<u>תרגיל בית 05 אביתר שמש 322623182</u>

שאלה 1 סעיף א 1) יהיו $a,b,c \in N$

נסמן: $a = k_a*b + r_a,\ \ c = k_c*b + r_c$

לכן: $a \bmod b = r_a,\ c \bmod b = r_c$

לכן: $((a \bmod b)+(c \bmod b)) \bmod b = (r_a+r_c) \bmod b$

בנוסף, $a+c = k_a*b+r_a + k_c*b+r_c = (k_a+k_c)*b + (r_a+r_c)$

לכן, $a+c \bmod b = r_a+r_c \bmod b$ (כי k_a,k_c מתחלקים ב- b ללא שארית, לכן גם k_a+k_c)

לכן, $(a+c) \bmod b = ((a \bmod b)+(c \bmod b)) \bmod b$

שאלה 1 סעיף א 2) יהיו $a,b,c \in N$

נסמן: $a = k_a*b + r_a,\ \ c = k_c*b + r_c$

לכן: $a \bmod b = r_a,\ c \bmod b = r_c$

לכן: $((a \bmod b)*(c \bmod b)) \bmod b = (r_a*r_c) \bmod b$

$a * c = (k_a*b+r_a) * (k_c*b+r_c) = (k_a*k_c * b^2 + k_a*b *r_c+ k_c*b*r_a+ r_a+r_c)$

לכן: $(a*c) \bmod b = (r_a+r_c) \bmod b$

לכן: $((a \bmod b)*(c \bmod b)) \bmod b = (a*c) \bmod b$

שאלה 1 סעיף א 3) יהיו $a,b,c \in N$.

נסמן: $a = k_a*b + r_a$

נוכיח באינדוקציה ש – $(a \bmod b)^c = a^c \bmod b$

בסיס: $c = 1$: $(a \bmod b)^1 = a^1 \bmod b$

צעד: נניח כי נכון עבור $c = n$ טבעי, כלומר $(a \bmod b)^c = a^c \bmod b$

נוכיח כי $(a \bmod b)^{c+1} = a^{c+1} \bmod b$

$(a \bmod b)^{c+1} = (a \bmod b)^c * (a \bmod b) = a^c \bmod b * a \bmod b = (a^c*a) \bmod b = a^{c+1} \bmod b$

המעבר המסומן בכחול נובע מהנחת האינדוקציה. המעבר המסומן באדום נובע מהסעיף הקודם.

שאלה 1 סעיף ב) יהיו g, p, x, y כך ש : $x = g^a \bmod p,\ y = g^b \bmod p$.

הסוד של צד א': a. הסוד של צד ב': b.

הסוד המשותף: $g^{ab} \bmod p = (g^a * g^b) \bmod p\ = ((g^a \bmod p)\ (g^b \bmod p)) \bmod p$

נניח כי מצאנו a' כך ש $g^a \bmod p = g^{a'} \bmod p$

לכן: $((g^a \bmod p)(g^b \bmod p)) \bmod p = ((g^{a'} \bmod p)(g^b \bmod p)) \bmod p = (g^{a'}*g^b) \bmod p = g^{a'b} \bmod p$

לכן, יש $p-1$ אפשרויות עבור b.

וכך, גילינו ביעילות את הסוד המשותף.

שאלה 2 סעיף א 1) בהינתן n**,** מספר הביטים של הקלט, טווח הערכים של המספר x הוא: $2^{n-1} \le x \le 2^n-1$

אם הקלט שאנחנו מקבלים הוא 1, רשימת המחלקים תהיה ריקה(כי 1 לא ראשוני) לכן k המינימלי הוא 0.

בנוסף, כאשר המספר הוא 2^{n-1} **,** כמות המחלקים היא n-1(2,2,2,2,…,2 n-1 פעמים).

כל עוד כמות הביטים היא n**,** לעולם לא נוכל לחלק את המספר ב-n מחלקים ובפרט מחלקים ראשוניים(המחלק הקטן ביותר הוא 2, וכאשר נכפיל את 2 n פעמים, נקבל 2^n, מספר שמיוצג ע"י n+1 ביטים).

לכן, בהינתן קלט עם n ביטים, אורך הרשימה factors היא באורך: $0 \le k \le n-1$**.**

שאלה 2 סעיף א 2) ננתח את סיבוכיות זמן הריצה במקרה הגרוע:

O(1) – if verify

assert is_sorted(factors) – ראינו בהרצאה שבמקרה הגרוע(עוברת על כל רשימת המחלקים והיא אכן מסודרת), סיבוכיות זמן הריצה היא כמספר הביטים של p_i, כלומר a_i, לכן סיבוכיות זמן הריצה הכוללת של is_sorted(): $O(\sum_{i=1}^{k} a_i)$.

O(1) – number = 1

for p in factors – רץ k פעמים.

O(1) – if verify, לכן לאחר k איטרציות: O(k) = O(1).

assert(is_prime(p)) – ראינו בהרצאה שסיבוכיות זמן הריצה של הפעולה הזו היא $O(n^3)$, לכן לכל p_i, סיבוכיות זמן הריצה היא $O(a_i^3)$. לכן, עבור k הריצות : $O(\sum_{i=1}^{k} a_i^{\,3})$

number *= p – הכפלת מספרים היא בסיבוכיות כמות הביטים של המספרים הללו, לכן: $O(a_i * t)$ כך ש t הוא מספר הביטים הנוכחי של number. חשוב לציין ש number הוא בעצם מכפלה של כל המחלקים p(על מנת ליצור את המספר), לכן סיבוכיות זמן הריצה הכוללת של פעולה זו לאחר k איטרציות: $O(\prod_{i=1}^{k} a_i)$

O(1) – self.number = number

O(1) – self.factors = factor

לכן, סיבוכיות זמן הריצה הכוללת: $O(1)+ O(\sum_{i=1}^{k} a_i) + O(\sum_{i=1}^{k} a_i)^3 + O(\prod_{i=1}^{k} a_i) + O(1)+O(1)$
$=O(1)+O(n)+O(n^3)+O(n)+O(1)+O(1) = O(n^3)$

יהי n מספר הביטים של הקלט. נבחר קלט עבורו גודל קבוצת המחלקים הוא 1(קיים כזה בהכרח, הואר גם בפיאצה). זמן הריצה עבור הקלט הזה הוא $O(n^3)$.

הוכחנו כי במקרה הכללי חסם עליון הוא $O(n^3)$, וחסם תחתון עבור מקרה תלוי n גם הוא $O(n^3)$, ולכן $O(n^3)$ חסם הדוק.

שאלה 2 סעיף ד 1) בשורה הראשונה, מגדירים set ריק: my_set

בשורה לאחריה, מגדירים רשימה של רשימות (בשם lists), שכל רשימה היא המחלקים של אחד המספרים בקלט.

לאחר מכן, מגדירים מילון לכל רשימת factors, שכל זוג ברשימה הוא מחלק :0(לדוגמא עבור factors [2,2,3] המילון ייראה כך: {2:0, 3:0}, כי במילון לא יכולים להיות מפתחות זהים). שם המילון: f_i_d

האיטרציה היא על כל רשימת factors, ובתוכה איטרציה על כל איבר ב factors הנוכחי.

מוסיפים כל איבר לסט my_set , ומעדכנים את המילון הרלוונטי(המילון של factors עבור האיבר הנוכחי) ב 1=+, מה שבעצם סופר בכל מילון, כמה פעמים כל מחלק מופיע.

לאחר מכן, מגדירים מילון חדש שמכיל את כל המחלקים(my_set מכיל את כל המחלקים של כל המספרים בקלט), והערך לכל אחד מהמחלקים הוא 0(על מנת לספור כמה פעמים כל מחלק מופיע במקסימום בכל אחד המספרים). שם המילון: f_m_d.

עוברים באיטרציה על כל מילון ב f_i_d, ובתוכו על כל מפתח. אם כמות הפעמים שהוא מופיע במילון הנוכחי גדולה מכמות הפעמים שהוא מופיע במילון הכולל f_m_d, מעדכנים את הערך הרלוונטי ב f_m_d.

מגדירים רשימה חדשה, result. הרשימה תכיל כל מספר בכל המחלקים(f_m_d), וכמות הפעמים שכל מספר מופיע, היא כמות הפעמים שהוא הופיע במקסימום בכל factors.

כך מייצרים את הרשימה של כמות הפעמים המקסימלית שכל מחלק מופיע בכל אחד מהמספרים בקלט, ומייצרים אובייקט מהמחלקה FactoredInteger עבור המחלקים הללו.

שאלה 2 סעיף ד 2) מכך שכל מספר בקלט הוא ראשוני, אנחנו יודעים שכל רשימת factors תהיה באורך 1.

לכן:

הגדרת my_set – O(1)

הגדרת lists – שרשור של m רשימות באורך 1: O(m)

הגדרת f_i_d – הגדרת m מילונים שכל מילון מורכב מ factors(שכידוע באורך 1) : O(m)

הלולאה הראשונה תרוץ m פעמים, ובכל הרצה, הלולאה הפנימית תרוץ פעם אחת(המספרים ראשוניים לכן המחלק יחיד).

my_set.add(F) – O(1)

f_i_d[i][f] +=1 – O(1)

לכן, סה"כ הסיבוכיות של הלולאה: m*(O(1)+O(1)) = O(m)

הגדרת f_m_d – O(m)(מעבר על כל המחלקים, שבמקרה הגרוע בו כל המספרים הראשוניים שונים, כמות המחלקים היא ככמות המספרים, ולכן O(m)).

הלולאה השנייה תרוץ m פעמים(על כל m המילונים ב f_i_d), והלולאה הפנימית תרוץ פעם 1(כמו שאמרנו מקודם, לכל מספר יש מחלק ראשוני יחיד).

if f_d[f] > f_m_d[f] – O(1)

f_m_d[f] = f_d[f] = O(1)

לכן, סה"כ הסיבוכיות של הלולאה: m*(O(1)+O(1)) = O(m)

הגדרת result = O(1)

For k,v in f_m_d.items(): - הלולאה תרוץ m פעמים, ובכל פעם תשרשר ל-result את כמות הפעמים(v) שהמחלק (k) מופיע, במקרה הגרוע פעם אחת לכל מחלק, ולכן בסה"כ O(m)

לכן, סיבוכיות זמן הריצה הכוללת: O(1)+O(m)+O(m)+O(m)+O(m)+O(m)+O(1)+O(m) = O(m)

O(m)

שאלה 5 סעיף ב)

```
def prefix_suffix_overlap(lst, k):
    output = []   O(1)
    for i in range(len(lst)): n iterations
        for j in range(len(lst)): n iterations
            if i!=j:  O(1)
                if lst[i][:k] == lst[j][-k:]:  O(k)
                    output.append((i,j))  O(1)
    return output O(1)
```

סיבוכיות זמן הריצה של הלולאה הפנימית:

$O(1)+O(k)+O(1) = O(k)$

הלולאה הפנימית רצה n פעמים, לכן סיבוכיות זמן הריצה של הלולאה הפנימית:

$n*O(k) = O(nk)$

הלולאה החיצונית רצה n פעמים, לכן סיבוכיות זמן הריצה של הלולאה החיצונית:

$n*O(nk) = O(n^2k)$

לכן סיבוכיות הזמן הכוללת:

$O(1)+O(n^2k)+O(1) = O(n^2k)$

אם משווים תו תו בתתי המחרוזות, המקרה הגרוע ייתקבל כאשר בהשוואת שתי המחרוזות, התו הראשון שיהיה שונה הוא התו האחרון(התו ה-K של המחרוזת הראשונה).

שאלה 5 סעיף ה)

ניתוח הסיבוכיות ע"פ הגדרות השאלה:

$O(nk)$

```
# 5d
def prefix_suffix_overlap_hash1(lst, k):
    output = []   O(1)
    dic = Dict(len(lst))  O(n)
    for i in range(len(lst)): n*O(k) = O(n*k)
        dic.insert(lst[i][:k],i)  O(k)
    for i in range(len(lst)): n*O(k) = O(n*k)
        check = dic.find(lst[i][-k:]) O(k)
        for t in check: O(1)
            if i != t:  O(1)
                output.append((t,i))  O(1)
    return output
```