אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2016 2017

תרגיל בית מספר 3 - להגשה עד 11 בדצמבר 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 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.

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. $\log_2(n^5) = O(\log_{10}(\frac{n}{2}))$
2. $log\ ((\sum_{k=1}^{n}(k^2))^3) = O(log\ \sqrt{n})$
3. $10n^2 + 7n\ =\ O(n^2 - \sqrt{n})$
4. לכל פונקציה $f(n)$, אם $f(n) = O(\log n)$, אז $2^{f(n)} = O(n)$

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):
       lst.extend(range(i+10))
```

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

.3
```
def f3(lst):
   n = len(lst)
   for i in range(n):
       lst.extend(range(len(lst)-500, len(lst)+500))
```

.4
```
def f4(lst):
   n = len(lst)
   for i in range(n):
       lst = lst + list(range(len(lst)))
```

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

תהי $f: \mathbb{N}^+ \to \mathbb{Z}$ פונקציה שמקבלת כקלט מספר טבעי (גדול מ-0) ומחזירה כפלט מספר שלם. נאמר ש- f גדלה מונוטונית אם לכל ערך x מתקיים ש $f(x+1) > f(x)$. בהינתן פונקציה f שגדלה מונוטונית, נרצה למצוא את הערך n המינימלי עבורו f מחזירה ערך גדול ממש מ-0 . (מאחר ש- f גדלה מונוטונית אז מתקיים לכל $x \geq n$ ש- $f(x) > 0$ וכמו כן לכל $x < n$ מתקיים ש $f(x) \leq 0$.)

בשאלה זו הניחו כי סיבוכיות הזמן עבור הפעלת f על קלט כלשהו הינה $O(1)$.

א.

I. השלימו את מימוש הפונקציה `find_first_positive1` שמקבלת כקלט פונקציה $f: \mathbb{N}^+ \to \mathbb{Z}$ הגדלה מונוטונית ומחזירה את הערך n המינימלי עבורו f מחזירה ערך גדול ממש מ- 0.
שימו לב כי בסעיף זה המימוש הוא נאיבי, ללא ניסיון לייעל את זמן הריצה.

```
def find_first_positive1(f):
    n = 1
    while ____________________________________________:
        ____________________________________________
    return n
```

II. ציינו בקובץ ה- pdf מהי סיבוכיות זמן הריצה של `find_first_positive1` כפונקציה של הפלט n, כאשר n הינו הערך המינימלי שמקיים כי $f(n) > 0$? תנו תשובה במונחי סדר גודל $O(\ldots)$, הדוקה ככל שתוכלו ונמקו בקצרה.

ב. השלימו בקובץ השלד את הפונקציה `find_first_positive_range` שמקבלת פונקציה גדלה מונוטונית $f: \mathbb{N}^+ \to \mathbb{Z}$ ושני מספרים חיוביים, $a \leq b$ ומחזירה את המספר השלם הראשון n בקטע $[a, b]$ עבורו $f(n) > 0$. אם לא קיים כזה, על הפונקציה להחזיר None. כמו כן, הפונקציה צריכה לרוץ בסיבוכיות $O(\log(b-a+1))$.

דוגמאות הרצה:

```
>>> f = lambda x : x - 10
>>> find_first_positive_range(f, 1, 12)
11
>>> find_first_positive_range(f, 12, 20)
12
>>> find_first_positive_range(f, 5, 10) #returned None
>>>
```

ג.

I. השלימו בקובץ השלד את הפונקציה find_first_positive2 שמקבלת כקלט פונקציה $f: \mathbb{N}^+ \to \mathbb{Z}$ שגדלה מונוטונית ומחזירה את הערך n המינימלי עבורו f מחזירה ערך גדול ממש מ-0. על הפונקציה find_first_positive2 לרוץ בסיבוכיות זמן $O(\log n)$, כאשר n הינו הערך המינימלי שמקיים כי $f(n) > 0$.

על הפונקציה להשתמש בפונקציה find_first_positive_range שכתבתם בסעיף ב'.

דוגמת הרצה:

```
>>> f = lambda x : x - 10
>>> find_first_positive2 (f)
11
```

```
def find_first_positive2(f):
    ____________________________________________
    while____________________________________________:
        ____________________________________________
    return ____________________________________________
```

II. ציינו בקובץ ה- pdf מדוע הפונקציה שכתבתם עונה על דרישות הסיבוכיות.

שאלה 3

רשימה של n מספרים שונים זה מזה תיקרא רשימה בי-טונית (bitonic list) אם קיים אינדקס $0 \le k \le n-1$ כך שכל האיברים עד לאינדקס k (כולל) מופיעים בסדר עולה ממש, וכל האיברים החל מאינדקס k מופיעים בסדר יורד ממש.
לדוגמה: הרשימות `[1, 2, 3, 4]` , `[-3, 1, 2, 3, 80, 100, 6]` הן רשימות בי-טוניות.

א. השלימו בקובץ השלד את מימוש הפונקציה find_maximum שמקבלת כקלט רשימה בי-טונית של n מספרים שונים זה מזה, ומחזירה את האינדקס $0 \le k \le n-1$ שבו מתקבל הערך המקסימלי. על סיבוכיות הזמן של הפונקציה במקרה הגרוע להיות $O(\log n)$.
דוגמת הרצה:

```
>>> find_maximum([-3, 1, 2, 3, 80, 100, 6])
5
```

ב. השלימו בקובץ השלד את מימוש הפונקציה sort_bitonic_list(blst), שמקבלת כקלט רשימה בי-טונית blst, ומחזירה רשימה חדשה שאיבריה הם איברי blst ממויינים מקטן לגדול. השתמשו בפונקציה מסעיף א' ובפונקציה merge מהכיתה. תזכורת: merge מקבלת שתי רשימות ממוינות ויוצרת ביעילות רשימה חדשה ממוינת, המכילה את איברי שתיהן. לדוגמה:

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

דוגמת הרצה:

```
>>> sort_bitonic_list([-3, 1, 2, 3, 80, 100, 6])
[-3, 1, 2, 3, 6, 80, 100]
```

```
def sort_bitonic_list(blst):
    ____________________________________________
    return merge(_____________________ , ___________________ )
```

ג. מיכל מציעה להשתמש בפונקציה selection_sort (מיון מיזוג) שראינו בכיתה, לצורך מיון רשימה בי-טונית. לדוגמה:

```
>>> selection_sort([-3, 1, 2, 3, 80, 100, 6])
[-3, 1, 2, 3, 6, 80, 100]
```

לדעתה של מיכל, אמנם סיבוכיות זמן הריצה במקרה הגרוע על רשימה כלשהי (במונחים אסימפטוטיים של $O(\ldots)$) תהיה $O(n^2)$ אבל עבור רשימה בי-טונית זמן הריצה יהיה טוב יותר מאשר המקרה הגרוע ואף טוב יותר מסיבוכיות הזמן של הפונקציה sort_bitonic_list מסעיף ב'. אמיר, לעומתה, חושב להיפך, כלומר שסיבוכיות selection_sort גם במקרה זה גדולה יותר. דניאל טוען שהוויכוח מיותר, כי לשתי הפונקציות אותו חסם סיבוכיות זמן ריצה עבור רשימות בי-טוניות. ציינו בקובץ ה pdf מי לדעתכם צודק, וציינו מהי הסיבוכיות של שתי הפונקציות הנ"ל על רשימה בי-טונית כתלות באורך הרשימה n. תנו תשובה כחסם אסימפטוטי הדוק ככל שתוכלו.

שאלה 4

א. 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=200. 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.

a. $T(n) = O(\log n)$

b. $T(n) = O(n)$

c. $T(n) = O(n^2)$

d. $T(n) = O(2^n)$

ב. הגדרה: סיבוכיות זיכרון, או מקום של אלגוריתם, היא הכמות המקסימלית של תאי זיכרון שהאלגוריתם מקצה בכל שלב במהלך פעולתו, מעבר למקום שתופס הקלט שלו.
בכיתה ראינו שלושה אלגוריתמים בסיסיים במדעי המחשב: חיפוש בינארי, מיון בחירה, ומיזוג. לכל אחד מהאלגוריתמים, ציינו מהי סיבוכיות הזיכרון (space complexity) שלו, כתלות בגודל הקלט שלו, במונחים אסימפטוטיים (כלומר באמצעות הסימון O(…)).

שאלה 5

בכיתה ראינו את האלגוריתם מיון-בחירה (selection sort) למיון רשימה נתונה. האלגוריתם כזכור רץ בסיבוכיות זמן $O(n^2)$ עבור רשימה בגודל n. בקרוב נראה 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: זוגות (a,b) כאשר a ו- b מוגבלים להיות מספרים שלמים בין 0 ל-99. נייצג זוג כזה בפייתון ע"י אובייקט מסוג tuple.
נגדיר ש- (a1,b1) < (a2,b2) אם"ם a1<a2 או שמתקיים a1=a2 וגם b1<b2.

השלימו בקובץ השלד את הפונקציה sort_pairs(lst) שמקבלת כקלט רשימה lst של זוגות מסוג tuple כמתואר. על הפונקציה להחזיר רשימה חדשה ממויינת בסדר עולה (ולא לשנות את lst עצמה). על הפונקציה לרוץ בזמן $O(n)$ במקרה הגרוע, כאשר n הוא אורך הרשימה lst. אין צורך לבדוק את תקינות רשימת הקלט. רמז: שימו לב שטווח המספרים של כל איבר בזוג (0 עד 99) נחשב קבוע מבחינת ניתוח הסיבוכיות.

דוגמת הרצה:

```
>>> import random
>>> lst = [(random.choice(range(100)), random.choice(range(100))) for i in
range(12)]
```

```
>>> lst
[(9, 7), (78, 24), (17, 74), (53, 81), (40, 43), (79, 82), (84, 46), (68, 53),
(92, 95), (60, 38), (20, 62), (72, 57)]
>>> srt_lst = sort_pairs(lst)
>>> srt_lst
[(9, 7), (17, 74), (20, 62), (40, 43), (53, 81), (60, 38), (68, 53), (72, 57),
(78, 24), (79, 82), (84, 46), (92, 95)]
>>> srt_lst == sorted(lst)
True
```

שאלה 6

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. בהינתן שתי פונקציות f1(x) ו- f2(x), שתיהן גזירות (כלומר עומדות בתנאי הנדרש עבור שיטת ניוטון רפסון), נרצה למצוא ערך x בו הפונקציות נפגשות, כלומר f1(x) == f2(x). למשל עבור:

```
f1 = lambda x:4*x
f2 = lambda x:x**2+3
```

הפונקציות נפגשות בנקודות x=1 ו- x=3.

השלימו בקובץ השלד את הפונקציה equal(f1, f2) המחזירה ערך x כזה, אם קיים. יש להשלים שתי שורות בלבד. הפונקציה תקרא ל- NR.

כל המגבלות החלות על פעולתה של NR כמובן חלות גם על equal. למשל, יכול להיות שקיימת נקודת מפגש, אבל equal לא תמצא אותה כי NR לא החזירה תשובה כנדרש. יש להשלים שתי שורות בלבד.

ב. מה יקרה כאשר יועברו ל- equal שתי פונקציות שאין להן כלל נקודת מפגש? להלן שני מקרים כאלו. עליכם להחליט עבור כל מקרה, האם:

1) equal תחזיר None ותודפס השורה מתוך NR שמתחילה ב- "zero derivative"
2) equal תחזיר None ותודפס השורה מתוך NR שמתחילה ב- "no convergence"
3) אף על פי שאין לפונקציות נקודת מפגש equal תחזיר ערך מספרי כלשהו (שכמובן אינו נכון).

```
>>> equal(lambda x:x, lambda x:x**2+1)

>>> equal(lambda x:x, lambda x:x+1)
```

את תשובתכם כיתבו בקובץ ה pdf והסבירו בקצרה.

ג. 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} (עד כדי הדיוק שמצויין בפונקציה NR). הניחו כי ל- f אכן קיימת פונקציה הופכית. יש להשתמש בפונקציה מהסעיף הקודם. דוגמת הרצה:

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

סוף