אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

<u>תרגיל בית מספר 3 - להגשה עד 27/11/2021 בשעה 23:55</u>

קראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

<u>הגשה</u>:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- התשובות בקובץ ה pdf חייבות להיות מוקלדות **ולא** בכתב יד.
- השתמשו בקובץ השלד skeleton3.py כבסיס לקובץ ה py אותו אתם מגישים.
  לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.

- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם hw3_012345678.pdf ו- hw3_012345678.py.
- הקפידו לענות על כל מה שנשאלתם.

- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
  להנחיה זו מטרה כפולה:
  1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
  2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

<u>הערות כלליות לתרגיל זה ולתרגילים הבאים:</u>

- כיוון שלמדנו בשבועות האחרונים כיצד לנתח את זמן הריצה של הקוד שלנו, החל מתרגיל זה ולאורך שארית הסמסטר (וכן במבחן) נדרוש שכל הפונקציות שאנו מממשים תהיינה יעילות ככל הניתן. לדוגמה, אם ניתן לממש פתרון לבעיה בסיבוכיות $O(logn)$, ואתם מימשתם פתרון בסיבוכיות $\Theta(n)$, תקבלו ניקוד חלקי על הפתרון.
- אין להשתמש בספריות חיצוניות אלא אם נאמר אחרת.

<u>שאלה 1</u>

א. הוכיחו או הפריכו את הטענות הבאות. ציינו תחילה בברור האם הטענה נכונה או לא, ואח״כ הוכיחו / הפריכו באופן פורמלי תוך שימוש בהגדרת $O(\cdot)$.

<u>הנחיה</u>: **יש להוכיח / להפריך כל סעיף בלא יותר מ-5 שורות.**

הפתרונות הם קצרים, ואינם דורשים מתמטיקה מתוחכמת. אם נקלעתם לתשובה מסורבלת וארוכה, כנראה שאתם לא בכיוון. לאורך השאלה n הוא משתנה ואינו קבוע, כל הפונקציות הן מהטבעיים לעצמם ($f, g: \mathbb{N} \to \mathbb{N}$) והאופרטור log הוא לפי בסיס 2.

1. $16^{logn} = O(n^3)$
2. $n^2 logn + n(logn)^2 = O(n(logn)^2)$
3. לכל $k \in \mathbb{N}$ קבוע ופונקציות $f_1, \dots, f_k$, $g_1, \dots, g_k$ כך שלכל $1 \le i \le k$ מתקיים $f_i = O(g_i)$, מתקיים כי

$$\sum_{i=1}^{k} f_i = O\left(\sum_{i=1}^{k} g_i\right)$$

<u>תזכורת</u>: סכום של k פונקציות היא פונקציה המוגדרת על ידי

$$\left(\sum_{i=1}^{k} f_i\right)(x) = \sum_{i=1}^{k} f_i\,(x)$$

4. לכל $k \in \mathbb{N}$ קבוע ופונקציות $f_1, \dots, f_k$, $g_1, \dots, g_k$ כך שלכל $1 \le i \le k$ מתקיים $f_i = O(g_i)$, מתקיים כי

$$\sum_{i=1}^{k} f_i = O\left(\max_{1\le i\le k}\{g_i\}\right)$$

<u>תזכורת</u>: מקסימום של k פונקציות היא פונקציה המוגדרת על ידי

$$\max_{1\le i\le k}\{g_i\}\,(x) = \max_{1\le i\le k}\{g_i(x)\}$$

5. אם $f_1(n) = O\big(g_1(n)\big)$ וגם $f_2(n) = O\big(g_2(n)\big)$ אז $f_1(n) \cdot f_2(n) = O\big(g_1(n) \cdot g_2(n)\big)$
6. אם $f_1(n) = O\big(g_1(n)\big)$ ו $f_2(n) = O\big(g_2(n)\big)$ אז $f_1 \circ f_2(n) = O\big(g_1 \circ g_2(n)\big)$

<u>תזכורת</u>: $f \circ h(n) = f\big(h(n)\big)$

7. קיימים קבועים $b > 0, t > 0$ כך ש- $n^n = O(b^{nt})$

ב. הוכיחו כי לכל $\varepsilon > 0$ קבוע ולכל $k \in \mathbb{N}$ קבוע חיובי הטענה הבאה אינה נכונה:

$$(2^k + \varepsilon)^{logn} = O(n^k)$$

ג. לכל אחת משתי הפונקציות הבאות, נתחו את סיבוכיות זמן ריצתה במקרה הגרוע כתלות ב-n (אורך הרשימה L).
הניחו כי פעולות אריתמטיות (כמו גם המתודות הנקראות מהספרייה math) ופעולות append רצות בזמן $O(1)$.
ציינו את התשובה הסופית, ונמקו. על הנימוק להיות קולע, קצר וברור, ולהכיל טיעונים מתמטיים או הסברים מילוליים, בהתאם לצורך.
על התשובה להינתן במונחי $O(\ldots)$, ועל החסם להיות הדוק ככל שניתן. למשל, אם הסיבוכיות של פונקציה היא $O(n)$ ובתשובתכם כתבתם $O(nlogn)$, התשובה לא תקבל ניקוד (על אף שפורמלית O הוא חסם עליון בלבד).

.1

```
def f1(L):
    n = len(L)
    while n > 0:
        n = n // 2
        for i in range(n):
            if i in L:
                L.append(i)
    return L
```

.2

```
def f2(L):
    n = len(L)
    res = []
    for i in range(500, n):
        m = math.floor(math.log2(i))
        for j in range(m):
            k=1
            while k<n:
                k*=2
                res.append(k)
    return res
```

<u>שאלה 2</u>

בשאלה זה נעסוק בייצוג של float במחשב.

בשונה מגרסת ה-64 ביטים שראיתם בהרצאה ובתרגול, בשאלה זו נשתמש בייצוג של 32 ביטים.

נייצג מספר ממשי על ידי המחרוזת $b_0 \dots b_{31}$, כאשר $b_i \in \{0,1\}$ לכל $0 \le i \le 31$. נפענח את המחרוזת באופן הבא:

- $sign = b_0$ - ביט יחיד הקובע את סימן המספר.
- $exp = b_1 \dots b_8$ - מספר בייצוג בינארי בן 8 ביטים (יתכנו אפסים מובילים), מתקיים:

$$0 \le exp \le 2^8 - 1 = 255$$

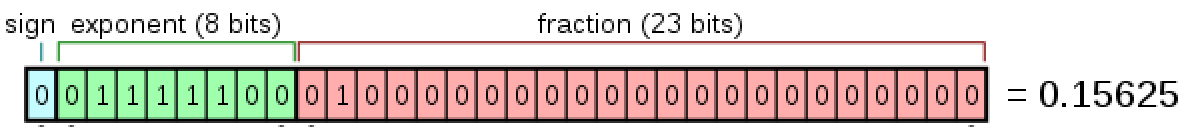
- $fraction = b_9 \dots b_{31}$ - מספר בינארי בן 23 ביטים המייצג את החלק השברי של המספר, ומחושב על ידי:

$$fraction = \sum_{i=1}^{23} b_{i+8} 2^{-i}$$

והנוסחה לחישוב המספר היא:

$$\boxed{num = (-1)^{sign} \cdot 2^{exp-127} \cdot (1 + fraction)}$$

דוגמה להמחשה:



בדוגמה זו $exp = (01111100)_2 = (124)_{10}$, $fraction = 1 \cdot 2^{-2} = \frac{1}{4}$ ו- $sign = 0$, ולכן המספר הממשי שמחרוזת זו מייצגת הוא $(-1)^0 \cdot 2^{124-127} \cdot \left(1 + \frac{1}{4}\right) = 0.15625$

א. הוכיחו כי כל מספר שניתן לייצג בשיטת ייצוג זו ניתן לייצג באופן מדויק גם בשיטת הייצוג של 64 ביטים שראיתם בכיתה.

ב. תנו דוגמה למספר שניתן לייצג ב-64 ביטים אך לא ניתן לייצג ב-32 ביטים בתחום $[1,2)$. כתבו את המספר כסכום של מספרים בבסיס עשרוני (לדוגמה $(1 + \frac{1}{4}$), ולא את הייצוג הבינארי המפורש.

ג. ממשו את הפונקציה bin_to_fraction(binary) אשר מקבלת מחרוזת בינארית באורך כלשהו, המייצגת חלק שברי במספר, ומחזירה את המספר כ-float. מותר לקרוא לפונקציה int.

דוגמה הרצה:

```
>>> bin_to_fraction('01101')
0.40625
>>> bin_to_fraction('1010000')
0.625
```

ד. ממשו את הפונקציה bin_to_float(binary) אשר מקבלת מחרוזת בינארית באורך 32 המתארת ייצוג של מספר ממשי כמתואר לעיל, ומחזירה את המספר כ-float. יש לממש את הפונקציה בעזרת כתיב למבדא (lambda), כפי שהיא מופיעה בקובץ השלד (החליפו את None במימוש שלכם). מותר לקרוא לפונקציה int ולפונקציה מסעיף ג'.
דוגמת הרצה:

```
>>> bin_to_float('00111110001000000000000000000000')
0.15625
```

ה. ממשו את הפונקציה is_greater_equal(bin1,bin2) המקבלת שתי מחרוזות בינאריות באורך 32 המייצגות מספר ממשי כמתואר לעיל, ומחזירה True אם המספר שמייצג bin1 גדול או שווה למספר שמייצג bin2.
בסעיף זה **<u>אסור</u>** להמיר את המחרוזות למספרים, בפרט אסור לקרוא לפונקציות מסעיפים ג' וד'.

ו. כל השאלות הבאות מתייחסות לשיטת הייצוג ב-<u>32 ביטים</u>. יש <u>לפרט בקצרה</u> את חישוביכם בקובץ ה-PDF.

i. בהינתן $k \in \mathbb{Z}$ מספר שלם כלשהו, כמה מספרים שונים ניתנים לייצוג בקטע $[2^k, 2^{k+1})$?

ii. כמה מספרים שונים ניתנים לייצוג בקטע $[3,10]$?

iii. כמה מספרים שונים ניתנים לייצוג בקטע $(0,1]$?

iv. מהי מידת הדיוק בקטע $[32,64)$? כלומר, מהו ההפרש המינימלי בין שני מספרים שונים הניתנים לייצוג בקטע?

שאלה 3

שאלה זו תעסוק בחישוב נומרי.

הערה: כזכור, חישובים אריתמטיים של אובייקטים מטיפוס float הם לא מדויקים מטבעם, אך בשאלה זו נתייחס לחישוב כאילו הוא מדויק. בפרט, ניתן להתעלם משגיאות הנובעות מחוסר דיוק של float.

א. בסעיף זה נממש פונקציה שמקרבת שורש של מספר ממשי חיובי.

ניעזר בעובדה הבאה: לכל $x \in \mathbb{R}^+$ מספר ממשי חיובי, ניתן לכתוב את x כסכום (אולי אינסופי) מהצורה

$$x = \frac{1}{a_1} + \frac{1}{a_1 \cdot a_2} + \frac{1}{a_1 \cdot a_2 \cdot a_3} + \cdots$$

כאשר לכל n טבעי, האיבר a_n הוא מספר טבעי המחושב על ידי החוקיות הבאה:

1. תנאי התחלה: a_1 הוא המספר הטבעי המינימלי כך ש - $x \geq \frac{1}{a_1}$
2. כלל נסיגה: לכל $n \geq 2$ מתקיים כי $a_n \geq a_{n-1}$ וכן a_n הוא המספר הטבעי המינימלי עבורו מתקיים

$$x \geq \frac{1}{a_1} + \frac{1}{a_1 a_2} + \cdots + \frac{1}{a_1 a_2 \ldots a_n}$$

שימו לב: במקרים מסוימים הסדרה שתיארנו מסתיימת לאחר מספר סופי של איברים, וזה תקין לחלוטין (הבחנה מעניינת: הסדרה תהיה סופית אמ"ם המספר x הוא רציונלי).

דוגמה לחישוב הסדרה עבור $x = \sqrt{2}$ (מומלץ להמשיך לפתח עד לפחות $n = 4$ כדי לוודא שההגדרה ברורה לכם):

- $a_1 = 1$ - כי $\sqrt{2} \geq \frac{1}{1}$
- $a_2 = 3$ - כי $\sqrt{2} < \frac{1}{1} + \frac{1}{1 \cdot 2}$ אבל $\sqrt{2} \geq \frac{1}{1} + \frac{1}{1 \cdot 3}$

שימו לב שככל שאנו מתקדמים בסדרה, הסכום הולך ומתקרב ל-$\sqrt{2}$.

ממשו את הפונקציה approx_root(x,e) המקבלת מספר ממשי חיובי x ומספר ממשי חיובי e, ומוצאת קירוב e עבור **<u>שורש x</u>**. בפרט, הפונקציה תחזיר אובייקט מסוג tuple אשר יכיל את זוג האובייקטים הבאים:

1. רשימה המכילה את ערכי הסדרה החל מ-a_1, ועד הראשון a_n שמקיים

$$\sqrt{x} - \left(\frac{1}{a_1} + \frac{1}{a_1 a_2} + \cdots + \frac{1}{a_1 a_2 \ldots a_n}\right) < e$$

2. הסכום המתאים לערכי a_i אלו, קרי הקירוב שהתקבל ל-$\sqrt{x}$.

הנחיה מחייבת: אסור להשתמש בפונקציה מובנית אשר מחשבת שורש של מספרים (בפרט, אין להעלות בחזקת 0.5 או להשתמש ב-math.sqrt).

שימו לב שסעיף זה איננו סעיף ברקורסיה, על אף שהשאלה נראית רקורסיבית באופייה.

דוגמת הרצה:

```
>>> approx_root(2, 0.1)
([1, 3], 1.3333333333333333)
>>> approx_root(2, 0.02)
([1, 3, 5], 1.4)
```

ב. בהרצאה ראיתם אלגוריתם לקירוב של π באמצעות שיטת מונטה-קרלו. בסעיף זה נקרב את [הקבוע e](#) בצורה דומה.
נתאר את המשחק הרנדומי הבא: בכל סיבוב של המשחק נגריל מספר אקראי בתחום $[0,1]$. נפסיק לשחק כאשר סכום כל המספרים שהגרלנו מתחילת המשחק עבר את 1. באופן יותר פורמלי:

1. בסיבוב ה-k של המשחק נגריל את המספר r_k באופן אחיד מתוך התחום $[0,1]$.
2. אם $\sum_{i=1}^{k} r_i > 1$ – נפסיק את המשחק, אחרת נעבר לסיבוב הבא.

נסמן ב-n את מספר הסיבובים ששיחקנו, כלומר את הסיבוב הראשון שבו $\sum_{i=1}^{n} r_i > 1$.
מסתבר שהערך <u>הממוצע</u> של n הוא המספר e (ניתן להוכיח זאת בכלים הסתברותיים).

i. ממשו את הפונקציה approx_e(N) אשר מסמלצת <u>את המשחק המתואר</u> N פעמים (שימו לב – כל משחק מכיל כמה סיבובים), ומחזירה קירוב מתאים למספר e. השתמשו בפונקציה random.random().
<u>הערה</u>: מומלץ לממש פונקציית עזר שמסמלצת משחק יחיד, ואז לקרוא לפונקציה זו N פעמים (בדומה לשאלת הרולטה בתרגיל 2).

ii. הריצו את הפונקציה על הערכים $N = 100, 1000, 10000$ **<u>וכתבו את התוצאות בטבלה בקובץ ה-PDF</u>**.

iii. מהו זמן הריצה של הפונקציה שמימשתם <u>במקרה הגרוע ביותר</u>?

שאלה 4

בשאלה זו הניחו כי פעולות אריתמטיות והשוואת מספרים מתבצעות בזמן קבוע.

רשימה L היא <u>כמעט ממוינת</u> אם כל איבר בה נמצא לכל היותר במרחק אינדקס אחד מהמיקום שלו ברשימה הממוינת.
כלומר, אם $\mathrm{arg\_}sort(i)$ הוא האינדקס של $L[i]$ ברשימה הממוינת sorted(L) אז

$$\mathrm{arg\_}sort(i) \in \{i-1, i, i+1\}$$

לדוגמה, הרשימה [2, 1, 3, 5, 4, 7, 6, 8, 9] היא כמעט ממוינת.

א. נרצה לממש פעולת חיפוש ברשימה <u>כמעט ממוינת</u>.
i. השלימו את הפונקציה find בשלד, שמקבלת את רשימה כמעט ממוינת L ומספר שלם s ומחזירה את האינדקס i כך ש $L[i] == s$ אם s הוא איבר ברשימה L, אחרת מחזירה $None$. למשל, עבור L מהדוגמה ו-$s = 5$, הפונקציה תחזיר 3 (כי המספר 5 נמצא באינדקס 3 ברשימה L). עבור $s = 11$ הפונקציה תחזיר None (כי המספר 11 לא נמצא ברשימה L).
ii. מה היא סיבוכיות זמן הריצה? הסבירו בקצרה.

ב. נרצה למיין רשימה <u>כמעט ממוינת</u> במקום (in-place), <u>ללא</u> שימוש ברשימת עזר.
i. השלימו את הפונקציה sort_from_almost(lst) בשלד, שמקבלת רשימה כמעט ממוינת וממיינת אותה ללא שימוש ברשימת עזר (או כל מבנה בעל גודל יותר מ O(1)).
ii. הסבירו בקצרה את הפתרון שלכם ואת סיבוכיות זמן הריצה שלו.

ג. סעיף זה לא עוסק ברשימה כמעט ממוינת.
מינימום מקומי ברשימה L הוא כל אינדקס i שקטן או שווה לשכניו המידיים. כלומר, כל אינדקס המקיים:

$$(i == 0 \; or \; L[i] \le L[i-1]) \; and \; (i == n-1 \; or \; L[i] \le L[i+1])$$

לדוגמא, ברשימה [5, 6, 7, 5, 1, 1, 99, 100] האינדקסים 0, 4, 5 הם מינימום מקומי.

i. האם בכל רשימה לא ריקה של מספרים יש מינימום מקומי? נמקו את תשובתכם.
ii. השלימו את הפונקציה find_local_min בשלד, שמקבלת רשימה לא ריקה של מספרים (לא ממוינים וייתכנו חזרות) ומחזירה אינדקס i של מינימום מקומי (אם יש יותר מאחד אז ניתן לבחור שרירותית). למשל, עבור הרשימה מהדוגמה תשובה של 0, 4 או 5 תהיה תקינה.
iii. מהי סיבוכיות זמן הריצה? הסבירו בקצרה.

<u>שאלה 5</u>

בכיתה ראינו את האלגוריתם מיון-בחירה (selection sort) למיון רשימה נתונה. האלגוריתם כזכור רץ בסיבוכיות זמן $O(n^2)$ עבור רשימה בגודל n. ראינו גם אלגוריתם מיון-מהיר יעיל יותר (quicksort), שרץ בסיבוכיות זמן ממוצעת $O(n \log n)$. לפעמים, כאשר יש לנו מידע נוסף על הקלט, אפשר למיין בסיבוכיות זמן טובה מזו. למשל, בשאלה זו, נעסוק במיון של רשימה שכל איבריה מוגבלים לתחום מצומצם יחסית: מחרוזות באורך k , עבור $k > 0$ נתון כלשהו, מעל האלפבית a, b, c, d, e שמכיל 5 תווים.

ההשוואה בין זוג מחרוזות תהיה לקסיקוגרפית, כלומר השוואה מילונית רגילה.

<u>הערות</u>:

1. בשאלה זו אסור להשתמש בפונקציות מיון מובנות של פייתון.
2. בניתוח הסיבוכיות בשאלה זו נניח שהשוואה של זוג מחרוזות באורך k מבצעת בפועל השוואה של התווים של המחרוזות משמאל לימין, ובמקרה הגרוע תהיה מסיבוכיות זמן $O(k)$.
3. לשם פשטות ניתוח הסיבוכיות נתייחס הן לפעולות אריתמטיות והן לפעולות העתקה של מספרים ממקום למקום בזכרון כפעולות שרצות בזמן קבוע.

א. השלימו בקובץ השלד את הפונקציה string_to_int(s) שמקבלת כקלט מחרוזת s באורך k <u>בדיוק</u> שמורכבת מהתווים a, b, c, d, e ומחזירה מספר שלם בין 0 ל $5^k - 1$ כולל, המייצג את הערך הלקסיקוגרפי היחסי של המחרוזת. על הפונקציה להיות חד-חד-ערכית. סיבוכיות הזמן שלה צריכה להיות $O(k)$.

ב. השלימו בקובץ השלד את הפונקציה int_to_string(k, n) , ההפוכה לזו מסעיף א', שמקבלת כקלט מספר שלם k גדול מ0 , וכן מספר שלם n בין 0 ל $5^k - 1$ כולל ומחזירה מחרוזת s באורך k <u>בדיוק</u> שמורכבת מהתווים a, b, c, d, e. גם על פונקציה זו להיות חד-חד-ערכית. סיבוכיות הזמן שלה צריכה להיות $O(k)$.
שימו לב שפונקציה זו צריכה לקיים לכל $0 \le i \le 5^k - 1$:

string_to_int(int_to_string(k, i)) == i

דוגמת הרצה:

```
>>> for i in range(5**3):
        if string_to_int(int_to_string(3, i)) != i:
            print("Problem with ", i)
>>> alphabet = ["a","b","c","d","e"]
>>> lst = [x+y+z for x in alphabet for y in alphabet for z in alphabet]
>>> for item in lst:
        if int_to_string(3, string_to_int(item)) != item:
            print("Problem with ", item)
>>> #Nothing was printed
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

בסעיפים הבאים נממש פונקציות מיון באמצעות ההמרה שהגדרנו זה עתה. נבחן שתי שיטות שונות לממש את המיון – אחת המשתמשת בזכרון עזר גדול ולה זמן ריצה קצר במיוחד, ואחת המשתמשת בזכרון עזר מינימלי ולה זמן ריצה ארוך במיוחד.

ג. השלימו בקובץ השלד את הפונקציה sort_strings1(lst, k) שמקבלת כקלט רשימה lst של n מחרוזות כמתואר ומספר חיובי k כך שכל מחרוזת ברשימה הינה באורך k בדיוק. על הפונקציה להחזיר רשימה <u>חדשה</u> ממויינת בסדר עולה (ולא לשנות את lst עצמה). בנוסף לרשימת הפלט שהיא בגודל n כמובן, על הפונקציה להשתמש ברשימת עזר בעלת 5^k איברים.
עליכם להשתמש בפונקציות מסעיפים א', ב'.
על הפונקציה sort_strings1 להיות מסיבוכיות זמן $O(kn + 5^k)$.

ד. בקובץ ה pdf הסבירו מדוע הפונקציה מסעיף ג' עומדת בדרישות הסיבוכיות.

ה. השלימו בקובץ השלד את הפונקציה sort_strings2(lst, k) שמקבלת קלטים כמו הפונקציה מסעיף ג', ובדומה לפונקציה הקודמת עליה להחזיר רשימה <u>חדשה</u> ממויינת בסדר עולה (ולא לשנות את lst עצמה).
הפעם מותר להשתמש בזכרון עזר מגודל $O(k)$, לא כולל רשימת הפלט שעליכם לייצר שגודלה הוא n. בפרט, בסעיף זה אסור להשתמש ברשימת עזר כמו בסעיף הקודם. על הפונקציה להיות מסיבוכיות זמן $O(5^k \cdot kn)$.

ו. בקובץ ה pdf הסבירו מדוע הפונקציה מסעיף ה' עומדת בדרישות סיבוכיות הזמן והזיכרון.

<u>שאלה 6</u>

שאלה זו היא השלב האחרון בתהליך משוב העמיתים שהתבצע לאורך הסמסטר. בשלב זה, תקבלו במהלך הימים הקרובים את ההערכות שנתנו לכם חבריכם וחברותיכם לקבוצה בתרגיל הבית הקודם. לאחר מכן, תתבקשו לממש שוב את הפונקציה `max_even_seq(n)` ולמלא סקר קצר על השתתפותכם בתהליך.

אנחנו מודים לכם מאוד על שיתוף הפעולה!

א. קראו את ההערכות שנתנו לכם חבריכם וחברותיכם לקבוצה בתרגיל הבית הקודם על ידי כניסה לרכיב משוב עמיתים במודל ולחיצה על ההגשה שלכם. לאחר מכן, ממשו שוב, **בצורה הטובה ביותר שביכולתכם**, את הפונקציה `max_even_seq(n)` שבקובץ השלד `max_even_seq.py` (<u>ולא</u> בקובץ השלד שבו נמצאות שאר שאלות הקוד) והגישו אותו בתיבת ההגשה המיועדת לכך[1] (<u>ולא</u> בתיבת ההגשה של תרגיל בית 3).

ב. מלאו את הסקר המסכם של תהליך משוב העמיתים[1].

[1] גם תיבת ההגשה וגם הסקר יהיו זמינים במודל לאחר קבלת ההערכות.