אוניברסיטת ת"א
סמסטר א', תשפ"ה
הגשה עד ה-29.12 ב-moodle

תרגיל בית 3 במבני נתונים

על כל התשובות להיות מנומקות. בכל שאלה יש לבחור במימוש היעיל ביותר האפשרי מבחינת סיבוכיות זמן. יש לענות על השאלות במקומות המוגדרים לכך.

<u>שאלה 1</u>

א. תארו מבנה נתונים התומך בפעולות הבאות על קבוצה S מתחום בעל סדר מלא.

- Median(S): מחזיר את החציון ב-S (אם מס' האיברים זוגי – חציון תחתון, כלומר האיבר שימוקם במקום ה-n/2 אם נמיין את האיברים)
- Min(S): מחזיר את האיבר הקטן ביותר ב-S
- Max(S): מחזיר את האיבר הגדול ביותר ב-S
- Insert(x,S) : מוסיף איבר x ל-S
- Delete(x,S) : מניחים ש-x נמצא לפני הפעולה ב-S. הפעולה מוציאה את x מ-S.

על הפעולות median, min, max לקחת $O(1)$ זמן במקרה הגרוע, ועל הפעולות insert ו-delete לקחת $O(\log n)$ זמן במקרה הגרוע. כמו כן תארו בקצרה כיצד לבצע כל פעולה.

ב. שנו במידת הצורך את מבנה הנתונים שהגדרתם בסעיף הקודם כך שנוכל גם למצוא את האלמנט ה- i בגודלו בזמן $O(\log(i \; mod \; \frac{n}{2}))$, לדוגמה עבור $i = \frac{n}{2} + 5$ העלות של הפעולה צריכה להיות $O(\log(5)) = O(1)$. תארו במדויק כיצד לבצע פעולה זו.

א

א. נתחזק עץ AVL בו כידוע הפעולות insert ו-delete לוקחות $O(\log n)$ זמן במקרה הגרוע. בעץ נתחזק מצביעים לצמתים median, min, max כך שגישה אליהם תהיה ב $O(1)$ זמן במקרה הגרוע. העץ יתמוך בפעולות select. בכל הכנסה ומחיקה נעדכן את המצביעים לערכים median, min, max בעזרת $select(uppervalun(n\backslash 2))$ $select(1)$ $select(n)$ בהתאמה.

ב. בנוסף לעץ הראשי נתחזק 2 עצי finger- tree נוספים, עץ אחד בו כל האיברים קטנים מהחציון ועץ שני בו כל האבירים גדולים מחציון. בכל הכנסה ומחיקה מהעץ הראשי מסעיף א נעדכן את העצים הללו (נוסיף או נמחק איבר בהתאם לאיבר שנוסף והחציון שהתעדכן) פעולת מחיקה והכנסה ב2 עצי העזר תיקח גם כן $O(\log n)$ בהתאם לעצי AVL, מכך זמן המחיקה וההכנסה ישארו ב$O(\log n)$
עבור מציאת החציון עצמו נשתמש במצביע הפשוט מסעיף א וכמו שאמרתי כבר הדבר בהכרח יקח זמן קבוע.
עבור מציאת $i < \frac{n}{2}$ נחפש בעץ בו כן איברים קטנים מחציון וכבר עבור $i > \frac{n}{2}$ נחפש בעץ בו כן האיברים גדולים מהחציון. וכן בכל תת עץ נחפש את האיבר ה$i \ mod \ \frac{n}{2}$ בגודלו. כידוע בעץ finger- tree זמן החיפוש של האיבר ה k הינו $O(\log(k))$. ובכך נקיים את הנדרש.

שאלה 2

תכננו מבנה נתונים המכיל מפתחות טבעיים ללא חזרות (כלומר המפתחות ייחודיים) ותומך בפעולות Insert,Delete,Search בזמן $O(\log n)$ וכן בפעולה IncreaseAll(k), אשר מוסיפה את הטבעי k לכל המפתחות, בזמן $O(1)$. בנוסף, על מבנה הנתונים לתמוך בפעולה $\mathrm{EvenLess}(x)$ שמחזירה בזמן $O(\log n)$ את סכום המפתחות הזוגיים במבנה שערכם לכל היותר x (נניח ש-x נמצא במבנה), כאשר n הוא מספר המפתחות במבנה.

חי

נתחזק משתנה צד בשם Increase כך שבכל קריאה IncreaseAll(k) נוסיף k למשתנה כך שכל קריאה כמובן תתרחש בזמן קבוע $O(1)$. בנוסף נתחזק עץ AVL כך שבכל צומת נתחזק את המשתנים num of even, sum of even, num of odd, sum of odd ערכי הצמתי בעץ יתעלמו מערך Increase.
ניתן לתחזק ערכים אלו בזמן קבוע לכל צומת כך שעבור v.key איבר זוגי:

$$v.numofeven \leftarrow v.left.numofeven + v.right.numofeven + 1$$
$$v.sumofeven \leftarrow v.left.sumofeven + v.right.sumofeven \ + v.key$$
$$v.numofodd \leftarrow v.left.numofodd + v.right.numofodd$$
$$.sumofodd \leftarrow v.left.sumofodd + v.right.sumofodd$$

עבור v.key איבר אי זוגי נתחזק את האיברים באופן דומה.

הכנסה ומחיקה- יתרחשו באופן דומה לעץ avl רגיל כפי שראינו בכיתה ב$O(\log n)$ בעוד שערכי הצמתי יתוחזקו בזמן קבוע.

search – כדי למצוא את האיבר הx נחשב את האיבר ה $\mathrm{x}^* = \mathrm{Increase} - x$, החישוב ידרוש זמן קבוע. ואז נבצע חיפוש של x^* כפי שראינו בכיתה בעץ AVL מה שידרוש $O(\log n)$ זמן.
$\mathrm{EvenLess}(x)$ – נחפש את האיבר x בעזרת search. עבור IncreaseAll זוגי מכך ש even = even + even נחזיר את:

$$return = increase * x.numofeven + x.sum\,ofeven$$

עבור Increase זוגי מכך ש odd = even + odd ו even = odd + odd נחזיר את:

$$return = increase * x.numofodd + x.sum\,ofodd$$

בשיטה הזו נחזיר את סכום הזוגים הנמצאים שנמצאים מתחת לאיקס וכן שאר הפעולות למעט החיפוש יתרחשו בזמן קבוע ולכן $\mathrm{EvenLess}(x)$ תחזיר את הנדרש ב- $O(\log n)$.

שאלה 3

נתון עץ AVL בו מאוחסנים מפתחות טבעיים שונים זה מזה. בכל צומת v בעץ שמור size(v), גודל תת העץ של v.

א. תארו אלגוריתם המוצא את הטבעי המינימלי שאינו שייך לעץ בזמן $O(log\ n)$.
רמז: חישבו מהי משמעות הביטוי $k - rank(k)$, כאשר $rank(k)$ מציין את מספר המפתחות בעץ הקטנים או שווים ל-k.

ב. נכליל את הסעיף הקודם - נסמן את קבוצת המפתחות השמורים בעץ ב-S, כאשר $|S| = n$.
נגדיר: nextMissingAfter(i) - המספר הטבעי המינימלי $j \notin S$ הגדול מ-i .
למשל, עבור עץ שקבוצת המפתחות השמורים בו היא S = {11,10,9,6,5,4,2,1}, מתקיים NextMissingAfter(5) = 7, NextMissingAfter(9) = 12, NextMissingAfter(3) = 7.
הראו כיצד ניתן לחשב את הפונקציה nextMissingAfter(i) בסיבוכיות זמן $O(log\ n)$.

א. בהתייחס אל הרמז: עבור צומת המקיימת k = rank(k) מכך שהמפתחות המאוחסנים שונים זה מזה וטבעיים בהכרח הצמתים עד לצומת הינם סדרה חשבונית בהפרש d=1. מכך נרצה למצוא את האיבר המקסימלי המקיים k = rank(k) ולהוסיף לו 1. נתחיל מהשורש אם השורש מקיים k = rank(k) נשמור את ערך הצומת במשתנה x נמשיך לבדוק האם קיים עוד צומת המקיימת את התנאי בתת העץ הימיני שכן נרצה למצוא את המספר המקסימלי המקיים את התנאי. אם השורש לא מקיים את התנאי נמשיך לתת העץ השמאלי ונפעל באותה הדרך. לבסוף נחזיר x+ .מעבר זה עד עד תחתית העץ בוודאי יתרחש ב$O(log\ n)$

ב. נחפש את האיבר i בעץ ב$O(log\ n)$ נחשב את i – rank(i) *נרצה למצוא את האיבר המקסימלי k המקיים:*

$$k - rank(k) \leq i - rank(i)$$

נחזיר k +1. נתחיל את האלגוריתם מסעיף א באיבר הi ואז נמשיך כמו בסעיף א באותם סט בדיקות. (אם האיבר i לא קיים בעץ נוסיף אותו ב$O(log\ n)$).

<u>שאלה 4</u>

א. הוכיחו כי בעץ AVL בגובה h, כל העלים בעומק לפחות $h/2$.

ב. הוכיחו כי כל סדרה בת n הכנסות לעץ AVL, גם הטובה ביותר וגם הגרועה ביותר, היא בעלות $\theta(n\log n)$ (הכנסות עם חיפוש שמתחיל מהשורש).

ג. בתרגול ניתחנו מצב בו בהינתן m ועץ AVL מתבוננים בתת העץ המינימלי שמכיל את m המפתחות הקטנים ביותר. מצאו חסם עליון אסימפטוטי של גודל תת-עץ זה כפונקציה של m.
תזכורת: בהרצאה הוכחתם כי בעץ AVL בעל n צמתים בגובה h מתקיים ש- $h \leq \log_{\phi} n$.

א. אראה באינדוקציה:
עבור נסמן ב x_h את העומק של העלה הכי קרוב לשורש בעץ בגובה h.
עבור h =1 או h = 0 התנאי בוודאי מתקיים.
נניח כי הטענה מתקיימת עבור עצים בגובה $h-1$, $h-2$ ונראה כי :

$$x_h = 1 + \min\left(x_{h-1,}\, x_{h-2}\right) \geq 1 + \frac{h-2}{2} = \frac{h}{2}$$

*ניקח מינימום של שני האיברים האנ״ל כי ייתכן שלאחר הוספת איבר לעץ נצטרך לסובב אותו.
ב. חסם עליון: ראינו בכיתה כי כל הכנסה של איבר לעץ היא $O(log\ n)$ ומכך הכנסה של n איברים בהכרח תהיה מ $O(nlog\ n)$.
חסם תחתון: נכניס לעץ $\frac{n}{2}$ איברים. מכך גובה העץ הינו לפחות חסם תחתון של: $\log\left(\frac{n}{2}\right)$. נסיק מסעיף א׳ כי עומק העלה הכי קרוב לעץ הינו $\frac{\log\left(\frac{n}{2}\right)}{2}$. בכל הכנסה נצטרך להגיע לעלה החיצוני בעץ ובמינימום לעבור מסלול באורך $\frac{\log\left(\frac{n}{2}\right)}{2}$ איברים.
ומכך בהכנסה של $\frac{n}{2}$ האיברים הנותרים נקבל כי נצטרך לפחות:

$$\frac{n}{2} * \frac{\log\left(\frac{n}{2}\right)}{2} = \Omega(\mathrm{n}log(n))$$

סה״כ הנדרש מתקיים.
ג) גובה של תת העץ הוא לכל היותר $\log_\phi m + 2$. וכן ראינו גם שלעץ בגובה h יש לכל היותר $2^{h+1} - 1$ צמתים. ולכן מספר הצמתים בתת העץ יהיה חסום באופן הבא:

$$2^{\log_\phi m+3} - 1 = 8 * 2^{\log_\phi m} - 1 = O(2^{\log_\phi m})$$

<u>שאלה 5</u>

הציעו מימוש למבנה נתונים התומך בפעולות הבאות, על ישרים מהצורה $y = ax + b$:

- Search(a,b) – האם הישר $y = ax + b$ נמצא במבנה?
- Insert(a,b) – הכנסת הישר $y = ax + b$ למבנה, אם אינו חותך בקטע $[0,1]$ אף ישר אחר שנמצא כבר במבנה. כלומר הישר החדש ייכנס אם אין שום ישר אחר במבנה שנחתך איתו בנקודה (x_1, y_1) המקיימת $0 \leq x_1 \leq 1$. אם תנאי זה לא מתקיים הפעולה לא תבצע דבר.

סיבוכיות הזמן הדרושה עבור שתי הפעולות היא $O(\log n)$ במקרה הגרוע, כאשר n הוא מספר הישרים במבנה. תארו תחילה מה כולל המבנה שלכם (כלומר איזה מידע נשמר וכיצד) ולאחר מכן את הפעולות השונות, כולל הסבר קצר מדוע הן עומדות בדרישות הסיבוכיות.

<u>רמז</u>: ניתן לייצג ישר ע"י שתי נקודות במישור.

בהתאם לרמז נייצג כל ישר בעזרת 2 נקודות $(0, y_1), (1, y_2)$, מרציפות בהכרח יש כאלו לכל ישר. בהינתן ישר מציאת ערכי y_1 y_2 יתרחש בזמן קבוע.

לאחסון המידע נשתמש בעץ *AVL* כך שכל צומת מכילה את הערכים (y_1, y_2) . נכניס ערכים לעץ כך ששני ערכי ה-*y* בכל צומת שמאלית קטנים ממש מ2 ערכי ה-*y* של צומת האב וכל 2 ערכי ה-*y* של הצמתים הימניים גדולים ממש מערכי האב. כלומר :

$$[(0,a),(1,b)] > [(0,c),(1,d)] \Leftrightarrow a > c \ \wedge b > d$$
$$[(0,a),(1,b)] < [(0,c),(1,d)] \Leftrightarrow a < c \ \wedge b < d$$
$$[(0,a),(1,b)] = [(0,c),(1,d)] \Leftrightarrow a = c \ \wedge b = d$$

נשים לב כי עבור 2 ישרים בעלי הערכים $(y_1, y_2), (y_1^*, y_2^*)$ במקרה בו $y_1 \geq y_1^*, y_2 \leq y_2^*$ או $y_1 \leq y_1^*, y_2 \geq y_2^*$ בהכרח הישרים נחתכים בקטע $[0,1]$ וכן גם במקרה לא נכניס את הישר השני מבינהם שנכנס לעץ. בכך התנאי להכנסה מתקיים.

ראינו בכיתה כי עץ *AVL* מקיים כי סיבוכיות הזמן הדרושה עבור שתי הפעולות Search וInsert היא $O(\log n)$ במקרה הגרוע וכן בדיקה של שני ערכים y_1, y_2 תתרחש בזמן קבוע.

שאלה 6

בשאלה זו תתבקשו להציע מבנה נתונים אשר מתחזק קבוצת מספרים ממשיים. על מבנה כזה לתמוך בפעולות הבאות:

- פעולת ()NewSet שמייצרת קבוצה חדשה וריקה.
- פעולת Insert(a) שמוסיפה את המספר a לקבוצה.
- פעולת Delete(a) שמוחקת מהקבוצה את המספר a אם שייך אליה.
- פעולת Find(a) שבודקת אם המספר a שייך לקבוצה.
- SeparateInterval(a,b) אשר מקבלת אינטרוול (a,b) ומפצלת את הקבוצה הנוכחית S לשתי קבוצות $S \cap (a,b)$ ו- $S \setminus (a,b)$.
- IntersectionSize(a,b) אשר מקבלת אינטרוול (a,b) ומחזירה את הגודל $|S \cap (a,b)|$ עבור הקבוצה הנוכחית S.

על כל הפעולות לרוץ במקרה הגרוע בזמן לוגריתמי במספר האיברים בקבוצה.

ברוח מטלת בית זו נגדיר עץ AVL המתחזק size של כל צומת.

()Newset כמובן תרוץ בזמן קבוע

ראינו בכיתה כי פעולות אלו רצות בזמן לוגריתמי במקרה הגרוע insert, delete, find

נשתמש ב split ו join עבור שאר הפעולות:

SeparateInterval(a,b): במידה a וb לא נמצאות בקבוצה נוסיף אותם עם insert בזמן לוגריתמי. נבצע split לפני a ובכך נקבל 2 עצים אחד מכיל את כל הגדולים מa נסמנו בA1 ואחד את כל הקטנים מ a נסמנו בA2. בהכרח $b \geq a$ ולכן נמצא את b בעץ A1.

שוב נפצל את A1 ע״פ b. נקבל 2 עצים 3A גדול מb ו A4 קטן מb . סה״כ בהכרח $i \in A4 \iff a < i < b$

מכך נחבר את 2 העצים הנותרים 3A ו-A2 בעזרת join כך שהם מייצגים את הקבוצה $S \setminus (a, b)$. וכן $A4$ ייצג את הקבוצה $S \cap (a, b)$. אם a או b היו שייכים לקבוצה מלכתחילה נוסיף אותם לקבוצה $S \setminus (a, b)$.

ראינו בכיתה כי לפעולות split join סיבוכיות לוגרתמית במספר איברי הקבוצה ומכך בהכרח גם לפעולה זו סיבוכיות שכזו.

IntersectionSize(a,b) - מכך שהקבוצה שהגדרנו בתור עץ avl מתחזקת size של כל צומת ניצור בעזרת SeparateInterval(a,b) את הקבוצה $S \cap (a, b)$ וכן נחזיר את size של השורש. ובכך גם את גודל הקבוצה.