מבוא מורחב למדעי המחשב - תרגיל בית 5

בר צדוק - 211964515, רועי בוגין - 209729524

שאלה 2:

ב.

b. הפונקציה קוראת לפונקציית עזר רקורסיבית. בפונקציית העזר, כל הפעולות שאינן הקריאה לפונקציה הרקורסיבית הן בסיבוכיות $O\left(1\right)$. הפונקציה הרקורסיבית תיקרא על כל צומת בעץ פעם אחת (כל צומת תקרא לפונקציה על בניה ואין מעגל בעץ) ולכן סיבוכיות זמן הריצה של הפונקציה היא $O\left(n\right)$ כאשר n הוא מספר הצמתים בעץ.

ג.

האלגוריתם יעבוד בדומה לשיטת הצב והארנב שלמדנו בתרגול. נעבור על איברי הרשימה עם ארבעה מצביעים. שני מצביעים l_1, r_1 יתקדמו בכל איטרציה Node אחד, אם next2 קיים, l_1 יעבור ל-next1 ו-r_1 יעבור ל-next2. בנוסף, שני המצביעים l_2, r_2 יתקדמו בכל איטרציה שני Node-ים, אם next2 קיים, l_1 יעבור ל-next1 ו-r_1 יעבור ל-next2. כמובן שאם next2 הוא None עבור Node ספציפי, כל המצביעים יעברו ל-next1.

אם באחת מהאיטרציות r_1 ו-r_2 מצביעים על אותו Node או l_1 ו-l_2 מצביעים על אותו Node הפונקציה תחזיר True (l_2 מתקדם מהר יותר מ-l_1 ועובר על אותם Node-ים ולכן הם ייפגשו רק אם יש מעגל, באופן דומה יקרה עם r_2 ו-r_1). אם l_2 וגם r_2 מגיעים לערך None, הפונקציה תחזיר False (שני המצביעים עוברים על כל מסלול אפשרי ולכן במקרה זה בהכרח אין מעגל). נשים לב שבהכרח אחד מהם יקרה ולכן האלגוריתם תמיד יחזיר ערך.

שאלה 3:

א.

נסמן $a = x + by$ (ולכן $x = a \mod b$) עבור $x, y \in \mathbb{N}$ כלשהם. נראה כי

$$a^c \mod b = (a + by)^c \mod b = \sum_{i=0}^{c} \frac{c!}{i!\left(c-i\right)!} x^i \left(by\right)^{c-i} \mod b$$

נשים לב שאם $c \neq i$ אז $\frac{c!}{i!(c-i)!} x^i \left(by\right)^{c-i}$ מתחלק ב-b (b מועלה בחזקה חיובית וידוע שלכל $i < c$ מתקיים $\frac{c!}{i!(c-i)!}$ מספר שלם ולכן לא מתקזז עם b). מכאן נובע ש-

$$a^c \mod b = 0 + \frac{c!}{c!\left(c-c\right)!} x^c \left(by\right)^{c-c} \mod b = x^c \mod b = (a \mod b)^c \mod b$$

ב.

על פי הסעיף הקודם, נשים לב שמתקיים

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

וכך ניתן לחשב את הדרוש.

שאלה 4:

א.

נשים לב שסיבוכיות זמן הריצה של הכפלת שני מספרים באורך היא x ו-y ביטים בהתאמה היא $O\left(xy\right)$. נראה כי כל ביט במספר הראשון מוכפל בכל ביט במספר השני ולאחר מכן מתבצעות במקרה הגרוע ביותר $O\left(xy\right)$ סכימות של שני ביטים. מכאן נובע שהסיבוכיות הכוללת של הפעולה היא $O\left(xy+xy\right)=O\left(xy\right)$.

נראה כי אנו מבצעים את הלולאה m פעמים (בכל איטרציה כמות הביטים של b קטנה ב-1), ובכל פעם מבצעים 2 פעולות כפל מסיבוכיות שאינה זניחה. הכפלה של $a*a$ היא בסיבוכיות של n^2 בפעם הראשונה, וכל פעם מגדילה את מספר הביטים בa פי 2. כך שבהרצה הk, כמות הביטים של a תהיה 2^kn ביטים ולכן הסיבוכיות הכפל תהיה $\left(2^kn\right)^2=4^kn^2$.

נראה כי הresult כל פעם גדל במספר הביטים של a. כלומר, אם בהרצה הk מספר הביטים בresult הוא b, לאחר מכן גודל הresult יהיה $b+2^kn$. כיוון שהresult מתחיל מ1, נראה כי בהרצה הk, גודל הresult יהיה $\left(2^k-1\right)n$ $\sum_{i=0}^{k-1}2^in=n\frac{2^k-1}{1}=$. ולכן הסיבוכיות של הכפלת $a*result$ תהיה $\left(2^k-1\right)2^kn^2=4^kn^2-2^kn^2$. כעת, נוכל לחשב את הסיבוכיות הכוללת:

$$\sum_{i=0}^{m}4^kn^2+4^kn^2-2^kn^2=n^2\left(\sum_{i=0}^{m}4^k+\sum_{i=0}^{m}4^k-\sum_{i=0}^{m}2^k\right)=n^2\left(2\cdot\frac{4^{m+1}-1}{3}-2^{m+1}+1\right)=O\left(4^mn^2\right)$$

שאלה 5:

ב.

לולאת ה-for החיצונית עוברת על n המחרוזות במערך ובכל איטרציה, לולאת ה-for הפנימית עוברת גם היא על n המחרוזות. בכל איטרציה של הלולאה הפנימית (חוץ מאחת שבה שתי המחרוזות זהות) והלולאה מבצעת slice באורך k לכל אחת מהמחרוזות בסיבוכיות $O\left(2k\right)=O\left(k\right)$ ולאחר מכן משווה בין שתי המחרוזות בסיבוכיות $O\left(k\right)$ ואם הן מתאימות היא מוסיפה אותן לרשימה בסיבוכיות $O\left(1\right)$. לכן, סיבוכיות זמן הריצה של הפונקציה במקרה הגרוע ביותר היא $O\left(n\cdot\left(n-1\right)\cdot\left(k+k\right)\right)=O\left(n^2k\right)$.

המקרה הגרוע ביותר מתקבל כאשר הרישא של כל המחרוזות זהה לסיפא של כולן (עד כדי תו אחרון), במקרה זה הפונקציה תשווה בין כל k התווים של כל זוג מחרוזות (לפני שתוסיף אותן לרשימה או שיתגלה שהן לא זהות).

ה.

יצירת ה-Dict נעשית בסיבוכיות זמן $O\left(n\right)$. לאחר מכן הלולאה הראשונה עוברת על כל איברי הרשימה, לכל אחד מהם היא יוצרת substring של המחרוזת בסיבוכיות $O\left(k\right)$, ועבור פעולת ה-insert הפונקצה מחשבת את ה-hash בסיבוכיות $O\left(k\right)$. כלומר, הלולאה הראשונה רצה בסיבוכית $O\left(nk\right)$.

לאחר מכן, הלולאה השנייה עוברת על כל איברי הרשימה, לכל אחד מהם היא יוצרת substring של המחרוזת בסיבוכיות $O\left(k\right)$ ומריצה את הפונקצייה find. פעולת ה-hash מתבצעת בסיבוכיות $O\left(k\right)$ ובממוצע יהיו $\frac{\ln(n)}{\ln\ln(n)}$ התנגשויות hash שפונקציית find תחפש בהן את ה-key המתאים. השוואת ה-keys תיעשה בסיבוכיות $O\left(k\right)$ עבור כל מחרוזת ב-Dict עם אותו ערך hash ותחזיר לבסוף רשימה ריקה. לכן, פונקציית find תרוץ בסיבוכיות $O\left(\frac{\ln(n)}{\ln\ln(n)}\cdot k\right) = O\left(k+\frac{\ln(n)}{\ln\ln(n)}\cdot k\right)$. מכאן נובע כי הלולאה השנייה רצה בסיבוכיות $O\left(nk\frac{\ln(n)}{\ln\ln(n)}\right) = O\left(n\left(k+k\frac{\ln(n)}{\ln\ln(n)}\right)\right)$ ולכן סיבוכיות הזמן הכוללת של הפונקציה היא $O\left(nk\frac{\ln(n)}{\ln\ln(n)}\right) = O\left(n+nk+nk\frac{\ln(n)}{\ln\ln(n)}\right)$.

ז.

לאחר הרצה של הקוד עבור רשימה של 1000, 2000 ו4000 מחרוזות באורך של 1000 תווים, ומדידת ממוצע של 100 הרצות עבור כל אחת משלושת הפונקציות, והגענו למסקנות הבאות:

ההרצה של prefix_suffix_overlap לקחה זמן הכי ארוך באופן משמעותי, פי 50 מההרצה של $hash1$ ופי 100 מההרצה של $hash2$, עבור הרצה על 1000 מחרוזות, ומשמעותית יותר מזאת עבור מספר גדול יותר של מחרוזות. פונקציה זו לא כללה שום ממואיזציה, והסיבוכיות שלה היא הגדולה ביותר מבין השלושה, כך שתוצאה זו אינה מפתיעה.

ההרצה של prefix_suffix_overlap_hash1 היתה מהירה בהרבה מהזמן שלקח לקודמת, שינוי זה נובע מהשימוש במבנה נתונים $Dict$ אשר חוסך את השוואת הרישא של כל אחת מהמחרוזות עבור כל מעבר על סיפא, בעזרת שמירת כל הרישות ב$Dict$ קודם לכן, היה ניתן לראות שיפור משמעותי בזמני הריצה כתוצאה מיעול זה.

ההרצה של prefix_suffix_overlap_hash2 הייתה המהירה מבין השלושה, היא השתמשה בטכניקה זהה לפונקציה הקודמת, אך בעזרת מבנה הנתונים המובנה של פייתון $dict$, נראה כי פונקציה זו רצה מהר יותר פי 2 ביחס לפונקציה $hash1$ עבור הרצה של 1000 מחרוזות, ועבור הרצה של 4000 מחרוזות, פונקציה זו רצה בממוצע פי 3 מהר יותר מ$hash1$, גדילה אשר מעידה על סיבוכיות טובה יותר, הנובעת ככל הנראה כתוצאה מההבדלים במימוש הפנימי של $dict$ של פייתון לעומת $Dict$ הנכתב בתרגיל.