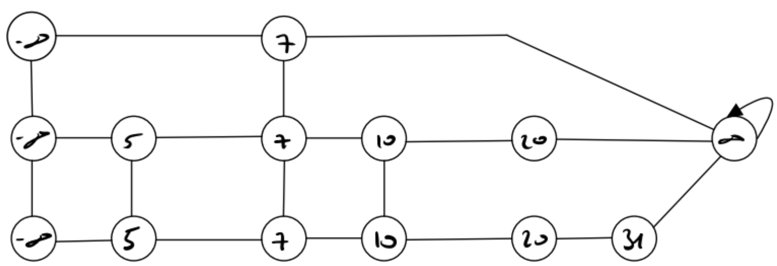
<u>תרגיל יבש 2 במבני נתונים</u>

לירן רדל, 314989427 אנה גריגוריבקר 321931396

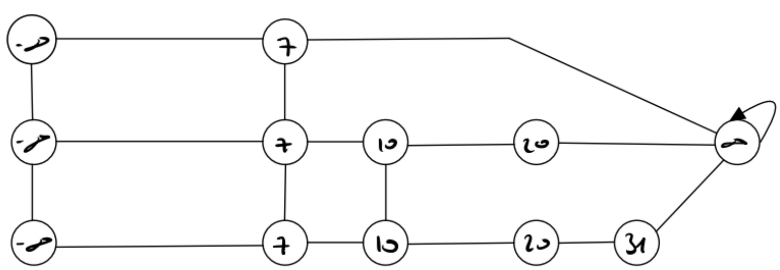
<u>שאלה 1</u>

<u>סעיף 1:</u>

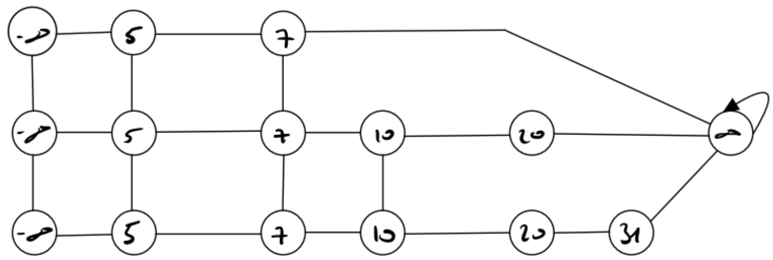
נפריך בעזרת דוגמא נגדית, נתבונן על רשימת הדילוגים:



נוציא את 5 מהרשימה ע"י $delete(5)$:



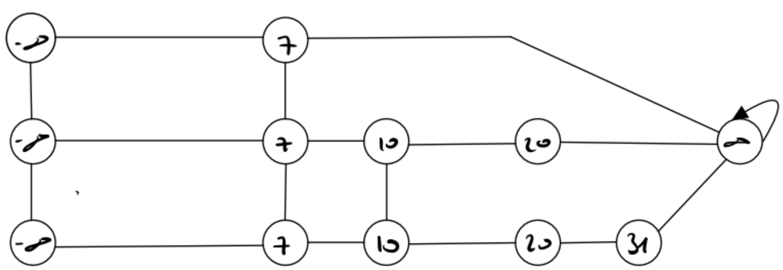
כעת נבצע פעולת $insert(5)$:



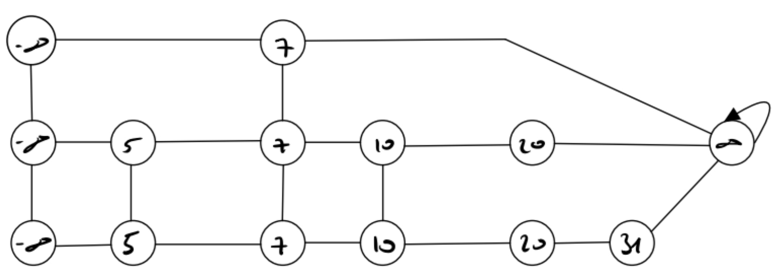
נשים לב שכיוון שבפעולת ההכנסה, ההכנסה לכל רמה מעל הרמה התחתונה נעשת לפי עקרון "הטלת המטבע",
לכן יתכן מצב כמו המצב הנ"ל בו נוציא איבר ולאחר מכן נכניס אותו לכמות שונה של רמות מזו שהוא הופיע בהן לפני ההוצאה.

<u>סעיף 2:</u>

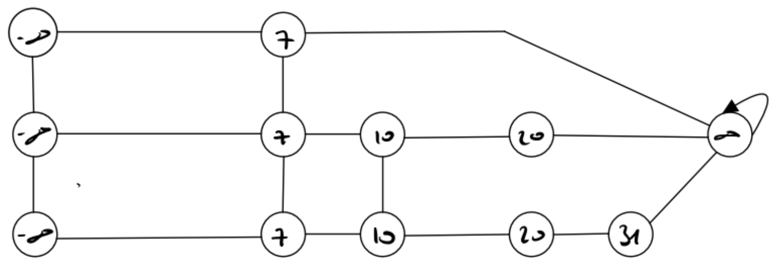
הטענה לא נכונה , נפריך באמצעות דוגמא נגדית:



כעת נבצע פעולת $insert(5)$, נשים לב ש-5 לא מופיע ברשימה:



כעת נבצע פעולת $delete(5)$:



נשים לב כי הרשימה לא השתנתה ממצבה ההתחלתי, בנוסף, לאורך ביצוע הפעולות שמרנו על רשימת דילוגים תקינה ולכן הפרכנו את הטענה כנדרש.

סעיף 3:

טענה לא נכונה, נפריך באמצעות דוגמה נגדית, נקח T_L - עץ פיבונצ'י בגובה h ראינו בהרצאה כי מדובר בעץ AVL
T_R - עץ בינארי מלא בגובה h, גם עץ AVL תקין.
T_L ו-T_R עצי AVL באותו הגובה ולכן העץ שהם בניו גם הוא עץ AVL תקין.
ראינו בהרצאה כי $|T_L| = \Theta\left(\phi^h\right)$ וכי $|T_R| = \Theta(2^h)$, מכאן נסתכל על היחס בין הביטויים כאשר $h \to \infty$

$$\lim_{h \to \infty} \frac{\phi^h}{2^h} \overset{*}{=} 0 \qquad * \ \phi < 2$$

מכאן נקבל ש - $|T_L| = \ o(|T_R|)$ ולכן $|T_L| \neq \Omega(|T_R|)$ ובפרט $|T_L| \neq \Theta(|T_R|)$.

סעיף 4:

נגדיר את מבנה הנתונים בצורה הבא:

עץ $Trei$ בו כל מחרוזת הינה תיאור של מוצר, כל צומת בעץ הינה $node$ אשר מכיל שדה של מחיר ,שדה של אות, שדה בו מצביע לאב ושדות של מצביעים לבנים(א"ב סופי ולכן מספר השדות סופי).
בנוסף נחזיק מכסנית בה נשמור את המחרוזות המייצגות את תיאורי המוצרים.

פירוט הפעולות:
$Init()$ - נאתחל עץ ריק עם ערכים דיפולטיביים בשדות סה"כ $O(1)$.

$Add(p, desc)$ - במידה וקיימת רישא של תיאור המוצר בעץ, נתקדם לאורך הרישא ולא נעדכן את המחיר באיברים של הרישא,במידה ונגיע לתו המסמל סיום מחרוזת נחזיר שגיאה שהמחרוזת קיימת כבר, אחרת, נוסיף איברים בהתאם , בכל איבר שנוסיף (כולל איבר המסמל סיום מחרוזת) נשים בשדה המחיר את הערך p.
לאחר מכן, נחזור באותה הדרך לאורך $desc$, נגיע לאיבר הראשון במסלול החזרה שיש לו יותר מבן אחד, נבדוק אם המחיר שנמצא באיבר שבפיצול גבוה יותר מ-p, אם כן נעדכן אותו להיות p.
בנוסף נבצע פעולת $push(desc)$ למכסנית, כפי שלמדנו בתרגול זה יתבצע ב-$O(1)$.

סה"כ נעבור על עד $2|desc|$ איברים ונבצע מספר קבוע של פעולות בנוסף למעברים, קיבלנו סיבוכיות של $O(|desc|)$.

$Undo()$ - נבצע פעולת $pop()$ על המחסנית ($O(1)$), נקבל את תיאור המוצר האחרון שהוכנס לעץ. נבצע מעבר בעץ $Trei$ לפי תיאור המוצר עד שנגיע לתו שמסמן את סוף המחרוזת, כעת, נמחק איברים עד שנגיע לאיבר שיש לו בן (לאחר שמחקנו את הבן שלו ששייך לתיאור המוצר אותו מסירים) , שם נפסיק את המחיקה ונסיים את הפעולה.
סה"כ עברנו במקרה הגרוע ב-$2|desc|$ איברים ובכל איבר ביצענו מספר קבוע של פעולות, סיבוכיות של $O(|desc|)$.

$Price(desc)$ - נבצע מעבר בעץ $Trei$ עד שנגיע לאיבר שמסמן את סוף המחרוזת התואמת את תיאור המוצר, נחזיר את הערך שמוחזק בשדה של המחיר, סיבוכיות של $O(|desc|)$.

$CheapestOfType(s)$ - נבצע מעבר בעץ $Trei$ עד שנגיע לאיבר האחרון ביצוג של s, נבדוק מה הערך שמוחזק בשדה המחיר שלו ונחזיר את הערך הזה. (דאגנו שהוא תמיד יעודכן למוצר הזול ביותר כשהכנסנו מוצר) , סה"כ סיבוכיות של $O(|s|)$.

שאלה 2

סעיף א:

ראשית נמצא את גבהי העצים, פעולה זו לפי ההרצאה תעשה בסיבוכיות של $O(\max\{\log(n_1),\ \log(n_2)\})$
נסמן את הגבהים h_1, h_2 בהתאמה.
נמצא את האיבר המקסימלי ב-T_1 נסמן את ערכו ב- x_1, נמצא את האיבר המינימלי ב-T_2 נסמן את ערכו ב-x_2, פעולות אלו נעשות גם כן ב- $O(\max\{\log(n_1),\ \log(n_2)\})$.
נחלק למקרים:

- $h_1 = h_2$:
ניקח את האינדקס המינימאלי של T_2 - x_2 , ניצור צומת שתכיל את האינדקס הזה. נקבע את השורש של T_2 כבן ימינה שלה ואת השורש של T_1 כבן שמאלי שלה. נשים לב כי כל העלים בעץ החדש נותרו באותו הגובה כנדרש, בנוסף , לצומת החדשה שיצרנו ישנם 2 בנים סה"כ כאשר המפתח היחיד בצומת קטן שווה למפתחות בתת העץ הימני וגדול ממש מהמפתחות בתת העץ השמאלי, קיבלנו עץ 2-3 תקין.
- $h_1 \neq h_2$: נניח בה"כ $h_2 < h_1$
נרצה לבצע הכנסה של העץ בעל הגובה הנמוך יותר - T_2 לעץ T_1.
נוסיף את x_2 בתור מפתח לצומת הימני ביותר בגובה $h_2 + 1$ בעץ T_1, נמצא את הצומת ע"י חיפוש בעץ 2-3 כפי שלמדנו בהרצאה. נוסיף את x_2 בתור מפתח בצומת זאת ונשרשר את השורש של T_2 להיות בן ימני שלה, במידה ובצומת לפני הוספת x_2 היה רק מפתח אחד (כלומר 2 בנים) סיימנו. במידה והיו 2 מפתחות (3 בנים) נבצע פעולות פיצול על העץ כפי שלמדנו בהרצאה, במקרה הגרוע

נגיע עד לגובה של שורש העץ + 1 כלומר נעבור ב $h_1 + 1 \; - h_2 + 1 < h_1$ צמתים , בכל צומת נבצע מספר קבוע של פעולות, לכן הסיבוכיות היא $O(\log(n_1))$.
בשני המקרים דאגנו שהעלים ישארו באותו הגובה, בנוסף לפי נכונות פעולת הפיצול שלמדנו בהרצאה נקבל שלכל צומת ישנם בין 2-3 בנים, קיבלנו עץ 2-3 תקין.
סה"כ סיבוכיות של $O(\max\{\log(n_1),\ \log(n_2)\})$.

סעיף ב:

נבצע את הפעולה באופן דומה לסעיף א' אך ללא הצורך למצוא את גובה העצים ואת הערך המינימלי/מקסימלי בכל אחד.
נחלק למקרים:

- אם $h_1 = h_2 = h$ ניצור איבר ובו מפתח שערכו הערך המינימלי של T_2 , נשים את העצים להיות הבנים שלו וסיימנו, סה"כ $O(h - h + 1) = O(1)$.
- אם $h_1 \neq h_2$, נניח בה"כ $h_2 < h_1$, נשים את T_2 להיות הבן הימני בצומת הימני ביותר בגובה $h_2 + 1$ בT_1, נוסיף את הערך המינמלי בT_2 כמפתח בצומת הזאת. אם לא ידרשו פיצולים, סיימנו. אחרת, יבוצעו מספר פיצולים החסום ע"י עומק הצומת $h_1 - h_2 + 1$.
  סה"כ סיבוכיות של $O(|h_2 - h_1| + 1)$.

סעיף ג:

לצורך פתרון סעיף זה נשתמש בפתרון שהצענו לסעיף ב', נתחיל באיחוד שני העצים בעלי הגובה הנמוך ביותר ונמשיך כך עד שנשאר עם עץ אחד, נשים לב כי חיבור של שני עצים בעלי אותו גובה , נסמן אותו ב $h -$ יחזיר עץ בגובה $h + 1$, ע"מ להראות כי אנו עומדים בסיבוכיות הנדרשת נוכיח טענת עזר:

$$|h_{i+1} - h_i'| + 1 = \begin{cases} h_{i+1} - h_i' + 1 & h_{i+1} \geqslant h_i' \\ |-1| + 1 & h_{i+1} < h_i' \end{cases} \quad 1 \leqslant i \leqslant k-1 \quad h_i'$$

- גובה העץ שהתקבל לאחר איחוד של i עצים.

המקרה הראשון טריוויאלי, נוכיח את המקרה בו $h_{i+1} < h_i'$ באינדוקציה
נגדיר טענה, אם מתקיים $h_{i-1}' > h_i$ אז:

$$h_{i-1}' \leqslant h_i + 1$$

$$h_{i-1}' - hi \leqslant 1$$

בסיס האינדוקציה:

מס' איטרציה	h'_{i-1}	h_i	חסם גובה העץ שיתקבל מאיחוד של h'_{i-1} ו-h_i	$h'_{i-1}-h_i$
$i=1$	0	h_1	h_1	0
$i=2$	h_1	h_2	h_2+1 (1)	-1
$i=3$	h_2+1	$h_3=h_2$	h_2+1 (2)	1
$i=3'$	h_2+1	$h_3>h_2$	h_3+1 (1)	$\leqslant 0$
$i=4$	h_2+1	$h_4>h_3=h_2$ (3)	h_4+1 (1)	$\leqslant 0$
$i=4'$	h_3+1	$h_4=h_3$	h_3+1 (2)	1

(1) מקרה בו נקבל עץ שגובהו גבוה ב-1 מגובה העץ הגבוה יותר באיחוד, כלומר התווספה רמה לעץ, במקרה זה לשורש יהיו 2 בנים בדיוק.

(2) לשורש היו 2 בנים לפי (1) לכן אין צורך להוסיף רמה נוספת.

(3) לפי הנתון, לא ייתכן מצב בו יש שלושה עצים באותו הגובה.

צעד:

נניח נכונות של הטענה עבור איחוד של i עצים באותו האופן, ונוכיח אותה עבור איחוד עם העץ ה-$i+1$, כלומר נרצה לאחד עץ בעל גובה $h_i{}'$ עם העץ T_{i+1} שגובהו h_{i+1}.

מהנחת האינדוקציה נקבל:

$$h_i - h'_i \leqslant 1$$
$$h_i \geqslant h'_i - 1$$

בנוסף מהנתון ידוע כי:

$$h_{i+1} \geqslant h_i$$

מכאן נקבל:

$$h_{i+1} \geqslant h_i \geqslant h'_i - 1$$
$$h_{i+1} \geqslant h'_i - 1$$
$$h_{i+1} - h'_i \geqslant -1$$
$$h'_i - h_{i+1} \leqslant 1$$

אז בסה"כ:

$$1 \leqslant i \leqslant k-1 \quad |h_{i+1} - h_i{}'| + 1 = \begin{cases} h_{i+1} - h_i{}' + 1 & h_{i+1} \geqslant h_i{}' \\ |-1| + 1 & h_{i+1} < h_i{}' \end{cases}$$

כעת נבצע את איחוד העצים מהעץ בעל הערכים הנמוכים ביותר, העץ בעל אינדקס 1 ועד העץ ה-k לפי האלגוריתם בסעיף ב'.
בהנחה שבכל איטרציה i מתקיים $h_{i+1} \geqslant h_i'$, אז נקבל שסה"כ:

$$(h_k - h_{k-1}) + 1 + (h_{k-1} - h_{k-2}) + 1 + . . . + (h_2 - h_1) + 1 = h_k - h_1 + k = O(h_k - h_1 + k)$$

במידה ובכל איטרציה מתקיים $h_{i+1} < h_i'$ נקבל:

$$k \cdot (|-1| + 1) = O(k)$$

מכאן עבור כל שילוב של שני המקרים הנ"ל, נבצע k איטרציות שסיבוכיותן הכוללת קטנה מלבצע את שני המקרים הנ"ל בכל איטרציה, לכן נקבל:

$$T(k) \leqslant (h_k - h_{k-1}) + 1 + (h_{k-1} - h_{k-2}) + 1 + . . . + (h_2 - h_1) + 1 + k \cdot (|-1| + 1) =$$

$$h_k - h_1 + 3k = O(h_k - h_1 + k)$$

<u>סעיף ד:</u>

ע"מ לממש את הפעולה נבצע חיפוש של המפתח x בעץ לפי חיפוש רגיל בעץ 2-3 , כאשר בכל רמה אותה נעבור:

- אם הגענו לעלים - נסיים.
- אחרת - x נמצא באחד מתתי העצים של הצומת הנוכחי.

- נקח את הבנים שמשמאל לבן שאליו אנו יורדים בהמשך החיפוש, נסמן אותם $T_{1,2}$, $T_{1,1}$. בנוסף נסמן את המפתח שאיתו השוונו במציאת הבנים השמאליים ביותר ונכנה אותו $k_{1,1}$. נשים לב שבשני העצים האלו כל האיברים קטנים מ-x .

- נקח את הבנים שמימין לבן שאליו אנו יורדים בהמשך החיפוש. נסמן אותם $T_{2,1}, T_{2,2}$. בנוסף נסמן את המפתח שאיתו השוונו במציאת הבנים הימניים ביותר ונכנה אותו $k_{2,1}$. נשים לב שבשני העצים האלו כל האיברים גדולים מ-x.

נמשיך בתהליך עד שנגיע ל-x, נקבל סדרת עצים עם לכל היותר 2 עצים לכל רמה בה עברנו בחיפוש, סה"כ שתי סדרות כאלה, אחת בה לכל העצים ערכים גדולים או שווים ל-x ואחת בה לכל העצים ערכים קטנים מ-x.
כעת נשים לב שמכיוון שמדובר בעץ 2-3 כל שני עצים סמוכים בסדרה הנ"ל יכולים להגיע מאותה רמה, ואז הם באותו הגובה, או מרמות עוקבות ואז בהכרח העץ הבא בסדרה גבוה ב-1 מהעץ הנוכחי , בנוסף, לא יתכנו 3 עצים בעלי אותו גובה , בהתאם להנחה של סעיף ג'.
כעת נוכל לאחד כל סדרה כזאת של עצים לפי האלגוריתם בסעיף ג', נשים לב שמספר העצים בכל סדרה הוא

לכל היותר $2\log n$ (2 עצים בכל רמה), מכאן הסיבוכיות של ביצוע איחוד שתי סדרות העצים שקיבלנו תהיה:

$$T(n) = O(h_k - h_1 + k) + O(h_k - h_1 + k) = 2 \cdot O(\log n - 0 + 2\log n) = O(\log n)$$

שאלה 3

סעיף א:

נגדיר את המבני הנתונים הבא על מנת לפתור בעיה זו:

- עץ דרגות AVL שייצג את נקודות ההתחלה של כל קטע כאשר הצמתים בעץ ימויינו לפי int, ובנוסף כל צומת תחזיק שדה שישמור את מספר הצמתים של תת העץ הימני, השמאלי ועצמו יחדיו, ושדה שימנה את מספר המופעים של נקודת התחלה זו למידה ויש מספר קטעים המתחילים באותה נקודה.
- עץ דרגות AVL שייצג את נקודות הסיום של כל קטע כאשר הצמתים בעץ ימויינו לפי int, ובנוסף כל צומת תחזיק שדה שישמור את מספר הצמתים של תת העץ הימני, השמאלי ועצמו יחדיו, ושדה שימנה את מספר המופעים של נקודת הסיום זו למידה ויש מספר קטעים המתחילים באותה נקודה.
- שדה שיכיל את k.
- 2 שדות שנקרא להן centric_start, centric_end שיכילו את קצוות הקטע שיעניין אותנו ויתעדכנו עם האלגוריתם.

הסבר הפונקציות ומימושן:

$Init(k)$: נאתחל 2 עצים ריקים ב-$O(1)$, את השדה שמכיל את k ל-k, ואת 2 השדות הנוספים ל-0. בסה"כ סיבוכיות של $O(1)$.

$Insert((x, y))$: ראשית נחפש את x בעץ של נקודות ההתחלה בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה. אם נמצאה, נגדיל ב-1 את השדה של מספר המופעים של נקודה זו ונעדכן את גודל השדה של מספר הצמתים בתת-העץ של כל צומת אחורה במסלול בחיפוש ב-1. אחרת, נכניס את x אל העץ ונעדכן את הדרגה תחת כל צומת אחורה במסלול החיפוש כפי שנלמד בהרצאה, ולאחר ההכנסה תעדכן את השדה של centric_start על ידי מציאת נקודת ההתחלה שלפנייה או איתה ביחד קיימות k נקודות התחלה על ידי מציאת הצומת בעלת הדרגה הקרובה ל-k או בעלת דרגה זו בסיבוכיות של $O(logn)$ מאחר וזהו חיפוש לאורך מסלול החיפוש. נבצע באופן דומה אותו אלגוריתם עבור עץ קטעי הסיום ונעדכן בהתאמה את centric_end. כל אחד מהם בסה"כ $O(logn)$ פעולות ולכן הסיבוכיות הינה בסה"כ $O(logn)$.

$Delete((x, y))$: ראשית נחפש את x בעץ של נקודות ההתחלה בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה. אם לא נמצאה נחזיר שגיאה, ואם נמצאה, אם מספר המופעים שלה גדול מ-1 נחסיר ב-1 את מספר המופעים שלה ונעדכן את גודל השדה של מספר הצמתים בתת-העץ של כל צומת אחורה במסלול החיפוש

בסיבוכיות של $O(logn)$.
אחרת(אם שווה ל-1), נסיר את הצומת מהעץ בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה ונעדכן באותו האופן את מספר הצמתים לאורך מסלול החיפוש אחורה בסיבוכיות של $O(logn)$, ולבסוף באותו האופן שתואר בפעולת Insert
נעדכן את centric_start ב-$O(logn)$. נבצע באופן דומה אותו אלגוריתם עבור עץ קטעי הסיום ונעדכן בהתאמה את centric_end. כל אחד מהם בסה"כ $O(logn)$ פעולות ולכן הסיבוכיות הינה בסה"כ $O(logn)$.

$IsCentric((x,y))$: אם x⩾centric_start וגם y⩽ centric_end אז נחזיר אמת, אחרת שקר. מדובר בפעולות בודדות ולכן בסה"כ $O(1)$ כנדרש.

סעיף ב:

בסעיף זה בניגוד לסעיף קודם, מעניין אותנו רק את ערכי x בפעולת Insert של הקטע, כלומר רק התחלות הקטעים שאנו מכניסים למערכת כי אילו ההשוואות היחידות שאנו מבצעים. לכן נגדיר את מבני הנתונים כך שיהיה מורכב מעץ דרגותAVL שייצג כפי שתיארנו את התחלות הקטעים, כאשר כל צומת תכיל את השדות הנוספים הבאים חוץ מהשדה הנוסף לחישוב הדרגה ומהמפתח: שדה של כמה מופעים של הקטע נמצאים בקבוצה A, שדה של כמה מופעים של הקטע נמצאים בקבוצה B, שדה של כמה קטעים יש בתת העץ של צומת זה בקבוצה A, שדה של כמה קטעים של תת העץ של צומת זה נמצאים בקבוצה B ומצביע לרשימה שכל איבר בה הוא מחזיק קטע זה.

<u>הסבר הפונקציות ומימושן:</u>

$Init()$: נאתחל עץ ריק ב-$O(1)$.

$Insert((x,y),G)$: ראשית נחפש את האיבר בעץ בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה. אם קיים, נעדכן את השדה של כמה קטעים כאלו קיימים בקבוצה G ב-1 ובנוסף את השדה של מספר המופעים של קטע זה בתת העץ בקבוצה G ב-1. לאחר מכן נעבור על מסלול החיפוש ונחסר כל פעם את מספר הקטעים החופפים מ-A ומ-B בעזרת השדות הנוספים שתיחזקנו(מספר הקטעים שבתת העץ של A פחות מספר הקטעים בתת העץ של B), אם גדול מ- 0 נפנה ימינה ואם קטן מ-0 נפנה שמאלה בעץ ולכן עדיין נשמרת סיבוכיות של $O(logn)$, ואם במהלך אחורה במסלול החיפוש מצאנו הפרש ששווה ל-0, נשים את המצביע אליו ברשימה מאחר ומצאנו קטע המתאים(במהלך כל חיפוש מחדש קטע זה עשויי להתעדכן).

$InA=InB()$: נפנה אל המצביע ששמרנו, אם קיים אז נחזיר קטע זה ב-$O(1)$, ואם לא נחזיר null.

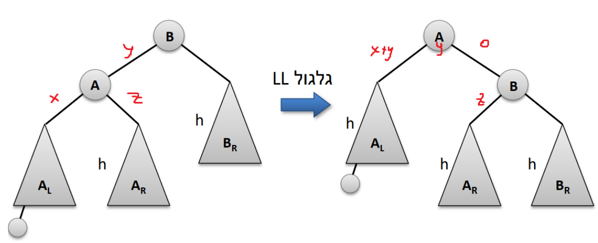
שאלה 4

סעיף א:

נגדיר את המבני נתונים הבא על מנת לפתור בעיה זו:
- עץ כמעט שלם AVL הממויין לפי int שהוא ישמש אותנו כמפה עבור האנשים במערכת, כלומר המפתח יהיה ה-id שלהם וה-value יהיה שנת הלידה שלהם
- עץ כמעט שלם AVL הממויין לפי int כאשר מפתח המיון הוא לפי שנים, וכל צומת בעץ תכיל 3 שדות נוספים - self weight, right weight, left weight אשר נסביר עליהם בהמשך(אפשר להתייחס לright/left weight כמן שדה על הקשתות בעץ).

<u>הסבר הפונקציות ומימושן:</u>
$init()$: נאתחל 2 עצים ריקים ב-$O(1)$.

$AddPerson(id,\ year)$: ראשית, נכניס את האדם לעץ לפי id, כאשר המפתח הוא ה-id(שידוע כי הינו יחודי), ונשים ב-value את השנה שלו. כפי שנלמד בהרצאה פעולת הכנסה זאת כולל טיפול בגלגולים קוראת ב-$O(1)$.
לאחר מכן נחפש את $year$ בעץ השנים בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה(הסבר על כמות האיברים בעץ בסוף התיאור של סעיף זה). אם מצאנו אותו, אז אין צורך להוסיפו שוב וסיימנו את הפונקצייה בסה"כ של $O(logn)$.
אחרת, נוסיף את $year$ לעץ השנים ב-$O(logn)$ כפי שנלמד בהרצאה, עם הערכים Self.Left,Right Weight מאותחלים לאפס. הדבר היחידי שישאר לנו לטפל זה הגלגולים. נסתכל על גלגול LL בצומת החדשה שהכנסו ועל האב שלה:



כפי שניתן לראות בציור, על מנת שהאלגוריתם שלנו לא ייפגע, כאשר נבצע גלגולים לאיזון העץ, אל הSelf.W של הצומת A נוסיף את ה-Left.W של הצומת B, את הRight.W של A נעביר אל ה-Left.W של B, ל-Left.W של A נוסיף את הLeft.W של B, וה-Right.W החדש של A יהיה 0. גלגול RR ייצבע אותו הדבר רק בצורה הפוכה(גלגול סימטרי) ושאר הגלגולים LR, RL ניתן לבצע בעזרת LL,RR כפי שנלמד. כמות הפעולות בכל גלגול הינה סופית ולכן אינה פוגעת באלגוריתם, לכן בסה"כ הפעולה לוקחת $O(logn)$ כנדרש.

$IncreaseGrant(y1,\ y2,\ a)$: ראשית, נוסיף את $y1,\ y2$ לעץ השנים שלנו(אם אינם קיימים), ונטפל בגלגולים לאחר ההכנסה של שניהם באותו האופן שבו טיפלנו בגלגולים בפונקציית $AddPerson$. לאחר מכן נבצע את האלגריתם הבא:
ראשית נחפש את הצומת הראשונה בעץ שתהייה בטווח שלנו על ידי השוואה החל מראש העץ האם הוא גדול מ-y1 וגם קטן מ-y2. אם גדול מ-y2, נפנה שמאלה בעץ, ואם קטן מ-y1 נפנה ימינה, עד שנמצא את הצומת הראשונה שבטווח. לאחר מציאתו, נוסיף אל ה-Self.W שלו את הערך a ולאחר מכן נתפצל פעם אחת בלבד ל-2 אלגוריתמים לבן הימני והשמאלי שלו(נסביר את האלגוריתם הימני, השמאלי על אותו העקרון רק הפוך בפניות כלומר נבדוק האם כל צומת גדולה מ-y1 ובמקום לשנות את Left.W נשנה את Right.W). לאחר הפנייה ימינה נסתכל על הצומת ונבדוק האם היא קטנה מ-y2. אם כן, וודאי כי תת-העץ השמאלי שלה גם בטווח מאחר וזהו עץ חיפוש, ולכן נוסיף אל Self.W שלה את הערך a, וגם אל Left.W את הערך a, ונמשיך בחיפוש אל הבן הימני. אם אינו בטווח, אז נלך אל הבן השמאלי בעץ ונמשיך את האלגוריתם, מאחר וייתכן כי הבן השמאלי בתחום. כך נמשיך את האלגוריתם עד שנגיע אל עלה (גם במקרה שאין בן שמאלי והצומת בתחום נשנה את Self.W ואת Left.W באותה המידה). אנחנו מטיילים על $log(n)$ איברים בסה"כ. בסה"כ אנחנו מבצעים פיצול אחד לכן במקרה הגרוע אלגוריתם זה ייקח לנו
$O(2logn)$, ולכן בסה"כ פעולה זו לוקחת לנו $O(logn)$ כנדרש.

$GetGrantAmount(id)$: ראשית נחפש את הצומת של האדם בעץ לפי id לפי ה-id שניתן לנו ב-$O(logn)$ כפי שנלמד בהרצאה, וניקח משם את ה-value שלו שזה השנה שבה נולד. לאחר מכן נחפש את value בעץ של השנים ב-$O(logn)$ ונבצע במהלך מסלול החיפוש את האלגוריתם הבא שיתבצע על ידי סכימה: אם פנינו ימינה בעץ, נכניס אל הסכימה את Right.W של הצומת שממנה פנינו, ואם נפנה שמאלה נסכום את Left.W של הצומת שממנה פנינו, וכשנגיע אל הצומת שמכילה את השנה אותה אנו מחפשים(value) נסכום את Self.W שלה, וזהו המענק שמגיע למי שנולד באותה השנה ונחזירו. לאורך מסלול החיפוש אנו מבצעים $O(1)$ של פעולות והחיפוש כפי שהוסבר קורה ב-
$O(logn)$ ולכן בסה"כ סיבוכיות של $O(logn)$ כנדרש.

*מאחר ובעץ השנים אנו מוסיפים גם איבר כל פעם שבנאדם נוסף למערכת וגם 2 איברים כל פעם שעושים את פעולת increase, כלומר גודל העץ של השנים תלויי גם במספר האנשים וגם במספר המענקים שניתנו(על כל מענק נוספים במקרה הגרוע 2 איברים לעץ), ולכן כאשר אני אומר n אני מתכוון למספר המקסימלי בין מספר המענקים שניתנו למספר האנשים מה שמסתדר עם הסיבוכיות בגלל בקשר הישיר.

<u>סעיף ב:</u>

על מנת לטפל בכך שכעת כל אדם ייקבל את המענק שמגיע לו רק לאחר הצטרפותו למערכת, נעשה כמה שינויים קטנים על מנת לתמוך באפשרות זו:
במבני הנתונים שלנו בעץ לפי id נוסיף שדה לכל צומת בנוסף לשנה שבה באדם נולד, שנקרא לה amount_owned.
בפעולת $AddPerson$ כאשר נוסיף אדם, לאחר הוספתו אף העצים המתאימים נחשב בנוסף גם את המענק שמגיע לשנה בה נולד כאשר הצטרף באותו אופן שמחושב בפונקציית $GetGrantAmout$, ואת המענק הזה

נכניס עם מינוס אל הערך amount_owned של האדם בעץ לפי id (פעולה זו לוקחת $O(logn)$ לכן לא פוגעת בסיבוכיות), וכאשר נחשב את $GetGrantAmount$ נוסיף בחישוב בסוף את amount_owned אל הסכום שאנו סוכמים(מציאתו לוקחת $O(logn)$ מה שאינה פוגעת בסיבוכיות), כנדרש.

סעיף ג:

נוסיף אל המבני הנתונים שלנו מערך בגודל K/10 שיעזור לנו לניווט בין המערכים ושדה שיכיל את k.

תיאור הפונקציות:

$Init(k)$: נאתחל מערך בגודל K/10 בסיבוכיות של $O(1)$ כפי שנלמד בהרצאה (ונכניס לשדה המכיל את k את k), וסיבוכיות המקום של זה הינה $O(k)$.

$GetLargestInDecade(d)$: על מנת לתמוך בפונקצייה זו בסיבוכיות הנדרשת, כל פעם שנבצע את הפעולה $IncreaseGrant(y1,\ y2,\ a)$, ראשית נבדוק האם קיימים מערכים בגודל K/10 שיכיל עבור כל שנה בעשור של y1, y2 לחישוב מי מהעשור קיבל את המענק הכי גדול על ידי חיפוש במערך העזר שלנו לפי הנוסחא הבאה: $(y1-k)\%10$
את התא במערך(ללא השארית החלוקה), ונבדוק לפי מה שנלמד בהרצאה האם קיים שם ערך זבל. אם כן ניצור מערך לעשור זה בגודל 10 ונשים שם מצביע למערך זה, ואם לא אז נפנה למערך זה ועבור השנים החל מ-y1 עד לסוף העשור שלו נוסיף במערך את הערך a(כל מערך בגודל 10 שמייצג עשור מייוצג ככה שבתא הראשון זה לדוגמא 1990, השני 1991 וכך עד 1999), ועבור y2 נמצא באותו אופן את המערך של העשור שלו או ניצורו ונוסיף אל התאים החל מ_y2 ועד לתחילת העשור שלו את הערך a(יש לציין כי המערכים בהתחלה תמיד יהיו מאותחלים ל-0), ולא ניגע בעשורים בין השנים y1 ל-y2 מאחר ועל מנת לדעת מי קיבל את המענק הכי גדול מעניין אותנו מי קיבל הכי הרבה, כלומר את ההפרשים ביניהם ולא בהכרח את גודל המענק לכן לא נשנה כלום. זוהי תוספת של $O(1)$ פעולות אל הפונקצייה ולכן אינה משנה את סיבוכיותה.
כעת כאשר נקרא את הפונקצייה $GetLargestInDecade(d)$ נחפש כפי שהסברנו את מערך העשור d במערך הראשי שלנו, ונשווה בין כל התאים במערך(יש 10 לכן פעולות בודדות) ונחזיר את השנה בעלת הערך הכי גדול שהיא בעלת המענק הכי גדול(או אם יש כמה שווים אחד מהם). (אם יהיה ערך זבל סימן שאף אחד לא קיבל מענק בעשור זה לכן פשוט נחזיר את d לדוגמא). כל הפעולות הינן בודדות לכן פונקצייה זו קוראת בסיבוכיות של
$O(1)$ כנדרש.