# lab 12

- Explorar os dados relacionados com migrações. fazendo interpretação de algumas das métricas.
- Junto são apresentadas algumas formas de clacular métricas relevantes.
- Consultar ainda: https://networkx.github.io/documentation/stable/ (https://networkx.github.io/documentation/stable/) e como complemento aos slides da UC : https://www.analyticsvidhya.com/blog/2018/04/introduction-to-graph-theory-network-analysis-python-codes/ (https://www.analyticsvidhya.com/blog/2018/04/introduction-to-graph-theory-network-analysis-python-codes/)

In [ ]:
```
# Bibliotecas a utilizar:
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
G = nx.DiGraph()
G.add_edge(a,b, weight= c)
for a,b,c in zip(df.origem, df.destino, df.numeros):
        G.add_edge(a,b, weight= c)
G.degree(weight='weight')
degree=G.degree()
dicionário com graus
degree_values = dict(degree).values()
G.nodes
G.edges
# Desenhar gráficos:
nx.draw(G, dim=100, with_labels=True)
#ou
plt.figure(figsize=(20, 20))
po1 = nx.spring_layout(G,scale=2)
nx.draw(G,  node_size=10, pos=po1, edge_color='y', label='r', with_labels=True)
plt.show()
#ou então alterar po1 para po1 = nx.random_layout(G)
degree =[deg for node, deg in nx.degree(G)]
max(degree)
min(degree)
kavg = np.mean(degrees)
print(kavg)
# Converter directed to undireted:
G1 = G.to_undirected()
Coeficiente de clustering
nx.clustering(G1)
# proximidade centralidade da rede
nx.closeness_centrality(G)
# eigenvector centralidade da rede
nx.eigenvector_centrality(G)
# grau de centralidade
nx.degree_centrality(G)
# número de componentes ligados
nx.number_connected_components(G1)
# obter o nó ligado a um componente
nx.node_connected_component(G1, 'João')
```