שאלה 1

א. נשתמש ברשימת דילוגים דטרמיניסטית. האיברים ברשימה יהיו מס' הקומות השונים. בנוסף נחזיק מצביע Curr(בשורה התחתונה ביותר של האיברים) עבור הקומה הנוכחית.

מימוש הפעולות:

- Init():

ניצור ונאתחל את רשימת הדילוגים (עם איברים +∞ ו-∞- בקצוות כפי שנלמד בהרצאה) ואיבר בודד 0 שמציין את הקומה ההתחלתית O(1). בנוסף נעדכן את המצביע של הקומה הנוכחית להצביע על איבר ה-0 O(1). סה"כ סיבוכיות זמן O(1) כנדרש.

- AddStop(k):

נבצע insert(k) ברשימה ונכניס את האיבר הk במקומו ברשימת הדילוגים. O(log(n)) כנדרש.

- NextStop():

ניגש לאיבר השמור לנו במצביע Curr, ונבדוק האם איבר הnext שלו מאותחל. במידה ולא, לא קיימת קומה נוספת, ונדפיס את הקומה הנוכחית ונסיים. אחרת, נקדם את המצביע Curr להצביע על האיבר הnext ונדפיס את ערכו. מס' קבוע של פעולות O(1) ולכן סה"כ O(1) כנדרש.

סיבוכיות המקום הנדרשת לרשימת דילוגים דטרמיננסטית הינה $O(nlogn)$ כאשר n הוא מספר האיברים.

ב. נשמור את מספר הקריאות לפעולה $AddStop(k)$ במשתנה $counter$. נשתמש במערך דינאמי A באופן הבא: בכל קריאה ל-$AddStop(k)$ נבדוק האם $counter \geq \frac{n}{2}$ כאשר n הוא גודל המערך הנוכחי. אם כן – נגדיל את המערך פי שניים. כל תא i במערך יחזיק ערך 0 אם המעלית אינה עתידה לעצור בקומה ה-$i+1$ ו-1 אחרת. בנוסף, נשמור את מספר הקומה הנוכחית ב-f.

- $Init()$ – נאתחל מערך בגודל 2 שכולו אפסים. נאתחל: $counter = 0, n = 2, f = 0$.
איתחול מערך ב-$O(1)$ כפי שנלמד בהרצאה וכן מספר פעולות קבוע. בסה"כ $O(1)$.
- $AddStop(k)$ – כאמור, נבדוק האם מתקיים $counter \geq \frac{n}{2}$ ונעדכן את גודל המערך בהתאם. אם המערך גדל, נאתחל את התאים שנוספו ל-0. נבצע $A[k-1] = 1$.
- $NextStop()$ – נעדכן את הקומה הנוכחית לתא הבא שבו מופיע הערך 1 ונדפיס את האינדקס שלו + 1.

הוכחת סיבוכיות

- $AddStop(k)$ – נשתמש בשיטת הצבירה. ניקח סדרה של m פעולות של $AddStop$.
נחלק את הסדרה ל-L מקטעים באורכים $m_1, m_2, \ldots, m_L$. מקטע כזה יכיל אוסף פעולות הכנסה בין פעולת שינוי גודל המערך (לא כולל) עד לפעולת השינוי העוקבת (כולל).
נראה שכל מקטע באורך m_l רץ בזמן $O(m_l)$ ומכך נסיק שסדרה בת m פעולות תתבצע ב-$O(m)$. נביט במקטע m_l כלשהו. נסמן ב-n את גודל המערך לאורך מקטע זה. זמן הריצה הכולל של המקטע מורכב מ-m_l פעולות הכנסה למערך אשר לוקחות $O(1)$ כל אחת ופעולת שינוי גודל המערך בסיום המקטע ב-$O(n)$.
נשים לב שמספר פעולות ההכנסה עד לשינוי הגודל הבא הוא $\Omega(n)$. כלומר, $n = O(m_l)$
ולכן זמן הריצה הכולל הוא: $m_l \cdot O(1) + O(n) = O(m_l)$.

לכן סדרה בת m פעולות תתבצע ב-$O(m)$, כלומר הסיבוכיות של פעולת $AddStop(k)$ היא $O(1)$ משוערך, כנדרש.

- $NextStop()$ – נשתמש בשיטת הצבירה. ניקח סדרה של m פעולות $NextStop()$ בהן הקומה משתנה (כלומר קיימת קומה עם מספר הגדול ממש ממספר הקומה הנוכחית). נשים לב שלאורך סדרה כזאת גודל המערך קבוע שכן לא התבצעו פעולות $AddStop$. נסמן את גודל המערך ב-n. לכן, בסה"כ נקבל ש-$m = O(n)$ וכן שהאיבר הבא במערך בו המכיל "1" נמצא במרחק לכל היותר $O(n)$ מהקומה הנוכחית, כאשר n קבוע ולכן סיבוכיות פעולת ה-$NextStop()$ הינה $O(1)$ משוערך.
  כפי שראינו בתרגול, סיבוכיות המקום של מערך דינמי הינה $O(k)$

<u>שאלה 2</u>

א. נשתמש בעץ דרגות AVL הממוין מיון ראשוני לפי ערך ה-x של הנק' ומיון משני לפי ערך ה-y של הנק'. בנוסף נחזיק 2 משתנים Ksmallest, Kbiggest שיחזיקו מצביע לאיבר ה-k הקטן ביותר וה-k הגדול ביותר בהתאמה. נחזיק תא זיכרון נוסף לשמירת ערך ה-k.

<u>מימוש הפעולות:</u>

- Init(k):
ניצור עץ AVL ריק O(1) ונשמור את ערך ה-k שקיבלנו בתא הזיכרון המתאים O(1). בנוסף, נאתחל את המצביעים Ksmallest, Kbiggest. סה"כ O(1) כנדרש.
- Insert((x,y)):
נכניס את האיבר (x,y) לעץ הAVLO(log(n)). ונשתמש בפעולת select באופן הבא:
- Ksmallest=select(k+1) – זהו הקטע שיש k איברים קטנים ממנו בעץ.
- Kbiggest=select(n-k) – זהו הקטע שיש k איברים גדולים ממנו בעץ.
2 פעולות הselect לוקחות O(2log(n)), ולכן בסה"כ סיבוכיות הזמן הינה O(log(n)) כנדרש.
- Delete((x,y)):
נסיר את האיבר (x,y) מעץ ה-AVL (O(logn(n))). נעדכן את המצביעים בעזרת פעולת select באופן דומה:
- Ksmallest=select(k+1) – זהו הקטע שיש k איברים קטנים ממנו בעץ.
- Kbiggest=select(n-k) – זהו הקטע שיש k איברים גדולים ממנו בעץ.
2 פעולות הselect לוקחות O(2log(n)), ולכן בסה"כ סיבוכיות הזמן הינה O(log(n)) כנדרש.
- IsCentric((x,y)):
נבדוק האם ערך ה-x שהתקבל גדול או שווה לערך ה-x של Ksmallest וגם נבדוק האם ערך ה-y שהתקבל קטן או שווה לערך ה-y של Kbiggest – אם שני התנאים מתקיימים נחזיר True, אחרת נחזיר False. סה"כ מס' קבוע של פעולות O(1) ולכן סיבוכיות הזמן הינה O(1) כנדרש.
סיבוכיות המקום הנדרשת לעץ AVL הינה $O(n)$.

ב. נשתמש ב2 עצי דרגות מסוג 2-3, כאשר העץ הראשון מסמל את הקבוצה A, ואילו העץ השני את הקבוצה B. העץ יהיה ממוין באופן הבא: מיון ראשוני ע"פ ערך ה-x של הקטע ומיון משני לפי ערך ה-y. בנוסף, כל עלה ב-A ישמור את מס' העלים שנמצאים לפניו, ואילו כל עלה ב-B ישמור את מס' העלים אחריו. המידע הנוסף שיישמר בכל עץ הינו:
-בכל צומת פנימית ב-A נשמור את מס' העלים שבתת-העץ השמאלי שלו.
-בכל צומת פנימית ב-B נשמור את מס' העלים שבתת-העץ הימני שלו.
בנוסף, נחזיק רשימה של קטעים שמקיימים את התנאי הדרוש לפעולה השלישית.
Init(): נאתחל שני עצי דרגות 2-3 ריקים ונאתחל רשימה ריקה. O(1).
Insert((x,y),G): נכניס את האיבר אל העץ המתאים (A או B) O(log(n)). בנוסף נעלה על מסלול החיפוש של האיבר שהוכנס ונעדכן מהעלה עד לשורש את המידע הנוסף. נעדכן את מס' העלים לפניו\אחריו בהתאם לעץ (ע"י פעולת rank((x,y)) בשימוש במידע הנוסף על מנת לקבוע כמה עלים קטנים ממנו יש בA\גדולים ממנו בB בהתאמה – O(log(n))).
כעת, אם הרשימה אינה ריקה נבדוק בדיקה נוספת עבור האיבר שנמצא ברשימה – אם הוא אינו מקיים את התנאי יותר נסירו מהרשימה, אם הוא מקיים נסיים.

אחרת אם הרשימה ריקה - נוסיף את הקטע לעץ השני O(log(n)) ונבדוק את הערך של מס' העלים לפניו\אחריו עבור העץ השני (שוב על ידי שימוש ב-rank((x,y))). אם נמצא כי מס' העלים לפניו\אחריו בעץ הראשון שווה למס' העלים אחריו\לפניו בעץ השני, נשמור את הקטע בראש הרשימה.

ולכן סה"כ O(log(n)) כנדרש.

StartBeforeInA=StartAfterInB():

נחזיר את הקטע שנמצא בראש הרשימה, אם הרשימה ריקה נחזיר שגיאה. O(1).

סיבוכיות המקום הנדרשת לעצי ה-2-3 הינה $O(2n) = O(n)$

שאלה 3

א. תחילה "נמקם" את העץ T1 משמאל לT2 כך שעומק העלים זהה בשני העצים (הגובה לאו דווקא שווה).
מהנתון כי כל מפתח בT1 קטן מכל מפתח בT2 נקבל כי שמורת עץ 2-3 נשמרת, ובנוסף כל אחד מן העצים הללו הוא בעצמו עץ 2-3 ולכן השמורה הפנימית בהם נשמרת גם כן. כעת נרצה לאחד את העצים לעץ 2-3 יחיד. נפעל לפי המקרים הבאים(באופן רקורסיבי כלפי מעלה לכיוון השורש של העצים – נתחיל מהרמה מעל העלים בעץ):

- אם העץ השמאלי (T1) הוא בעל 2 בנים נצרף לאביו כבן שלישי ימני את הבן השמאלי ביותר באותו העומק בעץ השני כך שיחזיק כעת 3 בנים. נעדכן את האינדקס הנוסף של צומת האב ע"פ המינימום של T2 (נמצא בתא השמאלי ביותר בעץ. מציאתו תיקח $O(\log n_2)$). נמשיך כלפי מעלה באופן רקורסיבי.
- אחרת, אם יש לו שלושה בנים, נמשיך כלפי מעלה.
- אם הגענו לשורשים של העצים והם באותו הגובה ניצור שורש חדש ונחבר לו את שני השורשים של T1 ו T2, נעדכן את האינדקסים בהתאם ונסיים.
- אחרת, אחד העצים גבוה יותר מהשני, ולכן כאשר נגיע ברקורסיה לשורש העץ הנמוך יותר, נחבר את השורש שלו לאב של ההאיברים האחרים בעץ השני ונעדכן את האינקסים של האב בהתאם (אם כבר יש לו 3 בנים, ניצור שורש חדש נוסף ונחבר אליו את השורש של העץ הנמוך יותר בתוספת אחד הבנים הצמודים אליו באותה הרמה של העץ האחר ונמשיך באופן זה עד שנגיע לגובה העץ המקס' מבינהם).

בסיור שתיארנו מהרמה התחתונה ועד לשורש העץ הגבוהה יקח לנו גם $O(\max\{\log(n_1),\log(n_2)\})$ (לפי גובה העץ הגבוה מבינהם) ובכל רמה אנו מבצעים מס' קבוע של פעולות תיקון $O(1)$. ולכן בסה"כ סיבוכיות הזמן של האלגוריתם המוצע הינה $O(\max\{\log(n_1),\log(n_2)\})$ כנדרש.
סיבוכיות המקום הינה $O(1)$

ב. נשתמש באלגוריתם הדומה לזה של סעיף א', כאשר במקום לחפש בכל שלב את המינימום והמקסימום של תת העצים, נשתמש בפעם הראשונה בערכי המינימום והמקסימום הנתונים ונמשיך ברקורסיה כלפי מעלה. מכיוון שאנחנו "מתחילים" משורש העץ הנמוך ומתקדמים כלפי מעלה לעבר גובה השורש של העץ הגבוה כאשר אנחנו מבצעים פעולות קבוע $O(1)$ בכל רמה, ובנוסף במקרה הגרוע מוסיפים איבר נוסף שהוא השורש המשותף לשני העצים, נקבל שהסיבוכיות הינה $O(|h_1-h_2|+1)$ כנדרש.
סיבוכיות המקום הינה $O(1)$

ג. עבור כל $i=1,2,\dots,k$ נבצע $join(T,T_{i+1})$ כאשר פעולת ה-$join()$ היא זאת שהוגדרה בסעיף הקודם, ו-$T=join(T_1,join(T_2,\dots,join(T_{i-1},T_i)\dots))$. סיבוכיות $join(T_i,T_{i+1})$ הינה $O(|h_i-h_{i+1}|+1)$.

נשים לב שאם היו שלושה עצים מאותו הגובה h, אז ייתכן שגובה העץ המאוחד היה תלוי בכמות העצים המאוחדים, ולא חסום ע"י חסם הסיבוכיות מהסעיף הקודם.
במילים, בכל שלב נאחד את העץ ה-i+1 עם איחוד כל העצים הקודמים (ראשית נאחד את העץ הראשון והשני, את איחודם עם השלישי וכן הלאה). סה"כ $k-1$ איחודים. הסיבוכיות תהיה, אם כך:

$$\sum_{i=1}^{k-1} O(|h_{i+1}-h_i|+1) \overset{*}{=} O\left(\sum_{i=1}^{k-1} h_{i+1}-h_i+\sum_{i=1}^{k-1}1\right) \overset{**}{=} O\left(h_k-h_1+k-1\right)=O\left(h_k-h_1+k\right)$$

* מהנתון $h_1 \leq \cdots \leq h_k$.
** טור טלסקופי.
סיבוכיות המקום הינה $O(k)$

ד. תחילה נבצע חיפוש של x בעץ. במהלך הירידה במסלול החיפוש נשמור שתי קבוצות של עצים: אלו שמכילים ערכים הגדולים מ-x (כלומר, הצמתים הנמצאים "ימינה" מ-x עבור כל צומת במסלול החיפוש); ואלו שמכילים ערכים הקטנים מ-x (באותו האופן). כאשר נגיע ל-x נוסיף אותו לקבוצת העצים הקטנים.
נשים לב שגודל הקבוצות הוא לכל היותר $2 * logn$, מכיוון שבכל ירידה במסלול החיפוש נשמור לכל היותר שני עצים של ערכים הגדולים מ-x ואחד קטן; או הפוך: שני עצים של ערכים הקטנים מ-x ואחד הגדול. נשים לב כי גבהיי העצים הללו חסומים מלמעלה ע"י $logn - 1$ (עבור תת-העץ שיוצא מהשורש, כלומר הפיצול הראשון שבמסלול החיפוש) ומלמטה ע"י 0 (עבור עלים שהינם אחים של x). עבור כל עץ נשמור את ערכי המקסימום והמינימום שלו באופן הבא: ערך המינימום יהיה ערך הצומת הראשון שקטן מ-x וערך המקסימום יהיה ערך הצומת האחרון שגדול מ-x.
כעת, נשתמש בסעיף ג' ע"מ לאחד את קבוצת העצים הקטנים לעץ אחד ואת קבוצת העצים הגדולים לעץ שני. בסה"כ לכל עץ $O\left(h_k - h_1 + k\right) = O\left(\log n - 1 - 0 + 2\log n\right) = O(\log n)$ ולכן בסה"כ $O(logn)$.
סיבוכיות המקום הינה $O(logn)$

שאלה 4

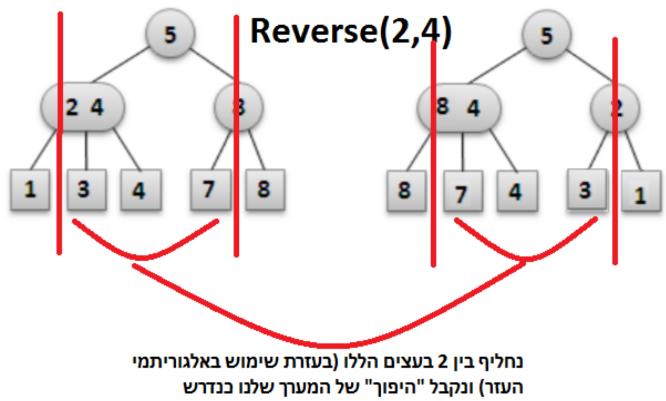
נשתמש בשני עצי 2-3, בעלי מידע נוסף של מס' הבנים בתת העץ(3 שדות שונים לכל תת עץ עבור כל צומת פנימי). עץ אחד יהיה מסודר תחילה לפי סדר הכנסת המערך ואילו השני יהיה העתק "מראה" שלו, ובכך יאפשר לנו לבצע "רוורס" על האיברים.

מימוש הפעולות:

- Init(A,n): נאתחל עץ 2-3 ריק O(n) ונכניס את האיברים לפי סדר הופעתם במערך אל העלים כך שהעלה השמאלי ביותר מכיל את הערך הראשון במערך והימני ביותר מכיל את הערך האחרון במערך. נבצע סיור נוסף (PostOrder) ונעדכן את מס' הבנים בתת העץ הרלוונטי עבור כל צומת פנימי O(n). בנוסף נבצע אלגוריתם זהה עבור עץ ה"מראה" השני, רק שסדר ההכנסה יהיה הפוך מהסדר הקודם, כלומר האיבר האחרון במערך יהיה העלה השמאלי ביותר בעץ והאיבר הראשון במערך יהיה בעלה הימני ביותר, גם כאן נמצא עדכון בעזרת הסיור. O(n). ולכן בסה"כ סיבוכיות הזמן O(n) כנדרש.
- Get(k): נרצה להחזיר את האיבר שנמצא במקום ה-k במערך. נעשה זאת כך: נבצע את אלגוריתם select(k) כאשר האלגוריתם מחפש את האיבר ה-k, האלגוריתם ישתמש במידע הנוסף של מס' הבנים בכל תת עץ עבור כל צומת פנימי בעץ ה2-3 שלנו. הסיור לוקח סיבוכיות זמן של עומק העץ ולכן O(log(n)) כנדרש.

- Reverse(i,j): נבדוק האם i>j במידה וכן נסיים. אחרת, נבצע Get(i-1) ו-Get(j) בשני העצים גם בעץ המקורי וגם בעץ המראה O(4log(n)).
  נשתמש באלגוריתם Split משאלה קודמת על מנת לפצל כל עץ ל-3 עצים :
  -עץ בעל מפתחות קטנים מi
  -עץ בעל כל המפתחות בין i לj
  -עץ בעל כל המפתחות הגדולים מj.
  (כלומר נשתמש באלגוריתם פעמיים לפי הערכים שמצאנו עבור כל עץ – סה"כ 4 הפעולות O(4log(n))).
  כעת נשתמש באלגוריתם Join משאלה קודמת(כאשר נשלח לו את העץ T1 שאנו רוצים שיהיה השמאלי, וT2 כעץ ימני), ונבצע חיבור על העץ ה"אמצעי" של עץ המראה לשני העצים של הקצוות בעץ המקורי(תוך שמירה על הסדר המקורי שהיה), ואת העץ האמצעי של העץ המקורי נאחד בעזרת האלגוריתם אל תוך העץ ה"מרכזי" בעץ המראה(שמירה על הסדר כנ"ל – זהו החילוף אשר מוסבר בדוגמא למטה).
  נשים לב שגובה העצים זהה ולכן סיבוכיות Join הינה O(log(n)). בנוסף, נבצע שוב את פעולות Get על העץ החדש ונתקן לאורך מסלול החיפוש מהעלים אל השורש, את המידע הנוסף , מס' הבנים בתת העץ על פי הערכים שרשומים בצמתים הפנימיים, נתחיל לבצע את התיקון מעל העלה ונעלה כלפי מעלה (מס' קבוע של פעולות עבור כל רמה)– נשים לב כי לא שינינו את המידע הנוסף בצמתים ה"שמאליים" לi ולאלו ה"ימניים" לj, ולכן תיקון זה מספיק על מנת לתחזק ולתקן את המידע הנוסף ששמור בעץ שלנו. סה"כ קיבלנו O(6log(n)).
  נשים לי כי העצים עדיין מקיימים שהם מראה אחד של השני, כי רק שינינו את טווח הערכים בין i לj, כלומר כעת "הפכנו" פעם אחת את הערכים נקבל ש"תמונת המראה שלהם" היא בדיוק המערך הקודם וכך הלאה.
  סה"כ סיבוכיות הזמן הינה O(log(n)) כנדרש.

איור לדוגמא:

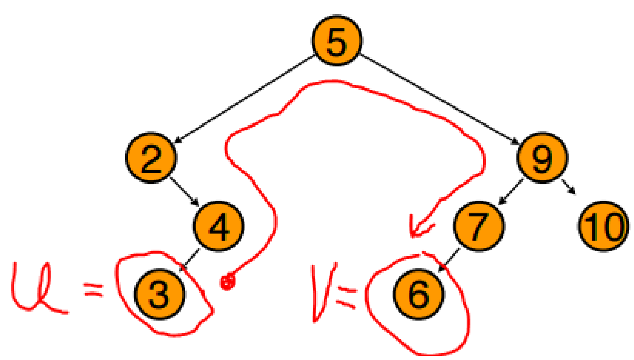


סיבוכיות המקום הינה $O(2n) = O(n)$

<u>שאלה 5</u>

א. לא נכון. נראה טענה חזקה יותר: לא קיים עץ בינארי עם לפחות שני צמתים עם ערכים שונים ולו סיורי pre-order ו-post-order זהים. בכל סיור pre-order השורש של העץ יהיה הראשון בסיור, בעוד שבאף סיור post-order זה לא מתקיים.

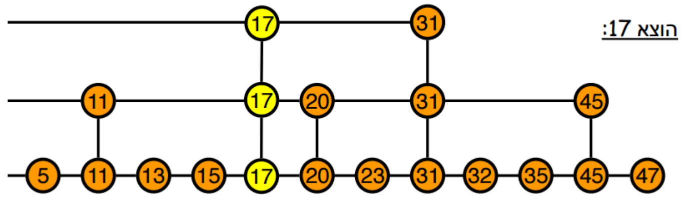
ב. לא נכון, נפריך בעזרת דוגמה נגדית:



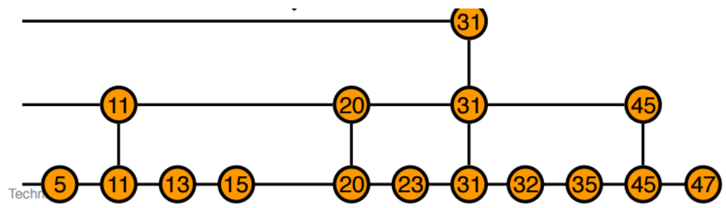
עבור u=3 ו-v=6 מתקיים כי u<v אך הצמתים במסלול בינהם <u>אינם</u> ממויינים לפי מפתחות בסדר עולה (**4,2**,5,9,7)

ג. לא נכון. יהי h הגובה של תת-העצים של v (הגבהים זהים מכיוון שמדובר בעץ 2-3). לכל $0 \neq n \in \{L, M, R\}$ מתקיים $2^h \le n \le 3^h$ (לכל צומת לפחות שני בנים ולכל היותר שלושה בנים), ובנוסף מתקיים $2^h = o(3^h)$. לכן, אם $L = 2^h$ ו-$R = 3^h$ נקבל ש-$L \neq \Theta(L+M+R)$ בכל מקרה.

ד. לא נכון, נפריך בעזרת דוגמא נגדית:



עבור delete(17), ברשימה הנ"ל, נקבל את הרשימה הבאה:



כעת עבור הפעולה insert(17) לא בהכרח נקבל את אותה הרשימה מכיוון שברשימת דילוגים אי-דטרמיניסטית מס' ההופעות של איבר שנוסף בעזרת insert תלוי בהטלות המטבע שנבצע, ולכן במקרה ולא נקבל 3 הטלות מטבע בדיוק של "1" ואז הטלה של "0", הרי שלא יהיו 3 מופעים לאיבר כמו מקודם ומבנה S לא יישמר.

ה. הטענה נכונה.

פעולת insert(x) תכניס לנו איבר חדש למבנה S ומס' המופעים שלו ברמות השונות ייקבע לפי הטלות המטבע כפי שציינו בסעיף הקודם. כעת, בניגוד לסדר הפעולות בסעיף הקודם, פעולת delete(x) מוחקת את כל המופעים של האיבר x במבנה S ללא תלות במס' מופעיו וההטלות שהתקבלו בעת ביצוע פעולת הinsert. כלומר לפני הפעולה insert(x) היה לנו מבנה S ללא האיבר x בתוכו, ומיד לאחר סיום פעולת delete(x) נקבל את אותו המבנה S בדיוק – ללא האיבר S וללא שינוי האיברים האחרים.