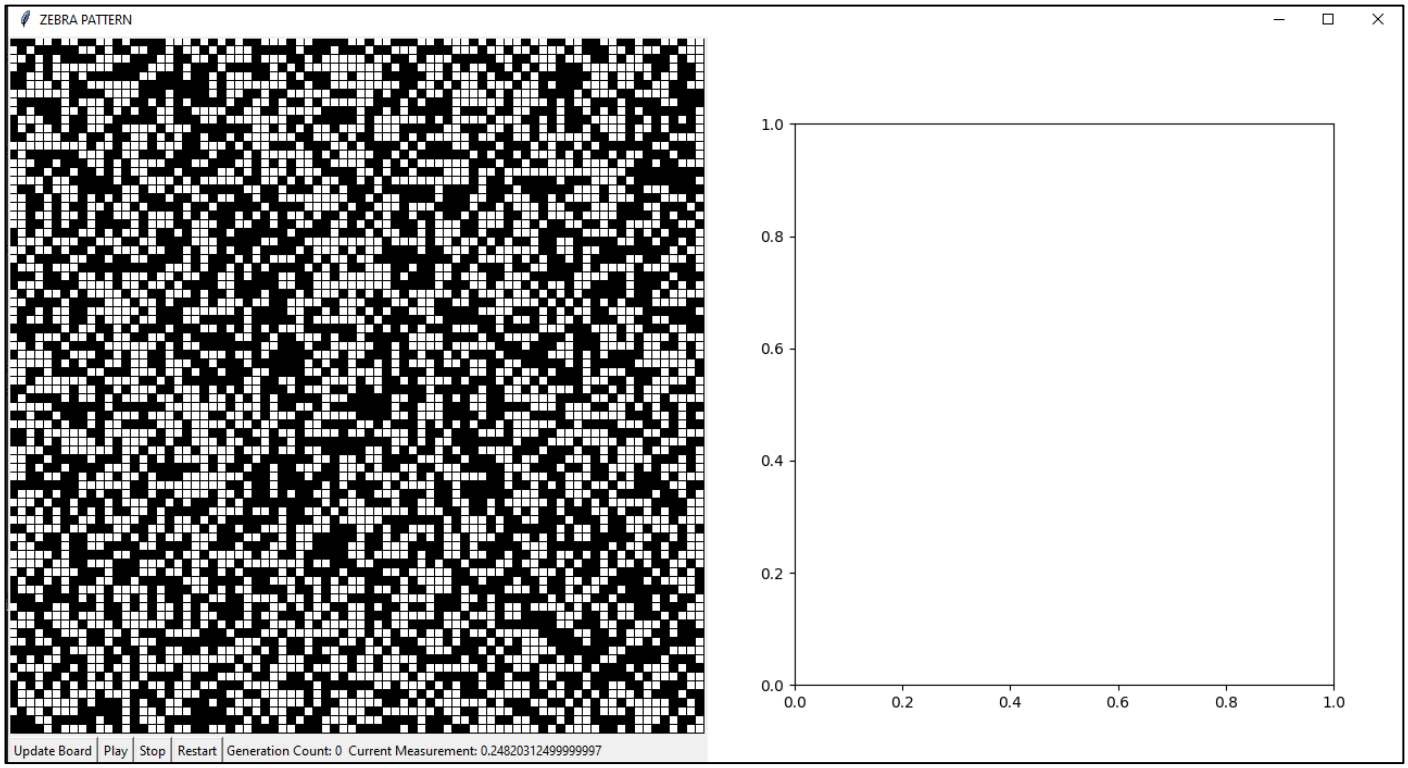
בס"ד

ביולוגיה חישובית – "מודל זברה"

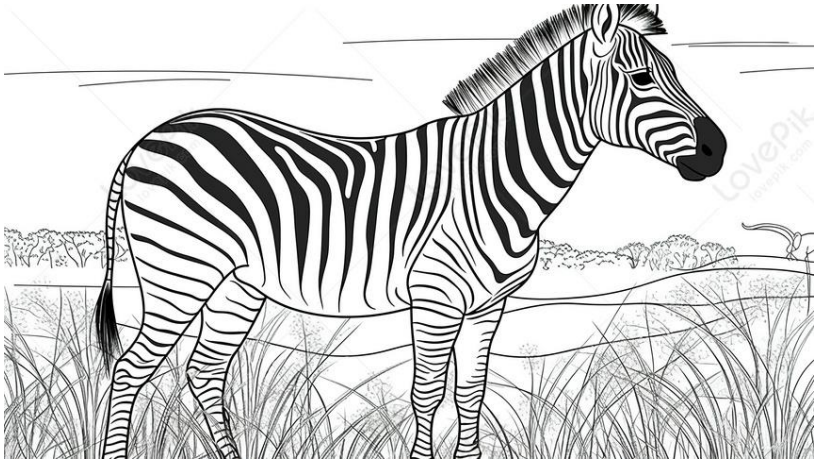
נופר גלוטמן 322631458 & יובל דהרי 209125939

דו"ח זה הינו עבור מכונה "מבוזרת" המקבלת מטריצה בגודל 80*80 המאותחלת רנדומית (50-50) בצבעי שחור לבן, ושואפת ליצירת תבנית "זברה" של פסים שחורים ולבנים בזה אחר זה (שחור-לבן-שחור-לבן...).



<u>דו"ח זה מחולק לשלושה חלקים:</u>

- תיאור והסבר לוגיקת המכונה.
- מדד לטיב "דור" נוכחי.
- דוגמות הרצה וניתוחן.



חלק א' - תיאור והסבר לוגיקת המכונה

הלוגיקה בה בחרנו הינה כזו:

```
# Function implementing the logic to determine the color of a cell
def machine(row, col):
    """Apply the machine logic to determine the color of a cell."""
    # Extract colors of neighboring cells
    (color_upper_neighbor, color_lower_neighbor, color_right_neighbor,
     color_left_neighbor, color_upper_right_neighbor, color_upper_left_neighbor,
     color_lower_right_neighbor, color_lower_left_neighbor) =
     neighboring_colors(row, col)

    # Apply logic to determine the color of the current cell based on its neighbors
    # Checking the column neighbors and matching them if they are the same color
    if color_upper_neighbor == color_lower_neighbor:
        return color_upper_neighbor
    else:
        # List of all neighbors in the right and left column
        neighbors = [
            color_right_neighbor, color_upper_right_neighbor,
            color_lower_left_neighbor, color_left_neighbor,
            color_upper_left_neighbor, color_lower_right_neighbor
        ]

        # Check all permutations of 3 neighbors from the sides and return the opposite
          color if they are all the same
        for comb in combinations(neighbors, 3):
            if len(set(comb)) == 1:
                return COLORS[1] if comb[0] == COLORS[0] else COLORS[0]
```

---

1) החזר את צבעי שמונת השכנים הקרובים של התא (בצורה ציקלית) וצור להם משתנים.
2) בדוק האם צבע התא העליון שלי שווה לצבע התא התחתון שלי:
   a) אם כן – שנה את הצבע שלי לצבע הזה.
   b) אחרת –
      i) צור רשימה של ששת צבעי השכנים הקרובים אליי מהצדדים שלי.
      ii) עבור על כל הפרמוטציות של שלושה איברים מתוך הרשימה
         (1) צור מהם קבוצה (מצמצם כפילויות) ובדוק האם גודל הקבוצה שווה ל-1, כלומר כלל הצבעים בה זהים:
            (a) אם כן בדוק:
               (i) אם הצבע שחור – שנה את הצבע שלי ללבן.
               (ii) אם הצבע לבן – שנה את הצבע שלי לשחור.

כעת, נסביר את הרעיון מאחורי הלוגיקה.

על מנת להגיע למצב בו ישנם פסים מתחלפים של שחור ולבן, אנו צריכים כי כל עמודה תהיה באותו הצבע ובו זמנית שונה מצבע העמודות השכנות לה.

לכן, התנאי הראשון הינו טריוויאלי – בדיקה של צבעי השכנים הצמודים אליי בעמודה שלי והתאמת עצמי אליהם במידה והם באותו הצבע.

במידה ותנאי זה איננו מתקיים ישנן אפשרויות רבות מה לעשות, ובחרנו ללכת על זו המעניינת ביותר. כיוון שכעת אין משמעות לצבעי השכנים הצמודים אליי בעמודה שלי (כי אחד שחור ואחד לבן) אנו מעבירים את כלל המשקל שלנו להסתכלות על העמודות השכנות לנו.

נשים לב, שישנם שישה שכנים צמודים אלינו מעמודות אלה, ובעצם פי שלוש יותר מידע פוטנציאלי מאשר השכנים מהעמודה שלנו.

כעת, נעבור על כלל הפרמוטציות בגודל שלוש מתוך ששת השכנים ונבדוק האם ישנם שלושה באותו הצבע, במידה ומצאנו שלושה בעלי אותו הצבע, נבחר לעצמנו את הצבע ההפוך להם.

נשים לב שלוגיקה זו נותנת לנו שני דברים מהותיים:

1) על פי עקרון "שובך היונים" אנו מקבלים הבטחה שישנם לפחות שלושה שכנים מתוך ששת השכנים עליהם אנו עוברים שהם בעלי אותו הצבע, כך בעצם אנו יודעים שאנחנו לא צריכים לבצע צעד נוסף בלוגיקה לאחר צעד זה.
2) מתוך ההבטחה הקודמת נגזר, שאנו נותנים התייחסות ללפחות מחצית (אם לא יותר) מהשכנים הצמודים לנו מהצדדים וכך בעצם הולכים על פי הרוב ומבצעים את האדפטציה האישית שלנו בצורה מיטבית.

חלק ב' – מדד לטיב "דור" נוכחי

המדד בו בחרנו ממומש באופן הבא:

```
# Function to calculate the measure of the board
def get_measure():
    """Calculate how close the matrix to 'zebra' form."""
    horizontal_consistency_count = 0
    vertical_consistency_count = 0

    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLUMNS):
            current_color = MATRIX_COLORS[CURRENT_SPIN][row][col]

            # Extract colors of neighboring cells
            (color_upper_neighbor, color_lower_neighbor, color_right_neighbor,
             color_left_neighbor, _, _, _, _) = neighboring_colors(row, col)

            # Check horizontal neighbors
            if current_color != color_left_neighbor
               and current_color != color_right_neighbor:
                horizontal_consistency_count += 1

            # Check vertical neighbors
            if current_color == color_upper_neighbor
               and current_color == color_lower_neighbor:
                vertical_consistency_count += 1

    # Calculate the consistency measures
    total_cells = BOARD_ROWS * BOARD_COLUMNS
    m_horizontal = horizontal_consistency_count / total_cells
    m_vertical = vertical_consistency_count / total_cells

    # Return the average between the measures
    return (m_horizontal + m_vertical) / 2
```

---

1) אתחל משתנה עבור כמות התאים המסודרים ביחס ללוגיקת השורות.
2) אתחל משתנה עבור כמות התאים המסודרים ביחס ללוגיקת העמודות.
3) בצע לולאה על כלל התאים כך -
   a) חלץ את צבע התא הנוכחי.
   b) חלץ את צבעי השכנים הצמודים אליי הנמצאים על הצלב שלי.
   c) בדוק האם הצבע שלי שונה מצבע השכן הימני שלי ומצבע השכן השמאלי שלי.
      i) אם כן – הוסף 1 לכמות התאים המסודרים ביחס ללוגיקת השורות.

d) בדוק האם הצבע שלי זהה לצבע השכן מעליי ולצבע השכן מתחתיי.
   i) אם כן – הוסף 1 לכמות התאים המסודרים ביחס ללוגיקת העמודות.
4) חשב את גודל כלל המטריצה
5) חשב את הכמות היחסית של התאים המסודרים ביחס ללוגיקת השורות.
6) חשב את הכמות היחסית של התאים המסודרים ביחס ללוגיקת העמודות.
7) החזר את הממוצע בין הכמות היחסית של התאים המסודרים ביחס ללוגיקת השורות לבין הכמות היחסית של התאים המסודרים ביחס ללוגיקת העמודות.

כעת, נבאר את רעיון המדד בו בחרנו ומדוע הוא טוב.

לשם ההבנה נתבונן לרגע במצב אליו אנו שואפים ברמת כל תא בנפרד, כלומר כיצד תא מסוים רוצה את ה"עולם" סביבו במצב מושלם. במצב זה אלו הנמצאים מעליו ומתחתיו יהיו בצבע שלו ואלו הצמודים לו מימין ומשמאל יהיו בצבע ההפוך ממנו.

כאשר אנו מבינים זאת, טיב המדד הינו ברור - אנו בעצם עוברים על כלל התאים ובודקים כמה מהם נמצאים במצב "מושלם" ביחס לשכנים שלהם.

כיצד אנו מבצעים זאת? אנו סוכמים את יישור הקו ביחס לעמודות בנפרד וביחס לשורות בנפרד, מחלקים בך כמות התאים על מנת לבדוק יחסית את כמות התאים הנמצאים במצב טוב, ומבצעים ממוצע בין שני המדדים (האחד שלוקח בחשבון את יישור הקו של העמודות והאחד המתייחס לשורות), כך בסך הכל אנו מקבלים את אחוז התאים הנמצאים במצב מושלם ביחס לשכנים שלהם.

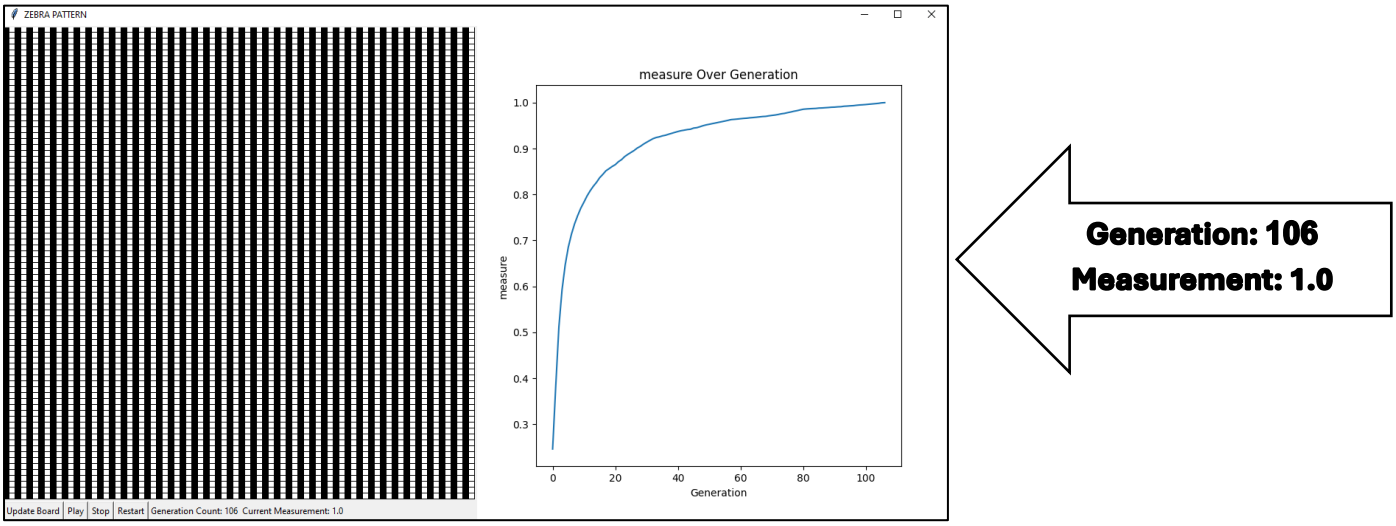
חלק ג' – דוגמות הרצה וניתוחן

ביצענו 10 דוגמות הרצה[1], עם חסם עליון של 300 הרצות.

נשים לב, כי כלל הדוגמות הגיעו לכדי התכנסות (בין אם למצב מושלם ובין אם ללולאה) בממוצע לאחר 114~ דורות, על כן, עצרנו את ההרצות שיוצגו כעת לאחר התכנסותן. ככלל ניתן לראות כי המדד עולה בצורה אקספוננציאלית ויפה מיד בתחילת התהליך בכל ההרצות.

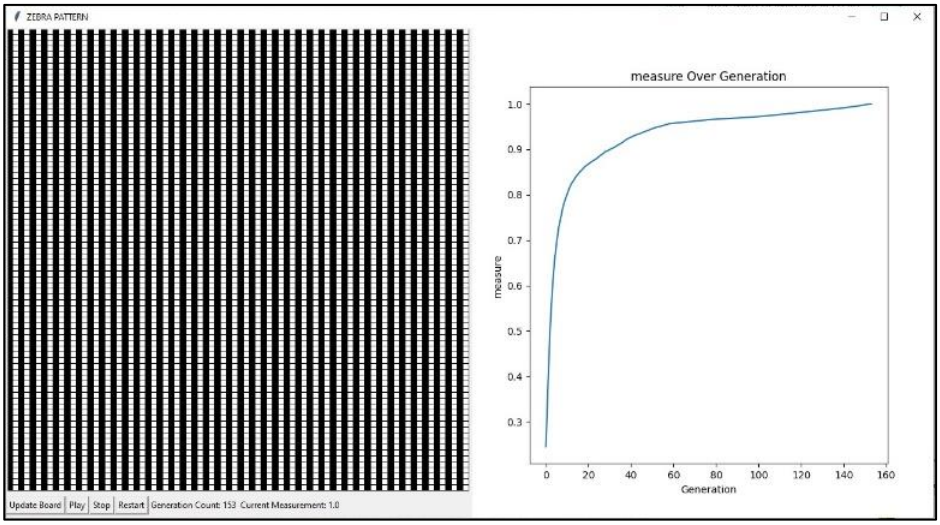
לשם פשטות ההסבר נחלק את דוגמות ההרצה לשלוש קטגוריות:

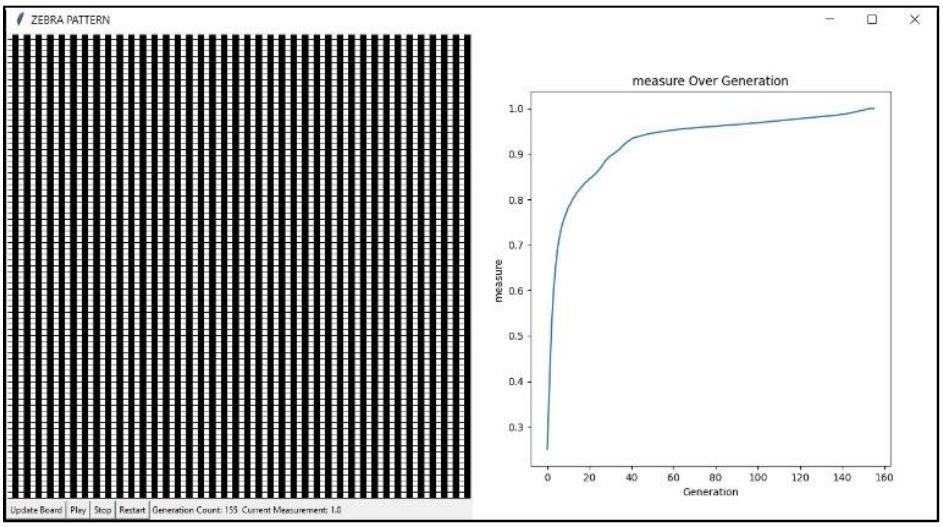
1) התכנסות מלאה – חמישה מקרים
במקרים אלו אנו רואים, שאכן ישנה התכנסות יפה ומלאה למצב האידיאלי של עמודה עמודה.



[1] רוב ככל ההרצות שלנו מתכנסות בצורה מושלמת או כמעט מושלמת, לשם הגיוון בדו"ח ביצענו המון הרצות כדי לקבל גם אחת מיוחדת.

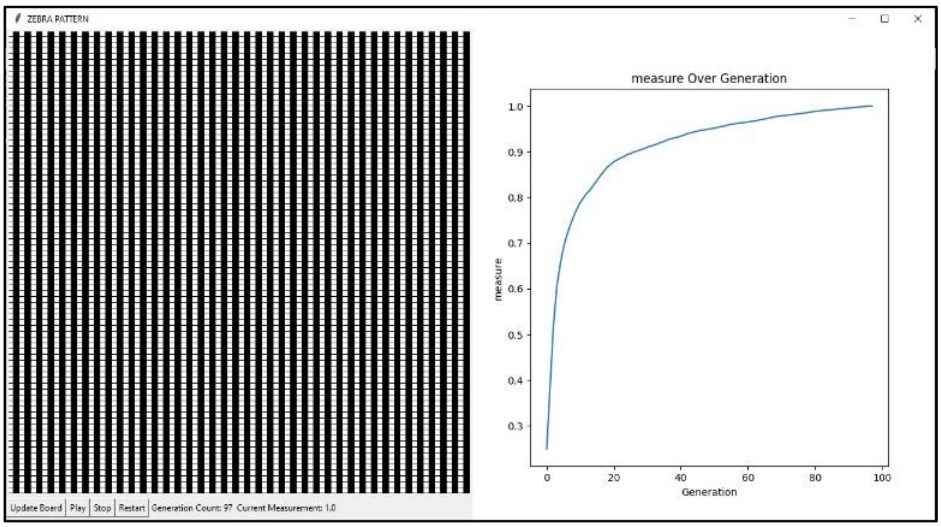
Generation: 153
Measurement: 1.0

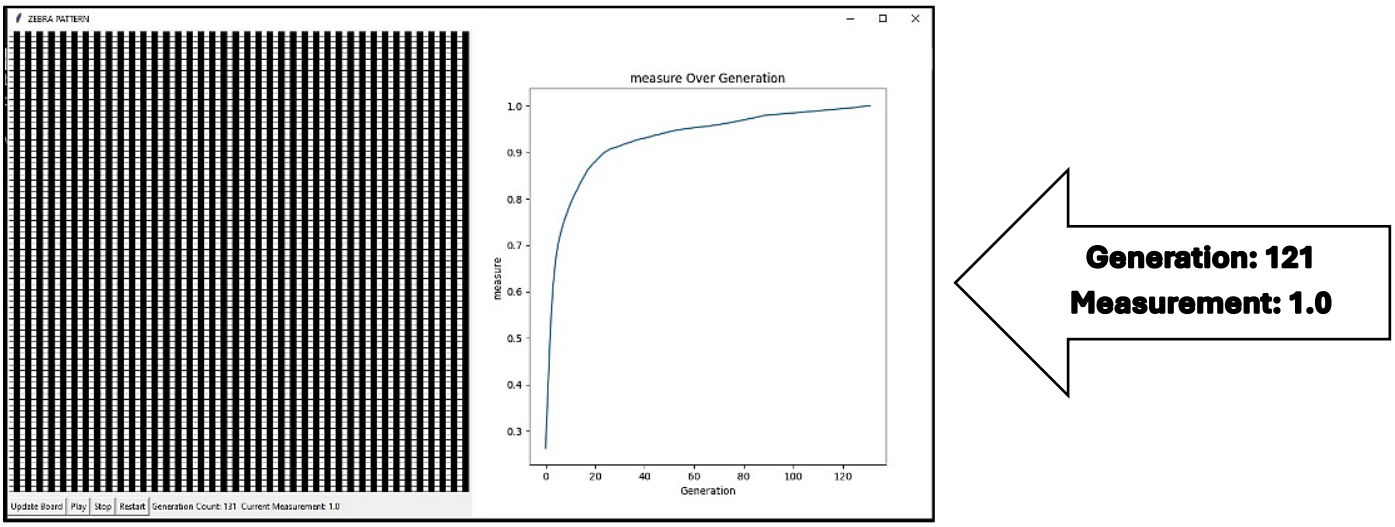




Generation: 155
Measurement: 1.0

Generation: 97
Measurement: 1.0



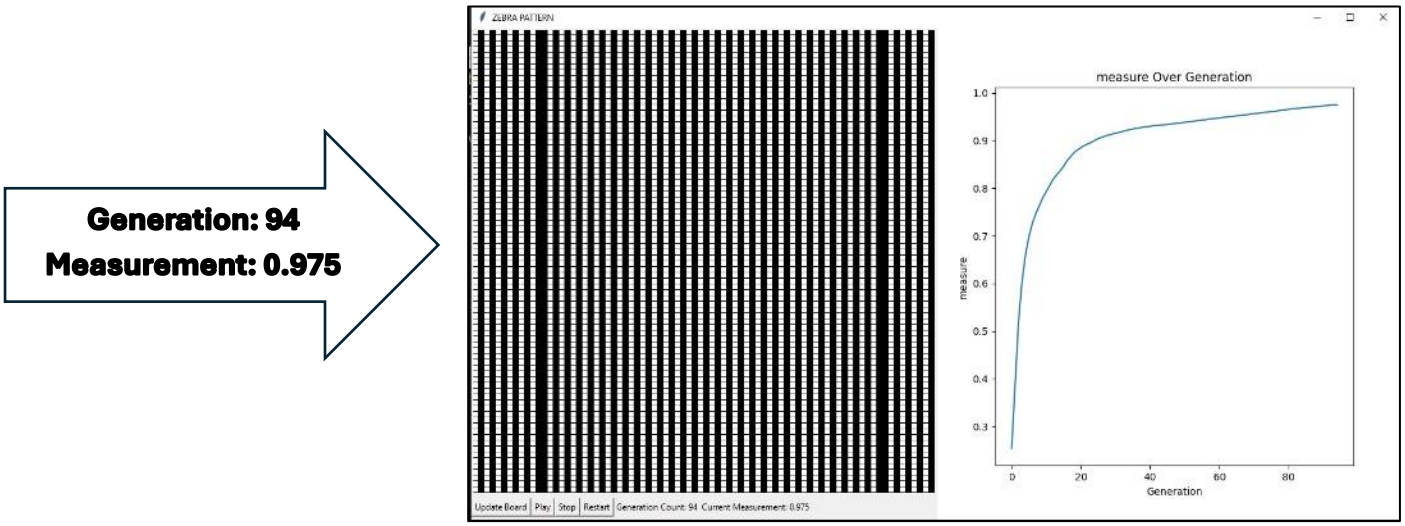


מעבר לאינדיקציה הברורה מהמטריצה המסודרת, ניתן לראות שאכן האמד הגיע לכדי 1.0 דבר המייצג כי ההתכנסות הושלמה בהצלחה.

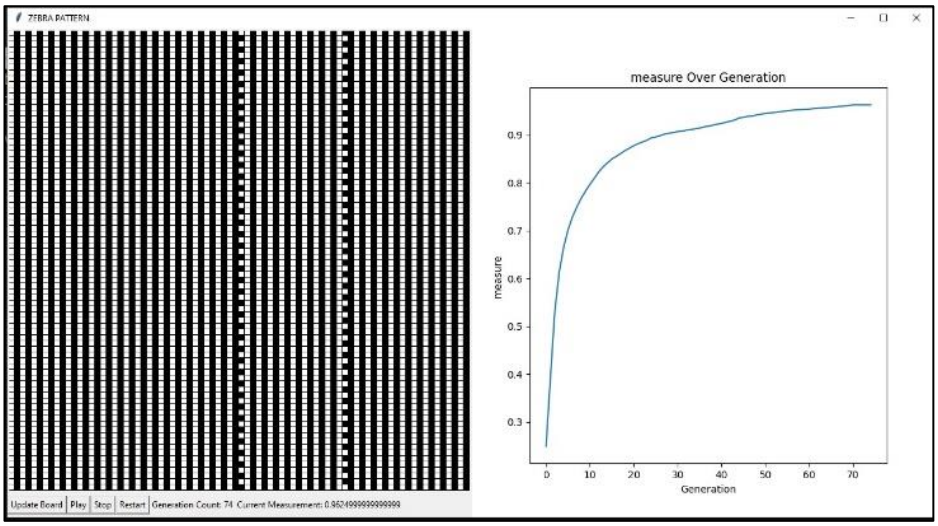
2) <u>התכנסות 'כמעט' מלאה</u> – שלושה מקרים

במקרים אלו אנו רואים, שישנה התכנסות כמעט מלאה ומושלמת.

במקרה הראשון, קיבלנו כי ישנן פעמיים 2 עמודות שחורות ברצף, העמודה הכפולה יכולה להיווצר בפשטות כאשר ישנן שתי עמודות צמודות שיש להן צבע דומיננטי מאוד זהה, דבר שנמוך הסתברותית אך יכול לקרות. במקרה כזה לא תהיה התייחסות (לא מהותית) לעמודות מהצדדים ולכן תהיה התכנסות זהה לשתיהן.



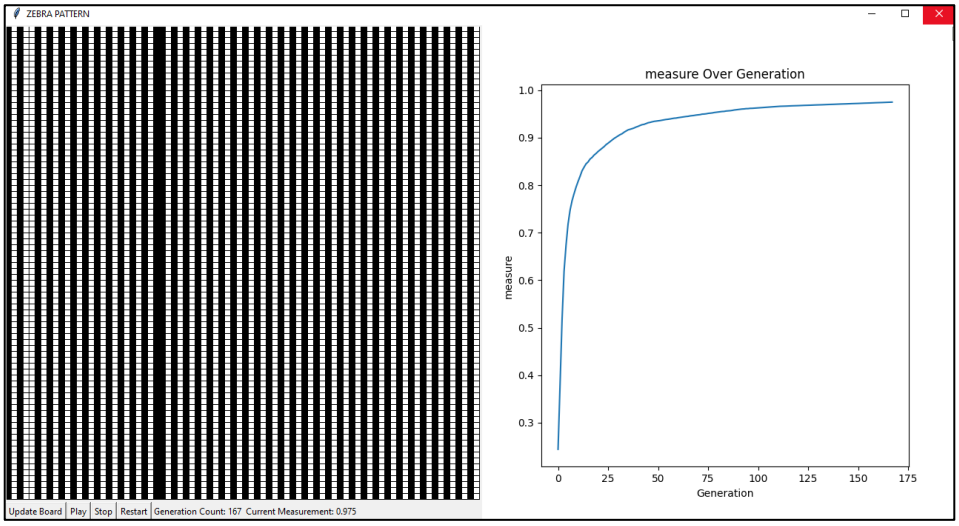
במקרה השני, נקבל 2 עמודות מפוספסות, דבר שיכול לקרות על פי הלוגיקה של המכונה, כיוון שכשם שהזכרנו לעיל התנאי הראשון שנבדק הוא על התא העליון והתחתון, דבר הגורם למצב מפוספס להישמר (במידה והתקבע).



Generation: 74
Measurement: ~0.9625

המקרה השלישי, דומה מאוד למקרה הראשון, רק שכאן אנו מקבלים עמודה כפולה לבנה ועמודה כפולה שחורה. ניתן גם לראות שהמדד בשתיהן מציג תוצאה זהה כפי שהיינו מצפים – 0.975.

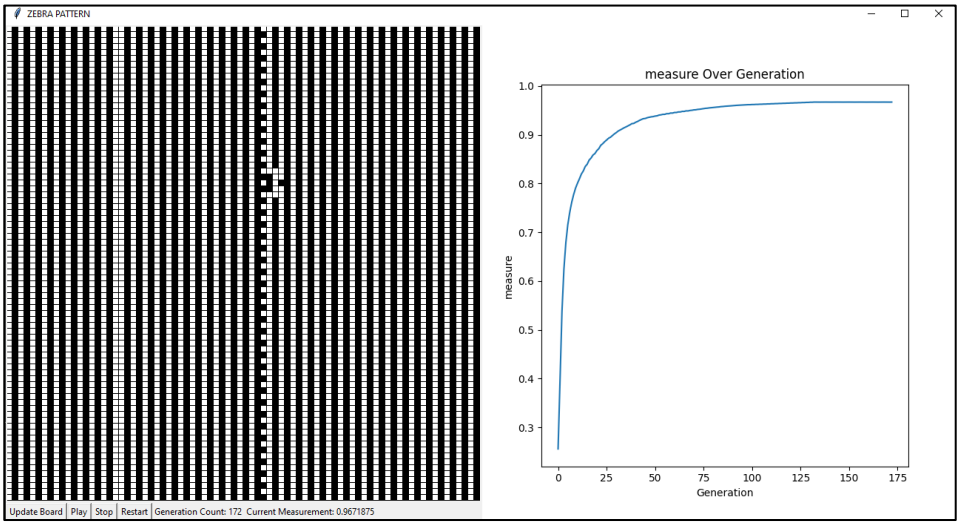
Generation: 167
Measurement: 0.975



3) <u>מקרים מיוחדים</u> – מקרה יחיד

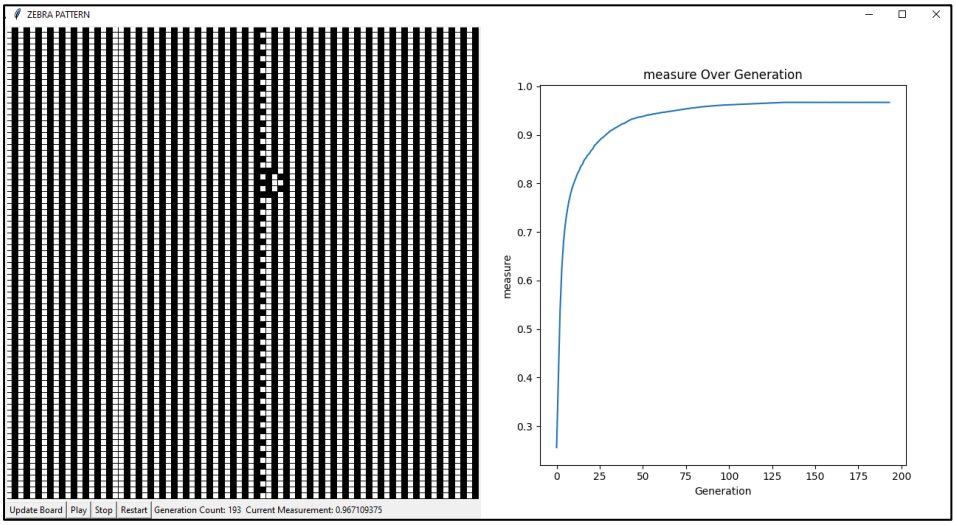
המקרה הראשון מראה על כמעט התכנסות מושלמת, עם פס כפול יחיד ופס מפוספס עם "מוטציה" קטנה המחוברת אליו, המציגה תבנית החוזרת על עצמה.

נשים לב, שאף במקרה "מיוחד" מעין זה המכיל "מוטציה", אנו רואים שישנה התכנסות יחסית טובה, יש צורה מוגדרת היטב לרוב ככל המטריצה, אך כן ישנן "סטיות" כאלו ואחרות שנכנסו למן לולאה משלהן. חשוב לשים לב, שגם בדוגמה שהצגנו ההתכנסות ההתחלתית מאוד טובה ועולה אקספוננציאלית.

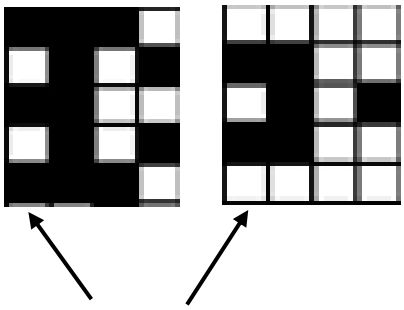


Generation: 172
Measurement: ~0.9672

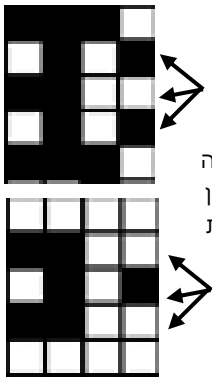
Generation: 193
Measurement: ~0.9671



על החלק המפוספס והעמודה הכפולה כבר דיברנו בעבר, לכן נתבונן על הלולאה של ה"מוטציה" המעניינת שנוצרה ובחלק מהסיבות ללולאה שלה.



העמודות מתחלפות זו עם זו כי הן מפוספסות



שלושת הקוביות האלה
מתחלפות בניהן כי הן
בפני עצמן מפוספסות