<u>תרגיל בית מספר 3 - להגשה עד 10 בדצמבר (יום רביעי) בשעה 23:55</u>

קיראו 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.
- 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. אם $f(n) = O\big(g(n)\big)$ אז $2^{f(n)} = O(2^{g(n)})$.
2. $\log(n!) = O(nlogn)$
3. לכל קבוע $k > 1$ מתקיים $\sum_{i=1}^{n} k^i = O(k^n)$.

ב. לכל אחת מהפונקציות הבאות, נתחו את סיבוכיות זמן ריצתה כתלות ב- n (אורך הרשימה lst). הניחו כי כל פעולה בודדת (ובכלל זה פעולות אריתמטיות) 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 $O(\dots)$, 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):
    i=5
    while i<len(lst):
        print(lst[i])
        i **= 2
```

2.
```
def f2(lst):
    i = len(lst)
    for j in range(i, i+10000):
        while i>0:
            for k in range(i):
                print(k)
            i -= 2
```

שאלה 2

בשאלה זו נעסוק באלגוריתם ניוטון-רפסון.

להלן תזכורת לקוד של הפונקציות NR ו- diff_param שנלמדו:

```
1. from random import *
2.
3. def diff_param(f,h=0.001):
4.     return (lambda x: (f(x+h)-f(x))/h)
5.
6. def NR(func, deriv, epsilon=10**(-8), n=100, x0=None):
7.    """ Given a real valued func and its real value derivative,
8.    deriv, NR attempts to find a zero of function, using the
9.    Newton-Raphson method.
10.   NR starts with a an initial x0 (default value is None
11.   which is replaced upon execution by a random number distributed
12.   uniformly in (-100.,100.)), and performs n=100 iterations.
13.   If the absolute value function on some x_i is smaller
14.   than epsilon, None is the returned value """
15.
16.   if x0 is None:
17.      x0 = uniform(-100.,100.)
18.   x = x0; y = func(x)
19.   for i in range(n):
20.      if abs(y) < epsilon:
21.         print(x, y, "convergence in", i, "iterations")
22.         return x
23.      elif abs(deriv(x)) < epsilon:
24.         print("zero derivative, x0=", x0," i=", i, " xi=", x)
25.         return None
26.      else:
27.         print(x,y)
28.         x = x - func(x)/deriv(x)
29.         y = func(x)
30.   print("no convergence, x0=", x0," i=", i, " xi=", x)
31.   return None
```

א. עבור f1 כלשהי, הרצת הפקודה NR(f1, diff_param(f1), x0=0) גרמה להדפסה המופיעה בשורה 24 והוחזר הערך None (שורה 25). עבור f2 כלשהי, הרצת הפקודה NR(f2, diff_param(f2), x0=0) גרמה למספר הדפסות המופיעות בשורה 27, ולבסוף הודפסה שורה 21 והוחזר ערך מספרי (שורה 22), שנסמנו root.
ציינו איזו מהאפשרויות הבאות תיתכן, והסבירו מדוע מתקבלות התוצאות הנ"ל עבור התשובה שסימנתם.
ההסבר צריך להיות קצר, אך לכלול פירוט של החישוב שגרם להחזרת None עבור f1.

1. f1 = lambda x : x**100000000 - 1 וגם f2 = lambda x : x**2 - 1 וגם root קרוב ל-1.
2. f1 = lambda x : x**100000000 - 1 וגם f2 = lambda x : x**2 - 1 וגם root קרוב ל-0.
3. f1 = lambda x : x**2 - 1 וגם f2 = lambda x : x**100000000 - 1 וגם root קרוב ל-1.
4. f1 = lambda x : x**2 - 1 וגם f2 = lambda x : x**100000000 - 1 וגם root קרוב ל-0.

ב. כעת נרצה לחשב את הפונקציה ההופכית של פונקציה נתונה, תוך שימוש באלגוריתם ניוטון רפסון. הפונקציה ההופכית של פונקציה f (מסומנת f^{-1}) מקיימת לכל x ו- y: $f^{-1}(y) = x$ אם ורק אם $f(x) = y$.

1. השלימו בקובץ השלד את הפונקציה source, שמקבלת כקלט פונקציה f וערך y, ומחזירה x עבורו מתקיים בקירוב טוב $f(x) = y$. תוכלו להניח כי קיים x יחיד כזה. על source לקרוא ל- NR (פתרונות אחרים לא יתקבלו). "קירוב טוב" ייקבע ע"י הערך epsilon של NR, ואין לשנות ערך זה, או ערכי ברירת מחדל אחרים. אין צורך לטפל במקרים בהם NR מחזירה None. שימו לב כי בגלל הניחוש ההתחלתי האקראי של NR, התוצאה יכולה להיות שונה מריצה לריצה (כפי שניתן לראות להלן):

```
>>> lin = lambda x: x+3
>>> source(lin,5)
2.0000000003798846
>>> source(lin,5)
1.9999999995163051
```

2. השלימו בקובץ השלד את הפונקציה inverse (יש להשלים שורה אחת בלבד), שמקבלת כקלט פונקציה f ומחזירה את ההופכית שלה f^{-1}. הניחו כי ל- f אכן קיימת פונקציה הופכית. יש להשתמש בפונקציה מהסעיף הקודם. דוגמת הרצה:

```
>>> inverse(lin)(5)
1.9999999998674198
```

ג. השלימו בקובץ השלד את הפונקציה repeated שמקבלת פונקציה מספרית f ומספר שלם $n > 0$, ומחזירה את ההפעלה הרצופה ה–n ית של f. כלומר את הפונקציה שמתקבלת מהרכבה של f על עצמה n-1 פעמים. למשל, עבור $n = 2$ repeated תחזיר את הפונקציה $f \circ f$.
לדוגמא:

```
>>> def  square(x):
       return x*x
>>> repeated(square, 2)(5)
625
```

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

נתון מספר שלם עשרוני N. מיכל ואמיר שיחקו עם המספר להנאתם, כמתואר בסעיפים הבאים. בכל אחד מהסעיפים, רישמו ביטוי למספר העשרוני המתקבל. תנו תשובה כתלות ב- N ו/או k, והסבירו את החישוב. יש לפשט את הביטוי ככל שניתן. ביטוי המכיל טור לא יתקבל.

א. מיכל רשמה את המספר בכתיב בינארי, ואמיר הוסיף למספר k פעמים 1 מימין.

ב. מיכל רשמה את המספר בכתיב בינארי, ואמיר הוסיף למספר 1 מצד שמאל.

שאלה 5

בוועדת ההוראה של ביה"ס למדעי המחשב באוני' חשובה כלשהי במרכז ישראל התקבלה ההחלטה לאסור את השימוש בחיפוש בינארי בקורס המבוא. ציטוט מפרוטוקול הוועדה:

"... אין באלגוריתם שום ערך והוא סתם מסובך".

מאוכזבים ומודאגים מהשלכות ההחלטה על עתידם של בוגרי מדעי המחשב, החליט צוות הקורס לתת בתרגיל בית שאלה שתציג אלטרנטיבה - חיפוש טרינארי (בתקווה שלא ישימו לב שזה כמעט אותו דבר): בכל שלב, האלגוריתם יחלק את הרשימה שלו ל- 3 חלקים במקום 2: בערך שליש האיברים השמאליים, בערך שליש האמצעיים, ובערך שליש הימניים. האלגוריתם יחליט עם איזה שליש יש להמשיך באמצעות בדיקת שני הגבולות בין החלקים. "בערך" – מותר שהחלקים יהיו בגדלים ששונים לכל היותר ב- 1 זה מזה.

א. ממשו בקובץ השלד פונקציה בשם ternary_search(key, lst) המקבלת מספר key ורשימה ממוינת lst של מספרים, ופועלת בשיטה הנ"ל להחזרת אינדקס בו נמצא key ב- lst, או None אם הוא לא נמצא. אם key נמצא ביותר מאינדקס אחד, יוחזר אחד האינדקסים, שרירותית. אין צורך לבדוק תקינות הקלט. הפונקציה לא מדפיסה דבר. לדוגמה:

```
>>> ternary_search(3,[1,2,3,4,5])
2
>>> ternary_search(1,[2,3,4,5])
>>>
```

<u>הנחיה</u>: על המימוש שלכם להיות איטרטיבי (כלומר באמצעות לולאות) ולא רקורסיבי (רקורסיה, למי שמכיר את המושג, תילמד בהמשך הקורס).

<u>הערה</u>: הקוד בשאלה זו ייבדק גם באופן ידני כחלק מתהליך הבדיקה של תרגיל זה. הקפידו על סגנון תכנות נאות, והימנעו מסיבוך של הקוד שאיננו הכרחי.

ב. מהי סיבוכיות הזמן של אלגוריתם החיפוש הטרינארי המתואר לעיל כתלות באורך הרשימה n?

הפרידו לשני מקרים: המקרה הטוב ביותר (רשימה באורך n ומפתח שעבורם זמן הריצה יהיה מינימלי) והמקרה הגרוע ביותר (כנ"ל, מקסימלי).

תנו תשובתכם במונחי O(...), הקפידו שתהיה הדוקה ככל שתוכלו, ונמקו כל אחד מהמקרים **בלא יותר משורה אחת**.

ג. (בונוס 5 נק') ניתן להכליל את המעבר מחיפוש בינארי לטרינארי, לחיפוש k-ארי: בכל שלב, האלגוריתם יחלק את הרשימה שלו ל- k חלקים "בערך" שווים. האלגוריתם יחליט עם איזה חלק יש להמשיך באמצעות בדיקת k-1 הגבולות בין החלקים. נשים לב שחיפוש כזה מבצע k-1 השוואות בכל איטרציה.

תנו ביטוי למספר ההשוואות שעושה אלגוריתם החיפוש ה- k-ארי עבור רשימה בגודל n, כתלות ב- n וב- k, וציינו מהו ה- k עבורו מספר זה מינימלי.

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. בהמשך הקורס נכיר אלגוריתם מיון יעיל יותר, שירוץ בסיבוכיות זמן $O(n\log n)$. 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: 26 האותיות הקטנות (lowercase) באנגלית (a-z).
במקרה כזה אפשר לבצע מיון בסיבוכיות זמן לינארית באורך הרשימה.
עליכם להשלים בקובץ השלד את הפונקציה sort_char_list שמקבלת כקלט רשימה lst שבה כל איבר הינו אות כלשהי מתוך a-z בלבד. על הפונקציה להחזיר רשימה חדשה ממויינת לקסיקוגרפית (ולא לשנות את lst) . כאמור, על הפונקציה לרוץ בזמן $O(n)$ במקרה הגרוע, כאשר n הוא אורך הרשימה lst.
בתחילת הפונקציה מופיעה המחרוזת alphabet ועליכם להשתמש בה. אין להשתמש בפונקציות ord(), chr() של פייתון שילמדו בהמשך הקורס.
דוגמת הרצה:

```
>>> sort_char_list(list("ghbtffsraaia"))
['a', 'a', 'a', 'b', 'f', 'f', 'g', 'h', 'i', 'r', 's', 't']
```

סוף