אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2020

תרגיל בית מספר 2 - להגשה עד 24 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 2020

שאלה 1

להלן פונקציה שהופכת בין מפתחות (keys) לערכים (values) של מילון.

```
def reverse_dict(d):
     return {y:x for x,y in d.items()}
```

א. הריצו את reverse_dict על מילון שמייצג פונקציה (מתמטית) כלשהי שאינה חד-חד ערכית. הסבירו במשפט מה מתרחש. הריצו את reverse_dict על מילון שבו ה- values הם טיפוסים בפייתון שאינם immutable, והסבירו במשפט מה הבעיה.

כאשר פונקציה מקבלת פרמטר mutable, היא יכולה לשנות אותו במקום (in-place), או לייצר אובייקט חדש. לדוגמה, הפונקציה sorted של פייתון יכולה לשמש למיון של רשימות, והיא יוצרת רשימה חדשה (ממויינת) בזיכרון. לעומת זאת כאשר כותבים ()L.sort עבור רשימה L, המיון נעשה במקום (ומה מחזירה הפונקציה ()sort של המחלקה list? בידקו).

ב. האם הפונקציה reverse_dict הנ״ל מייצרת מילון חדש או משנה את האיברים של המילון שהיא מקבלת? כתבו את תשובתכם בקובץ ה- pdf. הצדיקו את תשובתכם ע״י שימוש בפונקציה id של פייתון וצירוף הפלט לתשובה.

ג. להלן ניסיון לממש פונקציה כנ״ל, שבה השינוי יהיה במילון שהתקבל בפרמטר לפונקציה (כלומר לא ייווצר מילון חדש, אלא הערכים הפנימיים של d ישתנו).

```
def reverse_dict(d):
     for k in d:
          d[d[k]] = k
          d.pop(k)
```

הסבירו בדיוק מדוע הפונקציה הזו לא עובדת נכון. לשם כך, מומלץ להיעזר ב- Pythontutor שתראו בתרגול 3, או להוסיף פקודות הדפסה, כדי לעקוב אחר מה שמתרחש.

ד. ממשו בקובץ השלד גרסה של הפונקציה הנ״ל שנקראת reverse_dict_in_place(d), שתשנה את המילון שהיא קיבלה, **במקום** (כלומר לא ייווצר מילון חדש, אלא הערכים הפנימיים של d ישתנו). ודאו שהכתובת בזיכרון של המילון שהועבר לפונקציה לא השתנה. לדוגמה:

```
>>> d = {1:2, 3:"a"}
>>> hex(id(d))
'0x2aab198'
>>> reverse_dict_in_place(d)
>>> d
{2: 1, 'a': 3}
>>> hex(id(d))
'0x2aab198'
```

ה. אמיר רצה לכתוב פונקציה שתקבל כקלט רשימת מפתחות (ייחודיים) keys ורשימת ערכים vals ותחזיר מילון בו המפתח של הערך במקום ה- i ברשימה keys הוא המפתח במקום ה- i ברשימה vals. אמיר כתב את הפונקציה הבאה:

```
def build_dict(keys, vals):
    d = {}
    for key in keys:
        for val in vals:
            if keys.index(key) == vals.index(val):
                d[key] = val
    return d
```

מיכל לעומת זאת, כתבה פונקציה אחרת:

```
def build_dict2(keys, vals):
    d = {}
    for i in range(len(keys)):
        d[keys[i]] = vals[i]
    return d
```

נניח שהקלט לשתי הפונקציות הוא רשימות keys ו- vals באורך n כל אחת. תנו ביטוי מדויק למספר האיטרציות שתבצע כל אחת מן הפונקציות כפונקציה של n.

ו. הריצו את הקוד הבא, צרפו לקובץ ה- pdf את התוצאות, והסבירו כיצד התוצאות מתיישבות עם הביטויים שמצאתן בסעיף הקודם.
הנחיה: בקובץ השלד **<u>אין</u>** להגיש את הקוד המצורף אלא להריץ אותו בנפרד ולתעד את התוצאות בקובץ ה-pdf בלבד.

```
import time

keys = [i for i in range(1000)]
vals = [i for i in range(1000)]


start = time.perf_counter()
build_dict(keys, vals)
end = time.perf_counter()
print("build_dict time: ", end - start)

start = time.perf_counter()
build_dict2(keys, vals)
end = time.perf_counter()
print("build_dict2 time: ", end - start)
```

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, במצגת 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 ביטוי מדויק כפונקציה של n ו/או m למספר הקטן ביותר האפשרי של פעולות כפל שמתבצעות ע"י power.

תנו ביטוי מדויק כפונקציה של n ו/או m למספר הגדול ביותר 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
```

ג. להלן המימוש שראינו בכיתה עבור modular exponentiation שבהינתן a, b, c מחשבת את $a^b \% c$:

```
def modpower(a, b, c):
    """ computes a**b modulo c, using iterated squaring
        assumes b is a nonnegative integer """
    result = 1
    while b > 0:
        if b % 2 == 1:
            result = (result * a) % c
        a = (a * a) % c
        b = b // 2
    return result
```

השלימו בקובץ השלד את הפונקציה modpower_new שבהינתן a, b, c מחשבת את $a^b \% c$ בדרך מעט שונה מאשר זו של modpower. תוכן הפונקציה נמצא בקובץ השלד בהערה. עליכם להסיר את הפקודה pass שבראשית הפונקציה, להסיר את סימני ההערות מהקוד ולהשלים את השורות החסרות בלבד. בקוד אותו אתם משלימים אין לבצע פעולות מודולו.

```
def modpower_new(a, b, c):
    """ computes a**b mod c using iterated squaring
        assumes b is nonnegative integer  """
    result = 1 # a**0
    while b > 0:
        if b % 4 == 0:
            result = (__________________________) % c
            a = (______________________________) % c
        if b % 4 == 1:
            result = (__________________________) % c
            a = (______________________________) % c
        if b % 4 == 2:
            result = (__________________________) % c
            a = (______________________________) % c
        if b % 4 == 3:
            result = (__________________________) % c
            a = (______________________________) % c
        b = b // _______
     return result
```

<u>שאלה 3</u>

בשאלה זו נעסוק בהיבטים שונים של ייצוג מספרים בבסיסים שונים.

א. כמה ביטים יש בייצוג הבינארי של המספר העשרוני 11^{2020}?

ענו על השאלה בשתי דרכים שונות: (1) תוך שימוש בנוסחה שראיתם בתרגול, (2) באמצעות הפיכת המספר מ- integer למחרוזת מתאימה 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
```

<u>הנחיה</u>: 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. ממשו את הפונקציה dec(binary) (קיצור של decrement) אשר מקבלת מחרוזת המייצגת מספר טבעי (גדול או שווה ל 1) בכתיב בינארי (כלומר מחרוזת המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי לאחר חיסור של 1.

הערות:

- מחרוזת שמייצגת מספר בינארי לא תכיל אפסים מובילים משמאל (למשל, "001" איננה תקינה שכן היא מכילה אפסים מובילים משמאל)
- ניתן להניח שהמחרוזת "0" לא תינתן כקלט.

דוגמאות הרצה:

```
>>> dec("1")
'0'
>>> dec("101")
'100'
>>> dec("100")
'11'
>>> dec(dec("100"))
'10'
```

<u>הנחיה</u>: אין להמיר את binary לבסיס עשרוני. בפרט, אין להשתמש כלל בפונקציה int של פייתון או בפונקציה convert_base מתרגול 3.

ד. ממשו את הפונקציה sub(bin1, bin2) אשר מקבלת שתי מחרוזות המייצגות מספרים טבעיים בכתיב בינארי (כלומר מחרוזות המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי המתקבל מחיסור bin2 מ-bin1. כמו בסעיף הקודם: מחרוזת שמייצגת מספר בינארי לא תכיל אפסים מובילים (משמאל), למעט המספר 0 שמיוצג ע"י המחרוזת "0".

הניחו (אין צורך לבדוק זאת) כי המספר bin2 <u>קטן או שווה</u> למספר bin1.

דוגמאות הרצה:

```
>>> sub('10', '0')
'10'
>>> sub('0', '0')
'0'
>>> sub('1000', '11')
'101'
```

<u>הנחיה</u>: אין להמיר את מחרוזות הקלט לבסיס עשרוני. בפרט, אין להשתמש כלל בפונקציה int של פייתון או בפונקציה convert_base מתרגול 3. בנוסף, על הפונקציה sub לקרוא לפונקציה dec מהסעיף הקודם.

ה. נתון מספר טבעי עשרוני N. בן ונעם 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. בן רשם את המספר בכתיב בינארי, ונעם הוסיף למספר k פעמים 1 מימין.

II. בן רשם את המספר בכתיב בינארי, ונעם הוסיף למספר 1 מצד שמאל.

III. בן רשם את המספר בכתיב בינארי, ונעם הוסיף למספר k פעמים 1 מצד שמאל.

ו. בשקף 14 של מצגת 4b נטען כי מספר בעל d ספרות עשרוניות ידרוש לכל היותר $\lceil d\log_2 10\rceil$ ביטים בכתיב בינארי. למעשה, הדבר נכון לכל שני מספרים טבעיים $b, c>1$: מספר בעל d ספרות בבסיס b ידרוש לכל היותר $\lceil d\log_c b\rceil$ ספרות בבסיס c.
הוכיחו טענה זו. השתמשו בטענה כי מספר N בעל d ספרות בבסיס כלשהו $b>1$ מקיים $b^{d-1} \le N < b^d$.
שימו לב כי ניתן לכתוב את הוכחה מלאה בכ- 2-3 שורות קצרות, עם כמה מעברים אלגבריים פשוטים.

ז. מהו מספר הפעולות הבינאריות **המקסימלי** שמתבצע בפעולת חיבור של שני מספרים טבעיים כפונקציה של n, כאשר n הינו מספר הביטים בייצוג כל אחד משני המספרים? סיפרו רק פעולות חיבור של שני ביטים (העתקת ביט למשל לא תיספר כפעולה).
מה יהיה מספר הביטים המקסימלי בתוצאה?
הסבירו את תשובותיכם לשתי השאלות.

שאלה 4

בהינתן מחרוזת לא ריקה s באורך n ומספר טבעי k שמקיים $1 \leq k \leq n$, נרצה לדעת האם s מכילה תת מחרוזת רצופה באורך k שחוזרת על עצמה.

למשל המחרוזת $s =$ "$ababa$" מכילה תת מחרוזת באורך 3 שחוזרת עצמה ("aba") אבל לא מכילה תת מחרוזת כזו באורך 4.

א. השלימו בקובץ השלד את מימוש הפונקציה *has_repeat1(s, k)* שמקבלת מחרוזת *s* לא ריקה ומספר טבעי k ומחזירה *True* אם *s* מכילה תת מחרוזת רצופה באורך *k* שחוזרת על עצמה, אחרת מחזירה *False*. אין צורך לבדוק את תקינות הקלטים שהפונקציה מקבלת.
במימושכם עליכם להשתמש בזיכרון עזר מסוג אוסף (למשל רשימה, *set* וכו') שיכיל לכל היותר *n* איברים, כאשר *n* הינו אורך המחרוזת *s*. בנוסף, אין להשתמש בלולאה מקוננת (לולאה בתוך לולאה).

ב. השלימו בקובץ השלד את מימוש הפונקציה *has_repeat2(s, k)* שפועלת בדומה לפונקציה מסעיף א׳. בשונה מהמימוש הקודם הפעם אין להשתמש בזיכרון עזר כנ״ל, אך מותר להשתמש בלולאה מקוננת.

שאלה 5

הוסיפו לקובץ השלד המצורף מימוש לפונקציה sum_divisors(n), אשר מקבלת מספר שלם חיובי n, ומחזירה את סכום המחלקים-ממש שלו.

<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.

דוגמאות הרצה:

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

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2020

<u>שאלה 6</u>

בשאלה זו נעסוק בבדיקת השערת גולדבאך. השערת גודלבאך אומרת כי כל מספר זוגי גדול מ-2 הוא סכום של שני מספרים ראשוניים.

לקובץ התרגיל צורף הקובץ primes.txt אשר מכיל את כל המספרים הראשוניים עד 10000.

א. השלימו בקובץ השלד את הפונקציה parse_primes(filename) שמקבלת את שם הקובץ כמחרוזת ומחזירה קבוצה (set) של כל המספרים הראשוניים הכתובים בקובץ.

ב. השלימו בקובץ השלד את הפונקציה check_goldbach_for_num(n, primes_set) אשר מקבלת מספר זוגי גדול מ-2 בשם n וקבוצת מספרים ראשוניים primes_set, ומחזירה True אם המספר מקיים את ההשערה עבור הקבוצה (כלומר, ישנם שני ראשוניים בקבוצה אשר סכומם הוא n) ו- False אחרת. ניתן להניח כי הקלט תקין.

ג. השלימו את הפונקציה check_goldbach_for_range(limit, primes_set) אשר מקבלת מספר גדול מ-2 בשם limit וקבוצת מספרים ראשוניים primes_set ובודקת את ההשערה עבור כל המספרים הזוגיים הגדולים מ- 2 עד ל- limit (לא כולל). כלומר, הפונקציה תחזיר True אם כל מספר זוגי הגדול מ-2 וקטן מ-limit ניתן להצגה כסכום של שני ראשוניים בקבוצה primes_set ו-False אחרת.

בדקו את ההשערה עם limit=10000 וקבוצת הראשוניים שיצרתם בסעיף א׳. <u>**כתבו בקובץ ה- pdf מהו זמן הריצה של הפונקציה.**</u>

ד. כעת נרצה לאסוף סטטיסטיקות על המספרים שמקיימים את ההשערה ובפרט, כמה זוגות ראשוניים יכולים להרכיב מספר מסוים. למשל, 7+23=30 אבל גם 11+19=30. כלומר יתכנו מספר זוגות ראשוניים שמרכיבים את אותו מספר זוגי. כמו בסעיפים הקודמים, הניחו כי הקלט primes_set הוא אוסף מספרים ראשוניים.

- השלימו בקובץ השלד את הפונקציה check_goldbach_for_num_stats(n, primes_set) אשר תחזיר את מספר זוגות הראשוניים ב-primes_set שיכולים להרכיב מספר n המתקבל כקלט.
- השלימו בקובץ השלד את הפונקציה check_goldbach_stats(limit, primes_set) אשר תחזיר מילון בו המפתחות יהיו מספר זוגות הראשוניים ב-primes_set המרכיבים את כל המספרים עד limit (לא כולל). הערך של כל מפתח יהיה כמות המספרים הזוגיים שיכולים להיות מורכבים ממספר זוגות זה. למשל, אם יש חמישה מספרים זוגיים שיכולים להיות מורכבים ע״י שני זוגות ראשוניים, המילון יכיל את הכניסה d[2] = 5.
  <u>הערה :</u> שימו לב לא לספור זוגות פעמיים!
- הריצו את check_goldbach_stats(limit, primes_set) עבור limit=1000 <u>**וצרפו לקובץ ה- pdf את המילון המוחזר**</u>.

<u>דוגמאות הרצה :</u>

```
>>> check_goldbach_for_num_stats(20, primes_set)
2 # 20 שני זוגות ראשוניים מרכיבים את
>>> check_goldbach_for_num_stats(10, primes_set)
```

2 # שני זוגות ראשוניים מרכיבים את 10

>>> check_goldbach_stats(11, primes_set)

{1: 3, 2: 1} # 4,6,8 מורכבים מזוג אחד ו- 10 מורכב משני זוגות

<u>סוף.</u>