Task4

Ising model adapted to use a network for social connections can be conceptualized as follows:  
 Each node in the network represents an individual, and their associated "spin" or opinion is analogous to their magnetic alignment in the traditional Ising model. However, instead of a regular lattice or grid structure, the Ising model operates on a network where nodes are connected by edges representing social ties or interactions.

The energy function of the Ising model on a network is modified to consider interactions between connected nodes:

Here, the sum is taken over pairs of neighboring nodes connected by an edge, denoted by . E represents the energy of the system, is the coupling constant determining the strength of interaction, and and are the opinions (spins) of nodes and respectively.

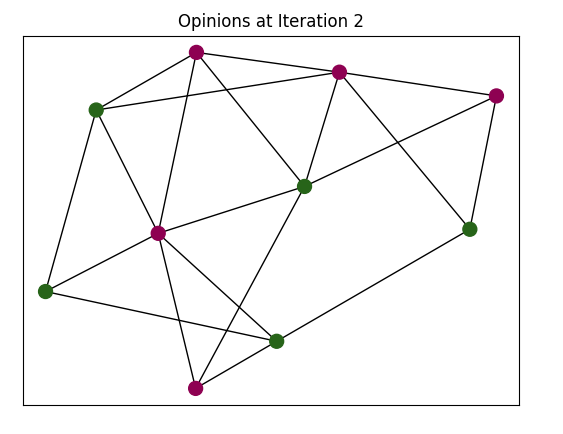
The evolution of the Ising model on a network involves iteratively updating the opinions of nodes based on their interactions with neighboring nodes, similar to the traditional Ising model. However, instead of considering adjacent nodes in a grid, interactions are now determined by the network structure, where nodes interact with their connected neighbors.

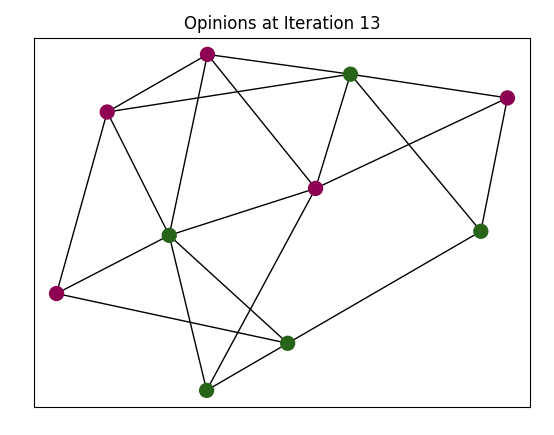
Network evolving refers to the dynamic changes in the structure of the social network over time. This includes the formation of new social connections, the dissolution of existing connections, and the rewiring of edges, reflecting changes in social interactions among individuals. These changes in the network structure can influence the spread and evolution of opinions among individuals.

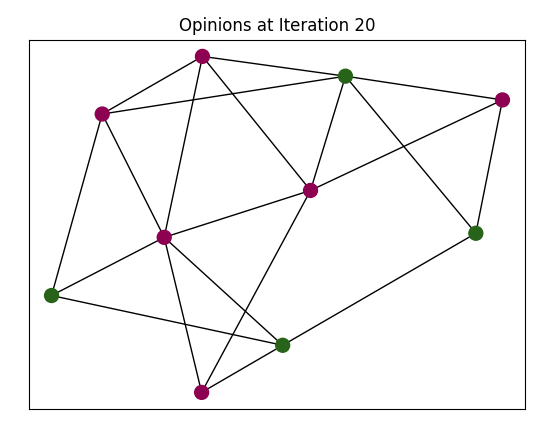
Mean opinion in the context of a social network represents the average opinion value across all individuals in the network at a given time. It provides insights into the overall sentiment or consensus within the network and how it evolves over time. By plotting the mean opinion over time, we can observe trends in opinion formation, convergence, or divergence within the social network.

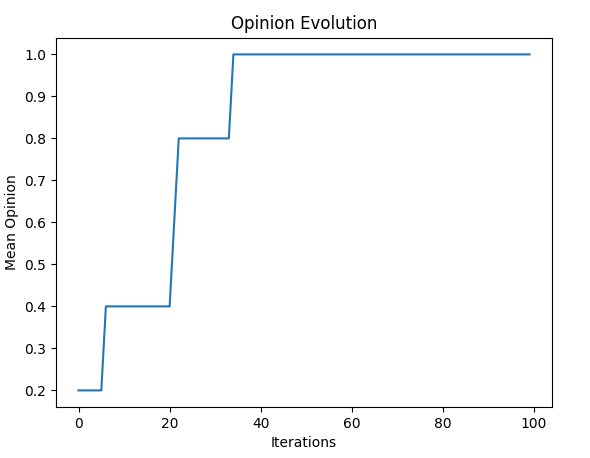
In summary, the lsing model on a network captures the dynamics of opinion formation and evolution within a social network, where interactions between individuals are represented by the network structure, and the evolution of opinions is influenced by changes in social connections over time.

python assignment.py -ising\_model -use\_network 10

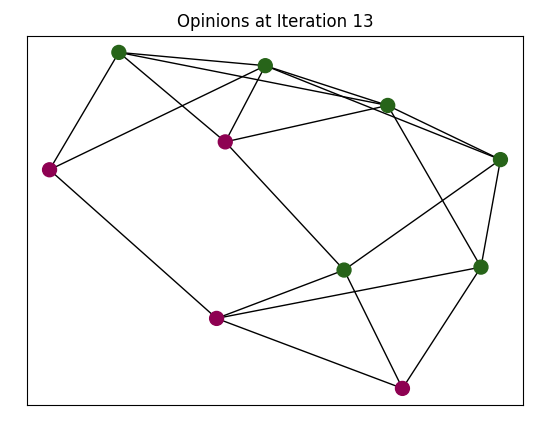
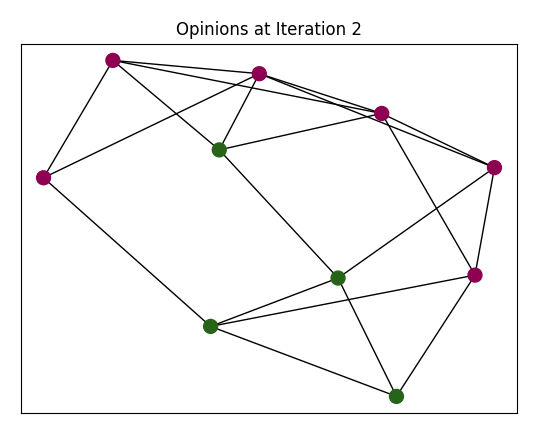


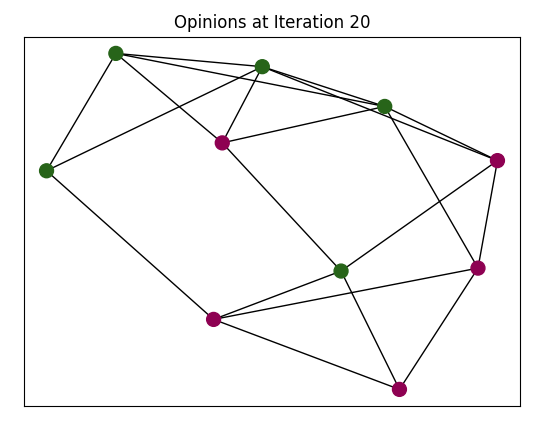


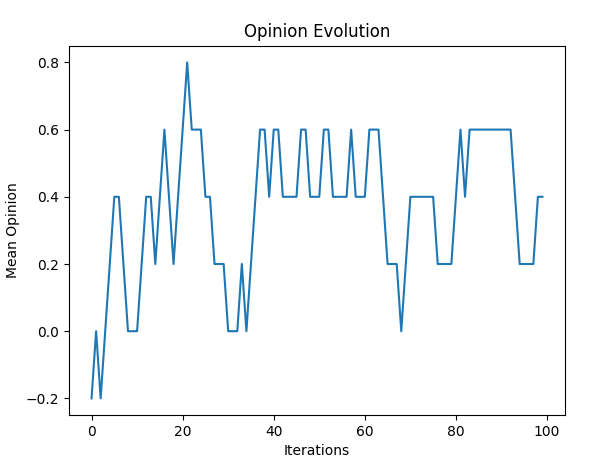




python assignment.py -defuant -use\_network 10







PYTHON CODE  
*#=======================================*import numpy as np  
import matplotlib.pyplot as plt  
import networkx as nx  
import argparse  
from matplotlib.animation import FuncAnimation  
def initialize\_opinions(num\_nodes):  
 return np.random.choice([-1, 1], size=num\_nodes)  
def simulate\_ising\_model(network, num\_iterations):  
 num\_nodes = network.number\_of\_nodes()  
 opinions = initialize\_opinions(num\_nodes)  
 mean\_opinion = []  
 for \_ in range(num\_iterations):  
 mean\_opinion.append(np.mean(opinions))  
 node = np.random.choice(list(network.nodes()))  
 neighbors = list(network.neighbors(node))  
 total\_opinion = opinions[node]  
 for neighbor in neighbors:  
 total\_opinion += opinions[neighbor]  
 average\_opinion = total\_opinion / (len(neighbors) + 1)  
 if average\_opinion > 0:  
 opinions[node] = 1  
 elif average\_opinion < 0:  
 opinions[node] = -1  
 return opinions, mean\_opinion  
def simulate\_defuant\_model(network, num\_iterations, threshold):  
 num\_nodes = network.number\_of\_nodes()  
 opinions = initialize\_opinions(num\_nodes)  
 mean\_opinion = []  
 for \_ in range(num\_iterations):  
 mean\_opinion.append(np.mean(opinions))  
 node = np.random.choice(list(network.nodes()))  
 neighbors = list(network.neighbors(node))  
 if len(neighbors) > 0:  
 neighbor\_indices = np.random.choice(neighbors, size=int(len(neighbors) / 2), replace=False)  
 neighbor\_opinions = opinions[neighbor\_indices]  
 disagreement\_count = np.sum(np.abs(neighbor\_opinions - opinions[node]) > threshold)  
 disagreement\_fraction = disagreement\_count / len(neighbor\_indices)  
  
 if disagreement\_fraction > 0:  
 new\_opinion = np.mean(neighbor\_opinions[np.abs(neighbor\_opinions - opinions[node]) > threshold])  
 opinions[node] = new\_opinion  
 return opinions, mean\_opinion  
def plot\_opinion\_evolution(mean\_opinion):  
 plt.plot(mean\_opinion)  
 plt.xlabel(**'Iterations'**)  
 plt.ylabel(**'Mean Opinion'**)  
 plt.title(**'Opinion Evolution'**)  
 plt.show()  
def animate\_opinions(opinions, network):  
 fig, ax = plt.subplots()  
 pos = nx.spring\_layout(network)  
 nodes = nx.draw\_networkx\_nodes(network, pos, node\_color=opinions, cmap=plt.cm.PiYG, node\_size=100)  
 edges = nx.draw\_networkx\_edges(network, pos)  
 def update(i):  
 ax.clear()  
 new\_opinions, \_ = simulate\_ising\_model(network, 1)  
 nodes.set\_array(new\_opinions)  
 nx.draw\_networkx\_nodes(network, pos, node\_color=new\_opinions, cmap=plt.cm.PiYG, node\_size=100, ax=ax)  
 nx.draw\_networkx\_edges(network, pos, ax=ax)  
 ax.set\_title(**f'Opinions at Iteration** {i + 1}**'**)  
 anim = FuncAnimation(fig, update, frames=range(20), interval=500)  
 plt.show()  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 parser = argparse.ArgumentParser()  
 parser.add\_argument(**'-ising\_model'**, action=**'store\_true'**, help=**'Run Ising Model'**)  
 parser.add\_argument(**'-defuant'**, action=**'store\_true'**, help=**'Run Defuant Model'**)  
 parser.add\_argument(**'-use\_network'**, type=int, help=**'Size of the network'**)  
 args = parser.parse\_args()  
 if args.use\_network:  
 network\_size = args.use\_network  
 if network\_size > 0:  
 network = nx.connected\_watts\_strogatz\_graph(network\_size, 4, 0.2)  
 else:  
 network = nx.random\_regular\_graph(4, abs(network\_size))  
 else:  
 *# Default to using a grid array* network\_size = 10  
 network = nx.grid\_2d\_graph(network\_size, network\_size)  
 if args.ising\_model:  
 opinions, mean\_opinion = simulate\_ising\_model(network, 100)  
 animate\_opinions(opinions, network)  
 plot\_opinion\_evolution(mean\_opinion)  
 elif args.defuant:  
 opinions, mean\_opinion = simulate\_defuant\_model(network, 100, 0.5)  
 animate\_opinions(opinions, network)  
 plot\_opinion\_evolution(mean\_opinion)  
*##python assignment.py -ising\_model -use\_network 10  
##python assignment.py -defuant -use\_network 10*