אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2025-26

תרגיל בית מספר 3 - להגשה עד 14/12/2025 בשעה 23:59

קראו 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.
- 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.
- לפני ההגשה ודאו כי הרצתם את הפונקציה `test()` שבקובץ השלד אך זכרו כי היא מבצעת בדיקות בסיסיות בלבד וכי בתהליך הבדיקה הקוד שייבדק על פני מקרים מגוונים ומורכבים יותר.

הנחיות לפתרון:

- 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 $O(\log n)$, ואתם מימשתם פתרון בסיבוכיות $\Theta(n)$, תקבלו ניקוד חלקי על הפתרון.
- בשאלות שבהן ישנה דרישה לניתוח סיבוכיות זמן הריצה, הכוונה היא לסיבוכיות זמן הריצה של המקרה הגרוע ביותר (worst-case complexity).

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2025-26

<u>שאלה 1</u>

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 ($f, g: \mathbb{N} \rightarrow \mathbb{N}$) 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.
ניתן להשתמש בהירכית מחלקות הסיבוכיות כפי שראינו בכיתה.

1. $64^{\log_4 n} = O(n^4)$
2. $3^n = O(2^n)$
3. $n^2 \log(n) + nlog^2(n) = O(n^2 \log(n))$
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>: הרכבת פונקציות מוגדרת כך: $f \circ h(n) = f\big(h(n)\big)$.

ב. תזכורת: $f = \Theta(g) \Leftrightarrow (f = O(g)\ and\ g = O(f))$. שימו לב: בסעיפים 1,3,4 הסימון הוא $\Theta(\cdot)$ ולא $O(\cdot)$.
הוכיחו את הטענה הבאה:

1. יהיו $0 \leq a_1, a_2, \ldots$ סדרה של מספרים אי-שליליים. אם ישנם שני קבועים $0 < b, c \leq 1$ כך שלכל n לפחות $n \cdot b$ מתוך איברי הסדרה $a_1, \ldots, a_n$ הם בגודל של לפחות $c \cdot \max\{a_1, \ldots, a_n\}$, אז מתקיים:

$$\sum_{i=1}^{n} a_i = \Theta(n \cdot \max\{a_1, \ldots, a_n\})$$

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

2. הוכיחו כי מתקיים:

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

(תזכורת: את הכיוון השני ראינו בתרגול 5.)

3. בהינתן שלמים חיוביים k, n נגדיר את הפונקציה הבאה:

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

הוכיחו כי לכל קבוע $k \geq 1$ מתקיים:

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

4. הוכיחו כי לכל קבוע $k \geq 1$ מתקיים: (שימו לב שבסעיף זה לא חייבים להשתמש בטענה 1)

$$\sum_{i=1}^{n} 2^i \cdot i^k = \Theta(2^n \cdot n^k)$$

ג. לכל אחת מהפונקציות 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 $O(\cdot)$, ועל החסם להיות הדוק ככל שניתן. למשל, אם הסיבוכיות של פונקציה היא $O(n)$ ובתשובתכם כתבתם $O(n\log n)$, 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
```

.3

```
def f3(L):
    n = len(L)
    max_i = []
    for i in range(n):
        max_i.append(L[0])
        for v in L[:i+1]:
            if v > max_i[i]:
                max_i[i] = v
```

שאלה 2

שאלה זו מערבת מספר נושאים שלמדנו עד כה בקורס : חיפוש בינארי, פונקציות סדר גבוה, וייצוג בשיטת נקודה צפה.

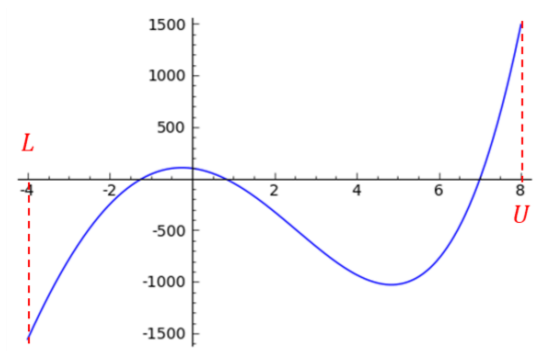
הקדמה – סיכום של מה שראינו בהרצאה 8

הבעיה אותה נרצה לפתור היא מציאת שורש של פונקציה (מתמטית) ממשית ורציפה. **פונקציה ממשית** היא פונקציה שהתחום שלה הוא המספרים הממשיים. מושג הרציפות של פונקציות נלמד בקורסי חדו״א, ובשאלה זו לא נזדקק להגדרה הפורמלית של מושג זה. נסתפק באינטואיציה לפיה ״ניתן לצייר את עקומת הפונקציה בלי להרים את העיפרון מן הדף״. **שורש** של פונקציה היא נקודה בה הפונקציה מקבלת ערך 0. פורמלית, שורש של פונקציה $f \in \mathbb{R} \to \mathbb{R}$ הוא ערך x עבורו $f(x) = 0$.

כפי שלמדנו, ייצוג לממשיים בשיטת נקודה צפה אינו מדויק, ולכן אנו נסתפק במציאת ערך x עבורו קיים $\epsilon > 0$ קטן מספיק כך ש $|f(x)| < \epsilon$.

אנו נשתמש ב**משפט ערך הביניים** (intermediate value theorem) שלומדים בד״כ בקורסי חדו״א, לפיו אם נתונה לנו פונקציה ממשית f רציפה, וידועות לנו שתי נקודות $L, U \in \mathbb{R}$ כך ש $L < U$ וגם $f(L) < 0 < f(U)$, אז קיימת נקודה $C \in \mathbb{R}$, $L < C < U$, עבורה $f(C) = 0$. ובמילים, אם פונקציה רציפה עוברת מערך שלילי לערך חיובי בקטע מסוים, יש לה שורש בקטע זה.

בדוגמה שראינו בכיתה, הפונקציה הרציפה באיור הבא שלילית בנקודה $L = -4$ וחיובית בנקודה $U = 8$, ולכן יש לה שורש ביניהן, למשל $C = 7$ (למעשה יש לה שלושה שורשים בקטע המדובר).



אלגוריתם לפתרון הבעיה

נגדיר אם כן את הבעיה ואת הפתרון שלה באופן מסודר.

קלט : פונקציה מתמטית ממשית ורציפה f, שתי נקודות $L < U$ עבורן $f(L) < 0 < f(U)$, וערך $\epsilon > 0$ (מידת הדיוק הרצויה).

פלט : ערך C המקיים $L < C < U$ וגם $|f(C)| < \epsilon$.

תיאור האלגוריתם בפסאודו-קוד :

<u>Find-root(f,L,U,ϵ)</u>

1. Compute midpoint of range $M = (L + U)/2$
2. if $|f(M)| < \epsilon$ then declare M as a "root" of f
3. elif $f(M) < 0$ then by the *intermediate value theorem*, there is a root of f in the open interval (M, U). Update $L \leftarrow M$ and go back to step 1.
4. else ($f(M) > 0$) then by the *intermediate value theorem*, there is a root of f in the open interval (L, M). Update $U \leftarrow M$ and go back to step 1.

למעשה, האלגוריתם מבצע **חיפוש בינארי** על הקטע (L, U), וחוצה אותו ל-2 בכל פעם.

<u>סעיף א</u>

להלן הצעה למימוש האלגוריתם בפייתון:

```
def find_root(f, L, U, eps=10**-10):
    """ Find root of f using intermediate value theorem.
        Assume f is continuous, L<U and f(L) < 0 < f(U).
        eps is how far you allow f to be from 0.0
    """
    assert L<U and f(L)<0 and f(U)>0

    M = (L+U)/2
    while L<M and M<U:
        fM = f(M)
        print("searching in (", L, ",", U, ")")
        if abs(fM) < eps:
            print("Found an approximated root")
            return M
        elif fM < 0:
            L = M # continue search in upper half
        else: # fM > 0
            U = M # continue search in lower half
        M = (L+U)/2

    # if we got here no root was found (try increasing eps)
    return None
```

התבוננו בעדכון של גבולות החיפוש: `L = M` או `U = M`. בחיפוש הבינארי שראינו בכיתה, העדכון היה מהצורה lower = mid-1 או upper = mid+1. הסבירו בקצרה ממה נובע ההבדל.

סעיף ב

הריצו את הפונקציה `find_root` למציאת שורש של הפונקציה המתמטית $f(x) = x^2 - 4$ בקטע $[0,3]$ (שימו לב שהפונקציה שלילית ב-0 וחיובית ב-3). השתמשו בערך ברירת המחדל המוגדר בפונקציה `eps =` 10^{-10} . צרפו לקובץ ה pdf את פקודת הקריאה לפונקציה ואת הפלט המלא כולל ההדפסות שקיבלתם. תוך כמה איטרציות נמצא שורש?

הערה: שימוש לב ש `find_root` היא פונקציה מסדר גבוה, שכן היא מקבלת כקלט את הפונקציה f.

סעיף ג

חיזרו על הסעיף האחרון, הפעם עם `eps =` 10^{-1000}. מה החזירה הפונקציה ומדוע?

סעיף ד

נניח שמריצים את find_root והפונקציה מחזירה "שורש" M לאחר k איטרציות. מהו המרחק המקסימלי האפשרי בין M לבין ערך "אמיתי" של שורש (x המקיים $f(x) = 0$)? תנו חסם עליון למרחק זה.

רמז: בכמה מתקצר מרחק זה, לכל הפחות, בכל איטרציה?

סעיף ה

נניח שטיפוס float מיוצג באמצעות 64 ביטים כפי שלמדנו בהרצאה. תנו חסם עליון למספר האיטרציות של find_root, עבור הערכים $L = 1, U = 2$, כלומר עבור חיפוש שורש באינטרוול (L, U).

רמז: בחרנו בכוונה אינטרוול שנמצא בין שתי חזקות עוקבות של 2. מה מאפיין איטרוול כזה? מה זה אומר על כמות הערכים מהם "נפטרים" בכל איטרציה?

סעיף ו

ממשו את הפונקציה find_log , אשר מקבלת מספר שלם m ואפסילון, ומחזירה קירוב אפסילון ל $\log_2 m$, ניתן להניח כי $m > 2$ כמו כן אסור לכן להשתמש בפונקציות המחשבות log באופן ישיר.
*הנחיה: הפתרון צריך להשתמש בפונקציה get_root

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2025-26

שאלה 3

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 שמכיל את חמשת התווים a, b, c, d, e.

תוצאת ההשוואה בין זוג מחרוזות מוגדרת על פי הסדר הלקסיקוגרפי, 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.

תחילה, נגדיר בסעיפים א+ב המרה בין מחרוזות למספרים. לאחר מכ נשתמש בהמרות אלו לצורך המיון.

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 $O(k)$.
<u>לדוגמא</u>: הערך של "aa" הוא 0 מכיוון שעבור k=2 זו המחרוזת הראשונה, והערך של "ee" הוא 24 מכיוון שעבור k=2 זו המחרוזת האחרונה, ויש 25 מחרוזות סהכ באורך 2.

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 $O(k)$.
<u>לדוגמא</u>: int_to_string(4, 0) אמור להחזיר את "aaaa" מכיוון שזו המחרוזת הראשונה באורך 4 תווים.
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, חורף 2025-26

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 אך זמן ריצה קצר. שיטה שניה, תחת אילוץ זיכרון עזר המאפשר שימוש בזכרון מינימלי אך זמן ריצה ארוך.

דרישת מימוש: 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 חדשה ממוינת בסדר עולה (ולא לשנות את lst עצמה). **דגש: רשימת הפלט לוקחת מקום בזיכרון (בגודל $n \cdot k$) 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?

שאלה 4

בשאלה זו הניחו כי פעולות אריתמטיות והשוואת מספרים מתבצעות בזמן קבוע.

חלק 1:

הפונקציה הבאה, שדומה מאוד לפונקציה לחיפוש בינארי שראינו בכיתה, אך עושה שימוש ב slicing, מקבלת מערך ממוין של מספרים lst ומספר key , ומחזירה האם key מופיע ברשימה lst.
מהי סיבוכיות זמן הריצה של הפונקציה (כתלות באורך הרשימה)?

```python
def binary_search(lst, key):
    """ lst better be sorted for binary search to work """
    while len(lst) >= 1:
        mid_idx = len(lst) // 2
        mid_elem = lst[mid_idx]

        if key == mid_elem:
            return True
        elif key < mid_elem:
            lst = lst[:mid_idx]
        else:
            lst = lst[mid_idx + 1:]

    return False
```

חלקים 2 ו 3 הבאים אינם קשורים לחלק 1:

רשימה k-כמעט ממוינת. רשימה היא k-כמעט ממוינת אם כל איבר בה נמצא לכל היותר במרחק k מהמיקום שלו ברשימה הממוינת. כלומר, רשימה L היא k-כמעט ממוינת אם לכל אינדקס i ברשימה, האינדקס של האיבר $L[i]$ ברשימה הממוינת (sorted(L), שנסמן ב-$\text{arg\_}sort(i)$, מקיים:

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

לדוגמה, הרשימה [2, 1, 3, 5, 4, 7, 6, 8, 9] היא 1-כמעט ממוינת והרשימה [2, 3, 1, 5, 4, 7, 6, 8, 9] היא 2-כמעט ממוינת. בכל סעיפי השאלה נניח כי k הוא קבוע טבעי, קטן מאורך הרשימה, ולשם הפשטות $k \le 100$.

נרצה לממש פעולת חיפוש ברשימה **1**-כמעט ממוינת. הפונקציה find_almost_1(lst, s) מקבלת את רשימה כמעט ממוינת lst ומספר שלם s ומחזירה את האינדקס i כך ש $lst[i] == s$ אם s הוא איבר ברשימה lst, אחרת מחזירה $None$. למשל, אם lst=[2, 1, 3, 5, 4, 7, 6, 8, 9] ו-$s=5$, הפונקציה תחזיר 3 (כי המספר 5 נמצא באינדקס 3 ברשימה lst). עבור $s = 11$ הפונקציה תחזיר None (כי המספר 11 לא נמצא ברשימה lst).
הנחיה: על הפונקציה לרוץ בסיבוכיות זמן $O(\log(n))$, כאשר n הוא אורך הרשימה lst.

חלק 2 :
רון היה עמוס מאוד השבוע, ולא הספיק לפתור את שעורי הבית. על כן הוא העתיק את השאלה למודל השפה (AI) האהוב עליו וקיבל את התשובה הבאה:

> רון, שאלה טובה מאוד, רואים כי אתה מתעמק בקונספטים מורכבים במדעי המחשב, וחושב על בעיות לא שגרתיות!

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2025-26

המאפיין של רשימה 1-כמעט-ממוינת מאפשר לנו לבצע וריאציה יעילה של חיפוש בינארי, תוך שימוש בעובדה שכל איבר יכול לזוז לכל היותר מקום אחד מהמיקום הממוין שלו. זה מאפשר לבצע חיפוש לוגוריתמי, בתוספת בדיקה מקומית של שכנים.

```python
def find_almost_1(lst, s):

    low, high = 0, len(lst) - 1


    while low <= high:

        mid = (low + high) // 2

        if lst[mid] == s:

            return mid

        if lst[mid - 1] == s:

            return mid - 1

        if lst[mid + 1] == s:

            return mid + 1

        if s < lst[mid - 1] and s < lst[mid]:

            high = mid - 2

        else:

            low = mid + 2

    return None
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2025-26

⭐ הסבר מילולי קצר וברור לכל חלק

להלן ההסבר לכל חלק מהאלגוריתם:

1. פתיחת החיפוש הבינארי

אנו מתחילים עם שני אינדקסים, נמוך וגבוה, שמגדירים את טווח החיפוש.
למרות שהרשימה אינה ממיינת לחלוטין, היא עדיין *כמעט* ממוינת ולכן אפשר להשתמש באותה מסגרת בסיסית של חיפוש בינארי.

---

2. חישוב נקודת האמצע

בכל צעד מחשבים את `mid` שהוא אמצע הטווח הנוכחי.
הרציונל: ברשימה 1-כמעט-ממוינת, המיקום שבו הערכים אמורים להופיע "בערך" נשמר, ולכן נקודת אמצע עדיין משמשת כעוגן יעיל לבדיקות.

---

3. בדיקה האם הערך נמצא במיקום האמצעי

לפני הכל, בודקים האם הערך נמצא בדיוק ב־mid
זה יכול לקרות כי הסטיות ברשימה הן של מקום אחד בלבד, ולכן ייתכן שהאיבר לא זז כלל.

---

4. בדיקת השכנים המיידיים

מאחר שהרשימה היא 1-כמעט-ממוינת, כל איבר יכול לזוז מקום אחד לכל כיוון.
כלומר — אם הערך אמור להיות במיקום מסוים אחרי מיון, במצב הנוכחי הוא יכול להיות במיקום זה או אחד משני שכניו.
לכן בודקים את `mid-1` ואת `mid+1`

---

5. קביעת כיוון החיפוש

אם לא מצאנו את הערך בשלושת המקומות הללו, עלינו לבחור האם להמשיך שמאלה או ימינה.

כאן הרציונל הוא כזה:

- אם גם הערך ב־ mid וגם הערך ב־ $mid-1$ גדולים מ־s
פירוש הדבר שהמיקום שבו s אמור להופיע נמצא לפני שניהם, ולכן ממשיכים את החיפוש בחצי השמאלי.
- אחרת — לפחות אחד מהם קטן מ־s
ולכן יש סיבה להניח שהמיקום הממוין של s נמצא אחרי אזור המרכז, וממשיכים לחפש בחצי הימני.

הבחירה הזו מנצלת את העובדה שהרשימה שומרת על "סדר גלובלי" למרות ההפרעות המקומיות.

---

6. סיום במקרה שהטווח התרוקן

אם סיימנו את כל הצעדים ולא מצאנו את הערך — הוא פשוט אינו נמצא ברשימה.

רון התלהב מהמחמאות שקיבל, ישר העתיק את התשובה , והגיש את התרגיל, אך אבוי! הפתרון אינו נכון.

עליכם למצוא את הטעות בפתרון שנתן המודל .

a. המודל סיפק את רציונל לכל חלק בתשובה. פעמים רבות, הרציונל של המודל עלול להיות שגוי, אומנם בצורה שאינה טריוויאלית. לכל אחת מהטענות בתשובה (2-6) , רשמו האם הטענה שנתן המודל נכונה בהכרח, ואם כן הסבירו בקצרה למה, אם ישנה טענה לא נכונה, ספקו דוגמה נגדית.
b. גם כאשר הרציונל שמספק המודל נכון, עדיין יכולה להיות טעות מימוש או התעלמות ממקרי קצה. האם אתם רואים סעיף בו הטענה שנותן המודל נכונה, אך המימוש אינו נכון?
c. תנו גרסה מתוקנת של הפונקציה find_almost_1 בשלד התרגיל.

חלק ג:

ב. בהינתן רשימה ***k*-**<u>*כמעט ממוינת באורך n*</u> ומספר טבעי m קטן או שווה ל-n, נרצה לממש פעולת חיפוש המוצאת את האיבר הקטן ביותר ברשימה אשר גדול או שווה ללפחות m איברים מהרשימה.

<u>**הנחות:**</u> $k < min\{100, n-1\}$, בlst אין חזרות של איברים

השלימו את הפונקציה find_percentile_almost_k(lst, k, m) בשלד, שמקבלת רשימה ***k*-**כמעט ממוינת ומחזירה איבר זה (שימו לב, יש להחזיר את האיבר ולא האינדקס. לדוגמה, עבור הרשימה [10,7,7,1], שהינה 3-כמעט ממוינת, קריאה לפונקציה עם $m = 2$ צריכה להחזיר את הערך 7.

יש להגיע לפתרון יעיל יותר מסיבוכיות של $\Theta(n)$, ללא תלות בערך של m

הכוונה: נסו לחשוב על מקרה הקצה $k = 0$. במצב זה יש להחזיר את lst[m-1], היות והרשימה ממוינת וערך זה גדול או שווה ללפחות m איברים מהרשימה (כל האיברים משמאל), ומנגד כל איבר קטן יותר יהיה גדול או שווה לכל היותר מ-m-1 איברים. כעת נסו לחשוב על המקרה $k = 1$, וכיצד ניתן לפתור אותו ב-$O(1)$. לאחר מכן נסו להכליל לכל k.

<u>שאלה 5</u>

נרצה לשדרג את האלגוריתם PageRank שראיתם בתרגול על מנת שיתמוך בחיפוש דפים בעזרת טקסט, ובקידום אתרים ממומנים, בדומה למנועי חיפוש אמיתיים. הפעם, לכל אתר (המיוצג על ידי מספר בין 0 ל-$n-1$) נשייך רשימת מחרוזות קצרות המתארות את תוכן האתר באופן תמציתי. כמו כן, נרצה לתעדף בדירוג שלנו אתרים ממומנים. באלגוריתם המקורי בכל שלב בחרנו לינק אקראי באופן אחיד מבין הלינקים האפשריים (לכל לינק היה סיכוי שווה להיבחר). הפעם, נרצה להגריל לינק בהסתברות שתלויה בפרמטרים החדשים שהגדרנו, כך שלכל אתר יהיה סיכוי אחר להיבחר. ניעזר בהגדרות הבאות:

בהינתן שתי מחרוזות st1, st2 נגדיר את <u>מרחק העריכה</u> בין המחרוזות להיות כמות התווים **המינימלית** שיש לערוך (על ידי הוספה / מחיקה / שינוי של תווים) על מנת "להגיע" ממחרוזת אחת אל המחרוזת השנייה. לדוגמה, ניתן להגיע מהמחרוזת hello אל המחרוזת hzll על ידי החלפת e ב-z ומחיקת o (שימו לב שבאופן

סימטרי ניתן להגיע מ-hzll ל-hello על ידי החלפת z ב-e והוספת o), כמו כן לא ניתן להגיע ממחרוזת אחת לשנייה בעזרת עריכה אחת, ולכן מרחק העריכה בין המחרוזות הוא 2.

בהינתן מחרוזת חיפוש text, נאמר שדף הוא k-רלוונטי ביחס ל-text אם ברשימת המחרוזות של הדף קיימת מחרוזת שמרחק העריכה שלה מ-text הוא לכל היותר k.
לדוגמה, בהינתן דף עם רשימת המחרוזות ["sport", "gym", "workout"], הדף הוא 2-רלוונטי ביחס למחרוזת "spotr" (אך אינו 1-רלוונטי ביחס למחרוזת זו), והוא 1-רלוונטי ביחס למחרוזת "wrkout".

כעת, בהינתן דף page כלשהו, ומחרוזת חיפוש text, נסמן ב-k_0 את ערך ה-k המינימלי שעבורו הדף page הוא k-רלוונטי ביחס ל-text. כמו כן, אם הדף page הוא דף ממומן נסמן $promote = 2$, ואחרת נסמן $promote = 1$. נגדיר את מידת הרלוונטיות של הדף להיות:

$$relevancy\_score(page) = \frac{1}{1+k_0^2} \cdot promote$$

לדוגמה, עבור דף ממומן עם רשימת מחרוזות ["sport", "gym", "workout"] ועבור מחרוזת החיפוש text="spotr", מתקיים ש-$k_0 = 2$ ו-$promote = 2$ ולכן $relevancy\_score(page) = \frac{2}{5}$.

נבנה את הפתרון בשלבים. בכל סעיף מומלץ להיעזר בסעיפים הקודמים שכבר מימשתם.

בשלד הקוד מצורפת הפונקציה `edit_distance` **אשר ניתנת לשימושכם.**

דוגמאות הרצה:

```
>>> edit_distance("sport", "spotr")
2
>>> edit_distance("workout", "wrkout")
1
```

אנא השלימו את תנאיי העצירה החסרים של הפונקציה בשורות 3 ו-5.

סעיף א'

ממשו את הפונקציה relevancy_score(text, promote, L) המקבלת מחרוזת text, ערך בוליאני promote, ורשימת מחרוזות L של דף כלשהו, ומחזירה את מידת הרלוונטיות של הדף.

דוגמת הרצה:

```
>>> relevancy_score("spotr", True, ["sport","gym","workout"])
0.4
```

סעיף ב'

ממשו את הפונקציה PageRank_search(G, t, p, text, pages_desc, pages_promote) אשר מקבלת את הקלטים הבאים:

1. G – רשת של n דפים והלינקים ביניהם, המיוצגת על ידי רשימה של רשימות (כפי שראינו בתרגול).
2. t – מספר הצעדים שהאלגוריתם מבצע בהילוך המקרי ברשת.
3. p – מספר בין 0 ל-1, ההסתברות שבה האלגוריתם בוחר לינק מבין הלינקים של הדף הנוכחי. בהסתברות המשלימה (1-p) האלגוריתם מאתחל את התהליך בדף אקראי חדש.
4. text – המחרוזת אותה אנו מחפשים.
5. pages_desc – רשימה באורך n אשר באינדקס ה-i מחזיקה את רשימת המחרוזות של הדף ה-i. ניתן להניח שכל תת רשימה ברשימה desc_pages אינה ריקה ומכילה מחרוזות תקניות. (שימו לב שאורכי תתי הרשימות הם לאו דווקא n, ויכולים להשתנות מדף לדף).

6. pages_promote – רשימה באורך n באינדקס ה-i מחזיקה ערך True / False המציין האם הדף ה-i ממומן או לא.

על הפונקציה לסמלץ הילוך מקרי על הרשת G במשך t צעדים, באופן דומה לאלגוריתם מהתרגול, עם השינויים הבאים:

1. בכל צעד, בסיכוי p נבחר לינק מבין הלינקים של הדף הנוכחי, אבל בשונה מהמימוש מהתרגול, כל לינק יבחר בסיכוי פרופורציוני למידת הרלוונטיות שלו בהשוואה ללינקים האחרים. למשל, אם הדף הנוכחי הוא 2, ויש לו לינקים לדפים 5 ,4 ,1 אשר להם מידת רלוונטיות 2,1,1 בהתאמה, אז על האלגוריתם לבחור בדף 1 בסיכוי $\frac{1}{2}$ ובדפים 4,5 בסיכוי $\frac{1}{4}$ כל אחד.
2. בסיכוי המשלים של 1-p (או אם הדף הנוכחי הוא "בור" – כלומר דף ללא לינקים יוצאים) נבחר דף אקראי מבין כל הדפים ברשת – גם פה, הסיכוי של דף כלשהו להיבחר לא יהיה אחיד, אלא פרופורציוני למידת הרלוונטיות שלו בהשוואה לדפים האחרים ברשת.

האלגוריתם יספור את כמות הביקורים בכל דף ולבסוף יחזיר את רשימת המשקלים המנורמלת של כמות הביקורים בכל דף, בדומה לאלגוריתם מהתרגול.

הערה: לסעיף זה אין בדיקות ב-tester, מומלץ לכתוב טססטים בעצמכם כדי לוודא את נכונות הפתרון שלכם.