CMSE 201 Project

Name : Student

Section : XXX

Research Question

"How does a virus like covid spread in a classroom over a period of time? How can different seating arrangements affect the spread?"

Resources

To answer my question, I will be using an Agent-based model to create models of classrooms with a seating capacity of 100(10x10) with three different seating arrangements. Then using ABM, I plan to track the spread of the virus in every setting and evaluate which of them is a good seating arrangement.

Background and Motivation

The last 2 years have been hard for all mankind but I feel that students have been some of the people that suffered the most. While moms and dads have welcomed the privilege of working from home during the pandemic, students have dreaded online learning and longed for their classes to be in-person. I have drawn my motivation for the project from here and hope that my study can help explore different ways of seating students in classrooms to mitigate the risks associated with having in-person classes during a pandemic.

Methodology

I plan to create models of 3 different types of seating arrangements in a classroom with a capacity of 100 (10x10) and then track the spread of virus using ABM. Following is the classroom settings I will work with:

1. A fully occupied classroom
2. Students seated diagonally like a single color on a chessboard (50% capacity)
3. Students seated in alternate rows (50% capacity)

To model the spread of the virus, we will work with some assumptions/rules:

1. A student sitting in close proximity to an infected student gets exposed regardless of their relative position.
2. A student exposed to the virus becomes infectious the next day (unit of time for this study).
3. Once the school identifies an infectious student, it asks them to stop coming to class and take rest.
4. Students can report back to class two weeks after they were found to be infected.
5. 6% of randomly selected students are affected on the first day in consideration.
6. For now, we are assuming that the virus is mostly non life threatening and mortality rate is negligible.

Modeling and Visualization

Import Relevant Libraries

In [23]:
```
import numpy as np
import random
%matplotlib inline
import matplotlib.pyplot as plt

from IPython.display import display, clear_output
import time
```

DEFINITION OF FUNCTIONS

Functions for modeling the classroom as a 2D array for 3 different seating arrangements

In [24]:
```
# A fully filled classroom

def set_board_full(board_size=10):

    # A value of 1 represents healthy students
    class_board = np.ones((board_size,board_size),dtype='int64')

    # randomly set 6 students (6%) in class to be infected
    count = 0
    infected = []
    while count != 6:
        i = random.choice(range(0,9))
        j = random.choice(range(0,9))
        if [i,j] not in infected:
            class_board[i,j] = 2
            infected.append([i,j])
            count += 1

    return class_board


# A diagonally filled classroom

def set_board_diagonal(board_size=10,infected_percent=0.05):

    class_board = np.zeros((board_size,board_size),dtype='int64')

    # Seating students diagonally
    for i in range(board_size):
        for j in range(board_size):
            if i%2==0 and j%2==0:
                class_board[i,j] = 1
            elif i%2!=0 and j%2!=0:
                class_board[i,j] = 1

    # Set 6 % of students(3) to be infected
    count = 0
    infected = []
    while count != 3:
        i = random.choice(range(0,9))
        j = random.choice(range(0,9))
        if [i,j] not in infected and class_board[i,j] == 1:
            class_board[i,j] = 2
            infected.append([i,j])
            count += 1

    return class_board


# A classroom filled by alternate rows

def set_board_alternate(board_size=10,infected_percent=0.05):

    class_board = np.zeros((board_size,board_size),dtype='int64')

    # Seating students in alternate rows
    for i in range(board_size):
        for j in range(board_size):
            if i%2==0:
                class_board[i,j] = 1

    # Set 6 % of students(3) to be infected
    count = 0
    infected = []
    while count != 3:
        i = random.choice(range(0,9))
        j = random.choice(range(0,9))
        if [i,j] not in infected and class_board[i,j] == 1:
            class_board[i,j] = 2
            infected.append([i,j])
            count += 1

    return class_board
```

Functions for checking if a particular seat in the classroom exists (if a cell is within the 2D array) and getting all the seats that are in close proximity to a seat

In [25]:
```
def onBoard(i, j, cell):
    ni = cell.shape[0]
    nj = cell.shape[1]
    if i < ni and i >= 0:
        if j < nj and j >= 0:
            return True
        else:
            return False
    else:
        return False


def getNeighborValues(i,j, board):

    neighborhood = [(i-1, j), (i, j-1), (i+1, j), (i, j+1)]

    neighbor_values = []
    for neighbor in neighborhood:
        if(onBoard(neighbor[0],neighbor[1],board)==True):
            neighbor_values.append(board[neighbor[0]][neighbor[1]])

    return neighbor_values
```

Function for updating the state of the classroom everyday

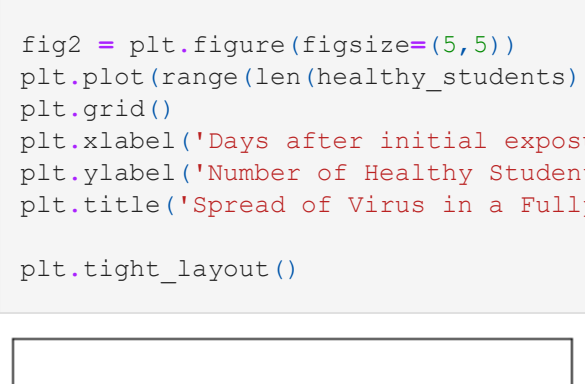
In [26]:
```
def advance_board(class_board, isolated_students, isolated_days):

    new_board = np.zeros_like(class_board)

    for i in range(class_board.shape[0]):
        for j in range(class_board.shape[1]):

            if class_board[i,j] == 2:
                new_board[i,j] = 0
                isolated_students.append([i,j])
                isolated_days.append(0)

            if class_board[i,j] == 1:
                new_board[i,j] = 1
                check = getNeighborValues(i,j,class_board)
                if 2 in check:
                    new_board[i,j] = 2

    for i in range(0,len(isolated_days)):
        isolated_days[i] += 1
        if isolated_days[i] == 14: # A student can return to class 14 days after showing up to class infected
            ind = isolated_students[i]
            isolated_days[i] = -1
            new_board[ind[0],ind[1]] = 1

    temp_1 = []
    temp_2 = []

    for i in range(0,len(isolated_days)):
        if isolated_days[i] != -1:
            temp_1.append(isolated_students[i])
            temp_2.append(isolated_days[i])

    isolated_students = temp_1
    isolated_days = temp_2

    return new_board, isolated_students, isolated_days
```

Function for plotting the classroom as a figure

In [27]:
```
def plotgrid(array):

    #
    x_range = np.linspace(0, array.shape[1]-1, array.shape[1])
    y_range = np.linspace(0, array.shape[0]-1, array.shape[0]) # repeat for the y/vertical axis

    #
    x_indices, y_indices = np.meshgrid(x_range, y_range)

    #
    healthy_x = x_indices[array == 1];
    healthy_y = y_indices[array == 1];
    infected_x = x_indices[array == 2];
    infected_y = y_indices[array == 2];

    #
    plt.plot(healthy_x, array.shape[0] - healthy_y - 1, 'bs', markersize=10) # reverses direction of y-axis
    plt.plot(infected_x, array.shape[0] - infected_y - 1, 'rs', markersize=10)  # repeat for infected students

    #
    plt.xlim([-1,array.shape[1]])
    plt.ylim([-1,array.shape[0]])

    #
    plt.tick_params(axis='both', which='both',
                    bottom=False, top=False, left=False, right=False,
                    labelbottom=False, labelleft=False)
```

IMPLEMENTATION OF MODEL FOR FULLY FILLED CLASSROOM

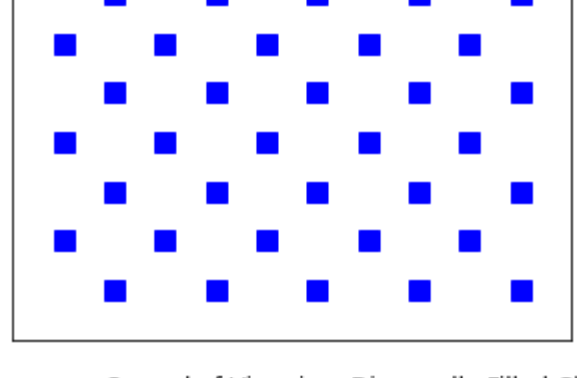
In [28]:
```
fig = plt.figure(figsize=(5,5))

_size = 10

class_model = set_board_full(board_size=_size)

plotgrid(class_model)
time.sleep(0.1)
clear_output(wait=True)
display(fig)
fig.clear()

healthy_students = [100]
isolated_students = []
isolated_days = []

# tracking spread of virus for a month
for i in range(0,29):

    class_model,isolated_students,isolated_days = advance_board(class_model,isolated_students,isolated_days)
    healthy_students.append(100-len(isolated_students))

    # creating the animation of the spread of virus
    plotgrid(class_model)
    time.sleep(0.1)
    clear_output(wait=True)
    display(fig)
    fig.clear()

    if healthy_students[-1] == 0:
        break

# exit the animation
plt.close()

fig2 = plt.figure(figsize=(5,5))
plt.plot(range(len(healthy_students)),healthy_students)
plt.grid()
plt.xlabel('Days after initial exposure')
plt.ylabel('Number of Healthy Students')
plt.title('Spread of Virus in a Fully Packed Classroom')

plt.tight_layout()
```

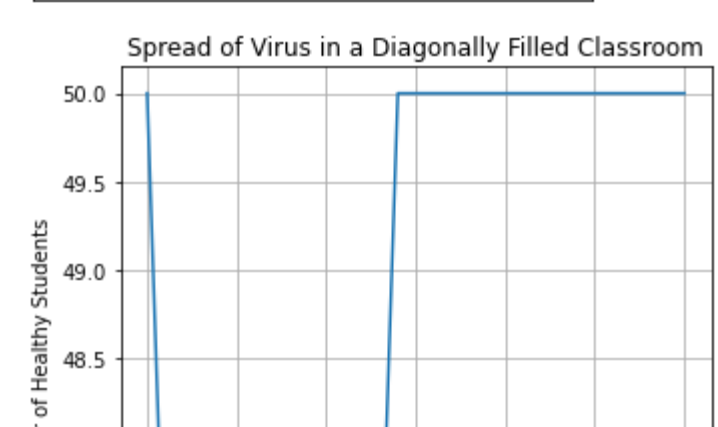


The model successfully runs. It shows that in a fully filled classroom with a capacity of seating 100 students where students are in close proximity, all students catch the virus in only about a week when just 6% of the students are initial carriers of the virus.

IMPLEMENTATION OF MODEL FOR DIAGONALLY FILLED CLASSROOM

In [29]:
```
fig = plt.figure(figsize=(5,5))

_size = 10

class_model = set_board_diagonal(board_size=_size)

plotgrid(class_model)
time.sleep(0.1)
clear_output(wait=True)
display(fig)
fig.clear()

healthy_students = [50]
isolated_students = []
isolated_days = []

# tracking spread of virus for a month
for i in range(0,30):

    class_model,isolated_students,isolated_days = advance_board(class_model,isolated_students,isolated_days)
    healthy_students.append(50-len(isolated_students))

    # creating the animation of the spread of virus
    plotgrid(class_model)
    time.sleep(0.1)
    clear_output(wait=True)
    display(fig)
    fig.clear()

    if healthy_students[-1] == 0:
        break

# exit the animation
plt.close()

fig2 = plt.figure(figsize=(5,5))
plt.plot(range(len(healthy_students)),healthy_students)
plt.grid()
plt.xlabel('Days after initial exposure')
plt.ylabel('Number of Healthy Students')
plt.title('Spread of Virus in a Diagonally Filled Classroom')

plt.tight_layout()
```

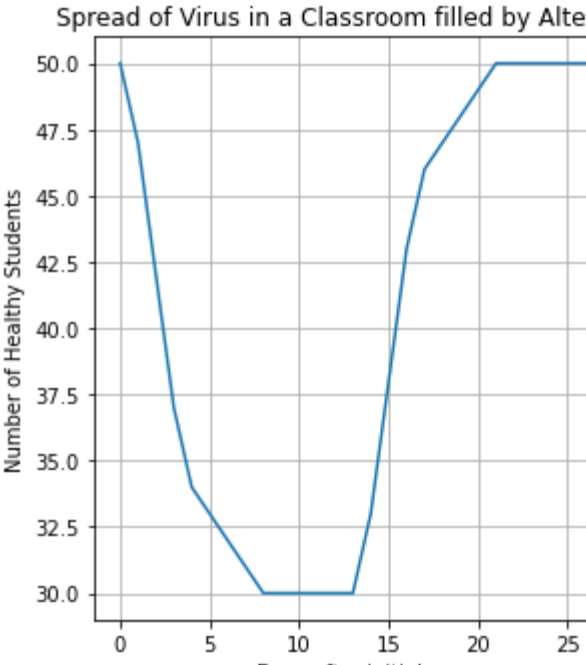


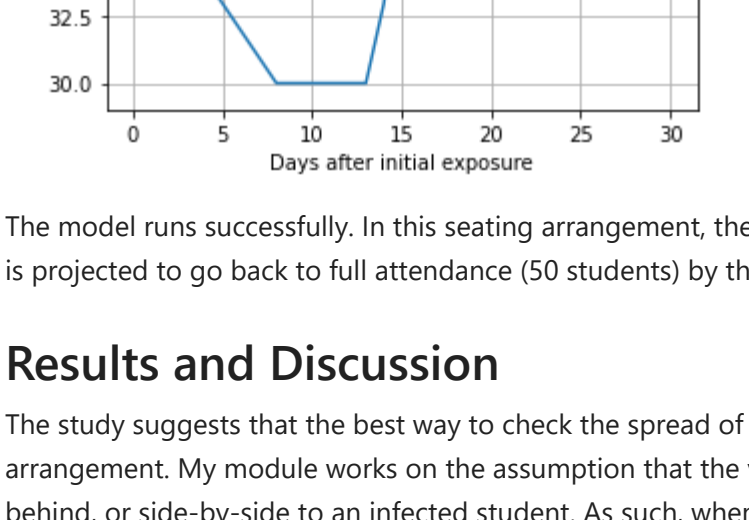


The model runs successfully and shows that seating students diagonally can check the spread of virus at the roots with a transmission rate of 0. Out of 50 students, nobody else apart from the initially affected 6% (3 students) catches the virus and the number of healthy students falls to 47 after the first day but goes back to 50 on the 14th day.

IMPLEMENTATION OF MODEL FOR CLASSROOM FILLED BY ALTERNATE ROWS

In [31]:
```
fig = plt.figure(figsize=(5,5))

_size = 10

class_model = set_board_alternate(board_size=_size)

plotgrid(class_model)
time.sleep(0.1)
clear_output(wait=True)
display(fig)
fig.clear()

healthy_students = [50]
isolated_students = []
isolated_days = []

# tracking spread of virus for a month
for i in range(0,30):

    class_model,isolated_students,isolated_days = advance_board(class_model,isolated_students,isolated_days)
    healthy_students.append(50-len(isolated_students))

    # creating the animation of the spread of virus
    plotgrid(class_model)
    time.sleep(0.1)
    clear_output(wait=True)
    display(fig)
    fig.clear()

    if healthy_students[-1] == 0:
        break

# exit the animation
plt.close()

fig2 = plt.figure(figsize=(5,5))
plt.plot(range(len(healthy_students)),healthy_students)
plt.grid()
plt.xlabel('Days after initial exposure')
plt.ylabel('Number of Healthy Students')
plt.title('Spread of Virus in a Classroom filled by Alternate Rows')

plt.tight_layout()
```





The model runs successfully. In this seating arrangement, the number of healthy students falls from 50 to 20 by the 9th day but by the class is projected to go back to full attendance (50 students) by the 23rd day.

Results and Discussion

The study suggests that the best way to check the spread of virus in a classroom is by seating half the capacity of students in a diagonal arrangement. My module works on the assumption that the virus spreads only in very close proximity, i.e., when a student is in front, behind, or side-by-side to an infected student. As such, when the students are seated diagonally, it makes it very hard for the virus to affect any other student in the class. Now the model is obviously a very simplified version of a classroom setting where students move around all the time, exchange greetings involving physical contact, and like to sit close to their friends, etc. But I believe it still holds the potential to suggest the best possible to seat students in a classroom, check spread of germs, and sustain in-person learning.

I feel the best thing about this project is that the code in the notebook can be modified / expanded upon to not only depict classrooms with different seating capacities but to also model various other seating arrangements. Moreover, we can also change which of the neighbors are exposed. For example, we can set the code to assume that the students seated diagonally to the infected student are also exposed, or that the student seated directly behind an infected student is not exposed. With this observation, I would like to bring the study to a conclusion.

THE END