

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programme | : | **B.Tech** | Semester | : | **Win Sem 21-22** |
| Course | : | **Web Mining Lab** | Code | : | **CSE3024** |
| Faculty | : | **Dr.Bhuvaneswari A** | Slot | : | **L7+L8** |
| Date | : | **19-03-2022** | Marks | : | **10 Points** |

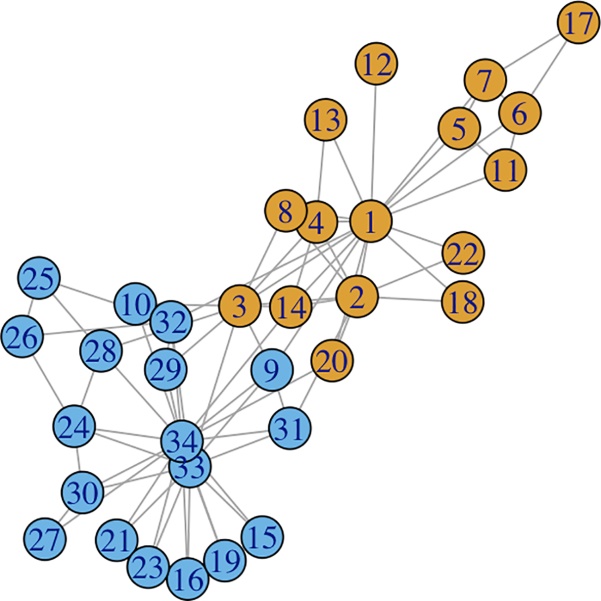
**GOOGLE COLAB LINK: https://colab.research.google.com/drive/12Z\_-XOeswT8Gg4LbZU-CgnXAoqlulIYr?usp=sharing**

**Exercise 10: GIRVAN-NEWMANALGORITHM**

* NetworkX package Implementation (5 Marks)
* Custom Implementation (5 Marks)

Implement the Girvan-Newman algorithm for the detection and analysis of community structure for the following network graph datasets (a) and (b).

1. Zachary karate club network dataset (refer folder A for networkX package implementation) 5 Marks



**CODE:**

import matplotlib.pyplot as plt

import networkx as nx

from networkx.algorithms.community.centrality import girvan\_newman

G = nx.karate\_club\_graph()

nx.draw(G, with\_labels=True)

ommunities = girvan\_newman(G)

node\_groups = []

for com innext(communities):

node\_groups.append(list(com))

for i inrange(len(node\_groups)):

print("Community",(i+1),":", node\_groups[i])

color\_map = []

for node in G:

if node in node\_groups[0]:

color\_map.append('orange')

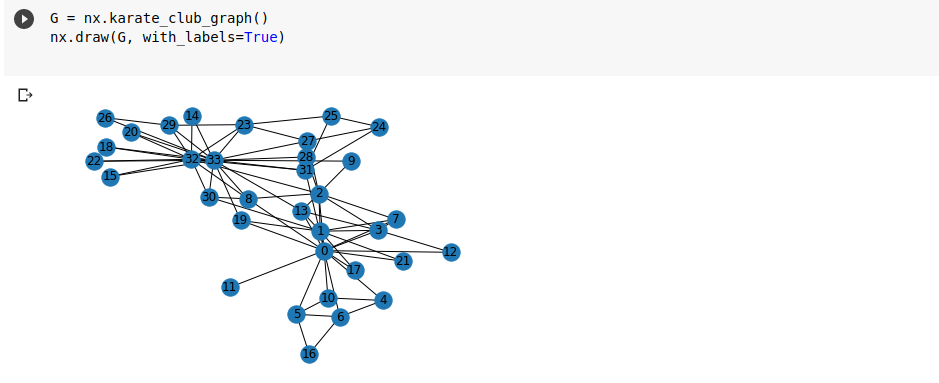
else:

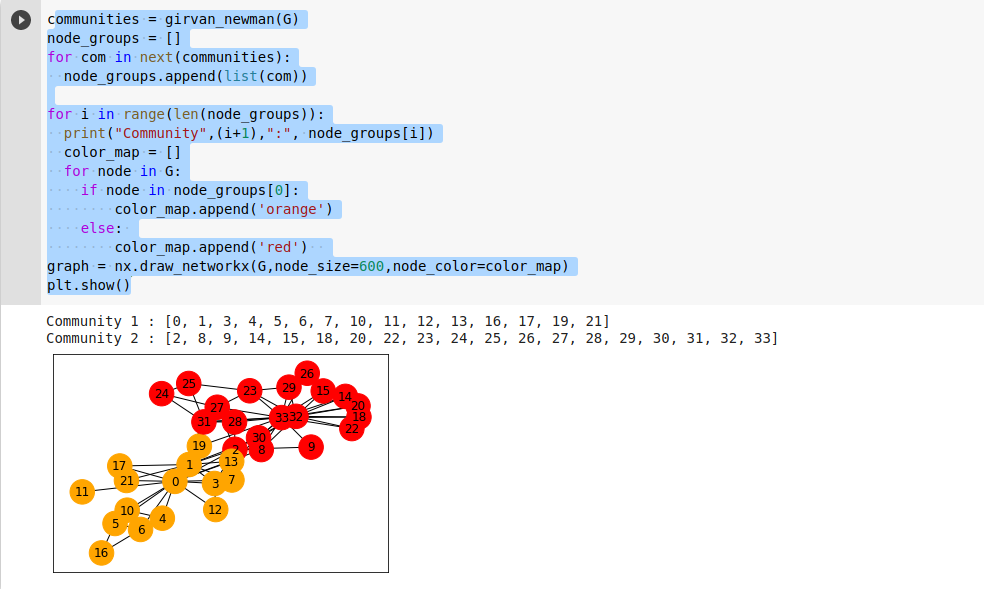
color\_map.append('red')

graph = nx.draw\_networkx(G,node\_size=600,node\_color=color\_map)

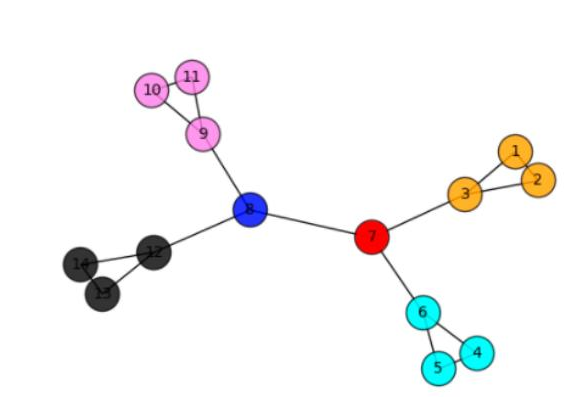
plt.show()

**OUTPUT:**





b. Facebook Friend Connection network dataset (refer folder B for custom implementation



**CODE:**

import community.community\_louvain as c

import networkx as nx

import matplotlib.pyplot as plt

import sys

import pylab

import copy

pylab.show()

defremoveEdges(G):

remove = []

b = nx.edge\_betweenness\_centrality(G)

max\_betweenness = b[max(b,key=b.get)]

for k,v in b.items():

if v==max\_betweenness:

remove.append(k)

G.remove\_edges\_from(remove)

graphs = list(nx.connected\_components(G))

d={}

counter = 0

for graph in graphs:

counter+=1

for node in graph:

d[node]=counter

if G.number\_of\_edges() == 0:

return [list(nx.connected\_components(G)),0,G]

m = c.modularity(d,G)

return [list(nx.connected\_components(G)),m,G]

if \_\_name\_\_=="\_\_main\_\_":

result\_communities=[]

G = nx.read\_edgelist("input.txt")

copyGraph = copy.deepcopy(G)

d={}

for node in G:

d[node]=0

initial\_modularity = c.modularity(d, G)

result\_communities.append([d,initial\_modularity,G])

while G.number\_of\_edges()>0:

subgraphs = removeEdges(G)

result\_communities.append(subgraphs)

G=subgraphs[-1]

for step in result\_communities:

if step[1]>initial\_modularity:

ng=step[0]

result=[]

modularity=step[1]

for graph in step[0]:

result.append(sorted([int(vertex) for vertex in graph]))

for community in result:

print(community)

d={};counter=0

for graph in ng:

for node in graph:

d[node] = counter

counter+=1

pos=nx.spring\_layout(copyGraph)

colors = ["violet","black","orange","cyan","red","blue","green","yellow","indigo","pink"]

for i inrange(len(ng)):

graph=ng[i]

nlist = [node for node in graph]

nx.draw\_networkx\_nodes(copyGraph,pos,nodelist=nlist,node\_color=colors[i%10],node\_size=500,alpha=0.8)

nx.draw\_networkx\_edges(copyGraph,pos)

nx.draw\_networkx\_labels(copyGraph,pos,font\_size=10)

plt.axis('off'

**OUTPUT:**

