אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

תרגיל בית מספר 3 - להגשה עד 10/04/2022 בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הנחיות לצורת ההגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- 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.
- עבור קובץ ה-pdf: מומלץ מאוד להקליד את התשובות בו. ניתן גם לכתוב את התשובות בכתב יד ברור ולסרוק אותן, אבל שימו לב שתשובות שיכתבו בכתב יד לא ברור לא יבדקו, וערעורים בנושא זה לא יתקבלו.
- עבור קובץ ה-py: השתמשו בקובץ השלד skeleton3.py כבסיס לקובץ אותו אתם מגישים.
- לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.

הנחיות לפתרון:

- הקפידו לענות על כל מה שנשאלתם.
- בכל שאלה, אלא אם מצוין אחרת באופן מפורש, ניתן להניח כי הקלט תקין.
- אין להשתמש בספריות חיצוניות פרט לספריות math, random, אלא אם נאמר במפורש אחרת.
- 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.
- כיוון שלמדנו בשבועות האחרונים כיצד לנתח את זמן הריצה של הקוד שלנו, החל מתרגיל זה ולאורך שארית הסמסטר (וכן במבחן) נדרוש שכל הפונקציות שאנו מממשים תהיינה יעילות ככל הניתן. לדוגמה, אם ניתן לממש פתרון לבעיה בסיבוכיות $O(logn)$, ואתם מימשתם פתרון בסיבוכיות $\Theta(n)$, תקבלו ניקוד חלקי על הפתרון.

טבלת גרסאות:

26.03.2022	גרסה ראשונה
27.03.2022, 14:30 **(אחרונה)**	שאלה 2 סעיפים ז', ח' (התייחסות שגויה למספור סעיפים קודמים).

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

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 $O(\cdot)$. לאורך סעיף זה n הוא משתנה ואינו קבוע, כל הפונקציות הן מהטבעיים לעצמם ($f, g: \mathbb{N} \to \mathbb{N}$) והאופרטור log הוא לפי בסיס 2.
הנחיה: יש להוכיח/ להפריך כל תת-סעיף בלא יותר מ-5 שורות. הפתרונות הם קצרים, ואינם דורשים מתמטיקה מתוחכמת. אם נקלעתם לתשובה מסורבלת וארוכה, כנראה שאתם לא בכיוון.

1. $16^{logn} = O(n^3)$
2. לכל $k \in \mathbb{N}$ קבוע ופונקציות $f_1, \dots, f_k$, $g_1, \dots, g_k$ כך שלכל $1 \le i \le k$ מתקיים $f_i = O(g_i)$, מתקיים כי

$$\sum_{i=1}^{k} f_i = O\left(\max_{1\le i\le k}\{g_i\}\right)$$

<u>תזכורת</u>: מקסימום של k פונקציות היא פונקציה המוגדרת על ידי

$$\max_{1\le i\le k}\{g_i\}(n) = \max_{1\le i\le k}\{g_i(n)\}$$

3. אם $f_1(n) = O\big(g_1(n)\big)$ וגם $f_2(n) = O\big(g_2(n)\big)$ אז $f_1(n) \cdot f_2(n) = O\big(g_1(n) \cdot g_2(n)\big)$.
4. אם $f_1(n) = O\big(g_1(n)\big)$ ו $f_2(n) = O\big(g_2(n)\big)$ אז $f_1 \circ f_2(n) = O\big(g_1 \circ g_2(n)\big)$.

<u>תזכורת</u>: $f \circ h(n) = f\big(h(n)\big)$

ב. יהיו $f, g: \mathbb{R}^+ \to \mathbb{R}^+$ פונקציות הפיכות. נסמן את הפונקציות ההופכיות שלהן f^{-1}, g^{-1}.
הוכיחו/ הפריכו:

$$f(x) = O(g(x)) \;\Rightarrow\; g^{-1}(x) = O(f^{-1}(x))$$

ג. תהא $a_1, \dots, a_n$ סדרה של מספרים אי-שליליים ונסמן $A = \max\{a_1, \dots, a_n\}$. כלומר, A הוא פונקציה של הסדרה $a_1, \dots, a_n$. שימו לב: בסעיפים 1,2 הסימון האסימפטוטי הוא $\Theta(\cdot)$ ולא $O(\cdot)$. (תזכורת: $f = \Theta(g) \leftrightarrow f = O(g)\ and\ f = \Omega(g)$).

1. הוכיחו את הטענה הבאה:
טענה: אם ישנם שני קבועים $0 < b, c \le 1$ כך שלפחות $n \cdot b$ מתוך איברי הסדרה $a_1, \dots, a_n$ הם בגודל של לפחות $c \cdot A$, אז מתקיים:

$$\sum_i a_i = \Theta(nA)$$

עבור סעיפים 2,3 יש חובה להשתמש בטענה שכתובה בסעיף 1. ניתן להשתמש בטענה זו גם ללא הוכחתה בסעיף 1.

2. הוכיחו כי מתקיים

$$\log n! = \Theta(n \log n)$$

תזכורת: כיוון אחד כבר ראינו בתרגול 5 ואין צורך לחזור על ההוכחה שלו.

3. בהינתן שלם חיובי k נגדיר את הפונקציה הבאה:

$$p_k(n) = \sum_{i=1}^{n} i^k$$

כאשר n הוא שלם חיובי. הוכיחו כי לכל קבוע k מתקיים כי

$$p_k(n) = \Theta(n^{k+1})$$

ד. לכל אחת משתי הפונקציות הבאות, נתחו את סיבוכיות זמן ריצתה במקרה הגרוע כתלות ב-n (אורך הרשימה L). הניחו כי פעולות אריתמטיות (כמו גם המתודות הנקראות מהספרייה math) ופעולות append רצות בזמן $O(1)$. 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(L):
    n = len(L)
    while n > 0:
        n = n // 2
        for i in range(n):
            if i in L:
                L.append(i)
    return L
```

2.

```
def f2(L):
    n = len(L)
    res = []
    for i in range(500, n):
        m = math.floor(math.log2(i))
        for j in range(m):
            k=1
            while k<n:
                k*=2
                res.append(k)
    return res
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

שאלה 2

בשאלה זה נעסוק בייצוג של float במחשב.

בשונה מגרסת ה-64 ביטים שראיתם בהרצאה ובתרגול, בשאלה זו נשתמש בייצוג של מחרוזת באורך 16 תווים בבסיס האוקטאלי (בסיס 8). כלומר, נייצג מספר ממשי על ידי המחרוזת $b_0 \dots b_{15}$, באשר $b_0 \in \{0,1\}$ ו- $b_i \in \{0,1,2,3,4,5,6,7\}$ לכל $1 \le i \le 15$. נפענח את המחרוזת באופן הבא:

- $sign = b_0$ - תו יחיד הקובע את סימן המספר כאשר הספרה 0 מייצגת סימן חיובי והספרה 1 מייצגת מספר שלילי. שימו לב שרק תו זה הוא לא בבסיס אוקטאלי, כלומר $b_0 \in \{0,1\}$.
- $exp = b_1 \dots b_3$ - מספר בייצוג אוקטאלי בן שלושה תווים (יתכנו אפסים מובילים), המוחשב על ידי:

$$exp = \sum_{i=1}^{3} b_i \cdot 8^{3-i}$$

כלומר מתקיים: $0 \le exp \le 8^3 - 1 = 511$.

- $fraction = b_4 \dots b_{15}$ - מספר בייצוג אוקטאלי בן 12 תווים המייצג את החלק השברי של המספר. ערך המספר מחושב על ידי:

$$fraction = \sum_{i=4}^{15} b_i \cdot 8^{3-i}$$

הנוסחה לחישוב המספר היא:

עבור מחרוזת שהיא כולה '0'-ים	$num = 0$
עבור שאר המחרוזות	$num = (-1)^{sign} \cdot 8^{exp-255} \cdot (1 + 7 \cdot fraction)$

דוגמה להמחשה:

sign	exp (3 chars)			fraction (12 chars)											
0	3	7	7	1	7	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

בדוגמה זו:

$$exp = (377)_8 = (3 \cdot 8^2 + 7 \cdot 8^1 + 7 \cdot 8^0)_{10} = (255)_{10}$$

$$fraction = 1 \cdot 8^{-1} + 7 \cdot 8^{-2} = \frac{1}{8} + \frac{7}{64} = \frac{15}{64}$$

ו- $sign = 0$. לכן המספר הממשי שמחרוזת זו מייצגת הוא:

$$num = +8^{255-255} \cdot \left(1 + 7 \cdot \frac{15}{64}\right) = 2.640625$$

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

השאלות בסעיפים א,ב,ג מתייחסות לשיטת הייצוג האוקטאלי שתוארה לעיל. יש לפרט בקצרה את חישוביכם בקובץ ה-pdf.

א. בהינתן $k \in \{-255, \dots, 255\}$, כמה מספרים שונים ניתנים לייצוג בקטע $[8^k, 8^{k+1})$?

ב. כמה מספרים שונים ניתנים לייצוג בקטע $[1,16)$?

ג. מהי מידת הדיוק בקטע $[32,64)$? כלומר, מהו ההפרש המינימלי בין שני מספרים שונים הניתנים לייצוג בקטע?

נתבונן במספרים שניתנים לייצוג מדוייק בשיטה הבינארית של 64 הביטים שראינו בכיתה, נסמן אותם ב-B.

ד. טענה: לא כל המספרים ב-B הם ניתנים לייצוג מדוייק בשיטה האוקטאלית המוצגת לעיל. הוכיחו זאת על ידי הבאת דוגמה למספר שניתן לייצוג (מדוייק) בשיטה הבינארית, אבל לא ניתן לייצוג (מדוייק) בשיטה האוקטאלית בת 16 הספרות המוצגת כאן. כיתבו את המספר המובא כדוגמה בייצוג הבינארי שלו.

ה. טענה: ישנם מספרים ב-B שהם כן ניתנים לייצוג מדוייק בשיטה האוקטאלית בת 16 הספרות המוצג כאן. הוכיחו את הטענה על ידי הבאת דוגמה למספר אחד כזה. כיתבו את המספר המובא כדוגמה בייצוג האוקטלי שלו.

סעיפים ו,ז,ח למימוש בקובץ ה-py.

ו. ממשו את הפונקציה oct_to_fraction(octal) אשר מקבלת מחרוזת אוקטאלית באורך 12 תווים. מחרוזת הקלט תייצג את הספרות 4,...,15 בייצוג האוקטאלי שהוגדר לעיל (התו השמאלי במחרוזת ייצג את התו באינדקס 4 בייצוג האוקטאלי). על הפונקציה לחשב את הערך fraction שיחושב ממחרוזת זו לפי נוסחת ההמרה, ומחזירה את המספר כ-float. מותר לקרוא לפונקציה int.
דוגמה הרצה:

```
>>> oct_to_fraction('1700000000000000')
0.234375
>>> oct_to_fraction('0770000000000000')
0.123046875
>>> oct_to_fraction('6210000000000000')
0.783203125
```

ז. ממשו את הפונקציה oct_to_float(octal) אשר מקבלת מחרוזת באורך 16 של תווים, באשר התו הראשון הוא מתוך $\{0,1\}$ ויתר התווים הם מתוך הספרות $\{0,1,2,3,4,5,6,7\}$. מחרוזת הקלט מתארת ייצוג אוקטאלי של מספר כמתואר לעיל. הפונקציה מחשבת את המספר שמיוצג לפי הנוסחה, ומחזירה אותו כ-float. יש לממש את הפונקציה בעזרת כתיב למבדא (lambda), כפי שהיא מופיעה בקובץ השלד (החליפו את None במימוש שלכם). מותר לקרוא לפונקציה int ולפונקציה מסעיף ו'.
דוגמת הרצה:

```
>>> oct_to_float('0400621000000000')
51.859375
```

ח. ממשו את הפונקציה is_greater_equal(oct1,oct2) המקבלת שתי מחרוזות באורך 16 של תווים מתוך הספרות $\{0,1,2,3,4,5,6,7\}$ המתארת ייצוג אוקטאלי של מספר כמתואר לעיל, ומחזירה True אם המספר שמייצג oct1 גדול או שווה למספר שמייצג oct2.
בסעיף זה **אסור** להמיר את המחרוזות למספרים, בפרט אסור לקרוא לפונקציות מסעיפים ו' ו-ז'.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

שאלה 3

שאלה זו תעסוק בחישוב של מספרים ממשיים באמצעות טכניקות שונות.

הערה: כזכור, חישובים אריתמטיים של אובייקטים מטיפוס float הם לא מדויקים מטבעם, אך בשאלה זו נתייחס לחישוב כאילו הוא מדויק. בפרט, ניתן להתעלם משגיאות הנובעות מחוסר דיוק של float.

א. בסעיף זה נממש פונקציה שמקרבת שורש של מספר ממשי חיובי.

ניעזר בעובדה הבאה: לכל $x \in \mathbb{R}^+$ מספר ממשי חיובי, ניתן לכתוב את x כסכום (אולי אינסופי) מהצורה

$$x = \frac{1}{a_1} + \frac{1}{a_1 \cdot a_2} + \frac{1}{a_1 \cdot a_2 \cdot a_3} + \cdots$$

כאשר לכל n טבעי, האיבר a_n הוא מספר טבעי המחושב על ידי החוקיות הבא:

1. תנאי התחלה: a_1 הוא המספר הטבעי המינימלי כך ש - $x \geq \frac{1}{a_1}$
2. כלל נסיגה: לכל $n \geq 2$ מתקיים כי $a_n \geq a_{n-1}$ וכן a_n הוא המספר הטבעי המינימלי עבורו מתקיים

$$x \geq \frac{1}{a_1} + \frac{1}{a_1 a_2} + \cdots + \frac{1}{a_1 a_2 \ldots a_n}$$

שימו לב: במקרים מסוימים הסדרה שתיארנו מסתיימת לאחר מספר סופי של איברים, וזה תקין לחלוטין (הבחנה מעניינת: הסדרה תהיה סופית אמ"ם המספר x הוא רציונלי).

דוגמה לחישוב הסדרה עבור $x = \sqrt{2}$ (מומלץ להמשיך לפתח עד לפחות $n = 4$ כדי לוודא שההגדרה ברורה לכם):

- $a_1 = 1$ - כי $\sqrt{2} \geq \frac{1}{1}$
- $a_2 = 3$ - כי $\sqrt{2} < \frac{1}{1} + \frac{1}{1 \cdot 2}$ אבל $\sqrt{2} \geq \frac{1}{1} + \frac{1}{1 \cdot 3}$

שימו לב שככל שאנו מתקדמים בסדרה, הסכום הולך ומתקרב ל-$\sqrt{2}$.

ממשו את הפונקציה approx_root(x,eps) המקבלת מספר ממשי חיובי x ומספר ממשי חיובי ε, ומוצאת קירוב-ε עבור $\sqrt{x}$. בפרט, הפונקציה תחזיר אובייקט מסוג tuple אשר יכיל את זוג האובייקטים הבאים:

1. רשימה המכילה את ערכי הסדרה החל מ-a_1, ועד a_n הראשון שמקיים

$$\sqrt{x} - \left(\frac{1}{a_1} + \frac{1}{a_1 a_2} + \cdots + \frac{1}{a_1 a_2 \ldots a_n}\right) < \varepsilon$$

2. הסכום המתאים לערכי a_i אלו, קרי הקירוב-ε שהתקבל ל-$\sqrt{x}$

הנחיה מחייבת: אסור להשתמש בפונקציה מובנית אשר מחשבת שורש של מספרים (בפרט, אין להעלות בחזקת 0.5 או להשתמש ב-math.sqrt).
שימו לב שסעיף זה איננו סעיף ברקורסיה, על אף שהשאלה נראית רקורסיבית באופייה.

דוגמת הרצה:

```
>>> approx_root(2, 0.1)
([1, 3], 1.3333333333333333)

>>> approx_root(2, 0.02)

([1, 3, 5], 1.4)
```

ב. בסעיף זה נראה שיטה לחישוב מקורב של הקבוע e.

נתאר את המשחק הרנדומי הבא: בכל סיבוב של המשחק נגריל מספר אקראי בתחום $[0,1]$. נפסיק לשחק כאשר סכום כל המספרים שהגרלנו מתחילת המשחק עבר את 1. באופן יותר פורמלי:

1. בסיבוב ה-k של המשחק נגריל את המספר r_k באופן אחיד מתוך התחום $[0,1]$.
2. אם $\sum_{i=1}^{k} r_i > 1$ – נפסיק את המשחק, אחרת נעבר לסיבוב הבא.

נסמן ב- n את מספר הסיבובים ששיחקנו, כלומר את הסיבוב הראשון שבו $\sum_{i=1}^{n} r_i > 1$.

עובדה: מסתבר שהערך <u>הממוצע</u> של n הוא המספר e (ניתן להוכיח זאת בכלים הסתברותיים).

i. ממשו את הפונקציה approx_e(N) אשר מסמלצת <u>את המשחק המתואר</u> N פעמים (שימו לב – כל משחק מכיל כמה סיבובים), ומחזירה קירוב מתאים למספר e. השתמשו בפונקציה random.random().
<u>הערה</u>: מומלץ לממש פונקציית עזר שמסמלצת משחק יחיד, ואז לקרוא לפונקציה זו N פעמים (בדומה לשאלת הרולטה בתרגיל 2).

ii. הריצו את הפונקציה על הערכים $N = 100, 1000, 10000$ **<u>וכתבו את התוצאות בטבלה בקובץ ה-pdf</u>**.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

שאלה 4

בשאלה זו הניחו כי פעולות אריתמטיות והשוואת מספרים מתבצעות בזמן קבוע. שני חלקי השאלה עוסקים בנושאים שונים.

חלק 1: רשימה כמעט ממויינת. רשימה L היא כמעט ממוינת אם כל איבר בה נמצא לכל היותר במרחק אינדקס אחד מהמיקום שלו ברשימה הממוינת. כלומר, אם $\text{arg\_}sort(i)$ הוא האינדקס של $L[i]$ ברשימה הממוינת sorted(L) אז

$$\text{arg\_}sort(i) \in \{i-1, i, i+1\}$$

לדוגמה, הרשימה [2, 1, 3, 5, 4, 7, 6, 8, 9] היא כמעט ממוינת.

א. נרצה לממש פעולת חיפוש ברשימה כמעט ממוינת.

i. השלימו את הפונקציה find(lst, s) בשלד, שמקבלת את רשימה כמעט ממוינת L ומספר שלם s ומחזירה את האינדקס i כך ש $L[i] == s$ אם s הוא איבר ברשימה L, אחרת מחזירה $None$.
למשל, עבור L מהדוגמה ו-$s = 5$, הפונקציה תחזיר 3 (כי המספר 5 נמצא באינדקס 3 ברשימה L).
עבור $s = 11$ הפונקציה תחזיר None (כי המספר 11 לא נמצא ברשימה L).

ii. מה היא סיבוכיות זמן הריצה? הסבירו בקצרה.

ב. נרצה למיין רשימה כמעט ממוינת במקום (in-place), ללא שימוש ברשימת עזר.

i. השלימו את הפונקציה sort_from_almost(lst) בשלד, שמקבלת רשימה כמעט ממוינת וממיינת אותה ללא שימוש ברשימת עזר (או כל מבנה בעל גודל יותר מ O(1)).

ii. הסבירו בקצרה את הפתרון שלכם ואת סיבוכיות זמן הריצה שלו.

חלק 2: חישוב חציון של רשימת ערכים. בהינתן רשימה של n ערכים מספריים טבעיים, חציון ערכי הרשימה $M \in \mathbb{N}$ הוא ערך המקיים את שני התנאים:

1. $|\{n \in L: n < M\}| \leq \frac{n}{2}$
2. $|\{n \in L: n > M\}| \leq \frac{n}{2}$

כלומר לכל היותר מחצית מערכי הרשימה גדולים מ-M וגם לכל היותר מחצית מערכי הרשימה קטנים מ-M.
הערה: חציון לפי הגדרה זו הוא לא בהכרח יחיד.
תהי L רשימה בת n ערכים טבעיים מהתחום $\{0, \dots, k-1\}$ עבור $k \in \mathbb{N}^+$.

ג. נרצה לחשב את החציון ללא גישה ישירה לרשימת הערכים L. לטובת החישוב, נקבל גישה לשתי פונקציות q_g ו-q_l (עבור ראשי התיבות של query-greater ו-query-lower בהתאמה). פונקציות אלה מכילות את הרשימה L ומקבלות כפרמטר מועמד m (מימוש הפונקציה שיוצרת את פונקציות q_g ו-q_l מובא בהמשך). בקבלת פרמטר m שלם, הפונקציה q_l מחזירה את הערך $|\{n \in L: n < m\}|$, והפונקציה q_g מחזירה את הערך $|\{n \in L: n > m\}|$. זמן הריצה של כל אחת מהפונקציות הוא $O(n)$.

להלן הקוד שמייצר את הפונקציות q_g ו-q_l:

```
def generate_queries(k = 100, n = 1000):
    L = []
    for i in range(n):
        L.append(random.randint(0, k-1))

    def q_g(m):
        size = 0
        for i in range(n):
            if L[i] > m: size +=1
        return size

    def q_l(m):
        size = 0
        for i in range(n):
            if L[i] < m: size +=1
        return size

    return q_l, q_g

k = 100000
n = 100
q_l, q_g = generate_queries(k, n)
```

הקוד לעיל מייצר את פונקציות השאילתות q_q, q_l . נשים לב: רשימה L מיוצרת באופן רנדומי, וערכיה שהם בתחום $\{0, \dots, k-1\}$ הם לא ממויינים כלל.

i. השלימו את הפונקציה compute_median(q_l, q_g, k, n) המקבלת את הפונקציות q_l, q_g ופרמטר k ומחזירה את החציון של ערכי הרשימה אשר נמצאת בפונקציות q_l, q_g.
ii. הסבירו בקצרה את הפתרון שלכם ואת סיבוכיות זמן הריצה שלו.

דוגמת הרצה:

```
>>> k = 100000
>>> n = 100
>>> q_l, q_g = generate_queries(k, n)
>>> M = compute_median(q_l, q_g, k, n)
>>> print("The value of the found median: {0}".format(M))
>>> count_g = q_g(M)
>>> count_l = q_l(M)
>>> print("num of elemnts > {0}: {1}".format(M, count_g))
>>> print("num of elemnts < {0}: {1}".format(M, count_l))
The value of the found median: 56249
num of elemnts > 56249: 50
num of elemnts < 56249: 50
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

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. ראינו 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: מחרוזות באורך k , עבור $k > 0$ נתון כלשהו, מעל האלפבית a, b, c, d, e שמכיל 5 תווים.

ההשוואה בין זוג מחרוזות תהיה לקסיקוגרפית, כלומר השוואה מילונית רגילה.

<u>הערות</u>:

1. בשאלה זו אסור להשתמש בפונקציות מיון מובנות של פייתון.
2. בניתוח הסיבוכיות בשאלה זו נניח שהשוואה של זוג מחרוזות באורך k מבצעת בפועל השוואה של התווים של המחרוזות משמאל לימין, ובמקרה הגרוע תהיה מסיבוכיות זמן $O(k)$.
3. לשם פשטות ניתוח הסיבוכיות נתייחס הן לפעולות אריתמטיות והן לפעולות העתקה של מספרים ממקום למקום בזכרון כפעולות שרצות בזמן קבוע.

א. השלימו בקובץ השלד את הפונקציה string_to_int(s) שמקבלת כקלט מחרוזת s באורך k <u>בדיוק</u> שמורכבת מהתווים a, b, c, d, e ומחזירה מספר שלם בין 0 ל $5^k - 1$ כולל, המייצג את הערך הלקסיקוגרפי היחסי של המחרוזת. על הפונקציה להיות חד-חד-ערכית. סיבוכיות הזמן שלה צריכה להיות $O(k)$.

ב. השלימו בקובץ השלד את הפונקציה int_to_string(k, n) , ההפוכה לזו מסעיף א', שמקבלת כקלט מספר שלם k גדול מ-0, וכן מספר שלם n בין 0 ל $5^k - 1$ כולל ומחזירה מחרוזת s באורך k <u>בדיוק</u> שמורכבת מהתווים a, b, c, d, e שערכה הלקסיקוגרפי הוא n. גם על פונקציה זו להיות חד-חד-ערכית. סיבוכיות הזמן שלה צריכה להיות $O(k)$.

שימו לב שפונקציה זו צריכה לקיים לכל: $0 \le i \le 5^k - 1$

string_to_int(int_to_string(k, i)) == i

דוגמת הרצה:

```
>>> for i in range(5**3):
        if string_to_int(int_to_string(3, i)) != i:
            print("Problem with ", i)
>>> alphabet = ["a","b","c","d","e"]
>>> lst = [x+y+z for x in alphabet for y in alphabet for z in
alphabet]
>>> for item in lst:
        if int_to_string(3, string_to_int(item)) != item:
            print("Problem with ", item)
>>> #Nothing was printed
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2022

בסעיפים הבאים נממש פונקציות מיון באמצעות ההמרה שהגדרנו זה עתה. נבחן שתי שיטות שונות לממש את המיון. השיטות יממשו את המיון תחת אילוצי זיכרון עזר שונים. שיטה ראשונה, תחת אילוץ זיכרון עזר המאפשר שימוש בזכרון גדול לה זמן ריצה קצר במיוחד. שיטה שניה, תחת אילוץ זיכרון עזר המאפשר שימוש בזכרון מינימלי ולה זמן ריצה ארוך במיוחד.

<u>דרישת מימוש</u>: השיטות ימומשו בפונקציות sort_strings1(lst, k), sort_strings2(lst, k). שתי הפונקציות מקבלות כקלט רשימה lst של n מחרוזות כמתואר ומספר חיובי k כך שכל מחרוזת ברשימה הינה באורך k בדיוק. על הפונקציות להחזיר רשימה <u>חדשה</u> ממויינת בסדר עולה (ולא לשנות את lst עצמה). דגש: רשימת הפלט היא לוקחת מקום בזיכרון (בגודל n) ולא נחשבת בחישוב האילוץ של זכרון העזר.

ג. השלימו בקובץ השלד את הפונקציה sort_strings1(lst, k) לפי דרישת המימוש, עם אילוץ זכרון העזר: על הפונקציה להשתמש ברשימת עזר בעלת 5^k איברים. על הפונקציה sort_strings1 להיות מסיבוכיות זמן $O(kn + 5^k)$.
הדרכה: עליכם להשתמש בפונקציות מסעיפים א', ב'.

ד. בקובץ ה pdf הסבירו מדוע הפונקציה מסעיף ג' עומדת בדרישות סיבוכיות הזמן.

ה. השלימו בקובץ השלד את הפונקציה sort_strings2(lst, k) לפי דרישת המימוש, עם אילוץ זכרון העזר: על הפונקציה להשתמש בזכרון עזר מגודל $O(k)$. בפרט, בסעיף זה אסור להשתמש ברשימת עזר כמו בסעיף הקודם. על הפונקציה להיות מסיבוכיות זמן $O(5^k \cdot kn)$.

ו. בקובץ ה pdf הסבירו מדוע הפונקציה מסעיף ה' עומדת בדרישות סיבוכיות הזמן והזיכרון.

חומר למחשבה (לא להגשה):

מבחינת זמן ריצה, וללא תלות בזכרון, מהו היחס בין n, k עבורו המימוש בסעיף ג' מנצח את selection-sort? ועבור quick-sort?