

ביולוגיה חישובית

80-512 89-512

תרגיל 1 – אוטומטים תאיים בימי הקורונה

מגישים (הגשה משותפת סטודנטים מביולוגיה חישובית ומדעי המחשב):

אייל הבר - (ביולוגיה חישובית)

אייל ברילינג - (מדעי המחשב)

קישור להורדת קובץ ההפעלה והקבצים הדרושים להרצה:

<https://github.com/EyalBrilling/Corona-waves-simulator>

התרגיל רץ על ווינדוס

הקדמה

בתרגיל התבקשנו לסמלץ התפשטות של מחלת הקורונה בקרב אוכלוסייה. המרחב שבו האוכלוסייה נמצאת הינו מטריצה 200×200 , כאשר כל פריט באוכלוסייה יכול לאחסן אחת ממשבצות המטריצה, ללא חפיפות. בנוסף לכל פריט באוכלוסייה יש 3 מצבים אפשריים: **בריא**, **חולה**, **חסין**. בתחילת הסימולציה אחוז מסוים של פריטים מהאוכלוסייה נמצא במצב חולה והשאר במצב בריא. כאשר פריט חולה נמצא בסמיכות / שכנות לפריט בריא, קיימת הסתברות מסוימת P בה הפריט הבריא יהפוך לחולה. לאחר מספר ימי חולי (המוגדרים על ידי המשתמש כ- X), הפריט הופך לחסין. ואינו יכול להידבק יותר שוב.

מבנה הסימולציה

בתחילת הסימולציה מבקשים מהמשתמש להכניס מספר פרמטרים המשפיעים על תהליך הסימולציה. N – מספר היצורים, D – אחוז החולים הראשוני באוכלוסייה, R – אחוז היצורים שנעים מהר, X – מספר דורות עד להחלמה

הסימולציה עצמה פועלת באופן הבא:

- (1) יוצרים רשימה של יצורים על פי הכמות המבוקשת על ידי המשתמש
- (2) יוצרים את הגריד שעליו היצורים מסתובבים
- (3) מציירים את הגריד שיוצג ויזואלית למשתמש
- (4) בלולאה עד אשר אין עוד יצורים חולים:
 - (א) מדביקים את היצורים
 - (ב) מזיזים את היצורים.
 - (ג) מציירים את הגריד מחדש על פי המצב החדש לאחר ההדבקה והתזוזה

יצירת היצורים:

יצרנו מחלקה חדשה בשם creature על מנת לייצג אובייקט של יצור:

```
# Creature Constructor
class Creature:
    def __init__(self, health_status, x_Coordinate, y_Coordinate, speed, remaining_sick_days=X):
        # in the start can be HEALTHY(=1) or SICK(=2)
        self.health_status = health_status
        self.x_Coordinate = x_Coordinate
        self.y_Coordinate = y_Coordinate
        self.speed = speed # normal or fast(=10 steps per frame)
        self.remaining_sick_days = remaining_sick_days
```

כאשר אנו יוצרים את רשימת היצורים, אנו מגדילים מיקומים על הגרید תוך כדי דאגה שאין שני יצורים על אותו מיקום:

```
# random coordinates without same indexes
randomCoordinatesIndexes = sample(list(product(range(200), repeat=2)), k=N) # create random unique tuples
```

הפונקציה sample מחזירה רשימה בגודל N המכילה מיקומי x,y בצורה כזאת שאין שני מיקומים זהים.

בנוסף אנו מגדילים את שאר התכונות ההתחלתיות של היצורים

(מצב בריאותי: חולה או בריא ומהירות: רגיל - זז משבצת אחת או מהר - זז 10 משבצות)

תכונות אלו יתפזרו בין היצורים בהתאם לפרמטרים שהוכנסו על ידי המשתמש שגם הגדיר את מספר ימי החולי X

(מספר דורות שיעברו עד שיצור חולה הופך לחסין)

הדבקה

בשלב ראשון - אנו עוברים על כל היצורים ומחלקים אותם לשתי קבוצות: חולים ובריאים

בשלב השני -

```
# check neighbors of sick creatures
for location in sick_locations:
    x, y = location[0], location[1]
    neighbors = [(x, y + 1), (x, y - 1), (x + 1, y), (x - 1, y), (x - 1, y - 1), (x - 1, y + 1), (x + 1, y - 1),
                 (x + 1, y + 1)]
    for neighbor in neighbors:
        if neighbor in healthy_locations and random.random() < P:
            healthy_locations[neighbor].health_status = SICK # this neighbor is infected (SICK == 2)
```

עבור כל יצור חולה, נגדיר את המיקומים המקיפים אותו ("שכנים").

עבור כל מיקום - נבדוק האם נמצא בו יצור בריא. אם כן, אזי בהסתברות מסוימת P הוא יחלה.

בצורה כזאת, אם קיים יצור בריא שמקיפים אותו כמה יצורים חולים - יש לו כמה הזדמנויות להידבק.

הערה - ההסתברות להידבק P תלויה באחוז החולים הנוכחי D באוכלוסיה ביחס לסף מוגדר T ,

כאשר $D > T$ - היצורים מקיפים על הנחיות הקורונה וההסתברות להדבקה קטנה: $P = P_LOW$

כאשר $D \leq T$ - היצורים פחות מקיפים על הנחיות הקורונה וההסתברות להדבקה גדלה: $P = P_HIGH$

תזוזה

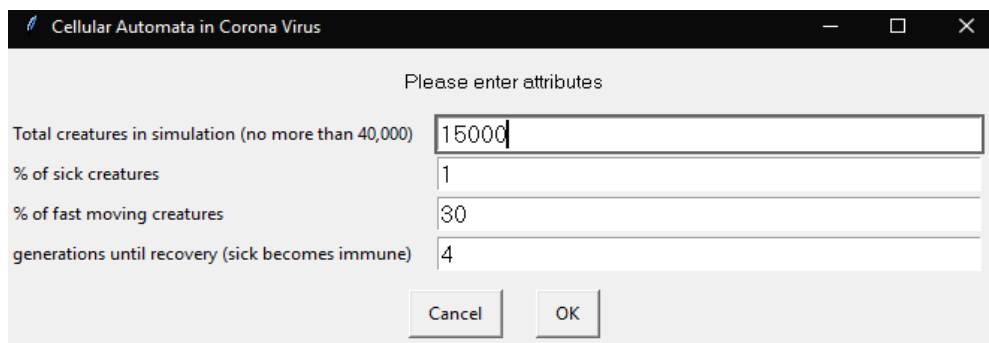
```
if random.random() < (1 / 9): # stay in (x,y) with some % . go to next creature if chosen
    continue
# check other movement options
random.shuffle(creature_locations[location]) # randomize the neighbors
for neighbor in creature_locations[location]: # check each neighbor based on open spot
    if neighbor not in creature_locations.keys():
        del creature_locations[(x, y)] # remove the the location before moving him to the new location
        creature.x_Coordinate = neighbor[0] # assign a new x coordinate
        creature.y_Coordinate = neighbor[1] # assign a new y coordinate
        location = (neighbor[0], neighbor[1]) # assign a new location
        creature_locations[location] = 'relocated' # add the NEW updated location to the location list
        # this creature has been relocated,so we move on to the next creature
        # this location will be now seen as occupied to the following creatures
        break
```

אנו דואגים שאין שני יצורים על אותה משבצת על ידי שמירת כל מיקומי היצורים במילון כ-key עם המיקומים האפשריים שהם יכולים לזוז אליהם. אם המיקום שיצור רוצה לזוז אליו חוזר כאשר אנו ניגשים למילון, המשמעות היא שכבר יש שם יצור אחר. במקרה זה, אנו ממשיכים לחפש האם מתאפשרת תזוזה עבור כיוונים אחרים בהם קיים תא ריק (עם מיקום שלא נמצא במפתחות המיקומים). ברגע שהיצור כן זז, נמחק את המיקום מרשימת המיקומים (כדי לפנות מקום) ונעדכן ונוסיף את המיקום החדש.

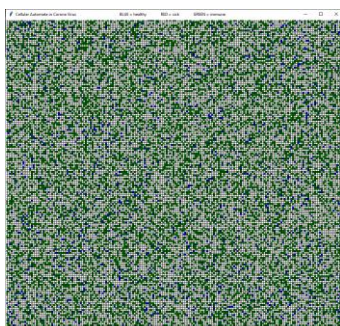
נדגיש שאנו עושים shuffle על רשימת המיקומים עצמם, ולכן התזוזה גם היא רנדומלית. במקרה הגרוע ביותר - ליצור אין מקום לזוז וגם לא נבחר רנדומלית שהוא יישאר במקום, הוא מחויב להישאר במקום, ואז לא נשנה את המיקום שלו בכלל ונמשיך ליצור הבא.

ניתוח גלי ההדבקה

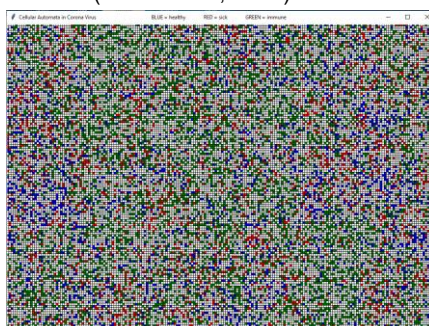
מצאנו שכאשר הפרמטרים באופן הנ"ל (ומוגדרים כערכי ברירת מחדל למשתמש):



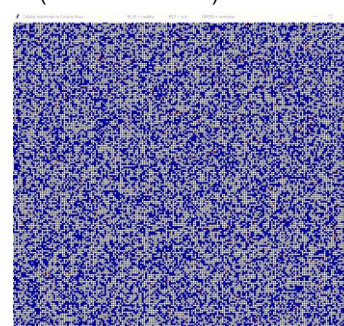
מצב סופי (בריאים וחסנים בלבד)



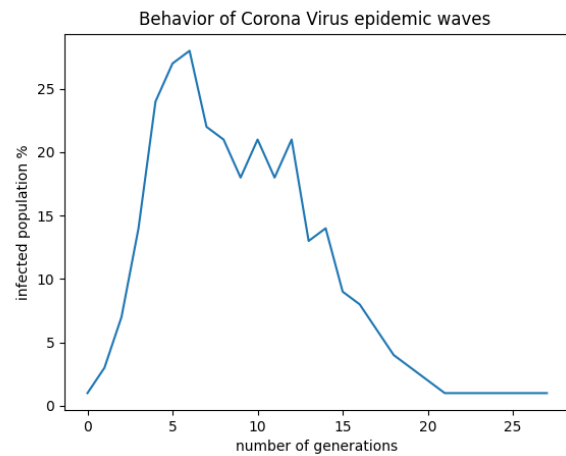
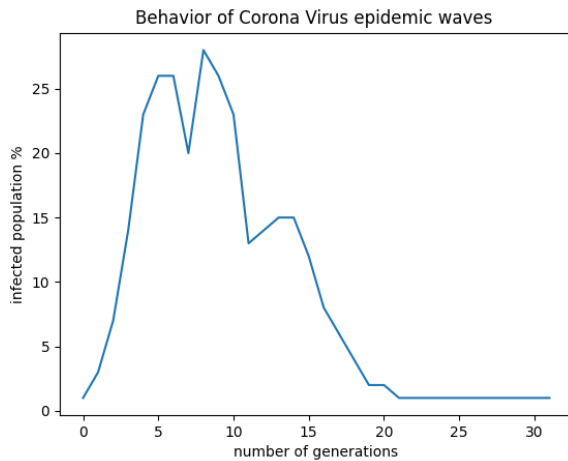
מצב ביניים (בריאים, חולים וחסנים)



מצב התחלתי (בריאים וחולים בלבד)

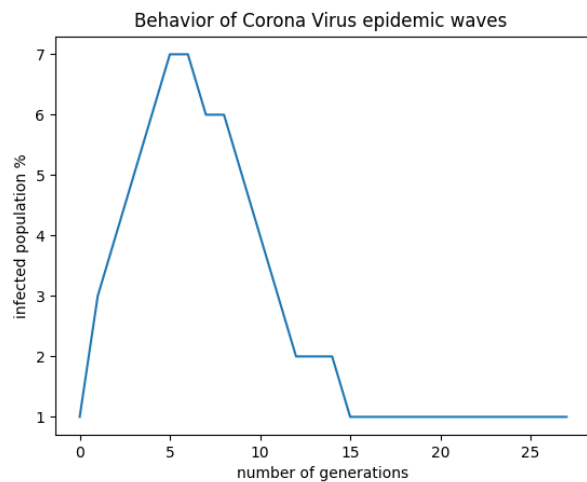
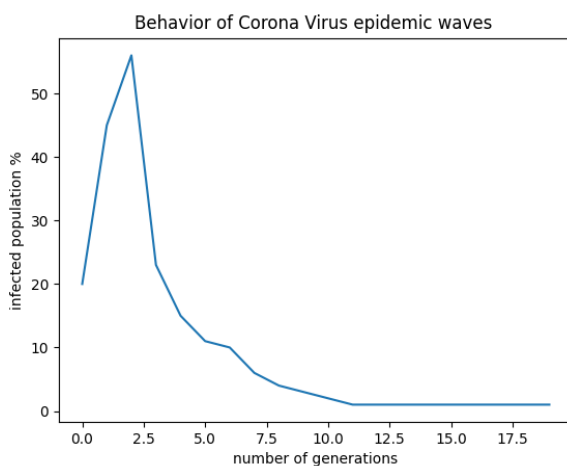


רואים תופעה של גלי קורונה, בהם קיימים לפחות 3 פיקים (עלייה וירידה בכמות החולים במהלך הסימולציה):



אנו מסיקים מכך שעל מנת שיוצרו גלי הדבקה, אסור שיותר מדי מהאוכלוסייה ידבקו בבת אחת או שפחות מדי חולים באותו רגע. ההבחנות הבאות תומכות בכך -

מצאנו שעם מספר נמוך מדי של ימי מחלה (<4) - אין ליצורים מספיק זמן להעביר את המחלה והיא פשוט דועכת בין אם אחוז החולים ההתחלתי גבוה או נמוך (עבור $X=1$ הסימולציה נגמרת כמעט מיד עקב זמן החלמה מהיר מדי):



מצד שני, עם מספר גבוה של ימי מחלה (>4) קיבלנו דפוס דומה. הרוב נדבקים מהר, ולכן גם מהר מפתחים חסינות:

