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

תרגיל בית מספר 2 - להגשה עד 27 בנובמבר 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 skeleton2.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 hw2_012345678.pdf ו- hw2_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 2016-2017

שאלה 1

אלגוריתם שמבצע את פעולתו **בַּמָּקוֹם** (in-place algorithm) הוא אלגוריתם אשר בכל רגע נתון במהלך פעולתו, מעתיק לכל היותר כמות קבועה מהקלט שלו (כלומר כמות שאינה תלויה בגודל הקלט) למשתני עזר. כלומר בכל רגע נתון במהלך ריצת האלגוריתם, כמות זכרון העזר בה הוא משתמש (מעבר לזיכרון שצורכים משתני הקלט שלו) הינה קבועה. בפרט, אלגוריתם כזה לא יכול ליצור העתק של הקלט שלו (מדוע?), אלא משנה אותו במידת הצורך.

ראשית, נבחן בהקשר זה את פעולת היפוך סדר באיברים של רשימה נתונה בשם lst.

א. האם הפקודה lst = lst[: :-1] הופכת את סדר האיברים **במקום**? הסבירו בקצרה את תשובתכם.

ב. הפקודה lst.reverse() (המתודה reverse של המחלקה list) הופכת את סדר האיברים של הרשימה lst **במקום**. מה מחזירה הפונקציה? מדוע לדעתכם אין פונקצית reverse **במקום** עבור מחרוזת? הסבירו.

ג. הוסיפו לקובץ השלד המצורף מימוש לפונקציה reverse_sublist(lst,start,end), אשר מקבלת רשימה ושני אינדקסים, והופכת **במקום** את סדר האיברים בחלק הרשימה שמתחיל באינדקס start ונגמר באינדקס end-1.

לשם הפשטות ניתן להניח כי 0<=start<end<=len(lst).

דוגמאות הרצה:

```
>>> lst = [1, 2, 3, 4, 5]
>>> reverse_sublist (lst,0,4)
>>> lst
[4, 3, 2, 1, 5]
>>> lst = ["a","b"]
>>> reverse_sublist (lst,0,1)
>>> lst
["a", "b"]
```

כעת נדון בבעיית החלוקה: בהינתן רשימת מספרים שונים זה מזה נרצה לסדר בה את האיברים מחדש באופן הבא: אם נקרא לאיבר השמאלי ביותר "ציר" (pivot) אז כל האיברים שקטנים מהציר יופיעו משמאל לכל האיברים שגדולים ממנו. הציר יופיע בין שני החלקים.

<u>שימו לב</u>: אין חשיבות לסדר הפנימי בתוך כל אחד משני החלקים.

ד. השלימו בקובץ השלד המצורף את מימוש הפונקציה divide_list(lst) שמקבלת רשימת מספרים שונים זה מזה, ומחזירה <u>רשימה חדשה</u> מחולקת כמתואר לעיל (כאשר האיבר השמאלי ביותר הינו איבר ה"ציר"). אין להשתמש בפונקציות מוכנות של פייתון (גם לא פונקציות מיון)! הפונקציה אינה משנה את הרשימה המקורית lst. עליכם להשלים 2 שורות.

```
def divide_list(lst):
    pivot = lst[0]
    smaller = ____________________________________
    greater = ____________________________________
    return smaller + [pivot] + greater
```

פלט אפשרי לדוגמא (יתכנו פלטים חוקיים נוספים):

```
>>> lst
[4, 1, 7, 8, 3, 5, 6, 2]
>>> divide_list(lst)
[1, 3, 2, 4, 7, 8, 5, 6]  # יתכנו פלטים חוקיים נוספים
>>> lst
[4, 1, 7, 8, 3, 5, 6, 2]
```

ה. הוסיפו לקובץ השלד המצורף מימוש לפונקציה divide_list_inplace(lst) שמקבלת רשימת מספרים שונים זה מזה, ומחלקת אותה כמתואר לעיל - **במקום**. הפונקציה תחזיר None.
אין להשתמש בפונקציות מוכנות של פייתון (גם לא פונקציות מיון)! יש לבצע את החלוקה המתוארת באופן ישיר.

פלט אפשרי לדוגמא (יתכנו פלטים חוקיים נוספים):

```
>>> lst = [4,1,7,8,3,5,6,2]
>>> divide_list_inplace(lst)
>>> lst
[2, 1, 3, 4, 8, 7, 5, 6]
```

הנחיות הגשה:

1. בקובץ ה pdf הגישו:
   - את התשובות לסעיפים א',ב'.
2. בקובץ ה py הגישו:
   - את הפונקציות שמימשתם בסעיפים ג',ד', ה'.

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

שאלה 2

בשאלה זו ננתח את מספר פעולות הכפל שמתבצעות בחישוב a^b **(בשאלה זו הניחו כי a,b שלמים חיוביים)**.

הפונקציה הבאה (שראינו בכיתה, בהרצאה 5) מקבלת שני פרמטרים a,b ומחשבת את a^b.

```
def power(a,b):
    """ computes a**b using iterated squaring """
    result=1
    while b>0: # b is nonzero
        if b % 2 == 1: #  b is odd
            result = result*a   # (פעולת כפל אחת)
        a = a*a  # (פעולת כפל אחת)
        b = b//2
    return result
```

נרצה לספור כמה פעולות כפל מתבצעות ע"י הפונקציה power (השורות בהן מבוצעת פעולת כפל מסומנות לנוחיותכם בקוד שלעיל).

נתונים לנו שני מספרים עשרוניים: a שהייצוג הבינארי שלו מכיל n ביטים, ו- b שהייצוג הבינארי שלו מכיל m ביטים.

א. תנו <u>ביטוי מדויק</u> כפונקציה של n ו/או m למספר <u>הקטן ביותר</u> האפשרי של פעולות כפל שמתבצעות ע"י power?

תנו <u>ביטוי מדויק</u> כפונקציה של n ו/או m למספר <u>הגדול ביותר</u> האפשרי של פעולות כפל שמתבצעות ע"י power?

שימו לב שהתשובות בשני הסעיפים צריכות להיות כלליות (עבור כל n ו- m), ולא עבור מספרים ספציפיים.

בכל אחד מהסעיפים הללו הסבירו את התשובה, וציינו מה צריך להיות המבנה של a ו/או b כדי שנקבל מספר של פעולות כפי שציינתם.

ב. השלימו בקובץ השלד את הפונקציה power_new שבהינתן a, b מחשבת את a^b תוך ביצוע אותו מספר של פעולות כפל כמו הפונקציה power שלמעלה. יש להשלים 3 שורות בלבד.

```
def power_new(a,b):
    """ computes a**b using iterated squaring """
    result = 1
    b_bin = bin(b)[2:]
    reverse_b_bin = b_bin[: :-1]
    for  bit in reverse_b_bin:
     ____________________________________________
     ____________________________________________
     ____________________________________________
    return result
```

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

שאלה 3

בשאלה זו נעסוק בהיבטים שונים של ייצוג מספרים בבסיסים שונים.

א. כמה ביטים יש בייצוג הבינארי של המספר העשרוני 3^{2017}?

ענו על השאלה בשתי דרכים שונות: (1) תוך שימוש בנוסחה שראיתם בתרגול, (2) באמצעות הפיכת המספר מ-integer ל-string, ומדידת אורך המחרוזת באמצעות len().

צרפו לקובץ ה pdf את התשובה, וכן שתי שורות קוד בודדות (לא כולל פקודת import) – אחת לכל אופן פתרון - שמבצעות את החישוב.

הדרכה: מומלץ להשתמש בפונקציה log(x,base) של המודול math, אשר מחשבת את הלוגריתם של x בבסיס base.

ב. ממשו את הפונקציה inc(binary) (קיצור של increment) אשר מקבלת מחרוזת המייצגת מספר טבעי בכתיב בינארי (כלומר מחרוזת המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי לאחר תוספת של 1.הערה: מחרוזת שמייצגת מספר בינארי לא תכיל אפסים מובילים (משמאל), למעט המספר 0 שמיוצג ע"י המחרוזת "0".

דוגמאות הרצה:

```
>>> inc("0")
'1'
>>> inc("1")
'10'
>>> inc("101")
'110'
>>> inc("111")
'1000'
>>> inc(inc("111"))
'1001'
```

להלן המחשה של אלגוריתם החיבור של מספרים בינאריים (בדומה לחיבור מספרים עשרוניים עם נשא (carry)):

```
          1  _(carried digits)
        1 0 1 (binary)
+           1
-------------
=       1 1 0
```

הנחיה: אין להמיר את binary לבסיס עשרוני. בפרט, אין להשתמש כלל בפונקציה int של פייתון או בפונקציה convert_base מתרגול 3. יש לממש את האלגוריתם בהתאם להמחשה: ישירות באמצעות לולאות.

ג. נתון מספר טבעי N. בכל אחד מהסעיפים הבאים, ציינו מהו המספר המינימלי ומהו המספר המקסימלי של ביטים שדרושים לייצוג הבינארי (בסיס 2) של מספר זה. אם המינימום והמקסימום שווים זה לזה, הסבירו מדוע זה המצב. (המשך השאלה בעמוד הבא)

I. המספר הוא טבעי בעל 5 ספרות בבסיס עשר

II. המספר הוא טבעי בעל 5 ספרות בבסיס 16 (הֶקסָדצימלי), והספרה השמאלית ביותר שלו הינה גדולה או שווה ל 8 (כלומר ערכו של המספר בבסיס עשרוני הוא <u>לפחות</u> 524288).

ד. נתון מספר טבעי 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.

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

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

<u>שאלה 4</u>

בהינתן טבעי n, כל טבעי k <u>**שקטן ממש**</u> מ-n ומחלק את n נקרא **מחלק ממש** של n.

סדרת מחלקים היא סדרת מספרים, שהראשון שבהם הוא מספר טבעי כלשהו, וכל איבר שווה לסכום המחלקים ממש של קודמו בסדרה. אם מגיעים ל- 0 – הסדרה מסתיימת (שכן ל- 0 אין מחלקים).

למשל, להלן סדרת מחלקים שמתחילה ב- 45:

$$45 \mapsto 33 \mapsto 15 \mapsto 9 \mapsto 4 \mapsto 3 \mapsto 1 \mapsto 0$$

סדרות מחלקים נחלקות ל- 3 סוגים:

1. סדרות שמסתיימות ב- 0 (סדרות סופיות).
2. סדרות שלא מגיעות ל- 0 לעולם (אינסופיות), שנכנסות למעגל מחזורי כלשהו.
3. סדרות שלא מגיעות ל- 0 לעולם (אינסופיות), שלא נכנסות למעגל מחזורי כלשהו (סדרות כאלו חייבות להיות בלתי חסומות).

ישנן סדרות שלא ידוע לאיזה סוג הן שייכות. למעשה לא ידוע אם בכלל יש סדרות מהסוג השלישי – זוהי שאלה פתוחה במתמטיקה.

א. הסבירו בקצרה מדוע יש אינסוף סדרות מהסוג הראשון (רמז: מספרים ראשוניים), ומדוע יש סדרות מהסוג השני (רמז: מספרים מושלמים).

ב. ידוע שכל המספרים בתחום בין 1 ל- 275 (כולל) מתחילים סדרה ששייכת לאחד משני הסוגים הראשונים הנ"ל (לגבי 276 אגב, לא ידוע כיום לאיזה סוג הוא שייך...). נרצה לדעת כמה מתוכם הם התחלה של סדרה סופית (מסוג 1).

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

לשם כך עליכם להשלים בקובץ השלד את הפונקציות הבאות:

3. הפונקציה sum_divisors(n), אשר מקבלת מספר שלם חיובי n, ומחזירה את סכום המחלקים-ממש שלו.

דוגמאות הרצה (בעמוד הבא):

```
>>> sum_divisors(4)
3
>>> sum_divisors(220)
284
```

<u>הנחייה מחייבת</u>: בפונקציה לא תהיה יותר מלולאה אחת. עבור קלט n, על הלולאה שבפונקציה לבצע לא יותר מ- $\sqrt{n}$ איטרציות. פונקציות שמבצעות מספר איטרציות בסדר גודל גבוה יותר (למשל בערך $n/2$ איטרציות) אינן עומדות בתנאי זה.

4. הפונקציה is_finite(n) שמחזירה True אם סדרת המחלקים שמתחילה במספר n היא סופית, כלומר שייכת לסוג 1, ו- False אם היא שייכת לסוג 2.

דוגמת הרצה:

```
>>> is_finite(4)
True
>>> is_finite(220)
False
```

<u>הנחיה מחייבת</u>: השתמשו ב- sum_divisors.

5. הפונקציה cnt_finite(limit) שמחזירה כמה מספרים בין 1 ל- limit (כולל) הם התחלה של סדרה סופית.

<u>הנחיה מחייבת</u>: הפונקציה תקרא ל- is_finite, וזו בתורה תקרא כאמור ל- sum_divisors.

בקובץ ה- pdf רישמו את התשובה עבור limit=275.

ג. התבוננו שוב בהגדרת הפלט של is_finite. מיכל הציעה לדרוש שהפונקציה תחזיר False בכל מקרה שהסדרה אינה מסוג 1, כלומר גם כאשר היא מסוג 2 <u>וגם כאשר היא מסוג 3</u>. אמיר התנגד להוספת דרישה זו לתרגיל. מדוע?

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

שאלה 5

בשאלה זו נתאר מספר פונקציות שעוסקות במציאת אורכה של תת-המחרוזת הרצופה המשותפת הארוכה ביותר (Longest Common Substring, מקוצר כ LCS) של שתי מחרוזות.

הגדרה: המחרוזת s היא LCS של המחרוזות s1 ו- s2 אם ורק אם מתקיימות התכונות הבאות:

1. המחרוזת s היא תת מחרוזת של s1.
2. המחרוזת s היא תת מחרוזת של s2.
3. אין מחרוזת שארוכה יותר מ s ומקיימת את תנאים 1 ו 2.

לדוגמה, ה LCS של "ababc" ושל "dbabca" היא "babc", וה LCS של "xxx" ושל "abcd" היא "" (המחרוזת הריקה). שימו לב שייתכן כי קיימת יותר מ LCS אחת לזוג מחרוזות נתון.

נסמן ב n1, n2 את אורכי המחרוזות s1, s2 בהתאמה.

א. השלימו בקובץ השלד את מימוש הפונקציה has_common(s1, s2, k) אשר בודקת האם קיימת תת-מחרוזת משותפת באורך k לשתי מחרוזות נתונות s1, s2. נשתמש בפונקציה זו במימוש הפונקציה lcs_length_1 (המופיעה בהמשך) שמחשבת את אורך ה LCS המקסימלי של זוג מחרוזות נתון.

```
def has_common(s1,s2,k):

   if len(s1)<k or len(s2)<k:
       return False
   something = []
   for i in ____________________________________________:

       something.append(_____________________________________)

   for i in_____________________________________________:

       if ____________________________________________________:

           return True

   return False


def lcs_length_1(s1,s2):
    k=1
    while has_common(s1,s2,k) == True:
        k+=1
    return k-1
```

דוגמאות הרצה (המשך בעמוד הבא):

```
>>> has_common("ababc", "dbabca", 5)
False
>>> has_common("ababc", "dbabca", 4)
True
```

```
>>> has_common("ababc", "dbabca", 3)
True
>>> has_common("ababc", "dbabca", 2)
True
>>> has_common("", "dbabca", 2)
False

>>> lcs_length_1("ababc", "dbabca")
4
```

ב. השלימו בקובץ השלד את הפונקציה lcs_length_2, אשר מקבלת שתי מחרוזות ומחזירה את אורך ה LCS שלהן בדרך נוספת. הפונקציה תפעל באופן הבא: היא תשתמש בטבלת עזר שנסמנה m , בעלת n1 שורות ו- n2 עמודות. טבלה זו תהיה למעשה רשימה דו מימדית, כלומר רשימה באורך n1 שאיבריה הם רשימות באורך n2 כל אחת.

התא שנמצא בשורה ה-i ובעמודה ה-j של m יחושב באופן הבא:

- אם s1[i] שונה מ- s2[j], אז m[i][j] יהיה 0.
- אחרת, m[i][j] יחושב מתוך ערכו של תא סמוך כלשהו שחושב קודם (עליכם להבין לבד מאיזה וכיצד).

האלגוריתם יעבור על תאי הזיכרון m לפי סדר השורות משמאל לימין וימלא את הערך המתאים בכל תא, תוך ניצול המידע שנשמר בתאים הקודמים לו.

```
def lcs_length_2(s1,s2):

    if len(s1)==0 or len(s2)==0:
        return 0
    m = [[0]*len(s2) for i in range(len(s1))]
    for i in range(len(s1)):
        for j in range(len(s2)):

    return max([_____ for l in m])
```

דוגמאות הרצה:

```
>>> lcs_length_2("ababc", "dbabca")
4
>>> lcs_length_2("dbabca", "ababc")
4
>>> lcs_length_2("xxx", "ababc")
0
```

ג. נתונה הפונקציה gen_str שמשמשת ליצירת מחרוזת אקראית מתווים מתוך אלפבית נתון באורך נתון n.

```
import random
def gen_str(n, alphabet):
    return "".join([random.choice(alphabet) for i in range(n)])
```

אמיר השתמש בפונקציה gen_str על מנת ליצור שתי מחרוזות אקראיות שונות באורך 4000 כל אחת, וניסה למצוא את אורך ה LCS באמצעות שתי הפונקציות lcs_length_1, lcs_length_2. אמיר מדד את הזמנים שלקח לכל אחת מהפונקציות לרוץ.
להלן הקוד שהריץ:

```
n=4000
alphabet = "abcdefghijklmnopqrstuvwxyz"
print("two random strings of length", n)
s1 = gen_str(n, alphabet)
s2 = gen_str(n, alphabet)

t0 = time.clock()
res1 = lcs_length_1(s1,s2)
t1 = time.clock()
print("lcs_length_1", res1, t1-t0)

t0 = time.clock()
res2 = lcs_length_2(s1,s2)
t1 = time.clock()
print("lcs_length_2", res2, t1-t0)
```

הפלט שקיבל הינו:

```
two random strings of length 4000
lcs_length_1 5 0.39284177015987076
lcs_length_2 5 3.0407691484716426
```

מיכל יצרה מחרוזת אקראית אחת באורך 4000 באמצעות הפונקציה gen_str והריצה את הקוד הבא:

```
n=4000
alphabet = "abcdefghijklmnopqrstuvwxyz"
print("two identical random strings of length", n)
s1 = gen_str(n, alphabet)
s2 = s1

t0 = time.clock()
res1 = lcs_length_1(s1,s2)
t1 = time.clock()
print("lcs_length_1", res1, t1-t0)

t0 = time.clock()
res2 = lcs_length_2(s1,s2)
t1 = time.clock()
print("lcs_length_2", res2, t1-t0)
```

הפלט שקיבלה הינו :

```
two identical random strings of length 4000
lcs_length_1 4000 4.498908671782961
lcs_length_2 4000 3.0692516020038885
```

בקובץ ה pdf הסבירו בקצרה (לא יותר מפיסקה אחת) מדוע מתקבלות תוצאות אלו, ובפרט, שימו דגש בהסבר <u>על השוני</u> בין זמני הריצה של lcs_length_1 <u>ועל הדמיון</u> בין זמני הריצה של lcs_length_2.

<u>סוף.</u>