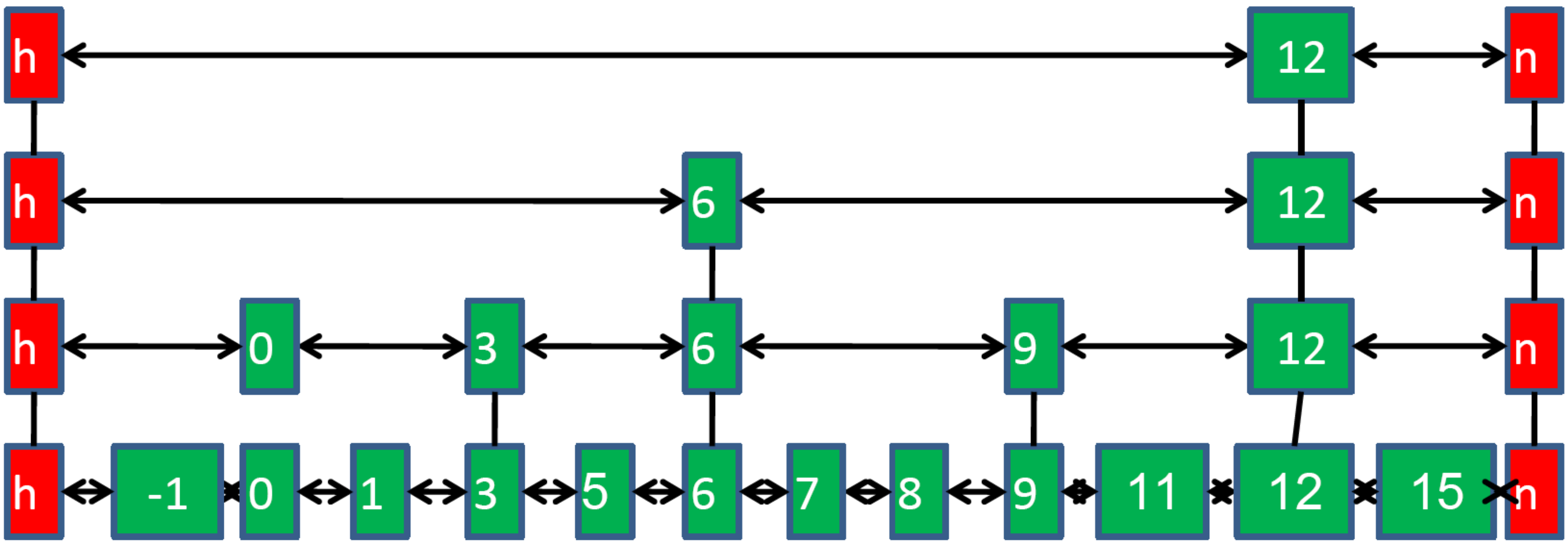
Data Structures

הרצאה 22 – רשימת דילוג

Skip List

רשימת דילוג – Skip List

המדען **William Pugh** המציא את **רשימת הדילוג** ב- **1990** מתוך רצון למצוא מבנה נתונים פשוט שיחליף את עצי החיפוש המאוזנים **היעילים**, אך **מסורבלים**, כמו עצי **AVL** ו**אדום**-**שחור**

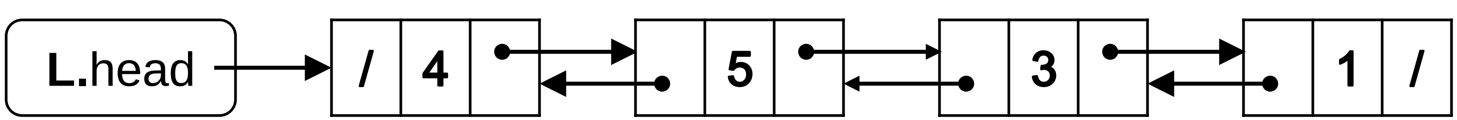


תזכורת: רשימה מקושרת

- ניתן להכניס/לשלוף איבר בזמן ריצה **O(1)**
- יתרון על מערך הוא **דינמיות** - ניתן להוסיף איברים בלי לדאוג להקצאת זיכרון מראש
- אבל ב**חיפוש**, רשימה **ממוינת** פחות יעילה ממערך **ממוין**
  - הסיבוכיות של חיפוש היא **O(n)** ברשימה מקושרת
  - לעומת **O(logn)** במערך **ממוין** (ע"י חיפוש בינארי)

היום נלמד על סוג מתוחכם של רשימה מקושרת

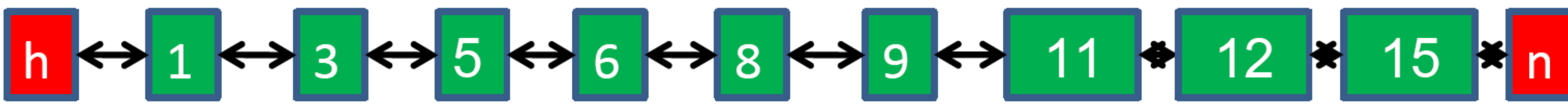
- מאפשרת הכנסת / מחיקת / חיפוש איבר בזמן **O(logn)** בדומה לעצי חיפוש בינריים מאוזנים



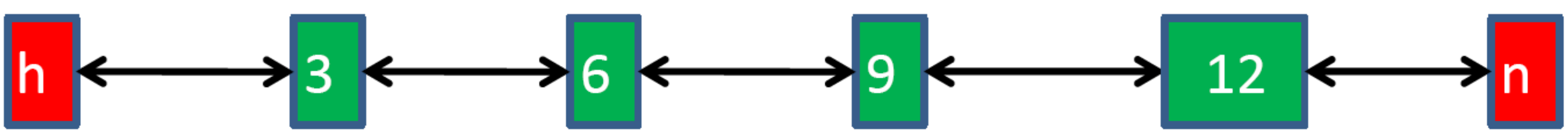
רשימת דילוג – Skip List

רשימת דילוג היא אוסף של **רשימות ממוינות**

- כל אחת היא **תת-רשימה** של זאת אחריה
- לדוגמה, נבנה רשימה מקושרת דו-כיוונית לאוסף הממוין {1,3,5,6,8,9,11,12,15}

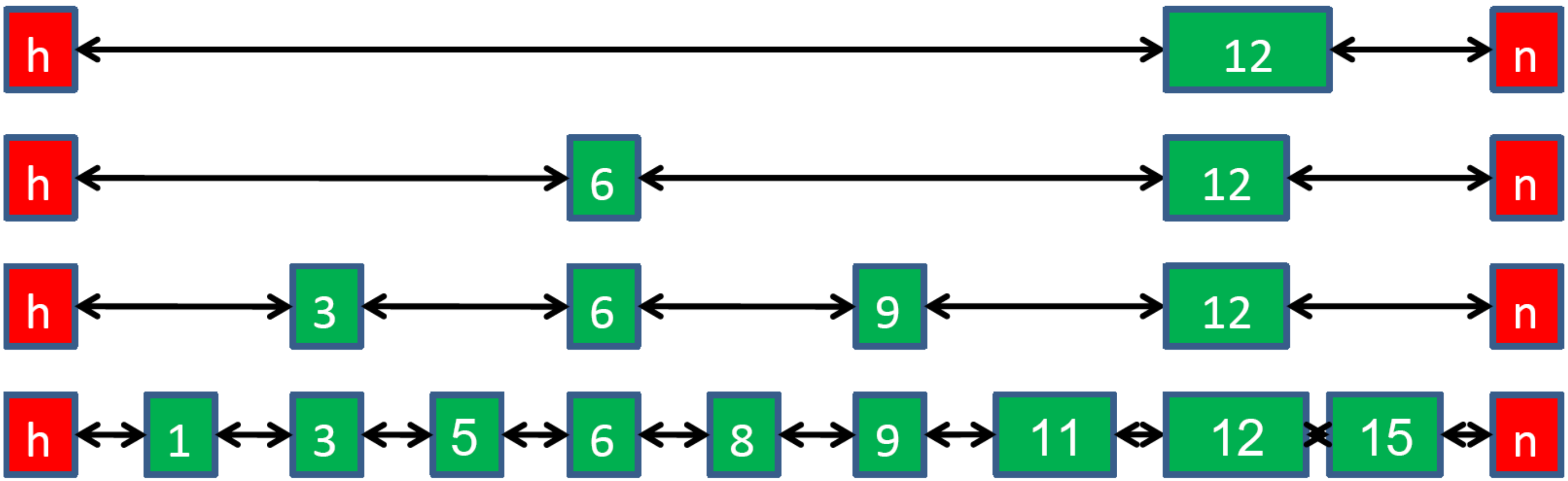


- שימו לב שהראש **ריק** ושהאיבר האחרון הוא **null**
- עכשיו נבנה עוד רשימה, שהיא **דילוג** של הרשימה למעלה, כלומר יש בה רק **כל איבר שני**



רשימת דילוג – Skip List

נמשיך כך עד שנקבל אוסף של רשימות, כל אחת דילוג על זאת שאחריה:



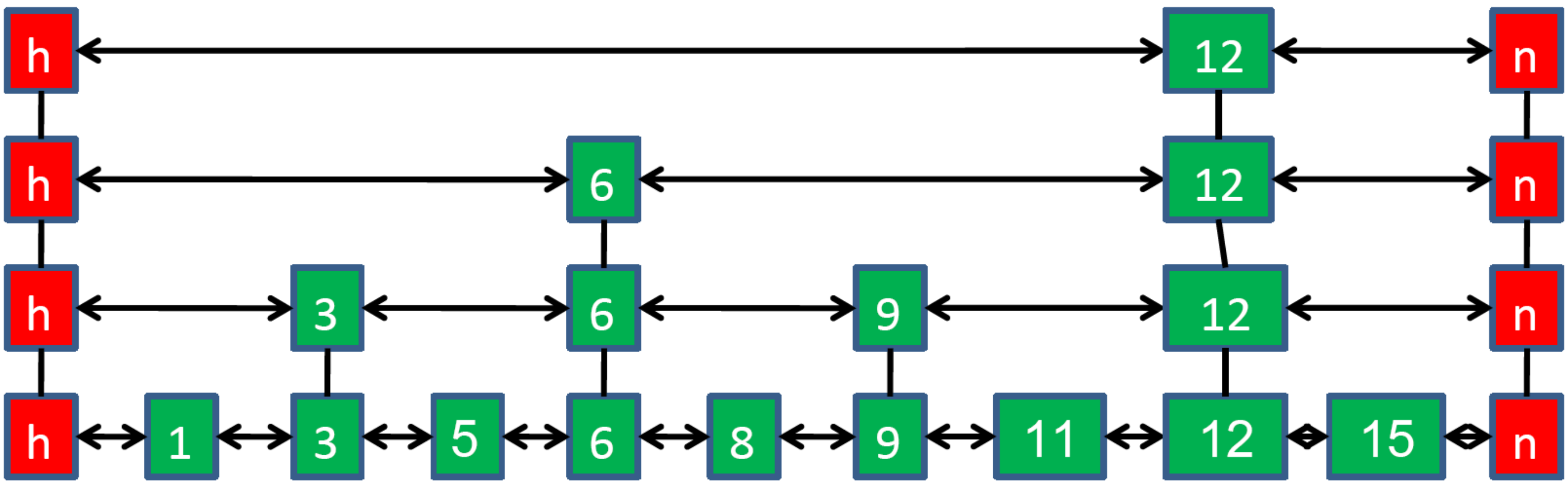
- ברור שמספר הרשימות הוא **O(logn)**
- כמות הזיכרון הנדרשת היא רק פי **2** מהרשימה הראשונית

$$\mathbf{n} + \mathbf{n/2} + \mathbf{n/4} + \ldots + \mathbf{1} = \mathbf{n}\cdot(\mathbf{1} + \mathbf{1/2} + \mathbf{1/4} + \ldots) < \mathbf{2n}$$

רשימת דילוג – Skip List

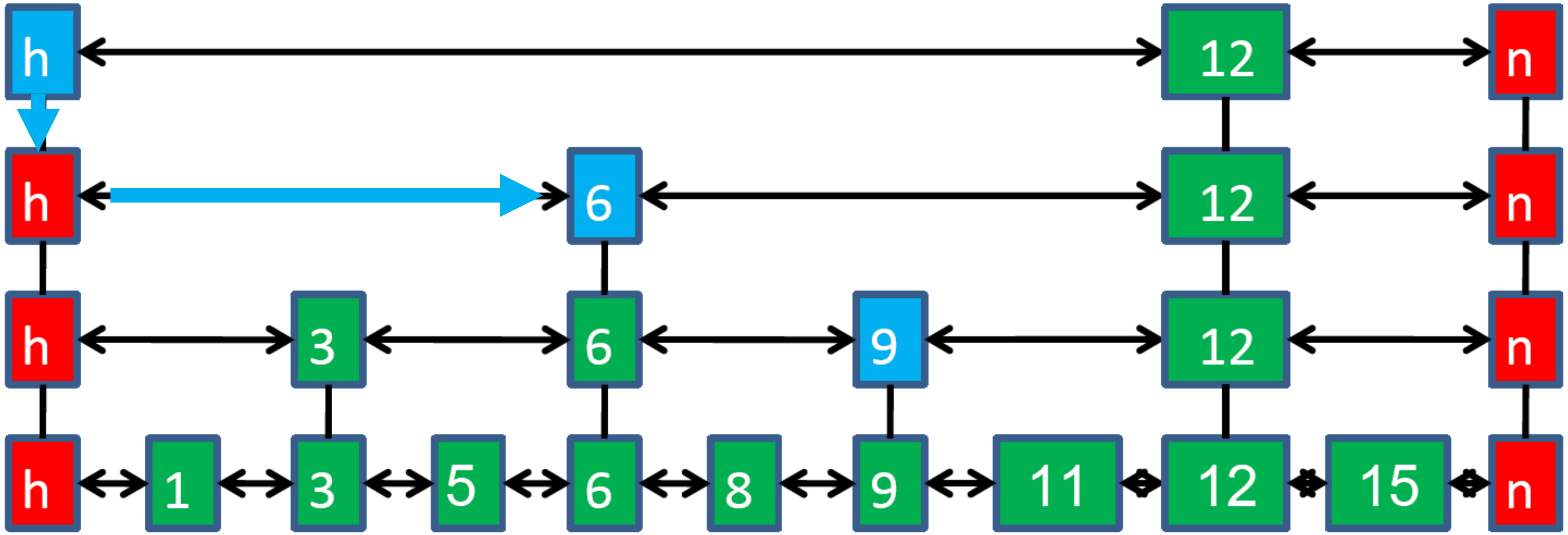
עכשיו נחבר את הרשימות ביחד:

- ניתן לכל צומת להצביע על העותק של עצמו ברמות שכנות
- לכל צומת ישנו קישור ל- **next**, **previous**, **up**, **down**



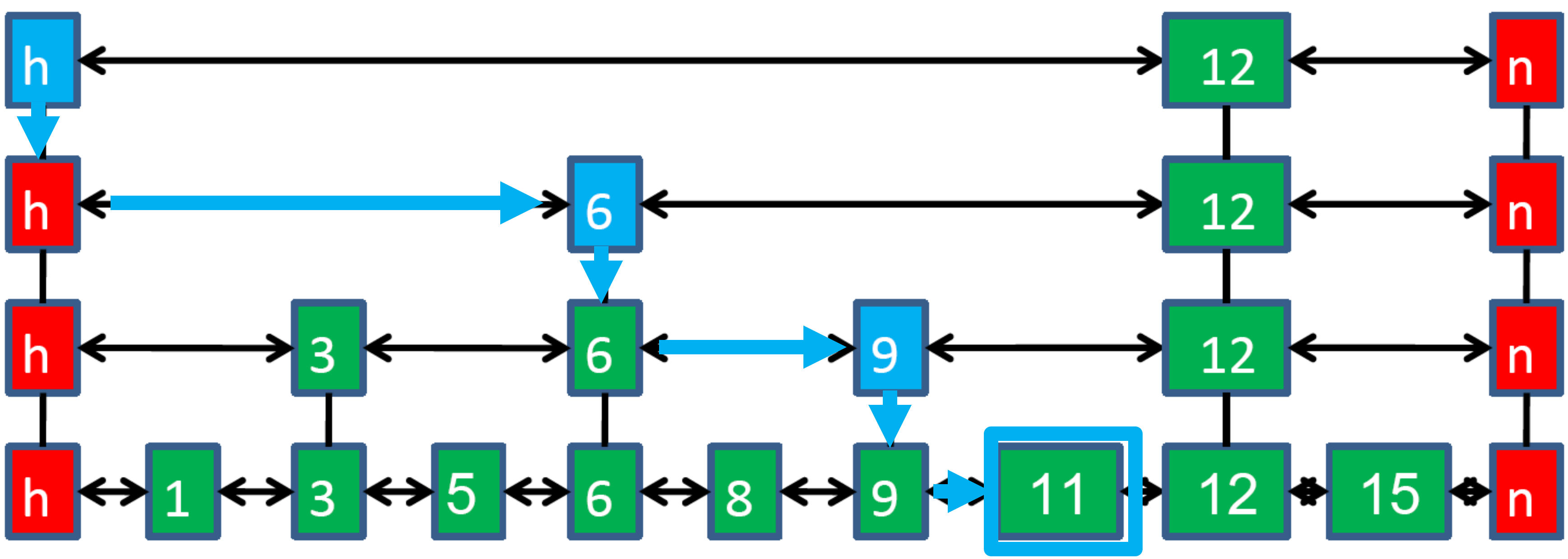
חיפוש ברשימת דילוג

- מתחילים מהרשימה העליונה, ומוצאים את הצומת עם הערך הגדול ביותר שעדיין קטן או שווה לערך השאילתא
- למשל, אם נחפש **11**, אז נבדוק בשורה העליונה ונראה ש- **12** גדול מ- **11** , ולכן נבחר להישאר בראש הרשימה
- בשורה הבאה נתקדם ל- **6** כי ימינה ממנו **12** כבר גדול מדי



חיפוש ברשימת דילוג

- בשורה הבאה נתקדם ימינה ל- **9**
- ובשורה האחרונה ל- **11**
- עבור כל אחת מהשורות, ביצענו לכל היותר **שני צעדים** – כי אף פעם אין צורך בתוך אותה שורה ללכת פעמיים ימינה
- לכן זמן החיפוש הוא **O(logn)**

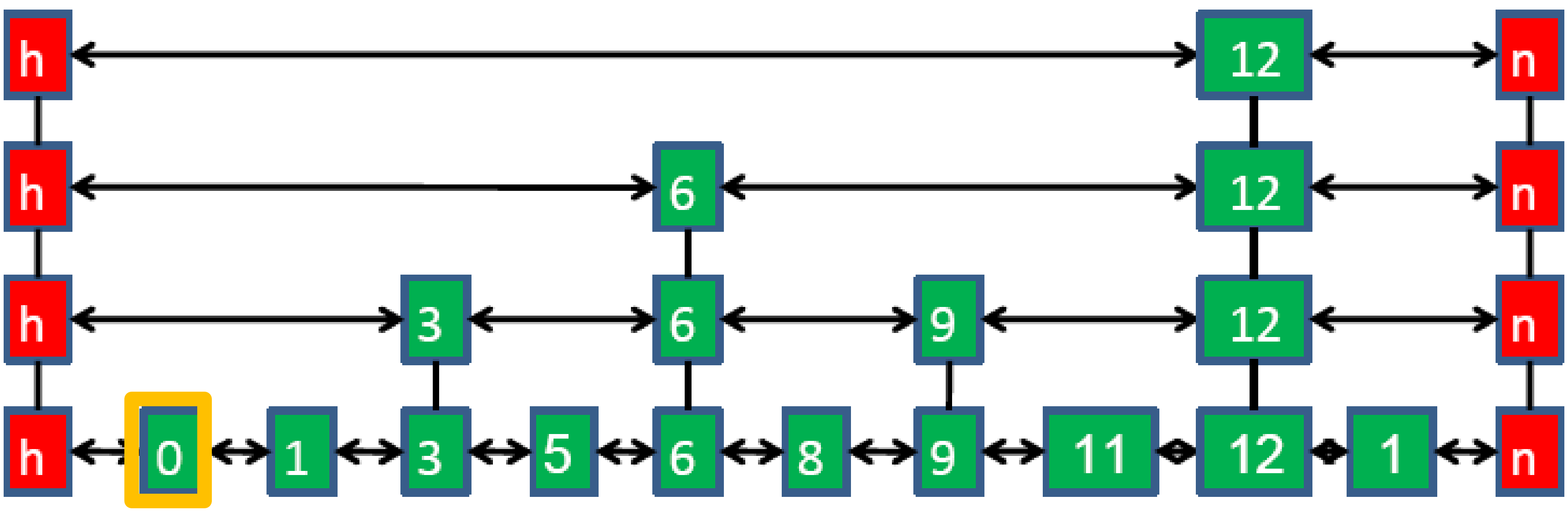


הכנסת איבר לרשימת דילוג

יש בעיה להוסיף למאגר איבר חדש, כי **קשה** לשמור **ביעילות** על התכונה שכל איבר שני נמצא ברשימה מעל

- לדוגמה, אם נוסיף את **0** – נצטרך לשנות את כל האיברים שנמצאים ברשימה מעליה, ובעצם בכל הרשימות כולם

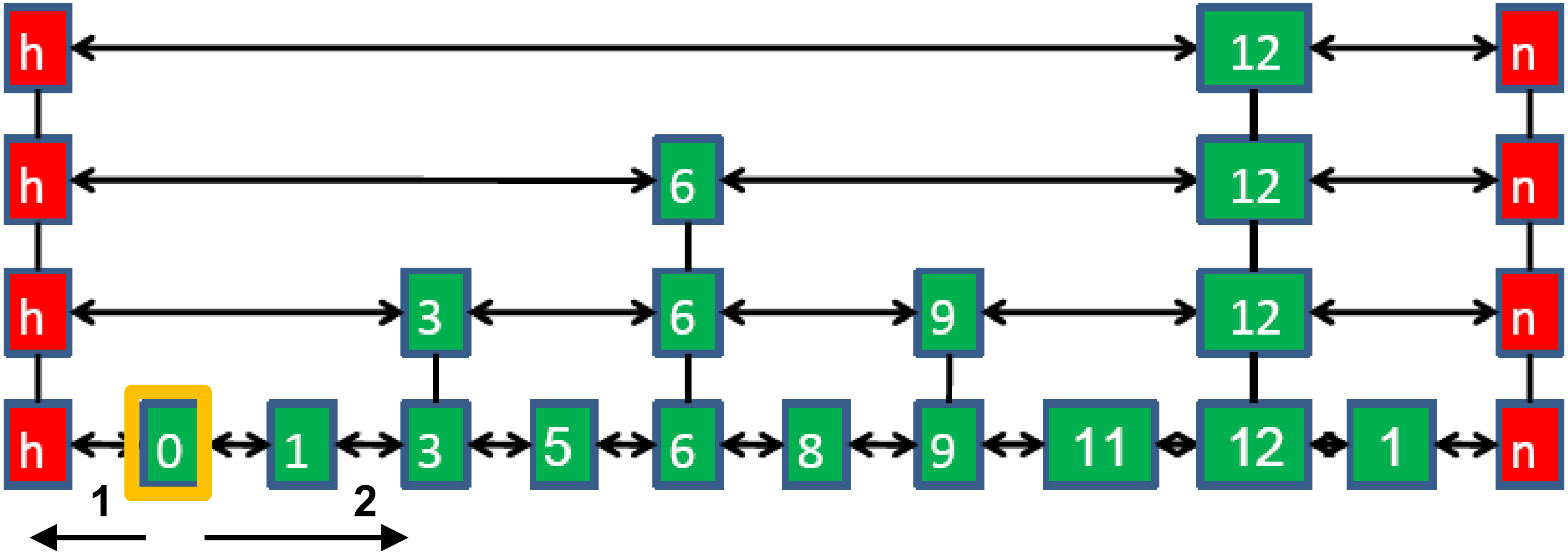
לכן **נחליש** את הדרישה שכל איבר **שני** נמצא ברשימה מעל



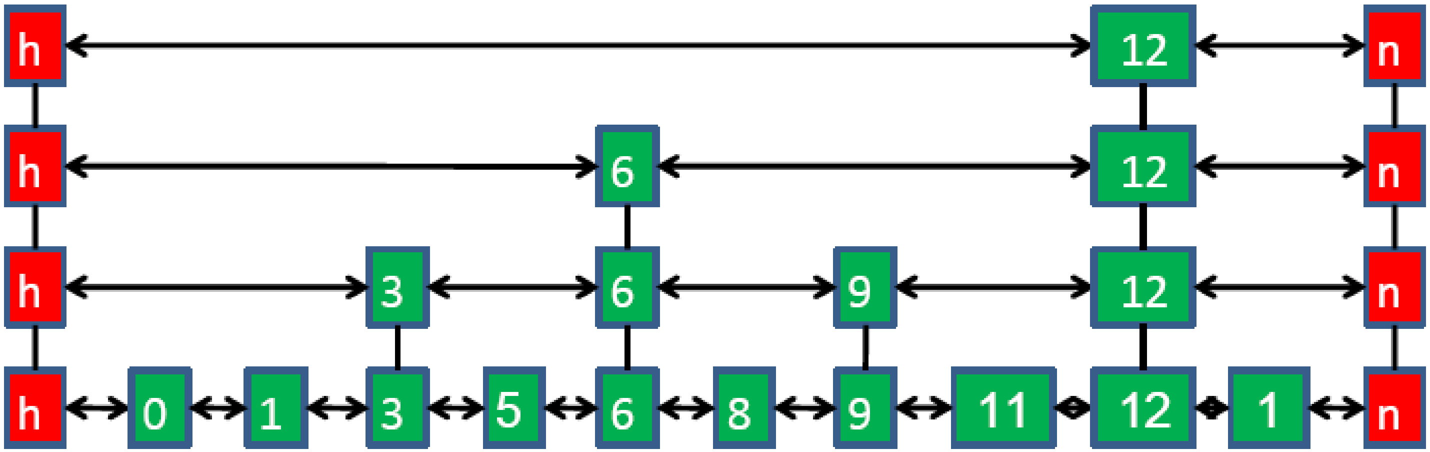
הכנסת איבר לרשימת דילוג

נחליש את הדרישה שכל איבר **שני** נמצא ברשימה מעל:

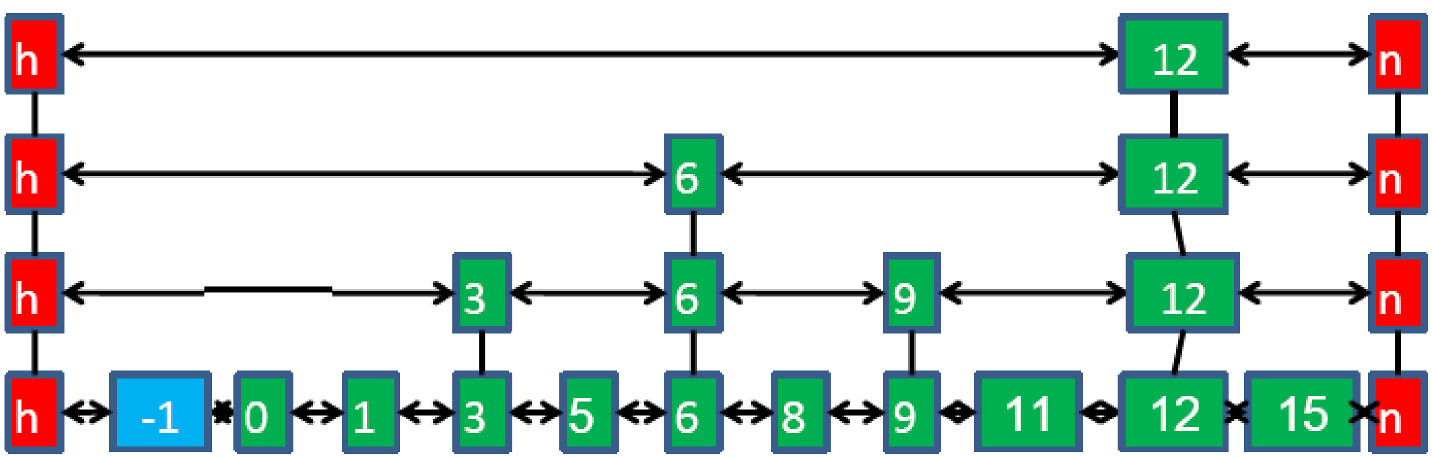
- נסתפק שכל איבר **שני** **או** **שלישי** נמצא ברשימה מעל
- למשל, אם נוסיף איבר **0**, אז הרשימה נשארת תקינה
- **ספירת הדילוג**: הולכים מ- **0** שמאלה עד לראשון עם אבא וימינה עד לראשון עם אבא וסופרים כמה איברים יש ביניהם



הכנסת איבר לרשימת דילוג

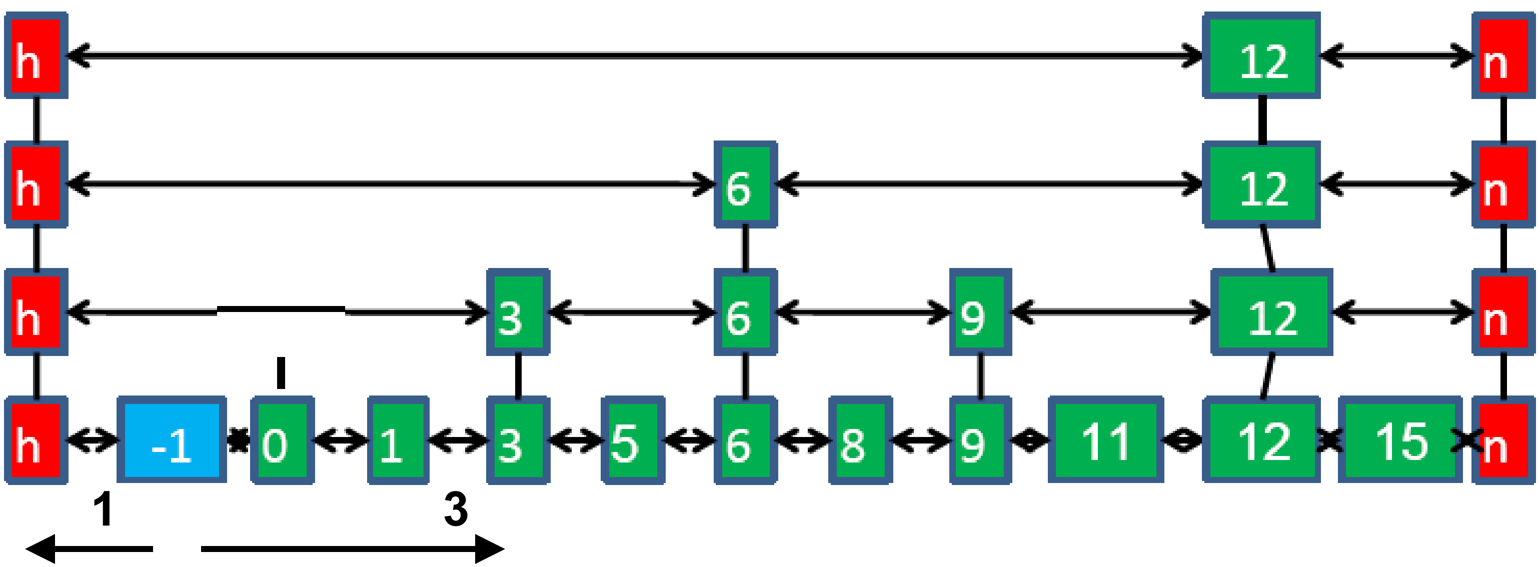


- אם עכשיו נוסיף את האיבר **-1** אז שלושת האיברים **-1,0,1** אינם נמצאים ברשימה מעל – ז"א דילוג לא חוקי בגודל **4**



הכנסת איבר לרשימת דילוג

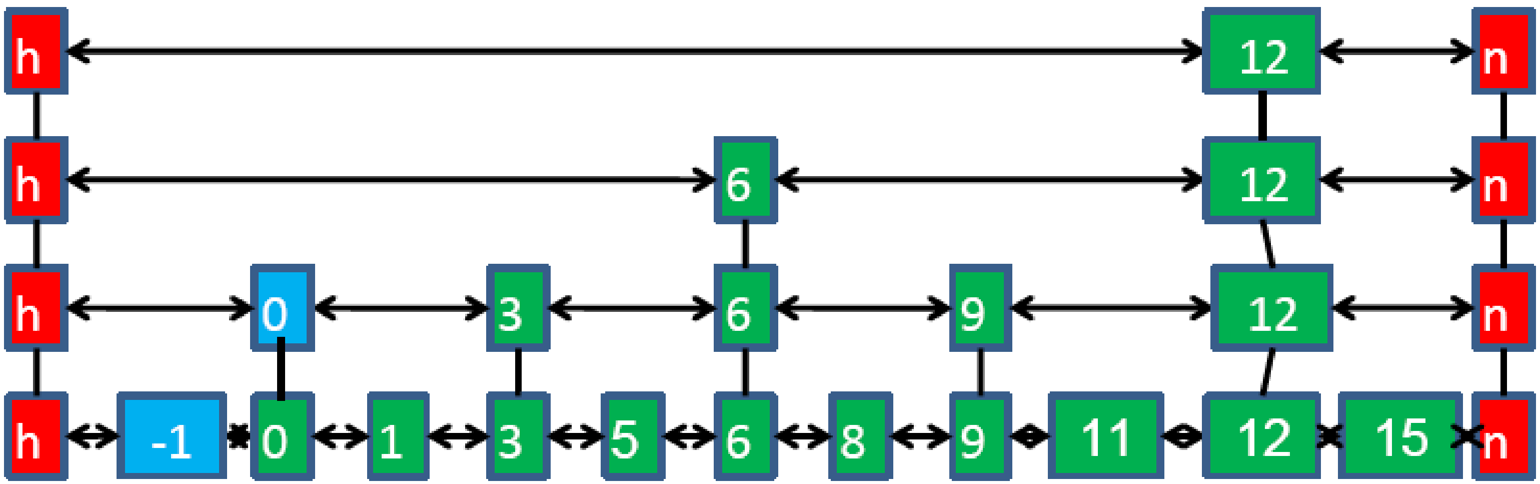
- **באופן פרקטי**: הולכים מ- **1-** שמאלה עד לראשון עם אבא וימינה עד לראשון עם אבא וסופרים כמה איברים יש ביניהם
- מכיוון שיש שלושה איברים כאלו (**-1,0,1**) אז צריך להוסיף לאמצעי (**0**) אבא, ולקשרו מימין ומשמאל לשני האבות הנ"ל



- כאן סיימנו כי הרשימה השלישית מלמטה כבר תקינה

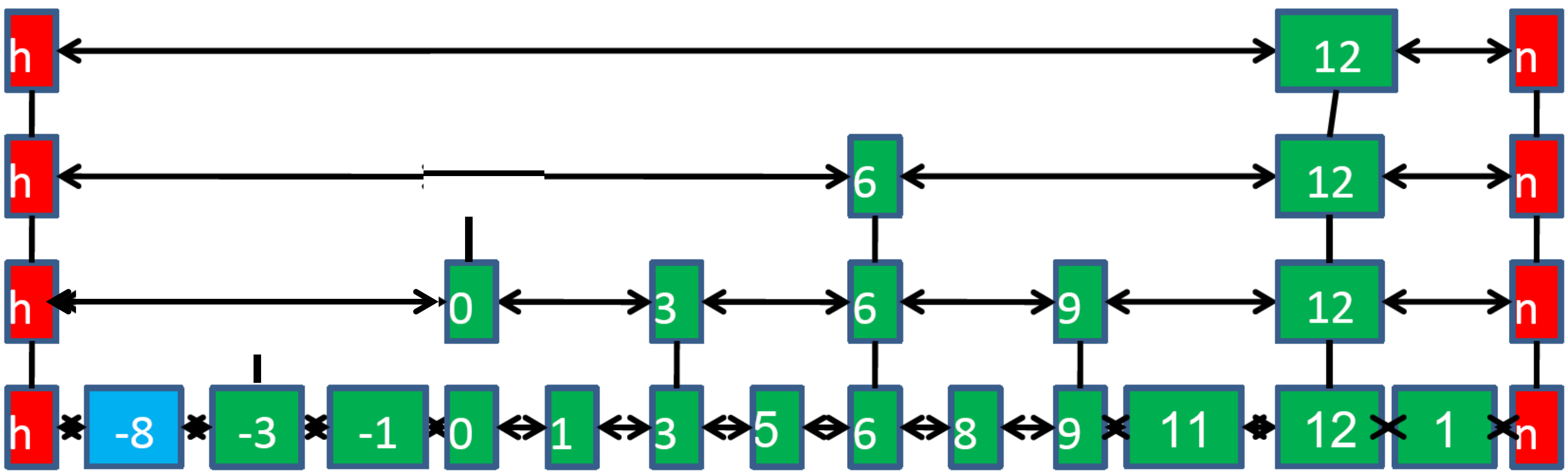
הכנסת איבר לרשימת דילוג

- באופן כללי, יתכן שהוספה של איבר לרשימה השנייה תגרום שעכשיו יהיו **3** איברים רצופים שלא נמצאים ברשימה שמעל
- אז נצטרך לתקן גם את הרשימה מעל
- זה יכול להמשך עד הרשימה העליונה



הכנסת איבר לרשימת דילוג

- לדוגמה, אם עכשיו נכניס את האיברים **-3,-8**
- אז ברשימה הראשונה יהיה רצף של **-8,-3,-1** ללא אבות
- לכן נצטרך להוסיף את האיבר **-3** לרשימה השנייה
- זה גורם לרצף של **-3,0,3** ללא אבות ברשימה השנייה
- לכן נצטרך להוסיף **0** לרשימה השלישית

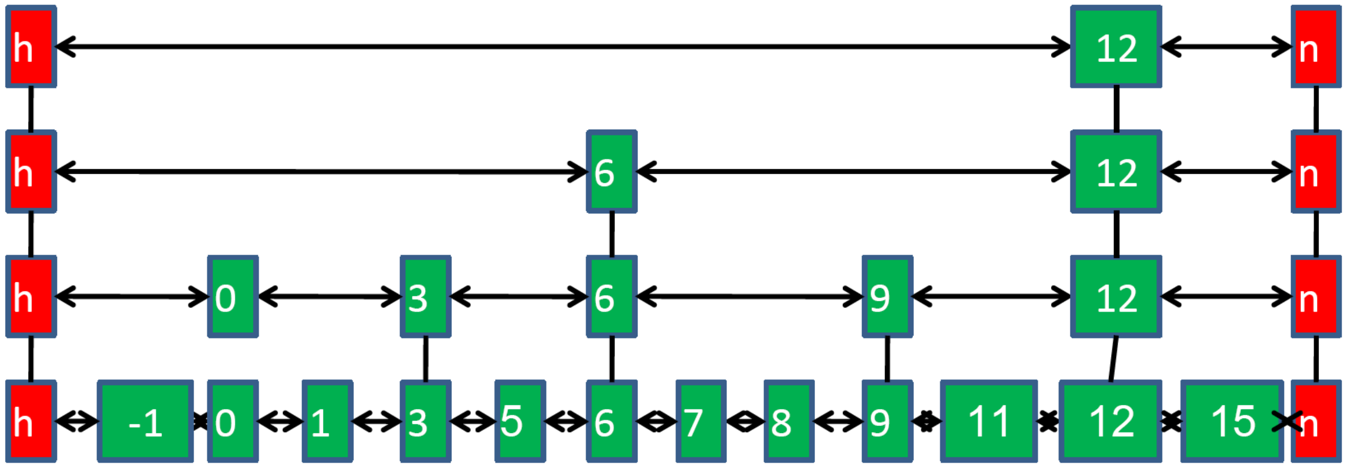


- זה יכול להמשך עד הרשימה העליונה
- כל הפעולות הן **מקומיות** – לכן זמן הכנסת איבר **O(logn)**

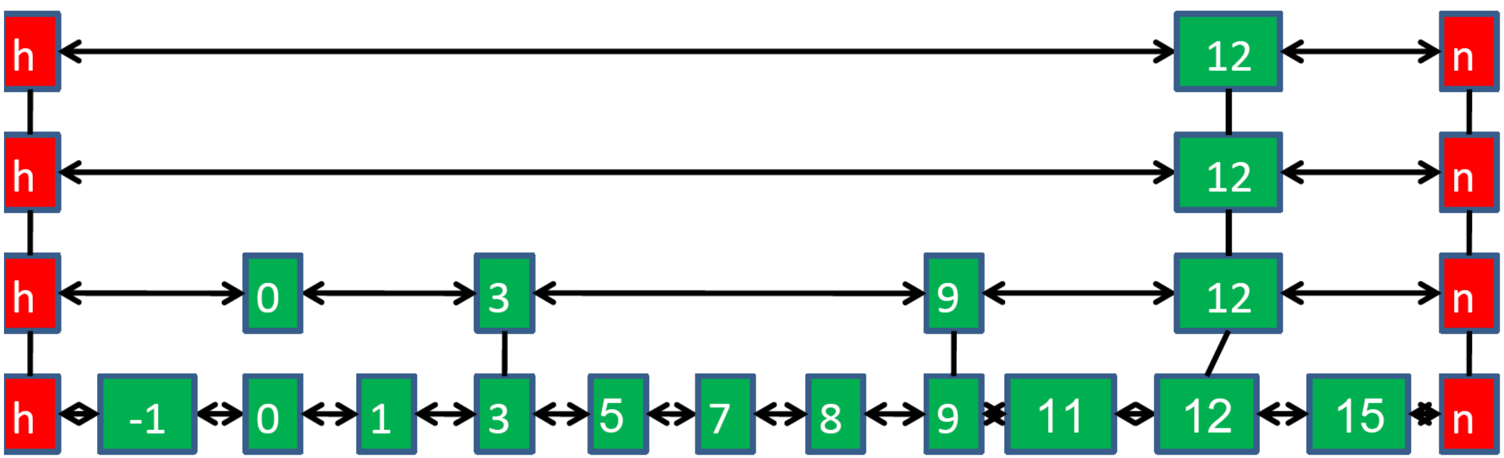
מחיקת איבר מרשימת דילוג

תחילה נמחק את האיבר **מכל** רשימה בה הוא מופיע

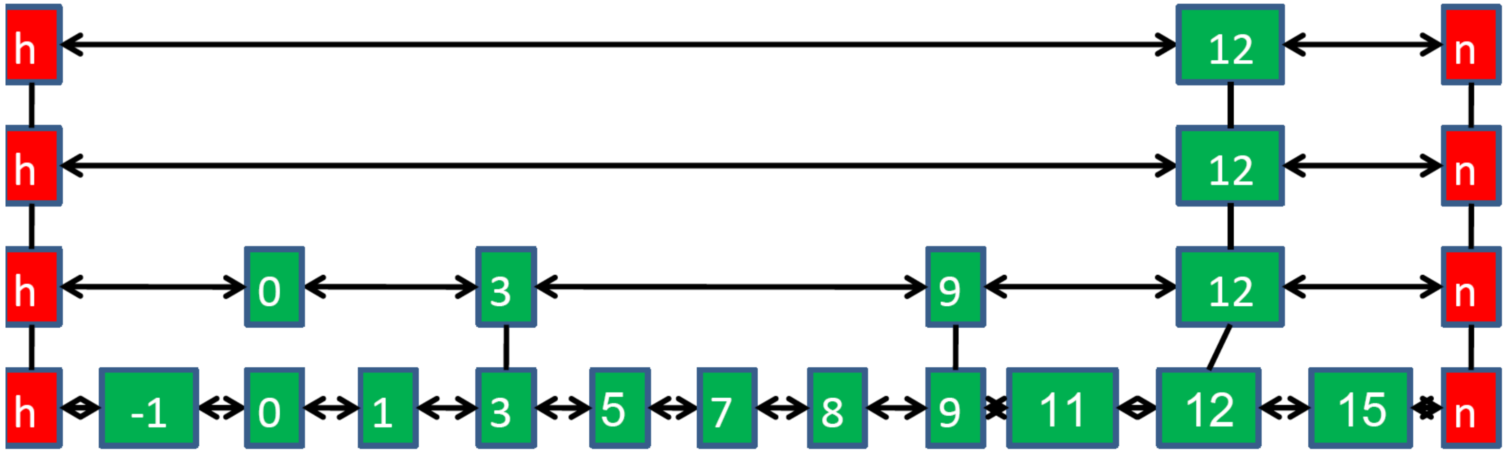
- לדוגמה, כדי למחוק את האיבר **6** מרשימת הדילוג הבאה



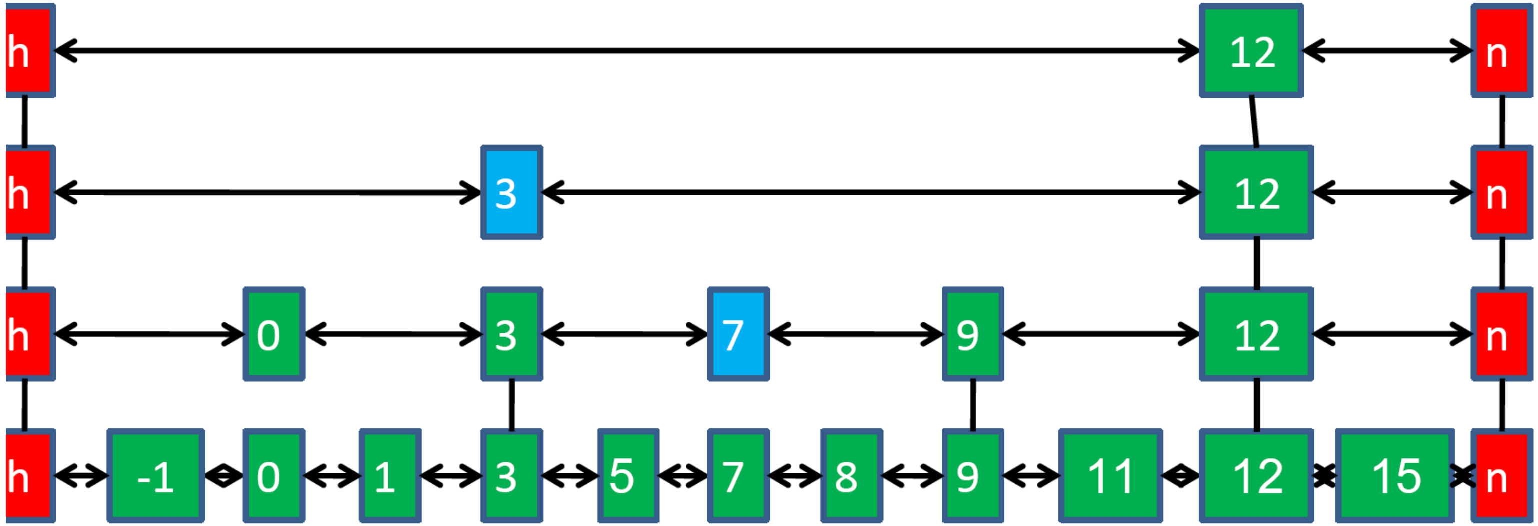
צריך למחוק את האיבר מכל השורות



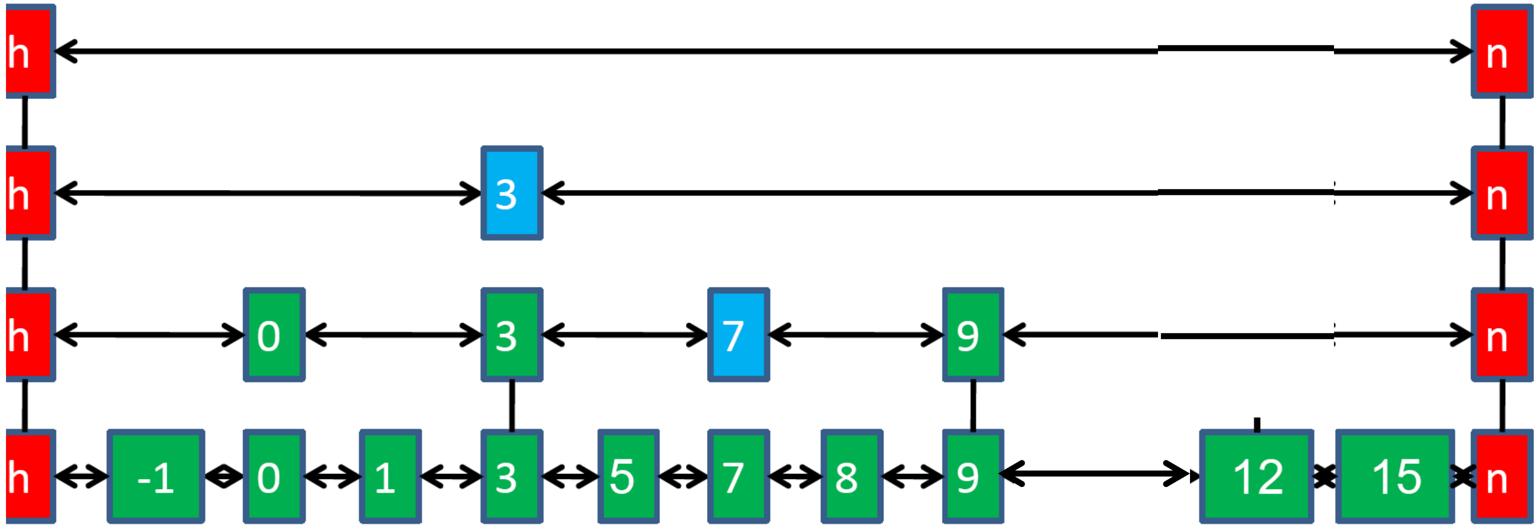
מחיקת איבר מרשימת דילוג



ואז לתקן את כל הרשימות **מלמטה למעלה**:

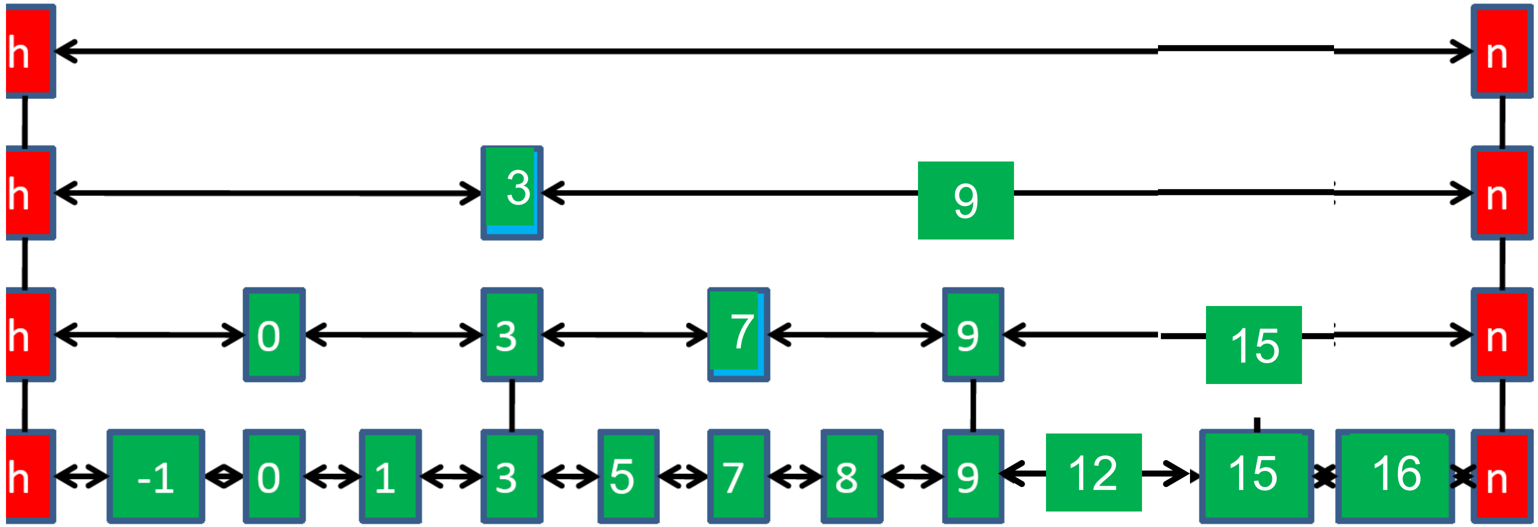


מחיקת איבר מרשימת דילוג



- עכשיו נרצה למחוק את **11**
- לאחר מחיקתו, **12** ו- **9** בשורה השניה מלמטה הם **בדילוג 0**
- נחליט שמחפשים את הראשון **משמאל** (שרירותי) עם אב (**9**)
- נשאיר אב זה כפי שהוא ונתקן את האב מצד ימין (**12**)
- כלומר נמחק את ה- **12** מהשורות השניה, שלישית ורביעית
- לאחר פעולה זו בדוגמה לעיל, סיימנו

מחיקת איבר מרשימת דילוג



- אבל אם בשורה הראשונה מלמטה היה גם למשל **16**
- אז צריך לבצע תיקונים מלמטה למעלה - ע"י הוספת **16** לשורה השנייה ו- **9** לשורה השלישית
- שימו לב כי בכל תהליך התיקון, השינויים הם **מקומיים** בכל שורה, כלומר בכל שורה התיקונים אורכים **זמן קבוע O(1)**
- לכן זמן ריצה למחיקת איבר תלוי בכמות השורות - **O(logn)**
- לא ידוע אלגוריתם עם זמן לוגריתמי **לאיחוד 2** רשימות דילוג

להעמקה - מדוע הדילוג הוא דווקא של **2**?

כלומר מדוע בחרנו את כמות השורות להיות **logn**?

ניסוי ראשון - שתי שכבות: שתי רשימות מקושרות דו-כיווניות ממוינות מקטן לגדול.

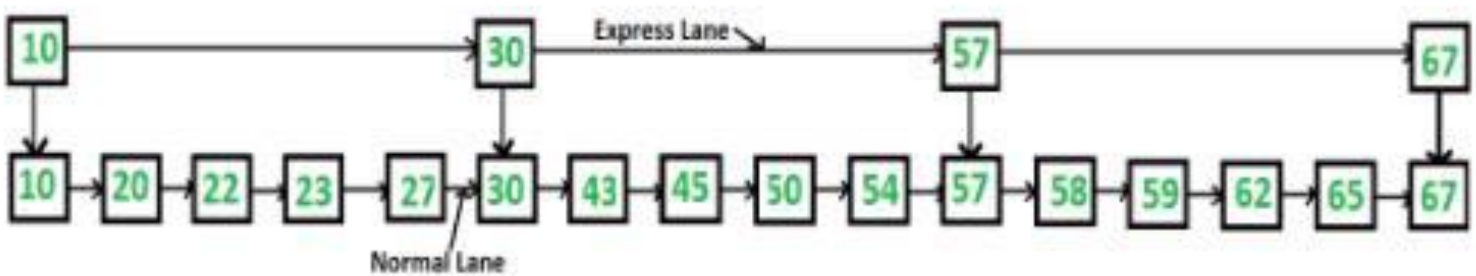
השכבה התחתונה היא רשימה המכילה את כל האיברים L_1, השכבה העליונה – רשימת הנציגים L_2.

נסמן: $n = |L_1|$ - מספר איברים ב-L_1 . $k = |L_2|$ - מספר איברים ב-L_2, נניח כי המרחק בין האיברים ברשימת הנציגים L_2 מספר קבוע.

זמן חיפוש האיבר הוא

$$T(k) = k + \frac{n}{k}$$

דוגמה 1 של רשימת דילוגים בעלת שתי שכבות ו- 16 איברים.



להעמקה

<u>ניסוי ראשון - שתי שכבות</u>: שתי רשימות מקושרות דו-כיווניות ממוינות מקטן לגדול.
השכבה התחתונה היא רשימה המכילה את כל האיברים L_1, השכבה העליונה – רשימת הנציגים L_2.
נסמן: $n = |L_1|$ - מספר איברים ב-L_1 . $k = |L_2|$ - מספר איברים ב-L_2, נניח כי המרחק בין האיברים ברשימת הנציגים L_2 מספר קבוע.
זמן חיפוש האיבר הוא

$$T(k) = k + \frac{n}{k}$$

פונקציה T מגיעה למינימום, כאשר $k = \sqrt{n}$:

$$T'(k) = 1 - \frac{n}{k^2}, \quad T'(k) = 0, n = k^2, \ k = \sqrt{n}$$

<u>אז זמן החיפוש מינימאלי הוא:</u>

$$T(k) = \sqrt{n} + \frac{n}{\sqrt{n}} = 2\sqrt{n} = O(\sqrt{n})$$

וכמות הזיכרון הוא $n + \sqrt{n}$ (space complexity).

להעמקה

<u>**ניסוי שני - שלוש שכבות:**</u> $n = |L_1|$, $k = |L_2|$, $m = |L_3|$.

$$T(k,m) = m + \frac{k}{m} + \frac{n}{k}$$

$$T'_m = 1 - \frac{k}{m^2} = 0, \qquad k = m^2,$$

$$T'_k = \frac{1}{m} - \frac{n}{k^2} = \frac{1}{m} - \frac{n}{m^4} = \frac{m^3 - n}{m^4} = 0 \quad \rightarrow$$

$$m^3 = n, \qquad m = \sqrt[3]{n}, \quad k = \sqrt[3]{n^2}$$

<u>אז זמן החיפוש מינימאלי הוא:</u>

$$T(k,m) = m + \frac{k}{m} + \frac{n}{k} = \sqrt[3]{n} + \frac{\sqrt[3]{n^2}}{\sqrt[3]{n}} + \frac{n}{\sqrt[3]{n}} = 3\sqrt[3]{n}$$

להעמקה

<u>מקרה כללי – k שכבות -</u> זמן החיפוש המינימאלי הוא

$$T = k\sqrt[k]{n} = kn^{1/k}$$

k אופטימאלי:

$$T'_k = n^{1/k} + k\left(-\frac{1}{k^2}\right)n^{1/k}ln(n) = n^{1/k} - \frac{n^{1/k}\ln(n)}{k} = 0 \rightarrow \boldsymbol{k} = \mathbf{ln}(\boldsymbol{n})$$

$$T = ln(n) \cdot n^{1/\ln(n)} = ln(n) \cdot n^{log_n(e)} = ln(n) \cdot e \rightarrow O(\log(n))\ ,$$

זמן החיפוש הוא $T = O(\log(n))$, הרפיה (relaxation): פער של 1 או 2 איברים.

לכן בחרנו דווקא **k = logn**