

Data from <https://snap.stanford.edu/data/web-Google.html> (<https://snap.stanford.edu/data/web-Google.html>)

In [94]:

```
def nice_print(v, digits=3):  
    format = '%%.%df' % digits  
    print ', '.join([format % e for e in v])
```

In [95]:

```
nice_print([.12333122, .13432221, .64442143])
```

0.123, 0.134, 0.644

In [96]:

```
nice_print([.12333122, .13432221, .64442143])
```

0.123, 0.134, 0.644

In [101]:

```
phone_number={'Gioele': 143222, 'Eugenio': 738221, 'Ludovica': 925446, 'Armand
```

In [104]:

```
names = ['Gioele', 'Eugenio', 'Ludovica', 'Armando', 'Harshney']  
numbers = [234, 554, 873, 185, 878]  
zip(names, numbers)
```

Out[104]:

```
[('Gioele', 234),  
 ('Eugenio', 554),  
 ('Ludovica', 873),  
 ('Armando', 185),  
 ('Harshney', 878)]
```

In [111]:

```
[nu+2 for na, nu in zip(names, numbers)]
```

Out[111]:

[236, 556, 875, 187, 880]

In [112]:

```
labels = ['A',  
          'B',  
          'C',  
          'D',  
          'E',  
          'F',  
          'G']  
  
pages = range(len(labels))  
  
page_labels = {page: label for page, label in zip(pages, labels)}
```

In [113]:

```
page_labels
```

Out[113]:

```
{0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G'}
```

In [114]:

```
links = [(1, 0),  
         (3, 0),  
         (0, 1),  
         (5, 2),  
         (6, 2),  
         (6, 5),  
         (5, 6),  
         (2, 6),  
         (0, 6),  
         (5, 4),  
         (4, 3)]
```

In [115]:

```

import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline

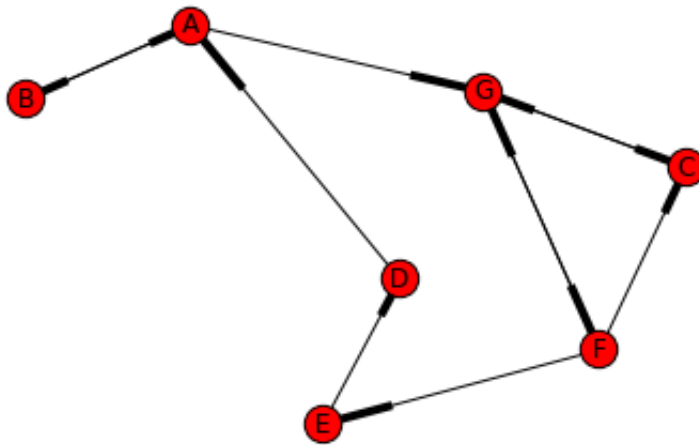
g = nx.DiGraph()

for p in pages:
    g.add_node(p)

for (a, b) in links:
    g.add_edge(pages[a], pages[b])

nx.draw(g, with_labels=True, labels=page_labels)

```



In [119]:

```

incidency = {}
for u in range(len(pages)):
    incidency[u] = []

for (a, b) in links:
    incidency[a].append(b)

```

In [120]:

incidency

Out[120]:

```

{0: [1, 6], 1: [0], 2: [6], 3: [0], 4: [3], 5: [2, 6, 4], 6: [2, 5]}

```

In [121]:

```
connection_matrix = []
for a in incidencey:
    for b in incidencey[a]:
        connection_matrix.append((b, a, 1./len(incidencey[a])))
```

In [122]:

```
connection_matrix
```

Out[122]:

```
[(1, 0, 0.5),
 (6, 0, 0.5),
 (0, 1, 1.0),
 (6, 2, 1.0),
 (0, 3, 1.0),
 (3, 4, 1.0),
 (2, 5, 0.3333333333333333),
 (6, 5, 0.3333333333333333),
 (4, 5, 0.3333333333333333),
 (2, 6, 0.5),
 (5, 6, 0.5)]
```

In [123]:

```
links_RDD = sc.parallelize(connection_matrix).cache()

import numpy as np
n = len(pages)
page_rank = np.ones(n)/n
old_page_rank = np.ones(n)
```

In [124]:

```
def l2distance(v, q):
    return sum([(q_el - v_el)**2 for v_el, q_el in zip(v, q)])

tolerance = 10e-7
max_iterations = 1000
```

In [125]:

```

iteration = 0
while(l2distance(old_page_rank, page_rank) >= tolerance and iteration < max_it
    old_page_rank = page_rank
    page_rank_rdd = (links_RDD
        .map(lambda (i, j, m): (i, m*page_rank[j]))
        .reduceByKey(lambda a, b: a+b)
        .collect()
    )
    page_rank = [0.] * n
    for (i, v) in page_rank_rdd:
        page_rank[i] = v

    nice_print(page_rank)

    iteration += 1

```

```

0.286, 0.071, 0.119, 0.143, 0.048, 0.071, 0.262
0.214, 0.143, 0.155, 0.048, 0.024, 0.131, 0.286
0.190, 0.107, 0.187, 0.024, 0.044, 0.143, 0.306
0.131, 0.095, 0.200, 0.044, 0.048, 0.153, 0.329
0.139, 0.065, 0.216, 0.048, 0.051, 0.165, 0.317
0.113, 0.069, 0.213, 0.051, 0.055, 0.158, 0.340
0.120, 0.057, 0.223, 0.055, 0.053, 0.170, 0.323
0.111, 0.060, 0.218, 0.053, 0.057, 0.161, 0.340
0.113, 0.056, 0.224, 0.057, 0.054, 0.170, 0.327
0.112, 0.056, 0.220, 0.054, 0.057, 0.164, 0.337
0.110, 0.056, 0.223, 0.057, 0.055, 0.168, 0.331
0.113, 0.055, 0.222, 0.055, 0.056, 0.166, 0.334
0.110, 0.056, 0.222, 0.056, 0.055, 0.167, 0.333
0.113, 0.055, 0.222, 0.055, 0.056, 0.167, 0.333
0.110, 0.056, 0.222, 0.056, 0.056, 0.166, 0.334
0.112, 0.055, 0.223, 0.056, 0.055, 0.167, 0.332
0.111, 0.056, 0.222, 0.055, 0.056, 0.166, 0.334
0.111, 0.055, 0.222, 0.056, 0.055, 0.167, 0.333
0.111, 0.056, 0.222, 0.055, 0.056, 0.166, 0.334
0.111, 0.055, 0.222, 0.056, 0.055, 0.167, 0.333
0.111, 0.056, 0.222, 0.055, 0.056, 0.166, 0.334

```

In [126]:

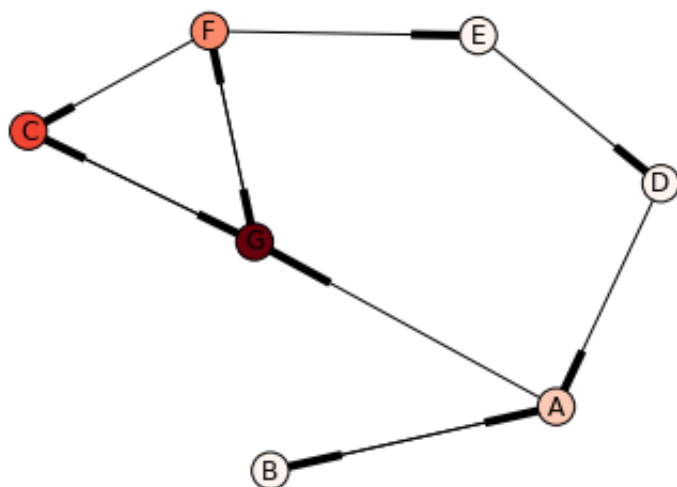
iteration

Out[126]:

21

In [127]:

```
nx.draw(g, with_labels=True, labels=page_labels, node_color=page_rank, cmap=pl
```



In [128]:

```

def get_graph(pages, links):
    g = nx.DiGraph()

    for p in pages:
        g.add_node(p)

    for (a, b) in links:
        g.add_edge(pages[a], pages[b])

    return g

def get_connection_matrix(pages, links):
    incidencey = {}
    for u in range(len(pages)):
        incidencey[u] = []

    for (a, b) in links:
        incidencey[a].append(b)

    connection_matrix = []
    for a in incidencey:
        for b in incidencey[a]:
            connection_matrix.append((b, a, 1./len(incidencey[a])))

    return connection_matrix

def get_page_rank(pages, links, verbose=False, tolerance=10e-7, max_iterations=100):
    connection_matrix = get_connection_matrix(pages, links)
    links_RDD = sc.parallelize(connection_matrix).cache()

    n = len(pages)
    page_rank = np.ones(n)/n
    old_page_rank = np.ones(n)

    iteration = 0
    while(l2distance(old_page_rank, page_rank) >= tolerance and iteration < max_iterations):
        old_page_rank = page_rank
        page_rank_rdd = (links_RDD
                        .map(lambda (i, j, m): (i, m*page_rank[j]))
                        .reduceByKey(lambda a, b: a+b)
                        .collect()
                        )
        page_rank = [0.] * n
        for (i, v) in page_rank_rdd:
            page_rank[i] = v

        if verbose:
            nice_print(page_rank)

        iteration += 1

    print ' %d iterations' % iteration

    return page_rank

```

In [129]:

```
page_rank = get_page_rank(pages, links)
```

21 iterations

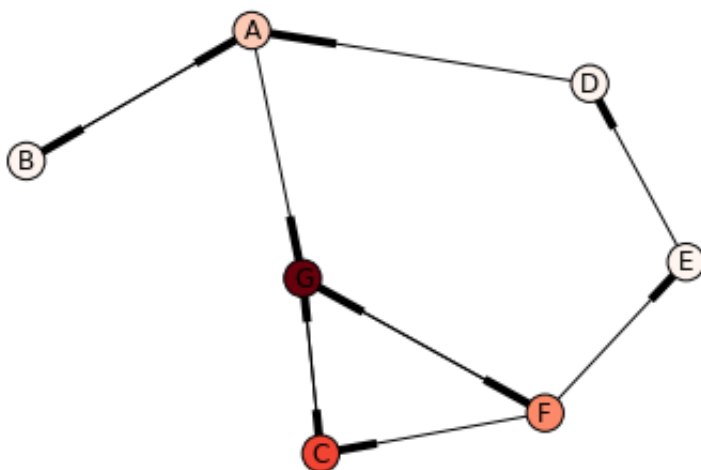
In [18]:

```
nice_print(page_rank)
```

0.111, 0.056, 0.222, 0.055, 0.056, 0.166, 0.334

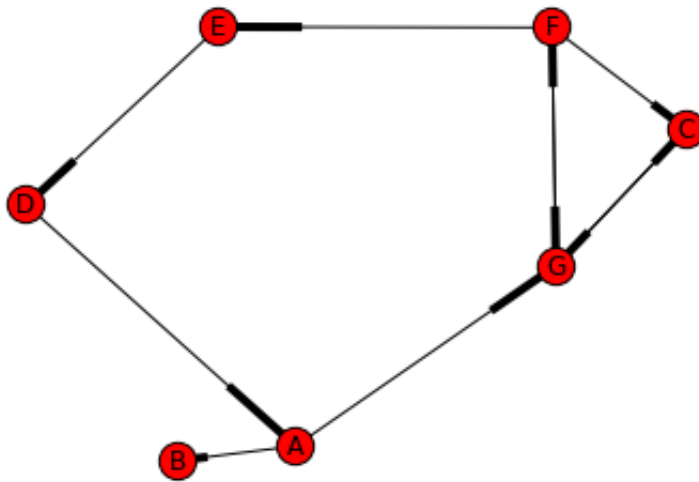
In [130]:

```
g = get_graph(pages, links)  
nx.draw(g, with_labels=True, labels=page_labels, node_color=page_rank, cmap=pl
```



In [131]:

```
links = [  
    (3, 0),  
    (0, 1),  
    (5, 2),  
    (6, 2),  
    (6, 5),  
    (5, 6),  
    (2, 6),  
    (0, 6),  
    (5, 4),  
    (4, 3),  
]  
  
g = get_graph(pages, links)  
nx.draw(g, with_labels=True, labels=page_labels)
```



In [132]:

```
page_rank = get_page_rank(pages, links)  
nice_print(page_rank)
```

```
73 iterations  
0.004, 0.002, 0.014, 0.004, 0.004, 0.010, 0.020
```

In [22]:

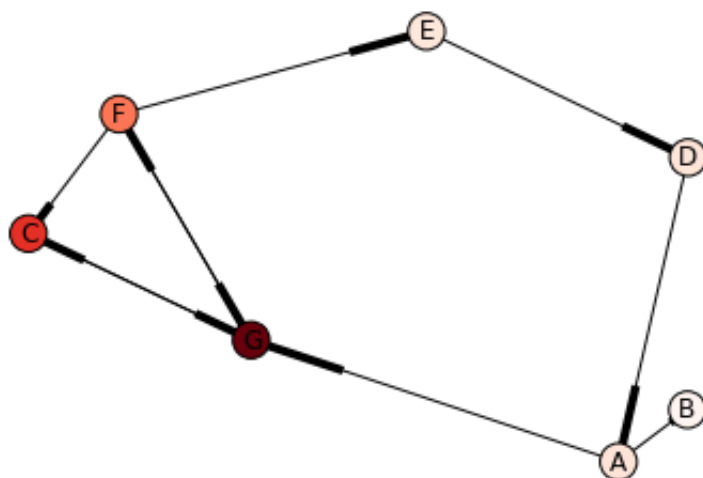
```
sum(page_rank)
```

Out[22]:

```
0.05721862657052279
```

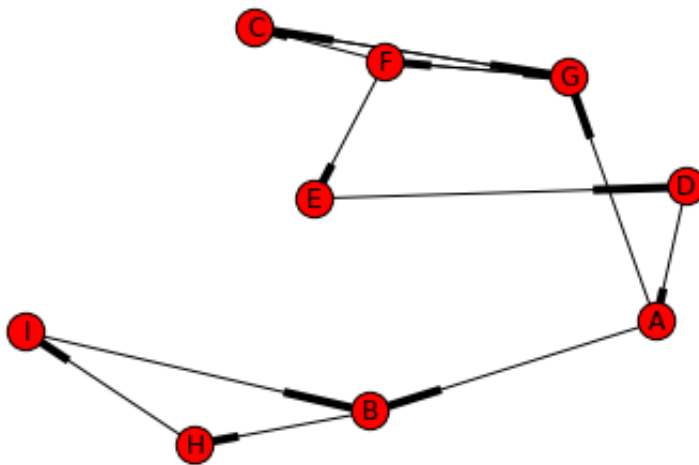
In [23]:

```
nx.draw(g, with_labels=True, labels=page_labels, node_color=page_rank, cmap=pl
```



In [24]:

```
labels = ['A',  
          'B',  
          'C',  
          'D',  
          'E',  
          'F',  
          'G',  
          'H',  
          'I']  
  
pages = range(len(labels))  
  
page_labels = {page: label for page, label in zip(pages, labels)}  
  
links = [  
    (3, 0),  
    (0, 1),  
    (5, 2),  
    (6, 2),  
    (6, 5),  
    (5, 6),  
    (2, 6),  
    (0, 6),  
    (5, 4),  
    (4, 3),  
    (1, 7), (7, 8), (8, 1)  
]  
  
g = get_graph(pages, links)  
nx.draw(g, with_labels=True, labels=page_labels)
```



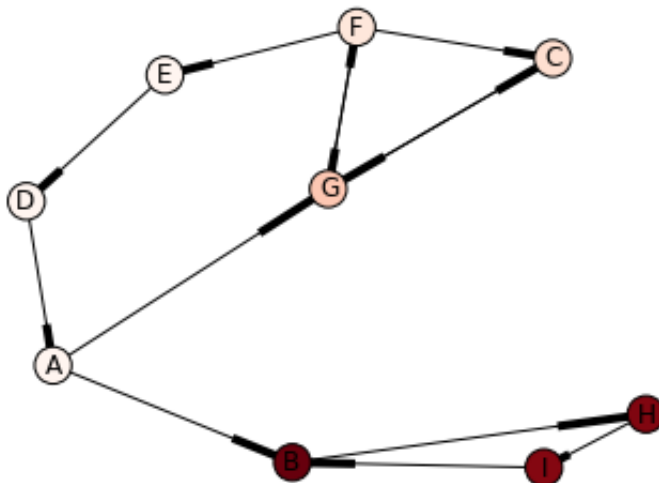
In [41]:

```
page_rank = get_page_rank(pages, links, tolerance=3e-4)
nice_print(page_rank)
```

```
31 iterations
0.013, 0.280, 0.047, 0.013, 0.012, 0.035, 0.068, 0.266, 0.264
```

In [42]:

```
nx.draw(g, with_labels=True, labels=page_labels, node_color=page_rank, cmap=pl
```



In [72]:

```
def get_page_rank(pages, links, beta=0.8, max_iterations=400, tolerance=1.e-5,
                  connection_matrix = get_connection_matrix(pages, links)
                  links_RDD = sc.parallelize(connection_matrix).cache()

    n = len(pages)
    page_rank = np.ones(n)/n
    old_page_rank = np.ones(n)

    iteration = 0
    while(l2distance(old_page_rank, page_rank) >= tolerance and iteration < ma
        old_page_rank = page_rank
        page_rank_rdd = (links_RDD
                        .map(lambda (i, j, m): (i, m*page_rank[j]))
                        .reduceByKey(lambda a, b: a+b)
                        .sortByKey()
                        .collect()
                        )
        page_rank = [1./beta * 1./n] * n
        for (i, v) in page_rank_rdd:
            page_rank[i] += (1-1./beta) * v

        if verbose:
            nice_print(page_rank)

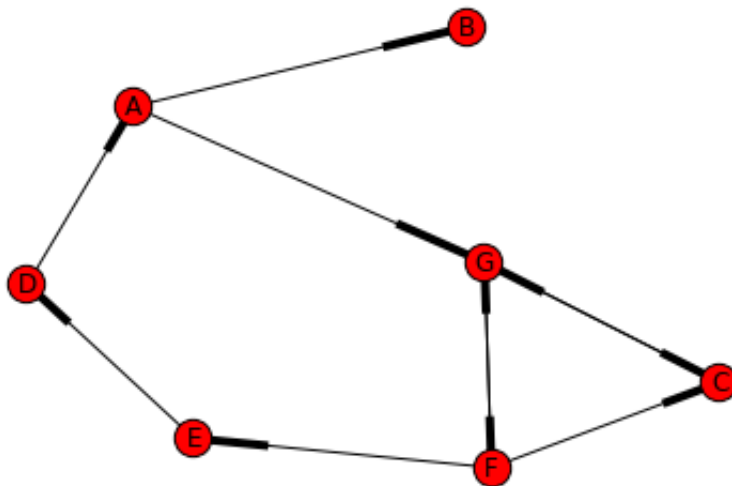
        iteration += 1

    print ' %d iterations' % iteration

    return page_rank
```

In [44]:

```
labels = ['A',  
          'B',  
          'C',  
          'D',  
          'E',  
          'F',  
          'G']  
  
pages = range(len(labels))  
  
page_labels = {page: label for page, label in zip(pages, labels)}  
  
links = [  
    (3, 0),  
    (0, 1),  
    (5, 2),  
    (6, 2),  
    (6, 5),  
    (5, 6),  
    (2, 6),  
    (0, 6),  
    (5, 4),  
    (4, 3),  
]  
  
g = get_graph(pages, links)  
nx.draw(g, with_labels=True, labels=page_labels)
```



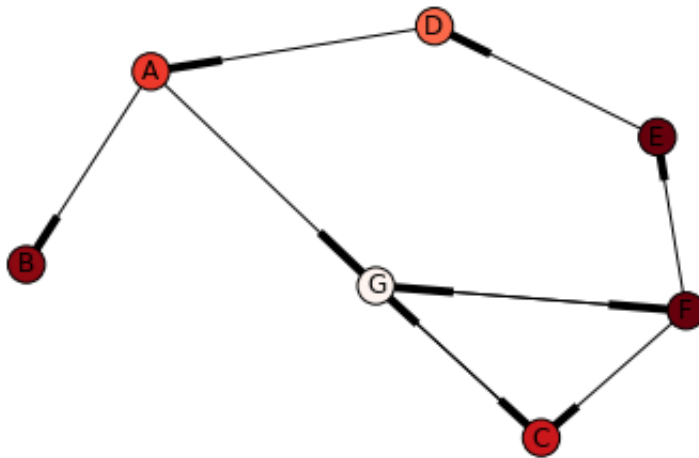
In [59]:

```
page_rank = get_page_rank(pages, links, 0.8)  
nice_print(page_rank)
```

```
6 iterations  
0.144, 0.161, 0.151, 0.137, 0.165, 0.165, 0.109
```

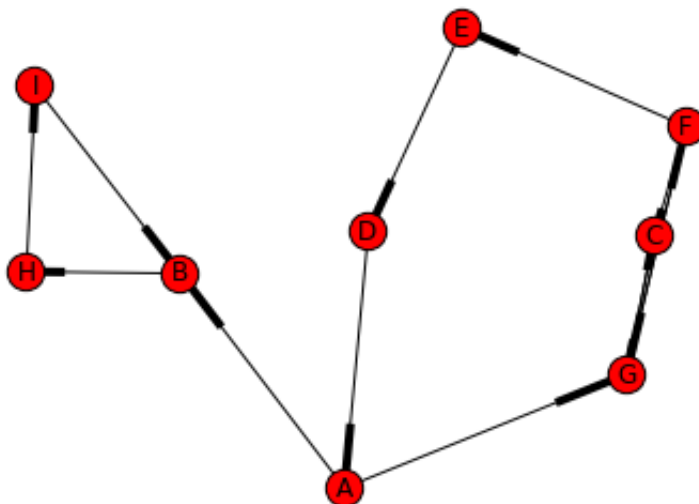
In [60]:

```
nx.draw(g, with_labels=True, labels=page_labels, node_color=page_rank, cmap=pl
```



In [61]:

```
labels = ['A',  
          'B',  
          'C',  
          'D',  
          'E',  
          'F',  
          'G',  
          'H',  
          'I']  
  
pages = range(len(labels))  
  
page_labels = {page: label for page, label in zip(pages, labels)}  
  
links = [  
    (3, 0),  
    (0, 1),  
    (5, 2),  
    (6, 2),  
    (6, 5),  
    (5, 6),  
    (2, 6),  
    (0, 6),  
    (5, 4),  
    (4, 3),  
    (1, 7), (7, 8), (8, 1)  
]  
  
g = get_graph(pages, links)  
nx.draw(g, with_labels=True, labels=page_labels)
```



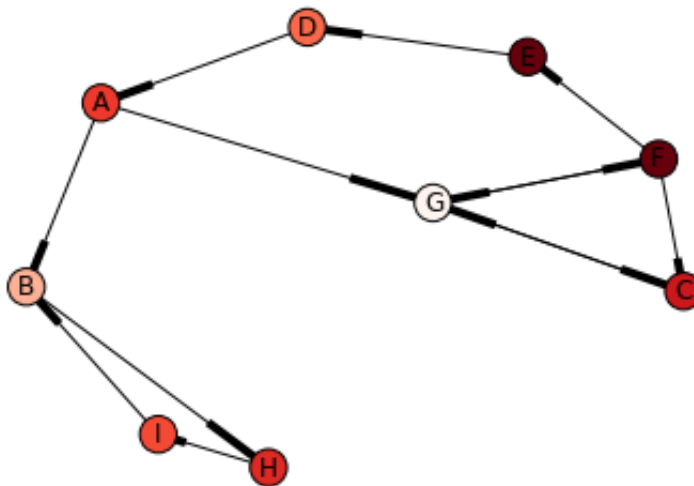
In [62]:

```
page_rank = get_page_rank(pages, links, 0.8)
nice_print(page_rank)
```

```
8 iterations
0.112, 0.097, 0.118, 0.107, 0.128, 0.128, 0.085, 0.115, 0.110
```

In [63]:

```
nx.draw(g, with_labels=True, labels=page_labels, node_color=page_rank, cmap=pl
```



In [64]:

```
def get_pages_and_links(file):
    web = sc.textFile(file)
    lines = web.map(lambda l: [int(s) for s in l.split('\t')]).collect()

    pages = set()
    page_index = {}
    n = 0

    for link in lines:
        for p in link:
            if p not in pages:
                pages.add(p)
                page_index[p] = n
                n += 1

    pages = [0] * n
    for p in page_index:
        pages[page_index[p]] = p

    links = []
    for (a, b) in lines:
        links.append((page_index[a], page_index[b]))

    return (pages, links)
```

In [65]:

```
pages, links = get_pages_and_links('data/small-web.txt')
```

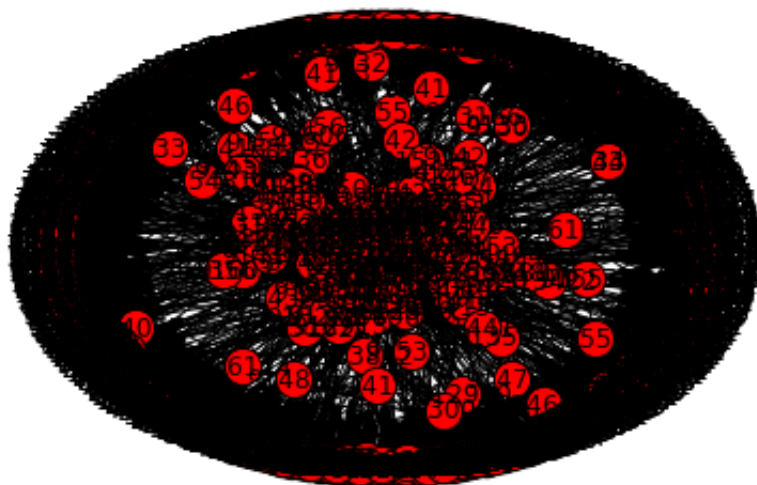
In [66]:

```
%matplotlib inline

g = get_graph(pages, links)
```

In [71]:

```
nx.draw(g, with_labels=True)
```



In [67]:

```
len(pages)
```

Out[67]:

1132

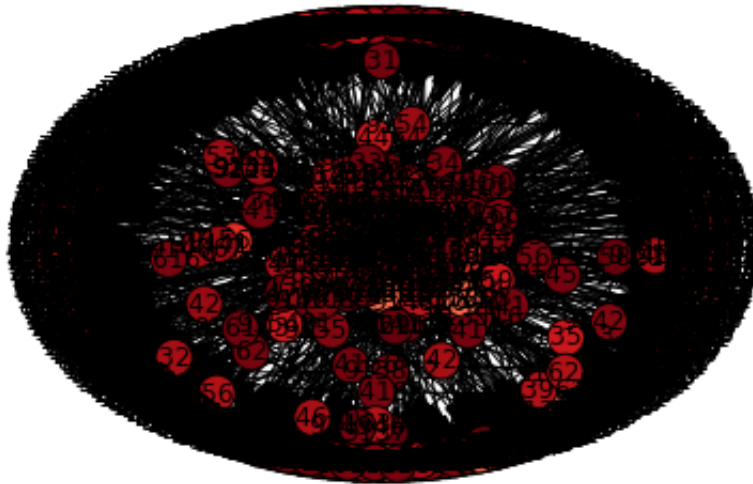
In [85]:

```
page_rank = get_page_rank(pages, links, 0.8, tolerance=1.e-8)
```

3 iterations

In [86]:

```
nx.draw(g, with_labels=True, node_color=page_rank, cmap=plt.cm.Red)
```



In [87]:

```
num_and_rank = zip(range(len(pages)), page_rank)
num_and_rank.sort(key = lambda e: -e[1])
```

In [88]:

```
num_and_rank[:5]
```

Out[88]:

```
[(0, 0.0011042402826855124),
 (14, 0.0011042402826855124),
 (22, 0.0011042402826855124),
 (38, 0.0011042402826855124),
 (40, 0.0011042402826855124)]
```

In [89]:

```
num_and_rank[-5:]
```

Out[89]:

```
[(188, 0.0008051752061248529),
 (229, 0.0006901501766784453),
 (230, 0.0006901501766784453),
 (274, 0.00067174617196702),
 (206, 0.0005521201413427562)]
```

In []: