תרגיל בית מספר 3 - להגשה עד 3 במאי (יום ראשון) בשעה 23:55

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

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- 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 012345678.pdf ו- 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.

שאלה 1

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.

<u>הנחיה</u>: **יש להוכיח / להפריך כל סעיף בלא יותר מ- 2 שורות**.
הפתרונות הם קצרים, ואינם דורשים מתמטיקה מתוחכמת. אם נקלעתם לתשובה מסורבלת וארוכה, כנראה שאתם לא בכיוון.
אם יש צורך, מותר להשתמש בעובדות הידועות הבאות: $\log n = O(n)$, $n^a = O(n^b)$ עבור $a<=b$, וכן בטענות שהוכחו בכיתה.
אלא אם צוין אחרת, log הוא לפי בסיס 2.

1. לכל קבוע $k > 1$ מתקיים $n(log_2 n)^k = O(n(log_{10} n)^k)$
2. $2^{\log_2 n} = O(2^{\log_{10} n})$
3. $\sqrt{2}^{\text{nlogn}} = O(n^{\sqrt{n/2}})$
4. $\sum_{i=1}^{\log n} \frac{i}{2^n} = O(1)$

ב. לכל אחת מהפונקציות הבאות, נתחו את סיבוכיות זמן ריצתה כתלות ב- n (אורך הרשימה lst). הניחו כי כל פעולה בודדת (ובכלל זה פעולת כתיבה של איבר ברשימה לזיכרון) דורשת O(1) זמן. ציינו את התשובה הסופית, ונמקו. על הנימוק להיות קולע, קצר וברור, ולהכיל טיעונים מתמטיים או הסברים מילוליים, בהתאם לצורך.

על התשובה להינתן במונחי O(…), 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(lst):
    n = len(lst)
    for i in range(n):
        l.extend(range(i))
```

.2
```
def f2(lst):
    n = len(lst)
    for i in range(n):
        l.extend(range(len(lst)))
```

.3
```
def f3(lst):
    n = len(lst)
    for i in range(n):
        l = l + list(range(len(lst)))
```

.4
```
def f4(lst):
    n = len(lst)
    for i in range(n):
        l.extend(range(len(lst), len(lst)+10000))
```

ג. הסבירו בקצרה מדוע הרצת קטע הקוד הבא תיכנס ללולאה אינסופית ולא תסתיים. היעזרו במה שלמדנו על מודל הזיכרון של פייתון.

```
l = [1,2,3]
for e in l:
   l += ["a"]
```

שאלה 2

על פי משפט ערך הביניים, אם קיימת פונקציה F שבקטע [a,b] הינה רציפה ומקיימת F(a) < 0 < F(b) או F(a) > 0 > F(b), אזי מובטח לנו שקיים לפחות שורש אחד של הפונקציה בקטע.

א. ממשו בקובץ השלד את הפונקציה עם החתימה הבאה:

```
def find_root(f, a, b, EPS=0.001)
```

הפונקציה מקבלת פונקציה מתמטית f (למשל בכתיב למבדה), קצוות של קטע (ערכי a ו-b) ותחום שגיאה אפסילון – ומחזירה ערך x, אשר ערך הפונקציה בו קרוב עד כדי אפסילון לאפס.

אנו מניחים בשאלה שהפונקציה רציפה (אין צורך לבדוק זאת!). אם שאר תנאי משפט ערך הביניים לא מתקיימים (לגבי ערכי הפונקציה בקצות התחום), אזי יוחזר הערך None.

הנחיה מחייבת: אין לממש באמצעות רקורסיה.

להלן דוגמת הרצה:

```
>>>find_root(lambda x : x - 1  , -10, 10)
1.0009765625
>>>find_root(lambda x : x**2  , -10, 10)
>>> #returned None. Why?
```

ב. מצאו שורש של הפונקציה $f(x) = x^2 - \cos(x)$ בתחום [-10 , 0] (עבור ערך האפסילון הדיפולטי).

ג. נסו למצוא שורש של הפונקציה $f(x) = x^5 + 2^{60}x^3 - 3^{60}x^2 + 7^{40}x - a$ עבור a=28250560813025601818088431283813497879892214878257 7 בתחום $[0 , 3^{100}]$. האם התוכנית עוצרת? אם לא, הסבירו מדוע. בהסבר שלכם, התייחסו לערך הפונקציה בנקודה $x = 12345678901$.

שאלה 3

בשאלה זו נדון במיזוג m-ערוצי. כלומר בהינתן קלט שהוא רשימה של $m \geq 2$ רשימות **ממויינות בסדר עולה** עלינו להחזיר רשימה אחת ממויינת (בסדר עולה) שמכילה את איברי כל הרשימות הנתונות. לשם פשטות ניתוח הסיבוכיות בשאלה זו, נניח שאורך כל אחת מהרשימות שניתנות כקלט הוא לכל היותר n. אך n איננו ידוע לכם ולכן אין להשתמש בו בקוד. נבחן 3 אסטרטגיות לפתרון הבעייה.
הערה: אין לשנות את הרשימות המקוריות שניתנו כקלט.

א. להלן מימוש אפשרי לפתרון הבעיה.

```
def multi_merge_v1(lst_of_lsts):
   all = [e for lst in lst_of_lsts for e in lst]
   merged = []
   while all != []:
        minimum = min(all)
        merged  += [minimum]
        all.remove(minimum)
   return merged
```

דוגמת הרצה:

```
>>> multi_merge_v1([[1,2,2,3],[2,3,5],[5]])
[1, 2, 2, 2, 3, 3, 5, 5]
```

מה סיבוכיות הזמן במקרה הגרוע של הפונקציה multi_merge_v1 כתלות ב-m וב-n ? תנו תשובה במונחי O(…) הדוקה ככל שניתן. הניחו כי הוספת איבר בסוף רשימה באמצעות += לוקחת O(1) זמן, וכי הפעולה remove מרשימה לוקחת גם כן O(1) זמן (הערה: בהמשך הקורס נבחן את תקפותה של ההנחה האחרונה). בנוסף, הניחו כי הפונקציה min פשוט עוברת פעם אחת על כל איברי הקלט שלה ומחירה איבר מינימלי, ולכן לוקחת O(n) זמן עבור רשימה בגודל n.
את תשובתכם הגישו בקובץ ה pdf.

ב. ממשו (בקובץ השלד) את הפונקציה multi_merge_v2(lst_of_lsts) שתפעל באופן הבא:
בדומה לפונקציה מהסעיף הקודם, הפונקציה תוסיף לרשימת הפלט בכל שלב את האיבר המינימלי הבא. אך לשם כך היא תשמור ברשימה נפרדת את ה"מועמדים" למינימום הבא. רשימה זו תכיל בכל שלב לכל היותר m אינדקסים, שיציינו את המיקומים של האיברים המינימלים בכל אחת מהרשימות (מבין האיברים שטרם התווספו לרשימת הפלט). בכל שלב כאמור ייבחר המינימום מבין מועמדים אלו (באמצעות הפונקציה min), יתווסף לרשימת הפלט, ובמקומו ייכנס מועמד חדש (איזה?).

<u>הנחיה מחייבת</u>: סיבוכיות הזמן של הפונקציה במקרה הגרוע צריכה להיות $O(m^2 n)$.

ג. עליכם להשלים בקובץ השלד את הפונקציה multi_merge_v3(`lst_of_lsts`). יש להשלים שתי שורות בלבד. עליכם לקרוא לפונקציה merge שראיתם בכיתה שממזגת שתי רשימות ממויינות.

```
>>> merge([1,2,2,3],[2,3,5])
[1, 2, 2, 2, 3, 3, 5]
```

```
def multi_merge_v3(lst_of_lsts):
 m = len(lst_of_lsts)
 merged = []

 for ________________________________________:

     ________________________________________

 return merged
```

ד. מהי סיבוכיות הזמן של הפונקציה multi_merge_v3 במקרה הגרוע כתלות ב- n, m? תנו תשובה במונחי O(…) הדוקה ככל שניתן.

שאלה 4

אמיר ומיכל קנו כל אחד לפטופ חדש. הלפטופ של מיכל מהיר פי 2 מזה של אמיר, כלומר הוא מסוגל לבצע פי 2 פעולות ליחידת זמן. שניהם מריצים על המחשבים שלהם את אותו אלגוריתם למשך דקה אחת.
בכל אחד מהסעיפים הבאים מצויין חסם הדוק לסיבוכיות הזמן של האלגוריתם.
עבור כל אחד מהסעיפים למטה, נניח שהמחשב של אמיר מסוגל לסיים בדקה את ריצתו על קלט מקסימלי בגודל n=100. בהינתן שחסם הדוק על סיבוכיות האלגוריתם הוא $T(n)$, רשמו מהו גודל הקלט המקסימלי עליו יסיים המחשב של מיכל לרוץ בדקה. ציינו את התשובה המספרית, ונמקו בשורה אחת.

א. $T(n) = O(\log n)$

ב. $T(n) = O(n)$

ג. $T(n) = O(n^2)$

ד. $T(n) = O(2^n)$

שאלה 5

א. ממשו (בקובץ השלד) את הפונקציה steady_state(L) שמקבלת כקלט רשימה L של מספרים שלמים שונים זה מזה, ממוינת בסדר עולה, ומחזירה כפלט אינדקס $i \geq 0$ המקיים $L[i] == i$. אם יש יותר מאינדקס אחד כזה, יוחזר אחד מהם – לבחירתכם (שרירותי). אם לא קיים כזה, יוחזר הערך None. דוגמאות הרצה:

```
>>> steady_state([-4,-1,0,3,5])
3
>>> steady_state([-4,-1,2,3,5])
2                              #could also return 3 instead
>>> steady_state([-4,-1,0,4,5])
>>>
```

<u>הנחיה</u>: על הפונקציה לרוץ בסיבוכיות זמן O(logn) כתלות באורך הרשימה n. פתרונות בסיבוכיות גבוהה יותר (למשל O(n) יאבדו ניקוד משמעותי.

ב. מדוע דרשנו שהמספרים יהיו שלמים? תנו דוגמה לרשימה באורך 5 שבה המספרים לא בהכרח שלמים (אבל מקיימת את יתר ההנחות), עבורה הפתרון שכתבתם לא יעבוד.

ג. מדוע דרשנו שהמספרים יהיו שונים? תנו דוגמה לרשימה באורך 5 שבה המספרים לא בהכרח שונים (אבל מקיימת את יתר ההנחות), עבורה הפתרון שכתבתם לא יעבוד.

ד. ממשו (בקובץ השלד) את הפונקציה cnt_steady_states(L) שמקבלת רשימה L כמו בסעיף הקודם, ומחזירה **<u>כמה</u>** אינדקסים i בה מקיימים $L[i] == i$.

דוגמאות הרצה:

```
>>> cnt_steady_states([-4,-1,0,3,5])
1
>>> cnt_steady_states([-4,-1,2,3,5])
2
>>> cnt_steady_states([-4,-1,0,4,5])
0
```

כרגיל, מותר ומומלץ לכתוב פונקציות עזר בהתאם לצורך.

<u>הנחיה</u>: על הפונקציה לרוץ בסיבוכיות זמן O(logn) כתלות באורך הרשימה n. פתרונות בסיבוכיות גבוהה יותר (למשל O(n) יאבדו ניקוד משמעותי.

שאלה 6

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.

כל איבר ברשימה אותה נרצה למיין הוא מספר x המקיים את אחד משני התנאים הבאים:

1. x הינו מספר שלם.
2. x הינו מספר מהצורה $x = y + 0.5$, עבור מספר y שלם כלשהו.

נסמן ב- k את המספר האי-שלילי השלם הקטן ביותר עבורו מתקיים: $(-k) \le x \le k$ לכל איבר x ברשימה הנתונה.

א. השלימו בקובץ השלד את הפונקציה sort_num_list(lst) שמקבלת כקלט רשימה lst כמתואר. על הפונקציה להחזיר רשימה חדשה ממויינת בסדר עולה (ולא לשנות את lst). על הפונקציה לרוץ בזמן $O(n+k)$ כאשר n הוא אורך הרשימה lst ו-k הוא חסם על ערכי הרשימה כמוגדר למעלה. אין צורך לבדוק את תקינות רשימת הקלט.

דוגמת הרצה:

```
>>> lst = [10, -2.5, 0, 12.5, -30, 0]
>>> sort_num_list(lst)
[-30, -2.5, 0, 0, 10, 12.5]        #The following result is valid as well:  [-30.0, -2.5, 0.0, 0.0, 10.0, 12.5]
```

ב. מה צריך לקיים k (כתלות ב- n) על מנת שסיבוכיות האלגוריתם שמימשתם תהיה עדיפה על הסיבוכיות הממוצעת של quicksort?

סוף