בסייעתא דשמיא

דו"ח ביולוגיה חישובית- תרגיל 2

פתרון חידת פוטושיקי- Futoshiki ע"י אלגוריתם גנטי

מגישות: שיר בן אהרון, תמר סעד

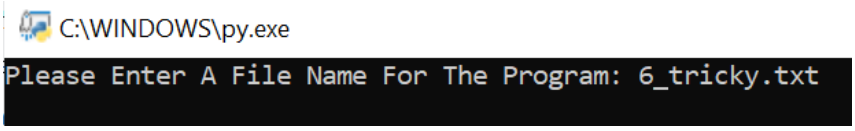
הוראות הרצה לקוד:

כדי להריץ את התוכנית יש צורך להריץ את קובץ ה- python הנתון באמצעות **דאבל קליק** על הקובץ מהתיקייה בה הוא נמצא, או באמצעות **windows cmd** ע"י שימוש בפקודה- python ex1.py. (אנחנו מניחות שקיים python על המחשב והוא מקושר ל-cmd).

<u>התוכנית מורידה בעצמה חבילות נדרשות להרצתה, ולא דורשת פרמטרים חיצוניים לפקודת ההרצה.</u>

(בשל הנאמר מעלה והורדת החבילות, יתכן כי ההרצה הראשונה תיכשל ולכן יש לנסות להריץ פעם נוספת).

- קלט מהמשתמש- בתחילת ההרצה המשתמש מתבקש להכניס את שם קובץ ה-txt אותו הוא מעוניין לטעון לתוכנית (קובץ עם רשימת האילוצים). <u>יש לתת את השם המלא עם הסיומת המתאימה לקובץ באופן הבא:</u>



- אם קובץ הקלט לא נמצא בתיקייה שבה ממוקם קובץ הריצה, יש להכניס בשם הקובץ את הנתיב המלא אליו.

תיאור הפלט:

- ריצה והדפסות- התוכנית מריצה את שלושת האלגוריתמים- Regular, Lamarck, Darwin בזה אחר זה לאורך 10,000 דורות, ומדפיסה למסך כל מאה דורות את שלושת ציוני ה-fitness הטובים ביותר ואת ממוצע ציוני ה-fitness של הדור הנוכחי.
- בסיום הריצה נשמרים בתיקייה ממנה הורצה התוכנית עבור כל אחד משלושת האלגוריתמים-

1. הלוח המייצג את הפתרון הטוב ביותר עבור כל אלגוריתם, ציון ה-fitness של הלוח, ציון ה-fitness הממוצע ומספר הדורות שלקחו להגעה לפתרון.

2. גרף המציג את ערך ה-fitness הטוב ביותר ביחס לממוצע ערכי ה-fitness באוכלוסייה לאורך הדורות.

<u>חלק א:</u>

מימוש האלגוריתם הגנטי:

- **גודל האוכלוסייה-** גודל האוכלוסייה שבחרנו לעבוד איתו הוא 100 לוחות המיוצגים בכל דור על ידי 100 מטריצות בגודל הקלט שהוזן לתוכנית.

- **מספר הדורות-** החלטנו להריץ כל אחד משלושת האלגוריתמים לאורך 10,000 דורות. במידה ונמצא פתרון ללוח לפני כן האלגוריתם עוצר ומדווח על כך למשתמש. בחרנו במספר דורות זה כיוון שראינו כי ישנו שיפור בערכי ה-fitness לאורך מספר ארוך יחסית של דורות כך שלעיתים מתקבל פתרון לאחר מספר דורות רב.

- **ייצוג הפתרונות-** הפתרונות מיוצגים כמטריצה של N*N בהתאם לגודל הלוח שהוזן כקלט. אילוצי המספרים ההתחלתיים שהוזנו כקלט הוכנסו למטריצת בסיס התחלתית איתה עובד האלגוריתם.
כל לוח בדור ההתחלתי נבנה על ידי בחירה אקראית למיקומים של המספרים מ-1-N בכל שורה, כך שאילוצי השורות יתקיימו תמיד. כך למעשה האילוצים הנוספים שנותר לפתור בבעיה הם רק על העמודות ועל סימני ה-">".

- **פונקציית ההערכה-** פונקציית "evaluate" היא הפונקציה שקובעת את ה-fitness. הפונקציה למעשה סופרת על כל לוח המייצג פתרון את האילוצים שאינם מתקיימים ומובילים לסתירה כלשהי. כל אילוץ שאינו מתקיים הוסיף נקודה לערך ה-fitness. מכאן נובע כי ככל שערך ה-fitness נמוך יותר הפתרון מוצלח יותר, ובהתאם פתרון לבעיה מתקבל כאשר ערך ה-fitness = 0 ובשלב זה עצרנו את ריצת האלגוריתם והחזרנו את לוח הפתרון.

כיוון ששורות המטריצות בהכרח מקיימות את האילוצים, על מנת לחשב את קיום האילוצים הנוספים ביצענו מעבר אחד על עמודות המטריצה ועל כל מופע של מספר כלשהו שמופיע יותר מפעם אחת הוספנו נקודה לציון. לאחר מכן עברנו על אילוצי סימני ה-"<" וגם כאן על כל שני מספרים שלא מקיימים את האילוץ הוספנו נקודה.

- **בחירת הדור הבא-** מתרחשת בפונקציה- "next_generation". כיוון שהגדרנו כי גודל האוכלוסייה הוא 100 בכל דור יצרנו מערך של 100 פתרונות באופן הבא:
  - שמרנו את 25 הלוחות בעלי ערך ה-fitness הנמוך ביותר מהדור הקודם, כלומר הלוחות הטובים ביותר.
  - הגרלנו 15 לוחות חדשים לגמרי על מנת להכניס גיוון לפתרונות ולמנוע התכנסות מוקדמת.
  - את 60 הלוחות הנותרים לדור יצרנו ע"י ביצוע הכלאות בין 40 הלוחות המתוארים לעיל.

- **ביצוע הכלאות-** פונקציית "get_crossovers" מקבלת רשימה של 40 מטריצות- 25 מתוכן הטובות ביותר מהדור הקודם ו-15 הן חדשות לגמרי. הפונקציה מבצעת הכלאות ומחזירה 60 מטריצות חדשות שיתווספו לדור הבא.

  על מנת להגדיל את ההסתברות שמטריצה טובה תבחר להכלאה יצרנו מערך בגודל 120 אליו הכנסנו-
  - 5 המטריצות הטובות ביותר מופיעות 8 פעמים כל אחת.
  - 15 המטריצות לאחר מכן מופיעות 4 פעמים כל אחת.
  - שאר המטריצות מופיעות פעם אחת כל אחת.

  לאחר מכן מתבצעת הגרלה רנדומלית של שני מספרים ממערך הלוחות בגודל 120 המייצגים את שתי המטריצות שיעברו הכלאה- "ההורים", ויצרו מטריצה נוספת לדור הבא.

  לטובת הכלאת שתי המטריצות, נבחר מספר רנדומלי בין אינדקס 0 לאינדקס N-1 המסמן את השורה בה יבוצע "חיתוך" בין המטריצות- עד שורה זו יהיו השורות של מטריצה אחת ומשורה זו והלאה יהיו השורות של המטריצה השנייה.

- **ביצוע מוטציות-** פונקציית "create_mutations" מקבלת רשימה של כל הלוחות לדור הבא- 100, ואת ההסתברות לביצוע מוטציה בלוח אותה קבענו להיות- 0.1. כך מספר המוטציות שמתרחש בפועל תלוי בגודל הלוח המתקבל. את 4 הלוחות הטובים ביותר השארנו מבלי לבצע בהם מוטציה, על מנת שציון הלוח הטוב ביותר יוכל רק לרדת ולא לעלות.

  הפונקציה בוחרת באקראי שורה ועמודה, מוודאה שבמיקום הנבחר אין מספר שהתקבל כחלק מאילוצי הלוח המקוריים, מגרילה בשורה שנבחרה מספר אחר (כדי להמשיך לשמר את אילוצי השורות) ומבצעת החלפה ביניהם.

- **בעיית ההתכנסות המוקדמת-** כדי להימנע מהגעה להתכנסות מוקדמת הכנסנו באלגוריתם את רכיב הלוחות האקראיים החדשים שנוצרים בכל דור, זאת על מנת להכניס גיוון לרשימת הלוחות. בנוסף, במהלך ריצת האלגוריתם מימשנו התחלה מחודשת של האלגוריתם לאחר 5,000 דורות, זאת משום שראינו כי לרוב לאחר מספר דורות זה אם לא נמצא עדיין פתרון ישנה התכנסות למינימום מקומי ולכן בחרנו לאתחל את כל הלוחות ולהתחיל מחדש.

<u>חלק ב:</u>

בחלק זה יצרנו שתי פונקציות נוספות בעלות פונקציונליות של אופטימיזציה על גבי האלגוריתם הגנטי הקיים.

אלגוריתם Lamarck:

כל פתרון עובר אופטימיזציה וה-fitness שלו נקבע אחריה. הדור הבא נוצר מהפתרונות וציוני ה-fitness שהתקבלו אחרי האופטימיזציה.

אלגוריתם Darwin:

כל פתרון עובר אופטימיזציה וה-fitness שלו נקבע אחריה. הדור הבא נוצר מהפתרונות המקוריים לפני האופטימיזציה ומציוני ה-fitness שהתקבלו לכל פתרון אחרי האופטימיזציה.

- **פונקציית האופטימיזציה-** האופטימיזציה ללוחות מתרחשת בפונקציה "Lamarck_evaluate". הפונקציה עוברת על כל פתרון ברשימת הפתרונות ושומרת את המיקומים בלוח בהם ישנן אי התאמות של האילוצים-
  - במידה ובעמודה מסוימת יש מספר המופיע יותר מפעם אחת נשמרים במערך המיקומים הנוספים של אותו המספר, ובמקביל נשמר מערך של המספרים החסרים באותה עמודה.
  - במידה ושני מספרים שלא מקיימים את תנאי ה-"<" נשמר המיקום של אחד מהם.

בסייעתא דשמיא

רשימת המיקומים הבעייתיים בלוח משמשת כמו מעין אתרי "Hot Spots" למוטציות עבורנו כך שהפונקציה מבצעת מוטציות ייעודיות במקומות אלה בלוח. עבור כל אחד מהמיקומים המופיעים ברשימה, מוגרל המספר החסר בעמודה מאותה השורה (שוב כדי לשמור על אילוצי השורות) ובמידה ושניהם אינם מספרים שהתקבלו כחלק מאילוצי הלוח מתבצעת ביניהם החלפה.

מספר האופטימיזציות שפתרון יכול לעבור הוא או גודל הלוח שהתקבל (5,6,7) או כאורך רשימת המיקומים הבעייתיים בלוח, הקצר יותר מבין שני הפרמטרים הללו.

תוצאות:

לצורך יכולת השוואה טובה בין האלגוריתמים השונים, יצרנו מספר לוחות משחק בגדלים וברמות קושי שונות (הפרמטרים ללוחות נלקחו מהאתר שניתן בתרגיל), והרצנו את הקוד שלנו 20 פעמים. בכל איטרציה התקבל כקלט אחד מהלוחות שיצרנו, באופן אקראי.

שמרנו בטבלאות את התוצאות שהתקבלו: עבור כל סוג אלגוריתם מהו ציון הלוח הטוב ביותר שהצליח להגיע אליו, מהו ממוצע הציונים בסוף ההרצה ומה מספר הדורות בריצה (כאשר מספר הדורות = 10,000 משמע שלא נמצא פתרון).

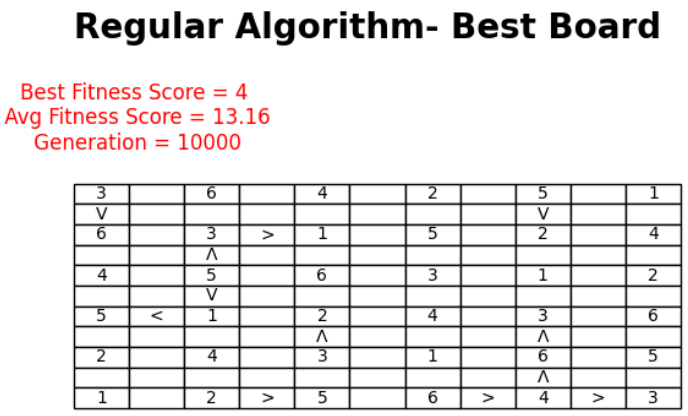
בדקנו עבור כל אלגוריתם את ממוצע התוצאות של כל הריצות, וקיבלנו כי:

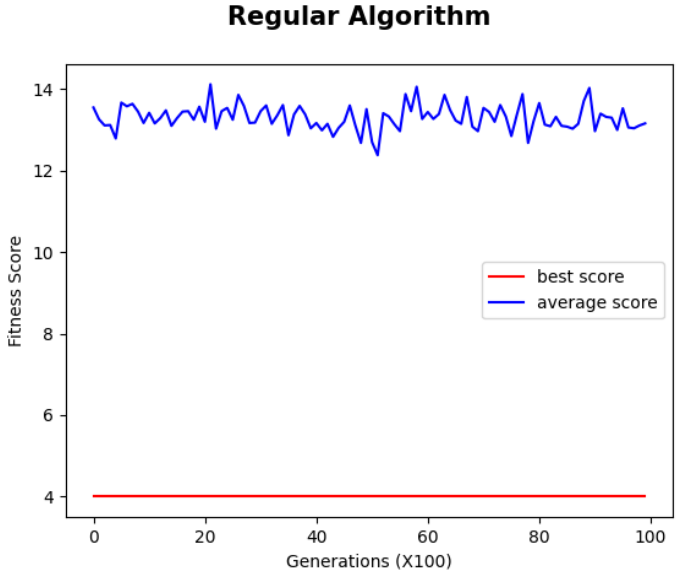
	רגיל	**למארק**	**דארווין**
ציון לוח טוב ביותר	2.1	0.7	1.9
ממוצע ציוני לוחות	11.848	7.572	9.084
מספר דורות בריצה	8575.5	6864.9	7934.25
אחוז ריצות בהן הגענו לפתרון	20%	45%	30%

כפי שניתן לראות בטבלה, את התוצאות הטובות ביותר קיבלנו מהפתרון הלמארקי, האלגוריתם הטוב ביותר אחריו היה של דארווין, והאלגוריתם הרגיל היה הכי פחות מוצלח.

בנוסף, כפי שכתבנו לעיל, כל הרצה יוצרת שני קבצי jpg. עבור כל אלגוריתם: אחד מתאר את הלוח האחרון שנוצר, והשני מראה את הציון הטוב ביותר ואת ממוצע הציונים לאורך ריצת האלגוריתם.

הבאנו פה דוגמה מהרצה אחת:

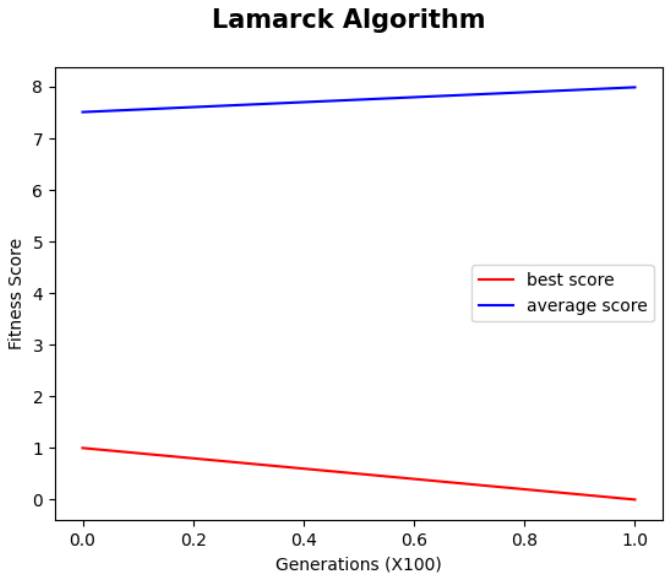




Lamarck Algorithm- Best Board

Best Fitness Score = 0
Avg Fitness Score = 7.99
Generation = 139

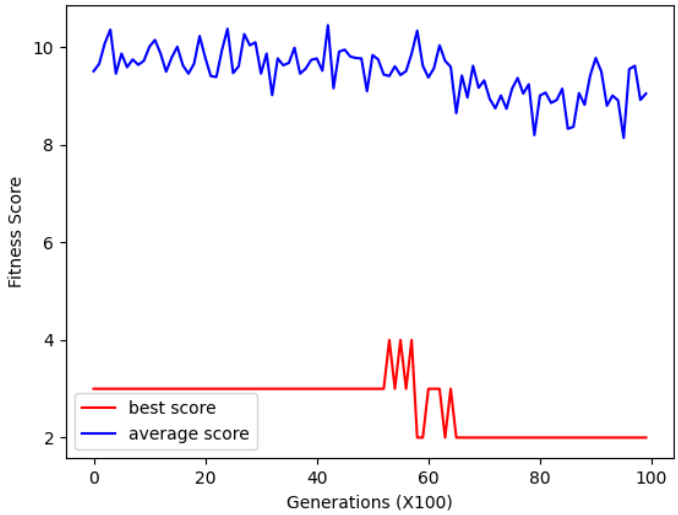
3		6		4		2		5		1
V								V		
1		4	>	2		6		3		5
		Λ								
4		5		3		1		6		2
		V								
2	<	3		5		4		1		6
				Λ				Λ		
5		1		6		3		2		4
								Λ		
6		2	>	1		5	>	4	>	3



Darwin Algorithm- Best Board

Best Fitness Score = 2
Avg Fitness Score = 9.05
Generation = 10000

2		6		4		3		5		1
V								V		
5		2	>	3		4		1		6
		Λ								
3		5		2		1		6		4
		V								
1	<	4		5		6		2		3
				Λ				Λ		
4		1		6		2		3		5
								Λ		
6		3	>	1		5	>	4	>	2



כפי שניתן לראות, האלגוריתם הרגיל והאלגוריתם של דארווין לא מצאו פתרון ולכן הלוח שמוצג אצלם אינו תקין. ניתן לראות זאת גם בכך שציון ה-fitness שלהם גדול מ-0, וכן הם רצו 10,000 דורות, כלומר מספר הדורות המקסימלי שהוקצה להם לרוץ. גרף הציון הממוצע של האלגוריתם נראה דומה בכל ההרצות אותן בדקנו: טווח תזוזה של בערך 1-+ לכל אורך האלגוריתם.
באלגוריתם הרגיל ובאלגוריתם הלמארקי ציון הלוח הטוב ביותר יכול רק לרדת ולא לעלות, ולכן גרף הציון הטוב ביותר תמיד ירד. באלגוריתם של דארווין, כיוון שאנחנו מסתכלים על ציונים שאינם של הלוחות ממש, הציון הטוב ביותר יכול גם לעלות.
האלגוריתם של למארק סיים לרוץ בתוך 139 דורות. כל נקודה בגרף מתווספת בקפיצות של 100 דורות, על מנת ליצור גרף קריא ופחות רועש (כפי שמצוין בכותרת ציר ה-Y), ולכן הגרף של למארק מכיל רק 2 נקודות. כיוון שהאלגוריתם של למארק הגיע לפתרון חוקי, ציון הלוח אצלו הוא 0 והלוח שהוא מציג הינו חוקי.

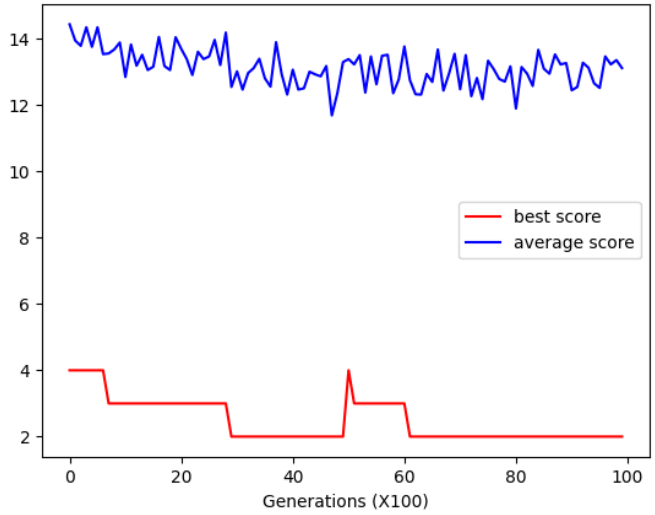
כמו כן, צירפנו גרף סיום הרצה של האלגוריתם של למארק על לוח של 7*7. האלגוריתם אמנם לא מצליח לפתור את הלוח, אבל ניתן לראות שהוא משתפר במהלך ההרצה שלו ומגיע לפתרון קרוב מאוד (בהרצה הזאת- לטעות אחת בלבד).

Lamarck Algorithm- Best Board

Best Fitness Score = 2
Avg Fitness Score = 13.12
Generation = 10000

2		3	>	1		7		5		6		4
V								V		V		Λ
1		5		7		4		3	>	2		6
						Λ						
4		2		5		6		1		7	>	3
		V										
6		1		2		3		7		4		5
						Λ		V				
3	<	4		6		5		2	>	1	>	7
Λ		Λ										
7		6		3	>	1	>	4		5	>	2
												V
5		7		4		2		6		3		1

Lamarck Algorithm



לסיום, נרצה לציין כי אף על פי שהדו"ח שלנו חרג במקצת מההקצאה ל-4 עמודים, נאמר בהרצאה שמותר ליצור דו"ח ארוך מהנאמר בתרגיל אם הדו"ח התארך עקב שימוש בגרפים.