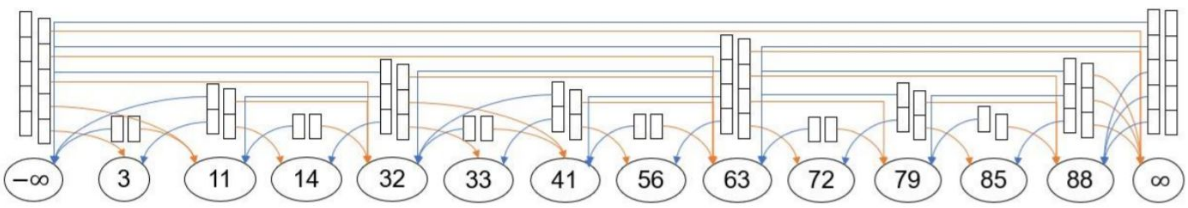
<u>מבני נתונים - עבודה 2, חלק ג'</u>

עמית יעקב 205916851
שירה שגב 208825349

<u>הקדמה</u>: מספר המפתחות הנתון הינו n. מלבד תכונות הDynamic Set ADT אשר מקיים המבנה, נניח בחלק זה כי מקיים תכונה נוספת: גודל המערך של החוליה הi $1 \le i \le n$ הינו x+1 כאשר x הינו המספר הטבעי הגדול ביותר שמקיים: $imod2^x = 0$, $(0 \le x \le logn)$, ו-n הוא מספר המפתחות ברשימת מערך הקומות.

<u>חיפוש</u>

1. <u>תיאור האלגוריתם lookup</u>: אלגוריתם זה מחפש מפתח ברשימה שמימשנו FloorsArrayList. עיקר החיפוש מתבצע בפונקציית עזר בשם find המקבלת כקלט את המפתח שאנו מחפשים ומחזירה floorsArrayLink. הלינק שיוחזר ע"י הפונקציה יהיה האיבר שאנו מחפשים, במידה וקיים, ואם אינו קיים, הפונקציה תחזיר את האיבר הקודם לו מבחינת גודל המפתח. ב lookUp נבדוק האם הלינק שהוחזר מחזיק את המפתח שחיפשנו, אם לא, נחזיר null.
אופן החיפוש בפונקציית העזר find: נרצה להתחיל את החיפוש במערך המצביעים של החוליה הראשונה ($-\infty$). משיקולי יעילות, מאחר וייתכן פער משמעותי בין N (מס' החוליות במבנה, הגובה ש$-\infty$ מאותחל להיות) לבין גובה המערך המקסימלי ברשימה שאינו $\pm\infty$, נשמור את ערך המערך המקסימלי, ונתחיל את החיפוש מגובה זה. בעת החיפוש נבצע השוואה בגובה נתון (i בקוד שכתבנו) בין האיבר עליו אנו "מטיילות" לבין האיבר אליו נשלחת ההצבעה מתא זה במערך, אם ערכו קטן מזה שאנו מחפשות, נעבור "לטייל" עליו, ואם הוא גדול יותר, נרד קומה במערך ונמשיך את ההשוואה שם.
נעצור באחד מהמקרים הבאים: אם מצאנו את האיבר, נחזירו. אחרת, נעצור כאשר נגיע לקומה התחתונה במערך.
2. <u>ניתוח זמן הריצה עבור lookup</u>: בסעיף זה נתייחס אל המבנה כמקיים את התכונה הנתונה אודות גודל המערך. נתון כי גודל המערך של החוליה הi $1 \le i \le n$ הינו x+1 כאשר x הינו <u>המספר הטבעי הגדול ביותר</u> שמקיים: $imod2^x = 0$, $(0 \le x \le logn)$. בשל כך, נבחין כי הגובה המקסימלי של מערך הקומות במבנה יהיה logn+1. המקרה המחמיר יתקבל בחיפוש כושל ולכן פעולות השוואה יתבצעו באופן לינארי לאורך המערך המקסימלי ממנו אנו מתחילות את החיפוש כך שבסה"כ, זמן הריצה של האלגוריתם במקרה המחמיר יהיה $\Theta(logn+1) = \Theta(logn)$.



<u>הכנסה</u>

בסעיף זה ננתח את זמן הריצה ההדוק ביותר עבור האלגוריתם Insert אשר מכניס מפתח למבנה, במקרה המחמיר ביותר. ניקח בחשבון עמידה בכל תנאי המבנה, נוסף על התכונה הנתונה בחלק זה של העבודה. כלומר, נתונים n מפתחות והתבקשנו לבצע הכנסה יחידה למבנה.
אלגוריתם ההכנסה שכתבנו, משתמש גם כן בשיטת העזר find שכתבנו עבור השיטה lookUp. הפעם, השיטה find תחזיר את האיבר הקודם לאיבר שברצוננו להכניס. בשל כך, כפי שהראנו בסעיף קודם, זמן הריצה עבור החיפוש יהיה $\Theta(logn)$. אולם, עבור הכנסת איבר נדרש עדכון של המצביעים במבנה. בהתאם לתנאי בחלק זה, גובה המערך של האיבר החדש יהיה logn במקרה הגרוע. באלגוריתם שכתבנו מבוצעות כ4 פעולות שינוי מצביעים עבור כל קומה במערך הקומות של האיבר החדש שאנו מכניסות. מכאן שזמן הריצה הינו $\Theta(4logn) \approx \Theta(logn)$.

<u>הוצאה</u>

1. <u>תיאור האלגוריתם remove</u>: אלגוריתם הוצאה/ הסרה בFloorsArrayList.
   - נקבל כקלט את החוליה שברצוננו להסיר (כידוע, החוליה מחזיקה את מערך המצביעים שלה במבנה).
   - במקרה שחוליה זו הינה בעלת המערך שגובהו המקסימלי במבנה, נעדכן את שדה המערך המקסימלי.
   - נבצע מעבר על מערך המצביעים ונרצה לעדכן את האיברים ש"מחוברים/קשורים" לחוליה שנסיר:
     - נשמור במשתנים את החוליה הקודמת, ואת החוליה הבאה.
     - עבור האיבר הקודם- במיקום שהצביע על החוליה שנסיר, יצביע כעת על החוליה הבאה ששמרנו.
     - עבור האיבר הקודם- במיקום שהצביע על החוליה שנסיר, יצביע כעת על החוליה הקודמת ששמרנו.
   - לבסוף נעדכן את שדה הsize.
2. <u>ניתוח זמן הריצה עבור remove</u>: בסעיף זה נחשב מהו זמן הריצה ההדוק ביותר עבור השיטה remove במקרה המחמיר ביותר. ניקח בחשבון עמידה בכל תנאי המבנה, נוסף על התכונה הנתונה בחלק זה של העבודה. כלומר, נתונים n מפתחות והתבקשנו לבצע הוצאה יחידה.
   נגדיר עבור סעיף זה את המקרה המחמיר ביותר כהסרת החוליה שגודל המערך שלה הוא המקסימלי (logn).
   כפי שתיארנו בסעיף הקודם, השיטה מקבלת את החוליה שנרצה להסיר, כלומר, את מערך המצביעים שלה. ולכן, זמן הריצה של הסרת איבר מהמבנה כולל עדכון מצביעים בלבד. עבור כל תא במערך המצביעים של החוליה שנסיר, נשמור את החוליה הקודמת ואת החוליה הבאה- שתי פעולות עבור כל קומה. סה"כ עבור כל תא במערך, יתבצעו 2 פעולות עדכון מצביעים .לכן, בשל ההנחה של חלק זה בעבודה, במקרה הגרוע יבוצעו $4arrSize\ =\ 4\log n$ פעולות שזמן הריצה שלהן הוא: $\theta(\ 4\cdot\ \log n)\ \sim\ \theta(\ \log n)$.

<u>גודל המבנה</u>

בסעיף זה נחסום מלמעלה את ה<u>מקום</u> שהמבנה דורש (ההנחה היא כי הוא מכיל n חוליות), בעודו מקיים את התכונה הנתונה בחלק זה של העבודה.מבנה זה מורכב מ-n חוליות אשר מזוהות ע"י המפתח שלהן, אך בנוסף מחזיקות מידע כמו מערך מצביעים (satellite data). נניח שכל מספר וכל מצביע נשמרים ב $O(1)$ מקום בזכרון, ולכן עבור חוליה, מערך המצביעים שלה יישמר ב $O(arrSize)$ (כפי שהוגדר בבנאי המחלקה Link). במקרה הגרוע, arrSize = logn+1 (כפי שהוכח ב1).

כעת, <u>נניח</u> כי עבור $1 \le i \le logn + 1$ (n-מס' החוליות במבנה), מס' החוליות שמערכן יהיה בגובה i הוא $[\frac{n}{2^{i-1}}]$ ($i_{max} = logn + 1$ מהחישוב $imod2^x = 0$, $(0 \le x \le logn)$ כאשר x+1 הינו גובה המערך).

לשם הוכחת ההנחה נשתמש ב<u>טענת עזר</u>: קיימת רק חוליה אחת במבנה שגובה המערך שלה הינו logn+1. חוליה זו מתקבלת ע"י x=logn.
<u>הוכחת טענת העזר</u>: נניח בשלילה כי קיימות שתי חוליות במבנה שגובהן הינו logn+1, מיקומן יהיה מתוך הקבוצה $\{1, 2, ..., n\}$. ניקח $t_1, t_2 \in \{1, 2, ..., n\}$ ($t_1, t_2 \le n$) ונראה כי $t_1 = t_2$. מההנחה בחלק ג', נסמן את גובהו של t_1 ב h_1 : $t_1 mod 2^{h_1} = 0$, באותו אופן עבור t_2 , נסמן את גובהו ב h_2 : $t_2 mod 2^{h_2} = 0$. $h_1 = h_2 = logn \Leftarrow t_1 mod 2^{logn} = t_2 mod 2^{logn} \Leftarrow t_1 modn = t_2 modn$ שני המקרים המקיימים את השיוויון הנ"ל הינם:
<u>מקרה א</u>: $t_1, t_2 < n$: $t_1 modn = t_1 = t_2 modn = t_2 \Leftarrow t_1 = t_2$ (מתכונות מודולו)- סתירה.
<u>מקרה ב</u>: $t_1, t_2 = n$: גורר $t_1 = t_2$ - סתירה.

<u>הוכחת ההנחה</u>: נוכיח באינדוקציה שלמה על i. כלומר, נראה עבור כל גובה במבנה, כי מס' החוליות במבנה שמערכן בגובה i הוא $[\frac{n}{2^{i-1}}]$.

<u>בסיס</u>:

- <u>רשימה בגודל 1</u>: מערך המצביעים יהיה באורך $1mod2^0 = 0 \Rightarrow x = 0 \Rightarrow x + 1 = 1$ לכן יש $[\frac{1}{2^0}] = 1$ חוליות בגובה 1 במערך.
- <u>רשימה בגודל 2</u>: עבור האיבר הראשון- מערך המצביעים באורך 1
  עבור האיבר השני- מערך מצביעים בגובה 2.
  עבור $i = 1 \Leftarrow \left[\frac{2}{2^0}\right] = 2$.
  עבור $i = 2 \Leftarrow \left[\frac{2}{2^1}\right] = 1$.

כלומר, עבור כל רשימה, בגובה i=1 כל n החוליות יופיעו, כלומר $[\frac{n}{2^{i-1}}] = n$. נבחין כי בשלב הבא, i=2, יימצאו כ $\frac{n}{2}$ חוליות וכך הלאה.. ככל ש-i יהיה גדול יותר, כך נצפה לפחות חוליות בגובה זה.
<u>הנחת האינדוקציה</u>: נניח נכונות הטענה עבור $j < i \le logn + 1$. כלומר עבור רשימה עם n חוליות, מס' החוליות בגובה j הינו $[\frac{n}{2^{j-1}}]$.
<u>צעד האינדוקציה</u>: נראה נכונות הטענה עבור i=logn+1. מהנחת האינדוקציה, קיימות $[\frac{n}{2^{j-1}}]$ חוליות שגובהן $\ge$x+1 במיקומים שונים ברשימה עבור x זה. בהצבת i=logn+1 נקבל $[\frac{n}{2^{logn+1-1}}] = [\frac{n}{n}] = 1$ חוליות בגובה מקסימלי זה (מטענת העזר).

כעת נובע שהמקום בזכרון שיישמר בכל קומה במבנה (עבור כל i) הוא $[\frac{n}{2^{i-1}}]$. כעת נסכום את סך הקומות במבנה (הקומות חסומות מלמעלה ע"י logn+1 כפי שהוסבר קודם לכן):

$$\sum_{i=1}^{O(\log n)+1}\left(\frac{n}{2^{i-1}}\right) = n\cdot\sum_{i=1}^{\infty}\frac{1}{2^{i-1}} = 2n\sum_{i=1}^{\infty}\frac{1}{2^{i}} = 2n\cdot\frac{0.5}{0.5} = 2n$$

לכל חוליה יש שני מערכי מצביעים אז נכפיל את הכל ב2 ונקבל 4n.
נוסיף את שני המערכים של הקצוות, כל אחד בגובה n ונקבל סה"כ 8n.
לכן, המקום בזיכרון הינו $O(8n) \approx O(n)$.