פתרון תרגיל מספר 2 ־ פתרון בעיות באלגוריתמים

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

3 בדצמבר 2020

1. **הערה:** את הרעיון הכללי של הבונוס פתרתי ביחד עם עידן אוריזן.
צ"ל: הטענה לא נכונה ואלגוריתם בסיסי לבעיה
הוכחה:
תחילה נראה שהטענה לא נכונה:
נסתכל על הנקודות $(10,5)$, $(3,4)$,
הפתרון החמדן יילך קודם 3 בציר ה־ x ואז 5 בציר ה־ y ויחזיר פתרון של 8.
נשים לב שיש פתרון יותר טוב שהוא לזוז 4 בציר ה־ y ואז עוד 1 בציר ה־ y ויחזיר פתרון של 5.
כלומר הפתרון החמדן לא בהכרח יחזיר פתרון אופטימלי.

הצעה לפתרון נאיבי:
אבחנה: בהינתן פתרון אופטימלי $p_1,\dots,p_m$ כך ש־ $p_{i_j},\dots,p_{i_{j+1}-1}$ הוא רצף הפעולות להגעה לנקודה (x_j,y_j) אזי קיים פתרון אופטימלי אחר $q_1,\dots,q_m$ המקיים $q_k=q_l$ לכל $k,l\in[i_j,i_{j+1}-1]$ לכל $1\le j\le n$. במילים פשוטות, קיים פתרון אופטימלי שבמעבר מכל נקודה r לבאה זז רק על אחד הצירים.
נוכיח את הטענה באינדוקציה על r:
<u>בסיס:</u> $r=0$, הטענה נכונה באופן ריק
<u>צעד:</u> נניח שהטענה נכונה ל־ $r-1$ ונוכיח ל־ r.
מהנחת האינדוקציה, קיים פתרון אופטימלי שבמעבר מכל נקודה j לבאה זז רק על אחד הצירים לכל $1\le j<r$ שנסמנו $p_1,\dots,p_m$ כך ש־ $p_{i_j},\dots,p_{i_{j+1}-1}$ הוא רצף הפעולות להגעה לנקודה (x_j,y_j) מהנקודה הקודמת וגם מתקיים $q_k=q_l$ לכל $k,l\in[i_j,i_{j+1}-1]$ לכל $1\le j<r$,
בשביל להגיע לנקודה ה־ r מהנקודה $r-1$ נעשו הפעולות $p_{i_r},\dots,p_{i_{r+1}-1}$,
נשים לב שמאופטימליות הפתרון, לא נעשתה פעולה והפוכה שלה (ימינה ואז שמאלה לדוגמא)
נניח בלי הגבלת הכלליות שהפעולות שנעשו הן $\uparrow,\rightarrow$.
נסמן שנעשו a פעולות $\rightarrow$ ו־ b פעולות של $\uparrow$, כלומר מאסוציאטיביות ניתן לכתוב את הפתרון בתור $\rightarrow a,\uparrow b$.
נניח בלי הגבלת הכלליות שהפעולות שנעשו בשביל להגיע לנקודה (x_r,y_r) הן $\rightarrow a$ (הערה: בגלל שבסוף מגיעים לשוויון באחד הקורדינטות אז a פעולות $\rightarrow$ או b פעולות $\uparrow$ מביאים אותנו בדיוק ל־ (x_r,y_r)).
נסתכל על הפתרון $p_1,\dots p_{i_r-1},\rightarrow a,\uparrow b,p_{i_{r+1}},\dots,p_m$, נשים לב שאורכו בדיוק כמו האופטימלי וגם מגיע לכל הנקודות וגם מתקיים כי $q_k=q_l$ לכל $k,l\in[i_j,i_{j+1}-1]$ לכל $1\le j<r$ וגם $q_k=\rightarrow=q_l$ לכל $k,l\in[i_r,i_r+a-1]$.
כלומר מצאנו פתרון $p_1,\dots p_{i_r-1},\rightarrow a,\uparrow b,p_{i_{r+1}},\dots,p_m$ המקיים את הנדרש.

לאחר אבחנה זאת, נשים לב שמספיק לזוז בין כל נקודה לבאה רק באחד הצירים, שמשאיר 2^n מסלולים לבדוק.
ההצעה לפתרון הנאיבי:

(א) לכל מסלול שעובר בנקודות כשכל פעם זזים באחד הצירים בלבד, נחשב את אורך המסלול ונעדכן את המינימלי אם המסלול שמצאנו יותר קטן מהמינימלי.

(ב) נחזיר את המסלול המינימלי שמצאנו

לפי הלמה שהוכחנו, האלגוריתם יחזיר את הפתרון האופטימלי וזמן ריצתו הוא $O(2^n)$ ועם שטח $O(n)$ בעזרת DFS.

את הרעיון הכללי של הבונוס פתרתי ביחד עם עידן אוריזן.
בונוס ראשון:

אבחנה1: נשים לב שאם כל הנקודות מוכלות בריבוע שמכיל $O(n)$ נקודות אז צלעו של השורש הוא $O(\sqrt{n})$.
אבחנה2: בשביל להגיע לנקודה (x_i, y_i) צריך להתקיים (x_i, y) או (x, y_i), נניח ש־ (x, y_i) אז נשים לב שבפתרון האופטימלי יהיה $x = 0$ או x בריבוע, כלומר יש לכל היותר $2\sqrt{n}$ אפשרויות לכל נקודה בעזרת אבחנה 1 איך להגיע אליה.
אלגוריתם:

(א) תחילה נמצא את הריבוע שכל הנקודות מוכלות בו (פשוט לשמור מינימום ומקסימום של נקודות)

(ב) נאתחל מערך A בגודל $O(n \cdot \sqrt{n})$ שישמור לכל נקודה את הדרך האופטימלית להגיע אליה

(ג) נעדכן $A[0, (0,0)] = 0$

(ד) לכל $1 \leq i \leq n$ בסדר עולה:

i. לכל דרך הגעה $p = (x, y)$ לנקודה: (יש $2\sqrt{n}$ כאלה לפי אבחנה1 + אבחנה2):

א'. נעדכן

$$A[i,p] = \min\left\{A[i-1,(x_{i-1},y)] + |x_i - x_{i-1}|, A[i-1,(x,y_{i-1})] + |y_i - y_{i-1}|\right\}$$

ב'. הערה: $x_0 = 0 = y_0$

(ה) נחזיר $\min_p A[n,p]$

תחילה נשים לב שהאלגוריתם רץ בזמן $O(n \cdot \sqrt{n})$ כי בכל לולאה עושים $O(1)$ חישובים עם $O(n \cdot \sqrt{n})$ שטח.
עתה הסיבה שהאלגוריתם נכון, היא לפי האבחנה שהייתה בהתחלה, שקיים פתרון אופטימלי שבין כל נקודה לבאה זז באחד הצירים, ו־ $A[i,p]$ שומר את הדרך המינימלית להגיע ל־ (x_i, y_i) דרך הנקודה p כשכל פעם זזים רק על אחד הצירים וניתן להוכיח טענה זאת באינדוקציה.
לאחר שטענה זאת נכונה, אנחנו יודעים את כל הדרכים האופטימליות להגיע לנקודה ה־ n לכל נקודת סיום אפשרית p ונחזיר את המינימלי בינהם שזה בדיוק מה שהאלגוריתם עושה.

בונוס שני זמן ריצה לא מוצלח:
אבחנה1: נגדיר $x_0 = y_0 = 0$, נשים לב שמהאבחנה שקיים פתרון אופטימלי שבין 2 נקודות זזים רק על אחד הצירים, ניתן להוכיח באינדוקציה שבפתרון אופטימלי זה להגיע לנקודה ה־ k יהיה מקורדיאנטות מהצורה (x_i, y_j) לכל $0 \leq i, j \leq n$.
עתה נחזור על הפתרון של הבונוס הראשון כששומרים לכל קודקוד $O(n)$ נקודות הגעה, כלומר לנקודה ה־ i נשמור $(x_i, y_j), (y_i, x_j)$ לכל $0 \leq j \leq n$.
הנכונות של הפתרון נשארת כמו בבונוס הראשון וזמן הריצה הוא $O(n \cdot n) = O(n^2)$, לצערי זה לא יותר טוב מ־ n^2 אבל זה קרוב יחסית.

מ.ש.ל.☺

2. **צ"ל:** הצעה למבנה נתונים שעובד.
הוכחה:
תחילה אסביר מה הרעיון ואז אכנס לפרטי המימוש, הרעיון הוא ליצור עץ קטעים אחד בהתחלה, ועץ AVL ריק שישמור את כל העצי קטעים לכל סדרה באורך n.
דבר זה ייאפשר לנו להתחיל עם עץ אחד כי כולם יצביעו לאותו מקום, וכל פעם שנעדכן ערך, נבדוק האם הוא כר בפוינטר של החזרה המתאימה לו ואם לא, ניצור את החזרה המתאימה לו רק למסלול שצריך לעדכן ונוסיף ל־ AVL.

עתה ננסה להסביר זאת פורמלית.
<u>אתחול:</u> ניצור מבנה נתונים שהוא עץ קטעים של $a_1, \ldots, a_n$ ובכל קודקוד בעץ שנוצר, במקום לשמור רק פוינטר, ונשמור לכל קודקוד ערך $coppied = False$, זה לוקח $O(n)$ זמן ריצה ושטח כי אנחנו מוסיפים עוד $O(1)$ עבודה לכל קודקוד.
בנוסף לזה נאתחל עץ AVL עם ערך יחיד $(-1, \text{pointer to the created tree})$ ונכניס תמיד לפי האיבר הראשון שלוקח $O(1)$ שטח ו־ $O(1)$ זמן ריצה.
<u>שאילתה:</u> נקבל שאילתה מהצורה $[i, j]$ ונצטרך להחזיר את הסכום של a_i עד a_j,

(א) נחלק את $[i, j]$ לקטעים $\left[i, \left(\left\lfloor \frac{i}{n} \right\rfloor + 1\right) \cdot n\right], \left[\left(\left\lfloor \frac{i}{n} \right\rfloor + 1\right) \cdot n + 1, \left(\left\lfloor \frac{i}{n} \right\rfloor + 2\right) \cdot n\right], \ldots, \left[\left\lfloor \frac{j}{n} \right\rfloor \cdot n, j\right]$,

(ב) נמפה כל קטע למספרו בחלוקה ב־n, כלומר $\left\lfloor \frac{i}{n} \right\rfloor, \left\lfloor \frac{i}{n} \right\rfloor + 1 \ldots, \left\lfloor \frac{j}{n} \right\rfloor$ ונעבוד עם מספרים אלו.

(ג) נאתחל מערך A בגודל $O\left(\left\lfloor\frac{i}{n}\right\rfloor - \left\lfloor\frac{j}{n}\right\rfloor\right)$ (חסום על ידי k)

(ד) לכל $\left\lfloor\frac{i}{n}\right\rfloor \le l \le \left\lfloor\frac{j}{n}\right\rfloor$:

i. נחפש בAVL את עץ הפוינטר לעץ הקטעים המתאים ל־ l, אם קיים נשמור אותו ב־ $A\left[l\right]$ המתאים אחרת נשמור ב־ $A\left[l\right]$ את הפוינטר המתאים לערך -1 (שהוסף באתחול).

(ה) עתה, יש לנו מערך של לכל היותר k פוינטרים שכל אחד מהם הוא עץ קטעים ונשאל על הקטע המתאים למספר הסידורי ונחזיר את סכומם

מדוע זה עובד? בהנחה ואכן בAVL שמורים פוינטרים לעץ קטעים של כל $\left[l\cdot n,(l+1)\cdot n-1\right]$ כאשר $1\le l\le k$ וגם בערך -1 שמור העץ קטעים המקורי ובמקרה שהיה שינוי לקטע הרצוי, אז יוסף העץ קטעים המתאים לעץ AVL אז הנכונות נובעת מנכונות עץ הקטעים.

בהנחה והפוינטרים שמורים בעזרת מבנה שתומך בחיפוש בזמן לוגמריתמי כמו עץ AVL, זמן הריצה הוא $O\left(k\cdot\log\left(k\right)\right)$ לשלבים א־ד, בשלב ה' נשים לב שכל הקטעים חוץ לכל היותר 2 הם באורך n ולכן התוצאה תמצא בעץ הקטעים בזמן $O\left(1\right)$ ולכל היותר 2 קטעים ירוצו בזמן $\log\left(n\right)$.

ונקבל זמן ריצה של $O\left(k\cdot\log\left(k\right)+\log\left(n\right)\right)$.

<u>עדכון:</u> נקבל $\left(i,x\right)$ ונעשה את השלבים הבאים:

(א) נחפש בעץ ה־ AVL את הפוינטר המתאים לערך ־ $\left\lfloor\frac{i}{n}\right\rfloor$, אם קיים נשתמש בו, אחרת נשתמש בפוינטר של הערך -1

(ב) נרד בעץ לקודקוד $i \mod n$ ונשמור את המסלול לקודקוד הזה ב־ $v_1\ldots,v_l$ כש־ v_1 זה השורש

(ג) לכל $1\le i\le l$ בסדר עולה: (באיטרציה זאת אנחנו יוצרים העתק של המסלול שאותו אנחנו רוצים לעדכן כדי לא להרוס את המסלול המקורי)

i. אם $v_i.coppied=False$, ניצור העתק שלו $\overline{v_i}$ ונשנה $\overline{v_i}.coppied=True$, אחרת נגדיר $\overline{v_i}=v_i$

ii. אם $i\ne 1$, נעדכן $\overline{v_{i-1}}.child=\overline{v_i}$ (במקום המתאים לירידה בעץ)

(ד) אם העתקנו את v_1 נכניס לעץ ה־ AVL את הערך $\left(\left\lfloor\frac{i}{n}\right\rfloor,\overline{v_1}\right)$

(ה) נריץ את האלגוריתם של עדכון עץ הקטעים לעדכון הערך $i \mod n$ מ־ $\overline{v_1}$

עקרונית האלגוריתם פשוט מעתיק את המסלול שעלול להיות מושפע מהשינוי שעץ הקטעים ייגרום ושומר אותו באופן ייחודי לקטע $\left[\left\lfloor\frac{i}{n}\right\rfloor\cdot n,\left\lfloor\frac{i}{n}\right\rfloor+n-1\right]$ אם הוא עוד לא קיים.

מנכונות עץ הקטעים המקורי והעובדה שאנחנו רק מעתיקים, האלגוריתם ישאיר לנו עץ קטעים תקין לקטע $\left[\left\lfloor\frac{i}{n}\right\rfloor\cdot n,\left\lfloor\frac{i}{n}\right\rfloor+n-1\right]$ ויוסיף אותו ל־ AVL אם צריך

זמן הריצה הוא $O\left(\log\left(n\right)+\log\left(k\right)+\log\left(n\right)\right)=O\left(\log\left(n\right)+\log\left(k\right)\right)$, כי להעתיק מסלול לוקח $O\left(\log\left(n\right)\right)$, להוסיף לעץ AVL שיכיל לכל היותר k כניסות לוקח $O\left(\log\left(k\right)\right)$ והרצה של עדכון בעץ קטעים לוקחת $O\left(\log\left(n\right)\right)$.

כלומר מצאנו מבנה נתונים שאתחול לוקח $O\left(n\right)$, שאילתה לוקחת $O\left(k\cdot\log\left(k\right)+\log\left(n\right)\right)\le O\left(\left(k+\log\left(n\right)\right)\log\left(k\right)\right)$ ועדכון לוקח $O\left(\log\left(k\right)+\log\left(n\right)\right)\le O\left(\log\left(n\right)\cdot\log\left(k\right)\right)$

מ.ש.ל.☺