אלעד בעל צדקה 312531973
חן לארי 209192798

דו"ח ביולוגיה חישובית

הוראות הרצה:

צריך להריץ את קובץ main.

לסעיף א:

על מנת להריץ את האלגוריתם בשיטה הקלאסית יש להריץ את

```
key, dec, steps = ga.run(max_generations)
```

לסעיף ב:

על מנת להריץ בשיטת דארווין יש להכניס לאובייקט GA את הפרמטר lemark=False

```
ga = GA(mutation_rate=mutation_rate, lemark=False)
```

ועל מנת להריץ בשיטת למארק יש להכניס לאובייקט GA את הפרמטר lemark=True

```
ga = GA(mutation_rate=mutation_rate, lemark=True)
```

ואז להריץ

```
key, dec, steps = ga.runSectionB(max_generations)
```

בנוסף, יש לייבא את הספריות הבאות: numpy, string, collections, random, matplotlib.pyplot, copy.

איך יוצגו הפתרונות?

הפתרונות יוצגו על-ידי מחרוזת בגודל 26 שמגדירה את הצופן שעל-פיו נוצר הטקסט המוצפן.
כך שאם במקום ה-'a' במחרוזת (מיקום 0) נכתבה האות 'w' אזי בהצפנה האות 'a' מיוצגת כ-'w' ולכן בפענוח האות 'w' תיוצג כ-'a'.
כמובן שבכתיבת הקובץ perm.txt ייצגני את הפתרון כפי שהתבקשנו.

אות אמיתית	צופן
y	A
x	B
...	...

מהי פונקציית ההערכה?

פונקציית ה-fitness שלנו מחשבת טעות יחסית לפי 3 שלבים:

1. ההפרש בין התדירות של האותיות בטקסט המוצפן לבין תדירות האותיות בשפה האנגלית.
2. ההפרש בין התדירות של שתי אותיות בטקסט המוצפן לבין תדירות של שתי אותיות בשפה האנגלית.
3. כמות המילים שפענחנו ואינן מופיעות ברשימת המילים הנפוצות באנגלית.

כל אחד מאלה הוא יחסי, כלומר מתחלק בסכום הכולל.
פונקציית ההערכה היא הסכום שלהם ואנו מנסים למזער אותה.

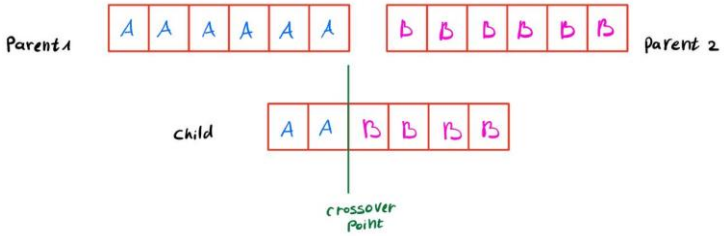
אלעד בעל צדקה 312531973
חן לארי 209192798

```python
def fitness(self, key):
    self.fitness_counter += 1
    decrypted_text = self.ciphertext.translate(str.maketrans(key, string.ascii_lowercase))
    fitness = 0

    # Calculate the frequency distribution of letters in the decrypted text
    decrypted_letter_count = collections.Counter(decrypted_text.lower())
    total_letters = sum(decrypted_letter_count.values())
    decrypted_letter_frequencies = {letter: decrypted_letter_count[letter] / total_letters for letter in self.letter_frequency}
    value = sum(abs(decrypted_letter_frequencies[letter] - self.letter_frequency[letter]) for letter in self.letter_frequency)
    fitness += value / len(self.letter_frequency)

    # Calculate the frequency distribution of two letters in the decrypted text
    decrypted_bigram_count = collections.Counter([decrypted_text[i:i + 2].lower() for i in range(len(decrypted_text) - 1)
                                                  if len(decrypted_text[i:i + 2].replace(" ", "").replace(".", "")
                                                  .replace(",", "").replace(";", "").replace("\n", "")) == 2])
    total_bigrams = sum(decrypted_bigram_count.values())
    decrypted_bigram_frequencies = {bigram: decrypted_bigram_count[bigram] / total_bigrams for bigram in self.bigram_frequency}
    value = sum(abs(decrypted_bigram_frequencies[bigram] - self.bigram_frequency[bigram]) for bigram in self.bigram_frequency)
    fitness += value / len(self.bigram_frequency)

    # Calculate the presence of common words in the decrypted text
    decrypted_words = decrypted_text.lower().split()
    value = sum(1 for word in decrypted_words if word not in self.common_words)
    fitness += value / len(self.common_words)
    return fitness
```

איך ביצעתם את פעולת הcross-over בין הפתרונות?

פעולת ה cross-over שלנו מתחלקת לכמה חלקים.

1. בחירת כל אחד משני ההורים על-ידי טורניר.
בתהליך זה בוחרים באופן רנדומלי מספר קבוע (היפר-פרמטר) של מתמודדים מהאוכלוסייה, והם "מתחרים" ביניהם ומתוכם נבחר הכי טוב (בעל תוצאת ה-fitness הטובה ביותר).
2. תהליך הcross-over.
לאחר שנבחרו שני ההורים, אנו מגרילים נקודת חיתוך שיוצרת ילד כך שעד אותה נקודה מועברים אליו הערכים של ההורה הראשון, והחל מאותה נקודה מופיעים הערכים של ההורה השני.
3. תיקון שגיאות.
יכולים להיווצר ילדים כך שאות אחת מופיעה כמה פעמים, וזו לא הצפנה נכונה. אנו מתקנים זאת.



כיצד מומשו מוטציות?

בחרנו באופן רנדומלי 2 אינדקסים בין 0 ל-26, והחלפנו בין מיקומים אלה במחרוזת.

האם/איך התייחסתם לבעיית ההתכנסות המוקדמת?

כדי להימנע (כמה שאפשר) מהתכנסות מוקדמת נקטנו כמה צעדים:

1. הגדלנו את כמות האוכלוסיה.
2. הגדלנו את אחוז המוטציות באוכלוסיה.

כמובן שעדיין יש הרצות עם התכנסות מוקדמת אך פחות.
יש לזכור שזהו אלגוריתם לא דטרמיניסטי ולכן בכל מקרה לא תמיד נקבל פענוח טוב.

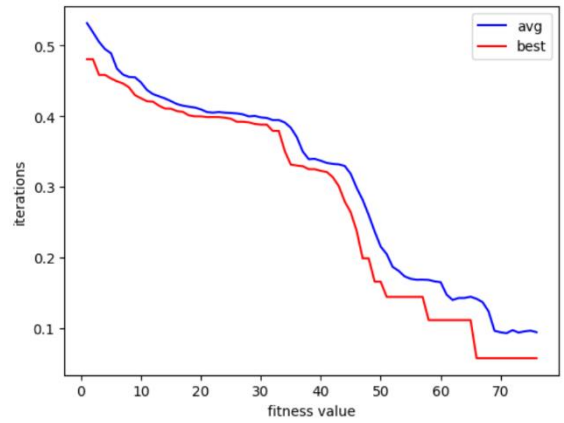
איך החלטתם מתי לעצור את הריצה?

אלעד בעל צדקה 312531973
חן לארי 209192798

כאשר לא היה שיפור בערך ה-fitness של הפענוח הטוב ביותר במשך מספר קבוע (היפר-פרמטר) של איטרציות, הפסקנו את הריצה.

השוואה בין האלגוריתם הקלאסי לאלגוריתם הדארוויני או הלמארקי:

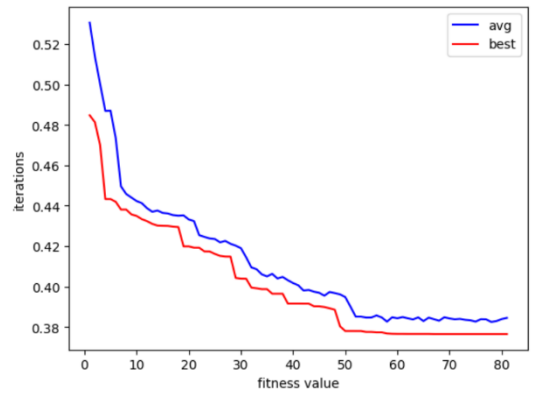
נראה כמה הרצות של האלגוריתם במקרה הקלאסי:




```
number of steps:  15200
```

```
a y
b x
c i
d n
e t
f o
g z
h j
i c
j e
k b
l l
m d
n u
o k
p m
q s
r v
s p
t q
u r
v h
w w
x g
y a
z f
```

במקרה זה קיבלנו פענוח קריא של הטקסט, אנו מניחים שזה הפענוח הנכון- כלומר הצלחנו לפענח תוך 15200 צעדים.




```
number of steps:  16200
```

```
a s
b k
c i
d n
e r
f o
g x
h z
i c
j d
k u
l l
m y
n m
o j
p w
q t
r v
s p
t q
u h
v e
w b
x g
y a
z f
```

במקרה זה הפענוח פחות הצליח.

נראה הרצות של האלגוריתם של למארק והאלגוריתם של דארווין.

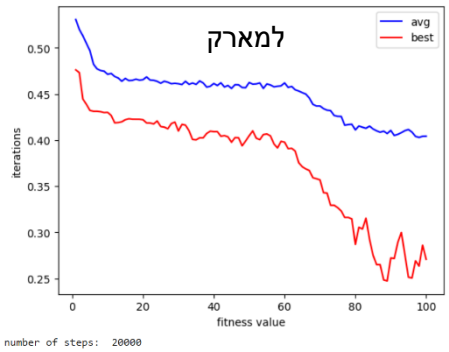
אלעד בעל צדקה 312531973
חן לארי 209192798

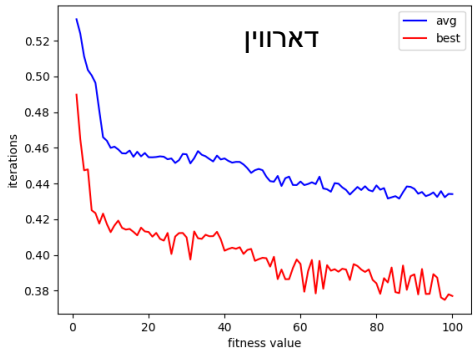
לקחנו את רעיון האופטימיזציה מהתרגיל ותחילה הצבנו N=3.

ראינו כי לשתי השיטות לוקח זמן להגיע עד להתכנסות, כמות האיטרציות ומספר הצעדים היו גבוהים.
אצל שיטת למארק ההתחלה הייתה נראית מבטיחה, אבל בהמשך השינויים היו כנראה דרמטיים מידי והאלגוריתם לא התכנס.
אצל שיטת דארווין, נראה כי האלגוריתם אינו קרוב כלל לתוצאה.

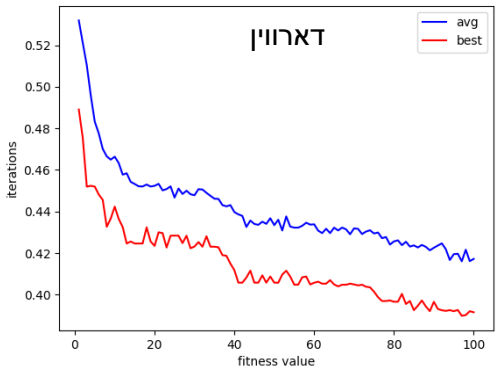
לכן שינינו ל N=2.

גם כאן הגענו לכמות איטרציות גבוהה ועדיין לא סיימנו. ננסה להקטין את אחוז המוטציות באוכלוסיה מ-30% ל-10%.



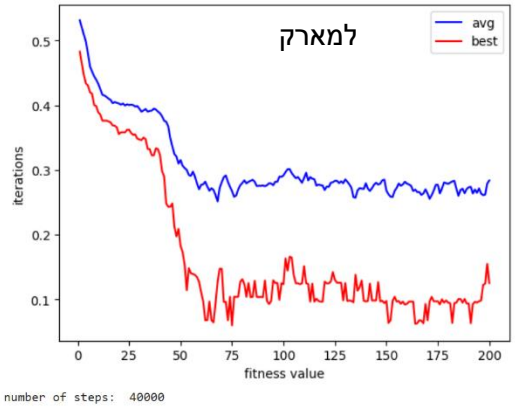


ובכל זאת עדיין האלגוריתמים לא הגיעו להתכנסות. ננסה N=1



האלגוריתם של דארווין כלל לא קרוב. נראה שייקח לו עוד המון איטרציות להגיע לתוצאה.
לעומתו ללמארק לוקח יותר זמן אך הצליח להתקרב לתוצאה:

אלעד בעל צדקה 312531973
חן לארי 209192798



a	y
b	x
c	i
d	n
e	t
f	o
g	q
h	z
i	c
j	e
k	b
l	l
m	d
n	u
o	k
p	m
q	s
r	v
s	p
t	j
u	r
v	h
w	w
x	g
y	a
z	f

באופן כללי, לאחר מספר הרצות של האלגוריתמים השונים, הגענו למסקנה כי האלגוריתם הכללי מביא בממוצע תוצאות טובות יותר מאשר האלגוריתמים של דארווין או למארק.
ובנוסף, האלגוריתם של למארק עדיף על האלגוריתם של דארווין.