נושאים מתקדמים באלגוריתמים - תרגיל 1

1. a. $n \geq m$, $A \in R^{n\times m}$, A היא totally monotone

10/10

אלגוריתם

1. "נחלק" את המטריצה A ל $\lceil \frac{n}{m} \rceil$ מטריצות:

$$B_1 = A[1:m, 1:m], \ldots, B_{\lceil \frac{n}{m} \rceil} = A[(\lceil \frac{n}{m} \rceil - 1)\cdot m + 1 : n, 1:m]$$

2. נפעיל על כל אחת מתת המטריצות SMAWK

נכונות:

1. כל תת מטריצה היא גם totally monotone (הראינו בשיעור).
2. כל תת מטריצה היא בגודל $m\times m$ חוץ מהאחרונה שהיא בגודל $x\times m$ $m \geq x$ ולכן ה SMAWK עובד ונחזיר תשובות נכונות
3. תתי המטריצות מכסות את כל A ולכן בסוף יש לנו אינדקס שמוצא מקסימום בכל שורה ב A.

סיבוכיות:

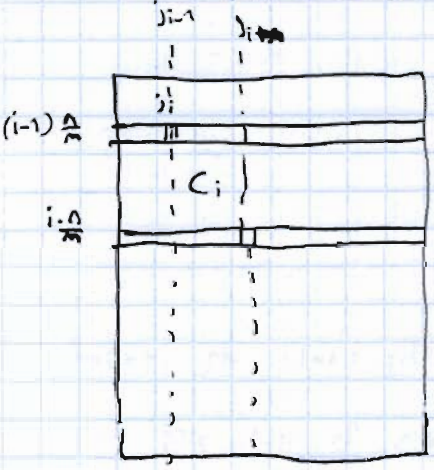
יש $\lceil \frac{n}{m} \rceil$ תתי מטריצות, $\lceil \frac{n}{m} \rceil \leq \frac{n}{m}+1$. ~~[illegible]~~ סיבוכיות SMAWK על כל תת מטריצה $O(m) \Rightarrow O(m\cdot(\frac{n}{m}+1)) = O(m+n)$

1. ט. אלגוריתם:

1. נפעיל SMAWK על מטריצה בגודל $m\times m$, כאשר השורות שלה יהיו $A[\frac{n}{m},1:m], A[\frac{2n}{m},1:m], \ldots, A[n,1:m]$ ונקבל $j_1,\ldots,j_m$ עמודות של מקסימום.
2. "נשאר" לנו m תת מטריצות (שבין השורות שהרצנו עליהן SMAWK) C_i, $1\le i\le m$ $C_i \leftarrow A[(i-1)\frac{n}{m}+1 : i\cdot\frac{n}{m}-1]$ $[j_{i-1}:j_i]$, נפעיל עליהן Divide & Conquer שראינו בכיתה.
3. נאחד את רשימות המקסימום ונחזיר אותן.

נכונות:

1. מנכונות SMAWK אלו המקסימום של השורות $\frac{n}{m},\frac{2n}{m},\ldots$ נכונים והן בעמודות $j_1,\ldots,j_m$
2. מ-1 ומונוטוניות המטריצה A: בשורות $[(i-1)\frac{n}{m}+1 : i\cdot\frac{n}{m}-1]$ המקסימום נמצאים בין $j_{i-1}:j_i$ (כאשר $j_0=1$)



3. ולכן מנכונות Divide & Conquer נמצא את המקסימום של כל C_i שהם המקסימום של כל השורות ב-A פרט ל-2.
4. איחוד הרשימות של כל C_i והרשימה של ה-SMAWK ייתן לנו את מיקומי כל המקסימום
5. הבהרה: רשימת מקסימה תהיה בצורה <column, startRow, endRow> ואיחוד שתי רשימות כמובן יאחד בין שני הייצוג column אחד אם הם בהפרש (בגלל המונוטוניות ⇐ עמודה אחת בשורות בהן המקסימום יהיה בעמודה הזו הן רצופות).

סיבוכיות:

1. SMAWK לוקח $O(m)$
2. D&C על מטריצה C_i לוקח $O((j_i-j_{i-1})\log\frac{n}{m})$

אבל $\sum_{i=1}^{m}(j_i-j_{i-1})=m \Leftarrow O(m\log\frac{n}{m}) \Leftarrow$ סה"כ $O(m(1+\log\frac{n}{m}))$

2. נתון $a_1,\dots,a_n$

10/10

אלגוריתם:

1. נעבור על כל האיברים $a_1,\dots,a_n$ ונחשב:

$S_i=\sum_{1}^{i} a_k$ (ע"י הוספת a_i ל S_{i-1}).

כל איבר שלילי $a_k<0$ נוסיף ל MIN [redacted]

2. נפעיל SMAWK, כאשר האלגוריתם יבקש איבר במטריצה A_{ij}:

אם $i\le j$ $\quad A_{ij}=S_j-S_i$

אחרת $\quad A_{ij}=MIN+j-j$

3. נעבור על n המקסימומים שהחזיר ה SMAWK ונחזיר את הגדול ביניהם.

נכונות:

נשים לב שהמטריצה עליה עובד ה SMAWK נראית כך:

S_{11}	S_{12}	S_{13}		
	S_{22}	S_{23}		
		S_{33}		
			S_{44}	
				$\ddots$

כאשר S_{ij} לא מוגדר עבור נזין ולכן "נמלא" את המטריצה בערכים $MIN+j-i$ מתחת לאלכסון הראשי. נשים לב שכל ערך מתחת לאלכסון הראשי קטן מכל הערכים במשולש העליון.

נראה שהמטריצה היא totally monotone, $A_{ik}<A_{il}\to A_{jk}<A_{jl}$

כאשר $i<j$, $k<l$. נחלק ל 5 מקרים:

א. ארבעת האיברים במשולש העליון: $j\le k$, $j\le l$, $i\le l$, $i\le k$

$$A_{ik}+A_{jl}=S_{ik}+S_{jk}=\sum_{\alpha=i}^{k}a_\alpha+\sum_{\beta=j}^{l}a_\beta=\sum_{\alpha=i}^{l}a_\alpha+\sum_{\beta=j}^{k}a_\beta=S_{il}+S_{jk}=A_{il}+A_{jk}$$

סידור מחדש של האיברים

נובע שאי השוויונים של [redacted] האינדקסים.

inv monge → totally monotone

$\Rightarrow A_{ik}+A_{jl}\le A_{il}+A_{jk} \Longrightarrow A_{ik}<A_{il}\to A_{jk}<A_{jl}$

ב. איבר אחד בלבד במשולש התחתון: וכן להיות רק A_{jk} (כי איבר אחד יגרור שגם A_{jk} יהיה). אז $j>k$

תמיד מתקיים ש $A_{jk} < A_{jl}$ כי $A_{jk} = MIN + k - j < MIN \le A_{jl}$

מינימום קטן שווה מכל האיברים במשולש העליון

ולכן תמיד מתקיימת מונוטוניות טוטאלית: $A_{jk} < A_{jl} \to A_{ik} < A_{il}$

ג. ישנם שני איברים במשולש התחתון:

i) A_{ik} , A_{jk} במשולש התחתון וכמו ב-ב' $A_{jk}<A_{jl}$ תמיד.

ii) A_{jl} , A_{jk}

נראה ש: $A_{jk} < A_{jl}$

$$A_{jk} = MIN + k - j \underset{k<l}{<} MIN + l - j$$

ד. ישנם שלושה איברים במשולש התחתון: רק A_{il} במשולש העליון

כמו ב ג ii) $A_{jk} < A_{jl}$ (שניהם במשולש התחתון).

ה. כל האיברים במשולש התחתון: שוב $A_{jk}<A_{jl}$. □

הראינו שהמטריצה היא totally monotone, עכשיו רק נראה שהמקסימום לא יכול להיות איבר מהמשולש התחתון: כפי שאמרנו כל איבר e במשולש התחתון קטן מכל האיברים במשולש העליון $e < MIN \le$ upper triangle.

עפ"י נכונות SMAWK

ולכן עבור כל שורה i נבחר מקסימום חוקי מבין S_{ij} [illegible] V_j

ועברנו על כל המקסימומים בשורה 3 ובחרנו את הגדול מביניהם □.

<u>סיבוכיות:</u>

1. עברנו על כל האיברים: $O(n)$

2. הרצנו SMAWK על מטריצה תמת כשגודל של אלמנט מיוצג ב $(n-1)\cdot(n)$ $O(1)-O(n)$

3. עברנו על n מקסימומים ובחרנו את הגדול $O(n)$.

3. ## אלגוריתם: $\frac{10}{10}$

1. נאתחל מערך Order() בגודל n בצורה אקראית עם מספרים 1:n.
2. נרוץ על כל השורות ב A:
3. נשים את האיברים של שורה i ב-A בסדר i ב-P
בצורה הבאה: $P[i,1:n] = A[i, order[1:n]]$
4. נמיין את $P[i,1:n]$ ונעדכן את המערך Order:
$Order[j]$ יהיה הטור ב A שבו האיבר (בשורה ה-i)
שגדול מ j-1 איברים בשורה ה-i. כלומר האינדקס k
של האיבר a_{ik} שאחרי מיון יהיה ב $P[i,j]$.
5. נחזיר את P.

נכונות:

1. מיינו כל שורה ב A נעתיק P. $\Leftarrow$ P היא ממויינת בשורות.

סיבוכיות:

1. העתקת כל איבר מ-A ל-P לוקח $O(n\cdot m)$
2. מיון שורה i בעזרת Insertion sort: $O(m + I_i)$

כאשר I זה מספר ההחלפות שמבצעים בין סדר הכנסה
לסדר הסופי של המערך הממוין.

מתבצעת
החלפה ~~[illegible]~~ כאשר יש שני איברים a_1, a_2 ו a_1 מופיע
אחרי a_2 במערך: $\boxed{\quad a_2 \quad a_1}$ אבל $a_1 < a_2$.

נשים לב שמהגדרת מונוטוניות אלכסונית, עבור כל שתי עמודות:
$j_1 < j_2$ ישנה רק שורה אחת כך ש:
$A[i,j_1] > A[i,j_2]$ וגם $A[i+1,j_1] < A[i+1,j_2]$

נניח בשלילה שיש עוד אחת i': אלכסונית מונוטונית

אם $i' < i$ אז $i'+1 \le i$ $A[i,j_1]<A[i,j_2] \Leftarrow A[i'+1,j_1] < A[i'+1,j_2]$

אם $i' > i$ אז $i+1 \le i'$ $A[i',j_1]<A[i',j_2] \Leftarrow A[i+1,j_1] < A[i+1,j_2]$

סתירה ~~[illegible]~~ ולכן ישנה אחת כזו.

- אבל בגלל שאנחנו מכניסים איברים ~~[illegible]~~ בשורה i לפי אותו [illegible]

אז יהיו שתי ~~[illegible]~~ עמודות j_1, j_2 תתבצע החלפה בשורה

i אם $A[i-1,j_1] \neq A[i-1,j_2]$ ובשורה ה-i הכנסנו אותם בסדר

$\boxed{\ \ \Big|\ a_{ij_2}\ \Big|\qquad\qquad\Big|\ a_{ij_1}\Big|}$ אזי $A[i,j_1] < A[i,j_2] \Leftarrow a_{ij_1} < a_{ij_2}$

אזי יהיו לי שני טורים j_1, j_2 תתבצע רק החלפה אחת באלגוריתם

בשורה i. ולכן סה"כ זמן המיון: $\sum_{i=1}^{n} O(m+I_i) =$

$$= O(m\cdot n) + \underbrace{\sum_{i=1}^{n} I_i}_{\text{מספר זוגות העמודות}} = O(m\cdot n) + O(m^2)$$

4. א. נגדיר את האינווריאנטות הבאות:

1. $r \le c$
2. עבור $j \le c$ $F(j) = f(j)$
3. עבור כל עמודה $j > c$ $f(j) = \min(N(j), \min(\{G(i,j) ; i \ge r\}))$

10/10

נוכיח שהאינווריאנטות נשמרות לאורך האלגוריתם:

באינדוקציה:

בסיס: $r = c = 0$

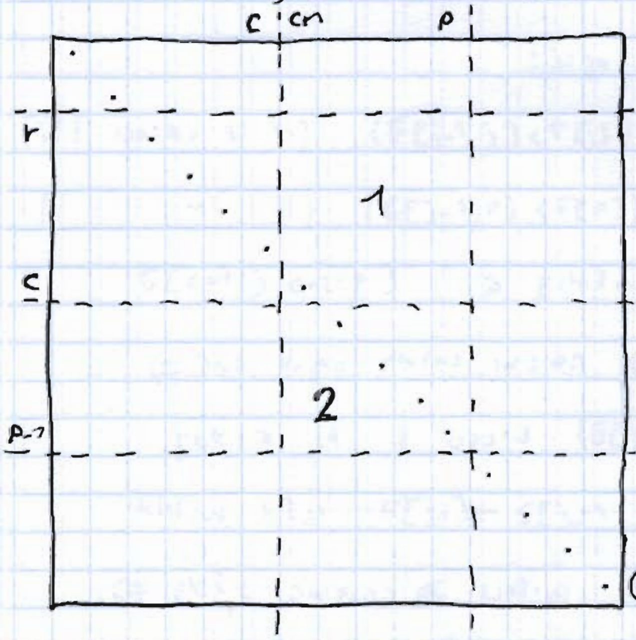
אינו' 1 - מתקיימת

אינו' 2 - מתקיימת באופן ריק

אינו' 3 - מתקיימת, כי על כל עמודה המינימום נמצא באחת השורות שלה.

הנחה: האינו' מתקיימות עד לאיטרציה נוכחית

צעד:



האלגוריתם מריץ SMAWK

על המטריצה '1' ושומר את

המינימות ב- $F[c+1:p]$

עפ"י הנחת האינדוקציה של

אינו' 3, המינימום נמצא

או ב 1 או כל 2 או ב

$N[c+1:p]$ (עבור העמודות $c+1:p$)

אז האלגוריתם מעדכן את $F[c+1:p] \leftarrow \min(N[c+1:p], F[c+1:p])$

ועכשיו המינימות של הטורים $c+2:p$ או ב F או ב 2.

עכשיו האלגוריתם רץ על $j=c+2:p$: נחלק למקרים:

1. $\Leftarrow$ נניח המינימות נמצאות ב $F[c+1:p]$ $\Leftarrow$

$(G[j-1,j] < F[j]) = false$ ולכן ניכנס ל if בראשון

וגם

$(G[j-1,p] < F[p]) = false$ ולכן ניכנס ל if השני.

$\Leftarrow$ נמשיך בלולאה עד $j=p+1$, ~~[illegible]~~,

r ~~= max{r, [illegible] F(p)}~~

⇐ אם המשכנו בלולאה עד $j=p+1$ ⇐ לא נכנסנו ל if

הראשון והשני ⇐ $G[j-1,p] \geq F[p]$ ⇐ עפ"י monge

$G[j-1, j:p] \geq$ מקבלים בשורה של $F[p]$ בעמודות $j:p$ בהתאם

⇐ אין מינימומים חדשים בשורה $j-1$, $c+1 \leq j-1 \leq p-1$.

⇐ המינימומים נמצאים ב $F[c+1:p]$ □

והשמות הבאות יתבצעו: $c' \leftarrow p$, $r' \leftarrow \max(r,$ שורה של $F[p])$

אינו' 1 מתקיימת כי $r' \leq c \leq c'$

אינו' 2 מתקיימת כי לפי הנחת האינדוקציה $F(1:c)=f(1:c)$

והוכחנו עכשיו ש $F(c+1:p)=f(c+1:p)$

אזי $F[1:c'] = f(1:c')$

אינו' 3 מתקיימת כי אם $r' \leftarrow r$ אז מהנחת האינדוקציה

אחרת שורה של $r' \leftarrow F(p)$ ומונוטוניות המינימה

$\forall j > c$ $f(j) = \min\{G(i,j) ; i \geq r\}$

2. $\exists j_0$ כך שיצאנו מהלולאה ב $j=j_0$:

אזי עבור $j<j_0$ $(G[j-1,j]<F[j])=false$ (כי לא נכנסנו ל if)

וגם $(G[j-1,p]<F[p])=false$ (")

* אז לפי 1 המינימום לא נמצאים ב $G[c+1:j, c+2:p]$

אז ⇐ $G[j_0-1,j_0]=f(j_0)$ ⇐ המינימום נמצא בתא הזה ⇐

$(G[j_0-1,j_0]<F[j_0])=true$ ניכנס ל if הראשון

נבצע השמה $F[j_0] \leftarrow G[j_0-1,j_0]$ ונצא מהלולאה.

⇐ נכנסנו ל if הראשון וביצענו את ההשמה ויצאנו ⇐

$(G[j_0-1,j_0]<F[j_0])=true$ ⇐ עפ"י * המינימום של טור

j_0 נמצאים בתא הזה ⇐ $G[j_0-1,j_0]=f(j_0)$. □

$j_0 \leq p$ ולכן ההשמה תתבצע: $r'=j_0-1$, $c'=j_0$

אינו' 1 מתקיימת $r' \leq c'$

אינו' 2 מתקיימת לפי הנחת האינדוקציה: $F[1:c]=f(1:c)$

לפי * : $F[c+1:j]=f(c+1:j)$ $c+1 \leq j < j_0$

וגם א: $F[j_0] \leftarrow G[j_0-1,j_0]=f(j_0)$

אזי $F[1:c']=f(1:c')$

אינו׳ 3 מתקיימת: [illegible] המינימום בעמודה j_0 היה בשורה

j_0-1, ממונוטוניות המטריצה המינימום בעמודות

$j>j_0$ יהיו בשורות $\geq j_0-1$, $c'=j_0$. לכן

$F(j)=\min\{G[i,j]\ ;\ i\geq r=j_0-1,\ j>c=j_0\}$

ב. $\Leftarrow$ $\left.\begin{matrix} G[j_0,p]<F[p] \\ G[j_0-1,j_0]\geq F[j_0] \end{matrix}\right\}$ $\Leftarrow$ נכנסנו ל if השני ולא נכנסנו למקרה א'

$\Leftarrow$ במינימום j_0 המינימום ב $F[j_0]$, עבור עמודות $j_0\leq j<p$

אולם לא יתקיים, המינימום יכול להיות ב $F(j)$ והוא יכול להיות

בתת מטריצה 'Z', ועבור עמודה p המינימום הוא לא $F[p]$

אחרי שורה j_0-2 לפני א'

והוא גם אחרי שורה j_0-2.

$\Rightarrow$ $F[j_0]=f(j_0)$, בשורה j_0-1 עמודה $j_0\leq j\leq p$ $G[j_0-1,j]=f(j)$

$\Leftarrow$ בגלל המטריצה מונוטונית $G[j_0-1,p]<F[p]$ $\Leftarrow$ לא נכנס

ל if הראשון ונכנס ל if השני ונבצע השמה:

$N[j+1:p]\leftarrow F[j+1:p]$, ונקבל מבוקשנו. □

והערכים הבאים יתקבלו $r'=j_0-1$, $c'=j_0$

אינו׳ 1 מתקיימת $r'\leq c'$

אינו׳ 2 מתקיימת מהנחת האינדוקציה: $F[1:c]=f(1:c)$

לפי *: $c+1\leq j\leq j_0$ $F[c+1:j]=f(c+1:j)$

ולפי ב: $F[j_0]=f(j_0)$

אז: $F[1:c']=f(1:c')$

אינו׳ 3 מתקיימת: המינימום בעמודה j_0 היה בשורה

של $F[j_0]$, $N[j_0+1:p]$ מכיל את

המינימומים של 'Z' כולו של העמודות

$j_0+1:p$ עבור השורות $1:j_0-2$ בגלל

ההשמה לא ובגלל * והנחת האינדוקציה ולכן עבור עמודות

$j>c'$ המינימום נמצא או ב N או

ב G בשורות $\geq r'=j_0-1$ □

□

אז: האינווריאנטות נשמרות לאורך האלגוריתם. בין איטרציות.

r או c גדלים בכל איטרציה, אינו׳ 1 $r\leq c$ $\Leftarrow$ c ואז n.

אינו׳ 2: עבור $n=c\geq j$ $\Leftarrow$ $F(j)=f(j)$ $\Leftarrow$ האלגוריתם נכון.

4. 6. כמו באלגוריתם של Wilber שראינו בכיתה, כל

איטרציה לוקחת זמן $O(c-r)$, SMAWK על מטריצה בגודל

$(c-r)\times(c-r)$ ו $p-(c+2)$ איטרציות בלולאה $2c-r+1-(c+2)=O(c-r)$

עכשיו נראה שכל איטרציה מגדילה את r או c ב $c-r$:

אם עצרנו ב j_0 בלולאה $p\ge j_0\ge c+2$: $r'-r=j_0-1-r\ge c-r$

אם לא עצרנו: $c'-c=p-c=2c-r+1-c=c-r+1\ge c-r$

ולכן כמות המידה פוטנציאלית לכמה שקידמנו את r,c.

מתחילים מ $c=r=0$ ומסיימים ב $r'\le c'=n$ ולכן

$r'-r+c'-c=r'+c'\le 2n$ <= סה"כ זמן הריצה $O(n)$.

5. a. <u>בנייה:</u>

1. נבנה 2D-Range tree, כאשר העץ הראשי ובנוי לפי ציר X, והעץ המשני לפי ציר Y.
2. בכל צומת v בעץ הראשי יש עץ משני T_v, ולכל צומת $u \in T_v$ נשמור את הנקודה עם ערך z מקסימלי מבין הנקודות בתת העץ של u.

9/16

<u>חיפוש:</u>

1. בהינתן $Query: [x_1, x_2, y_1, y_2]$ נחפש בדומה לעץ טווח תלת מימדי, רק כאשר אני מגיע בעצים המשניים T_v, כאשר אני מגיע לצומת פנימי $u \in T_v$ שאני יודע שכל צאצאיו בטווח X-Y, במקום להיכנס לעץ משני נוסף אני פשוט בודק את ערך Z של הנקודה השמורה אצלו ומשווה לערך Z של הנקודה ששמורה אצלי אם הוא יותר גדול אני שומר את הנקודה החדשה.
2. בסיום החיפוש מחזירים את הנקודה השמורה אצלי.

<u>נכונות:</u>

1. הנקודה שאני מחזיר בטווח השאילתה:

מנכונות חיפוש 3D-RangeTree ~~...~~ באלגוריתם נכנס לצמתים v בעץ הראשי של הנקודות בתת העץ שלו P_v בטווח השאילתה, לפי X. באופן דומה, עבור כל צומת $u \in T_v$, $P_u \subseteq P_v$ הן הנקודות בתת העץ של u לפי y, ואני שומר לכל היותר נקודה אחת מכל תת עץ כזה p, $p \in P_u \subseteq P_v$ ולכן היא בטווח $[x_1, y_1, x_2, y_2]$

2. הנקודה מקסימלית ב-z:

עבור כל P_u שמרנו את p המקסימלית ב-z, ובכל החיפוש שמרנו את המקסימלית מבין ה-p השונות.

סיבוכיות:

מקום: בדומה לניתוח בכיתה, יש את העץ הראשי שתופס $O(n)$ מקום, ויש את העצים המשניים, כל עץ משני תופס $O(n_v)$ מקום. כל נקודה נמצאת בתת עץ של $\log n$ צמתים בעץ הראשי ולכן $\sum_{v\in T} n_v = n\cdot\log n$. סה"כ - $O(n\log n)$

זמן בנייה: עבור כל צומת v, צריך למיין את ערכי y לכל n_v שלו. עולה $n_v\log n_v$, אזי:

$$\sum_{v\in T} n_v\log n_v \le \sum_{v\in T} n_v\cdot\log n = \log n\cdot\sum_{v\in T} n_v = \log n(n\log n) = (n\log^2 n)$$

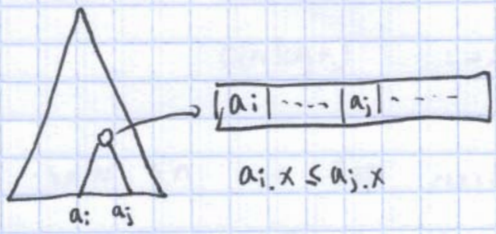
סה"כ - $O(n\log^2 n)$.

אפשר לחסוך
בבנייה ז' מיון
מיון אחד

חיפוש: שוב בדומה לניתוח בכיתה ~~עבור~~ אותה עץ דו מימדי (עץ). בעץ הראשי אני נתקל בלכל היותר $\log n$ צמתים v שאני צריך לבצע חיפוש אותה בהם. חיפוש אותה בכל אחד מהם ואיסוף המקסימום עולה $O(\log n_v)$, סה"כ: $\log n\cdot\log n = O(\log^2 n)$

בנ״ב:

1. נבנה 1D-Range tree לפי Y.
2. בכל צומת פנימי נחזיק מצביע למערך ממוין לפי x של כל הנקודות בתת העץ של הצומת.



חיפוש:

1. נעשה חיפוש טווח לפי y בעץ הראשי.
2. לכל צומת פנימית V שכל הנקודות בתת העץ שלו בטווח ניגש למערך הממוין בצומת ונשמור איברים ברשימה שקטנים (בציר ה-x) מ-α V (נלך משמאל לימין - מהקטן לגדול) וברגע שנגיע לאיבר שגדול מ-α נצא מהצומת ונמשיך בשאילתת הטווח.

נכונות:

1. כל הנקודות שהחזרנו הן בטווח: מנכונות Range tree נכנסנו לצומת פנימי u כך ש-P_u בטווח y של השאילתה, ובתוך כל צומת כזו נשמור רק את הנקודות שקטנות מ-α בציר ה-x.
2. כל הנקודות בטווח הוחזרו: נניח בשלילה כי יש נקודה בטווח שלא הוחזרה, p:
   i. אם לא נכנסנו לצומת שמכילה אותה זו סתירה לנכונות חיפוש ב-Range tree.
   ii. נכנסנו לצומת אבל הפסקנו לעבור על המערך לפניה - סתירה לכך ש-$p.x \le \alpha$.

<u>סיבוכיות:</u>

<u>מקום:</u> העץ הראשי תופס $O(n)$ מקום.

כל עץ משני תופס $O(n_v)$ מקום.

כל נקודה נמצאת בעץ המשני של $\log n$ צמתים v, ולכן

$\sum_{v \in T} n_v = n\log n$.

סה"כ: $O(n\log n)$

<u>זמן בנייה:</u> לבנות ~~עץ~~ עץ שווה חד מימדי לוקח $O(n\log n)$, ועבור

כל צומת $v \in T$ נמיין את כל הנקודות בצומת לפי x

$n_v \log n_v$. סה"כ $n\log n + \sum_{v \in T} n_v \log n_v = O(n\log^2 n)$

שוב, אפשר למיין פעם אחת

<u>זמן:</u> בחיפוש שווה ניכנס ל $O(\log n)$ צמתים פנימיים.

נחלק את הצמתים לשני סוגים:

i) צמתים שבהם אני שומר איברים, אני כמות העבודה

שנעשית בצומת כזו היא $O(k_v)$ (מס' האיברים בטווח הרלוונטי)

וכמות סה"כ העבודה עבור צמתים כאלו היא $\sum_v k_v = O(k)$

ii) צמתים שבהם אין נקודות בטווח, אני מגלה זאת כבר בחיפוש

הראשון (הבינארי) ולכן עבור כל צמת כזאת כמות העבודה

היא $O(1)$. סה"כ עבורן זה $O(\log n)$.

סה"כ הזמן יהיה הסכום של שני סוגי הצמתים $O(k+\log n)$