אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2021

תרגיל בית מספר 2 - להגשה עד 16/11/2020 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.
- מכיוון שניתן להגיש את התרגיל בזוגות, עליכם בנוסף למלא את המשתנה SUBMISSION_IDS שבתחילת קובץ השלד. רק אחת הסטודנטיות בזוג צריכה להגיש את התרגיל במודל.

- 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 2021

<u>הערה כללית לתרגיל</u>: בקובץ השלד המצורף יש בחלק מהפונקציות פקודת assert. הפקודה בודקת קיום של תנאי ובמידה והוא לא מתקיים היא מחזירה שגיאה (מסוג שגיאת זמן ריצה) ומסיימת את ריצת התוכנית. מטרת הפקודה היא לאפשר למתכנת לודא שאחרי פקודה זו התנאי מתקיים, כלומר זוהי הנחה על ערכי המשתנים בשלב זה בריצת התוכנית. לצורך בקרה שלכם על הארגומנטים (קלטים) שאתם נותנים לפונקציות, אנחנו ממליצים לא למחוק פקודות אלו.

<u>שאלה 1</u>

ממשו את הפונקציה poker_hand(hand), המקבלת כקלט מחרוזת המתארת "יד" פוקר ומחזירה מחרוזת המתארת את "חוזק" היד.
מחרוזת הקלט היא מהצורה "NS NS NS NS NS", המתארת 5 קלפים, כאשר כל קלף מתואר ע"י שני תווים שכנים- תו מסוג N מתאר את מספר הקלף ותו מסוג S מתאר את צורת הקלף. בין הקלפים יש תו רווח.
ערכי N האפשריים: 2 עד 9 וגם T, J, Q, K, A. ערכי S האפשריים: S, C, D, H:

תו N	ערך מספרי של התו
"2"	2
"3"	3
...	...
"9"	9
"T" (10)	10
"J" (נסיך)	11
"Q" (מלכה)	12
"K" (מלך)	13
"A" (אס)	14

תו S	צורה
"S"	עלה
"H"	לב
"D"	יהלום
"C"	תלתן

החוזקות האפשריות של יד פוקר (מהחלש לחזק):
"High Card" – קלף גבוה.
"One Pair" – זוג קלפים בעלי אותו המספר.
"Two Pairs" – שני זוגות קלפים בעלי אותו המספר.
"Three of a Kind" – שלושה קלפים בעלי אותו המספר.
"Straight" – כל 5 הקלפים בעלי מספרים עוקבים (למשל 7, 8, 9, T, J).
"Flush" – כל 5 הקלפים בעלי אותה הצורה.
"Full House" – שלשה וזוג (למשל 8, 8, 8, Q, Q).
"Four of a Kind" – ארבעה קלפים בעלי אותו המספר.
"Straight Flush" – כל 5 הקלפים גם עוקבים במספר וגם בעלי אותה הצורה.
"Royal Flush" – כמו "Straight Flush" אבל ספציפית עבור המספרים 10 עד A.

חוזק היד מוגדר לפי התנאי החזק ביותר שהיא מקיימת (המאוחר ביותר ברשימה הנ"ל).

- להבהרות נוספות על חוקי ידיים בפוקר ניתן להיוועץ בגוגל.
- שימו לב שבשאלה זו הערך המספרי של אס (A) הוא 14 בלבד.
- למניעת שגיאות דפוס שיגרמו לאיבוד נקודות בשאלה, הוספנו רשימה של מחרוזות התשובות האפשריות בתחילת הפונקציה בקובץ השלד – העתיקו אותן במדויק למקומות המתאימים בקוד שלכם. (לחילופין, עשו uncomment לרשימה והחזירו איבר מתאים מהרשימה.)

דוגמאות הרצה:

```
>>> poker_hand("5H 5C 6S 7S KD")
'One Pair'

>>> poker_hand("5D 8C 9S JS AC")
'High Card'

>>> poker_hand("3D 6D 7D TD QD")
'Flush'

>>> poker_hand("3C 3D 3S 9S 9D")
'Full House'

>>> poker_hand("AC TC JC KC QC")
'Royal Flush'

>>> poker_hand("AC TC JC KC QD")
'Straight'
```

שאלה 2

בשאלה זו נממש מספר פונקציות אקראיות, ללא שימוש בפונקציות אחרות מהספריה random מלבד הפונקציה הבסיסית random.random().

כזכור, הספריה random מכילה פונקציה בשם random(), שמחזירה מספר מטיפוס float בקטע [0,1), כאשר לכל מספר יש סיכוי שווה להיבחר:

```
>>> import random
>>> random.random()
0.13937543523525686
>>> random.random()
0.6376812941041776
```

א. ממשו את הפונקציה coin() שמחזירה True בסיכוי חצי ו False בסיכוי חצי.

ב. ממשו את הפונקציה uniform(a, b), שמחזירה מספר ממשי אקראי בקטע $[a, b)$, כאשר לכל מספר יש סיכוי שווה להיבחר. הניחו כי $b > a$.

ג. ממשו את הפונקציה choice(seq), שמקבלת אוסף סדור מסוג list או str ומחזירה באקראי את אחד מאיבריו, כאשר לכל איבר יש סיכוי שווה להבחר.

ד. ממשו את הפונקציה weighted_choice(seq, weights) שמקבלת אוסף סדור מסוג list או str ורשימת משקלים. הפונקציה מחזירה את האיבר seq[i] בסיכוי weights[i]. הניחו כי sum(weights) = 1 וגם len(seq) == len(weights).

ה. ממשו את הפונקציה get_biased_coin(p), שמקבלת הסתברות $0 \geq p \geq 1$ **ומחזירה פונקציה** שמטילה מטבע שמחזיר True בהסתברות p. השתמשו בתחביר lambda.

ו. ממשו את הפונקציה test_biased_coin(p, num_flips) אשר קוראת ל(get_biased_coin(p ואז בודקת האם פונקצית המטבע שהתקבלה באמת מחזירה True בהסתברות p. על הפונקציה להטיל את המטבע num_flips פעמים ולהחזיר את הסיכוי האמפירי (הנצפה) של המטבע להחזיר p.

<u>שימו לב</u>: הבדיקות של שאר סעיפי השאלה בפונקציה test לא בודקות שהפונקציות מחזירות תשובות בהסתברות הנכונה. בעזרת מנגנונים דומים לtest_biased_coin תוכלו לבדוק את שאר סעיפי השאלה.

<u>שאלה 3</u>

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

למשל המחרוזת "$ababa$" $= s$ 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 רשימה, שתכיל לכל היותר *n* איברים, כאשר *n* הינו אורך המחרוזת *s*. בנוסף, אין להשתמש בלולאה מקוננת (לולאה בתוך לולאה).

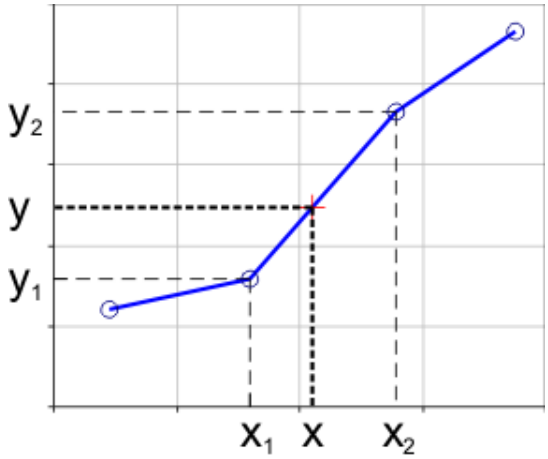
ב. השלימו בקובץ השלד את מימוש הפונקציה *has_repeat2(s, k)* שפועלת בדומה לפונקציה מסעיף א'. בשונה מהמימוש הקודם הפעם אין להשתמש בזיכרון עזר באורך משתנה (כמו רשימה או מבנה נתונים דומה. מותר מספר קבוע של משתני עזר), אך מותר להשתמש בלולאה מקוננת.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2021

שאלה 4

חמוטל, חוקרת בתחום רכב אוטונומי, עובדת על רכיב שמודד מהירות של רכב בזמנים שונים. מחשב הרכב שומר מדידות כזוג ערכים (x, y) (זמן ומהירות) בהפרשי זמן לא קבועים. לצורך ניתוח פעילות המכונית בדיעבד, נדרש לשערך את מהירות המכונית גם בזמנים שלא היתה בהם מדידה – כלומר בין מדידות. לשם כך, חמוטל החליטה להשתמש בטכניקה שנקראת *אינטרפולציה ליניארית*, בה כדי לשערך ערך של פונקציה בנקודה x מסוימת, מעבירים קו ישר בין נקודת המדידה הקרובה ביותר אליה משמאל ונקודת המדידה הקרובה ביותר אליה מימין ונותנים ערך y לנקודה לפי הערך של הקו הישר באותה הנקודה, כלומר, לפי המרחק היחסי של הנקודה משתי הנקודות השכנות. לדוגמה: אם יש מדידות בנקודות $(1, 10), (4, 40)$ אז ערך הy שניתן לנקודה הנמצאת ב $x = 3$ הוא 30.

מצורף גרף להמחשה: הנקודות המסומנות בעיגול כחול הן נקודות המדידה. הפונקציה תיתן את הערך y לנקודה x, לפי קו ישר בין הנקודות $(x_1, y_1), (x_2, y_2)$:



ממשו את הפונקציה interpolate(xy, x_hat) , אשר מקבלת אוסף נקודות מדידה (x, y), הניתנות כרשימה של זוגות סדורים (כלומר רשימה שכל איבר בה הוא tuple המכיל 2 מספרים) ורשימת ערכי x נוספים, $\hat{x}$. הפונקציה מחזירה את הנקודות לאחר אינטרפולציה – ערכי $\hat{x}$ הנתונים עם ערכי $\hat{y}$ המתאימים להם, כרשימה של זוגות סדורים. יש להחזיר את הנקודות באותו הסדר שהתקבלו.

הנחה: ערכי x_hat נמצאים בתוך טווח ערכי x, כלומר: $\forall \hat{x}_i \epsilon \hat{x}: \min(x) \leq \hat{x}_i \leq \max(x)$

דוגמאות הרצה:

```
>>> interpolate([(1, 10), (4, 40)], [4, 2, 3])
[(4, 40), (2, 20.0), (3, 30.0)]
>>> interpolate([(-3, 9), (-2, 4), (-1, 1), (0, 0), (1, 1), (2,
4), (3, 9)], [-2.5, -1.5, 0.5, 1.5, 2.5])
[(-2.5, 6.5), (-1.5, 2.5), (0.5, 0.5), (1.5, 2.5), (2.5, 6.5)]
```

הערה: ערכים שלמים ניתן להחזיר כint או float, כפי שמדגימה הדוגמה הראשונה.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2021

שאלה 5

הקדמה: שאלה זו משתמשת במבנה נתונים מסוג set. זהו מבנה נתונים שמייצג את האובייקט המתמטי קבוצה – אוסף של ערכים (למשל מספרים) ללא חשיבות לסדר וללא חזרות. set הוא טיפוס מובנה בפייתון, שמממש פעולות יעילות על קבוצה, כגון הוספה לקבוצה, חיתוך בין קבוצות וכו'. ניתן ליצור קבוצות בפייתון בעזרת סוגריים מסולסלים, או ע"י המרת רשימה לקבוצה בעזרת פקודת set:

```
>>> {2, 3, 5, 7, 7, 5}
{2, 3, 5, 7}
>>> set([2, 3, 5, 7, 7, 5])
{2, 3, 5, 7}
```

בנוסף, ניתן לבדוק האם איבר x נמצא בקבוצה s ע"י הפקודה x in s:

```
>>> 7 in {2, 3, 5, 7}
True
>>> 11 in {2, 3, 5, 7}
False
```

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 10000 המספרים הראשוניים הראשונים. עליכם לדאוג שקובץ זה ישב באותה תיקייה בה יושב קובץ השלד.

א. השלימו בקובץ השלד את הפונקציה check_goldbach_for_num(n, primes_set) אשר מקבלת מספר זוגי גדול מ-2 בשם n ורשימת 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.

דוגמאות הרצה:

```
>>> check_goldbach_for_num(10, {2, 3})
False
>>> check_goldbach_for_num(10, {2, 3, 5, 7})
True
```

ב. 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 2עד 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.

דוגמאות הרצה:

```
>>> check_goldbach_for_range(20, {2, 3, 5, 7, 11})
True
>>> check_goldbach_for_range(21, {2, 3, 5, 7, 11})
False
```

בדקו את ההשערה עם limit=10000 ורשימת הראשוניים שבקובץ המצורף. קראו את הקובץ בעזרת הפונקציה הנתונה parse_primes. **כתבו בקובץ ה pdf -מהו זמן הריצה של הפונקציה.**

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

- השלימו בקובץ השלד את הפונקציה check_goldbach_for_num_stats(n, primes_list) אשר תחזיר את מספר זוגות הראשוניים ב-primes_list 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 בprimes_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) כאשר limit=1000 וprimes_set היא קבוצת הראשוניים שנקראה מהקובץ ע"י parse_primes ו**<u>צרפו לקובץ ה-pdf את המילון המוחזר</u>**.

דוגמאות הרצה:

```
>>> check_goldbach_for_num_stats(20, primes_set)
2 # (שני זוגות ראשוניים מרכיבים את 20)
>>> 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 ו10 מורכב משני זוגות
```

<u>שאלה 6</u>

בשאלה זו נעסוק במימוש פעולות אריתמטיות על מספרים ביצוג בינארי.
את החומר הנדרש לפתרון שאלה זו נלמד בהרצאה של יום רביעי 4.11 ונתרגל בתרגול 4. ניתן לפתור את השאלה גם לפני התרגול.

להלן מספר הערות והנחיות התקפות לכלל הסעיפים בשאלה:

- לאורך השאלה נייצג מספרים בינאריים באמצעות מחרוזת המכילה את התווים "0" ו-"1" בלבד.
- לאורך השאלה אין לבצע המרה של אף מספר בינארי לבסיס עשרוני או לכל בסיס אחר. בפרט, אין להשתמש כלל בפונקציות int ו-bin של פייתון.
- לאורך השאלה, ניתן להניח כי מחרוזת הניתנת כקלט היא "תקינה", כלומר, מכילה אך ורק את התווים "0" ו-"1", וכי התו השמאלי ביותר במחרוזת הוא "1" (מלבד המחרוזת "0" אשר מייצגת את המספר 0). בפרט, מחרוזת למספר שאינו אפס לא תכיל אפסים מובילים והמחרוזת המייצגת את אפס תכיל "0" יחיד.
- לכל פונקציה בשאלה אשר מחזירה כפלט מחרוזת בינארית יש לוודא כי המחרוזת המוחזרת תקינה על פי ההגדרה הקודמת. (למשל, הפלטים "0100" ו-"000" אינם תקינים ואילו הפלטים "100" ו-"0" תקינים).
- לאורך השאלה נעבוד עם מספרים אי-שליליים בלבד. בפרט, ניתן להניח כי המחרוזות הבינאריות הניתנות כקלט לפונקציות השונות מייצגות מספרים אי-שליליים בלבד.
- הרצה של הפונקציות בשאלה על מחרוזות באורך של 20 ספרות צריכה להסתיים בזמן קצר (לכל היותר שניה).

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2021

סעיף א'

ממשו את הפונקציה add(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 מחיבור bin2 מ-bin1.

דוגמאות הרצה:

```
>>> add('10', '0')
'10'
>>> add('0', '0')
'0'
>>> add('1001', '11')
'1100'
```

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>: יש לממש את האלגוריתם בהתאם להמחשה: ישירות באמצעות לולאות.

סעיף ב'

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'
```

סעיף ג'

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.

<u>הנחיה</u>: יש לממש את הפונקציה בשורה אחת בלבד. יש להשתמש בסעיפים הקודמים.

דוגמאות הרצה:

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

סעיף ד'

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.

<u>הנחיה</u>: יש לממש את הפונקציה בשורה אחת בלבד. יש להשתמש בסעיפים הקודמים.

דוגמאות הרצה:

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

סעיף ה'

ממשו את הפונקציה mult(bin1, bin2) (קיצור של multiplication) אשר מקבלת שתי מחרוזות המייצגות מספרים טבעיים בכתיב בינארי (כלומר מחרוזות המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר את תוצאת הכפל של bin1 ב-bin2.

דוגמאות הרצה:

```
>>> mult("0", "10")
'0'
>>> mult("0", "0")
'0'
>>> mult("10", "1010")
'10100'
```

```
>>> mult("1", "1011")
'1011'
>>> mult("11", "111")
'10101'
```

הנחיה: ניתן ורצוי להשתמש בפונקציות מהסעיפים הקודמים.

הערה: לקבלת ניקוד מלא יש לממש את הפונקציה כך שפעולות על מחרוזות באורך 20 יסתיימו בפחות משניה אחת, כלומר, מימוש יחסית יעיל.

סוף.