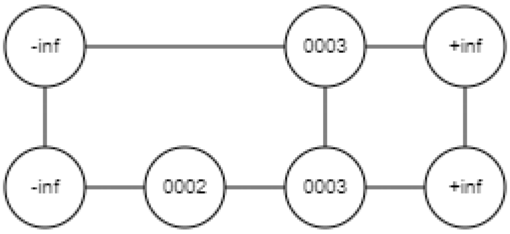
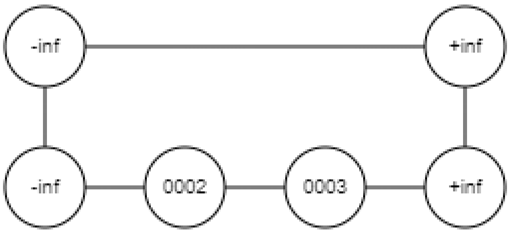
שאלה 1

סעיף 1 - הפרכה

לדוגמא: ניקח את רשימת הדילוגים הבאה:



וכעת נבצע הסרה של המספר 3. לאחר מכן, נוסיף את המספר 3. כעת, יש הסתברות שנוסיף את 3 נקבל את המצב הבא:

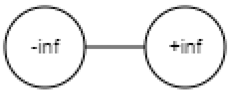


כלומר, נסיק כי הרשימה עלולה להשתנות.

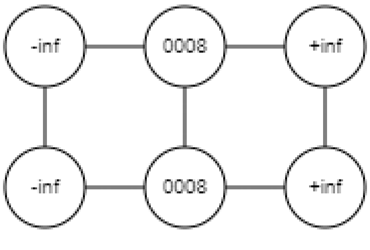
סעיף 2 - הפרכה

נניח כי במידה וקיימת שורה ריקה ברשימת הדילוגים (הכוללת את $\infty, -\infty$ בלבד), אזי היא לא נמחקת.

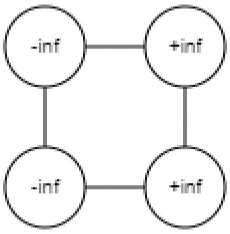
נבצע אתחול לרשימת דילוגים ונקבל:



לאחר מכן, נבצע את הפקודה $insert(8)$ ונקבל את רשימת הדילוגים הבאה:



כעת, לאחר הפקודה $delete(8)$ ונקבל את רשימת הדילוגים הבאה:



נשים לב כי שתי רשימות הדילוגים הם שונות.

סעיף 3 - הפרכה

ראשית נסמן T_h להיות עץ בגובה h לצורך נוחות. ניקח סדרת עצי AVL להלן: תת העץ הימני של העץ ה-T_h הינו עץ מלא בגובה $h-1$ ותת העץ השמאלי הוא עץ מסוג פיבונאצ'י שגובהו $h-1$.

כמו כן, אנו יודעים כי כמות הצמתים של תת העץ הימני הוא $|T_{R,h-1}| = 2^{h-1}-1$. בנוסף, גובה של תת העץ השמאלי הוא $|T_{L,h-1}| = n_{h-1}-1 = \frac{(\phi)^{h+1}-(\bar{\phi})^{h+1}}{\sqrt{5}}-1$.

מתקיים:

$$\begin{aligned}
\frac{|T_{L,h-1}|}{|T_{R,h-1}|} &= \frac{\frac{(\phi)^{h+1}-(\bar{\phi})^{h+1}}{\sqrt{5}}-1}{2^{h-1}-1} \overset{(*)}{<} \frac{\frac{2\cdot\phi^{h+1}}{\sqrt{5}}}{2^{h-1}-1} \\
&= \frac{2}{\sqrt{5}}\cdot\frac{\phi^{h+1}}{2^{h-1}-1} \\
&= \frac{2\phi^3}{\sqrt{5}}\cdot\frac{\phi^{h-2}}{2^{h-1}-1} \\
&< \frac{2\phi^3}{\sqrt{5}}\cdot\frac{\phi^{h-2}}{2^{h-2}} \\
&= \frac{2\phi^3}{\sqrt{5}}\cdot\left(\frac{\phi}{2}\right)^{h-2} \overset{h\to\infty}{\longrightarrow} 0
\end{aligned}$$

סעיף 4 - הפרכה

ניקח 2 עצי $2-3$ המקיימים $D\left(n\right) \neq O\left(1\right)$.

יהי עץ $2-3$ המכיל בכל צומת 2 בנים. במידה ו-n הוא זוגי נכניס צומת לצומת המהווה את הרמה הלפני האחרונה. נשים לב שבאופן זה גובה עץ זה הוא $\log_2\left(n\right)$.

כעת, ניקח עץ אחר המכיל בכל צומת 3 צמתים ובמידה ו-n לא מתחלק ב-3 נכניס את מספר הצמתים (כגודל שארית החלוקה של n ב-3) לעץ ברמה הלפני תחתונה. באופן זה נקבל שגובה עץ זה הוא $\log_3\left(n\right)$.

לפי הגדרה מתקיים:

$$\begin{aligned} D(n) &= \log_2(n) - \log_3(n) \\ &= \frac{\log_3(n)}{\log_3(2)} - \log_3(n) \\ &= \frac{\log_3(n) - \log_3(n)\log_3(2)}{\log_3(2)} \\ &= \frac{\log_3(n)(1-\log_3(2))}{\log_3(2)} \\ &= \frac{\log_3(n)\cdot 0.36907}{\log_3(2)} = O(\log n) \neq O(1) \end{aligned}$$

לכן, נסיק כי $D(n) \neq O(1)$.

סעיף 5 - הוכחה

הבחנה חשובה: נשים לב כי בכל צומת אנו עוברים לכל היותר 3 פעמים - נסתכל על המקרה בו אנו עוברים בצומת הנוכחית את מספר הפעמים הגדול ביותר:

- תחילה מגיעים מצומת האב לצומת הנוכחית, ממנה לתת העץ השמאלי.
- נעלה מתת העץ השמאלי לצומת הנוכחית ולאחר מכן נרד לתת העץ הימני.
- נחזור מתת העץ הימני לצומת הנוכחית ומשם לצומת האב.

בהינתן m פעולות אנו מעוניינים להגיע לסיבוכיות זמן של $O(m)$ ובכך נגיע לסיבוכיות משוערכת של $O(1)$.

מספר הצמתים בהם אנו עוברים חסום ע"י $2^{\lceil\log(m)\rceil+1}$ צמתים. לכן ניתן לחסום את המספר הכולל של מעברי הצמתים ע"י $3\cdot 2^{\lceil\log(m)\rceil+1}$ - נשים לב שמספר זה חוסם את מספר הפעולות שאנו מבצעים בסיור $inOrder$ ולכן ניתן להסיק כי סיבוכיות הזמן המשוערכת עבור m פעולות היא $O\left(3\cdot 2^{\lceil\log(m)\rceil+1}\right) = O(m)$.

שאלה 2

סעיף א'

ראשית, נמצא את המינימום והמקסימום ב- T_1, T_2 ואת גובהם (נסמנם h_1, h_2 בהתאמה).

נניח בה"כ $h_2 > h_1$. ניגשים לשורש של T_2 ונרד לבן השמאלי $h_2 - h_1$ פעמים ונסמן ב-v_0 את הצומת שאלייה הגענו.

$(*)$ במידה ול-v_0 יש אח אחד בלבד ניתן להסיק כי לאבא של v_0 (נסמנו ב-v_1) יש 2 בנים ובמקרה זה נחבר את שורש העץ של T_1 כבן שלישי שמאלי של v_1 ונוסיף את ה-key השמאלי ביותר של v_1להיות האיבר המינימלי של T_2.

אחרת, ל-v_0 יש 2 אחים. במקרה זה ניקח את v_0 ואת השורש של T_1 ונחברם לאבא משותף k_1 כאשר המפתח שיהיה בצומת זאת יהיה האיבר המינימלי של T_2. בנוסף, נסיר את ה-key השמאלי ביותר של v_1. כעת, נבצע איטרציה נוספת עבור גובה אחד מעל - כלומר נסמן את האב של v_1 ב-v_2 אם v_2 בעל 2 ילדים אזי פשוט נוכל לחבר את k_1 להיות הילד השלישי והאחרון של v_2 ונוסיף את ה-key השמאלי ביותר של v_2להיות האיבר המינימלי של T_2 . אחרת, v_2 הוא בעל 3 ילדים ואז ננתק את v_1 מאביו ונחבר את v_1 ו-k_1 לאב משותף שנסמנו k_2 כאשר המפתח שיהיה בצומת זאת יהיה האיבר המינימלי של T_2. בנוסף, נסיר את ה-key השמאלי ביותר של v_2. וכך נחזור באינדוקציה עד שנגיע לשורש העץ T_2 שהוא למעשה v_{h2-h1}. אם יש לו 2 ילדים אז רק נחבר את $k_{h_2-h_1-1}$ אליו כבן שלישי ונוסיף את ה-key השמאלי ביותר של $v_{h_2-h_1}$להיות האיבר המינימלי של T_2. במידה ויש לו 3 ילדים אז נחבר את $v_{h2-h1-1}$ואת $k_{h2-h1-1}$ לאבא משותף k_{h2-h1} כאשר המפתח שיהיה בצומת זאת יהיה האיבר המינימלי של T_2. בנוסף, נסיר את ה-key השמאלי ביותר של $v_{h_2-h_1}$. לאחר מכן, אז נחבר את k_{h2-h1} וv_{h2-h1} לאב משותף שיהיה השורש של העץ המאוחד כולו כאשר ה-key שלו יהיה המינימום של תת העץ T_2.

סהכ הסיבוכיות:

- מציאת איבר מינימלי ומקסימלי עלינו לרדת לרמה הנמוכה ביותר של 2 העצים ולכן הסיבוכיות עבור פעולה זו היא $\max\{h_1, h_2\} = \max\{\log(n_1), \log(n_2)\}$.
- עבור הירידה לצומת המתאימה נרד לכל היותר $h_2 - h_1$ פעמים ואנו יודעים כי $h_2 - h_1 < \max\{h_1, h_2\}$.
- בדיקות נוספות כגון: מספר הילדים בצומת, הלוואת ילד מאח, הוספת אב משותף ל2 ילדים - פעולות אלו מתבצעות ב-$O(1)$. פעולות אלו מתבצעות לכל היותר $h_2 - h_1$ פעמים ולכן הסיבוכיות עבור פעולה זו היא $O(h_2 - h_1 + 1)$.

סה"כ קיבלנו כי הסיבוכיות היא $\max\{\log(n_1), \log(n_2)\}$.

סעיף ב'

נבצע פעולות זהות לסעיף א' רק שכעת לא נצטרך למצוא את האיבר המינימלי והמקסימלי בכל עץ - דהיינו לא נצטרך לרדת לרמה התחתונה ביותר בכל עץ אלא רק לרדת משורש העץ T_2 (אם נניח בה"כ כי $h_2 > h_1$) $h_2 - h_1$ פעמים שמאלה כפי שהוסבר בסעיף הקודם.

סהכ הסיבוכיות:

- עבור הירידה לצומת המתאימה נרד לכל היותר $h_2 - h_1$ פעמים.
- בדיקות נוספות כגון: מספר הילדים בצומת, הלוואת ילד מאח, הוספת אב משותף ל2 ילדים - פעולות אלו מתבצעות ב-$O(1)$. פעולות אלו מתבצעות לכל היותר $h_2 - h_1$ פעמים ולכן הסיבוכיות עבור פעולה זו היא $O(h_2 - h_1 + 1)$.

סה"כ קיבלנו כי הסיבוכיות היא $O(h_2 - h_1 + 1)$. ולכן במקרה הכללי נקבל סיבוכיות זמן $O(|h_2 - h_1| + 1)$, כנדרש.

סעיף ג'

ראשית, נאחד את T_1, T_2 באמצעות סעיף ב' לעץ אחד שנסמנו S_1 (אנו יודעים את המינימום והמקסימום של 2 העצים ואת גובהם ולכן ניתן להשתמש בסעיף ב'). כעת, נשים לב שאנו יודעים מה המינימום של העץ S_1 ע"י השוואה של ערכי מינימום ומקסימום של 2 העצים. נשים לב שהגובה של S_1 הוא לכל היותר $h_2 + 1$. נשים לב שבאיטרציה הבאה אנו מעוניינים לאחד את S_1 וT_3 - במקרה זה סיבוכיות הזמן תיהיה $|h_3 - h(S_1)| + 1$ - ומכיוון שאנו מעוניינים לקבל ביטוי גדול ככל הניתן, נרצה להחסיר כמה שפחות, ולכן נרצה שגובהו של S_1 יהיה הגובה הקטן ביותר שניתן - כלומר h_2.

מכיוון שאין 3 עצים בעלי אותו גובה, לכן ניתן להתייחס למקרה הגרוע כאשר גובה העץ המאוחד הוא $h(S_i) = h_{i+1}$.

כעת נאחד את S_1 עם T_3 באמצעות סעיף ב' לעץ אחד שנסמנו S_2 (אנו יודעים את המינימום והמקסימום של 2 העצים ואת גובהם ולכן ניתן להשתמש בסעיף ב'). כעת, נקבל שסיבוכיות הזמן של איחוד העצים $h_3 - h_2 + 1$ ונשים לב שעל מנת לקבל את סיבוכיות הזמן הגרועה ביותר באיטרציה הבאה נסתכל על העץ כאשר גובהו הוא h_3 (תוך שימוש בהפעלת טיעון זהה לעיל).

נמשיך לאחד את העצים באינדוקציה, דהיינו $T_i \cup S_{i-2}$ עבור $3 \leq i \leq k$. ונקבל:

$$\sum_{i=2}^{k} h_i - h_{i-1} + 1 = h_k - h_1 + k$$

סעיף ד'

ראשית, נגדיר 2 $counter$: $i = j = 1$.

נסמן ב-v את שורש העץ. כעת, נבצע את הפעולות הבאות עד הגיענו ל-x או $NULL$ במידה ואיננו קיים.

נפצל למקרים:

- במידה ול-v יש 2 בנים אז נבצע את הפעולות הבאות:
  - אם x קטן מהערך של v, אז נסמן ב-t_i את תת העץ הימני של v ונשמור אותו בצד. בנוסף, נשמור את ערכו של v להיות שיהווה חסם תחתון לערכי t_1. כעת, נסמן $v = v \to leftSon$, דהיינו ניכנס לתת העץ השמאלי ונגדיל את i באחד.
  - אם x גדול שווה מהערך של v, אז נסמן ב-s_j את תת העץ השמאלי של v ונשמור אותו בצד. בנוסף, נשמור את ערכו של v להיות שיהווה חסם עליון לערכי s_1. כעת, נסמן $v = v \to rightSon$, דהיינו ניכנס לתת העץ הימני ונגדיל את j באחד.
  - אם x שווה לערכו של v נסמן צומת זו ב-s_0.
- במידה ול-v יש 3 בנים, נסמן ב-$key1, key2$ את המפתחות המסומנים על הצומת הנוכחית ונניח כי $key1 < key2$.
  - אם $x < key1$ נסמן ב-t_i את עץ זה אחרי השמטת הבן השמאלי שלו ונשמור אותו בצד. נסיר מ-v את $key1$ ונשמור אותו בצד אשר יהווה חסם תחתון לערכי t_i לצורך פעולות בהמשך התוכנית. כעת, נסמן $v = v \to leftSon$, דהיינו ניכנס לתת העץ השמאלי ונגדיל את i באחד.
  - אם $x > key2$ נסמן ב-s_j את עץ זה אחרי השמטת הבן הימני שלו ונשמור אותו בצד. נסיר מ-v את $key2$ ונשמור אותו בצד אשר יהווה חסם עליון לערכי s_j לצורך פעולות בהמשך התוכנית. כעת, נסמן $v = v \to rightSon$, דהיינו ניכנס לתת העץ הימני ונגדיל את j באחד.

– אם $key1 \ < \ x \ < \ key2$ נסמן ב-t_i את תת העץ הימני של v ונשמור אותו בצד. נשמור את $key2$ בצד אשר יהווה חסם תחתון לערכי t_i לצורך פעולות בהמשך התוכנית, ונגדיל את i באחד. בנוסף, נסמן ב-s_jאת תת העץ השמאלי של v ונשמור אותו בצד. נשמור את $key1$ בצד אשר יהווה חסם עליון לערכי s_j לצורך פעולות בהמשך התוכנית ונגדיל את j באחד. כעת, נסמן $v \ = \ v \ \rightarrow$ $middleSon$, דהיינו ניכנס לתת העץ האמצעי.

אם $v = null$ יוצאים מהתכנית.

אחרת מצאנו את הערך המבוקש x ובזמן הלולאה קיבלנו סדרת עצים $s_0, s_1, \ldots, s_n$ שבהם כל המפתחות קטנים שווים ל-x ומתקיים $h\left(s_0\right) < h\left(s_n\right) < h\left(s_{n-1}\right) < \ldots . < h\left(s_1\right)$. נוכל להפעיל את סעיף ג' (אנו יודעים את המינימום והמקסימום של העצים) ולקבל עץ אחד מאוחד שבו כל המפתחות קטנים שווים ל-x.

סיבוכיות הפעלת סעיף ג' היא $h(s_1) - h(s_0) + n + 1$. במקרה שלנו הגובה של s_1 הוא לכל היותר$log(n)$ והגובה של s_0 הוא 1. בנוסף, מספר העצים שמתקבלים בסדרת העצים הוא לכל היותר $log(n)$ מכיוון שכל עץ התקבל כאשר פנינו מצומת לאחד מהבנים ולכן מקסימום יש $log(n)$ פניות כאלה ולכן מקסימום $log(n)$ עצים כאלה.

סה"כ סיבוכיות: $log(n) - 1 + log(n) + 1$ שזה $O(log(n)$.

נבצע אותו דבר על סדרת העצים $t_1, \ldots.., t_m$ בהם כל המפתחות גדולים ממש מ-x ומתקיים $h\left(t_m\right) < \ldots < h\left(t_1\right)$ ונוכל להפעיל את סעיף ג' על סדרת עצים זו ולקבל עץ אחד מאוחד שבו כל המפתחות גדולים ממש מ-x.

סיבוכיות הפעלת סעיף ג' היא $h(t_1) - h(t_m) + m$.

במקרה שלנו הגובה של t_1 הוא לכל היותר $log(n)$ והגובה של t_1 הוא 2 לכל הפחות.

בנוסף, מספר העצים שמתקבלים בסדרת העצים הוא לכל היותר $log(n)$, מכיוון שכל עץ התקבל כאשר פנינו מצומת לאחד מהבנים, ולכן יש מקסימום $log(n)$ פניות כאלה ולכן מקסימום $log(n)$ עצים כאלה , סה"כ סיבוכיות: $log(n) - 2 + log(n)$ שזה $O(log(n)$.

שאלה 3

סעיף א'

נשמור עץ דרגות על בסיס עץ AVL כאשר בכל צומת v נשמור את ה-key (יום) וה-$value$ (מספר החולים ביום) ובנוסף נשמור 2 ערכים נוספים:

1. $max\_right\_tree\left(v\right)$ - זהו מספר החולים המקסימלי מבין צמתי העץ הימני של v.
2. $max\_left\_tree\left(v\right)$ - זהו מספר החולים המקסימלי מבין צמתי העץ השמאלי של v.

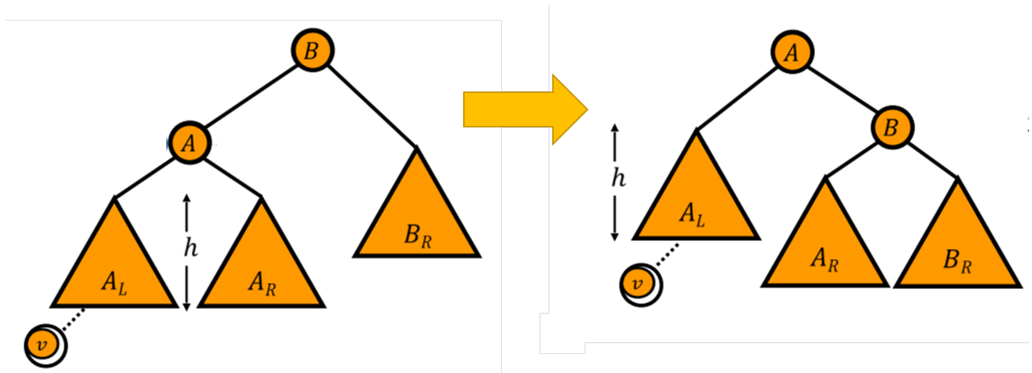
$Init\left(\right)$ - נאתחל עץ דרגות ריק.

$SetPositiveInDay\left(d,x\right)$ - נבצע הכנסת איבר לעץ עם מפתח d וערך x. כעת, נעבור על מסלול החיפוש של x בעץ מלמטה למעלה ולכל צומת נבצע גלגול בהתאם ומספר סופי של עדכונים ערכי $max-right-tree$ ו-$max-left-tree$ בהתאם. נשים לב שכיוון שגלגול הוא החלפה של מספר סופי של מצביעים גם עדכון ערכי $max-right-tree$ ו-$max-left-tree$ היא נעשית במספר סופי של פעולות.

$WorseBefore\left(d\right)$ - ראשית, נבצע חיפוש של הצומת בעל מפתח d בעץ. נסמנו ב-v. נחלק ל-2 מקרים:

- נסמן ב-$h\left(v\right)$ את מספר החולים השמורה בצומת v. כעת, נבצע את סדרת הפעולות עד שנגיע לעלה בעץ ושם נעצור ונחזיר את הדרוש.
  - במידה ו-$h\left(v\right) < max\_left\_tree\left(v\right)$ ניכנס לתת העץ השמאלי (כלומר $v \rightarrow left\_son$) על-מנת לחפש את מספרו של היום האחרון לפני d, מבין הימים אשר הוזנו למבנה, שבו מספר החולים היה גדול ממספר החולים ביום d. כעת, במידה והשדה $max\_right\_tree\left(v \rightarrow left\_son\right)$ גדול מ-$h\left(v\right)$ ניכנס לתת העץ הימני. אחרת, נבדוק את ערכו של הצומת הנוכחית בה אנו נמצאים, במידה ו-$h\left(v_0\right) < h\left(v \rightarrow left\_son\right)$ נחזיר את היום המתאים לצומת בה אנו נמצאים. אחרת, ניכנס לתת העץ השמאלי ונמשיך באינדוקציה עד הגעה לעלה.
    * למעשה המטרה באלגוריתם זה היא לקבל את היום האחרון לפני d (אנו אכן מחפשים יום לפני היום d מכיוון שאנו סורקים בתת העץ השמאלי) ובפרט אנו מחפשים יום שבו מספר החולים היה גדול ממספר החולים ביום d.
- נעלה ל-$parent\left(v\right)$. אם $parent\left(v\right)=NULL$ נחזיר -1.
- נסתכל על $parent\left(v\right)$ ונשאל מה ערך ה-day שלו, אם הוא גדול מערך ה-day של v_0 נמשיך לאיטרציה הבאה כאשר נסמן $parent\left(v_0\right)=parent\left(parent\left(v\right)\right)$, אחרת ערך ה-$day$ שלו קטן מערך day של v במקרה זה נבדוק את הערך $h\left(parent\left(v\right)\right)$, אם הוא היה גדול מערך $h\left(v\right)$, אז מצאנו את מה שחיפשנו ונחזיר את ערך ה-day של $parent\left(v\right)$, אחרת הערך $h\left(parent\left(v\right)\right)$ קטן שווה לערך $h\left(v\right)$.
במקרה הזה נבדוק את ערך ה-$max\_left\_tree\left(parent\left(v\right)\right)$ - אם הוא קטן שווה לערך $h\left(v\right)$ אז נסמן $parent\left(v\right)=parent\left(parent\left(v\right)\right)$ ונבצע איטרציה נוספת. אחרת, ערך ה-$max\_left\_tree\left(parent\left(v\right)\right)$ גדול ממש מ-$h\left(v\right)$ ואז נפנה לבן השמאלי של $parent\left(v\right)$. נסמנו $k\left(v\right)$ ונבצע את הבאים בלולאה:

1. אם $max\_right\_tree\left(k\left(v\right)\right)$ גדול ממש מ-$h\left(v\right)$ נסמן $k\left(v\right) = k\left(v\right) \rightarrow right\_son$, כלומר הבן הימני של $k\left(v\right)$ הוא ה-$k\left(v\right)$ החדש, ונבצע איטרציה נוספת.
2. אם $h\left(k\left(v\right)\right)$ גדול ממש מ-$h\left(v\right)$ אז מצאנו את הדרוש ונחזיר את ה-day של $k\left(v\right)$.
3. אם ערך $max\_left\_tree\left(k\left(v\right)\right)$ גדול ממש מ-$h\left(v\right)$, נסמן $k\left(v\right)=k\left(v\right)\rightarrow left\_son$, דהיינו הבן השמאלי של $k\left(v\right)$ הוא ה-$k\left(v\right)$ החדש, ונבצע איטרציה נוספת.



בגלגול LL למשל נשנה את הערכים הבאים:

- $max\_left\_tree\left(root\left(A_R\right)\right) = max\left\{root\left(A_R\right) \rightarrow value, max\_left\_tree\left(root\left(A_R\right)\right), max\_right\_tree\left(root\left(A_R\right)\right)\right\}$
- $max\_right\_tree\left(A\right) = max\left\{B \rightarrow value, max\_left\_tree\left(B\right), max\_right\_tree\left(B\right)\right\}$

עבור הגלגולים האחרים נבצע את השינוי ועדכון ערכי הצמתים באופן דומה.

סעיף ב'

מבנה הנתונים שנציע יהיה דומה למבנה הנתונים הנ:ל בתוספת שמירת המידע הנחוץ כדי להשתמש באלגוריתם $Select$ שראינו בתרגול.

$SetPositiveInDay\left(d, x\right)$ - נעדכן את המידע הנוסף בצמתים לצורך תקינות האלגוריתם $Select$ כפי שלמדנו בתרגול. שאר הפוקנציה כבסעיף הקודם.

$IsWorseSince\left(d_1, d_2\right)$ - ראשית, נקרא ל-$WorseBefore\left(d_2\right)$. נסמן את ערך ההחזרה של פונקציה זו ב-X.

בנוסף, נפעיל את אלגוריתם $Select$ על מנת למצוא את האינדקסים של האיברים d_1, d_2 בעץ. נסמן $Select\left(d_1\right) = i_1$ ו-$Select\left(d_2\right) = i_2$.

כעת, נחלק למקרים:

- במידה ו- $X \geq d_1$ במקרה זה נחזור $false$ - כי במקרה זה קיים יום בין d_1 ל-d_2 בו היו יותר חולים מב-d_2.
- במידה ו$d_2 - d_1 = i_2 - i_1$ וגם $X < d_1$ במקרה זה נחזור $true$.
- במידה ו$d_2 - d_1 \neq i_2 - i_1$ וגם $X < d_1$, נחזיר $unknown$ - כי במקרה זה לא ניתן לדעת התשובה בוודאות.

שאלה 4

סעיף א'

נציע מימוש של רשימת דילוגים דטרמיניסטית כאשר ברשימה התחתונה נשמור מצביע הרשימה כאשר בכל קריאה ל-$NextStop\left(\right)$ נקדם את המצביע ברשימה התחתונה ונחזיר את האיבר אליו הגענו.

$Init\left(\right)$ - נתאחל רשימת דילוגים ריקה ונתאחל את המצביע לרשימה בעלת כל האיברים להצביע על הרשימה הריקה הנוכחית שאותחלה זה עתה.

$AddStop\left(k\right)$ - הוספת תחנת עצירה בקומה k עבור המעלית - למעשה נוסיף את האיבר שערכו k לרשימת הדילוגים. נשים לב כי בכל הוספת איבר ייתכן והוספנו "קומה" לרשימת הדילוגים אך המצביע שמהווה איטרטור של איברי הרשימה איננו משתנה.

$NextStop\left(\right)$ - ניגש למצביע הנוכחי המהווה איטרטור של איברי הרשימה ב-$O\left(1\right)$, ניגש לאיבר הבא. כעת, במידה ואיבר זה שווה ל-∞ נישאר בקומה הנוכחית. אחרת, נדפיס אותו ונקדם את האיטרטור.

סעיף ב'

מבנה הנתונים באמצעותו נפתור את השאלה הוא מערך דינמי בשם arr. בנוסף, נשמור משתנה $maxSize$ אשר שומר את גודל המערך ומשתנה $currSize$ ששומר את גודלו הנוכחי של המערך. בנוסף, נשמור מצביע המהווה איטרטור בשם $iterator$ באיזו קומה המעלית נמצאת לשימוש $nextStop$.

$Init\left(\right)$ - נבצע אתחול מערך בגודל K (קבוע כלשהו כפי שראינו בתרגול ובנוסף נאתחל את $maxSize$ להיות K ו-$currSize$ להיות 1.

$AddStop\left(k\right)$ - ניגש למקום ה-k במערך ונאתחל אותו להיות $true$, נוסיף 1 ל-$currSize$. אם המערך חצי מלא , דהיינו $currSize == \frac{1}{2}maxSize$ אז נקצה מערך גדול פי 2 (בגודל $2 \cdot maxSize$) , נעדכן את $maxSize$ בהתאם , ונעתיק את האיברים.

$nextStop\left(\right)$ - ניגש לאיטרטור ונעבור על איברי המערך (עבור אינדקסים הגדולים מהאינדקס של האיטרטור) עד שנגיע לתא במערך שערכו $true$. לבסוף, נדפיס את מספר הקומה הנוכחי.

סיבוכיות משוערכת של $AddStop$ ו-$nextStop$:

נסתכל על סדרה כלשהי באורך m המורכבת מהפעולות הבאות :

כעת, נסמן ב-m_1 את מספר הפעולות של $AddStop\left(k\right)$ וב-m_2 את מספר הפעולות של $NextStop\left(\right)$.

נשים לב ראשית כי אם היו לנו m_1 פעולות $AddStop$ אז הקומה המקסימלית שאליה ניתן להגיע הינה $2 \cdot m_1$ מהנתון. לכן, $nextStop$ תעבור על איברי מערך בגודל $2 \cdot m_1$ לכל היותר עד שתגיע לאיבר הראשון שערכו הוא $true$.

נסמן ב-k_i את מספר הבדיקות שאנו מבצעים בקריאה ה-i לפונקציה $nextStop$. בנוסף, נשים לב כי $\sum_{i=1}^{m_2} k_i \leq 2 \cdot m_1$. כלומר, $\sum_{i=1}^{m_2} k_i$ מהווה את סך הבדיקות שאנו מבצעים עבור m_2 $nextStop$ פעמים - כל בדיקה האיטרטור עובר על איברי המערך החל מאיפה שנמצא עד הגיעו לתא הראשון שהוא $true$ ושם עוצר. נשים לב שלאחר שביצענו $k_1 + k_2 + \cdots + k_{m_2}$ בדיקות עלינו לקומה לכל היותר $2 \cdot m_1$ - מהנתון.

כמו כן, הסיבוכיות של $AddStop$ היא $O\left(1\right)$ כאשר במידה ונגדיל את המערך פי 2, ראינו בתרגול כי הגדלת המערך איננה פוגעת בסיבוכיות המשוערכת.

לכן, מתקיים:

$$Total\ time\ = m_1 \cdot O(1) + O\left(\sum_{i=1}^{m_2} k_i\right) \leq m_1 \cdot O(1) + O\left(\sum_{i=1}^{m_2} (k_i+1)\right)$$
$$= m_1 \cdot O(1) + O(m_2 + 2 \cdot m_1) = O(m)$$

לכן, ניתן להסיק כי הסיבוכיות המשעורכת עבור $AddStop(k)$ ו-$nextStop()$ היא $O(1)$, כנדרש.

שאלה 5

מבנה נתונים – נשתמש בשני מצביעים לשורשים של עצי $Trie$ כפי שלמדנו בהרצאה ומשתנה m שיהווה את מספר צמדי המילים כבדרישות התרגיל.

נשים לב כי לכל צומת פנימי יש כגודל $|\sum| + 1$. כל צומת מכיל את הרכיבים הבאים: ערך key הוא תו מסוים,

ערך $counter$ אשר מייצג כמה מילים יש שמתחילים באות זאת.

$Init(m)$ – נאתחל את שורש העץ T_1 ל-$null$, ונאתחל את שורש העץ T_2 ל-$null$. ונאתחל את השדה m להיות m.

נשים לב כי סיבוכיות הפעולות המוזכרות לעיל הינם $O(1)$.

$Insert(w1, w2)$– נבצע הכנסה רגילה לעץ כפי שהכרנו בהרצאה, כלומר נכניס את המילה הראשונה לעץ T_1 לפי סדר האותיות, נכניס את המילה השנייה לעץ T_2 לפי סדר האותיות. נתאר את תהליך הכנסת המילה הראשונה לעץ T_1 כאשר ההכנסה של המילה השנייה מתבצעת באופן זהה. נניח שהמילה w_1 בת k אותיות נבצע את הבאים בלולאה $i = 1\ to\ k$, (v_0 שורש העץ).

בשלב ה-i אנו נמצאים בצומת v_{i-1} נחפש להתקדם לצומת הבאה שערך ה-key שלה שווה לאות ה-i של המילה w_1. אם צומת כזאת לא קיימת, אז נוסיף צומת כזו עם ערך key השווה לאות ה-i של המילה w_1, ונאתחל את השדה $counter$ ל-1 ונסמן צומת זו ב-v_i.

אם מצאנו צומת שעונה על התנאי הנ"ל אז נתקדם אליה, נוסיף לשדה $counter$ שלה 1 ונסמן צומת זו ב-v_i ונתקדם לאיטרציה הבאה של הלולאה.

סיבוכיות הכנסת האיבר הראשון הינה $O(|w_1|)$ מכיוון שבכל איטרציה של הלולאה אנו מתקדמים צומת, בכל איטרציה $O(1)$ פעולות השוואה ועדכון שדות, ויש לכל היותר $|w1|$ איטרציות. סיבוכיות הכנסת האיבר השני הינה $O(|w_2|)$ מכיוון שבכל איטרציה של הלולאה אנו מתקדמים צומת, בכל איטרציה $O(1)$ פעולות השוואה ועדכון שדות, ויש לכל היותר $|w_2|$ איטרציות. סה"כ סיבוכיות ההכנסה של 2 המילים יחדיו הינה $O(|w1| + |w2|)$.

$Magic(w1, w2)$ – נאתחל משתנה $totalSmallerWords$ ו-$totalBiggerWords$ להיות מאותחלים ל-0 בהתחלה. נחפש את המילה w_1 בעץ t_1, נניח שיש ב-w_1 -k- אותיות. בשלב הראשון נתקדם מ-v_0 (שורש העץ) לצומת שערך ה-key שלה שווה לאות הראשונה של המילה w_1 נסמן צומת זו ב-v_1. נעבור על כל האחים של צומת זאת כאשר בכל ביקור בצומת אח נבדוק אם ערך key של אח זה הוא קטן ממש מערך האות הראשונה במילה $w1$.

במידה ומתקיים התנאי אז ניגש לערך $counter$ של צומת זו ונוסיף את ערכה ל-$totalSmallerWords$ (נשים לב שעברנו על מספר סופי של צמתים כלומר מקסימום מספר צמתים כגודל $|\sum| + 1$).

באופן כללי בלולאה עבור $i = 1\ \ to\ \ k$. (כאשר נסמן v_0 להיות שורש העץ) נבצע את הפעולות הבאות:

- בשלב ה-i נתקדם מ-v_{i-1} לצומת הבאה שערך ה-key שלה שווה לאות ה-i של המילה w_1 נסמן צומת זו ב-v_i. נעבור על כל האחים של צומת v_i כאשר בכל ביקור בצומת אח נבדוק אם ערך key של אח זה הוא קטן ממש מערך האות ה-i במילה w_1, במידה ומתקיים התנאי אז ניגש לערך $counter$ של צומת זו ונוסיף את ערכה ל-$totalSmallerWords$ (נשים לב שעברנו על מספר סופי של צמתים כלומר מקסימום מספר צמתים כגודל השפה ועוד 1).

כאשר סיימנו את הלולאה, המשתנה $totalSmallerWords$ מכיל את מספר המילים שקטנות לקסיקוגרפית מהמילה w_1. נבצע כעת אותו דבר בעץ T_2 עם המילה w_2 ונניח שיש בה s אותיות. באופן כללי, בלולאה עבור $j = 1\ to\ s$ נבצע את הפעולות הבאות (כאשר נסמן v_0 להיות שורש העץ):

- בשלב ה-j נתקדם מ-v_{j-1} לצומת הבאה שערך ה-key שלה שווה לאות ה-j של המילה w_2 נסמן צומת זו ב-v_j. נעבור על כל האחים של צומת v_j כאשר בכל ביקור בצומת אח נבדוק אם ערך key של אח זה הוא גדול ממש מערך האות ה-j במילה w_2. במידה ומתקיים התנאי, ניגש לערך $counter$ של צומת זו ונוסיף את ערכה ל-$totalBiggerWords$ (נשים לב שעברנו על מספר סופי של צמתים כלומר מקסימום מספר צמתים כגודל $|\sum|+1$). כאשר סיימנו את הלולאה $totalBiggerWords$ מכיל את מספר המילים שגדולות לקסיקוגרפית מהמילה w_2.

  – בסוף , כל מה שנותר לבדוק הוא האם $totalSmallerWords\ ,totalBiggerWords > m$ אם כן אז נחזיר $true$ אחרת נחזיר $false$.

סיבוכיות זמן

- סיבוכיות הלולאה הראשונה הינה $O(|w_1|)$ מכיוון שבכל איטרציה של הלולאה אנו מתקדמים צומת , בכל איטרציה $O(1)$ פעולות השוואה/עדכון שדות , מכיוון שאנו עוברים על כל האחים של הצומת באיטרציה בודדת אך יש מספר סופי של אחים לכל צומת (כגודל כגודל $1 + |\sum|$). בנוסף יש לכל היותר $|w_1|$ איטרציות כאורך המילה w_1 לכן סה״כ $O(|w_1|)$.

- סיבוכיות הלולאה השנייה הינה $O(|w_2|)$ מכיוון שבכל איטרציה של הלולאה אנו מתקדמים צומת , בכל איטרציה $O(1)$ פעולות השוואה/עדכון שדות , מכיוון שאנו עוברים על כל האחים של הצומת באיטרציה בודדת אך יש מספר סופי של אחים לכל צומת (כגודל כגודל $1 + |\sum|$). בנוסף יש לכל היותר $|w_2|$ איטרציות כאורך המילה w_2 לכן סה״כ $O(|w_2|)$.

סה״כ סיבוכיות הזמן של הפונקציה היא $O(|w_1| + |w_2|)$.