**אלגוריתמים גנטיים – דו"ח תרגיל בית 2**נעמה אומר 207644014 ומשה זאב הפטר 205381379

[**https://github.com/NaamaOmer/EX2\_NAAMA\_MOSHE**](https://github.com/NaamaOmer/EX2_NAAMA_MOSHE)

בתרגיל זה התבקשנו לממש אלגוריתם גנטי שעל ידו נפתור לוחות של חידת המספרים פוטושיקי- Futoshiki. הקוד נכתב בשפת python וביצענו אותו במבנה של OOP, נפרט בהמשך על ממימוש הפרטים באלגוריתם. על התוכנית שבנינו: בהינתן קלט של לוח משחק, ניסינו למצוא פתרון חוקי. התחלנו מהצבה רנדומית של מספרים במטריצה שיהוו את הדור הראשון, אך עם התערבות בתקינות השורות - ההצבה תהיה עם מופע אחד ויחיד בכל שורה מכל מספר במטריצה 1-N. לאחר מכן, בעזרת האלגוריתם האבולוציוני נייצר את הדורות הבאים בכך שנהרוג חצי מהאוכלוסייה, ונעשה הכלאה בין פתרונות לכדי יצירת 100 פתרונות חדשים לדור החדש.   
  
**פתרון חלק א' – אלגוריתם גנטי רגיל:**מימוש ייצוג הפתרונות: את הפתרונות בחרנו לייצג ע"י מופע של אובייקט מסוג "פתרון" (= Solution) ובו השדות הבאים:  
 א. **לוח-** שהוא מטריצה בגודל שהוגדר בקלט שנכנס, תחילה התוכנית מייצרת אובייקט מסוג לוח Board, מתוך הקלט שהתוכנית קיבלה. זהו בעצם קובץ טקסט עם קונבנציה מוסכמת לבדיקת התרגיל –בו מופיע כל המידע לבניית הלוח הגולמי – ללא מספרים שיהפכו אותו לפתרון כל שהוא.   
ב. **גודל הלוח-** על פי הקלט שהוכנס.  
ג. **התנאים שקיימים על** **הלוח –** סימני ה">", אשר עלינו להתאים את הלוח כדי שיענה על כל התנאים הללו והערכים הקבועים שהם מאותחלים ונשארים קבועים בכל פתרון.  
ד. **משתנה הערכה –** המשתנה מתעדכן ע"י פו' ההערכה שנקראת בקוד שלנו: evaluate, שהיא בעצם מתודה פנינית של האובייקט מסוג Solution.

מימוש של פו' הערכה: את פונקציית הערכה בחרנו ליישם ממש בצורה של מתודה בתוך מחלקת הפתרון כפי שהזכרנו בסעיף ד' בהסבר על המימוש הקודם. מטרתה לעלות על כל "טעות" שיש בפתרון הנוכחי שעליו היא עובדת – טעות יכולה להיות אחת מבין האופציות הבאות:  
 1. **אי תקינות של אי שוויון כלשהו בלוח –** ע"פ התנאים שקיבלנו בקובץ הקלט, כל אי שוויון על הלוח שערכיו אינן מקיימים את אי השוויון – זוהי טעות שעולה נקודה אחת לציון הערכה (כאמור, הציון הוא ציון שלילי, כלל שהוא גבוה יותר, כך מראה על יותר טעויות בפתרון – ולכן כלל שההערכה יוצאת גבוהה יותר, הפתרון נחשב טוב פחות.   
2. **קיום שהוא אינו יחיד לכל מספר בעמודה ובשורה.** כלומר, קיום ערך כלשהו יותר מפעם אחת בשורה, או יותר מפעם אחת בעמודה. כל קיום נוסף עולה נקודה אחת, שנכנסת למשתנה הערכה.

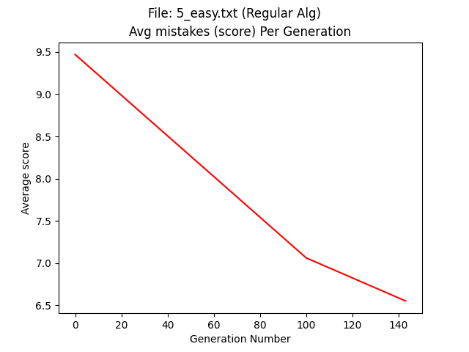
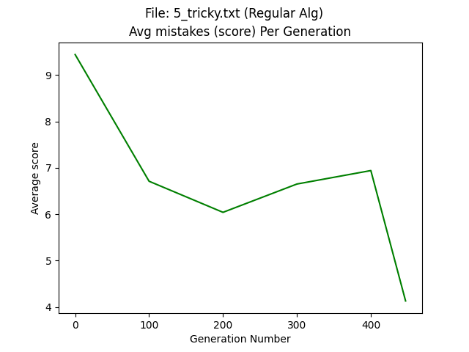
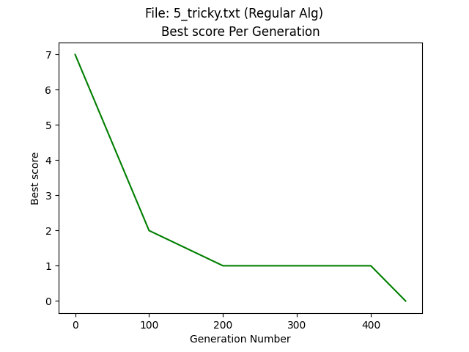
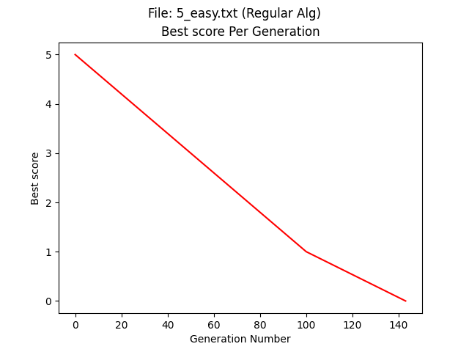
הכלאה (cross-over) בין פתרונות שונים: בחרנו לבצע את ההכלאה בין פתרונות ע"י לקיחת שורות שלמות מכל אחד מן "הפתרונות אב", כלומר כלל לא לשנות פרטים בתוך שורה. כך נוצר איזה שהוא קיבוע לתקינות השורות במטריצה, דבר שהכרחי כדי להגיע לפתרון מלא, ומשאיר לנו להתמודד עם הסדר של השורות אחת ביחס לשנייה, אך גם עם הסדר פנימי בשורות במקרים של אופטימיזציה/מוטציות אקראיות שמתרחשות בכל דור ודור.

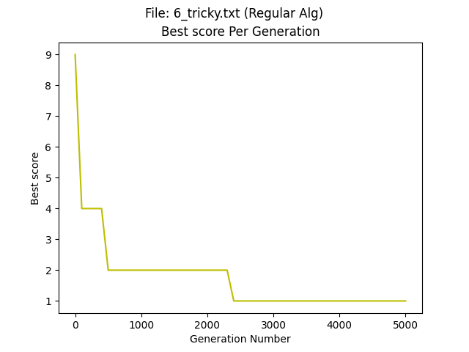
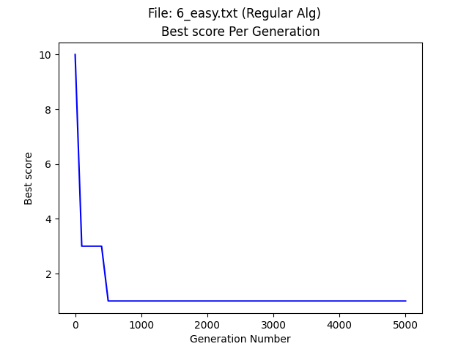
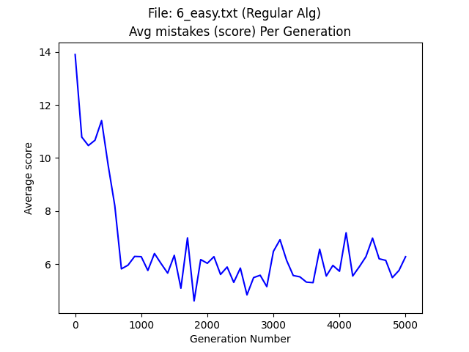
מימוש של מוטציות באלגוריתם:באלגוריתם שבנינו, מוטציה יכולה לקרות בהסתברות מסוימת, אך אינה קוראת בוודאות בכל פתרון בדור. המימוש התבצע באמצעות מתודה שמטרתה לבצע מוטציות בלוחות נקראת :mutation (מתודה ששייכת למחלקה (Futoshiki\_Solve, המקבלת כקלט את ההסתברות שבכלל תקרה מוטציה. המתודה אינה מחזירה דבר, אלא רק משנה את ערכי האובייקט מסוג פתרון שעליו עבדנו. המוטציה מיוצגת ע"י שחלוף של שני ערכים באותה השורה: ערך בשורה עם ערך אחר בשורה, בתנאי שבכלל ניתן להזיז את שניהם, כלומר שאף אחד מהם לא אחד מן המשתנים ההתחלתיים שמגיעים עם הלוח. את ההסתברות ליצירת מוטציה עבור כל לוח בכל דור, בחרנו לייצג ע"י סוג של הטלת מטבע, הסיכוי למוטציה הוא הסיכוי לקבל במטבע 1, שאומר שכן תתבצע מוטציה, ו0 אחרת – לא תתבצע מוטציה.

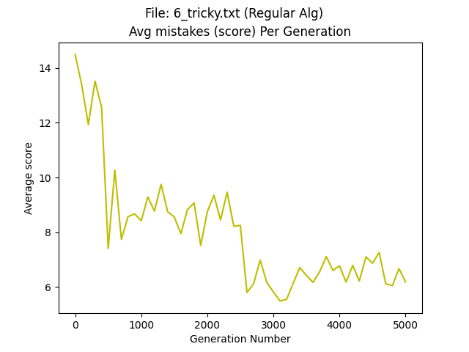
את בעיית ההתכנסות המוקדמת ניסינו לפתור באמצעות הגברת המוטציות. פרקטית בקוד עשינו זאת ע"י הגדלת הסיכויים לקבל 1 בתוך הפונקציה שמגרילה למתודה mutation האם לבצע מוטציה או לא, אם למתודה mutation לא הוכנס ערך, סיכוי ברירת המחדל הוא 0.05. ע"י שליחה של ערך p גבוה יותר, נוכל להגדיל את הסיכוי למוטציה, וכך גם את הסיכויים לפתרונות נוספים שיוסיפו שונות לפתרונות בזמן ההכלאה. כך פעלנו בכדי לעצור את ההתכנסות המוקדמת.

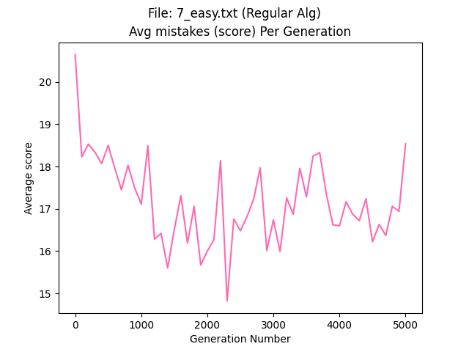
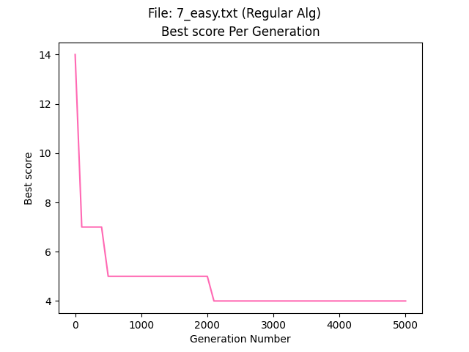
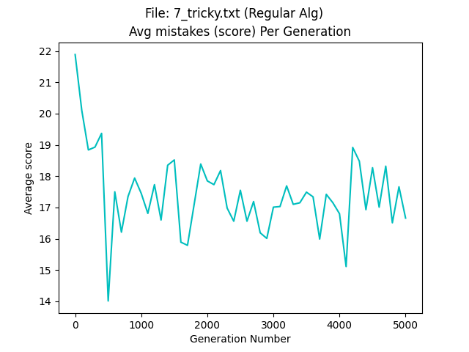
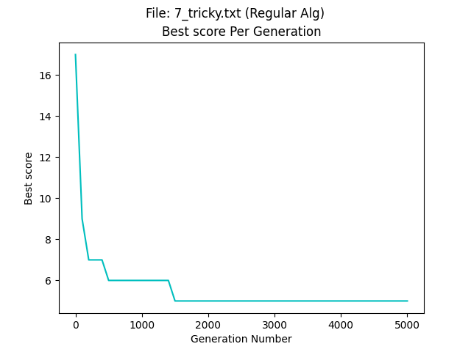
כיצד בחרנו את הדור הבא? לקחנו בכל דור את חצי ה"טובים" – 50 הפתרונות עם הציונים הנמוכים ביותר (כלומר עם הכי פחות טעויות) ואת השאר המתנו (מהמילה מוות). מתוך מי שנשאר, הגרלנו 2 מספרים עם bias לכיוון המוצלחים יותר (הראשונים יותר ברשימה), ועשינו את זה **100 פעמים.** כך יצרנו דור חדש.  
נציין כי כן בכל דור, החלטנו להשאיר 5 פתרונות עם הציונים הכי נמוכים לדור הבא. במקרה של אופטימיזציה עלתה השאלה האם יעיל להשאיר פתרונות בחוץ, כי אלה לא יעברו אופטימיזציה שיעבירו לדורות הבאים מכיוון שאינם מתחלקים. שיערנו כי ביצוע אופטימיזציה גם עליהם יכול לשפר את הביצועים של האלגוריתם. אך התבדינו בכך שראינו כי ביצוע אופטימיזציה על הלוחות שאנו משאירים לדור הבא דווקא גרמה לאלגוריתם להתעכב עד שמגיע לפתרון/ציון המינימלי שמתקרב לפתרון (1 or 2score of ). לכן, החלטנו להשאיר אופטימיזציה זו רק ללוחות חדשים שנוצרים. \*\*\*\* להראות דוגמה\*\*\*\*\*

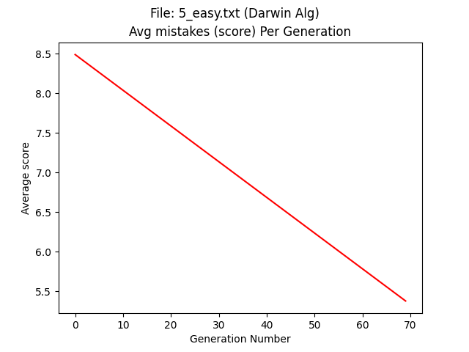
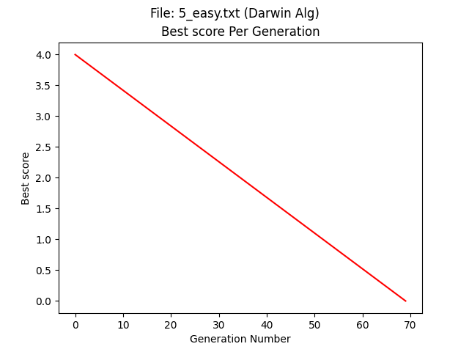
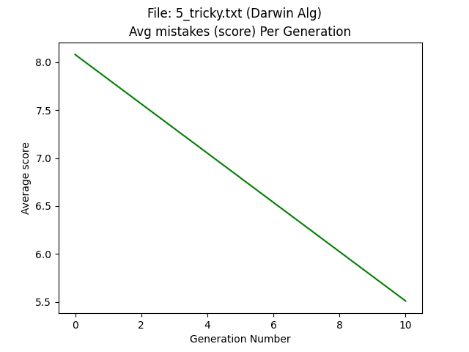
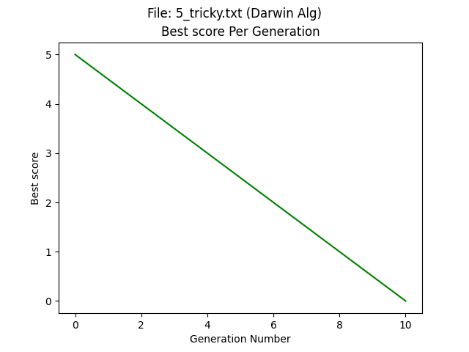
כמות הדורות שבחרנו להריץ את האלגוריתם: האלגוריתם רץ כל עוד 2 התנאים הבאים עדין לא התקיימו: עבר 50,000 דורות ולא נמצא פתרון, או שנמצא פתרון ואז אין סיבה להמשיך לרוץ. בעיקרון הריצה של האלגוריתם הוא עד שמתכנס לכדי פתרון עם ניקוד שהוא מושלם – 0 טעויות, או שאינו מצליח להגיע להתכנסות ועל כן נעצור כדי לא להיכנס לריצה אין סופית. בחרנו את 50,000 הדורות מכיוון שמהניסיונות הרבים שלנו בהרצת האלגוריתם ראינו כי אם לאחר 50,000 דורות לא מצאנו פתרון, הסיכוי לכך שנצליח למצוא עד 100,000 הוא נמוך. ולכן הסקנו כי באלגוריתם זה, 50,000 קריאות לפו' הערכה בכל דור ודור מספיק זמן כדי לבדוק האם האלגוריתם מצליח למצוא פתרון או לא.

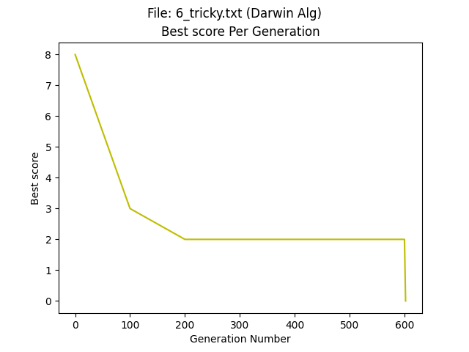
**פתרון חלק ב' – בחינת השפעת אבולוציה "למארקית" (Lamarck) על האלגוריתם הגנטי:**על מנת לבחון נושא זה ניתן לכל פתרון הזדמנות לבצע שיפור כלשהו של יכולתו (מעין אופטימיזציה לוקאלית) **ונאפשר העברת הפתרון המשופר לדור הבא.** אנחנו בחרנו בהחלפה בין ערכים בשני תאים אם התאים סותרים אילוץ. **נתיר לבצע מספר קבוע של צעדי אופטימיזציה בכל דור -** מספר צעדי האופטימיזציה כגודל המטריצה: למשל במטריצה בגודל 5\*5 נאפשר 5 צעדי אופטימיזציה בכל דור. אופטימיזציה לפתרון קוראת במתודה פנימית של המחלקה Solution ונקראת optimizer (), נפרט על דרך פעולתה כבסיס להשוואה בין האסטרטגיות השונות, שחלקן משתמשות באופטימיזציה ואחת לא, וגם השתיים שמשתמשות משתמשות באופטימיזציה בשלבים אחרים בקוד:   
המתודה מצפה לשני קלטים, מטריצה התחלתית, שהיא מטריצה "ריקה" מלבד התאים שמכילים ערכים ההתחלתיים של הפתרון שהוכנס, ואת גודל הלוח. התוכנית קודם כל, תרצה לבצע כמות תיקונים בהתאם לגודל המטריצה כפי שהסברנו לעיל. בכך שכמות התיקונים תהיה פרופורציונלית לגודל הלוח נוכל להגיע להתקרב לפתרון באופן יחסי לגודל הפתרון שיש להגיע אליו – למטריצה גדולה יותר נדרשים יותר תיקונים בממוצע. בתוך לולאה שרצה כמות פעמים כגודל הלוח – ננסה קודם כל לתקן אי שוויוניים שאינם מתקיימים בפתרון נוכחי שעליו המתודה רצה, נעשה זאת ע"י החלפה בין שני האיברים שלא מקיימים את אי השוויון במטרה שאולי לאחר ההחלפה הם יקימו אותו. פעולה זו נוחה מכמה סיבות: היא אינה "מבלגנת" את הפתרונות שהגענו אליהם עד כה בריצה, ושומרת על עקרון השורות התקינות לאורך הריצה. כלומר היא לא משנה את כמות הפעמים שמופיעה כל ספרה בשורה, אלא רק מחליפה את הסדר בניהם, אם אפשר ושניהם לא מקובעים. באם התוכנית נתקלה בשני ערכים שרוצה להחליף בניהם כדי לענות על אי השוויון, ואחד מהם מקובע והשני לא – התוכנית מחפשת ערך אחר שניתן להזיז אותו שהוא באותה שורה וגדול/קטן מן הערך שעליו אנו לא עונים – על פי הצורך. מבחינתנו בקוד left מציין את הערך שנמצא משמאל לאי השוויון ( גם אם מאונך) והוא בעצם הערך הגדול יותר כי הסימן הוא "<" בלבד לכל הלוח, ואילו הערך הright הוא הערך הקטן שנמצא מימין לאי השוויון. אם גם פעולה זו לא הצליחה, אז ננסה לתקן עמודות. אך אם נחליף גם כן רק את הערכים שלא עונים על אי השוויון כמו במקרה שהם באותה השורה, אולי נתקן את אי השוויון אבל כנראה נפגע בתקינות השורות שהחלפנו ערכים ביניהן. לכן, בחרנו לשחלף בין השורות כשהן מוחלפות כבלוק אחד. כלומר השורה התחתונה עכשיו תהיה העליונה, וההפך לגבי העליונה. את כל זה נעשה כל עוד מתקיימים שני תנאים – 1. לא סיימנו את מכסת התיקונים שאנו רוצים לבצע ללוח ו2. שנשארו עוד תנאים לתקן.   
**השוואה בין האסטרטגיות השונות:**באופן כללי ניתן לקבוע כי האלגוריתם הגנטי שמימשנו אכן מוצא פתרון לבעיית חידת המספרים פוטושיקי- Futoshiki לגדלי מטריצות של 5 בשלושת האסטרטגיות השונות (רגיל, דרוויני ולמארקי), ולגדלי מטריצות של 6 רק בשתי האסטרטגיות – דרוויני ולמארקי. תוך בדיקת הריצה עם קלטים שונים, ופרמטרים שונים כמו כמות הלוחות שאנו שומרים מן הדור הקודם, או כיצד קצב המוטציות משתנה ובאיזה משתנים הוא תלוי ראינו כי באופן כללי כמות הצעדים עד למציאת הפתרון משתנה, גם עבור אותם פרמטרים, מכיוון היא שתלויה ברכיב הסתברותי שהוא חלק מהאלגוריתם. אך כמובן שלגודל הלוח ישנה השפעה גדולה על כמות הצעדים על פתרון אם בכלל. ע"פ הביצועים שניתחנו על 3 גדלי לוח, כלל שהלוחות קטנים יותר קל יותר להגיע לפתרון. בנוסף, חשוב לנו לציין כי היו מקרים שהאלגוריתם פתר לוח בגודל של 7, אך זה היה במקרה של הרצה לאורך 17,500- וקרה לעיתים רחוקות. בקוד שאנו מגישים הקוד נפסח לאחר כ5,000 דורות ולכן סביר כי לא נראה הצלחה של פתרון לוח 7X7. לא רק כמות הצעדים מושפעת מגודל הלוח, אלא אם בכלל ניתן להגיע להתכנסות לכדי פתרון חוקי – באלגוריתם שאנחנו בנינו קלט של לוח שהוא בגודל 7 אינו צלח ולא הגענו לתשובה לגבי לוח תקין כלשהו. כדי לנסות כן להגיע להתכנסות – **ניסינו לשנות את כמות מוטציות\*(מפורט למטה) ,** לאפשר שמירה של כמות פתרונות טובים מהדור הקודם (כמו זקנים שחיים יותר בשל עמידות טובה יותר) ועדין לא צלחנו להגיע לפתרון. המחשה של שינוי הציון הטוב ביותר והציון הממוצע באוכלוסייה לאורך הדורות – 2 גרפים נפרדים. השוואה בין 3 סוגי האלגוריתמים השונים, עבור דרגות קושי שונות קל וטריקי – וגדלים שונים של הלוח לכל דרגת קושי – 5,6,7.   
לכל אסטרטגיה נבדקו הציון הממוצע הכי טוב לדור, והציון הכי טוב שנרשם לדור – ב3 גדלים שונים של מטריצות ו2 דרגות קושי לכל אחד:   
**1. אלגוריתם גנטי רגיל:**

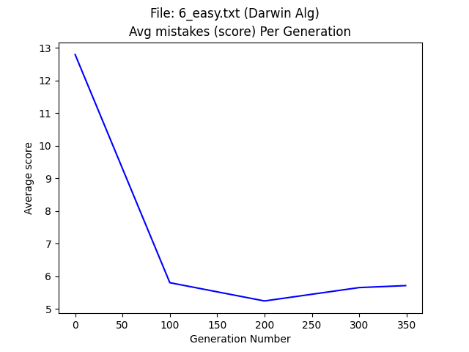
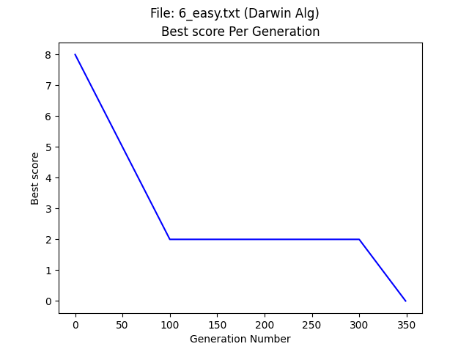
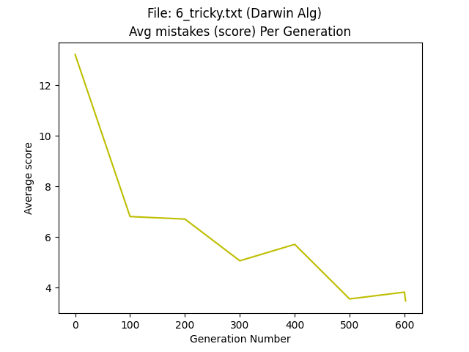


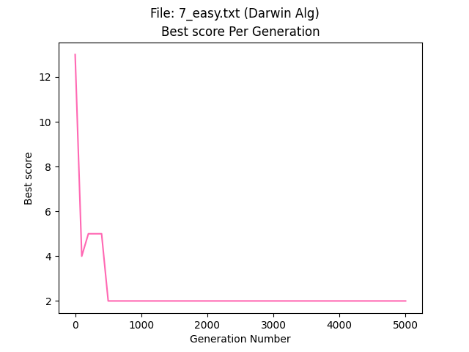
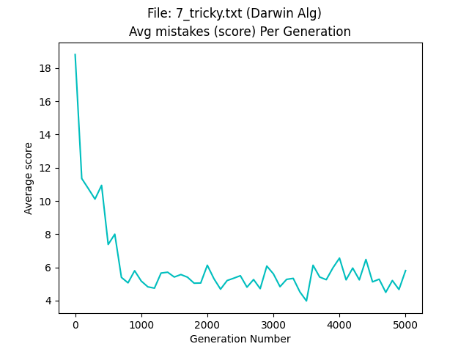
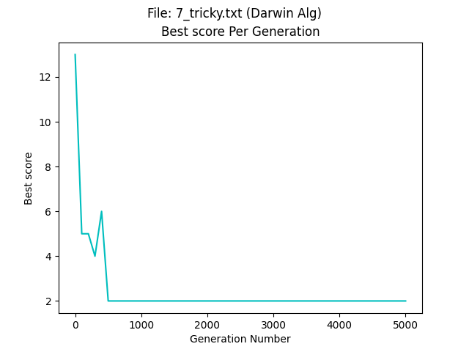
****

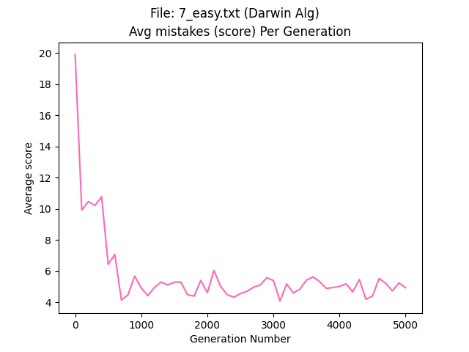
****

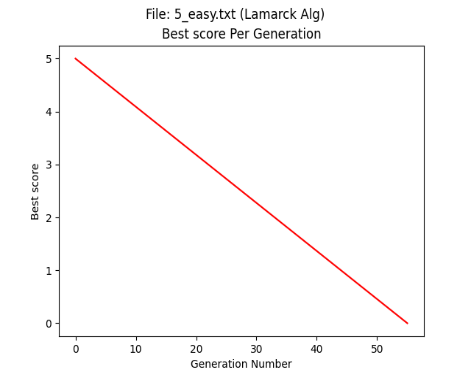
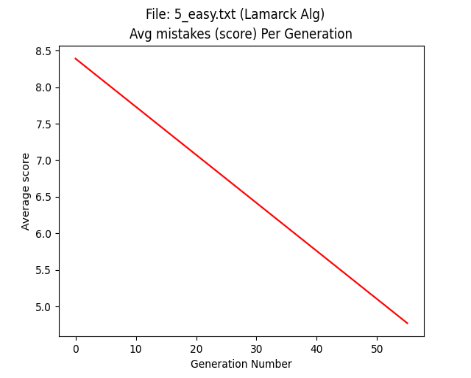
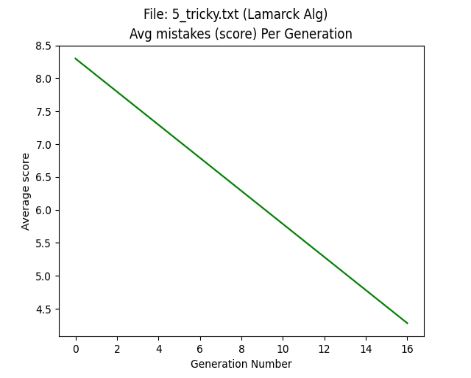
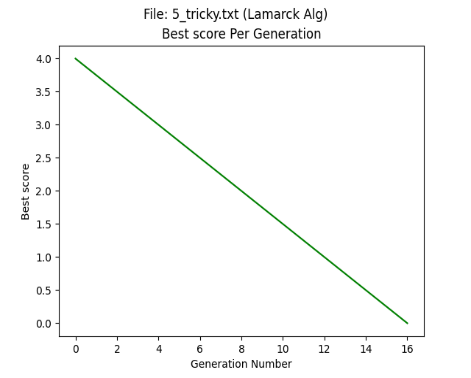
 **2. אלגוריתם גנטי דרוויני:**

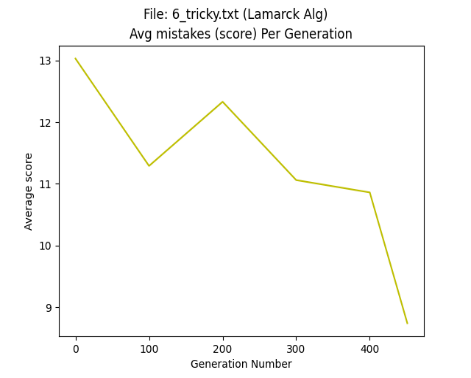
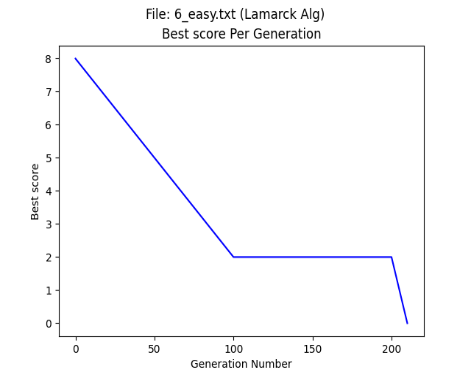
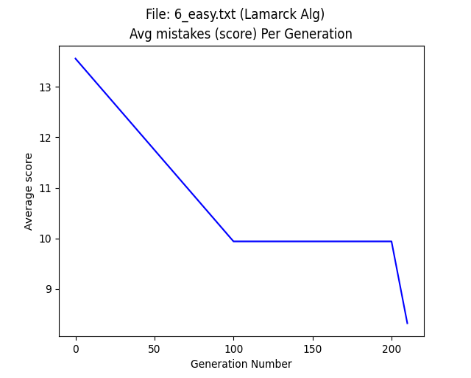


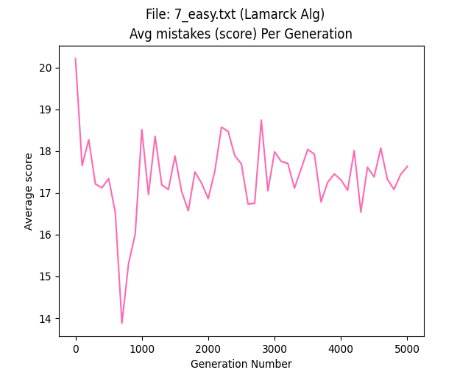
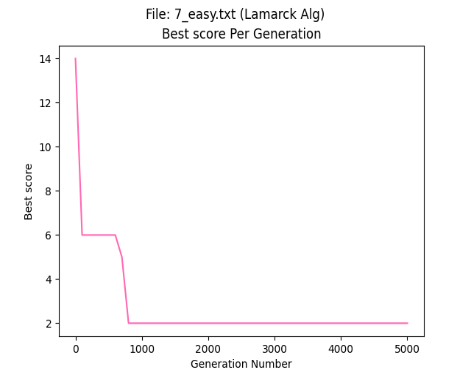
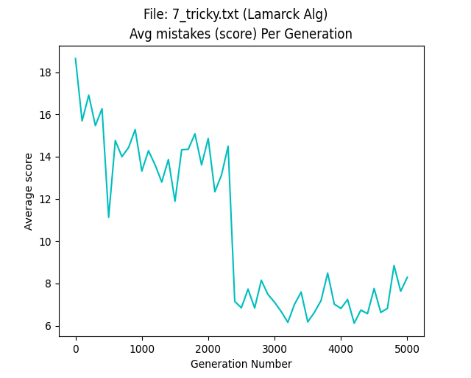
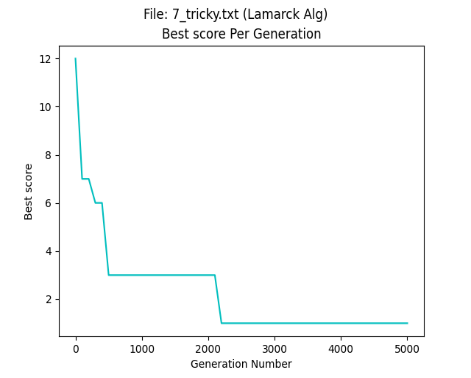






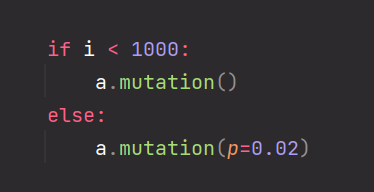
  
 **3. אלגוריתם גנטי למארקי:**





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| קושי | גודל | **גנטי רגיל** | **דרוייני** | **למארקי** |
| easy | 5X5 | הגיע ללוח סופי לאחר **כ140 קריאות**, ציון ממוצע **6.5** כשהאלגוריתם הסתיים | הגיע ללוח סופי לאחר **70 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 5.5 | הגיע ללוח סופי לאחר **50 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 4.5 |
| tricky | 5X5 | הגיע ללוח סופי לאחר **470 קריאות,** ציון ממוצע **4** כשהאלגוריתם הסתיים. | הגיע ללוח סופי לאחר **8 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 4 | הגיע ללוח סופי לאחר **16 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 5.5 |
| easy | 6X6 | לא הגיע ללוח חוקי. הציון הטוב **ביותר שהגיע אליו הוא 1 -טעות אחת.** הגיע אליו כבר אחרי 500 קריאות אך לא הצליח לעלות מאז. ציון ממוצע הטוב ביותר הוא 4, אל האלגוריתם סיים עם 6.2 | הגיע ללוח סופי לאחר **350 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 5.8 | הגיע ללוח סופי לאחר **215 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 8 |
| tricky | 6X6 | לא הגיע ללוח חוקי. הציון הטוב **ביותר שהגיע אליו הוא 1, טעות אחת.** הגיע אליו לאחר 2700 קריאות. ציון ממוצע הטוב ביותר היה 5.7 לאחר 3000 קריאות. | הגיע ללוח סופי לאחר **600 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 3.8 | הגיע ללוח סופי הגיע לאחר **470 קריאות** לפונקציית הערכה, ועם ציון ממוצע של 5.8 |
| easy | 7X7 | לא הגיע ללוח חוקי. הציון הטוב **ביותר שהגיע אליו הוא 4 (טעויות),** הגיע אליו לאחר כ2050 דורות, אך לא הצליח לעלות מאז. ציון ממוצע הנמוך ביותר היה 15, והאלגוריתם סיים עם ציון ממוצע של 18.5 | לא הגיע ללוח חוקי. הציון הטוב **ביותר שהגיע אליו הוא 2 (טעויות),** הגיע אליו כבר אחרי כמעט 600 קריאות אך לא הצליח לעלות מאז. ציון ממוצע הנמוך **ביותר 4 והגיע** אליו כבר לאחר 700 קריאות. | לא הגיע ללוח חוקי. הציון הטוב ביותר **שהגיע אליו הוא 2 (טעויות),** הגיע אליו כבר בקריאה ה850 בערך, אך לא הצליח לעלות מאז. ציון ממוצע נמוך ביותר ביותר **היה 14** ונרשם לאחר 800 קריאות. |
| tricky | 7X7 | לא הגיע ללוח חוקי. הציון הטוב **ביותר שהגיע אליו הוא 4 (טעויות),** הגיע אליו כבר לאחר 1500 קריאות ציון ממוצע הנמוך ביותר היה 14, אך האלגוריתם סיים עם ציון ממוצע של 16.7. | בד"כ לא הגיע ללוח חוקי. הציון הטוב **ביותר שהגיע אליו הוא 2 (טעויות),** הגיע אליו לאחר כ500 קריאות בלבד אך לא הצליח לעלות מאז. ציון ממוצע היה 5.8 כשהאלגוריתם נסגר, אך הטוב ביותר היה **4,** לאחר 3500 קריאות. | לא הגיע ללוח חוקי. הציון הטוב ביותר שהגיע אליו **הוא 1 -טעות אחת,** הגיע אליו לאחר 2000 וקצת קריאות, אך לא הצליח לעלות מאז. ציון ממוצע הנמוך ביותר היה **6** נרשם לאחר כ3000 קריאות, אך סיים עם ציון 8. |

אם נעשה ממוצע לכמות הצעדים שלקח להגיע ללוח חוקי פתור עבור כל אחת מהאסטרטגיות (נזנח את אי ההצלחות) , לקח בממוצע **187.75 קריאות לאלגוריתם הלמארקי** (אופטימיזציה ופו' הערכה לפני הכלאה), **257 קריאות לאלגוריתם הדרוויני** (אופטימיזציה ופו' הערכה אחרי הכלאה) **וכ305 קריאות עד ללוח חוקי עבור האלגוריתם הרגיל** (ללא אופטימיזציה). כלומר, נגיד כי האלגוריתם הלמארקי, בו ירושה של תכונות נרכשות הוא חלק מהמימוש, באמת מביא למציאת פתרון מהר יותר, משפר את הסיכויים פר לוח, להיות מתאים יותר, לאחר מכן האלגוריתם של דרווין, שמאפשר אופטימיזציה אך ללא קשר ישיר לדור הבא, ולבסוף האלגוריתם הגנטי הרגיל שיישמנו שהיה מוגבל ביכולתו לפתוח לוחות גדולים מ5 ולכן אם היו שואלים אותנו באיזה אלגוריתם כדאי להשתמש כדי לפתור את בעית המשחק, היינו בוחרים בסדר זה, נמליץ על למארק , אז על דרווין ואת האלגוריתם הגנטי הרגיל נתעדף אחרון.   
  
היו פרמטרים של האלגוריתם עצמו ששינו את מהירות האלגוריתם להגיע לפתרון עם ציון 0, אך אלו עם ההשפעה הניכרת ביותר היו:

1. ההתייחסות למוטציות   
2. השמירה של כמות מסויימת של יצורים=לוחות לדור הבא.   
מבחינת כמות היצורים שאנו שומרים כפי שהם לדור הבא, החלטנו על כמות של 5. התחלנו עם 10 לכולם, אך בניסיונות להגיע לשיפור בכמות הקריאות, ראינו כי לשמור רק 5 ממש מגבירה את מהירות האלגוריתם להגיע ללוח חוקי (ייצור = לוח). מבחינת המוטציות, לאחר ניסיונות רבים לשליטה בכמות המוטציות **כדי למנוע התכנסות מוקדמת,** וכדי לשפר את ריצת האלגוריתם מבחינת זמן, עלינו על הטובה ביותר בשביל האלגוריתם שלנו, וגם היא משתנה בין הגדלים של הלוחות והקושי שלהם. העיקרון היה, דווקא לא להוריד את כמות המוטציות באופן ליניארי עם הזמן, כי במקרה הגרוע, יכול להיות שנשאר עם פתרון שקרוב מאד להיות מושלם, אך כמעט בלי יכולת להשתנות, ולכן עם סיכוי זעיר להגיע לפתרון מושלם למרות שהניקוד שלו הוא 1 בלבד. כמובן שלעיתים היו ריצות שבהן אלגוריתמים הגיע לפתרון "מקרי" לאחר כמות מזערית של קריאות, וכשהציון הטוב ביותר היה גבוה מ6, אך לאורך ניסיונות רבים הבנו כי זה קורה בהסתברות נמוכה, וכי הגדרת האלגוריתם לא תמיד מביאה לשינוי ליניארי בתפקוד התוכנית. מפאת חוסר המקום לא נצרף לכאן הרבה דוגמאות למקרים בודדים שסותרים את תוצאות האלגוריתמים בתוחלת. **\*תמונה מן הקוד כדי לתאר את השליטה בכמות המוטציות:** כפי שדנו לעיל, שליטה בכמות המוטציות הייתה חמקמקה מבחינת ההשפעה על ביצועי האלגוריתמים, השפיעה אחרת על כל מקרה. בסה"כ הופתענו לגלות כי כדי למנוע התכנסות מאד מוקדמת דווקא עבד לנו בצורה טובה יותר לשלוט במוטציות בצורה הזו: אם מספר הדורות קטן מ1000 נתחיל בסיכוי הדיפולטיבי למוטציה שהוא 0.05 שהוא שיעור מוטציה גבוה הנועד למנוע התכנסות מוקדמת. לאחר שלב זה נוריד את הסיכוי להגריל 1 בפונקציית המוטציה ל0.02 כדי להישאר קרוב לפתרון, על מנת לא להתבדר – נוותר על מניעה של התכנסות בשלב זה. כך הפתרון יצא לנו היעיל ביותר, עם זמן מציאת פתרון הנמוך ביותר.

לקוראים ובאם נמשיך לעבוד עם אלגוריתם זה, כמובן שיכולים להיות בו עוד התאמות מבחינת כמות המוטציות – ומפאת חוסר זמן לא יכולנו לבדוק את כל הצירופים של הבדיקות ואיך משפיעים על תוצאות האלגוריתמים.  
יכול להיות שהיה שווה דווקא להעלות את הסיכוי למוטציה אף ביותר, בשלב שאם לא קרובים לפתרון מושלם, ואילו אם אנחנו בציון שהוא לא רחוק מאד מ0 אך גם עדין לא מספיק קרוב, אולי עדין נעלה את ההסתברות למוטציות במעט כדי לקדם תהליך. אולי בהעלאה כזו היינו יכולים להגיע לפתרון גם עבור לוח בגודל 7.  
בנוסף, עדיין לא פענחנו מדוע האלגוריתמים עם האופטימיזציה, במיוחד למארק, לא הצליחו לפתור לוח בגודל 7, לאורך 50,000 הרצות, כי כל העניין של אופטימיזציה הוא ממש להביא לשיפור הלוחות.  
אם היה לנו זמן נוסף היינו בודקים זאת יותר לעומק.