תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

10/10

1 **א**

נפרק את המטריצה ל- $\lceil \frac{n}{m} \rceil$ מטריצות בגודל $m \times m$
(ייתכן שהאחרונה קטנה יותר), בכ"א נריץ SMAWK, כי כל מטריצה
עדיין totally mono. סה"כ זמן ~~[illegible]~~ $O(m(\frac{n}{m}+1)) = O(n+m)$
~~[illegible]~~ (המונוטוניות)

ב

ניקח מהמטריצה את השורות הבאות: $0, \frac{n}{m}, \frac{2n}{m}, \ldots$
וניצור מטריצה חדשה בגודל $m \times m$. מטריצה זו עדיין
tot. mono., נריץ עליה SMAWK ונקבל מקס' בכל שורה כזו –
נסמן ב- $i(\max(r_k))$, כעת ידוע כי בין
r_k ל- r_{k+1} המקס' בכל שורה (הנ"ל) בין העמודות
$i(\max(r_k)) : i(\max(r_{k+1}))$

r_0, r_1

מכיוון שאנחנו במטריצה tot. mono, לכל תת מטריצה כזו
(בין השורות r_k ל- r_{k+1} והעמודות הנ"ל) נמצא מקס' בכל שורה
בשיטת divide & conquer (בסיבוכיות $m' \log n'$ עבור מטריצה בגודל $n' \times m'$)

אם אינדקסים [illegible]

<u>סיבוכיות</u>: (1) SMAWK תחילה ב- $O(m+m) = O(m)$

(2) שאר התהליך ב- $\underbrace{\left(\sum_k i(\max(r_k)) - i(\max(r_{k-1}))\right)}_{\le m} \cdot \log \frac{n}{m}$ (divide and conquer)

לכן סה"כ $O((1+\log \frac{n}{m}) \cdot m)$ כנדרש.

כי $i(\max(r_k))$ סדרה עולה מונוטונית.

2

<u>הרעיון</u>: נבנה מטריצה כך שבחלק המשולש העליון שלה, היא
תקיים את התכונה $s_{ij} =$ סכום האיברים מ- a_i ל- a_j ובחלק המשולש
התחתון שלה יהיה ערך שלילי מאוד, נריץ עליה SMAWK
ונחזיר את המקס' הכי גדול מבין אלו ש-SMAWK החזיר.
<u>סיבוכיות</u>: רק משתמשים ב-SMAWK (הגדרת המטריצה לא מפורשת
כפי שיתגלה בהמשך) ולכן הזמן לינארי.

המשך בדף הבא

המשך תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

(2) המשך

10/10

אופן בניית המטריצה בפועל - נחשב רק את השורה הראשונה במטריצה S באופן מפורש (בזמן לינארי) — בזמן S_{1j} זה הסכום מהאיבר הראשון לאיבר ה-j.

כמו כן נחשב את הערך הבא:

סכום המספרים השליליים הנתונים ~~[illegible]~~ פחות $2n$ $\quad X = \sum_{a_i<0} a_i - 2n$

לפחות n

X קטן ממש מ-S_{ij} בחלק המשולש העליון (ה-S_{ij} הכי שלילי מייצג במקרה הגרוע רצף של כל השליליים הנתונים, ו-X קטן ממש מערך זה)

כאשר SMAWK יבקש את S_{ij} נחזיר את הערך הבא:

בחלק המשולש העליון נחזיר את $S_{1j}-S_{1i}$

" התחתון " " $X+j-i$ (ערכים אלו קטנים מהמשולש העליון ולא יוחזרו כמקס')

ניתן להראות ש-S הוא totally mono-, כלומר $S_{ik}<S_{il} \to S_{kj}<S_{lj}$ עבור $j>i$, $l>k$. נחלק למקרים:

א. כל האיברים בחלק המשולש העליון — $S_{ik}<S_{il} \Rightarrow$ ~~[illegible]~~

$\Leftarrow S_{lj} = S_{kj} + \underbrace{S_{k+1,l}}_{>0} > S_{kj}$

ב. איבר אחד בחלק התחתון - זהו S_{kj} - מהגדרת החלק התחתון שהראינו $S_{lj}<S_{kj}$

ג. שני איברים בחלק התחתון - S_{kj}, S_{ki} - כמו ב', $S_{lj}<S_{kj}$

אם S_{lj}, S_{kj} שניהם בחלק התחתון, $\quad S_{kj} = X+k-j$

אז $S_{kj}>S_{lj}$ כי $l>k$ $\quad S_{lj} = X+l-k$

ד. רק S_{il} בחלק העליון - כמו ג', $S_{kj}>S_{lj}$

ה. כולם בחלק התחתון - גם כן כמו ג'.

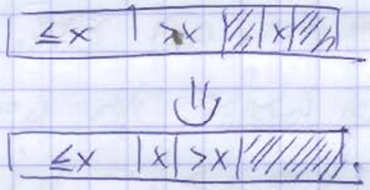
לכן בכל מקרה אפשר לעשות SMAWK.

המשך בדף הבא ←

המשך תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

10/10

(3) נזכר באופן פעולת מיון הכנסה: כל שלב לוקח איבר x במערך ו"מכניס" אותו:



לאורך כל המערך הממוין. זמן הריצה $O(n+d)$ כאשר n אורך המערך ו-d מספר ה"היפוכים" (Inversions) שצריך לעשות (באופן כללי יכול להגיע ל-$O(n^2)$. אבל לא - נסביר בהמשך למה)

אופן הפעולה: את השורה הראשונה פשוט נמיין ($O(n)$) $O(m)$

כל שורה הבאה נמיין כאשר סדר ההכנסה (תבנית מיון) (וגם הרצף!) נקבע כך: את האינדקס המקורי של האיבר בשורה r_k, שאחרי מיון מגיע למקום j בשורה. אז בשורה r_{k+1} נכניס את

$r_{k+1}[i(r_k,1)], r_{k+1}[i(r_k,2)], \ldots$

נכונות: מיון הכנסה בכל שורה עובד בכל סדר הכנסה, ובפרט זה שבחרנו.

זמן ריצה: נראה כי עבור כל ההחלפות שמבצעים d, $\sum_{k} d = O(m^2)$

ואז סה"כ זמן ריצה $\sum_{k=1} O(n+d_k)=O(mn+m^2)$

החלפה מתבצעת רק במצב בו $A[i-1,j_1] > A[i-1,j_2]$ וגם $A[i,j_2] < A[i,j_1]$ עבור $j_2 < j_1$ כלשהם. נראה כי מצב זה קורה לכל היותר פעם אחת לכל זוג עמודות j_1, j_2. בשלילה, יש עוד שורה כזו i'. אם $i < i'$,

$A[i',j_1] < A[i',j_2] \Leftarrow A[i-1,j_1] < A[i-1,j_2]$. אם $i' < i$,

$A[i,j_1] < A[i,j_2] \Leftarrow A[i',j_1] < A[i',j_2]$

כאשר המערכים נתונים מתכונת totally monotone, ובכל מקרה הגענו לסתירה.

לכן יתכנו החלפות רק כמספר הזוגות של העמודות $O(m^2)$

וסה"כ הסיבוכיות לפי החישוב קודם $O(mn+m^2)$

המשך תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

(4) ט

אינווריאנטה 1: $c \le r$ ← (זו האות ר)

הוכחה: בהתחלה $0=c=r$. בכל צעד, r ו-c משתנים כפול הקוד שמופיע בשאלה. אם הם משתנים בחלק ה-if, ברור שמתקיים $j=c>r=j-1$, אם הם משתנים בחלק ה-else, נזכור כי ה-SMAWK מחושב על תת המטריצה המוגדרת $r:c$, $c+1:p$, כלומר $r<c<c+1<p$ (אפשר לראות את זה אם מגדירים p שהוא $p=2c-r+1$ כדי לקיים את הגדרת האינדוקציה). כלומר $p>r$.

השורה שבה מחושב F(p) היא "שורה" חדשה מ-p (המטריצה משתנה בגודל)

אינווריאנטה 2: $f(j)=F(j)$ לכל $1 \le j \le c$

אינווריאנטה 3: לכל $j>c$, $f(j)=\min(N(j), \min_{i \le r} G(i,j))$

בתחילת האיטרציה הראשונה — אינו' 2 מתקיימת באופן ריק (העמודות מתחילות ב-1 ו-$c=0$), אינו' 3 מתקיימת מהגדרת f ו-G ($N(j)=\infty$).

שלב האינדוקציה - נעבור את מהלך הלולאה ונראה כי מתקיימות אינו' 2 ו-3. לצורך כך נוכיח אינווריאנטות נוספות של לולאת ה-for הפנימית (for j:=c+2 to p do, if...):

אינווריאנטה I: המקס' של העמודות $q:j$ אינו מתקיים בתת המטריצה $G[c+1:j, r:j]$ (בהתחלה עבור $j=c+2$ מתקיים באופן ריק)

אינווריאנטה II: יודעים את $G[j-1, h:j]$ (בהתחלה מתקיים לפי אינו' 2).

מקרה א': $G[j-1,j]<F[j]$: במקרה זה מתחילים איטרציה חדשה של הלולאה, החדשים הם $c=j$, ו-$r=j$. לכן אינו' 2 מתקיימת נכונה בהמשך שמקדמים את j. כך כשמקדמים את f(j). (עבור הקודמים עבור j הנוכחי, מאינו' 3 לעמודה j הנו ב-G כשר ...

(המשך בדף הבא ⇐)

אר

האינווריאנטה נכונה בתחילת כל איטרציה

10/10

המשך תשובה לשאלה 1 בנושאים מתקדמים באלגוריתמים

המשך

4 a או ב-N, מאינו' I המינ' הוא במטריצה שעשינו עליה SMAWK
או כמו הנוכחי, והמשוואה של מקרה ג' המינ' של כל העמודה j אכן
ב-$G[j-1,j]$ — הוא הנוכחי. כל שאר השורות (נפסלו).
המשך מקרה א', למה אינו' 3 ממשיכה להתקיים — בקצרה צריך
למה עבור העמודות $j+1$ והלאה המינ' הוא לא בשורות שאינו
פוסלים וכו' עד $j-2$. הדבר נובע מתנאי מקרה א' $F[j]<G[j-1,j]$
ומתכונת ה-totally monot. של מטריצה G.
מקרה ג', למה אינו' I+II ממשיכות להתקיים — לא צריך להראות
כי אין עוד אינדקציות של הטבלה הפנימית לכן שייכות אינו' אלו.
מקרה ב' $F[j] \le G[j-1,j]$ אבל $G[j-1,p] < F[p]$ —
אינו' 2 — עם עמודה $j-1$ מהנחת האינדוקציה + מקרה ב'. בעמודה
j עצמה (כאמור באינו' I "מת" (אין בו מינימום) לכן ההשוואה
$F[j] \le G[j-1,j]$ מספיקה כדי להראות ש- $f(j)=F(j)$
אינו' 3 — כל שעבר השורות r עד $j-2$ ~~[illegible]~~
המינ' של עמודה $j+1$ והלאה לא יכן. עבור העמודות עד p השורות
עד $j-1$ אולי המינ' שם לכן מאפסים ב-N. עבור שאר המקרים כאינו'
שאינו' לא שם מהנחת האינדוקציה לאינו' I + תכונת monot. tot. של G.
אינו' I + II — לא צריך להראות כי אין עוד אינדקציות של הטבלה הפנימית
מקרה ג' $F[j] \le G[j-1,j]$ וגם $F[p] \le G[j-1,p]$ —
אינו' 2 — מקפיצים את j רק ב-1 לכן צריך להראות (שמוצאים את) עבור j כה. נכון.
מאינו' I המינ' לא בשורה r עד $j-2$ לכן מההשוואה של מקרה ג' $f(j)=F(j)$
אינו' 3 — לא צריך להראות כי לא מתחילים אינדקציה חדשה של הטבלה הגדולה.
אינו' I — מתכונת totally monotone של $G[j-1, j \ldots p]$ גם בשורה $j-1$
בעמודות קטן לא יתקבלו מינימום, לכן אינו' זו ממשיכה להתקיים.
אינו' II — יודעים את $f(j)$ ולכן את $G[j, j+1 \ldots]$ ולכן האינו' ממשיכה להתקיים.

המשך בדף הבא ←

המשך תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

(4) ב

כל איטרציה לוקחת $\underline{O(r-l)}$:

– האיטרציה ה-i מריצה SMAWK בגודל $(r-l)\times(r-l)$

– הלולאה הנוספת רצה לכל היותר $r-l+1$ איטרציות.

כדי לקבוע כמה זמן לוקח כל האלגוריתם "נשייך" כל איטרציה לשורה או עמודה. __במקרה א'__ r גדול כך ש- $r-l \geq r-(l-j)$ לכן "נשייך" את זמן האיטרציה לשורות $l-1,\dots,r$

__במקרה ב__ אם $m<p$, נניח כי c גדול כך ש- $p-c \geq c-l$ ונשייך את הזמן לעמודות $1-p,\dots,c$, אם $m=p$ נשייך את הזמן לשורות $1-c,\dots,r$ (כי האיטרציה האחרונה במקרה זה)

__בסה"כ__ c ו-r לא קטנים, לכן כל שורה ועמודה "תשויך" רק מספר קבוע של פעמים.

$\Rightarrow$ זמן הריצה של האלגוריתם $\boxed{O(n+m)}$.

(5) א

__הרעיון__ - נבנה 2D range tree, כאשר העץ הראשי לפי x והעץ המשני לפי y. לכל צומת בעץ משני נשמור את הנקודה עם ערך z מקסימלי בתת העץ שלה.

__זמן הבנייה__ — לאחר בניית המבנה (עובר שוב על כל הצמתים במשניים כדי לחשב מקס' z בכל צומת, כל צומת לוקח מקס' מבין שני הבנים (בנוסף לערך שלו, מתחתיו העץ ריק לעלה), סה"כ עוברים פעם אחת על כל צומת במבנה לכן לא הוספנו זמן ריצה לבנייה 2D Range Tree רגיל - $O(n\log n)$

__מקום__ - הוספנו רק $O(1)$ לכל צומת לכל קו במבנה (מקס' z בצמתים משניים) לכן נשאר רגיל $O(n\log n)$

__חיפוש__ - כמו 2D Range Tree, כאשר מגיעים לנקודה שהיא בתוך המלבן x,y ~~[illegible]~~ (נקודה בעץ משני) נעצור שם ונשווה את ערך z שלה למקס' עד כה.

המשך בדף הבא ⇐

10/10 ✓

המשך תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

5 ב המשך

בתום החיפוש יהיה לנו ערך z מקסימלי ונקודה בעץ משני שמתאימה לערך זה (שאין דומה לה)
מכאן זו אפשר לרדת בעץ המשני לעלה כדי לקחת עבור עצא ספציפית.

נכונות - החיפוש יביא אותנו לכן. תתי-העצים המשניים בטווח כמו חיפוש רגיל ב- 2D Range Tree ולכן הגענו בדיוק לכל הנקודות בטווח (ומכוונות ל"אבא" שלהן בעץ).
מכיוון שכל צומת בעץ משני מכיל מקס z של צאצאיו ושני נשמור ערך z מקס' בהכרח נגיע לערך z מקס' בטווח.

זמן חיפוש - לא הוספנו זמן על חיפוש רגיל במבנה זה - $O(\log^2 n)$ (בדומה לשאילתה ב- 2D Range Tree רגיל רגילים.) (שלב $\log^2 n$ א-יתר שעלות)

5 ב

נבנה 1D Range Tree לפי y, כל צומת תכיל רשימה ממוינת לפי x של כל נקודות בתת-העץ של הצומת.

בנייה - נבנה עץ 1D Range Tree ב- $O(n\log n)$ וכל פיצול בבניה לתת-עץ ימני ושמאלי נשמור על סדר הנקודות ממוינות לפי x (מיון ב- $O(n\log n)$ לפי x מראש). בצורה כזו לא צריך למיין כל פעם לפי x. סה"כ זמן

$$O(n\log n) + \sum_{v} O(n_v) = O(n\log n) + O(n\log n) = \boxed{O(n\log n)}$$

גודל המבנה - העץ הראשי תופס $O(n)$ מקום וכל המשניים $\sum O(n_v) = O(n\log n)$ מקום, סה"כ $O(n\log n)$

אופן החיפוש - עבור כל שאילתה שמקבלים אנו מחפשים בעץ y ונלך לרשימת x שלו (הממוינת לפי הסדר) ונחזיר נקודות כל עוד קטנות מ- α.

נכונות - ב-y מתוך נכונות של 1D Range Tree.
ב-x, בשאלה לא החזרנו נקודה שאינה בטווח, יש לנו y (לא y) כך ש- $\alpha > y$ ⇐ סתירה. בשאלה החזרנו רק מה שמתאים - אבל $\alpha \leq \varepsilon$

המשך דף הבא ←

המשך תשובות לתרגיל 1 בנושאים מתקדמים באלגוריתמים

3ב המשך

אחרת היינו פוגעים אותה בהשוואה ⇐ סתירה.

<u>זמן שאילתה</u> – סה"כ זמן השאילתה $O(\log n + \sum_i k_i)$

כאשר k_i מס' הנקודות שהחזרנו ~~[illegible]~~ מהרשימה של צומת i. $\log n$ נובע מהחיפוש בעץ y.

ס איבר שמחזירים מופיע ב־$O(1)$ רשימות*, לכן $\sum_i k_i = O(k)$

k – כמות נקודות שיש להחזיר בתשובה, לכן סה"כ $O(\log n + k)$

* רשימות שהחיפוש בעץ y גורם לנו להיכנס אליהן.

5א

~~[illegible]~~

<u>הערה</u>

כדי להימנע ממצב בו יש הרבה נקודות עם אותם ערכי x,y. (עם z שונים) אפשר לבצע את העיבוד המקדים הבא ב־$O(n\log n)$:

נמיין את ט הנקודות לקסיקוגרפית (קודם לפי x, אח"כ y, אח"כ z) ואז נעבור עליהן ונמחק נקודות עם ערכי ~~[illegible]~~ z נמוכים מערכי z של נקודות אחרות עם אותם ערכי xy.