אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

<u>תרגיל בית מספר 3 - להגשה עד 11/07/2024 בשעה 23:59</u>

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

<u>הנחיות לצורת ההגשה</u>:

- 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.

<u>הנחיות לפתרון</u>:

- 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, אביב 2023-4

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.
ניתן להשתמש בהיררכית מחלקות הסיבוכיות כפי שראינו בכיתה.

1. $64^{\log_4 n} = O(n^4)$
2. $\log(n) = O(\log(\log n^4))$
3. 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: $f \circ h(n) = f\big(h(n)\big)$.
4. אם $f = O(6^n)$ ו-$g = 6^{O(n)}$ אז בהכרח $g = O(f)$.

   <u>הערה</u>: $g = 6^{O(n)}$ היא דרך מקוצרת לומר שקיימת פונקציה $h = O(n)$ כך ש-$g = 6^h$.

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

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

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 $k \ge 1$ מתקיים:

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

4. הוכיחו כי לכל קבוע $k \ge 1$ 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

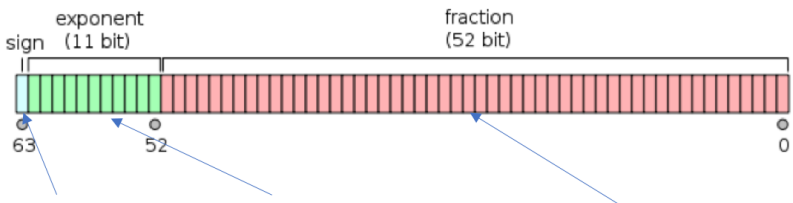
```
def f3(L):
    n = len(L)
    max_i = []
    for i in range(n):
        max_i.append(L[0])
        for v in L[:i+1]:
            if v > max_i[i]:
                max_i[i] = v
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

שאלה 2

תזכורת מההרצאה לייצוג float בפורמט IEEE 754:

($fraction = \sum_{i=1}^{52} b_i \cdot \frac{1}{2^i}$, כאשר b_1 הוא הביט הכי שמאלי בחלק הוורוד ו-b_{52} הכי ימני)



$$(-1)^{sign} \cdot 2^{exponent-1023} \cdot (1 + fraction)$$

א. המירו את הייצוגים הבאים ל float בפורמט IEEE 754:

a. 0 10000000000 11000…0

b. 1 10000000010 10000…0

ב. בשאלה זו נניח ש- $sign = 0$, וש-n הוא מספר שלם: $-1023 \le n \le 1024$.

a. הראו שבתחום $[2^n, 2^{n+1})$ בין כל שני מספרים סמוכים הניתנים לייצוג יש את אותו הפרש, וחשבו אותו כפונקציה של n.

<u>הערה</u>: בעת הדפסת מספר מטיפוס float הפקודה print ממירה אותו לבסיס עשרוני, ומבצעת לעיתים פעולות עיגול למספר העשרוני הקרוב ביותר. בשאלה זו אנחנו מתעלמים מכך ומתייחסים לערך המקורי שמיוצג בזיכרון.

נקרא להפרש מסעיף a "הרווח בתחום $[2^n, 2^{n+1})$".

b. פי כמה גדול הרווח בתחום $[2^{n+1}, 2^{n+2})$ מהרווח בתחום הסמוך $[2^n, 2^{n+1})$ (בסעיף זה $-1023 \le n < 1024$)?

c. איך סעיפים a ו- b היו משתנים אם היינו מקצים ביט נוסף ל-$fraction$?

d. הסבירו מדוע יש הבדל בין שתי תוצאות החישוב האחרונות בבדיקה הבאה (שימו לב לספרת האחדות השונה בתוצאה). בפרט, התייחסו לרווח בתחום הרלוונטי:

```
>>> 2 ** 53
9007199254740992
>>> 2 ** 53 + 1
9007199254740993
>>> 2.0 ** 53 + 1
9007199254740992.0
```

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

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 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 $O(k)$.
<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
```

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.

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?

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

שאלה 4

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

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:

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

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.

א. נרצה לממש פעולת חיפוש ברשימה **1**-כמעט ממוינת.

a. השלימו את הפונקציה find_almost_1(lst, s) בשלד, שמקבלת 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.

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

a. השלימו את הפונקציה sort_almost_k(lst, k) בשלד, שמקבלת רשימה **k**-כמעט ממוינת וממיינת אותה ללא שימוש ברשימת עזר (כלומר, לפונקציה מותר להשתמש בתאי הזכרון של הרשימה עצמה מהקלט, וב-$O(1)$ תאי זכרון נוספים לכל היותר). הפונקציה תמיד תחזיר None.

b. הסבירו בקצרה את הפתרון שלכם ואת סיבוכיות זמן הריצה שלו. לצורך ניתוח סיבוכיות הזמן הניחו כי k קבוע, כלומר מצאו חסם שתלוי ב-n בלבד.

ג. בהינתן רשימה **k**-כמעט ממוינת באורך n 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.

a. 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. אם כמה מהם שווים יש להחזיר את אחד מהם). 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 (ובהנחה ש-k קבוע).

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

שאלה 5

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 <u>k-רלוונטי ביחס ל-text</u> 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 <u>מידת הרלוונטיות של הדף</u> 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.

<u>סעיף א' (סעיף זה בנושא רקורסיה – ניתן לפתור אחרי תרגול 6)</u>

"מרחק עריכה" בין שתי מחרוזות מוגדר להיות כמות הפעולות הנדרשות על מנת לקבל מחרוזת אחת מהאחרת, כאשר פעולה אחת נחשבת שינוי או מחיקה של אות. נתאר את הפונקציה הרקורסיבית הבאה אשר מקבלת שתי מחרוזות ומחזירה את מרחק העריכה ביניהן.

```
def edit_distance(st1, st2):
    if len(st1) == 0:
        _____________
    if len(st2) == 0:
        _____________
    if st1[0] == st2[0]:
        return edit_distance(st1[1:], st2[1:])
    op1 = edit_distance(st1[1:], st2)
    op2 = edit_distance(st1, st2[1:])
    op3 = edit_distance(st1[1:], st2[1:])
    return min(op1, op2, op3) + 1
```

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 ב'

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 ג'

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 טסטים בעצמכם כדי לוודא את נכונות הפתרון שלכם.