import networkx as nx

import matplotlib.pyplot as plt

import random as r

import itertools as it

def disp(g):

edge\_lab=nx.get\_edge\_attributes(g,'sign')

pos=nx.circular\_layout(g)

nx.draw(g,pos,with\_labels=1,node\_color='yellow',node\_size=1000)

nx.draw\_networkx\_edge\_labels(g,pos,edge\_labels=edge\_lab,font\_size=20,font\_color='blue')

plt.show()

def disp\_comm(g):

f,s=coalition(g)

edge\_lab=nx.get\_edge\_attributes(g,'sign')

pos=nx.circular\_layout(g)

nx.draw(g,pos,with\_labels=1,nodelist=f,node\_color='yellow',node\_size=1000)

nx.draw(g,pos,with\_labels=1,nodelist=s,node\_color='red',node\_size=1000)

nx.draw\_networkx\_edge\_labels(g,pos,edge\_labels=edge\_lab,font\_size=20,font\_color='blue')

plt.show()

def list\_tri(g):

lst=list()

lst=[list(x) for x in it.combinations(g.nodes(),3)]

return lst

def sign\_tri(g):

lst=list\_tri(g)

s=list()

for ele in lst:

tmp=[list(x) for x in it.combinations(ele,2)]

temp=list()

for ele in tmp:

temp.extend(g[ele[0]][ele[1]]['sign'])

s.append(temp)

return s,lst

def unsatisfied(g):

s,lst=sign\_tri(g)

unhap=list()

for ele in range(len(s)):

cnt=s[ele].count('+')

if cnt!=3 and cnt!=1:

unhap.append(lst[ele])

return s,lst,unhap

def satisfy(g):

s,lst,unhap=unsatisfied(g)

count\_unhap=list()

while len(unhap)!=0:

count\_unhap.append(len(unhap))

choice=r.choice(unhap)

index=lst.index(choice) #Choose arbitary combination index

choose=r.randint(1,3) #Choose arbitary type of method

if s[index].count('+')==0:

if choose==1:

g[lst[index][0]][lst[index][1]]['sign']='+'

elif choose==2:

g[lst[index][1]][lst[index][2]]['sign']='+'

elif choose==3:

g[lst[index][0]][lst[index][2]]['sign']='+'

elif s[index].count('+')==2:

if choose==1:

if g[lst[index][0]][lst[index][1]]['sign']=='+':

g[lst[index][0]][lst[index][1]]['sign']='-'

else:

g[lst[index][0]][lst[index][1]]['sign']='+'

elif choose==2:

if g[lst[index][1]][lst[index][2]]['sign']=='+':

g[lst[index][1]][lst[index][2]]['sign']='-'

else:

g[lst[index][1]][lst[index][2]]['sign']='+'

elif choose==3:

if g[lst[index][0]][lst[index][2]]['sign']=='+':

g[lst[index][0]][lst[index][2]]['sign']='-'

else:

g[lst[index][0]][lst[index][2]]['sign']='+'

s,lst,unhap=unsatisfied(g)

return count\_unhap

def coalition(g):

first=list()

second=[]

nodes=[]

for each in g.nodes:

nodes.append(each)

chose=r.choice(nodes)

first.append(chose)

processed=[]

unprocessed=[chose]

for ele in unprocessed:

if ele not in processed:

neigh=g.neighbors(ele)

for var in neigh:

if g[ele][var]['sign']=='+':

if var not in first:

first.append(var)

if var not in unprocessed:

unprocessed.append(var)

elif g[ele][var]['sign']=='-':

if var not in second:

second.append(var)

processed.append(var)

processed.append(ele)

return first,second

n=8

g=nx.Graph()

d={1:'America',2:'Britain',3:'Russia',4:'France',5:'China',6:'India',7:'Pakistan',8:'Bangladesh'}

for ele in d.values():

g.add\_node(ele)

signs=['+','-']

list\_tri(g)

for ele in g.nodes():

for var in g.nodes():

if ele!=var:

g.add\_edge(ele,var,sign=r.choice(signs))

disp(g)

cu=satisfy(g)

disp(g)

plt.title("No. of unsatisfied nodes plot")

plt.xlabel("No. of such plots")

plt.ylabel("No. of unhappy nodes")

plt.bar([z for z in range(len(cu))],cu)

plt.show()

disp\_comm(g)