שאלה 1

סעיף א'

(ב) נניח שמתקיים עבור $n_0 = 8$, $c = \frac{1}{3}$:

יהי $n > n_0 = 8$. קיים $k \in \mathbb{N}$ כך ש $2^k \le n$ ואבל $2^{k+1} > n$. נעדכן השימור L באורך n ו $Var = 5$, כך ש $L[2^k] = 8$, $L[2^k - 1] = 5$,

$\forall i < 2^k - 1 \quad L[i] = 1$. לשאר איברי השימור לא מתייחס לשם הפורמלי ונאתחל אותם ב 10.

רשימה, נשים לב שלא יתכן שאינדקס הלולאה החיצונית יעבור את $L[2^k - 1]$ לא הי"ר מתקיימת קפיצה לוגריתמית מעבר לו שכן כל הגדלים יחרין הוניים unn) ובכל איטרציה אינדקס השימור i גדל לכל היותר ב $\frac{n-i}{2}$, כלומר אנו מתקדמים לפחות $\frac{1}{2}n$ ולכן היום שלא יתקיים $L[i] = Var$ עד ש $i = 2^k - 1$, נבדוק לפחות $\log n$ (איטרציות) איטרציות בלולאה הפנימית (i מאיק הון שווה ל $2^k - 1$ לכל i נכון $L[i] = 1$). מס' הקפיצות בלולאה שנגרר אישור. הרי הקפיצה של 2^0. מספר הקפיצות הוא k.

באיטרציה ה- i של הלולאה החיצונית נבדוק $k + 1 - i$ איטרציות בלולאה הפנימית שזה לכל הפחות $\log n - i$, וזה מבטיח שנעבור על כל $p.next\_list$.

אם כן, נבדוק סה"כ איטרציות:

$$\sum_{i=0}^{\log n - 1} \log n - i = \log^2 n + \sum_{i=0}^{\log n - 1} -i =$$

סכום סדרה חשבונית

$$\log^2 n - \frac{\log^2(n-1)}{2} + \frac{\log(n-1)}{2} > \log^2 n - \frac{\log^2(n)}{2} + \frac{\log(n-1)}{2} =$$

$$\frac{1}{2}\log^2 n + \frac{1}{2}\log(n-1) > \frac{1}{2}\log^2 n > \frac{1}{3}\log^2 n = c \cdot \log^2 n$$

כנדרש.

סעיף ב'

הלולאה החיצונית למעשה פועלת באותו האופן כמו בפתרון המקורי, אך כעת הלולאה הפנימית מוצאת את האיבר הבא עם חיפוש בינארי ולא עם בדיקה ידנית איבר-איבר.

לכן נקבל לכל היותר בלולאה הפנימית סיבוכיות של חיפוש בינארי על p.next_list, שכפי שכבר ראינו בבדיקה הידנית גורר סיבוכיות של O(logn), ולכן עם חיפוש בינארי תגרור סיבוכיות של O(log(logn)). נשים לב שמדובר בחסם הדוק כיוון שlogn הוא החסם ההדוק של חיפוש בינארי. לכן החסם ההדוק של כל הפונקציה הינו O(logn*log(logn)), שוודאי יעיל אסימפטוטית מO(logn*logn).

שאלה 2

סעיף א'

i. k יכול להיות 0 במידה ו-n=1 (וכמובן לא יכול להיות קטן מ0). הגורם המינימלי שיכול להיות ברשימה הוא 2, ולכן עבור n מספר טבעי שעבורו קיים i טבעי כך ש 2**i=n, נקבל את האורך המקסימלי של k, שהוא log על בסיס 2 של n.

ii. הפונקציה is_sorted מסיבוכיות $O(k)$.
מתכונות של מכפלה בינארית $\sum_{i=1}^{k} a_i \le n$ ולכן
$a_i \ge 2$ $\forall i$ כיוון ש $P_i \ge 2$.
כמות הפעולות בלולאה ה for (לא נ"חס הרצת ה number*=p)
כיוון שבונים $O(n^3)$ הי"א $O(\sum_{i=1}^{k} a_i^3)$ modpower $(\Theta(a_i^3))$.
מאי־שוויון כפולה $\sum_{i=1}^{k} a_i^3 \le (\sum_{i=1}^{k} a_i)^3 = n^3$ ולכן אכן
$O(n^3)$. נשים לב שבמי חסם העליון מהיוון שבמקרה ש number
מספר הראשוני P_1 =number, $k=1$, מתבצעת לולאה בלולאה ה for
מסיבוכיות של $\Theta(n^3)$, הנקרא.

סעיף ד'

i. הפונקציה למעשה בונה את רשימת הגורמים של הlcm כך שכל גורם מאיחוד רשימות הגורמים של המספרים שהתקבלו מופיע בה לפחות פעם אחת, ומספר הפעמים המדויק שהוא מופיע בה הוא כמספר הפעמים שהוא מופיע ברשימה שבה יש את הכמות המקסימלית של המופעים שלו מבין כל שאר הרשימות. איך היא עושה זאת? היא יוצרת רשימה של מילונים כך שכל מילון מכיל כמפתחות את הגורמים של המספר שהוא מייצג, וכערכים את מספר הפעמים שהגורם מופיע ברשימת הגורמים של המספר. היא יוצרת מילון נוסף שמכיל כמפתחות את כל הגורמים שמופיעים באיחוד רשימות הגורמים של כל המספרים, וכערכים את המספר המקסימלי של מופעים שיש לו מבין כל אחת מהרשימות (נעזרת במילון הקודם לשם כך).

לבסוף היא ממירה את מילון זה לרשימה כך שכל ערך של גורם מייצג את כמות המופעים של הגורם ברשימה, ויוצרת את הlcm על ידי רשימת הגורמים הזאת.

ii. האתחול של lists וf_i_d הוא כמובן מסיבוכיות כוללת של O(s). הלולאה המקוננת הראשונה עוברת למעשה על הגורמים של איחוד הגורמים מכל המספרים (במקרה הגרוע יש גורם אחד לכל מספר), O(s). הלולאה המקוננת השנייה עוברת גם היא על כל הגורמים, O(s). הלולאה שיוצרת את הרשימה משתמשת בf_m_d.items(), שמסדרת את כל הגורמים והמונים שלהם במבנה נתונים ולכן O(s), ועוברת עליה לאחר מכן בסיבוכיות של O(s). אם כן סך הסיבוכיות של הפונקציה O(s).

שאלה 4

סעיף ב'

בכל צעד של הרקורסיה מתבצעת קריאה שמתקרבת צומת אחת נוספת אל כיוון הצומת המבוקשת. במקרה הגרוע הצומת המבוקשת נמצאת בצד "הכבד" יותר של העץ, שכולל לכל הפחות n * (1-q) צמתים, בסופו כעלה. אם כן, קל לראות שכיוון שq קבוע מדובר בסיבוכיות של O(n).

שאלה 5

סעיף ב'

המקרה הגרוע מתקבל כשלכל שני אינדקסים 0 <= i, j <= n, i שונה מj, נקבל שהזוג הסדור (i, j) שייך לרשימה שהפונקציה מחזירה. דוגמה פשוטה למקרה כזה היא כשכל המחרוזות מורכבות מאותו התו (lst = ["a" * k for i in range(n)] למשל). במקרה כזה, מתבצעות בדיקות על כל הזוגות הלא רפלקסיביים של מחרוזות ברשימה שהתקבלה, סה"כ n*(n-1) בדיקות. בכל בדיקה יש שני slices באורך k, 2*k פעולות, והשוואה של שתי מחרוזות זהות באורך k, k פעולות. כלומר מתבצעות n*(n-1)*3k פעולות, סיבוכיות של O((n**2)*k).

סעיף ה'

בלולאה הראשונה אנחנו מכניסים את כל המחרוזות ברשימה לתוך המילון. יש n מחרוזות ועל כל מחרוזת מתבצע slicing בגודל k. סה"כ סיבוכיות O(n*k).

בלולאה השנייה אנחנו עוברים שוב על כל המחרוזות ברשימה. הפעם לכל מחרוזת עושים slicing באורך k, ומבצעים קריאה לfind עליה (find תמיד יחזיר מחרוזת ריקה ולכן לא יתבצע המשך הלולאה). הואיל וגודל המילון הוא n, בממוצע יש בכל תא במילון איבר אחד, ולכן בממוצע סיבוכיות הקריאה לfind היא O(1). לכן גם הלולאה השנייה מסיבוכיות O(n*k) ולכן סך כל הסיבוכיות של הפונקציה היא O(n*k) גם כן.