SME0130 - Redes Complexas

Structure of networks: Network Centrality

Professor: Francisco Aparecido Rodrigues, francisco@icmc.usp.br (mailto:francisco@icmc.usp.br).
Estudante: Bruno F. Bessa (num. 5881890), bruno.fernandes.oliveira@usp.br (mailto:bruno.fernandes.oliveira@usp.br)
Universidade de São Paulo, São Carlos, Brasil.

In [10]:

```
import numpy as np
import networkx as nx
import community as community_louvain
from community import community_louvain
import matplotlib.pyplot as plt
import math
from scipy import stats
from scipy.cluster.hierarchy import dendrogram
from itertools import chain, combinations
```

In [23]:

```
def get_graph_from_data_file(file_name='lesmis.txt', ncols=3):

    '''
    Defines a NetworkX graph based on data from file.
    Plots a visual representation of the graph
    '''

    file_path = 'data/' + file_name

    if ncols == 2:
        G = nx.read_edgelist("data/powergrid.txt", nodetype=int)
    else:
        G = nx.read_edgelist(file_path, nodetype=int, data=(('weight', float),))

    pos = nx.spring_layout(G)
    nx.draw(G, pos, node_color='b', node_size=50, with_labels=False)

    G = G.to_undirected()
    G.remove_edges_from(nx.selfloop_edges(G))
    Gcc = sorted(nx.connected_components(G), key=len, reverse=True)
    G = G.subgraph(Gcc[0])
    G = nx.convert_node_labels_to_integers(G, first_label=0)

    return G

def get_LFR(N=128, tau1=3, tau2=1.5, mu = 0.05, k=16, minc=32, maxc=32):

    G = nx.LFR_benchmark_graph(n = N, tau1 = tau1, tau2 = tau2, mu = mu, min_degree
    pos=nx.spring_layout(G)
    fig= plt.figure(figsize=(10,6))
    nx.draw(G, pos=pos, node_color = 'lightblue', with_labels = True)
    plt.show(True)

    return G
```

In [12]:

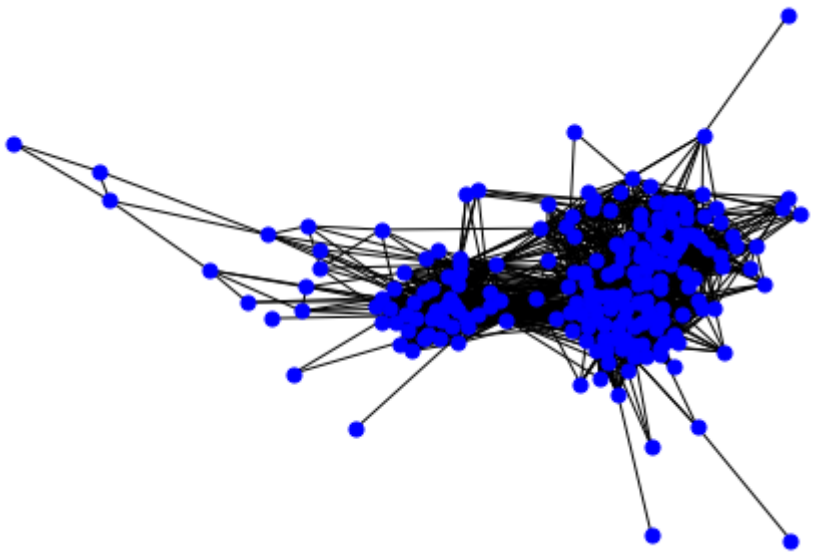
```
def communities_fastgreedy(G):

    c = list(nx.algorithms.community.greedy_modularity_communities(G))
    communities = np.zeros(len(G.nodes()))

    nc = 0
    for k in range(0, len(c)):
        communities[sorted(c[k])] = nc
        nc += 1
    return communities


def modularity(G, c):

    A = nx.adjacency_matrix(G)
    N = len(G)
    M = G.number_of_edges()
    Q = 0

    for i in np.arange(0,N):
        ki = len(list(G.neighbors(i)))
        for j in np.arange(0,N):
            if(c[i]==c[j]):
                kj = len(list(G.neighbors(j)))
                Q = Q + A[i,j]-(ki*kj)/(2*M)
    Q = Q/(2*M)

    return Q
```

Questions

1 - Calcule a modularidade para a rede Jazz usando método fastgreedy.

In [13]:

```
G = get_graph_from_data_file('jazz.txt')
```



In [14]:

```
communities = communities_fastgreedy(G)
Q = modularity(G, communities)

print("Modularidade da rede \"Jazz\" pelo método Fast Greedy: {:.4f}".format(Q))
```

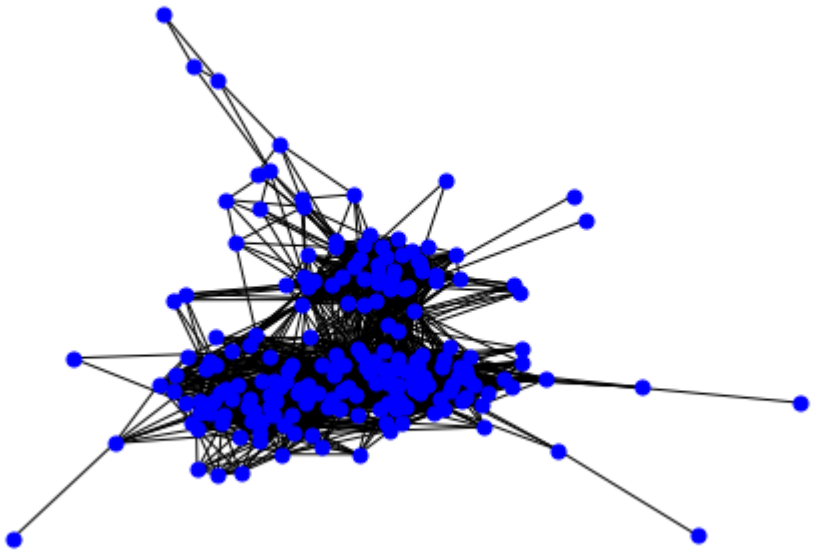
```
Modularidade da rede "Jazz": 0.4389
```

2 - Calcule a modularidade para a rede Jazz usando método Louvain.

In [22]:

```
G = get_graph_from_data_file('jazz.txt')
partition = community_louvain.best_partition(G)
Q = modularity(G, partition)

print("Modularidade da rede \"Jazz\": {:.4f} pelo método Louvain".format(Q))
```

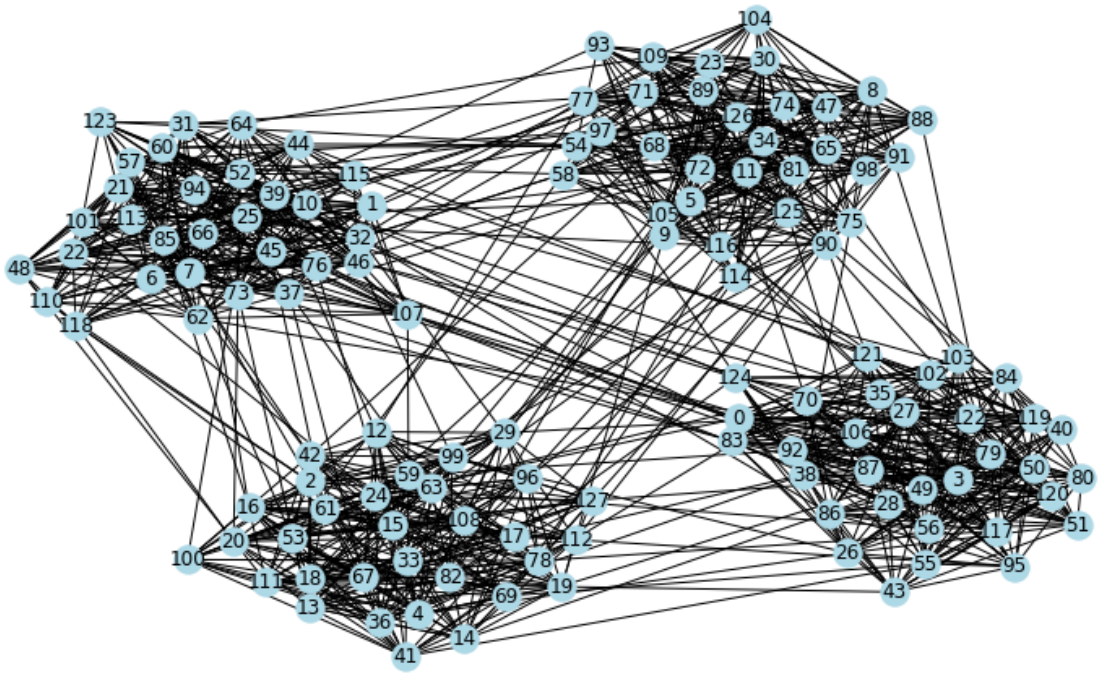
```
Modularidade da rede "Jazz": 0.4426 pelo método Louvain
```

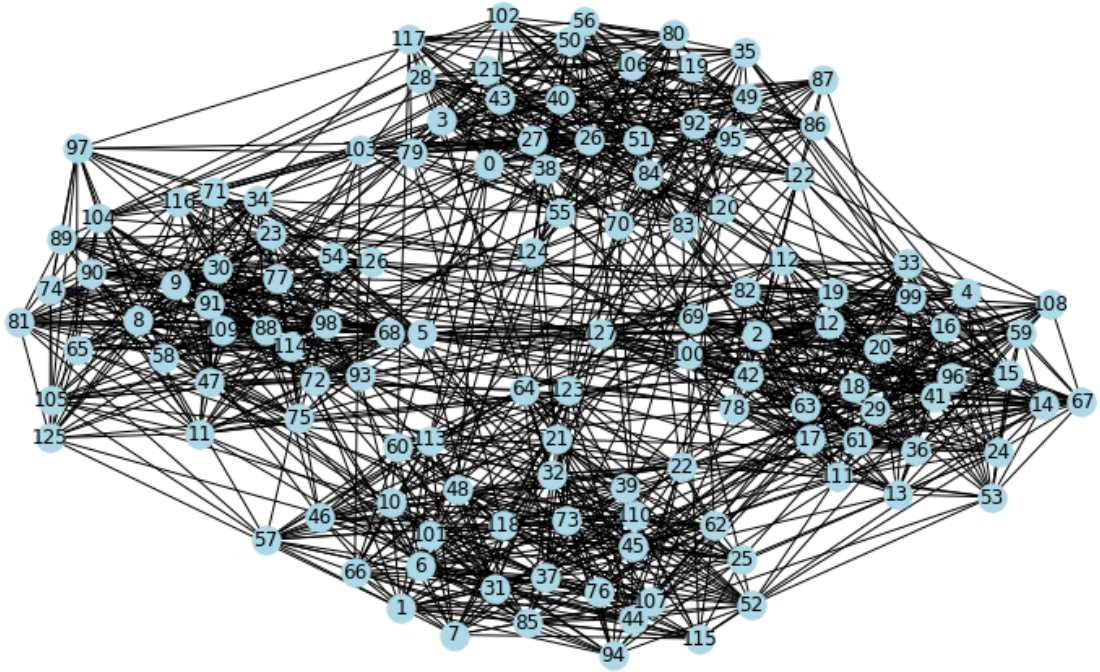


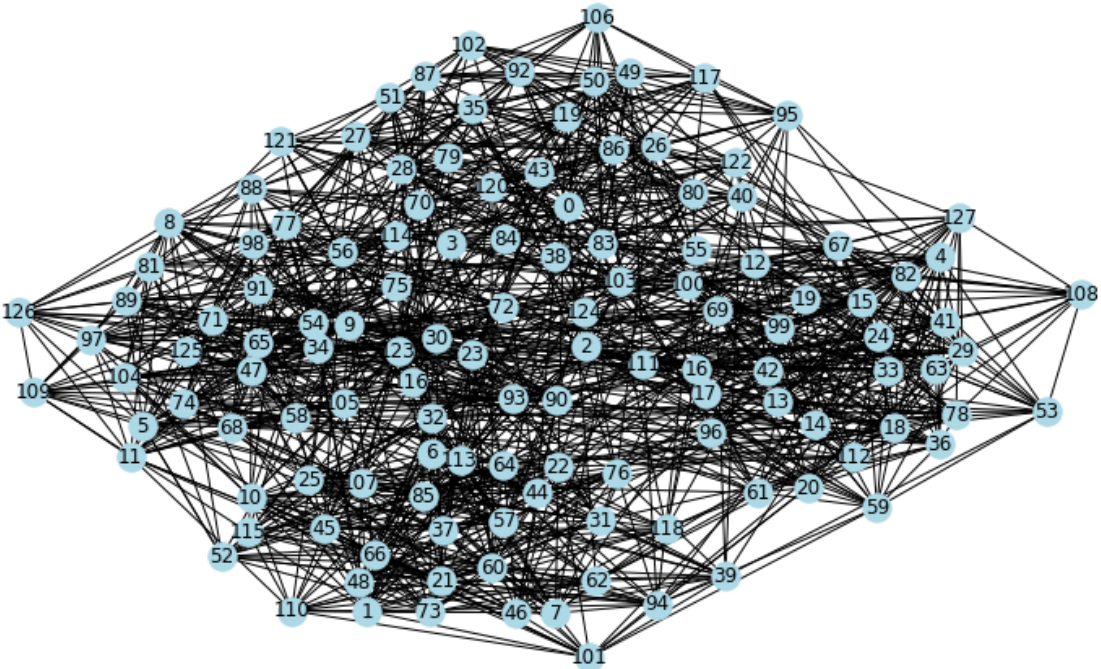
3 - Considere o método de geração de redes LFR_benchmark_graph. Obtenha os valores da modularidade para μ = 0.05, μ= 0.1 e μ= 0.2. Use o código a seguir para gerar as redes. Use o algoritmo de Louvain.

In [29]:

```
G3_1 = get_LFR(mu=0.05)
partition = community_louvain.best_partition(G3_1)
Q3_1 = modularity(G3_1, partition)

G3_2 = get_LFR(mu=0.1)
partition = community_louvain.best_partition(G3_2)
Q3_2 = modularity(G3_2, partition)

G3_3 = get_LFR(mu=0.2)
partition = community_louvain.best_partition(G3_3)
Q3_3 = modularity(G3_3, partition)

print("Modularidade das redes LFR pelo método de Louvain: {:.2f};{:.2f};{:.2f}".form
```





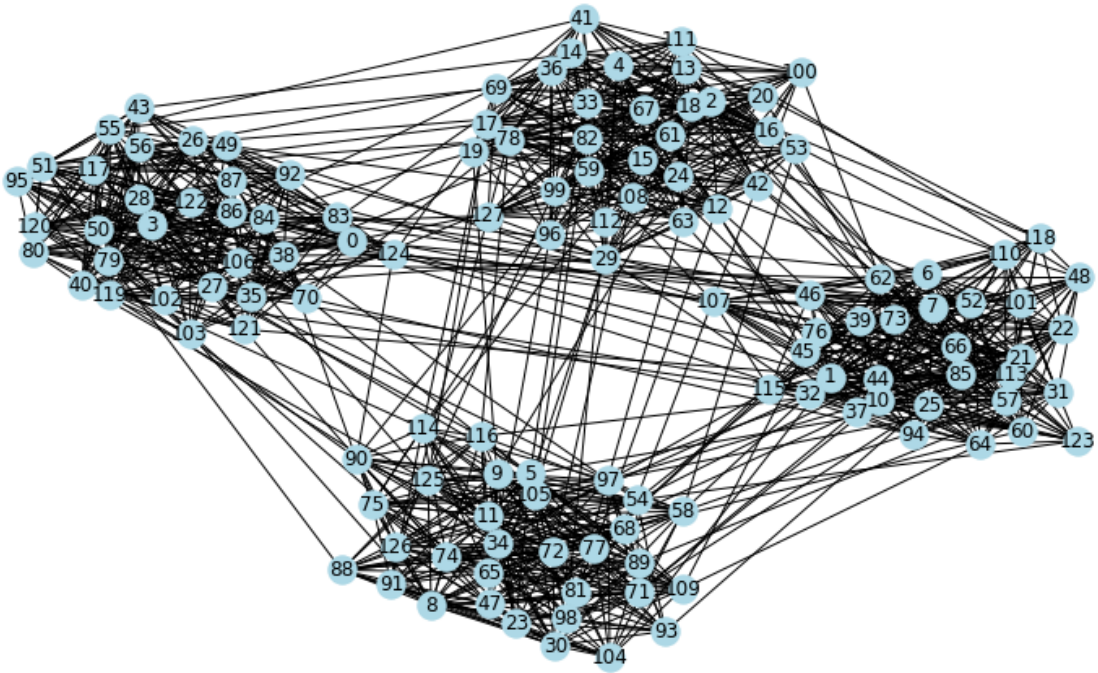


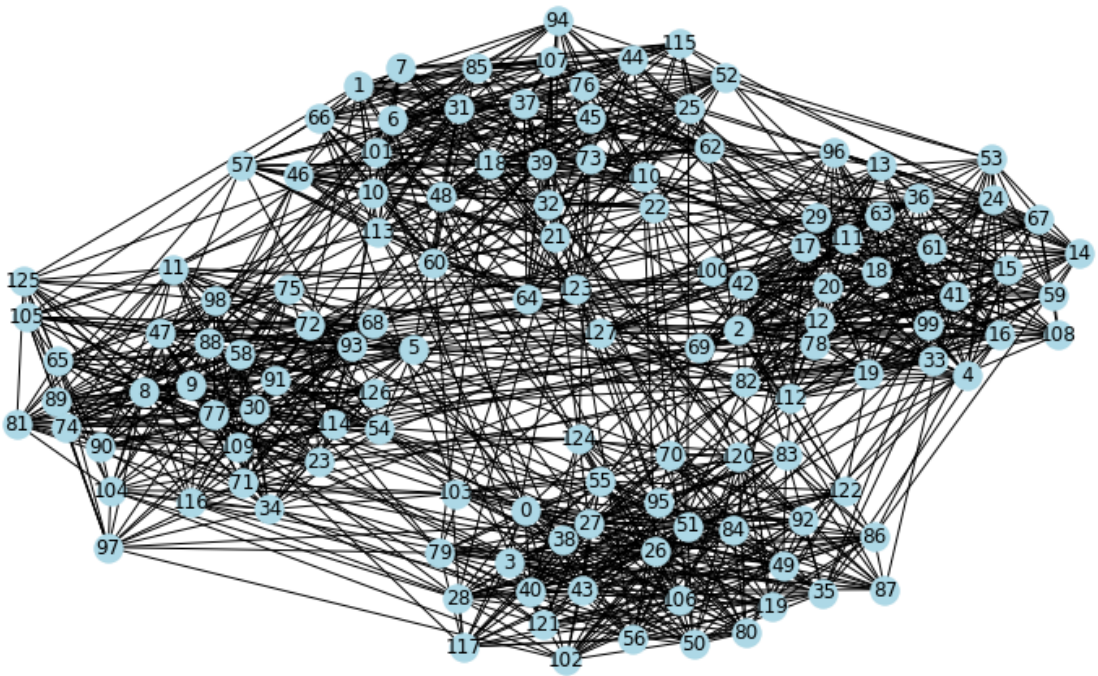
```
Modularidade das redes LFR pelo método de Louvain: 0.64;0.54;0.44
```

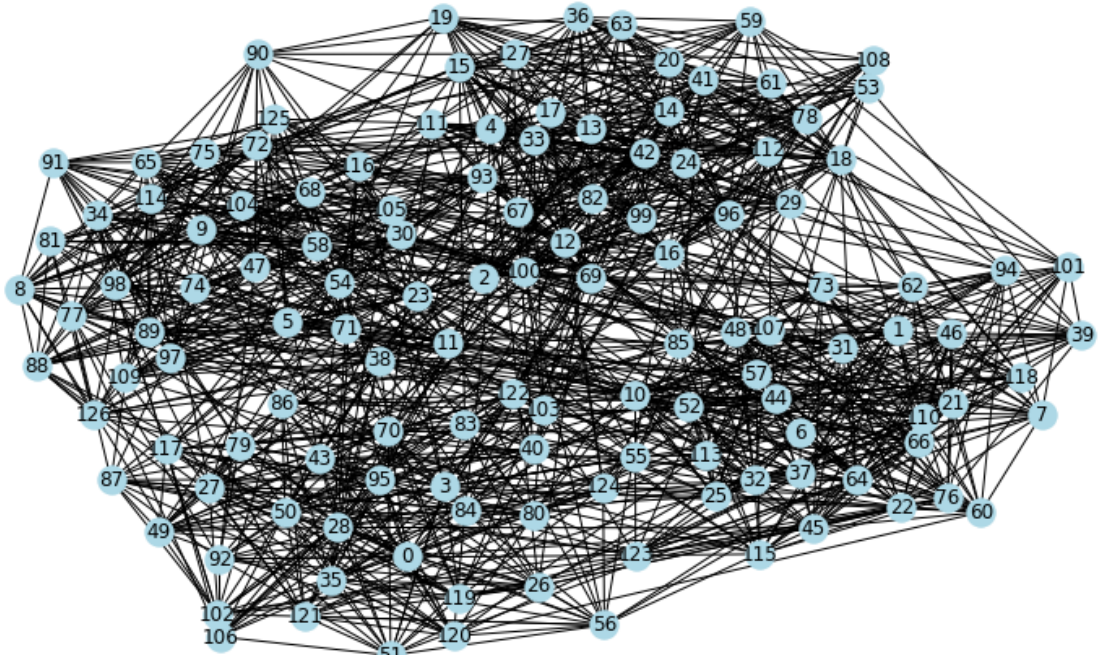
4 - Considere o método de geração de redes LFR_benchmark_graph. Obtenha os valores da modularidade para μ = 0.05, μ= 0.1 e μ= 0.2. Use o código a seguir para gerar as redes. Use o algoritmo fastgreedy.

In [30]:

```
G4_1 = get_LFR(mu=0.05)
communities = communities_fastgreedy(G4_1)
Q4_1 = modularity(G4_1, communities)

G4_2 = get_LFR(mu=0.1)
communities = communities_fastgreedy(G4_2)
Q4_2 = modularity(G4_2, communities)

G4_3 = get_LFR(mu=0.2)
communities = communities_fastgreedy(G4_3)
Q4_3 = modularity(G4_3, communities)

print("Modularidade das redes LFR pelo método Fast Greedy: {:.2f};{:.2f};{:.2f}".for
```







```
Modularidade das redes LFR pelo método Fast Greedy: 0.64;0.54;0.43
```