CS1001.py HW5

שאלה 1

סעיף ב׳

b. סיבוכיות זמן הריצה: $\boxed{\boldsymbol{O(n)}}$.

האלגוריתם בודק עבור כל צומת בעץ האם תכונת ״ערמת המינימום״ מתקיימת. כל קריאה רקורסיבית מייצגת צומת בעץ או $None$ (בן ״ריק״ של עלה). בנוסף, לכל צומת בעץ ניתן להגיע מצומת אחד בלבד (רק אב קורא לבן כלשהו). לכן יש $O(n)$ קריאות רקורסיביות, כי גם אם לכל צומת מ-n הצמתים בעץ היינו מניחים שמתבצעות שתי קריאות נוספות ל-$None$ (מה שכן קורה עבור העלים), היו $3n = O(n)$ קריאות רקורסיביות. עבור כל שלב רקורסיבי סיבוכיות הזמן היא קבועה $O(1)$: עבור צומת ״ריק״ זה טריוויאלי, אבל גם עבור צומת לא ריק מתבצעות בסה״כ השוואת מספרים (אנחנו לא מנתחים פה לפי מספר הביטים שלהם), בדיקת תנאים לוגיים ואופרטור בוליאני and בין 3 שלושה ערכים בוליאניים, כלומר $O(1)$. סה״כ סיבוכיות הזמן היא ככמות הקריאות הרקורסיביות $O(n)$.

שאלה 2

חלק ג׳

נסתמך על אלגוריתם ״הצב והארנב״ של $Floyd$ שראינו בתרגול. נגדיר שני צבים ושני ארנבים לצומת הראשון, שיתקדמו באופן דומה לאלגוריתם מהתרגול, אלא שיבחרו (לפחות עד לצומת הפיצול) בכל איטרציה בדרך שאין בה $None$. אם אין פיצול ברשימה, כלומר לא קיים צומת אשר בו שני המצביעים $next1$ ו-$next2$ שונים מ-$None$, אזי הרשימה ״מתנהגת״ כמו רשימה מקושרת ״רגילה״, ולכן הצב והארנב (למעשה שני הזוגות) ייפגשו אם ורק אם קיים מעגל בה, והאלגוריתם יפעל באותה סיבוכיות שראינו בתרגול (ועונה על תנאי השאלה). אחרת, כל זוג של צב וארנב עם אינדקס מתאים (ארנב1 וצב1 וכו׳) יפנה לתת-רשימה שונה בצומת הפיצול. אם באחת מתתי הרשימות אין מעגל, הארנב בתת רשימה זו יגיע לצומת בו שני המצביעים $next1$ ו-$next2$ שווים ל-$None$. אם קיים מעגל פנימי בתוך תת רשימה, הזוג המתאים ייפגש בדיוק כמו באלגוריתם מהתרגול. אם מגיעים מתוך תת רשימה לרשימת המזלג לפני הפיצול, הארנב המתאים יזהה באיזשהו שלב צומת פיצול, ומתנאי השאלה (צומת פיצול אחד לכל היותר) זה אותו צומת פיצול היחיד, כלומר יש מעגל. אם מגיעים מתת רשימה לתת הרשימה הנגדית לה (המסלול השני), אז או שמגיעים (עם הארנב) לקצה שלה, כלומר לצומת עם שני $None$, ואז אין מעגל, או שחוזרים לתת הרשימה הנגדית, והגענו למעגל שאותו יזהה הזוג שהדרך שלו אליו הייתה הקצרה ביותר. בסה״כ, עבור כל מקרה נחזיר $True$ אמ״ים יש מעגל, בזמן סופי ובסיבוכיות זמן $O(n)$ ומקום $O(1)$ (כך הוכחנו בתרגול עבור אלגוריתם הצב והארנב, עליו מתבסס האלגוריתם שהצעתי, בכל מקרה אפשרי).

שאלה 3

סעיף א׳

יהיו $a,b,c \in \mathbb{N}$. מתקיים: $a \bmod b = a - b \cdot \left\lfloor \frac{a}{b} \right\rfloor$. נסמן: $k \coloneqq \left\lfloor \frac{a}{b} \right\rfloor$, $r \coloneqq a \bmod b$ והם כמובן שלמים. כלומר:

$a = kb + r$. מתקיים:

$$(\boldsymbol{a \bmod b})^{\boldsymbol{c}}\ \boldsymbol{mod\ b} = r^c \bmod b = r^c - b \cdot \left\lfloor \frac{r^c}{b} \right\rfloor$$

$$\boldsymbol{a^c \bmod b} = (kb + r)^c \bmod b \underbrace{=}_{(\text{בינום})} \sum_{i=0}^{c} \binom{c}{i} k^i b^i r^{c-i} - b \cdot \left\lfloor \sum_{i=0}^{c} \binom{c}{i} k^i b^{i-1} r^{c-i} \right\rfloor =$$

$$= r^c + \cancel{ckbr^{c-1}} + \cancel{\binom{c}{2}k^2b^2r^{c-2}} + \cdots + \cancel{k^cb^c} - b \cdot \left\lfloor \frac{r^c}{b} + \cancel{ckr^{c-1}} + \cancel{\binom{c}{2}k^2br^{c-2}} + \cdots + \cancel{k^cb^{c-1}} \right\rfloor =$$

$$\underbrace{=}_{*} r^c - b \cdot \left\lfloor \frac{r^c}{b} \right\rfloor = (\boldsymbol{a \bmod b})^{\boldsymbol{c}}\ \boldsymbol{mod\ b}$$

כאשר המעבר שלפני האחרון, $*$, נובע מהקשר $\lfloor x + n \rfloor = \lfloor x \rfloor + n$ עבור x ממשי ו-n שלם, ובבירור, פרט ל-$\frac{r^c}{b}$, שאר המחוברים בהכרח שלמים עבור $r,\ c,\ k,\ b$ טבעיים.

סעיף ב'

נתונים a', p, g, $y = g^b \ mod \ p$, $x = g^a \ mod \ p$ ונניח לחומרה (לכל היותר) שכולם בעלי n ביטים (עבור x ו-y ניתן להסיק זאת מהשארית (הם לכל היותר $(p-1)$)). וכן מתקיים $g^{a'} \ mod \ p = g^a \ mod \ p = x$. נחשב את $y^{a'} \ mod \ p$. ראינו שחזקה מודולרית כאשר כל הארגומנטים בעלי לכל היותר n ביטים, עולה (לפי הקוד מהכיתה) לכל היותר $O(n^3)$. נקבל:

$$y^{a'} \ mod \ p = (g^b \ mod \ p)^{a'} \ mod \ p \underbrace{=}_{(\text{סעיף א'})} g^{a'b} \ mod \ p \underbrace{=}_{(\text{סעיף א'})} \left(g^{a'} \ mod \ p\right)^b \ mod \ p =$$
$$= (g^a \ mod \ p)^b \ mod \ p = g^{ab} \ mod \ p$$

כלומר אין זה משנה אם מצאנו $a' \neq a$ שמקיים $g^{a'} \ mod \ p = g^a \ mod \ p$, כדי לקבל את המפתח צריך לפעול באופן דומה לאליס, ולחשב את $y^{a'} \ mod \ p$. הסיבוכיות $\boldsymbol{O(n^3)}$ (ב-WC) יעילה.

שאלה 4

סעיף א'

טענה: סיבוכיות זמן הריצה היא $\boxed{\boldsymbol{O(4^m \cdot n^2)}}$.

הוכחה: מספר האיטרציות הוא כמספר הביטים של b (בשל פעולת החלוקה השלמה $b//2$), m. המכפלה של $result$ מתבצעת רק כאשר הביט הימני של b באותה איטרציה הוא 1, כלומר כמספר הביטים הדולקים בהצגה הבינארית שלו. במקרה הגרוע b הוא מספר מרסן, כלומר כל הביטים שלו דולקים (חזקה שלמה של 2 פחות 1), ולכן שתי פעולות הכפל ופעולת החלוקה השלמה מתבצעות בכל איטרציה. באשר לתנאי שבלולאה ולתנאי שב-$if \ statement$, הם מתבצעים בזמן קבוע לפי ההנחיות (כמו גם השמות). נחשב את כמות העבודה שמתבצעת ע"י כל אחת משלוש הפעולות בכל איטרציה.

$\boldsymbol{b = b//2}$: לפי ההנחיות, פעולה זו רצה בזמן לינארי ב-m, נניח לשם הפשטות שהעבודה היא $m-1$ (עבור m כלשהו). לאחר כל פעולה כזו, מספר הביטים של b קטן ב-1. סך העבודה:

$$(m-1) + (m-2) + \ldots + 0 = \sum_{i=0}^{m-1} i \underbrace{=}_{\left(\substack{\text{סכום} \\ \text{סדרה} \\ \text{חשבונית}}\right)} O(m^2)$$

$\boldsymbol{a = a * a}$: ראינו בכיתה שפעולת כפל של שני מספרים שלמים בעלי n ביטים כל אחד יכולה להתבצע בסדר גודל של n^2 פעולות, כלומר בסיבוכיות זמן של $O(n^2)$. יש להתחשב בכך שערכו של a גדל בריבוע בכל איטרציה. במקבילית של אלגוריתם כפל ארוך, תהיינה בערך $2n-1$ עמודות (n תווים של העליונה + $(n-1)$ הזחות) כלומר סדר גודל של $2n$ ביטים עבור a^2. כלומר בכל איטרציה מספר הביטים של a גדל פי 2. לשם הפשטות נניח שהעבודה עבור n ביטים היא n^2 (הקבוע הוא 1) סך העבודה:

$$\boldsymbol{length \ of \ a}: \quad n, 2n, 4n, \ldots, 4^{\frac{m-1}{2}} n$$

$$\boldsymbol{work}: \quad n^2 + 4n^2 + 16n^2 + \ldots + 4^{m-1}n^2 = n^2 \sum_{i=0}^{m-1} 4^i \underbrace{=}_{\left(\substack{\text{סכום} \\ \text{סדרה} \\ \text{הנדסית}}\right)} n^2 \cdot \frac{4^m - 1}{3} = O(4^m \cdot n^2)$$

$\boldsymbol{result = result * a}$: נניח שפעולת הכפל מתבצעת לפי האלגוריתם של כפל ארוך. $result$ קטן מ-a בכל איטרציה (כי מתחיל מ-1 ובכל פעם מוכפל ב-a בעוד ש-a מתחיל מ-a ובכל פעם מוכפל ב-a. נניח לחומרה ש-$result$ באורך ביטים של a ולכן סיבוכיות המכפלה היא $O(n^2)$. כעת ניתוח הסיבוכיות זהה לניתוח עבור $a = a * a$, וזאת כאמור בהחמרה למה שקורה בפועל (הגדלנו את $result$ בניתוח זה בכל איטרציה), ולכן הסיבוכיות פה היא גם $O(4^m \cdot n^2)$. הסיבה שגם עם ההחמרה נקבל חסם הדוק, היא שפעולת הכפל $\boldsymbol{a = a * a}$ היא המשמעותית ביותר, ולכן אם נחבר את כל העבודות, הסיבוכיות תהיה הסיבוכיות שלה. סה"כ אם נחבר את כל העבודות של שלוש הפעולות, נקבל שהסיבוכיות היא $O(2 \cdot 4^m \cdot n^2 + m^2) = \boldsymbol{O(4^m \cdot n^2)}$. כאשר בצעד האחרון השתמשנו בקשר שראינו מהכיתה ש-O של סכום פונקציות הוא O של הפונקציה המקסימלית.

שאלה 5

סעיף ב'

סיבוכיות זמן הריצה היא $\boxed{\boldsymbol{O(kn^2)}}$.

הפעולות שלפני הלולאה החיצונית מתרחשות בזמן קבוע. בכל איטרציה של הלולאה החיצונית מתבצע $slicing$ באורך k כלומר סיבוכיות זמן $O(k)$. הלולאה החיצונית רצה n פעמים ולכן פעולה זו עולה בסה"כ $O(kn)$. בלולאה הפנימית אנו מבצעים שוב $slicing$ באורך k, ואז משווים 2 מחרוזות באורך k. השוואה כזו (לפי ההנחיות) עולה $O(k)$ פעולות ב-WC, שמתקבל (עבור האלגוריתם להשוואת מחרוזות שמוצע בשאלה) כאשר 2 המחרוזות הללו זהות זו לזו (במקרה הגרוע כל שתי מחרוזות שנבדוק זהות). הוספה לסוף הרשימה - $O(1)$. סה"כ כמות העבודה בכל איטרציה פנימית עולה $O(k)$. כמות האיטרציות הפנימיות שתורמות $O(k)$ (השאר לוקחות $O(1)$ בגלל הדילוג): $(n-1)\cdot n = O(n^2)$. לכן סך כל העבודה: $O(kn^2+kn) = \boldsymbol{O(kn^2)}$ (O של סכום פונקציות הוא O של הפונקציה המקסימלית).

סעיף ה'

סיבוכיות זמן הריצה בממוצע היא $\boxed{\boldsymbol{O(kn)}}$.

בחרנו במילון עם n תאים (כי נצפה ל-n איברים שונים וזה די יעיל להגדיר כ-$O(n)$ איברים). הגדרת המילון עולה $O(n)$ לאתחול הטבלה. מבצעים n הכנסות. בקריאה ל-$insert$ מתבצע $slicing$ באורך k, $O(k)$. בתוך $insert$ חישוב פונקציית ה-$hash$ על מפתח באורך k עולה $O(k)$ (מודולו בזמן קבוע), וה-$packing$ עולה $O(k)$ (יצירת רשימה והעתקה אליה מחרוזת באורך k). סה"כ עלות ההכנסות היא $O(nk)$. בלולאה השנייה שרצה n פעמים, בכל איטרציה מפעילים את $find$ כאשר הקריאה אליה דורשת $slicing$ ועולה שוב $O(k)$. $find$ מפעילה את פי ה-$hash$ כמו קודם, $O(k)$. בממוצע יהיו $\alpha = \frac{n}{m} = 1$ איברים בתא, ולכן בממוצע מספר האיטרציות קבוע, אולם בכל איטרציה נעשית השוואת מחרוזות באורך k, $O(k)$ לפי ההנחה. סה"כ פעולת החיפוש בכל איטרציה בלולאה השנייה עולה $O(k)$ בממוצע. תחת הנחות הסעיף, לא ניכנס ללולאה הפנימית, כי $vals$ ריקה תמיד. לכן הלולאה השנייה עולה $O(nk)$ בממוצע. לכן בממוצע, סך העבודה היא $O(kn+n) = \boldsymbol{O(kn)}$ (O של סכום פונקציות הוא O של הפונקציה המקסימלית).

סעיף ז'

הפתרון הראשון התקבל כצפוי האיטי ביותר (די משמעותית), ולא בכדי – הוא משווה בין כל שתי רישות וסיפות (מלבד הרישא והסיפא של אותה מילה). זמן הריצה עבורו גם גדל באופן משמעותי יותר עבור הגדלת הקלטים, כי סיבוכיות הזמן שלו היא הגדולה ביותר. הפתרון השני והשלישי קרובים יותר מבחינת זמן הריצה, וגם גדלים בצורה דומה. הסיבה היא כי שני האחרונים פועלים באופן דומה - משתמשים בטבלאות $hash$ במימושם ופועלים לפי הרעיון: כל הרישות יוכנסו למילון תחילה, ואז נעבור על כל הסיפות ונבדוק לכל אחת אם היא נמצאת במילון. אף הסיבוכיות בממוצע זהה. אלא שהשלישי בכל זאת מהיר יותר. ניתן לתלות את הסיבות לכך בטיפול שונה בהתנגשויות (ויותר יעיל כנראה), אתחול לגודל קטן (השערה, בדומה ל-set) וגדילה של מבנה הנתונים בצורה יעילה יותר. כלומר מבחינה אסימפטוטית ממוצעת, החסמים של מימוש זה לעומת הקודם זהים, אבל בפועל הקבועים קטנים יותר, ולכן האחרון מהיר יותר.