In [1]:
```
import numpy as np
from scipy.sparse import csc_matrix
from scipy.sparse.linalg import eigs

edges_file = open('wisconsin_edges.csv', "r")
nodes_file = open('wisconsin_nodes.csv', "r")

# create a dictionary where nodes_dict[i] = name of wikipedia page
nodes_dict = {}
for line in nodes_file:
    nodes_dict[int(line.split(',',1)[0].strip())] = line.split(',',1)[

node_count = len(nodes_dict)

# create adjacency matrix
A = np.zeros((node_count, node_count))
for line in edges_file:
    from_node = int(line.split(',')[0].strip())
    to_node = int(line.split(',')[1].strip())
    A[to_node, from_node] = 1.0

## Add code below to (1) prevent traps and (2) find the most important
# Hint -- instead of computing the entire eigen-decomposition of a mat
# s, E = np.linalg.eig(A)
# you can compute just the first eigenvector with:
# s, E = eigs(csc_matrix(A), k = 1)
```

a)

In [13]:
```
# adding 0.001 to each entry of A
for i in range(len(A)):
    for j in range(len(A[0])):
                    A[i][j] += 0.001
# normalizing A
A = A / A.sum(0)

# eigen decompostion
w, v = np.linalg.eig(A)
```

In [25]:
```
v1 = v[:, np.argmax(w)] # first e-vector
v1 = np.absolute(v1)
v1_indx = np.argsort(v1)
print(v1_indx)
```

```
[2901 2186 2190 ... 1345 2312 5089]
```

b)

In [33]:
```
print("The page ranked 1st is titled :", nodes_dict[v1_indx[-1]])
```

```
The page ranked 1st is titled : "Wisconsin"
```

c)

In [31]:
```
print("The page ranked 3rd is titled :", nodes_dict[v1_indx[-3]])
```

```
The page ranked 3rd is titled : "Madison, Wisconsin"
```