פתרון תרגיל מספר 1 ־ פתרון בעיות באלגוריתמים

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

10 בנובמבר 2020

1. **צ"ל:** פתרון לבעיה ביעילות $O\left(n^3\right)$ או טוב יותר
הוכחה:

נסתכל על האלגוריתם הבא:

(א) נקבל רצף של נקודות $\{p_1, \ldots, p_n\}$, נמיין את הנקודות למערך S לפי היחס הבא

$$p_1 < p_2 \iff (p_1.x < p_2.x) \vee (p_1.x = p_2.x \wedge p_1.y < p_2.y)$$

נסמן את התוצאה של S ב־ $s_1 \ldots, s_n$.

(ב) נחשב מערך M כך ש־ $p_i = s_{M[i]}$ לכל $i \in [n]$.

(ג) ניצור מערך תוצאות C בגודל n.

(ד) עתה לכל $i \in [n]$ לפי סדר עולה:

i. נחלק למקרים לפני $p_i.val$, אם $p_i.val < 0$ (סומנה כבר) אז נעבור לאיטרציה הבאה, אחרת נמשיך לשלבים הבאים בלולאה

ii. נשנה את $p_i.val = -1$, ניצור רשימה מקושרת D שמכילה רק את p_i.

iii. לכל $j \in [M[i] + 1, n]$ לפי סדר עולה:

א'. אם $s_j.val \geq 0$, נעדכן

$$s_j.val = \max\{0, s_j.val - |\{p \in D \mid p.x < s_j.x \wedge p.y < s_j.y\}|\}$$

ב'. אם $s_j.val = 0$, נשנה את $s_j.val = -1$, נוסיף את s_j ל־ D.

iv. נעדכן $C[p] = i$ לכל $p \in D$

(ה) נחזיר את C.

נשאר להסביר מדוע האלגוריתם עובד ומדוע זמן הריצה הוא $O\left(n^3\right)$ או טוב יותר.
<u>מדוע נכון?</u>
יהיו $p_1, \ldots, p_n$ נקודות ו־ $s_1, \ldots, s_n$ מיון שלהן לפי סעיף 1 באלגוריתם.
תחילה יהיו p, q נקודות כך שסימון של p אמור לעדכן את ה־ val של q. יהיו $k, l \in [n]$ אינדקסים כך ש־ $s_k = p, s_l = q$ אזי מהיות שסימון של p אמור לעדכן את ה־ val של q אזי $s_k.x = p.x < q.x = s_l.x$ ומהיות המערך ממוין לפי ה־ x במקור, נסיק כי $k < l$. כלומר כל נקודה משפיעה אך ורק על הנקודות שאחריה במערך הממוין!
לכל נקודה p_i נגדיר

$$t_{i,j} = \text{number of marked points required from iterations } [i+1:n] \text{ to mark } s_j$$

עתה נרצה שבכל איטרציה i שבה צריך לעדכן תחילה את s_k, נרצה לעדכן את $t_{i,j}$ לכל $j \in [n]$, נשים לב מהאבחנה הקודמת כי $t_{i,j} = t_{i-1,j}$ לכל $j < k$ וגם

$$t_{i,j} = t_{i-1,j} - |\{s_l \mid k \leq l < j \wedge s_l \text{ was marked in the i'th iteration} \wedge s_l.x < s_j.x \wedge s_l.y < s_j.y\}|$$

לכל $j > k$.
כלומר בשביל לדעת את $t_{i,j}$ בכל רגע, מספיק בכל איטרציה לעבור על $j \in [k+1, n]$ ולעדכן לכל

$$s_j.val = t_{i,j} - |\{s_l \mid k \le l < j \wedge s_l \text{ was marked in the i'th iteration} \wedge s_l.x < s_j.x \wedge s_l.y < s_j.y\}|$$

נשים לב שבאלגוריתם שתיארתי מתקיים ש־ $D = \{s_l \mid k \le l < j \wedge s_l \text{ was marked in the i'th iteration}\}$ (מהנחת אינדוקציה ש־ $t_{i,j}$ עודכן לפי כלל זה עד האיטרציה ה־ i). וכאשר זהו D, כלל העדכון שקול ל-

$$t_{i,j} = t_{i-1,j} - |\{s_l \in D \mid s_l.x < s_j.x \wedge s_l.y < s_j.y\}|$$

וזה בדיוק מה שהאלגוריתם שתואר מלעיל עושה, הוא קודם ממיין כדי שהאבחנה הראשונה תתקיים. לאחר מכן הוא מוצא מיפויים כדי למצוא את s_k לכל p_i ולבסוף מבצע את העדכון שתואר פה עם שמירת התוצאות במערך עזר שמוחזר בסוף.
<u>זמן ריצה:</u>
עתה נשים לב ששלב 1 לוקח $O(n \log(n))$, שלב 2 לוקח $O(n^2)$ (אפשר בפחות אבל לא חשוב לתרגיל), שלב 3 לוקח $O(n)$, בשלב 4 נשים לב שכל נקודה p תהיה ב־ D בדיוק פעם אחת ובמקרה הכי גרוע תבדק מול כל נקודה אחרת ולכן כל נקודה תעשה $O(n)$ פעולות ויש n נקודות ולאחר מכן מהיות וסומנה לא תבדק שוב ולכן שלב זה לוקח $O(n^2)$, ושלב 5 לוקח $O(1)$.
לכן הזמן הכולל הוא $\boxed{O(n \log(n) + n^2 + n + n^2 + 1) = O(n^2)}$ שזה יותר טוב מהנדרש, לצערי לא הצלחתי את הבונוס על אף שחשבתי עליו הרבה.

מ.ש.ל.☺

2. **צ"ל:** פתרון לבעיית LCA ביעילות $(O(n^2), O(1), O(n^2))$
הוכחה:
נגדיר את האלגוריתם לפי שלב עיבוד מוקדם, שלה השאילתה ושטח.
<u>שלב עיבוד מוקדם:</u>

(א) נקבל עץ G, נמיין את המערך מיון טופולוגי ונסמן את תוצאת המיון ב־ $v_1, \ldots, v_n$
(ב) נקצה מערך A בגודל $|V|^2$.
(ג) נמלא $A[v_i, v_i] = v_i$ לכל $i \in [n]$
(ד) לכל $i \in [n]$:
i. לכל $j \in [i+1 : n]$ לפי סדר עולה:

$$A[v_i, v_j] = A[v_i, v_j.parent]$$
$$A[v_j, v_i] = A[v_i, v_j]$$

(ה) נשמור את A

<u>זמן ריצה של העיבוד המוקדם:</u> תחילה נשים לב שזמן הריצה של האלגוריתם הוא $O\left(|V| + |E| + |V|^2\right)$ ובעץ $|E| = |V| - 1$ ולכן זמן הריצה הוא $O\left(|V|^2\right) = O(n^2)$.
<u>נכונות שלב העיבוד המוקדם:</u> עתה נסביר ברעיון כללי מדוע בסוף הריצה $A[v_i, v_j]$ שומר את ה־ LCA של v_i, v_j (הוכחה באינדוקציה על סדר הריצה בשלב 4).
תחילה עבור $i = j$, מההגדרה ומאיך שמילאנו, אכן מקיים כי $A[v_i, v_j]$ שומר את ה־ LCA של v_i, v_j שהוא $v_i = v_j$.
עתה אם $i < j$ אז מקשירות הגרף מתקיים ש־ v_j הוא לא השורש ולכן $v_j.parent$ אכן מוגדר היטב.
נחלק ל2 מקרים:

(א) אם v_i מופיע במיון לפני $v_j.parent$ אז מסדר הריצה על ה־ jים נמלא את התא $A[v_i, v_j.parent]$ לפני $A[v_i, v_j]$ והוא יכיל את התוצאה הנכונה מהנחת האינדוקציה.

(ב) אם v_i מופיע אחרי $v_j.parent$ אז מסדר הריצה על ה־ i נמלא את התא $A[v_j.parent, v_i]$ לפני שנמלא את התא $A[v_i, v_j]$ וממילוי באופן סימטרי אז גם $A[v_i, v_j.parent]$ ימולא לפני $A[v_i, v_j]$ ויכיל את התוצאה הנכונה מהנחת האינדוקציה.

הערה כללית: מספיק ללכת רק לאבא של v_j ללא האבא של v_i רק בגלל שמיינו טופולוגית ולכן אנחנו יודעים שאם אחד הוא האבא של השני אז v_i הוא האבא של v_j ולא הפוך.

<u>שלב השאילתה:</u>

בהינתן u,v נחזיר $A[u,v]$ וזה ייקח $O(1)$, כנדרש. הסיבה לכך ש $A[u,v]$ יכיל את ה־ LCA של u,v מוסברת בשלב העיבוד המוקדם.

<u>שטח:</u>

אנחנו שומרים את העץ שלוקח $O(n)$ שטח ואת המערך A שלוקח $O(n^2)$ ולכן כמות השטח בכללי היא $= O(n^2+n)$ $O(n^2)$.

<u>מסקנה:</u> כלומר הראנו פתרון לבעיית LCA ביעילות $(O(n^2), O(1), O(n^2))$, כנדרש.

מ.ש.ל.☺