, {"Anna"},
 VertexSize → Thread[VertexList[g] → VertexDegree[g] / VertexCount[g]]]
```

Out[2]=



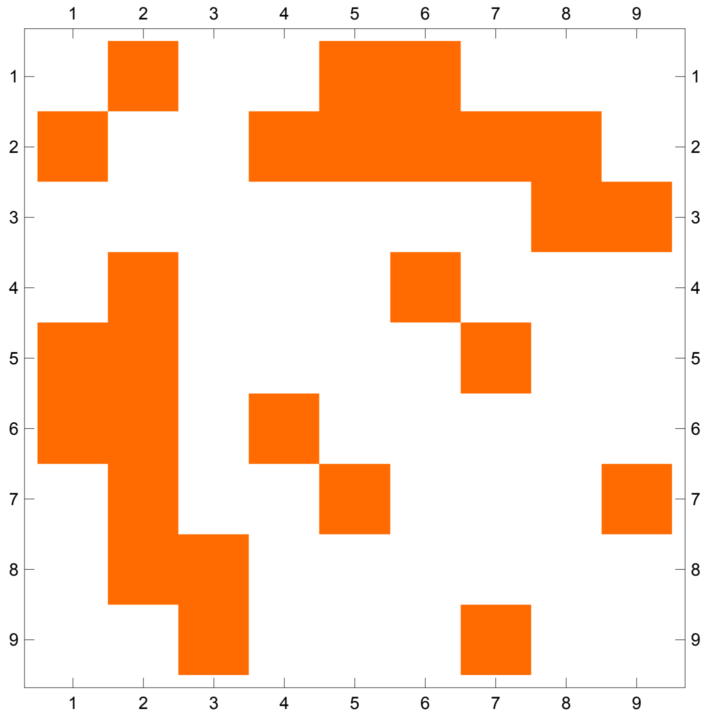
In[3]:=
```
Thread[VertexList[g] → VertexDegree[g]]
```

Out[3]=
```
{Rose → 3, Anna → 6, James → 2, Carol → 2,
 Nora → 3, Ben → 3, Larry → 3, Rudy → 2, Linda → 2}
```

A more general view point

In[4]:=
```
A = AdjacencyMatrix[g];
```

In[5]:=
```
MatrixPlot[A]
```

Out[5]=


We Can view the degree as

$$x_i = \sum_j A_{ij}$$

Or in Matrix notation

$$\tilde{x} = A\,\tilde{\mathbf{1}}$$

In[6]:=
```
Ones = Transpose[{ConstantArray[1, VertexCount[g]]}];
```

In[7]:=
```
A.Ones
```

Out[7]=
```
{{3}, {6}, {2}, {2}, {3}, {3}, {3}, {2}, {2}}
```

In[8]:=
```
Transpose[Ones].A
```

Out[8]=
```
{{3, 6, 2, 2, 3, 3, 3, 2, 2}}
```

Eigenvectors Centrality

- The basic idea: add “centrality points” to any neighbor a vertex has
- The credit of each neighbor should be proportional to it importance
- Let’s start with a guess x_i= 1 and then

$$x'_i = \sum_j A_{ij} x_j$$

Or in Matrix notation

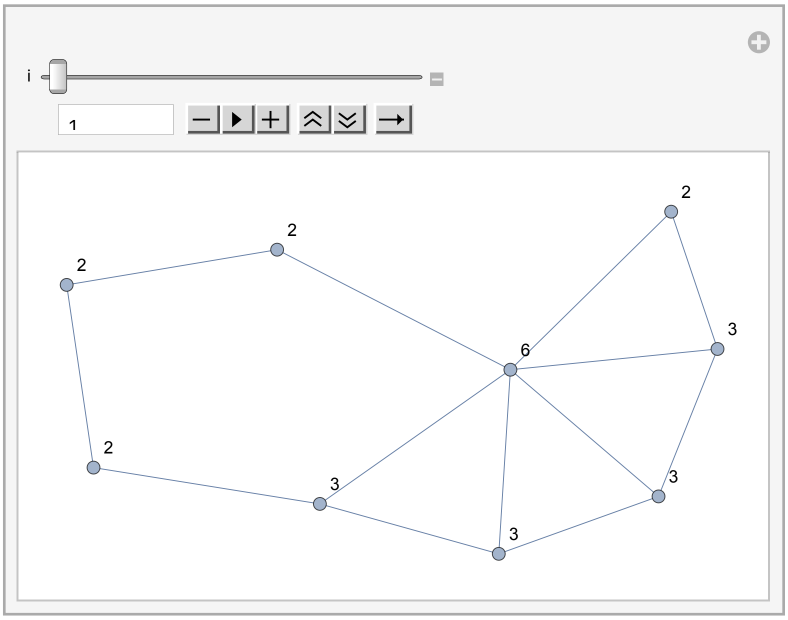
$$\tilde{x}' = A \tilde{x}$$

in time perspective

$$\tilde{x}_{t+1} = A \tilde{x}_t$$

In[9]:=
```
Manipulate[Graph[g, VertexLabels →
   Thread[VertexList[g] → Flatten[MatrixPower[A, i].Ones]]], {{i, 1}, 1, 100, 1}]
```

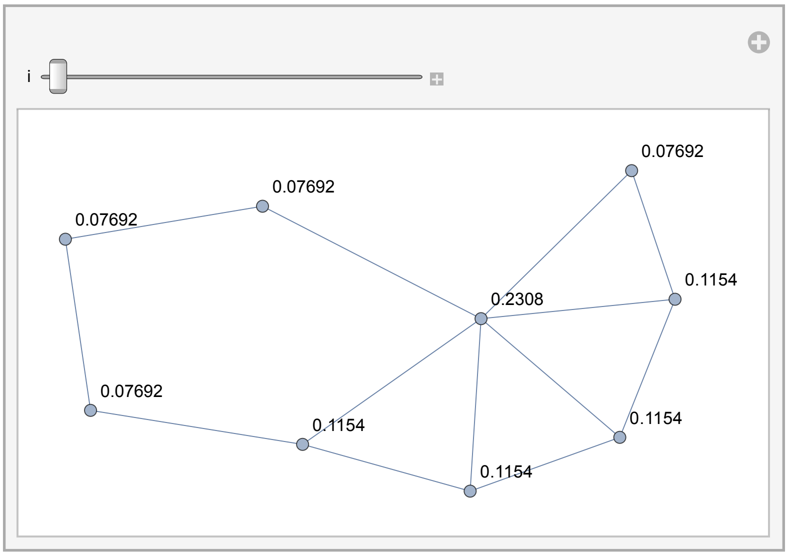
Out[9]=



What happen if we normalize $\tilde{x}_t$to 1? How?

In[12]:=
```
Manipulate[Graph[g,
  VertexLabels → Thread[VertexList[g] → N[Flatten[MatrixPower[A, i].Ones] /
      Total[Flatten[MatrixPower[A, i].Ones]], 4]]], {{i, 1}, 1, 100, 1}]
```

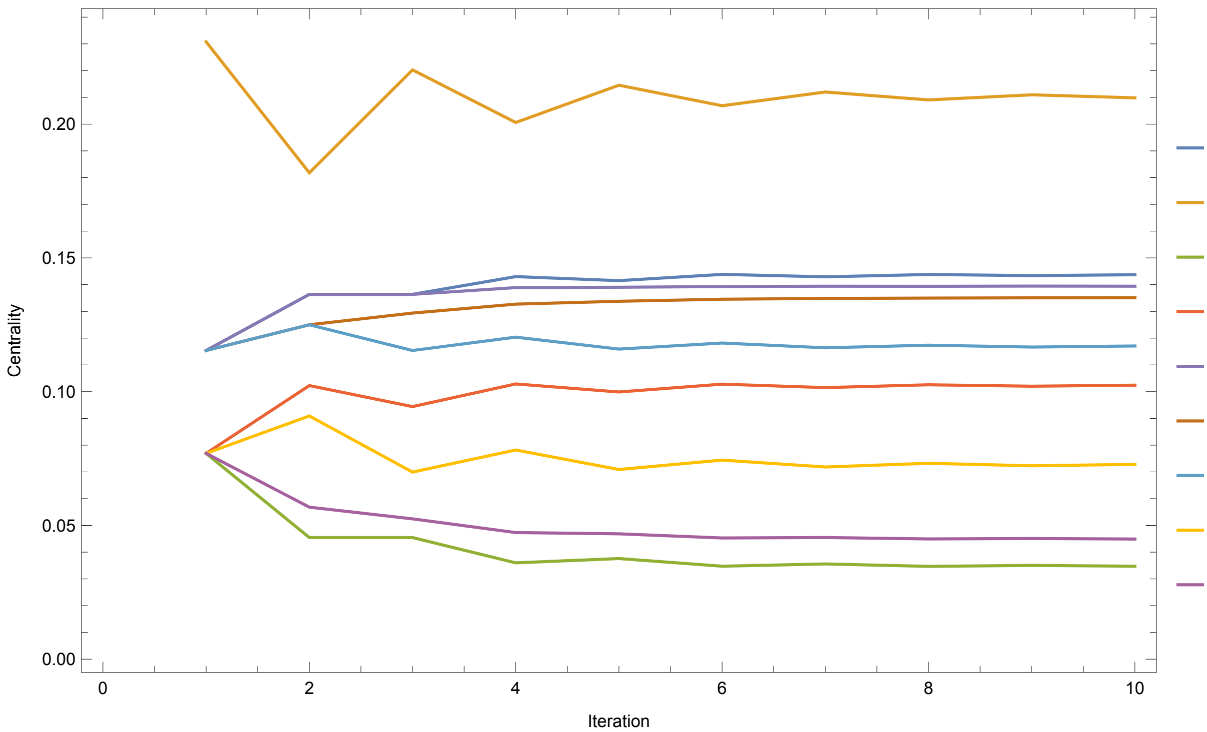
Out[12]=



In[13]:=
```
t = Table[Flatten[MatrixPower[A, i].Ones] /
    Total[Flatten[MatrixPower[A, i].Ones]], {i, 1, 10}];
```

In[21]:=
```
ListLinePlot[Transpose[t], Frame → True,
 FrameLabel → {"Iteration", "Centrality"},
 ImageSize → {600, 400}, PlotLegends → Automatic]
```

Out[21]=



- Let $x^{(t)}$ be the estimate after time t, and we start with $x^{(0)}$ then

$$\tilde{x}^{(1)} = A\,\tilde{x}^{(0)}$$

In[22]:=
```
A.Ones
```
Out[22]=
```
{{3}, {6}, {2}, {2}, {3}, {3}, {3}, {2}, {2}}
```

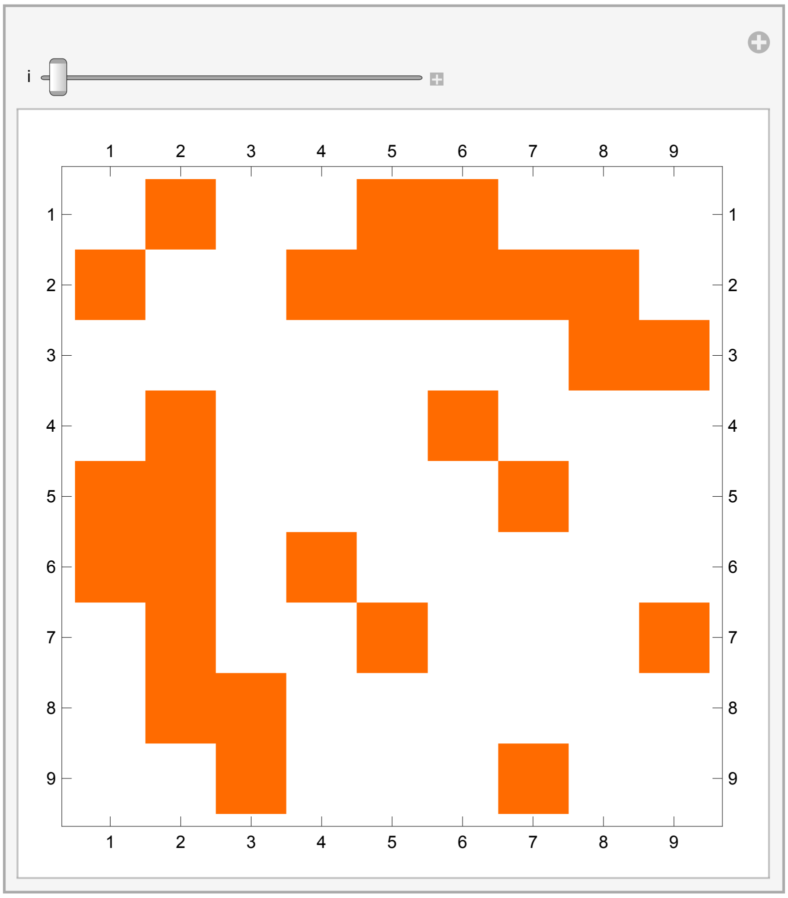
$$\tilde{x}^{(t)} = A\,\tilde{x}^{(t-1)} = A^t\,\tilde{x}^{(0)}$$

In[23]:=
```
A.A.Ones
```
Out[23]=
```
{{12}, {16}, {4}, {9}, {12}, {11}, {11}, {8}, {5}}
```

In[24]:=
```
A.A.A.Ones
```
Out[24]=
```
{{39}, {63}, {13}, {27}, {39}, {37}, {33}, {20}, {15}}
```

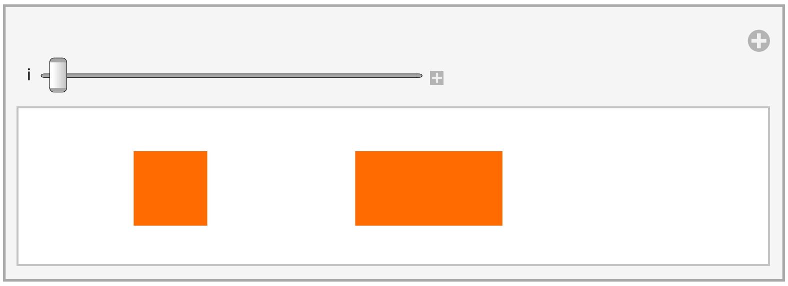
In[25]:=
```
MatrixPower[A, 4].Ones
```
Out[25]=
```
{{139}, {195}, {35}, {100}, {135}, {129}, {117}, {76}, {46}}
```

In[33]:=
```
Manipulate[MatrixPlot[MatrixPower[A, i]], {{i, 1}, 1, 100, 1}]
```
Out[33]=

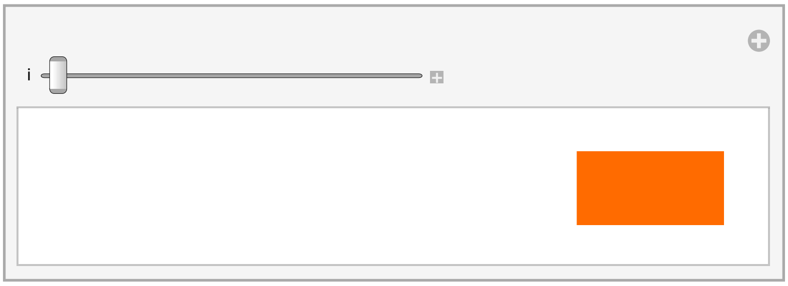


```
In[28]:= MatrixPlot[{EigenvectorCentrality[g]}, Frame → False]
Out[28]=
```

```
In[31]:= Manipulate[MatrixPlot[{MatrixPower[A, i][[1]]}, Frame → False], {{i, 1}, 1, 100, 1}]
Out[31]=
```




```
In[34]:= Manipulate[MatrixPlot[{MatrixPower[A, i][[3]]}, Frame → False], {{i, 1}, 1, 100, 1}]
Out[34]=
```



Eigenvectors Connection

- We can write $x^{(0)}$ as *a* linear combination of the eigenvectors $\boldsymbol{v}_i$ of A.

$$\tilde{x}^{(0)} = \sum_i c_i \, \tilde{v}_i$$

- Recall: Done by solving $V^T \tilde{c} = \bar{1}$. (since $x^{(0)} = \tilde{1}$)
- Where V is the matrix of Eigenvectors

In[35]:=
```
V = N[Eigenvectors[A]];
V // TableForm
```

Out[36]//TableForm=

3.19599	4.6799	0.774716	2.27682	3.1033	3.00713	2.60
-2.41535	4.98075	0.857134	-2.32784	0.182977	-0.113618	-2.9
-0.742713	-0.534434	-0.978576	-0.116964	1.16093	0.751434	-0.8
-0.386021	-0.105355	0.975655	-0.433662	0.0756076	-0.582877	0.61
0.942692	0.450748	-0.855955	0.597631	-0.513005	-1.33925	-0.6
2.67263	-1.54293	-9.43428	-8.3173	11.2864	-6.99877	10.4
0.507008	1.38205	-3.34943	-4.47344	-4.63225	2.77864	2.41
-0.562962	-0.212135	-0.186447	0.99193	-0.579485	0.581812	0.55
2.	-1.	1.	-1.	0.	1.	-1.

In[37]:=
```
Sol = LinearSolve[N[Transpose[V]], Ones]
Co = Flatten[Sol];
```

Out[37]=
```
{{0.332406}, {-0.0658231}, {0.0777478}, {0.512976},
 {-0.0131925}, {-0.020328}, {-0.0304831}, {0.115788}, {0.0909091}}
```

In[39]:=

$$\sum_{i=1}^{9} Co_{[[i]]} \, V_{[[i]]}$$

Out[39]=
```
{1., 1., 1., 1., 1., 1., 1., 1., 1.}
```

- Recall that for a eigen value λ_i (why?)

$$A \, \tilde{v}_i = \lambda_i \, \tilde{v}_i \quad \text{so} \quad A^t \, \tilde{v}_i = \lambda_i{}^t \, \tilde{v}_i$$

- Now we can write $x^{(t)}$

$$\tilde{x}^{(t)} =$$

$$A^t \, \tilde{x}^{(0)} = A^t \sum_i c_i \, \tilde{v}_i = \sum_i c_i \, A^t \, \tilde{v}_i = \sum_i c_i \, \lambda_i^t \, \tilde{v}_i$$

- No we can multiply and divide by λ_1^t to get

$$\tilde{x}^{(t)} = \lambda_1^t \sum_i c_i \left(\frac{\lambda_i}{\lambda_1}\right)^t \tilde{v}_i$$

- Where λ_1 is the **largest** eigenvalue of A

In[52]:=
```
λ = N[Eigenvalues[A]];
```

In[41]:=
```
Sort[λ, Greater]
```

Out[41]=
```
{3.37621, 1.58702, 1.02698, 0.372685, 0., -0.930088, -1.48671, -1.85527, -2.09084}
```

In[42]:=
```
Ordering[λ, All, Greater]
```

Out[42]=
```
{1, 4, 6, 8, 9, 7, 5, 3, 2}
```

- Let’s check

In[43]:=
```
A. V[[1]]
```

Out[43]=
```
{10.7903, 15.8003, 2.6156, 7.68703, 10.4774, 10.1527, 8.7832, 5.45462, 3.37621}
```

In[44]:=
```
λ[[1]] V[[1]]
```

Out[44]=
```
{10.7903, 15.8003, 2.6156, 7.68703, 10.4774, 10.1527, 8.7832, 5.45462, 3.37621}
```

In[45]:=
```
MatrixPower[A, 3].V[[1]]
```

Out[45]=
```
{122.997, 180.105, 29.8147, 87.623, 119.43, 115.729, 100.118, 62.1761, 38.4848}
```

In[46]:=
```
λ[[1]]^3 V[[1]]
```

Out[46]=
```
{122.997, 180.105, 29.8147, 87.623, 119.43, 115.729, 100.118, 62.1761, 38.4848}
```

In[47]:=
```
MatrixPower[A, 3].Ones
```

Out[47]=
```
{{39}, {63}, {13}, {27}, {39}, {37}, {33}, {20}, {15}}
```

In[48]:=

$$\lambda_{[[1]]}{}^3 \sum_{i=1}^{9} Co_{[[i]]} \left(\frac{\lambda_{[[i]]}}{\lambda_{[[1]]}}\right)^3 V_{[[i]]}$$

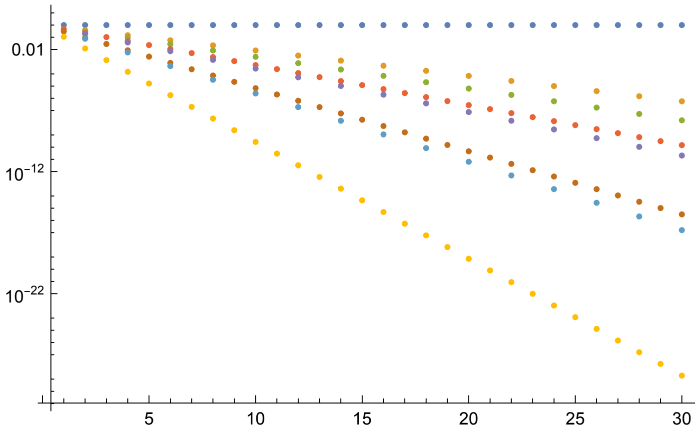
Out[48]=
```
{39., 63., 13., 27., 39., 37., 33., 20., 15.}
```

- But what happens to $\left(\frac{\lambda_{[[i]]}}{\lambda_{[[1]]}}\right)^t$ as $t \to \infty$?

In[54]:=
```
Transpose[Table[(λ / λ[[1]])^t, {t, 1, 30}]];
```

In[55]:= `ListLogPlot[Transpose[Table[(λ / λ[[1]])^t, {t, 1, 30}]], PlotRange → All]`

Out[55]=



In[56]:= `Manipulate[(λ / λ[[1]])^t, {{t, 1}, 1, 100, 1}]`

Out[56]=

t

{1., -0.619285, -0.549512, 0.47006,
-0.440348, 0.304181, -0.275483, 0.110386, 0.}

- So as t → ∞?

$$\tilde{x}^{(t)} =$$

$$\lambda_1^t \sum_i c_i \left(\frac{\lambda_i}{\lambda_1}\right)^t \tilde{v}_i = \lambda_1^t\, c_1\, \tilde{v}_1 + o(1) = \lambda_1^t\, c_1\, \tilde{v}_1$$

- Let verify the magic $\lambda_{[[1]]}{}^t \sum_{i=1}^{9} Co_{[[i]]} \left(\frac{\lambda_{[[i]]}}{\lambda_{[[1]]}}\right)^t V_{[[i]]} \to \lambda_{[[1]]}{}^t\, Co_{[[1]]}\, V_{[[1]]}$

In[60]:= $\text{Manipulate}\left[\text{TableForm}\left[\left\{\lambda_{[[1]]}{}^t \sum_{i=1}^{9} Co_{[[i]]} \left(\frac{\lambda_{[[i]]}}{\lambda_{[[1]]}}\right)^t V_{[[i]]} - \lambda_{[[1]]}{}^t\, Co_{[[1]]}\, V_{[[1]]}\right\}\right],\right.$
$\left.\{\{t, 1\}, 1, 100, 1\}\right]$

Out[60]=

t

-0.586768 0.747877 1.13056 -0.555214 -0.482743 -0.37482 0.080

In[59]:=
$$\text{Manipulate}\left[\text{TableForm}\left[\left\{\lambda_{[[1]]}{}^{t}\sum_{i=1}^{9}\text{Co}_{[[i]]}\left(\frac{\lambda_{[[i]]}}{\lambda_{[[1]]}}\right)^{t}V_{[[i]]},\ \lambda_{[[1]]}{}^{t}\,\text{Co}_{[[1]]}\,V_{[[1]]}\right\}\Big/\ \text{Total}\left[\lambda_{[[1]]}{}^{t}\sum_{i=1}^{9}\text{Co}_{[[i]]}\left(\frac{\lambda_{[[i]]}}{\lambda_{[[1]]}}\right)^{t}V_{[[i]]}\right]\right],\ \{\{t,1\},1,100,1\}\right]$$

Out[59]=

t

0.143608	0.210286	0.0348109	0.102306	0.139443	0.135122	0.11689
0.143608	0.210286	0.0348109	0.102306	0.139443	0.135122	0.11689

- So our centrality $\tilde{x}$

$$A\,\tilde{x} = \lambda_1\,\tilde{x}$$

- This is actually what we wanted

$$x_i = \frac{1}{\lambda_1}\sum_j A_{ij}\,x_j$$

- And the solution is $\tilde{x} = \tilde{v}_1$ (the first eigen vector, but there are infinite solutions...)

In[61]:=
```
V⟦1⟧
```
Out[61]=
```
{3.19599, 4.6799, 0.774716, 2.27682, 3.1033, 3.00713, 2.6015, 1.6156, 1.}
```

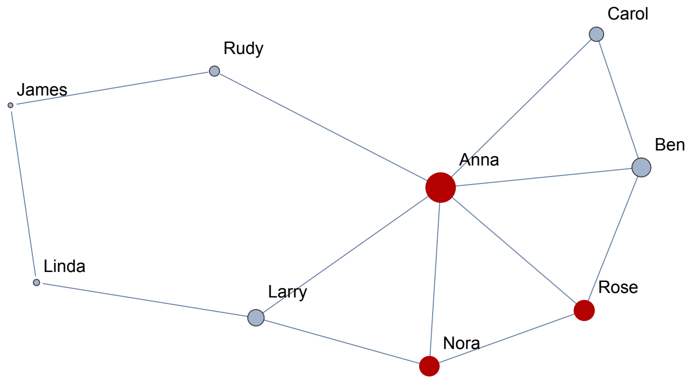
In[62]:=
```
V⟦1⟧ / Total[V⟦1⟧]
```
Out[62]=
```
{0.143608, 0.210286, 0.0348109, 0.102306,
 0.139443, 0.135122, 0.116895, 0.0725952, 0.0449338}
```

In[63]:=
```
EigenvectorCentrality[g]
```
Out[63]=
```
{0.143608, 0.210286, 0.0348109, 0.102306,
 0.139443, 0.135122, 0.116895, 0.0725952, 0.0449338}
```

- 3 nodes with highest Eigenvector Centrality

In[64]:=
```
HighlightGraph[g, VertexList[g][[Ordering[EigenvectorCentrality[g], 3, Greater]]],
 VertexSize → Thread[VertexList[g] → EigenvectorCentrality[g]]]
```

Out[64]=



- 5 nodes with highest Eigenvector Centrality - their values

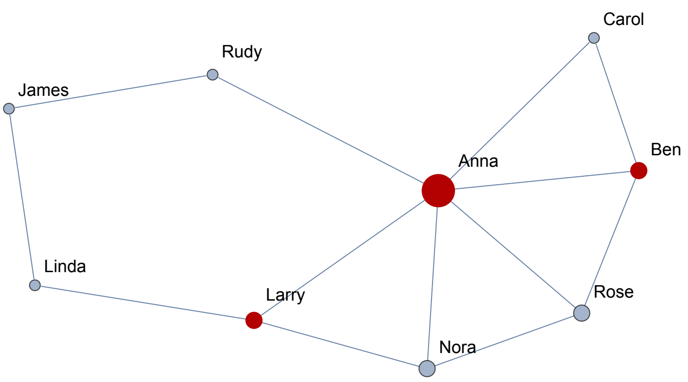
In[66]:=
```
Thread[VertexList[g][[Ordering[EigenvectorCentrality[g], 5, Greater]]] →
  EigenvectorCentrality[g][[Ordering[EigenvectorCentrality[g], 5, Greater]]]]
```

Out[66]=
```
{Anna → 0.210286, Rose → 0.143608,
 Nora → 0.139443, Ben → 0.135122, Larry → 0.116895}
```

- 3 nodes with highest Degree Centrality

In[67]:=
```
HighlightGraph[g, VertexList[g][[Ordering[DegreeCentrality[g], 3, Greater]]],
 VertexSize → Thread[VertexList[g] → VertexDegree[g] / Total[ VertexDegree[g]]]]
```

Out[67]=



- 5 nodes with highest Degree Centrality and normalized values (same degree same values)

In[68]:=
```
Thread[VertexList[g][[Ordering[DegreeCentrality[g], 5, Greater]]] →
  N[DegreeCentrality[g][[Ordering[DegreeCentrality[g], 5, Greater]]] /
    Total[ VertexDegree[g]]]]
```

Out[68]=
```
{Anna → 0.230769, Larry → 0.115385,
 Ben → 0.115385, Nora → 0.115385, Rose → 0.115385}
```

Some Problems

- Works well for undirected networks
- What about directed networks?
- In-degree 0 is problematic, will get centrality 0
- Can cause other nodes to have centrality 0
- So will work "well" only in strongly connected components
- Will not work in acyclic networks

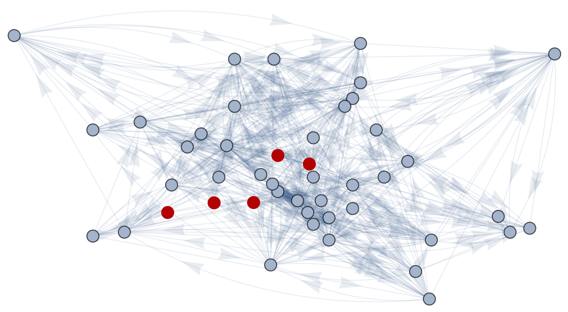
In[69]:=
```
EV = ExampleData[{"NetworkGraph", "EurovisionVotes"}];
```

In[70]:=
```
DirectedGraphQ[EV]
```

Out[70]=
```
True
```

In[71]:=
```
HighlightGraph[Graph[EV, EdgeShapeFunction → "Arrow"],
 VertexList[EV][[Ordering[EigenvectorCentrality[EV], 5]]]]
```

Out[71]=



In[72]:=
```
Sort[EigenvectorCentrality[EV]]
```

Out[72]=
```
{0., 0., 0., 0., 0., 0.00317425, 0.00607408, 0.00609656, 0.00677032, 0.00691068,
 0.00702961, 0.00729578, 0.0105628, 0.0106341, 0.0116941, 0.0122336,
 0.0123216, 0.0132339, 0.0142476, 0.0146546, 0.0151566, 0.0158825,
 0.0174053, 0.0201144, 0.0204298, 0.0207447, 0.0209845, 0.0260254,
 0.0265146, 0.0280835, 0.0307511, 0.0341079, 0.0348247, 0.0350353,
 0.0352269, 0.0363689, 0.0371292, 0.0372286, 0.0383986, 0.0411051,
 0.0416771, 0.0439427, 0.0440047, 0.0453036, 0.0483971, 0.0622233}
```

In[73]:=
```
VertexInDegree[EV][[Ordering[EigenvectorCentrality[EV], 5]]]
```

Out[73]=
```
{0, 0, 0, 0, 0}
```

PageRank Centrality

- The Google Story
- The basic idea is to normalize node contribution by their out-degree
- You can't give all the credit your have to everyone....

$$x_i = \alpha \sum_j A_{ji} \frac{x_j}{k_j^{out}} + \beta$$

- To solve a problem of $\frac{0}{0}$ set k_i^{out} to be at least 1
- Let D by the diagonal matrix with $D_{ii} = \max(k_i^{out}, 1)$
- In Matrix notation

$$\tilde{x} = \alpha\, A D^{-1}\, \tilde{x} + \beta\, \tilde{1}$$

- The solution for $\tilde{x}$ then is

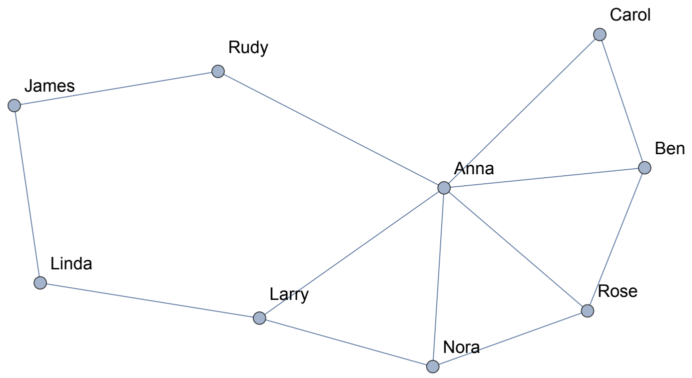
$$\tilde{x} = \beta\, \left(I - \alpha\, A D^{-1}\right)^{-1} \tilde{1}$$

- Since β only change the scale we can set β=1 and

$$\tilde{x} = \left(I - \alpha\, A D^{-1}\right)^{-1} \tilde{1} = D\, (D - \alpha\, A)^{-1}\, \tilde{1}$$

In[74]:=
```
α = .85
```
Out[74]=
```
0.85
```

In[80]:=
```
g
```
Out[80]=



In[75]:=
```
xx = Inverse[
    IdentityMatrix[9] - α A.Inverse[DiagonalMatrix[VertexOutDegree[g]]]].Ones
```
Out[75]=
```
{{6.57368}, {12.7542}, {5.40293}, {4.69521},
 {6.62989}, {6.66485}, {6.91943}, {5.10309}, {5.25675}}
```

In[76]:=
```
Flatten[xx] / Total[Flatten[xx]]
```
Out[76]=
```
{0.109561, 0.212569, 0.0900488, 0.0782536,
 0.110498, 0.111081, 0.115324, 0.0850514, 0.0876125}
```

In[77]:=
```
PageRankCentrality[g, .85]
```
Out[77]=
```
{0.109561, 0.212569, 0.0900488, 0.0782536,
 0.110498, 0.111081, 0.115324, 0.0850514, 0.0876125}
```

Let’s check

In[78]:=
```
α = .85
```
Out[78]=
```
0.85
```

- D^{-1} =

In[79]:=
```
Inverse[DiagonalMatrix[VertexOutDegree[g]]] // MatrixForm
```
Out[79]//MatrixForm=

$$\begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

- AD^{-1} =

In[81]:=
```
Inverse[DiagonalMatrix[VertexOutDegree[g]]].A // MatrixForm
```
Out[81]//MatrixForm=

$$\begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{6} & 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

- $\alpha AD^{-1} + (1-\alpha)\ (1/n)$ Ones[n, n] =

In[82]:=
```
RWP = α Inverse[DiagonalMatrix[VertexOutDegree[g]]].A +
   (1 - α) 1 / 9 ConstantArray[1, {9, 9}];
```

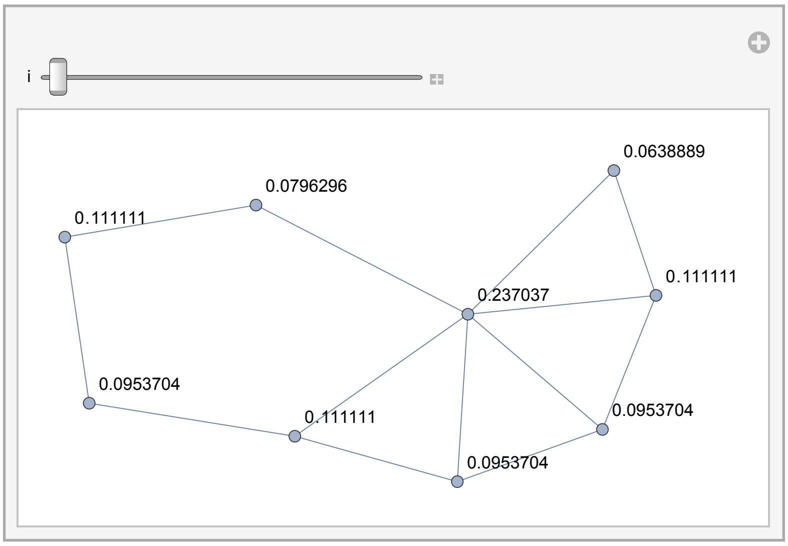
In[83]:=
```
RWP // MatrixForm
```

Out[83]//MatrixForm=

0.0166667	0.3	0.0166667	0.0166667	0.3	0.3	0.0166667	0.01666
0.158333	0.0166667	0.0166667	0.158333	0.158333	0.158333	0.158333	0.15833
0.0166667	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667	0.44166
0.0166667	0.441667	0.0166667	0.0166667	0.0166667	0.441667	0.0166667	0.01666
0.3	0.3	0.0166667	0.0166667	0.0166667	0.0166667	0.3	0.01666
0.3	0.3	0.0166667	0.3	0.0166667	0.0166667	0.0166667	0.01666
0.0166667	0.3	0.0166667	0.0166667	0.3	0.0166667	0.0166667	0.01666
0.0166667	0.441667	0.441667	0.0166667	0.0166667	0.0166667	0.0166667	0.01666
0.0166667	0.0166667	0.441667	0.0166667	0.0166667	0.0166667	0.441667	0.01666

In[89]:=
```
Manipulate[Graph[g, VertexLabels →
   Thread[VertexList[g] → N[Flatten[Transpose[MatrixPower[RWP, i]].(Ones / 9)],
      4]]], {{i, 1}, 1, 100, 1}]
```

Out[89]=



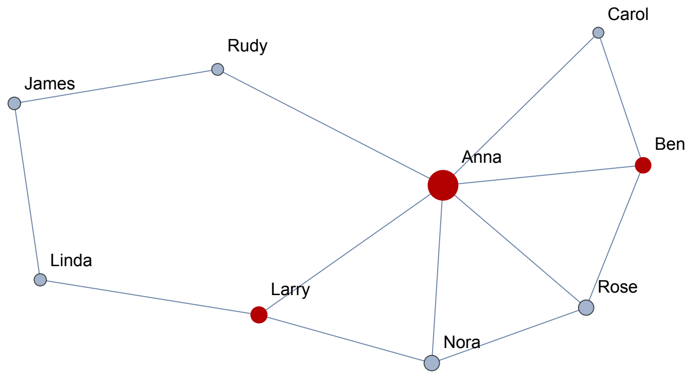

```
PageRankCentrality[g, .85]
```

{0.109561, 0.212569, 0.0900488, 0.0782536,
 0.110498, 0.111081, 0.115324, 0.0850514, 0.0876125}

- 3 nodes with highest Page Rank Centrality

In[90]:=
```
HighlightGraph[g,
 VertexList[g][[Ordering[PageRankCentrality[g, .85], 3, Greater]]],
 VertexSize → Thread[VertexList[g] → PageRankCentrality[g, .85]]]
```

Out[90]=



- 5 nodes with highest Page Rank Centrality and their values

In[92]:=
```
Manipulate[
 Thread[VertexList[g][[Ordering[PageRankCentrality[g, α], 5, Greater]]] →
   N[PageRankCentrality[g, α][[Ordering[PageRankCentrality[g, α], 5, Greater]]]]],
 {{α, 0.85}, 0, 1}]
```

Out[92]=

α

```
{Anna → 0.212569, Larry → 0.115324,
 Ben → 0.111081, Nora → 0.110498, Rose → 0.109561}
```

- Recall Eigenvalue Centrality

```
Thread[VertexList[g][[Ordering[EigenvectorCentrality[g], 5, Greater]]] →
  EigenvectorCentrality[g][[Ordering[EigenvectorCentrality[g], 5, Greater]]]]
```

```
{Anna → 0.210286, Rose → 0.143608,
 Nora → 0.139443, Ben → 0.135122, Larry → 0.116895}
```

The role of α

In[95]:=
```
Manipulate[α Inverse[DiagonalMatrix[VertexOutDegree[g]]].A +
   (1 - α) 1 / 9 ConstantArray[1, {9, 9}] // MatrixForm, {{α, 0.85}, 0, 1}]
```

Out[95]=

α

0.0166667	0.3	0.0166667	0.0166667	0.3	0.3	0.0166667	0.016
0.158333	0.0166667	0.0166667	0.158333	0.158333	0.158333	0.158333	0.15
0.0166667	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667	0.0166667	0.44
0.0166667	0.441667	0.0166667	0.0166667	0.0166667	0.441667	0.0166667	0.016
0.3	0.3	0.0166667	0.0166667	0.0166667	0.0166667	0.3	0.016
0.3	0.3	0.0166667	0.3	0.0166667	0.0166667	0.0166667	0.016
0.0166667	0.3	0.0166667	0.0166667	0.3	0.0166667	0.0166667	0.016
0.0166667	0.441667	0.441667	0.0166667	0.0166667	0.0166667	0.0166667	0.016
0.0166667	0.0166667	0.441667	0.0166667	0.0166667	0.0166667	0.441667	0.016

In[96]:=
```
Manipulate[PageRankCentrality[g, α], {α, 0, 1}]
```

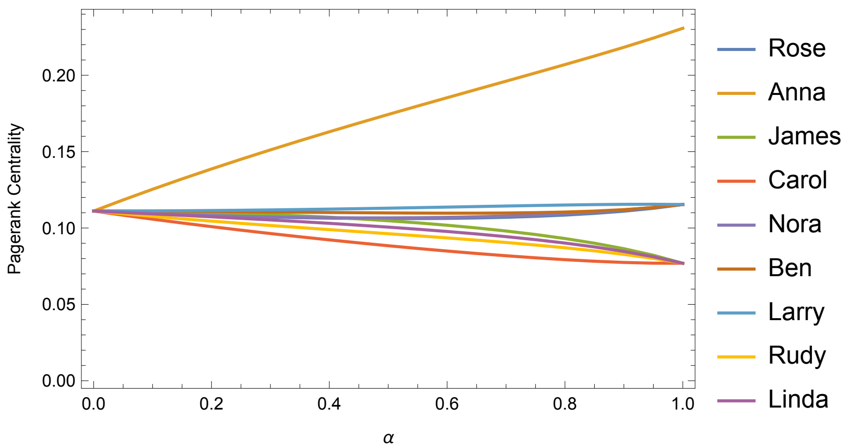
Out[96]=

α

```
{0.111111, 0.111111, 0.111111, 0.111111,
 0.111111, 0.111111, 0.111111, 0.111111, 0.111111}
```

In[97]:=
```
ListLinePlot[Transpose[Table[
    Transpose[{ConstantArray[α, 9], PageRankCentrality[g, α]}], {α, 0, 1, 0.05}]],
 PlotRange → All, Frame → True, FrameLabel → {"α", "Pagerank Centrality"},
 PlotLegends → VertexList[g]]
```

Out[97]=



The random walk connection

- The new matrix is a probability matrix! (every row sums to 1)
- This can be seen as a random walk on the (directed) graph (markov chain)
- The PageRank is the stationary distribution of the walk
- $\pi = \pi W$
  - W is the random walk matrix

In[98]:=
```
Eigenvalues[Transpose[RWP]]
```
Out[98]=
```
{1., -0.679299, 0.605967, -0.545527, -0.412877,
 0.345513, -0.300723, 0.136946, -4.12199 × 10^-17}
```

In[99]:=
```
Eigenvectors[Transpose[RWP]][[1]] / Total[Eigenvectors[Transpose[RWP]][[1]]]
```
Out[99]=
```
{0.109561, 0.212569, 0.0900488, 0.0782536,
 0.110498, 0.111081, 0.115324, 0.0850514, 0.0876125}
```

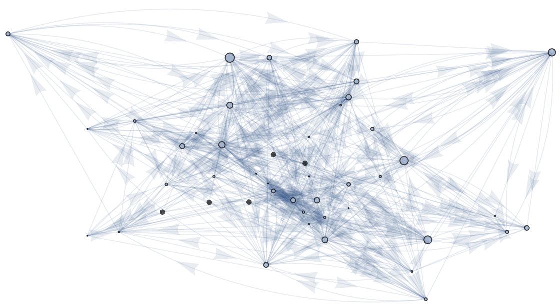
In[100]:=
```
N[Flatten[(Ones / 9)].MatrixPower[RWP, 30], 4]
```
Out[100]=
```
{0.109561, 0.212569, 0.0900487, 0.0782536,
 0.110498, 0.111081, 0.115324, 0.0850515, 0.0876126}
```

In[101]:=
```
PageRankCentrality[g, .85]
```
Out[101]=
```
{0.109561, 0.212569, 0.0900488, 0.0782536,
 0.110498, 0.111081, 0.115324, 0.0850514, 0.0876125}
```

Solve the Problems?

- No more zeros

In[102]:=
```
Sort[PageRankCentrality[EV, .85]]
```
Out[102]=
```
{0.00326087, 0.00326087, 0.00326087, 0.00326087, 0.00326087, 0.00687249,
 0.00705206, 0.00793215, 0.00841355, 0.00842358, 0.00927936, 0.0102776,
 0.0107251, 0.0116953, 0.0122916, 0.0125277, 0.0131506, 0.0132661, 0.015956,
 0.0159752, 0.0163831, 0.0167209, 0.0186467, 0.0190418, 0.0198363, 0.0221844,
 0.022645, 0.0239373, 0.0254536, 0.0273619, 0.0286472, 0.0306632, 0.0307535,
 0.0316488, 0.0319889, 0.0325549, 0.0326703, 0.034766, 0.0348847, 0.0367681,
 0.0385392, 0.0402534, 0.0456706, 0.0490839, 0.0513953, 0.0573582}
```
In[103]:=
```
Graph[EV, EdgeShapeFunction → "Arrow",
 VertexSize → Thread[VertexList[EV] → Rescale[PageRankCentrality[EV, .85]]]]
```
Out[103]=



Closeness Centrality

- How close is a node to the rest of the nodes
- The average distance between a node to the rest of the nodes

$$l_i = \frac{1}{n-1} \sum_j d_{ij}$$

- The Closeness centrality is then

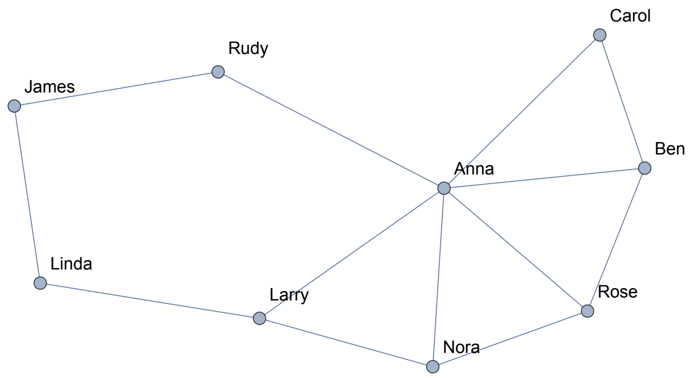
$$c_i = \frac{1}{l_i}$$

- Some disadvantages: in many graphs every one are close to every one else

In[104]:=

```
g
```

Out[104]=



In[105]:=

```
VertexList[g]
```

Out[105]=

{Rose, Anna, James, Carol, Nora, Ben, Larry, Rudy, Linda}

In[106]:=

```
GraphDistanceMatrix[g] // MatrixForm
```

Out[106]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 3 & 2 & 1 & 1 & 2 & 2 & 3 \\ 1 & 0 & 2 & 1 & 1 & 1 & 1 & 1 & 2 \\ 3 & 2 & 0 & 3 & 3 & 3 & 2 & 1 & 1 \\ 2 & 1 & 3 & 0 & 2 & 1 & 2 & 2 & 3 \\ 1 & 1 & 3 & 2 & 0 & 2 & 1 & 2 & 2 \\ 1 & 1 & 3 & 1 & 2 & 0 & 2 & 2 & 3 \\ 2 & 1 & 2 & 2 & 1 & 2 & 0 & 2 & 1 \\ 2 & 1 & 1 & 2 & 2 & 2 & 2 & 0 & 2 \\ 3 & 2 & 1 & 3 & 2 & 3 & 1 & 2 & 0 \end{pmatrix}$$

- Rose’s Distances

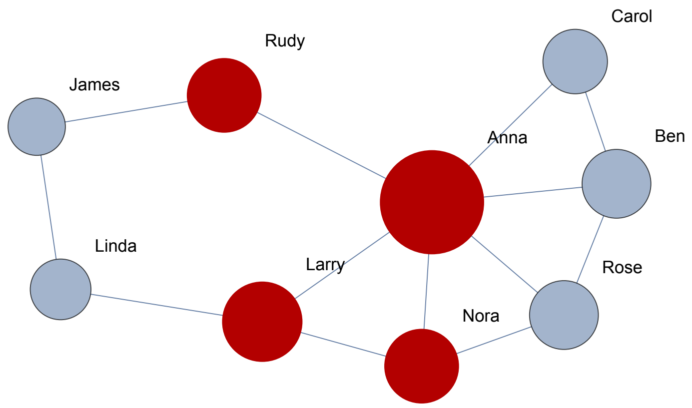
In[107]:=
```
GraphDistanceMatrix[g][[1]]
```
Out[107]=
```
{0, 1, 3, 2, 1, 1, 2, 2, 3}
```
In[108]:=
```
Total[GraphDistanceMatrix[g][[1]]]
```
Out[108]=
```
15
```

$$\frac{n - 1}{\sum_j d_{ij}}$$

In[109]:=
```
(VertexCount[g] - 1) / Total[GraphDistanceMatrix[g][[1]]] // N
```
Out[109]=
```
0.533333
```
In[110]:=
```
ClosenessCentrality[g]
```
Out[110]=
```
{0.533333, 0.8, 0.444444, 0.5, 0.571429, 0.533333, 0.615385, 0.571429, 0.470588}
```

- Let’s check

In[111]:=
```
HighlightGraph[g, VertexList[g][[Ordering[ClosenessCentrality[g], 4, Greater]]],
 VertexSize → Thread[VertexList[g] → ClosenessCentrality[g]]]
```
Out[111]=



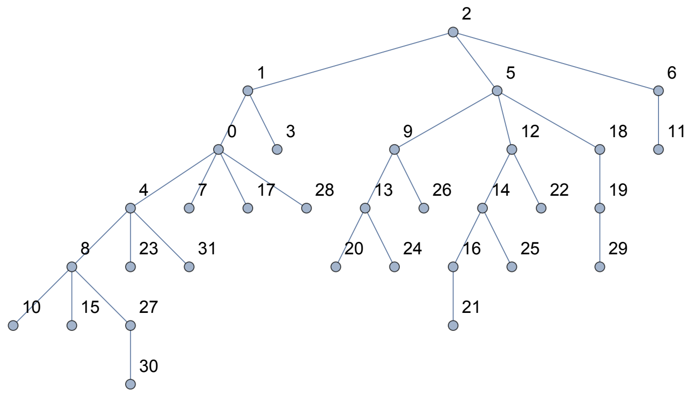
In[112]:=
```
Thread[VertexList[g][[Ordering[ClosenessCentrality[g], 5, Greater]]] →
  N[ClosenessCentrality[g][[Ordering[ClosenessCentrality[g], 5, Greater]]]]]
```
Out[112]=
```
{Anna → 0.8, Larry → 0.615385, Rudy → 0.571429, Nora → 0.571429, Ben → 0.533333}
```

- In Trees this is more important

In[117]:=

```
T = Graph[RandomInteger[#] ⟷ # + 1 & /@ Range[0, 30], VertexLabels → "Name"]
```

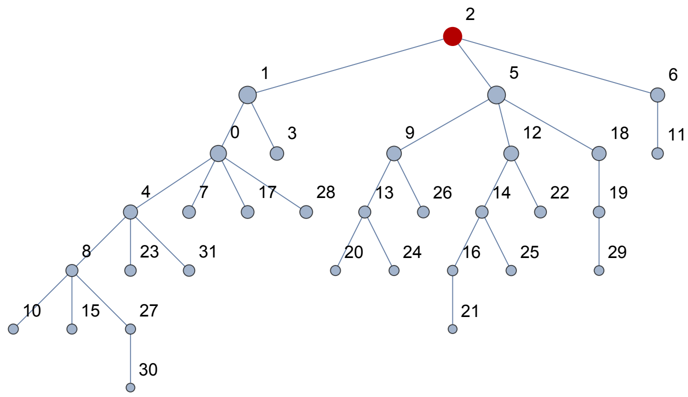
Out[117]=



In[118]:=

```
HighlightGraph[T, VertexList[T]⟦Ordering[ClosenessCentrality[T], 1, Greater]⟧,
 VertexSize → Thread[VertexList[T] → ClosenessCentrality[T]]]
```

Out[118]=

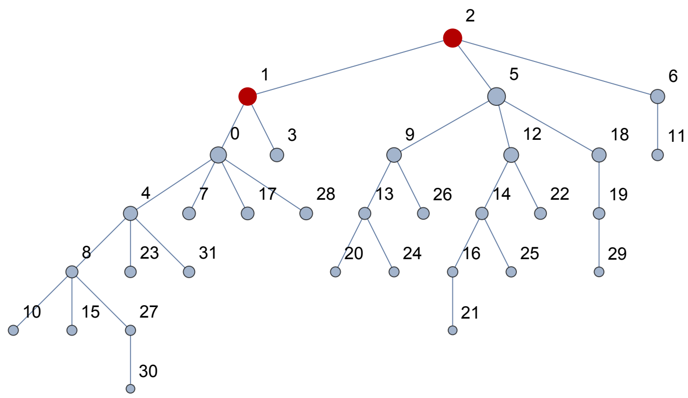


- Can be different from the Graph Center (The shortest longest path)

In[119]:=

```
HighlightGraph[T, GraphCenter[T],
 VertexSize → Thread[VertexList[T] → ClosenessCentrality[T]]]
```

Out[119]=

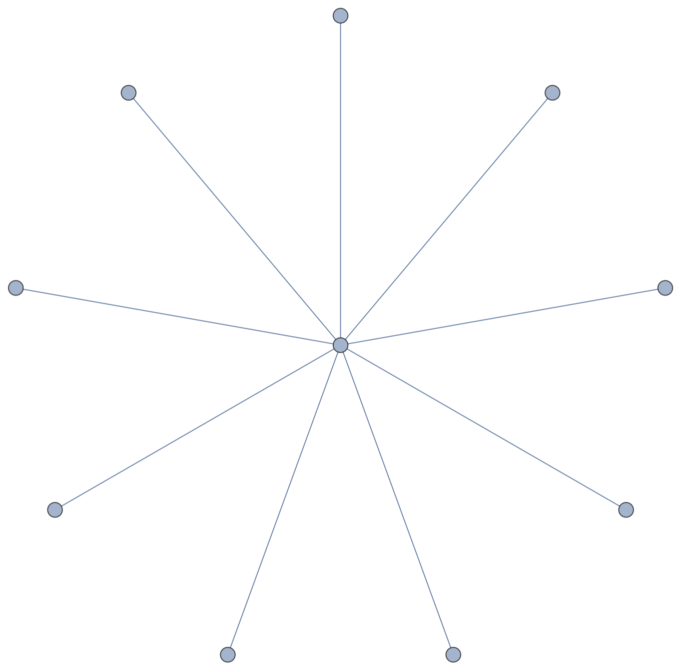


Betweenness Centrality

- How much “flow” pass via a node (or an edge)?
- Assuming information flows via shortest paths
- n_{st} - The number of shortest path from s to t
- n_{st}^{i} - The number of shortest path from s to t that pass through i

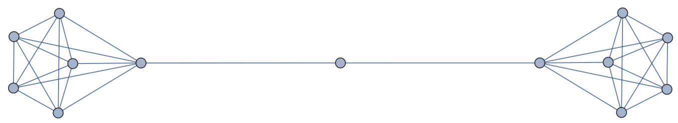
$$x_i = \sum_{st} \frac{n_{st}^{i}}{n_{st}}$$

In[120]:=
```
S = StarGraph[10]
```
Out[120]=



In[121]:=
```
BetweennessCentrality[S]
```
Out[121]=
```
{36., 0., 0., 0., 0., 0., 0., 0., 0., 0.}
```

In[122]:=
```
B = Graph[Union[Flatten[Table[Table[i ⟷ j, {j, i + 1, 6}], {i, 1, 6}]],
   Flatten[Table[Table[i ⟷ j, {j, i + 1, 12}], {i, 7, 12}]], {6 ⟷ 13}, {12 ⟷ 13}]]
```
Out[122]=



In[123]:=

```
BetweennessCentrality[B]
```

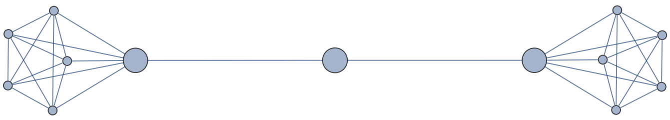
Out[123]=

```
{0., 0., 0., 0., 0., 35., 36., 0., 0., 0., 0., 0., 35.}
```

In[124]:=

```
Graph[B,
 VertexSize → Thread[VertexList[B] → Normalize[BetweennessCentrality[B] + 20]]]
```

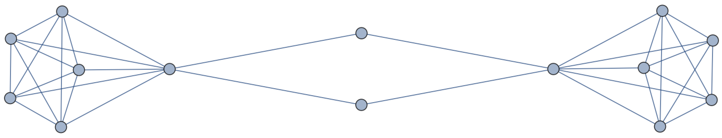
Out[124]=



In[125]:=

```
B1 = Graph[Union[Flatten[Table[Table[i ↔ j, {j, i + 1, 6}], {i, 1, 6}]],
   Flatten[Table[Table[i ↔ j, {j, i + 1, 12}], {i, 7, 12}]],
   {6 ↔ 13}, {12 ↔ 13}, {12 ↔ 14}, {6 ↔ 14}]]
```

Out[125]=



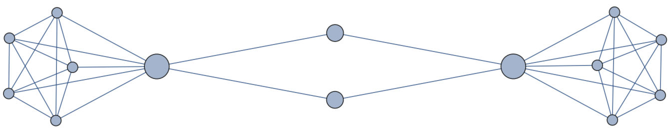
In[126]:=

```
BetweennessCentrality[B1]
```

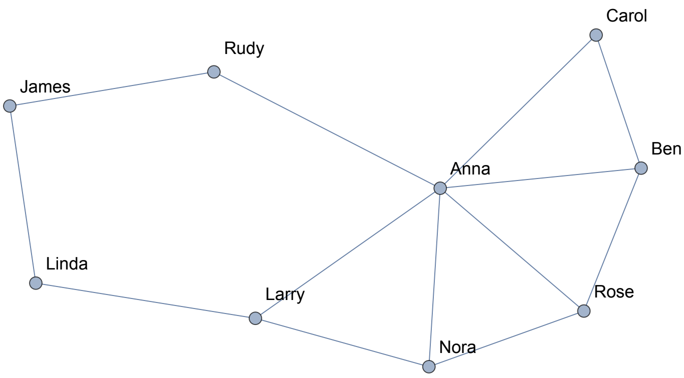
Out[126]=

```
{0., 0., 0., 0., 0., 40.5, 18., 18., 0., 0., 0., 0., 0., 40.5}
```

```
Graph[B1,
 VertexSize → Thread[VertexList[B1] → Normalize[BetweennessCentrality[B1] + 30]]]
```



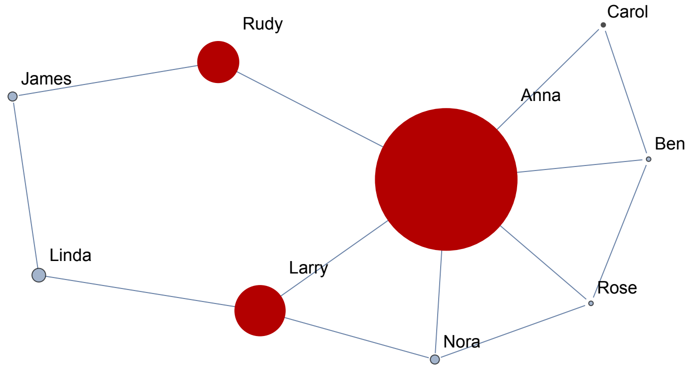
```
g
```



In[127]:=

```
HighlightGraph[g, VertexList[g]〚Ordering[BetweennessCentrality[g], 3, Greater]〛,
 VertexSize → Thread[VertexList[g] → Rescale[BetweennessCentrality[g]]]]
```

Out[127]=



In[128]:=

```
Thread[VertexList[g]〚Ordering[BetweennessCentrality[g], 5, Greater]〛 →
  N[BetweennessCentrality[g]〚Ordering[BetweennessCentrality[g], 5, Greater]〛]]
```

Out[128]=

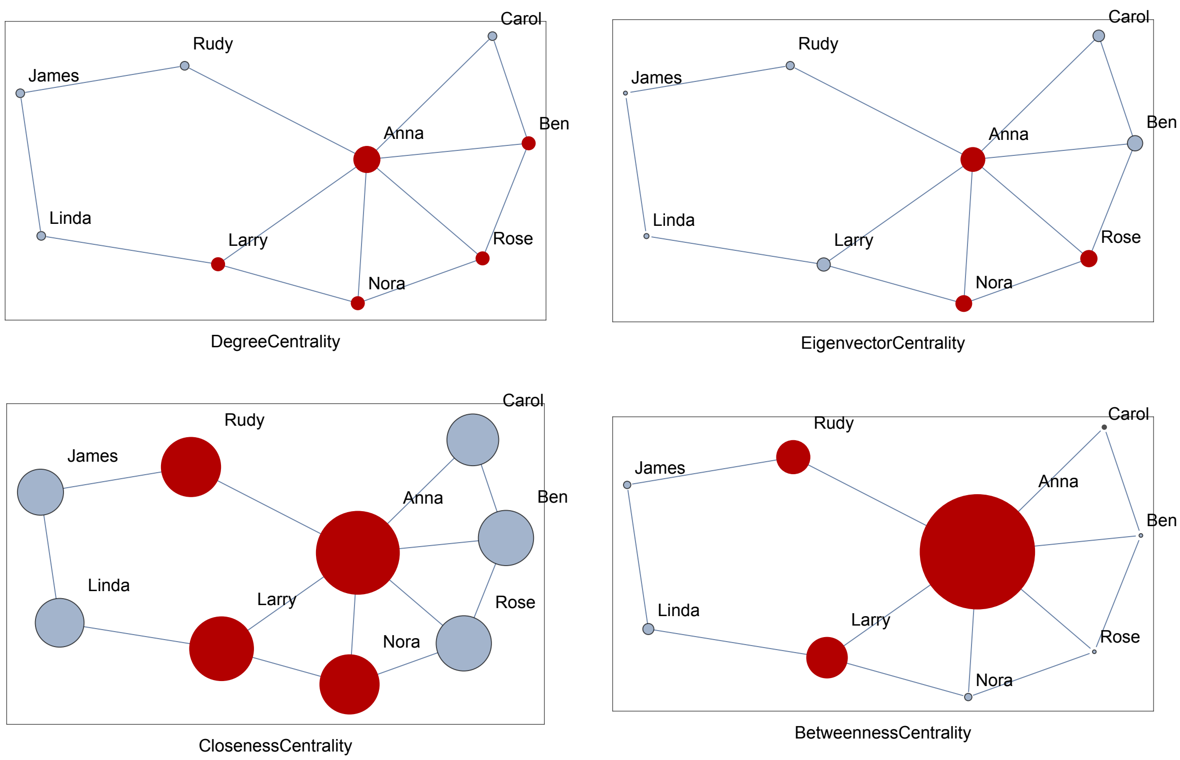
{Anna → 15.5, Larry → 5.5, Rudy → 4.5, Linda → 1.5, Nora → 1.}

Overview

In[129]:=

```
GraphicsGrid[{{HighlightGraph[g,
    VertexList[g]⟦Ordering[DegreeCentrality[g], 5, Greater]⟧, VertexSize →
     Thread[VertexList[g] → VertexDegree[g] / Total[ VertexDegree[g]]],
    Frame → True, FrameLabel → "DegreeCentrality"], HighlightGraph[g,
    VertexList[g]⟦Ordering[EigenvectorCentrality[g], 3, Greater]⟧,
    VertexSize → Thread[VertexList[g] → EigenvectorCentrality[g]],
    Frame → True, FrameLabel → "EigenvectorCentrality"], HighlightGraph[
    g, VertexList[g]⟦Ordering[PageRankCentrality[g, .85], 3, Greater]⟧,
    VertexSize → Thread[VertexList[g] → PageRankCentrality[g, .85]],
    Frame → True, FrameLabel → "PageRankCentrality"]}, {HighlightGraph[
    g, VertexList[g]⟦Ordering[ClosenessCentrality[g], 4, Greater]⟧,
    VertexSize → Thread[VertexList[g] → ClosenessCentrality[g]],
    Frame → True, FrameLabel → "ClosenessCentrality"], HighlightGraph[
    g, VertexList[g]⟦Ordering[BetweennessCentrality[g], 3, Greater]⟧,
    VertexSize → Thread[VertexList[g] → Rescale[BetweennessCentrality[g]]],
    Frame → True, FrameLabel → "BetweennessCentrality"]}}]
```

Out[129]=



In[130]:=

```
HighlightCentrality[g_, cc_, Title_] :=
  HighlightGraph[g, Table[Style[VertexList[g]〚i〛,
     ColorData["TemperatureMap"][cc〚i〛 / Max[cc]]], {i, VertexCount[g]}],
   VertexLabels → Table[i → Placed[cc〚i〛, Tooltip], {i, VertexCount[g]}],
   Frame → True, FrameLabel → Title];
```

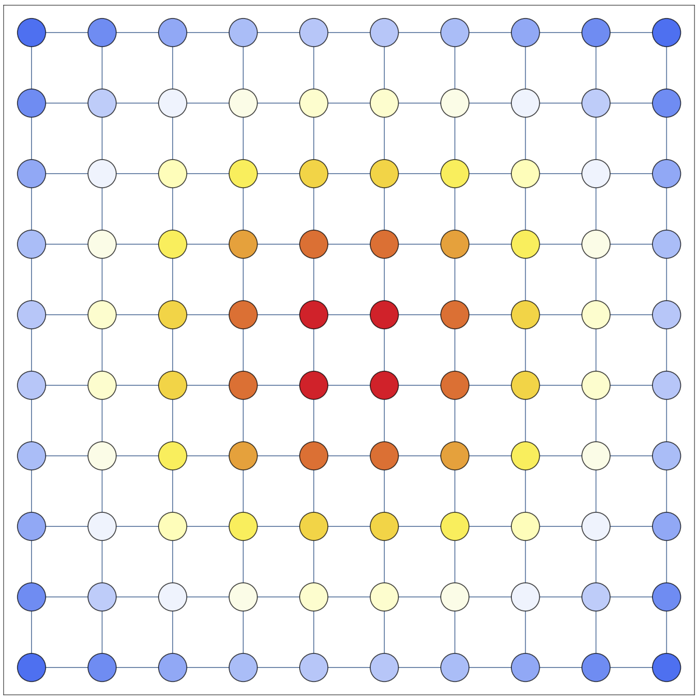
In[131]:=

```
g2 = GridGraph[{10, 10}, VertexSize → Large];
```

In[132]:=

```
HighlightCentrality[g2, EigenvectorCentrality[g2], "EigenvectorCentrality"]
```

Out[132]=

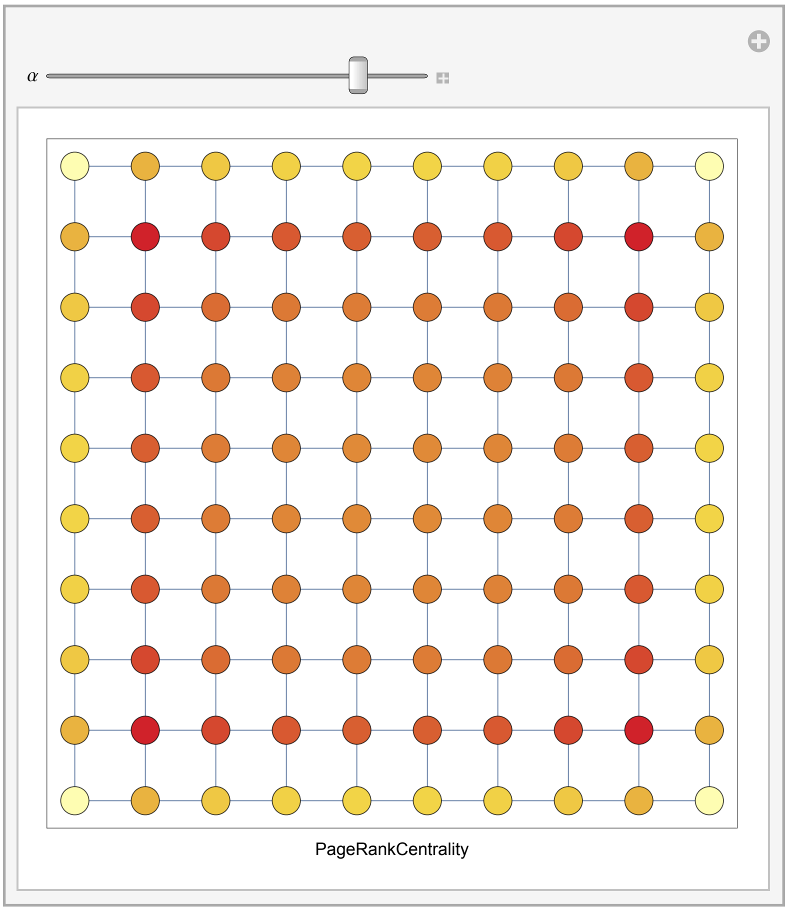


EigenvectorCentrality

In[134]:=

```
Manipulate[HighlightCentrality[g2,
  PageRankCentrality[g2, α], "PageRankCentrality"], {{α, 0.85}, 0, 1}]
```

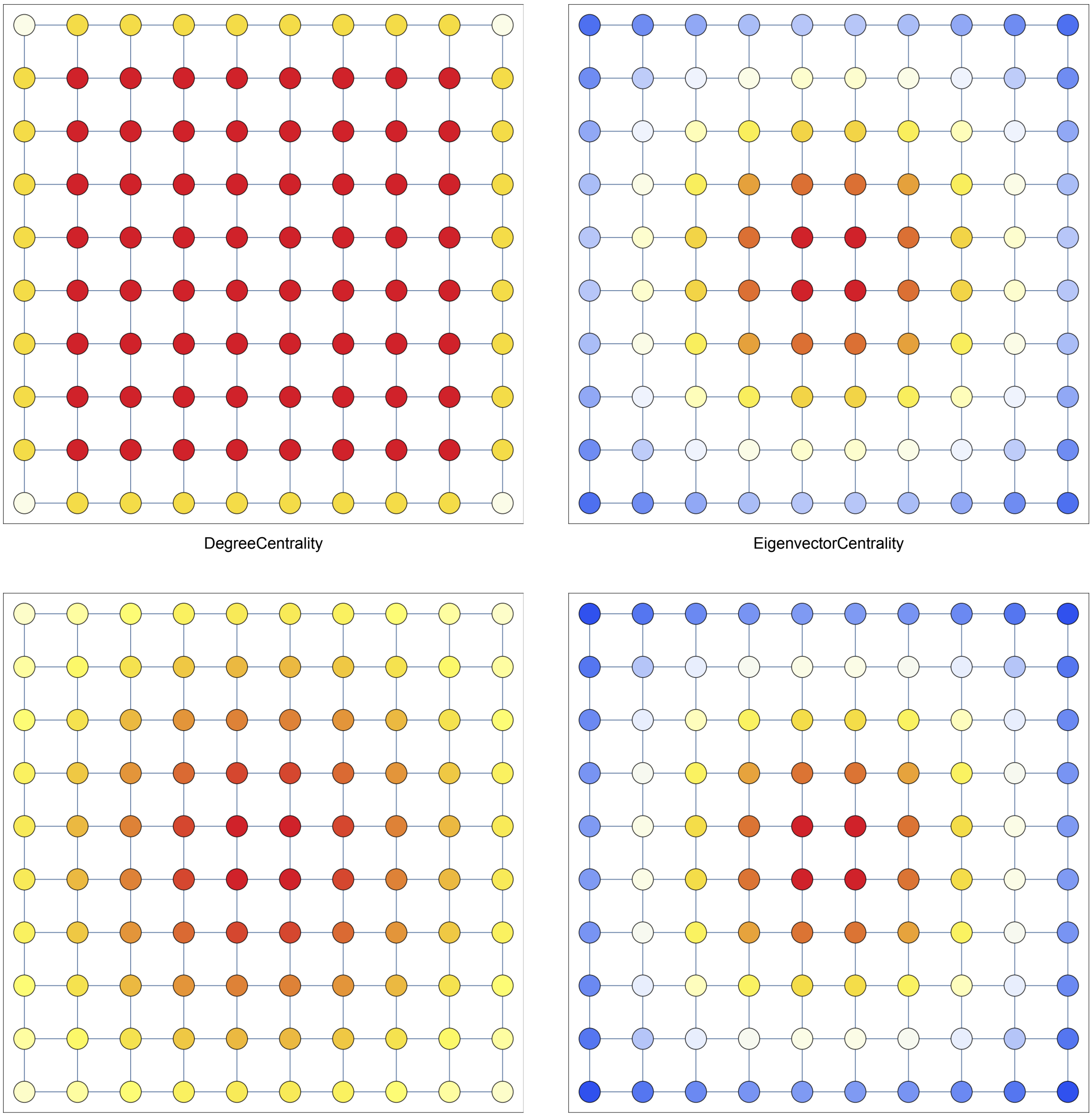
Out[134]=



In[135]:=

```
GraphicsGrid[
 {{HighlightCentrality[g2, DegreeCentrality[g2], "DegreeCentrality"],
   HighlightCentrality[g2, EigenvectorCentrality[g2], "EigenvectorCentrality"],
   HighlightCentrality[g2, PageRankCentrality[g2], "PageRankCentrality"]},
  {HighlightCentrality[g2, ClosenessCentrality[g2], "ClosenessCentrality"],
   HighlightCentrality[g2, BetweennessCentrality[g2],
    "BetweennessCentrality"]}}]
```

Out[135]=



DegreeCentrality

EigenvectorCentrality

ClosenessCentrality

BetweennessCentrality

Edge Betweenness

- Betweenness extend the notion of bridge and edges that have large social capital

$$e_i = \sum_{st} \frac{n^i_{st}}{n_{st}}$$

- Recall Example

In[136]:=
```
NS = ExampleData[{"NetworkGraph", "CoauthorshipsInNetworkScience"}];
```

In[137]:=
```
WLCC[g_] := Subgraph[g, First[WeaklyConnectedComponents[g]]];
```

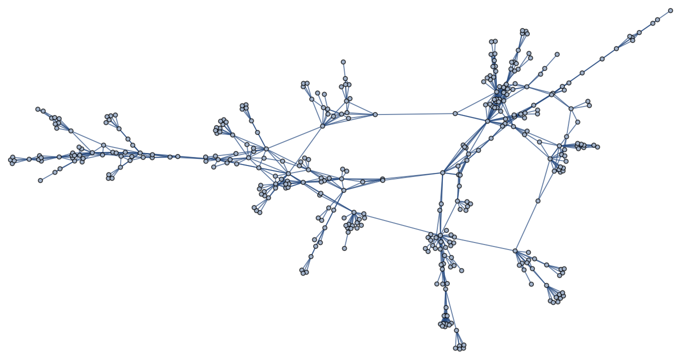
In[138]:=
```
GC = WLCC[NS];
```

In[139]:=
```
GCEL = EdgeList[GC];
```

In[140]:=
```
GCEW = PropertyValue[{NS, #}, EdgeWeight] & /@ EdgeList[GC];
```

In[141]:=
```
WG = Graph[GC, EdgeWeight → GCEW]
```

Out[141]=



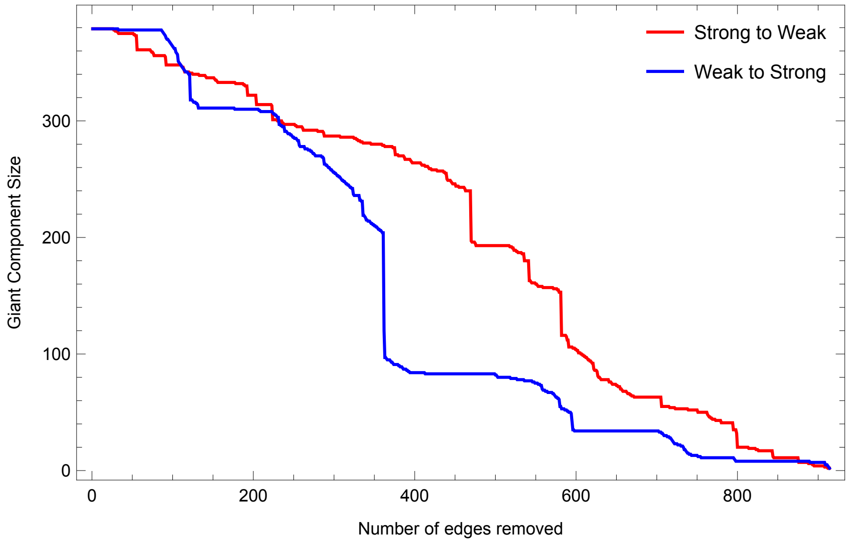
In[142]:=
```
EdgeWeightList[g_] := PropertyValue[{g, #}, EdgeWeight] & /@ EdgeList[g]
```

In[143]:=
```
SortEL = GCEL[[Ordering[EdgeWeightList[WG], All, Greater]]];
```

In[144]:=

```
ListLinePlot[
 {Table[Length@First@WeaklyConnectedComponents[Graph[Take[SortEL, i]]],
   {i, Length[SortEL], 1, -1}],
  Table[Length@First@WeaklyConnectedComponents[Graph[Take[SortEL, -i]]],
   {i, Length[SortEL], 1, -1}]}, PlotStyle → {Red, Blue, Green},
 PlotLegends → Placed[{"Strong to Weak", "Weak to Strong"}, {Right, Top}],
 FrameLabel → {"Number of edges removed", "Giant Component Size" } ,
 Frame → True]
```

Out[144]=



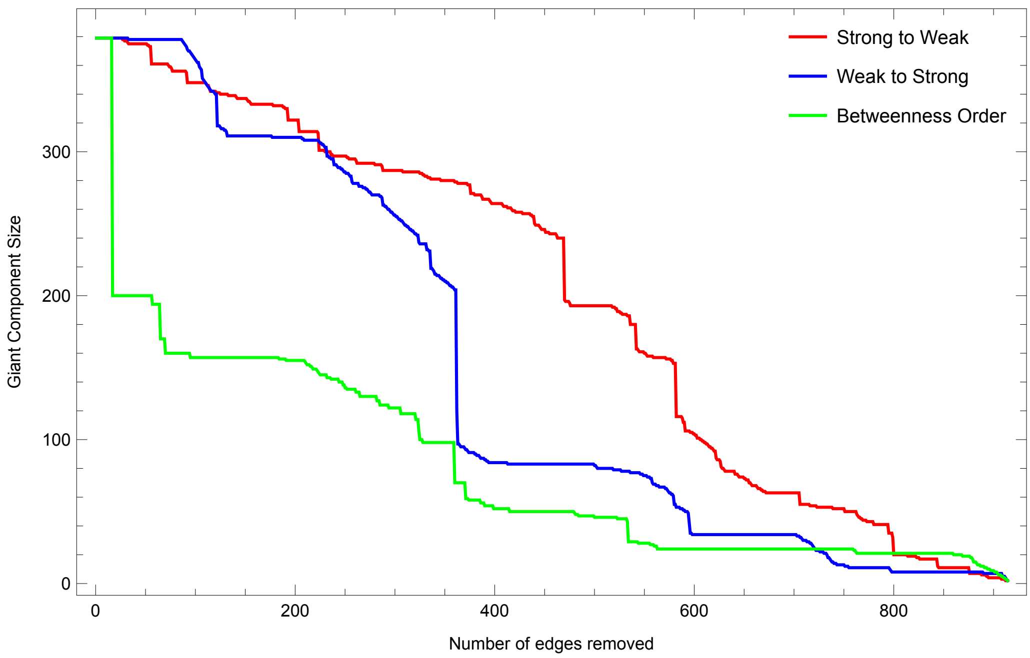
In[145]:=

```
SortBet = GCEL⟦Ordering[EdgeBetweennessCentrality[WG], All, Greater]⟧;
```

In[146]:=

```
ListLinePlot[
 {Table[Length@First@WeaklyConnectedComponents[Graph[Take[SortEL, i]]],
   {i, Length[SortEL], 1, -1}],
  Table[Length@First@WeaklyConnectedComponents[Graph[Take[SortEL, -i]]],
   {i, Length[SortEL], 1, -1}],
  Table[Length@First@WeaklyConnectedComponents[Graph[Take[SortBet, -i]]],
   {i, Length[SortBet], 1, -1}]},
 PlotStyle → {Red, Blue, Green}, PlotLegends → Placed[
   {"Strong to Weak", "Weak to Strong", "Betweenness Order"}, {Right, Top}],
 FrameLabel → {"Number of edges removed", "Giant Component Size" } ,
 Frame → True]
```

Out[146]=

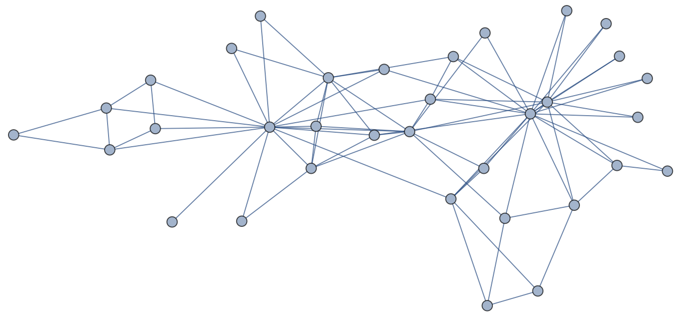


Using Edge Betweenness to Find Communities

In[147]:=

```
Karate = ExampleData[{"NetworkGraph", "ZacharyKarateClub"}]
```

Out[147]=



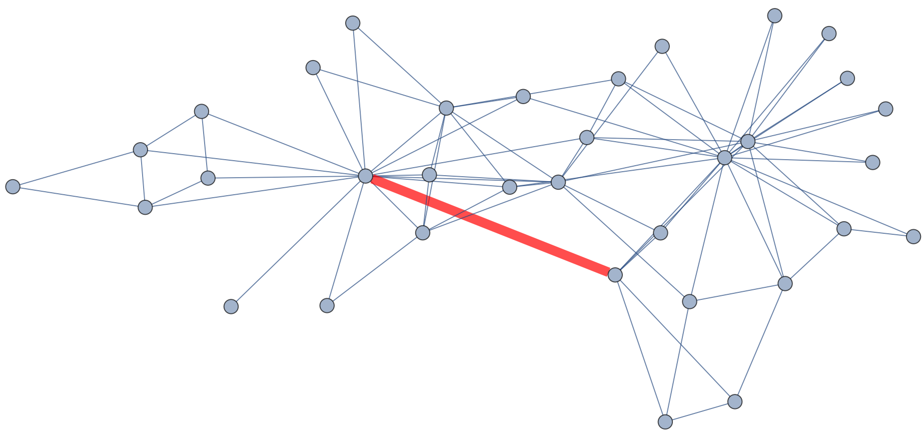
In[148]:=

```
TopEdgeBetweennessCentrality[g_] :=
 EdgeList[g]〚Ordering[EdgeBetweennessCentrality[g], -1]〛
```

In[155]:=

```
HighlightGraph[Karate,
 Style[TopEdgeBetweennessCentrality[Karate], Thickness[0.01], Red]]
```

Out[155]=



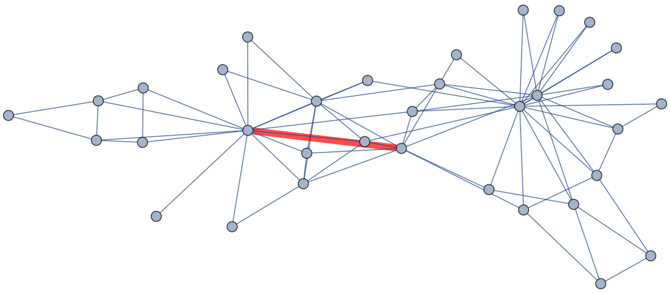
In[156]:=

```
K1 = EdgeDelete[Karate, TopEdgeBetweennessCentrality[Karate]];
```

In[157]:=

```
HighlightGraph[K1,
 Style[TopEdgeBetweennessCentrality[K1], Red, Thickness[0.01]]]
```

Out[157]=



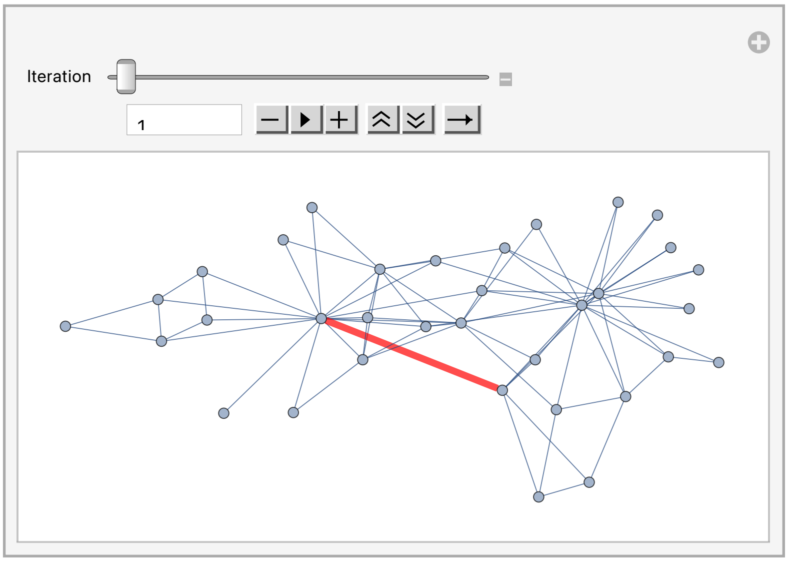
In[158]:=

```
NextBetweennessEdge[g_, i_] := Module[{graph = g, list = {}},
  For[j = 1, j ≤ i, j++,
   list = Union[list, TopEdgeBetweennessCentrality[graph]];
   graph = EdgeDelete[graph, TopEdgeBetweennessCentrality[graph]];];
  list]
```

In[160]:=

```
Manipulate[HighlightGraph[Karate,
  Style[NextBetweennessEdge[Karate, i], Red, Thickness[0.01]]],
 {{i, 1, "Iteration"}, 1, EdgeCount[Karate], 1}]
```

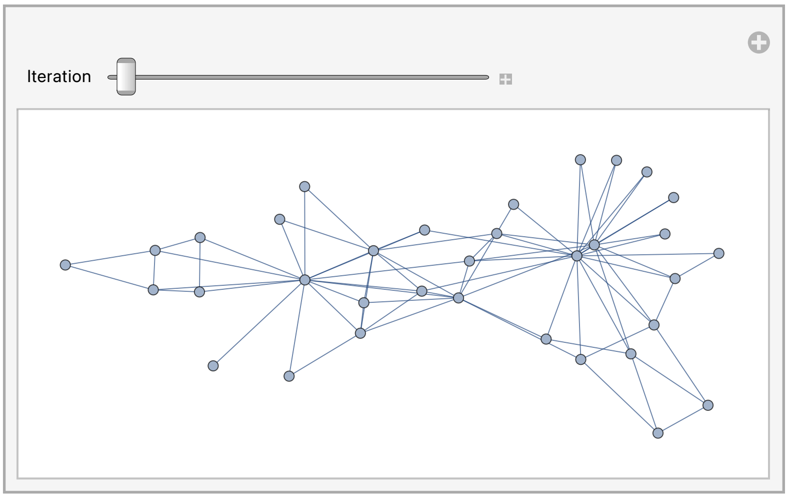
Out[160]=



In[174]:=

```
Manipulate[EdgeDelete[Karate, NextBetweennessEdge[Karate, i]],
 {{i, 1, "Iteration"}, 1, EdgeCount[Karate], 1}]
```

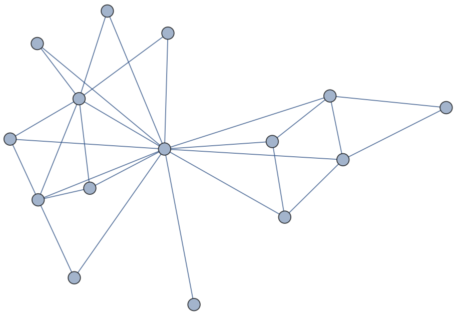
Out[174]=

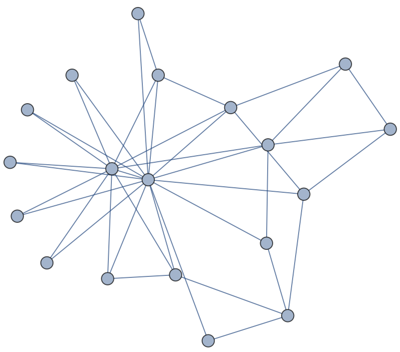


In[175]:=

```
Comm = EdgeDelete[Karate, NextBetweennessEdge[Karate, 11]]
```

Out[175]=

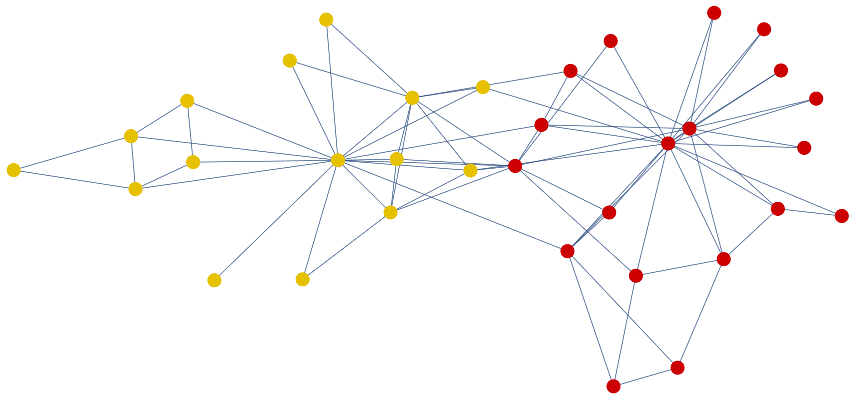




In[171]:=

```
HighlightGraph[Karate, ConnectedComponents[Comm]]
```

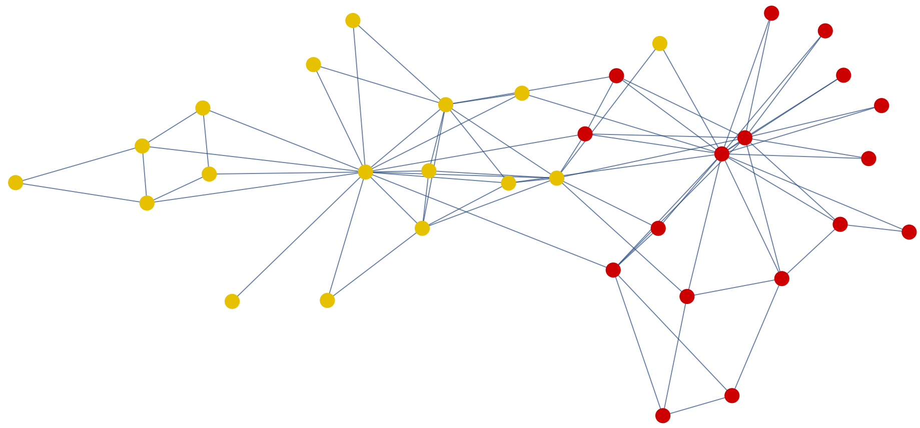
Out[171]=



In[172]:=

```
HighlightGraph[Karate, Reverse[FindGraphPartition[Karate]]]
```

Out[172]=

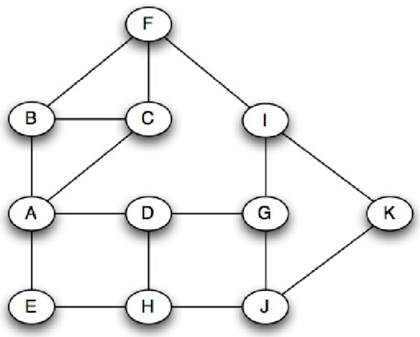


Computing Betweenness (in this example edge betweenness)

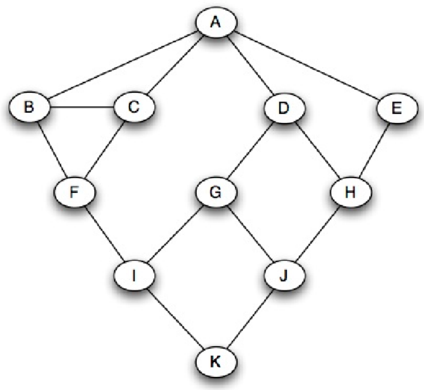
- $e_i = \sum_{st} \frac{n_{st}^i}{n_{st}}$
- Naive approach
  - n^2 shortest path, each can take O(m) (using BFS) so can take $O(mn^2)$
  - Can do in O(nm) by combining BFS
- Algorithm:

(1) Perform a breadth-first search of the graph, starting at A.
(2) Determine the number of shortest paths from A to each other node.
(3) Based on these numbers, determine the amount of flow from A to all other nodes that uses each edge.

- First Step

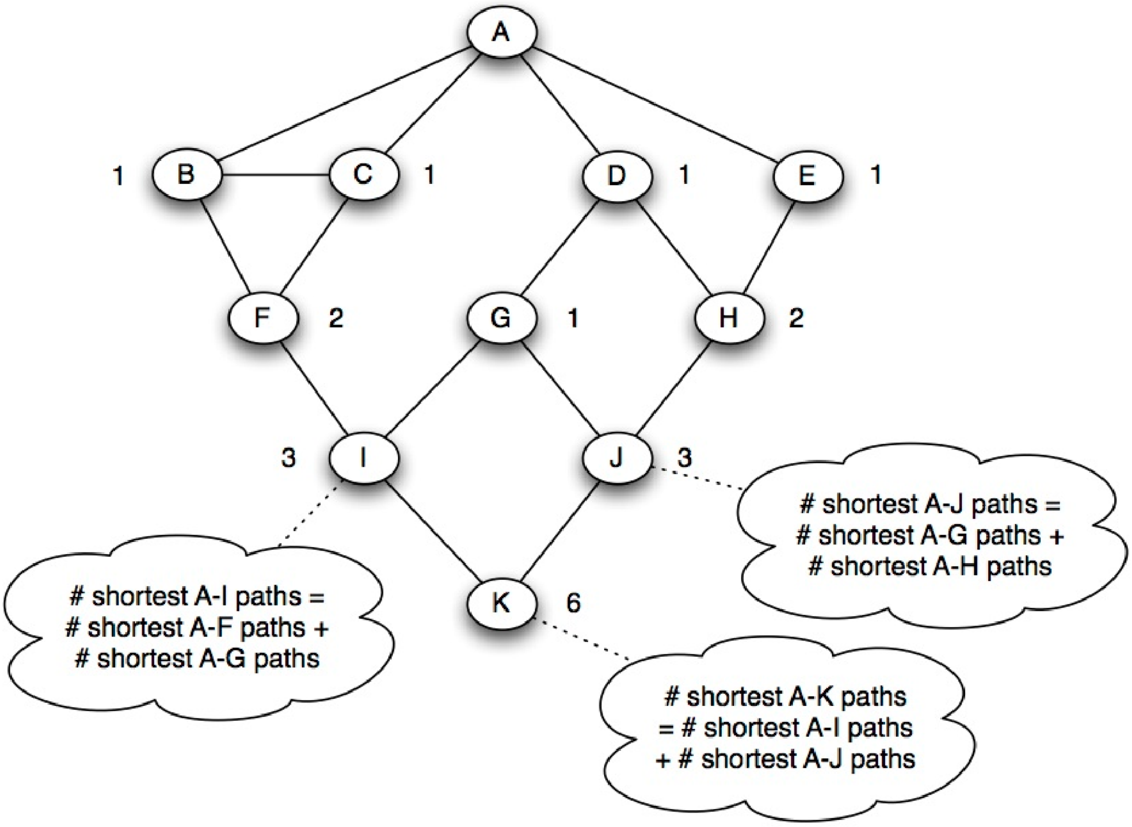


(a) *A sample network*

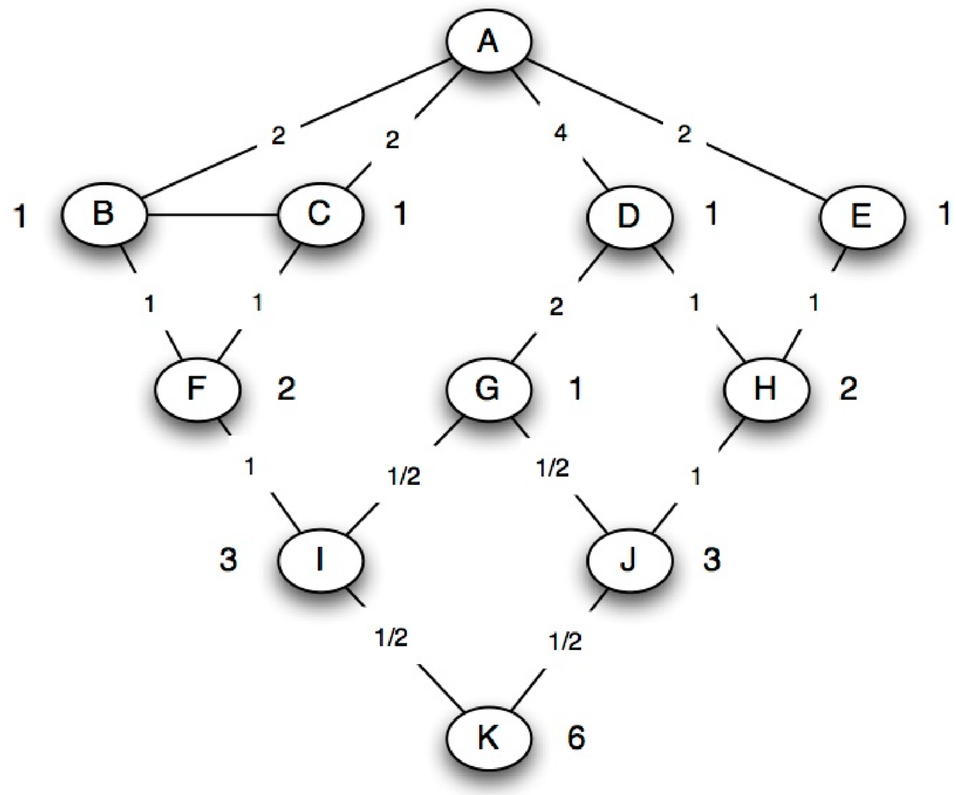


(b) *Breadth-first search starting at node A*

- Second Step - Counting shortest paths



- Third Step - Computing Betweenness
- Start from leaf - take the flow from your children and add 1 for yourself and split to your parents according to their weights.



- Repeat the same for each node and **sum up** the betweenness on **each edge** to get it's value
- We double count path from x to y and y to x so can divide by 2 for undirected graphs

Centrality Homework - For next week

- Compute centrality measures for "Network of Thrones" (when possible with and without weights).
- Present it as nice as possible (probably only showing top results) - bonus
- See figures in "Network of Thrones"
- See figures in https://en.wikipedia.org/wiki/Centrality