פתרון תרגיל מספר 4 - פתרון בעיות באלגוריתמים

שם: מיכאל גרינבאום, **ת.ז:** 211747639

16 בינואר 2021

1. **הערה:** את הרעיון הכללי של הבונוס פתרתי ביחד עם עידן אורזך.
צ"ל: אלגוריתם לבעיה ב- $O\left(n^2\right)$
הוכחה:
<u>אלגוריתם לבעיה:</u>

(א) נקבל מחרוזת S ($O\left(1\right)$)

(ב) נריץ את העיבוד המוקדם של בעית מנוע החיפוש עם המחרוזת S ($O\left(n\right)$)

(ג) בהרצאה ראינו דרך לחשב את $LCP\left(S\left[i:\right], S\left[j:\right]\right)$ ב- $O\left(1\right)$ לכל i, j בעזרת העיבוד המוקדם של בעית מנוע החיפוש, ונחשב את

$$t = \sum_{i=1}^{n}\sum_{j=i+1}^{n} 1_{LCP\left[S[i:],S\left[i+\frac{i+j}{2}:\right]\right]\geq\frac{i+j}{2}}$$

זה ייקח $O\left(n^2\right)$

(ד) נחזיר את t

כלומר זמן הריצה הוא $O\left(1\right)+O\left(n\right)+O\left(n^2\right)=O\left(n^2\right)$, כנדרש.

עתה מדוע האלגוריתם עובד?
נוכיח ש- $S\left[i:j\right]$ היא מהצורה $w \cdot w$ אם"ם $LCP\left(S\left[i:\right], S\left[i+\frac{i+j}{2}:\right]\right) \geq \frac{i+j}{2}$
וזה נכון בגלל ש

$$S\left[i:j\right] = w\cdot w \iff S\left[i:i+\frac{i+j}{2}\right] = w = S\left[i+\frac{i+j}{2}:j\right]$$
$$\iff LCP\left(S\left[i:\right], S\left[i+\frac{i+j}{2}:\right]\right) \geq |w| = \frac{i+j}{2}$$

כלומר מה שהאלגוריתם עשה הוא לעבור על כל תתי המחרוזות ולבדוק עבור כל אחת ב- $O\left(1\right)$ האם היא מהצורה $w\cdot w$ עם אותיות מ- S וסכם אם כן.

הערה: את הרעיון הכללי של הבונוס פתרתי ביחד עם עידן אורזך (לגבי המקרים שצריך להתייחס אליהם בלולאה).
בונוס:
<u>אלגוריתם לבעיה:</u>

(א) נקבל מחרוזת S ($O\left(1\right)$)

(ב) נריץ את העיבוד המוקדם של בעית מנוע החיפוש עם המחרוזת S ($O\left(n\right)$)

(ג) בהרצאה ראינו דרך לחשב את $LCP\left(S\left[i:\right], S\left[j:\right]\right)$ ב- $O\left(1\right)$ לכל i, j בעזרת העיבוד המוקדם של בעית מנוע החיפוש

(ד) נאתחל משתנה $t = 0$

(ה) לכל $1 \le k \le n$ נעשה את הדבר הבא:

i. נאתחל $i = n$

ii. אם $LCP\left[S\left[i-2k:\right], S\left[i-k:\right]\right] \ge k$, נעדכן $i = i - 1$, $t = t + 1$ (כלומר מצאנו סטרינג מהצורה $w \cdot w$ שמתחיל מ־ $i - 2k$ ו־ $|w| = k$, הצבה במה שראינו בחלק שלא בונוס)

iii. אחרת $LCP\left[S\left[i-2k:\right], S\left[i-k:\right]\right] = m < k$,

iv. נעדכן $i = i - (k - m)$ ונחלק ל־ 2 מקרים:

א'. אם $LCP\left[S\left[i-2k:\right], S\left[i-k:\right]\right] = k$, נעדכן $i = i - 1$, $t = t + 1$ (כלומר מצאנו סטרינג מהצורה $w \cdot w$ שמתחיל מ־ $i - 2k$ ו־ $|w| = k$)

ב'. אחרת $LCP\left[S\left[i-2k:\right], S\left[i-k:\right]\right] = m'$, נעדכן את $i = i - m'$

v. כל עוד $i \ge 1$ נחזור לשלב ii.

(ו) נחזיר את t

תחילה נראה כי $m' \le k - m$,
נסמן את הסטרינג שיש בחישוב ה־ LCP של m' ב־

$$S\left[i-2k:\right] = a_1 \dots a_{k-m} x_1 \dots x_m b_1 \dots b_{k-m} x_1 \dots x_m c_1 \dots c_{k-m} \dots$$

נשים לב ש־ $x_1, \dots, x_m$ מופיע פעמיים כי לפני שזזנו $k - m$ שמאלה ה־ LCP היה m.
עתה נניח כי $LCP\left(S\left[i-2k:\right], S\left[i-k:\right]\right) > k - m$, כלומר $a_1 \dots a_{k-m} = b_1 \dots b_{k-m}$, לכן מתקיים $a_1 \dots a_{k-m} x_1 \dots x_m = b_1, \dots b_{k-m} x_1 \dots x_m$.
כלומר $LCP\left(S\left[i-2k:\right], S\left[i-k:\right]\right) = k$ או ש־ $LCP\left(S\left[i-2k:\right], S\left[i-k:\right]\right) \le k - m$.
נשים לב שכמות הפעמים שנגיע ל־ ה' ii הוא $O\left(n \log\left(n\right)\right)$ מהנתון שיש לכל היותר $O\left(n \log\left(n\right)\right)$ תתי מחרוזות מהצורה $w \cdot w$.
באופן דומה כמות הפעמים שנגיע ל־ ה' iv א' הוא $O\left(n \log\left(n\right)\right)$.
נשים לב שבשאר הפעמים נגיע ל־ ה' iv ב' ובמקרה זה קידמנו את i בהתחלה $k - m$ ולאחר מכן ב־ $k - m'$ ולכן קידמנו אותו ־

$$k - m + (k - m') \ge k - m + (k - (k - m)) = k - m + m = k$$

כלומר נגיע ל־ ה' iv ב' לכל היותר $O\left(\frac{n}{k}\right)$ פעמים באיטרציה ה־ k, ולכן נגיע ל־ ט'ה לכל היותר $= O\left(\sum_{k=1}^{n} \frac{n}{k}\right)$ $O\left(n \cdot \sum_{k=1} \frac{1}{k}\right) = O\left(n \log\left(n\right)\right)$.
כלומר מספר האיטרציות של הלולאות יהיה לכל היותר $O\left(n \log\left(n\right)\right) + O\left(n \log\left(n\right)\right) + O\left(n \log\left(n\right)\right) = O\left(n \log\left(n\right)\right)$,
ולכן זמן הריצה של האלגוריתם הוא לכל היותר $O\left(n \log\left(n\right)\right)$ כי בכל איטרציה עושים $O\left(1\right)$ חישובים והעיבוד המוקדם הוא גם $O\left(n \log\left(n\right)\right)$.
נשאר להוכיח שאם $LCP\left(S\left[i-2k\right], S\left[i-k\right]\right) = m < k$ לא נמצא סטרינג מהצורה $w \cdot w$ ל־ $k - m$ תזוזות שמאלה של i.
יהי i, נסמן $\dots S\left[i-2k:\right] = x_1 \dots x_m b_1 \dots b_{k-m} x_1 \dots x_m c_1 \dots c_{k-m}$, כאשר $b_1 \ne c_1$ (אחרת ה־ LCP לא היה בדיוק m ויכול להיות $m + 1$).
יהי $j < k - m$, נשים לב כי $S\left[(i-j)-2k:\right] = a_1 \dots a_j x_1 \dots x_m b_1 \dots b_{k-m} x_1 \dots x_m c_1 \dots c_{k-m}$,
נשים לב כי

$$S\left[i-j-2k : i-j-k\right] = a_1 \dots a_j x_1 \dots x_m b_1 \dots b_{k-m-j}$$
$$S\left[i-j-k : i-j\right] = b_{k-m-j} \dots b_{k-m} x_1 \dots x_m c_1 \dots c_{k-m-j}$$

התו ה־ $m + j + 1$ בהם שונה כי באחד זה b_1 ובשני זה c_1 ולכן הם לא מהצורה $w \cdot w$ עבור $|w| = k$, כנדרש.
כלומר במקרה שבו $LCP\left(S\left[i-2k\right], S\left[i-k\right]\right) = m < k$ לא נמצא סטרינג מהצורה $w \cdot w$ ל־ $k - m$ תזוזות שמאלה של i ובשאר המקרים התקדמנו לבדוק את הסטרינג הבא.

מ.ש.ל.☺

2. **צ"ל:** אלגוריתם לבעיה ב־ $O\left(n^2\right)$
הוכחה:
תזכורת: שאילתה של בעית מנוע החיפוש מוצאת את המקסימום של ה־ LCP בין המחרוזות בעיבוד המוקדם למחרוזת הנתונה
<u>אלגוריתם לבעיה:</u>

(א) נקבל מחרוזות S_1, S_2 ($O(1)$)
(ב) נריץ את העיבוד המוקדם של בעית מנוע החיפוש עם המחרוזת S_1 ($O(n)$)
(ג) נאתחל $maxLCP = 0$, $bestString = ''$
(ד) לכל $1 \leq i \leq n$:

i. נחשב את השאילתה של $S_2[i:]$ של בעית מנוע החיפוש $O(n)$ עם העיבוד המוקדם של S_1, נשמור את האינקס שמנוע החיפוש מצא בסוף ב־ j (כלומר הLCP המקסימלי בין $S_1[k:]$ ל־ $S_2[i:]$ מתקבל עבור $j=k$)
ii. נחשב $currLCP = LCP(S_1[j:], S_2[i:])$ ($O(n)$)
iii. אם $maxLCP < currLCP$:
א'. $maxLCP = currLCP$
ב'. $bestString = S_2[i : i + currLCP]$

(ה) נחזיר את $bestString$.

נשים לב שזמן הריצה הוא $O(1) + O(n) + n \cdot O(n) + O(1) = O(n^2)$, כנדרש.

עתה מדוע האלגוריתם עובד?
נסמן $maxLCP = \max_{1 \leq k \leq n, 1 \leq l \leq n} LCP(S_2[l:], S_1[k:])$,
נסמן ב־ i את האינדקס המינימלי המקיים $maxLCP = \max_{1 \leq k \leq n} LCP(S_2[i:], S_1[k:])$
נשים לב שבאיטרציה ה־ i של האלגוריתם, הוא יימצא את $S_2[i : i + maxLCP]$ מנכונות השאילתה של בעית מנוע החיפוש.
מהיות ו־ i הוא האינדקס המינימלי שמגיע לערך $maxLCP$, המחרוזת $S_2[i : i + maxLCP]$ תשמר ב־ $bestString$.
בכל איטרציה לאחר מכן, ה־ $currLCP$ יהיה קטן מ־ $maxLCP$ ולכן לא נעדכן את ה־ $bestString$.
לכן המחרוזת שתוחזר מהאלגוריתם היא $S_2[i : i + maxLCP]$ שזאת מחרוזת עם ה־ LCP המקסימלי.
עתה נסמן את המחרוזת המשותפת המקסימלית של S_1 ו־ S_2 ב־ s ולכן מתקיים כי $s = S_2[m:p]$ עבור m, p כלשהם, ולכן

$$\begin{aligned}
\max_{1 \leq k \leq n} LCP(S_2[i:], S_1[k:]) = maxLCP &= \max_{1 \leq k \leq n, 1 \leq l \leq n} LCP(S_2[l:], S_1[k:]) \\
&\geq \max_{1 \leq k \leq n} LCP(S_2[m:p], S_1[k:]) \\
&\geq |s|
\end{aligned}$$

כלומר הראנו כי $|S_2[i : i + maxLCP]| = maxLCP \geq |s|$.
עתה נשים לב כי $S_2[i : i + maxLCP]$ היא מחרוזת שמופיעה גם ב־ S_1 מנכונות השאילתה של בעית מנוע החיפוש ולכן ממקסימליות האורך של s נקבל כי $|s| \geq |S_2[i : i + maxLCP]|$.
לכן קיבלנו כי $|s| = |S_2[i : i + maxLCP]|$, כלומר המחרוזת שהחזרנו היא גם מחרוזת משותפת ל־ S_1 ו־ S_2 וגם באורך מקסימלי ולכן מקיימת את דרישות האלגוריתם, כנדרש.

מ.ש.ל.☺