<u>תרגיל בית 5</u>

מגיש: תומר מילדוורט | 316081355

<u>1.</u>

<u>ב.</u>

סיבוכיות הזמן היא $O(n)$. הפונקציה תרוץ על כל צומת בעץ פעם אחת בלבד. בנוסף, בכל צומת תבוצע פעולת החזרה (כמובן ב-$O(1)$) או קריאה רקורסיבית נוספת. לכן בסה"כ סיבוכיות הזמן היא $O(n \cdot 1) = O(n)$.

<u>2.</u>

<u>ג.</u>

נגדיר פונקציה המחזירה *True* אם נמצא מעגל ו-*False* כאשר לא קיים מעגל.

נגדיר 2 משתנים חדשים: slow, fast אשר יוגדרו באופן הבא:

slow = fast = self

נפתח לולאת while באופן הבא: *while True:*. תנאי זה ידאג כי הלולאה תמשיך לרוץ כל עוד לא הוחזר שום ערך מהפונקציה.

ראשית, נרצה כי slow ינוע בכל פעם צומת אחת לפי הכלל הבא:

אם קיים next2 שאינו none, נגדיר את slow להיות next2. אם אין next2 כזה, נגדיר את slow להיות next1.

נגדיר כי fast נע בכל פעם 2 צמתים לפי הכלל הבא:

אם בצומת יש next2 שאינו none:

אם אמת, בדוק האם בצומת עליה מצביע next2 (הצומת הבא למעשה) קיים גם next2 שאינו none:

אם אמת, נגדיר fast להיות next2.next2.

אם שקר, בדוק האם בצומת זו קיים next1 שאינו none.

אם אמת, נגדיר את fast להיות next2.next1.

אם שקר, נחזיר *False*.

אם שקר, בדוק האם בצומת עליה מצביע next1 קיים next2 שאינו none:

אם אמת, נגדיר את fast להיות next1.next2

אם שקר, נבדוק האם בצומת זו קיים next1 שאינו none:

אם אמת, נגדיר את fast להיות next1.next1

אם שקר, נחזיר *False*.

כעת, נבדוק האם fast is slow:

אם אמת, נחזיר *True*.

לא נגדיר תנאי למצב בו התנאי הוא שקר, אלא נרצה שהלולאה תמשיך לרוץ.

למעשה, אם fast יפגוש את slow נווכח שישנו מעגל. לחילופין, אם נעמוד בתנאים המתאימים, נראה כי אין מעגל.

3.

א.

נוכיח כי:

$$\forall a,b,c \in \mathbb{N}.\ a^c\ mod\ b = (a\ mod\ b)^c$$

מכיוון ש-$a \in \mathbb{N}$, יהיו $x,n \in \mathbb{N}.\ b > n$ כך ש: $a = xb + r$. נאמר כי $n = a - xb$, $x = \left\lfloor \frac{a}{b} \right\rfloor$. מכאן $x\ mod\ b = 0$. ובנוסף מתקיים $a\ mod\ b = n$.

נשתמש בבינום של ניוטון על הביטוי $a^c\ mod\ b = (xb+n)^c\ mod\ b$.

$$(xb+n)^c\ mod\ b = \left(\sum_{i=0}^{c} \frac{c!}{i!\,(c-1)!}\left((xb)^i n^{c-i}\right)\right) mod\ b = \left(\frac{c!}{0!\,(c-0)!}(xb)^0 n^c\right)\ mod\ b$$
$$= \boldsymbol{n^c\ mod\ b}$$

נשים לב כי כל מספר b' שהוא למעשה כפולה של b יקיים $b'\ modb = 0$.

■

ב.

נחשב את $g^{ab}\ mod\ p$. ידוע כי $g^{a'}\ mod\ p = g^a\ mod\ p$ בהינתן $a', g, p, g^a\ mod\ p, g^b\ mod\ p$.

$$(g^b\ mod\ p)^{a'}\ mod\ p = g^{ba'}\ mod\ p = g^{a'b}\ mod\ p = \left(g^{a'}\ mod\ p\right)^b\ mod\ p$$
$$= (g^a\ mod\ p)^b\ mod\ p = \boldsymbol{g^{ab}\ mod\ p}$$

■

5.

ב.

ננתח את סיבוכיות הזמן של המקרה הגרוע. ראשית, המקרה הגרוע יתקבל כאשר נשווה את כלל המחרוזות אחת לשנייה ונמצא התאמה בכל את מההשוואות, שכן אז יתבצעו $O(k)$ פעולות לכל קריאת השוואה. נתחיל לנתח מתחילת הפונקציה.

ראשית, מתבצעת לולאת for באורך n איטרציות שמבצעת פעולת slicing בעלות קבועה של $O(k)$. לאחר מכן, נפתחת לולאה פנימית נוספת באורך n איטרציות גם היא. הלולאה משווה את ערכי i, j בעלות של $O(1)$, ולאחר מכן יוצרת משתנה חדש מהרשימה, באורך k, ולכן לוקחת $O(k)$ בעצמה. כעת מתבצעת השוואה בין שתי תתי-מחרוזות באורך k בסיבוכיות של $O(k)$ וזאת בהנחה שההשוואה בוצעה במלואה כפי שטענו קודם לכן. לבסוף מתבצעות 2 פעולות: יצירת המשתנה $couple$ והוספתו לרשימה $result$, שתי פעולות בסיבוכיות של $O(1)$ כל אחת.

בהתעלם מהפעולות הזניחות שבסיבוכיות של $O(1)$, בסך הכל אנו מבצעים $n \cdot (k + n \cdot (k + k))$ פעולות בשתי הלולאות יחדיו, ומכאן:

$$n \cdot \left(k + n \cdot (k + k)\right) = nk + n^2 2k \leq 3n^2 k$$

לכן קיימים:

$$n_0 = 1, k_0 = 1, c = 3$$

כך ש:

$$\forall k > k_0, n > n_0. nk + n^2 2k \leq cn^2 k$$

ומכאן סיבוכיות הזמן היא:

$$O(n^2 k)$$

ה.

הסיבוכיות בזמן ממוצע של האלגוריתם היא $O(nk)$. ראשית ניצור מילון מסוג $Dict$ באורך אורך הרשימה הנתונה, פעולה בסיבוכיות זמן $O(n) = O(m)$. נשים לב כי כמות הפריטים שנרצה להכניס לטבלת ה-hash **בממוצע** היא כאורך הרשימה lst המוגדרת כקלט לפונקציה ($\alpha = \frac{O(n)}{O(m)} = 1$).

לולאה ראשונה:

מתבצעות n איטרציות על פעולות בסיבוכיות של $O(k)$ ו-$O(1)$ (slicing ו-insert [המכיל פעולת hash ו-append], בהתאמה).

לולאה שנייה:

שוב מתבצעות n איטרציות על פעולה באורך קבוע $O(k)$. לאחר מכן נפתחת לולאה נוספת שעוברת על הרשימה המתקבלת מהמתודה $find$, מתודה הפועלת בזמן **ממוצע** של $O(1)$ כפי שהראנו קודם כתלות בגודל טבלת ה-hash. בנוסף מתבצעת השוואת integers בסיבוכיות של $O(1)$.

לכן, בסה"כ סיבוכיות הזמן של האלגוריתם בהתעלם מהפעולות הזניחות:

$$(n \cdot k) + (n \cdot k) = 2nk$$

לכן קיימים

$$k_0 = 1, n_0 = 1, c = 2$$

כך ש:

$$\forall k > k_0, n > n_0. 2nk \leq cnk$$

ולכן סיבוכיות הזמן היא

$$O(nk)$$

ו.

לאחר הרצת שלושת הפתרונות, גילינו כי פתרון ג' (סעיף ו') הוא המהיר ביותר, אחריו בזמן כפול נמצא המימוש השני (סעיף ד'). במקום השלישי, בזמן ארוך פי 5 בערך מפתרון ג', נמצא פתרון א' (סעיף א'). הפער בין פתרון ב' ל-א' היה יחסית זניח. מכאן ניתן להסיק כי השימוש בפונקציות המובנות של פייתון (שימוש ב-dict של פייתון) יעילות יותר באופן משמעותי מאשר שימוש במחלקות הנוצרות באופן ייחודי לשימושים מסוג לו נדרשנו בשאלה. מנגד, הפער בין פתרון ב' (בו השתמשנו במחלקה $Class$) ל-א' (בו התבססנו על $List$ של פייתון) מעיד על כך שעל אף שמבני הנתונים המובנים והמתודות הנלוות אליהן יעילים מאוד, ניתן ע"י לייעל את זמן הריצה ע"י שימוש במחלקה הנכתבת במיוחד לביצוע פעולות מסוימות לפי דרישה.