Lec36 Strong and weak relationships : Finding communities in Graph

Girvan Newman Algorithm

- Based on edge betweenness
- The edges which are connecting two communities tend to have high betweenness

GNA WIKI (https://en.wikipedia.org/wiki/Girvan%E2%80%93Newman_algorithm)

- Vertex betweenness is an indicator of highly central nodes in networks. For any node i, vertex betweenness is defined as the number of shortest paths between pairs of nodes that run through it. It is relevant to models where the network modulates transfer of goods between known start and end points, under the assumption that such transfer seeks the shortest available route.
- The Girvan–Newman algorithm extends this definition to the case of edges, defining the "edge betweenness" of an edge as the number of shortest paths between pairs of nodes that run along it.

In [2]:

```
import networkx as nx
import matplotlib.pyplot as plt
```

In [32]:

```
def edge_to_remove(G):
    dict1 = nx.edge_betweenness_centrality(G) #Key - edges as tuple ;Value - edge b
    list_of_tuples = dict1.items()#list has tuples, tuple has the above data
    #print "List of Tuples : ", list_of_tuples
    list_of_tuples.sort(key = lambda x:x[1], reverse = True) #Descending order
    return list_of_tuples[0][0] #(a, b) #edge with max edge betweenness

def girvan(G):
    # Returns connected components as subgrahs
    c = list(nx.connected_component_subgraphs(G))
    l = len(c)
    print 'The number of connected components are ', l
    while l == 1:
        G.remove_edge(*edge_to_remove(G)) # ((a, b)) -> (a, b)
        c = list(nx.connected_component_subgraphs(G))
        l = len(c)
        print 'The number of connected components are ', l

    return c
```

In [30]:

```
G = nx.barbell_graph(5, 0)
c = girvan(G)

for i in c: #i is a graph
    print i.nodes()
    print "----------", i.number_of_nodes()
```

```
The number of connected components are  1
List of Tuples :  [((0, 4), 0.13333333333333333), ((6, 9), 0.022222222
222222223), ((1, 4), 0.13333333333333333), ((5, 9), 0.1333333333333333
3), ((5, 6), 0.13333333333333333), ((1, 3), 0.022222222222222223),
 ((2, 3), 0.022222222222222223), ((0, 3), 0.022222222222222223), ((5,
 8), 0.13333333333333333), ((7, 9), 0.022222222222222223), ((0, 1), 0.
022222222222222223), ((8, 9), 0.022222222222222223), ((6, 7), 0.022222
222222222223), ((6, 8), 0.022222222222222223), ((3, 4), 0.133333333333
33333), ((1, 2), 0.022222222222222223), ((0, 2), 0.02222222222222222
3), ((4, 5), 0.5555555555555556), ((5, 7), 0.13333333333333333), ((7,
 8), 0.022222222222222223), ((2, 4), 0.13333333333333333)]
The number of connected components are  2
[0, 1, 2, 3, 4]
---------- 5
[8, 9, 5, 6, 7]
---------- 5
```

In [33]:

```
G = nx.karate_club_graph()
c = girvan(G)

for i in c: #i is a graph
    print i.nodes()
    print "----------", i.number_of_nodes()
```

```
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  1
The number of connected components are  2
[0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21]
---------- 15
[32, 33, 2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30,
 31]
---------- 19
```

In [19]:

```
nx.connected_component_subgraphs?
```

Lec37 Strong and weak relationships : Visualing communities using Gephi

- Settings -> Modularity -> Run
- Default resolution : 1
- Increase the resolution to get less communities and decrease the resolution to get less communities
- Use partition to differentiate edges
- Force atlas layout is better in this case (Increase repulsion strenth to get a better graph)