הוכן ע"י אמיר רובינשטיין

מבני נתונים
ומבוא לאלגוריתמים

נושא 9

עצי AVL
AVL trees

בתוכנית

סעיף 10.3 בספר "מבני נתונים"

section 9.8 in “Data structures, algorithms and Software principles in C”

- נלמד על עצי AVL – סוג של עצי חיפוש מאוזנים
- נכיר את פעולת ה"גלגול", שמטרתה לאזן את העץ

מוטיבציה

תזכורת

- מילון הוא ADT המוגדר ע"י פעולות הבאות: Insert, Delete, Search

ולפעמים גם: Minimum, Maximum, Successor, Predecessor

- ראינו מימוש למילון באמצעות עץ חיפוש בינארי

סיבוכיות הפעולות הנ"ל היא $\Theta(h)$, כאשר h הוא גובה העץ.

- $h = \Theta(\log n)$ במקרה הטוב ובממוצע
- $h = \Theta(n)$ במקרה הגרוע

- ישנם כמה סוגים של עצי חיפוש, שנקראים מאוזנים: $h = \Theta(\log n)$ במקרה הגרוע.

❖ עצי AVL (במצגת זו)
❖ עצי 2-3 (שהם מקרה פרטי של עצי +B)
❖ עצים אדומים שחורים (פרק 13 בספר הלימוד)
❖ ועוד...

עצי AVL

עצי AVL הומצאו ע"י שני מדעני מחשב: **A**delson-**V**elsky, **L**andis.

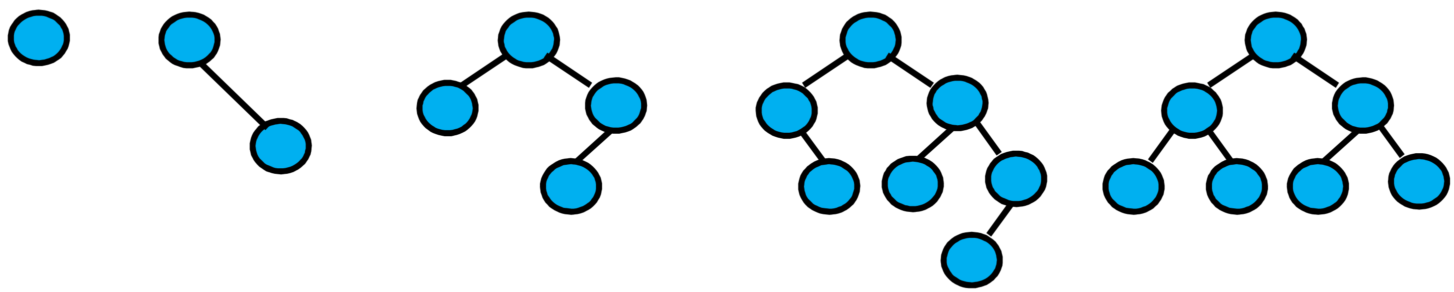
הגדרה

עץ AVL הוא עץ חיפוש בינארי שבו לכל צומת v מתקיימת התכונה:

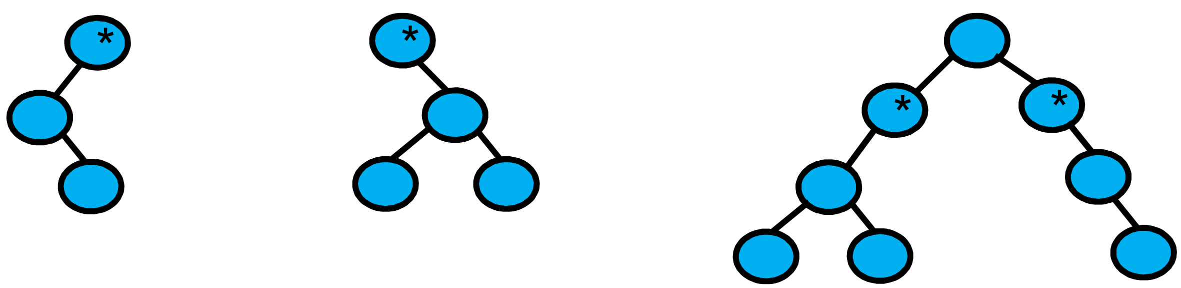
$$\left|h(\mathbf{left}[v]) - h(\mathbf{right}[v])\right| \le \mathbf{1}$$

תזכורת: גובה של עץ ריק מוגדר להיות 1-.

דוגמאות



דוגמאות נגד



* צמתים בהם מופר האיזון

חסם לגובה עץ AVL

טענה

עבור עץ AVL בעל n צמתים וגובה h מתקיים $h = \Theta(\log n)$.

הוכחה

חסם תחתון: מהו מספר הצמתים המקסימלי בעץ AVL בגובה h?

$$n \leq 2^{h+1}-1$$

$$h \geq \log(n+1) -1$$

$$h = \Omega(\log n)$$

חסם עליון: מהו מספר הצמתים המינימלי בעץ AVL בגובה h?

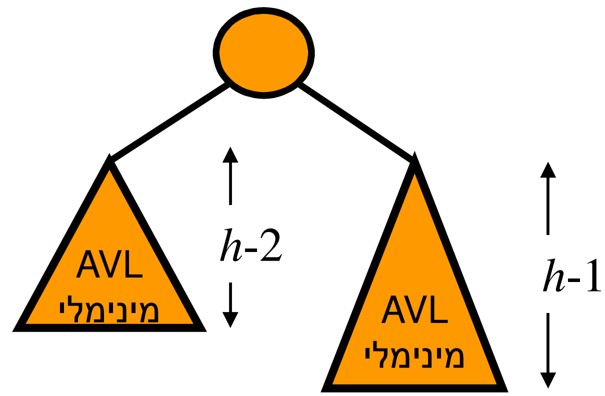
נוכיח כי $n = \Omega(\Phi^h)$ כאשר Φ הוא יחס הזהב: $\Phi = \frac{1+\sqrt{5}}{2}$

ומכאן נובע $h = O(\log_\Phi n)$.

חסם לגובה עץ AVL

חסם עליון (המשך)

מהו מספר הצמתים המינימלי בעץ AVL בגובה h?



(או הפוך)

עצים בעלי מבנה כזה נקראים **עצי פיבונאצ'י**.

תזכורת: סדרת פיבונאצ'י: $f_n = f_{n-1} + f_{n-2}$ $\quad f_1 = f_2 = 1$

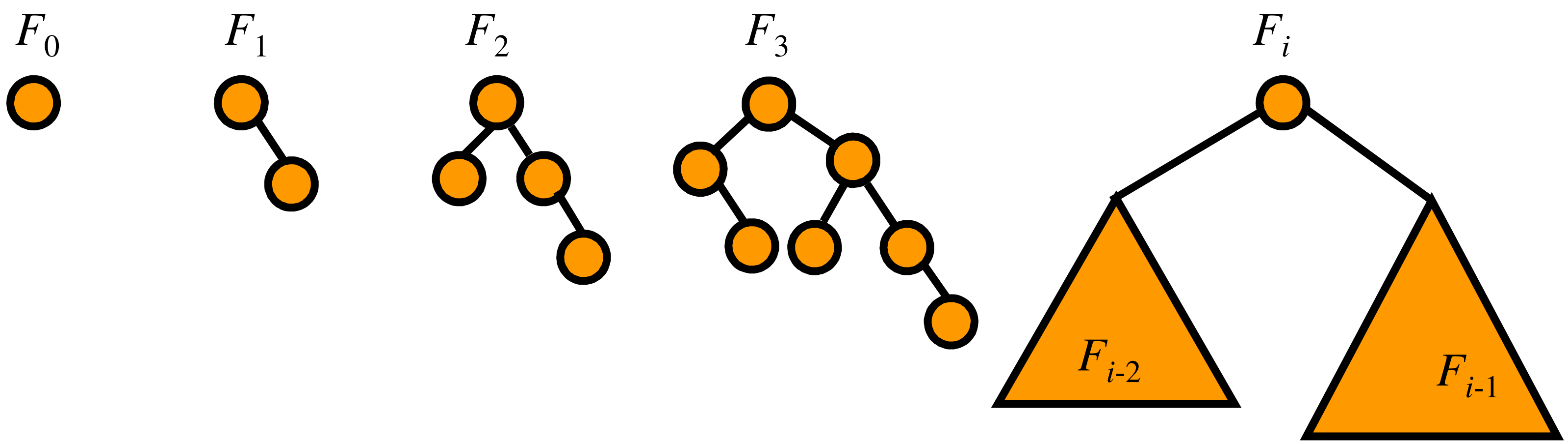
$$f_n = \frac{\Phi^n - \overline{\Phi}^n}{\sqrt{5}}$$ כאשר $$\Phi = \frac{1+\sqrt{5}}{2} \approx 1.608 \qquad \overline{\Phi} = \frac{1-\sqrt{5}}{2} \approx -0.618$$

Φ נקרא יחס הזהב

חסם לגובה עץ AVL

חסם עליון (המשך)

הגדרת עצי פיבונאצ'י ברקורסיה:



תכונות (תרגיל: הוכיחו כל אחת מהתכונות)

1. גובהו של F_h הוא h.
2. $|F_h| = |F_{h-1}| + |F_{h-2}| + 1$ (כאשר $|F_i|$ הוא מספר הצמתים ב- F_i).
3. F_h הוא עץ AVL בעל מספר צמתים מינימלי מבין כל עצי ה- AVL בגובה h.

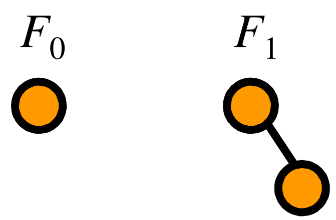
חסם לגובה עץ AVL

חסם עליון (המשך)

טענה: מספר הצמתים ב- F_h הוא $|F_h| = f_{h+3} - 1$ צמתים כאשר f_i הוא מספר פיבונצ'י ה- i.

הוכחה: באינדוקציה על h.

בסיס: עבור h=0 ו- h=1 הטענה מתקיימת:



$f_3 = 2$ $f_4 = 3$

צעד: נניח שהטענה נכונה לכל $h' < h$.

$$\left|F_h\right| = \left|F_{h-1}\right| + \left|F_{h-2}\right| + 1 = (f_{h+2} - 1) + (f_{h+1} - 1) + 1 = f_{h+3} - 1$$

ממה נובע כל מעבר?

חסם לגובה עץ AVL

<u>חסם עליון (המשך)</u>

<u>נסכם</u>: עבור עץ AVL בעל n צמתים וגובה h מתקיים :

(לפי תכונה 3)

$$n \geq |F_h|$$

(לפי הטענה)

$$n \geq |F_h| = f_{h+3} - 1 = \frac{\Phi^{h+3} - \overline{\Phi}^{h+3}}{\sqrt{5}} - 1 \geq \frac{\Phi^{h+3}}{\sqrt{5}} - 2$$

כלומר:

$$\sqrt{5}(n+2) \geq \Phi^{h+3}$$

נוציא $\log_\Phi$:

$$h + 3 \leq \log_\Phi\left(\sqrt{5}(n+2)\right)$$

$$h \leq \log_\Phi(n+2) + \log_\Phi\left(\sqrt{5}\right) - 3$$

$$\boldsymbol{h = O(\log n)}$$

סיבוכיות זמן לפעולות בעץ AVL

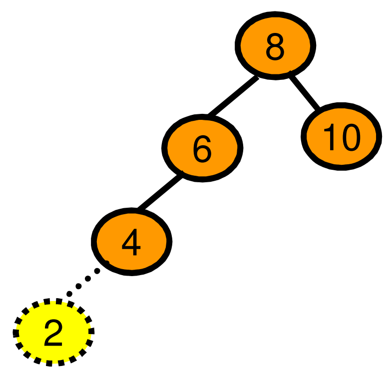
מסקנה: כל השאילתות מתבצעות על עץ AVL בזמן לוגריתמי במספר צמתיו.

- AVL-Search
- AVL-Minimum, AVL-Maximum
- AVL-Successor, AVL-Predecessor

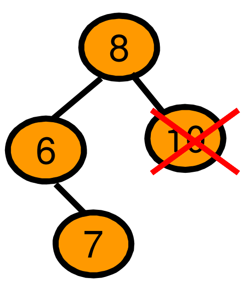
מה לגבי הכנסה והוצאה של איבר?

גם כן בזמן לוגריתמי, אבל פעולות אלו עלולות להפר את האיזון של העץ.

למשל: הוספת 2 לעץ הבא:



או מחיקת 10 מהעץ הבא:

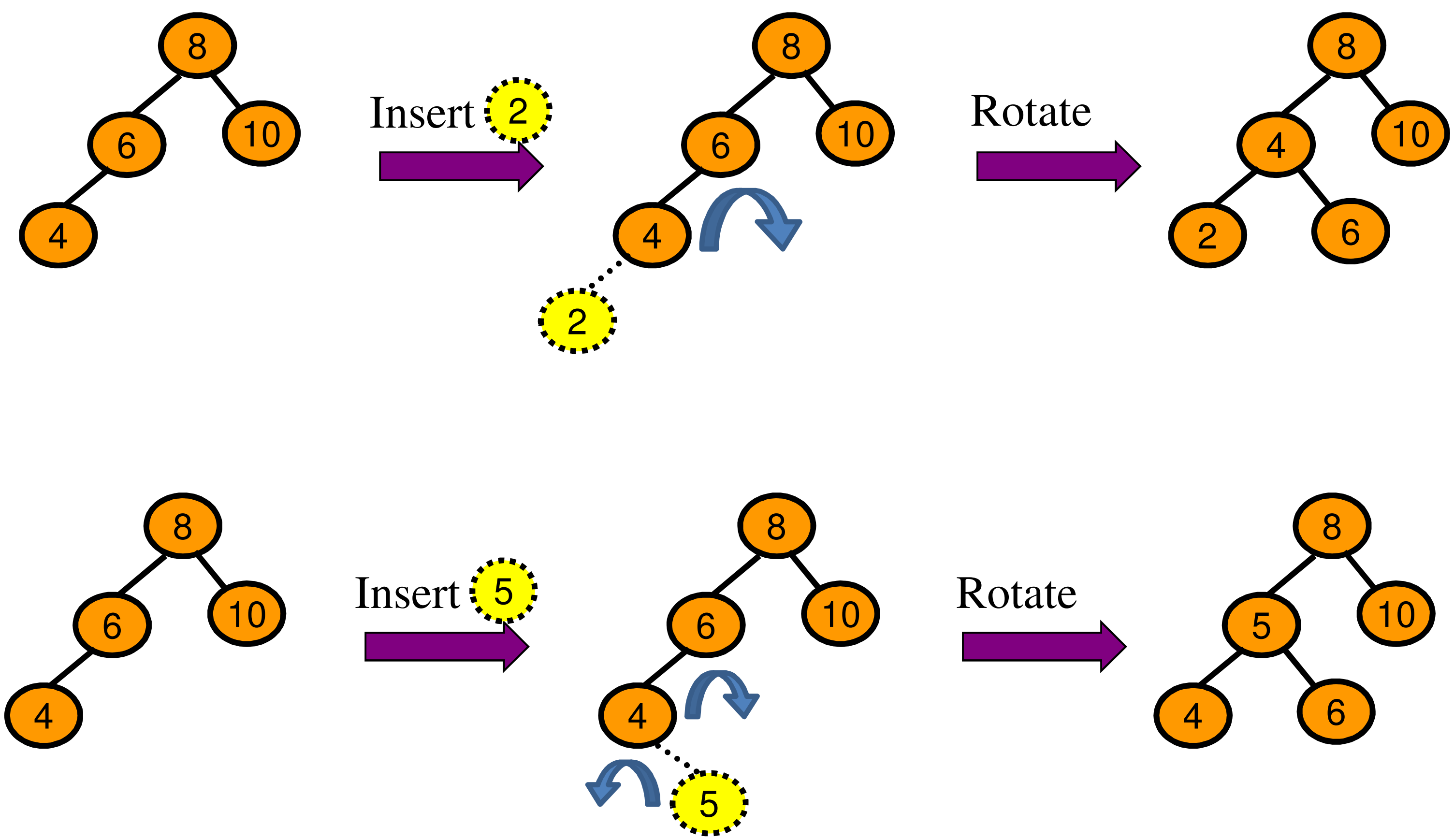


תיקון לאחר הכנסה - גלגולים

<u>מה אפשר לעשות אם הכנסה גורמת להפרת האיזון?</u>

לבצע <u>*גלגול*</u> בעץ – שינוי של כמה מצביעים כדי לאזן את הפרש הגבהים.

הנה כמה דוגמאות פשוטות.



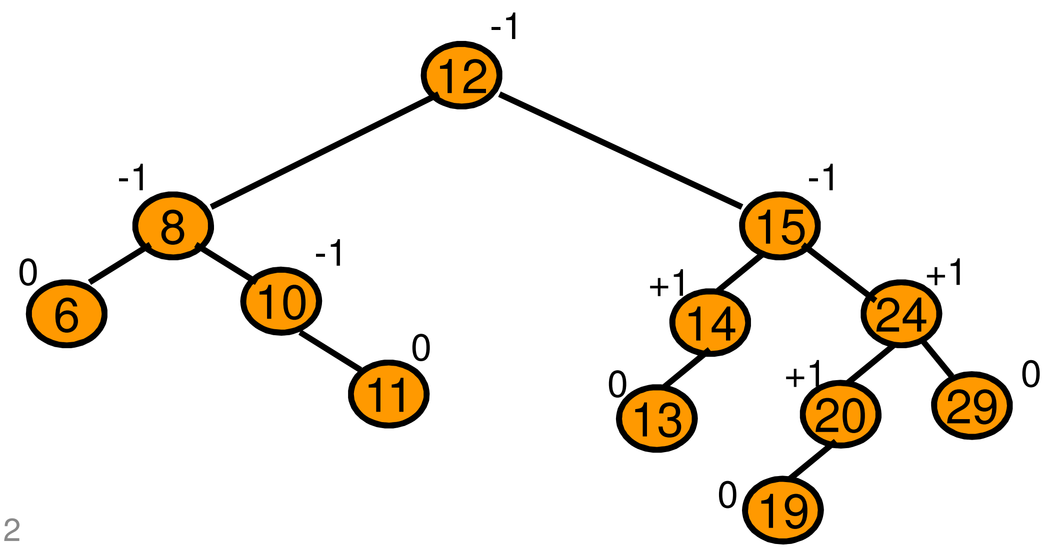
תיקון לאחר הכנסה - גלגולים

<u>הגדרה</u>:

<u>גורם האיזון</u> (balance factor) של צומת הוא ההפרש בין גובה תת העץ השמאלי לתת העץ הימני של הצומת.

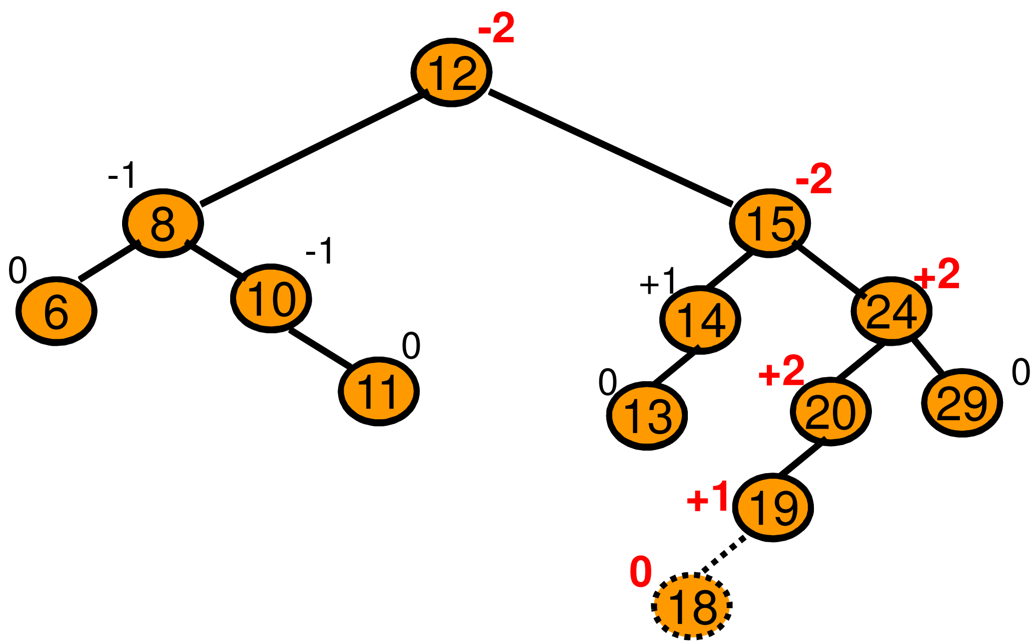
$$\mathbf{BF}[v] \;=\; h(\mathbf{left}[v]) \;-\; h(\mathbf{right}[v])$$

בעץ AVL תקין מתקיים לכל צומת v: $\quad |\,\mathbf{BF}[v]\,| \le 1$



תיקון לאחר הכנסה - אבחנות

נניח שהכנסנו את 18 לעץ.



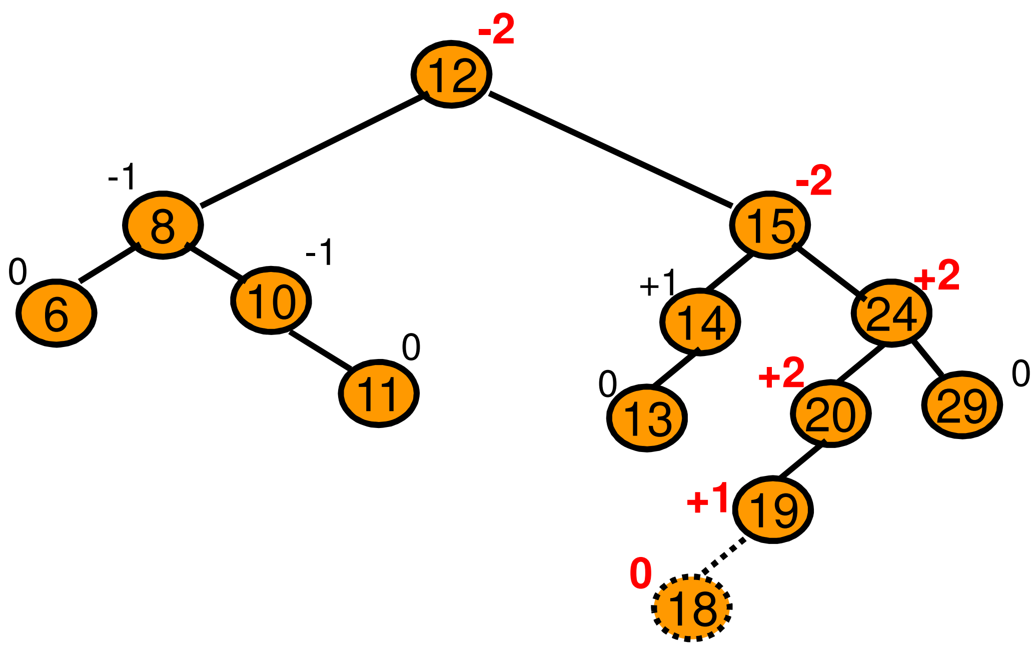
<u>שאלה</u>: אילו ערכי BF אפשריים כעת?

<u>אבחנה מס' 1</u>

לאחר הכנסה, גורם האיזון לא יכול להיות גדול מ- 2 בערכו המוחלט, כי הוא משתנה ב- 1 לכל היותר.

<u>תיקון לאחר הכנסה - אבחנות</u>

<u>שאלה</u>: מיהם הצמתים בהם ייתכן שינוי בגורם האיזון?



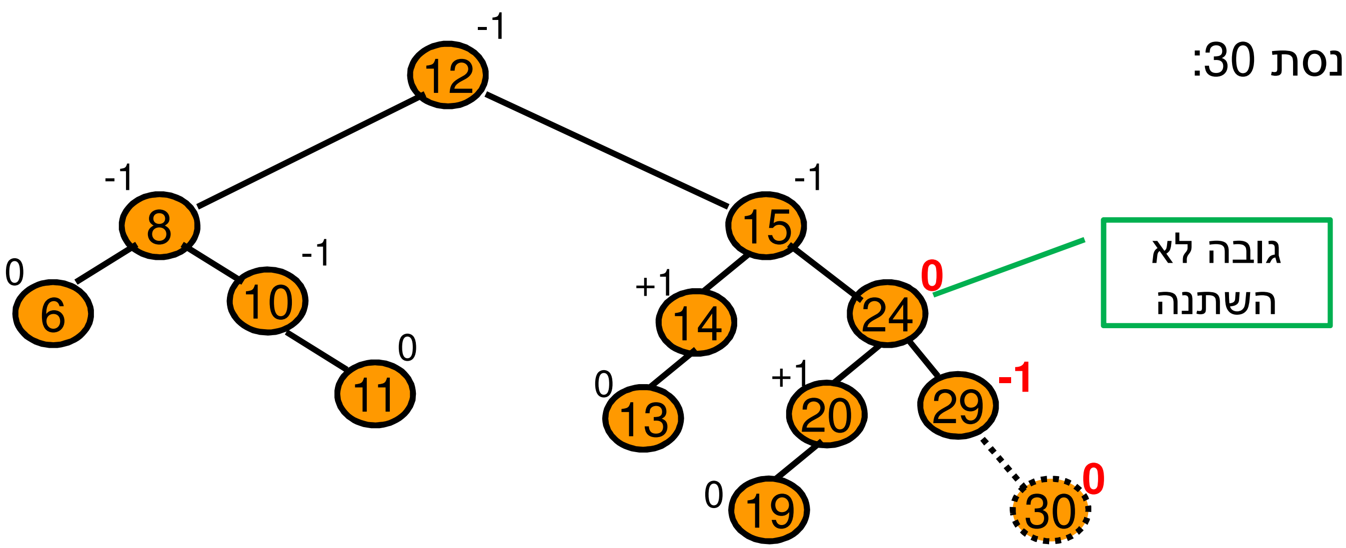
<u>אבחנה מס' 2</u>

הצמתים היחידים שאולי הופר בהם האיזון הם הצמתים לאורך מסלול ההכנסה.

<u>תיקון לאחר הכנסה - אבחנות</u>

<u>שאלה</u>: כיצד ייתכן שלא חל שינוי <u>בכל</u> הצמתים במסלול ההכנסה?

למשל בהכנסת 30:



<u>אבחנה מס' 3</u>

אם קיים במסלול הנ"ל צומת <u>שגובהו לא השתנה</u> בעקבות ההכנסה, אז גורמי האיזון בצמתים שמעליו במסלול לא השתנו.

תיקון לאחר הכנסה - אבחנות

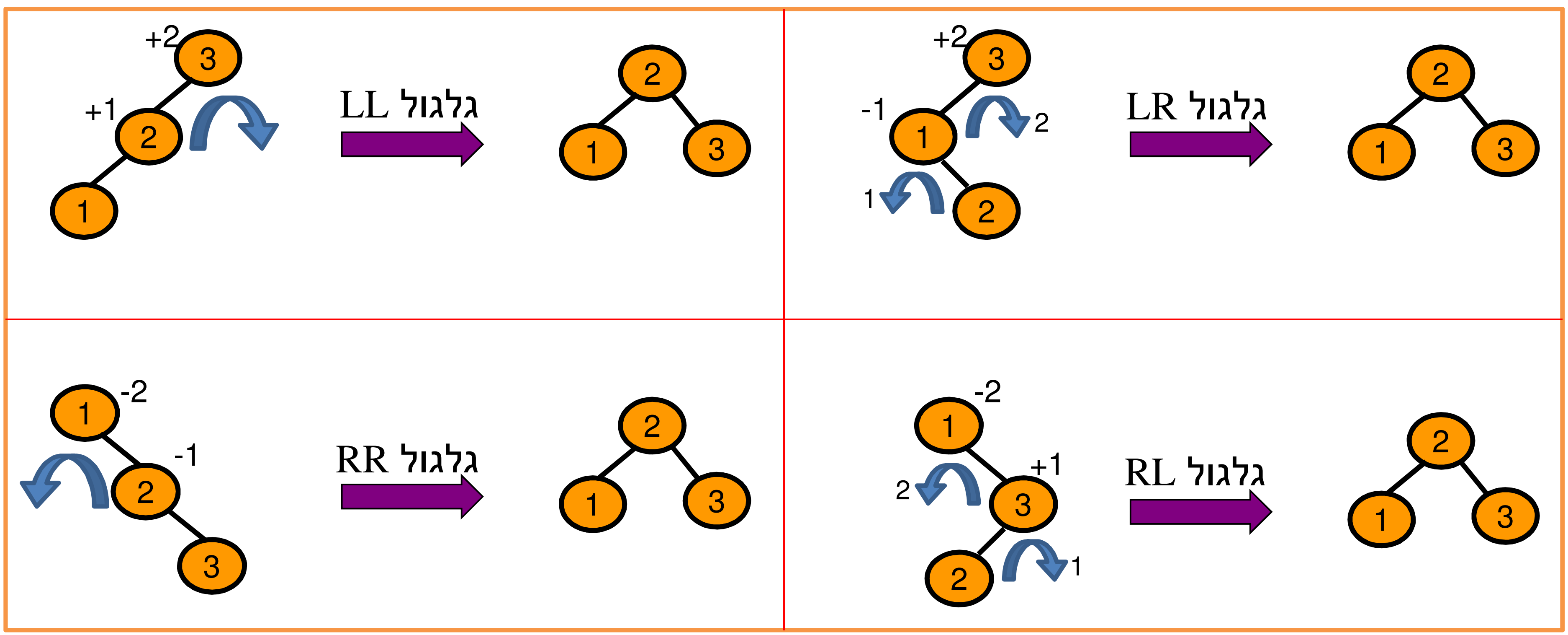
לאור 3 האבחנות, להלן קווים ראשונים לדמותו של אלגוריתם התיקון לאחר הכנסה לעץ AVL:

- אחרי הוספת צומת לעץ AVL, נעבור על הצמתים החל בצומת החדש כלפי מעלה לכיוון השורש.
- אם גורם האיזון בצומת כלשהו לא תקין (± 2) – נבצע גלגול מתאים (כפי שנראה מייד).
- נסיים כאשר נגיע לצומת שגובהו זהה לגובהו לפני ההכנסה, או כאשר נסיים לטפל בשורש.

תיקון לאחר הכנסה - גלגולים

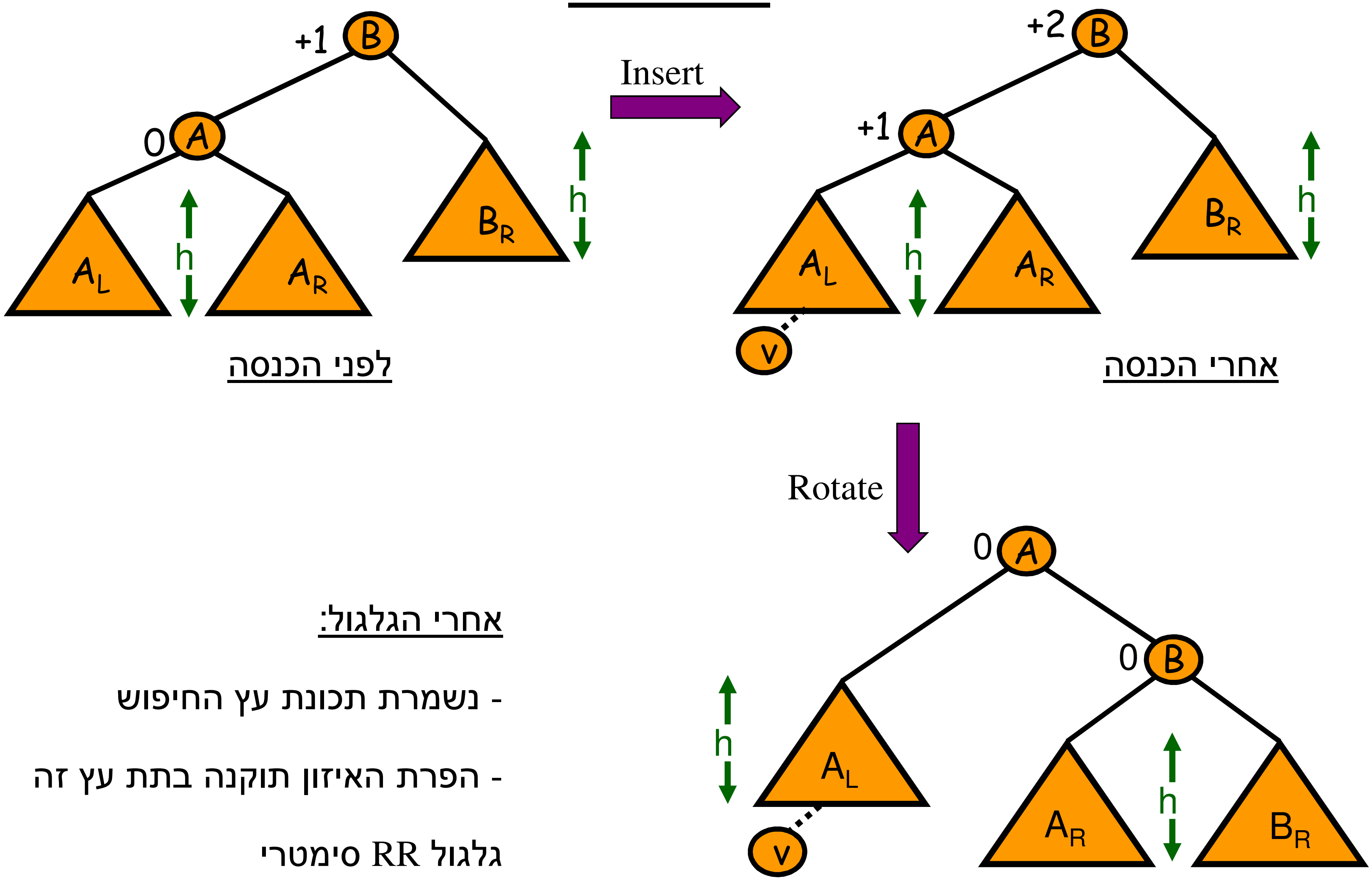
ישנם 4 סוגי גלגולים, המתאימים ל- 4 המצבים אפשריים של הפרת איזון*:

BF[v]	BF של הבן השמאלי	BF של הבן הימני	הגלגול המתאים
+2	+1		LL
-2		-1	RR
+2	-1		LR
-2		+1	RL



* 4 המצבים האלו מכסים את כל המקרים האפשריים.

גלגול LL



לפני הכנסה

אחרי הכנסה

אחרי גלגול

אחרי הגלגול:

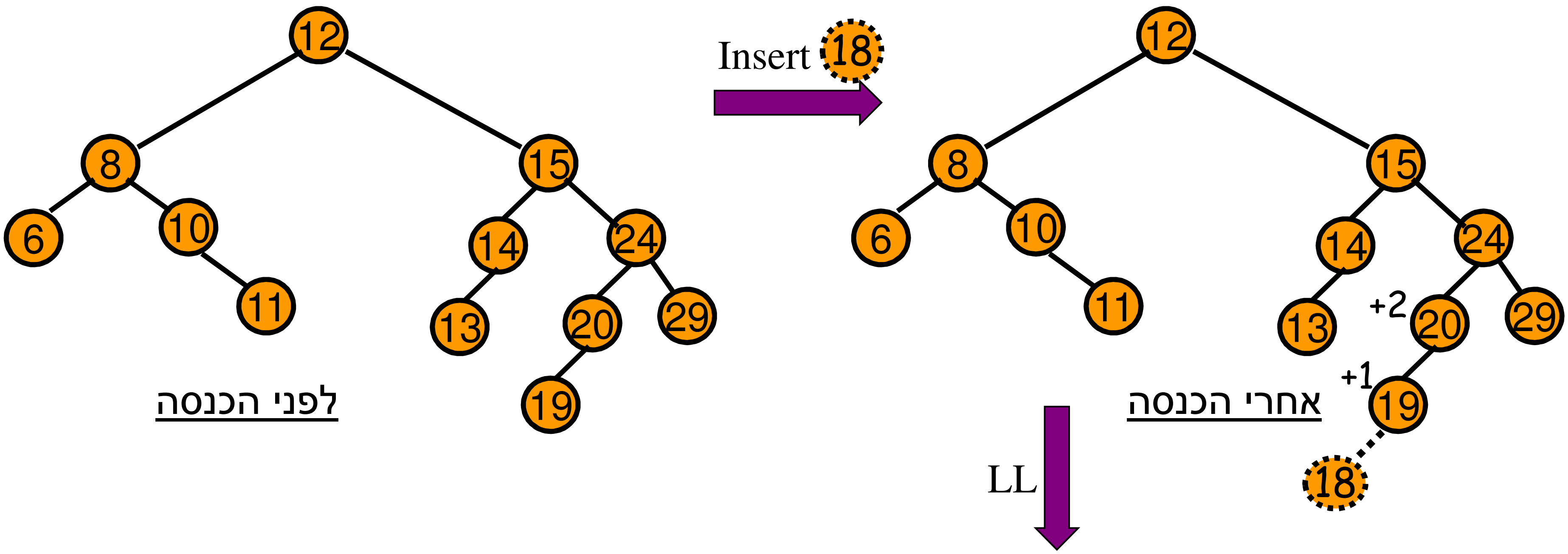
- נשמרת תכונת עץ החיפוש

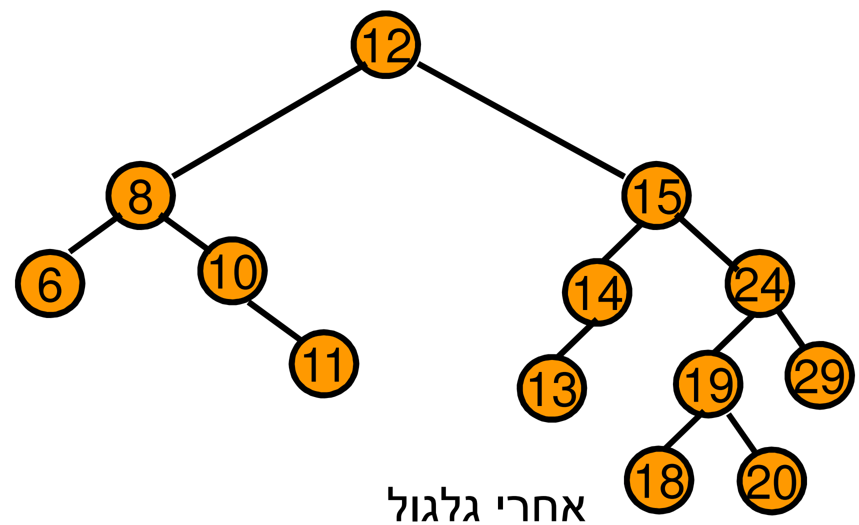
- הפרת האיזון תוקנה בתת עץ זה

גלגול RR סימטרי

גלגול LL

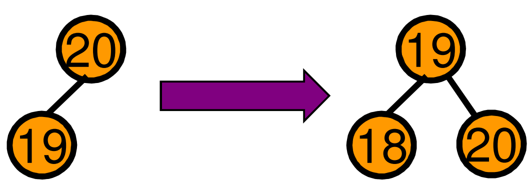
הדגמה



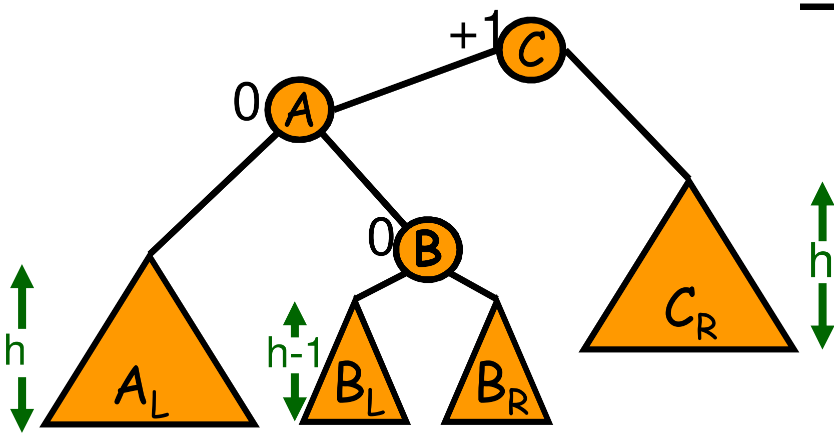


אחרי גלגול אחד נסיים, כי גובה

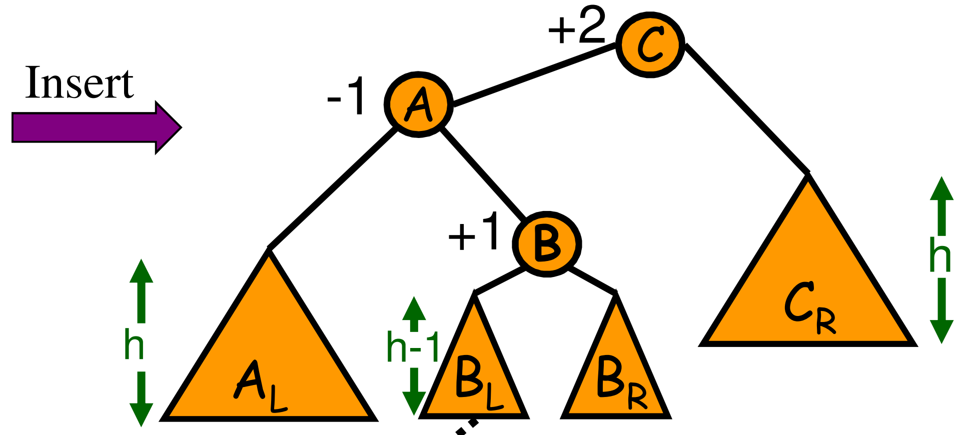
תת העץ לא השתנה



גלגול LR

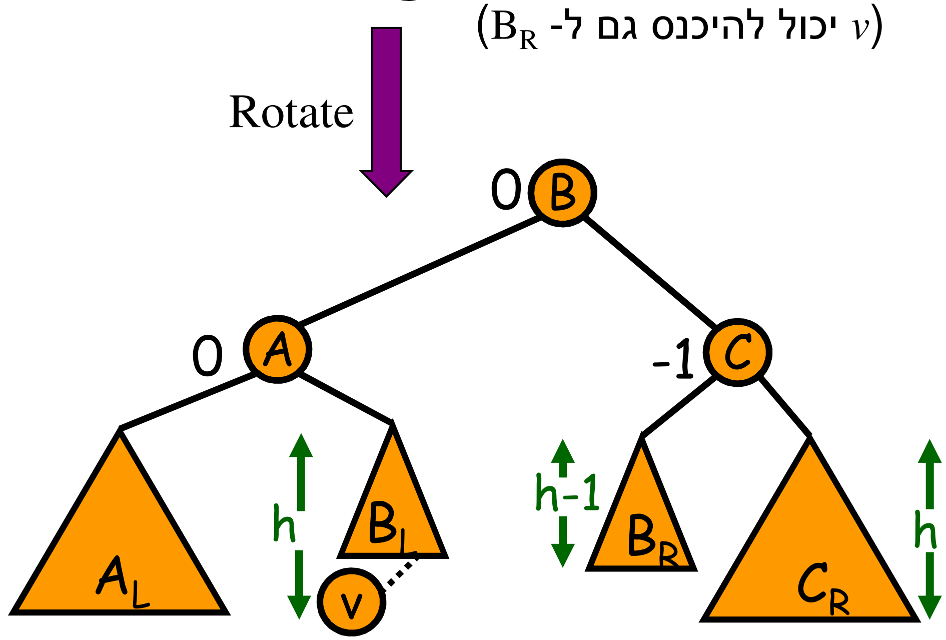


לפני הכנסה



אחרי הכנסה

(*v* יכול להיכנס גם ל- B_R)



אחרי גלגול

אחרי הגלגול:

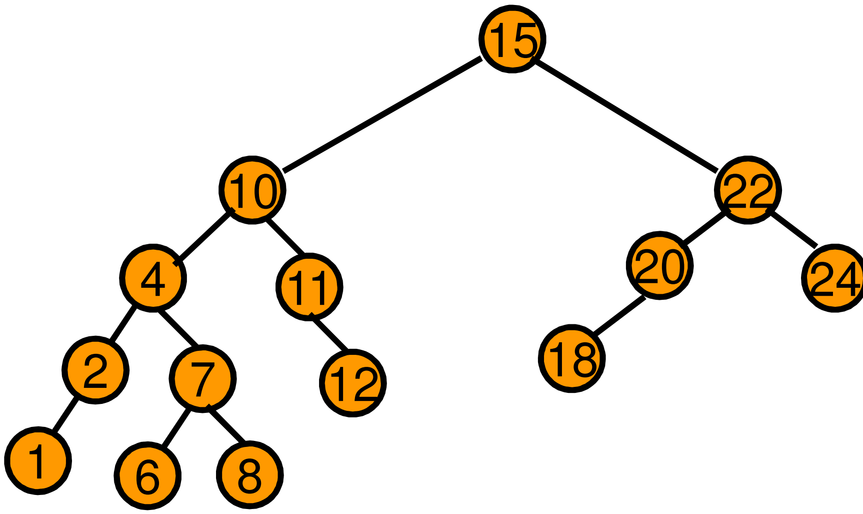
- נשמרת תכונת עץ החיפוש

- הפרת האיזון תוקנה בתת עץ זה

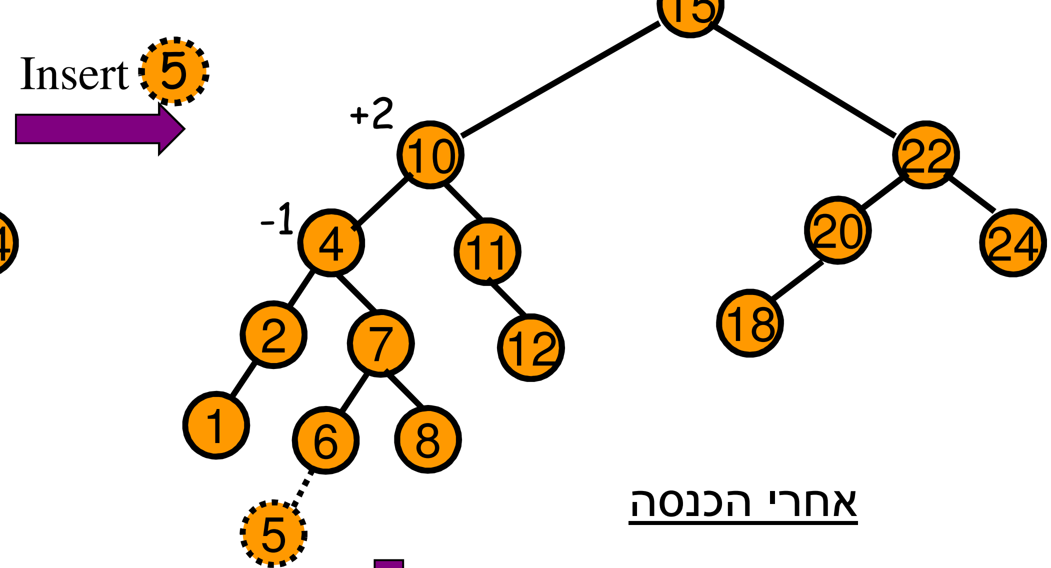
גלגול RL סימטרי

גלגול LR

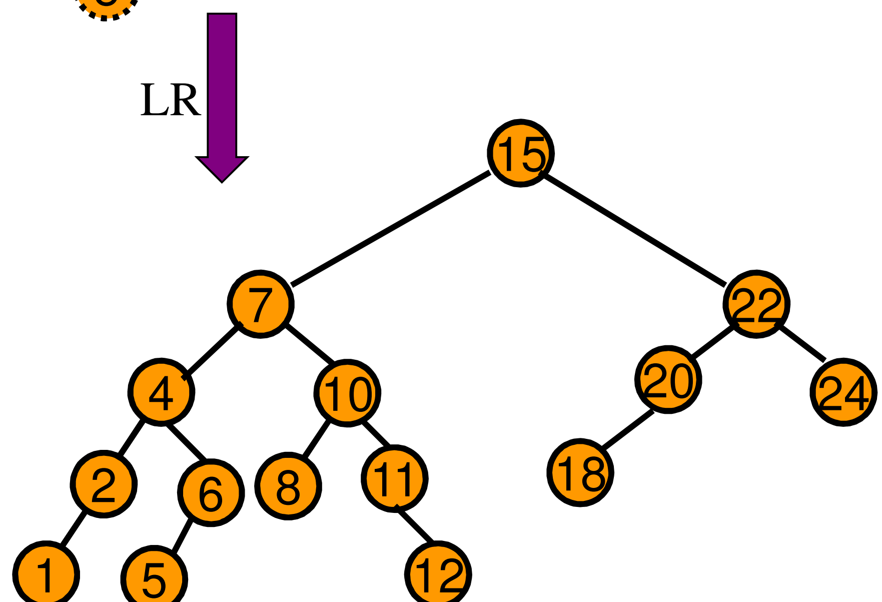
הדגמה



לפני הכנסה

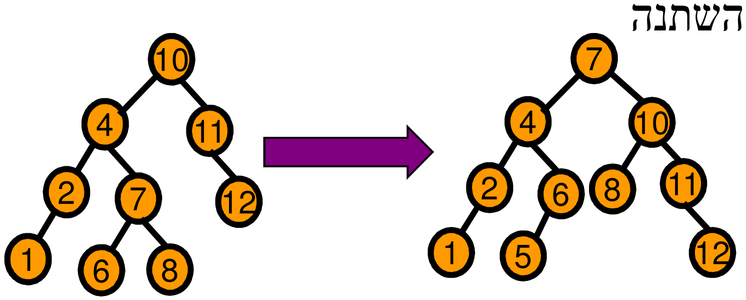


אחרי הכנסה



אחרי גלגול

אחרי גלגול אחד נסיים, כי גובה תת העץ לא השתנה

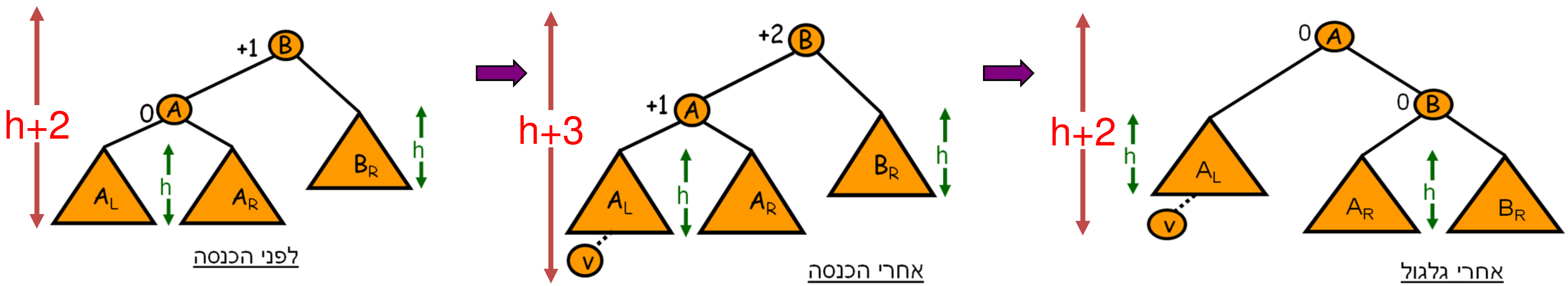


<u>תיקון לאחר הכנסה</u>

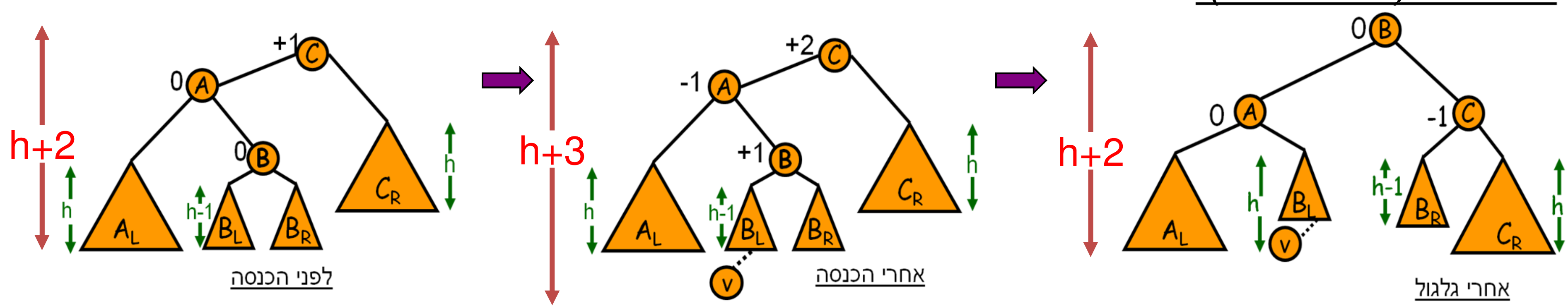
<u>טענה</u>: לאחר הכנסה נדרש <u>לכל היותר גלגול אחד</u> לתיקון העץ.

<u>הוכחה</u>: גלגול בעקבות הכנסה תמיד מחזיר את גובה תת העץ לגובהו המקורי שלפני ההכנסה.

<u>גלגול LL (סימטרי RR):</u>



<u>גלגול LR (סימטרי RL):</u>



אלגוריתם ההכנסה

AVL-Insert(T, z)

1. הכנס את z כרגיל (כמו בעץ חיפוש בינארי)
2. כל עוד $z \neq \text{Nil}$ בצע:
   2.1. חשב את $BF[z]$
   2.2. אם $|BF[z]| < 2$ וגובהו של z לא השתנה – סיים.
   2.3. אחרת אם $|BF[z]| < 2$ וגובהו של z השתנה, עבור לאיטרציה הבאה עם אביו של z.
   2.4. אחרת (כאן $|BF[z]| = 2$) בצע גלגול מתאים וסיים.

סיבוכיות זמן

הכנסה כרגיל לעץ חיפוש בינארי	$\Theta(h)$
מציאת מקום הגלגול (אם בכלל)	$\Theta(h)$
ביצוע הגלגול (אם מתבצע)	$\Theta(1)$
	$\Theta(h) = \Theta(\log n)$

תיקון לאחר הוצאה

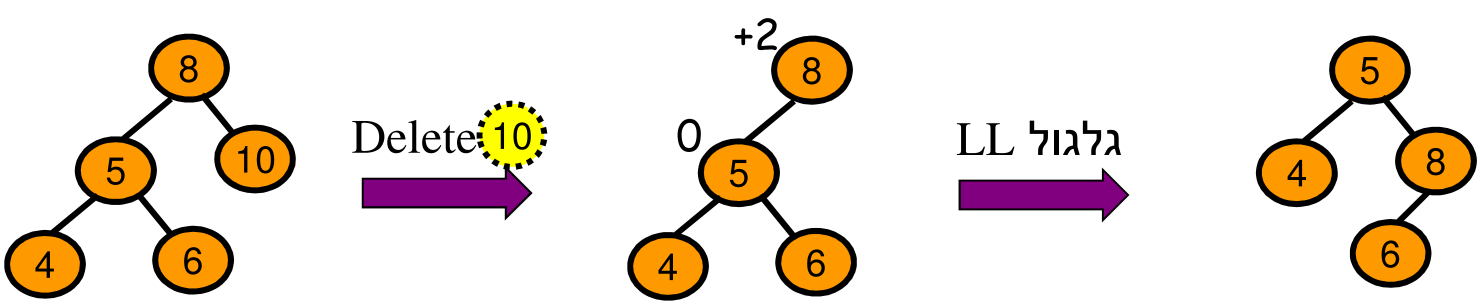
גם לאחר הוצאת צומת תיתכן הפרה של גורמי איזון.

אותם 4 סוגי גלגולים משמשים גם פה לתיקון ההפרה.

BF[v]	BF של הבן השמאלי	BF של הבן הימני	הגלגול המתאים
+2	+1 או 0		LL
-2		-1 או 0	RR
+2	-1		LR
-2		+1	RL

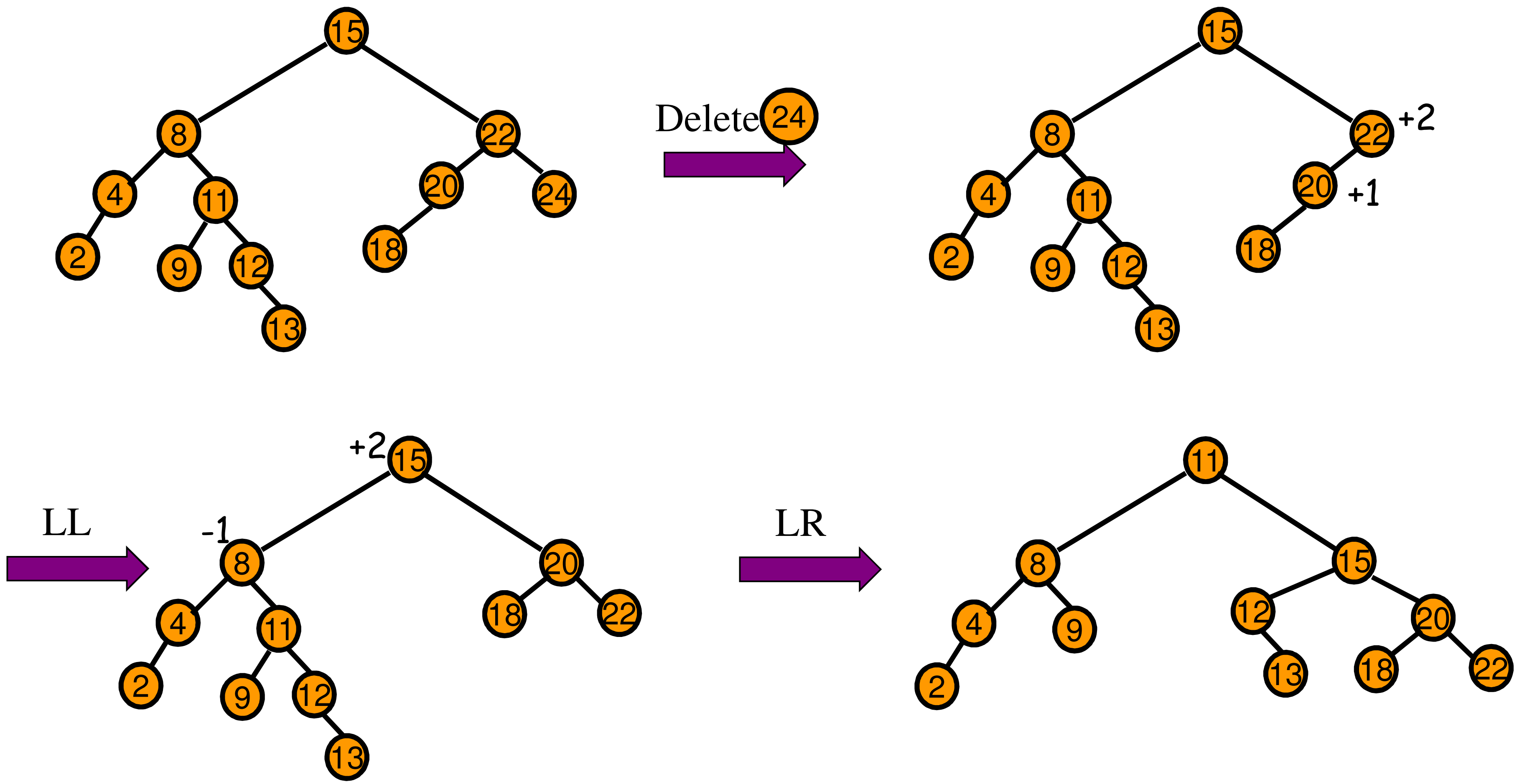
<u>שימו לב להבדל בין שתי הטבלאות.</u>

בהוצאה ייתכנו מצבים שאינם אפשריים בהכנסה. למשל:



תיקון לאחר הוצאה

<u>בהוצאה, ייתכן יותר מגלגול אחד</u> (ייתכן אף שיתבצע גלגול אחד בכל רמה של העץ).



אלגוריתם ההוצאה

AVL-Delete(T, z)

1. הוצא את z כרגיל (כמו בעץ חיפוש בינארי). יהי y אביו של הצומת שנמחק בפועל.
2. כל עוד $y \neq \mathrm{Nil}$ בצע:
   - 2.1. חשב את $BF[y]$.
   - 2.2. אם $|BF[y]| < 2$ וגובהו של y לא השתנה – סיים.
   - 2.3. אחרת אם $|BF[y]| < 2$ וגובהו של y השתנה, עבור לאיטרציה הבאה עם אביו של y.
   - 2.4. אחרת (כאן $|BF[y]| = 2$) בצע גלגול מתאים ~~וסיים~~ ועבור לאיטרציה הבאה עם אביו של y.

סיבוכיות זמן

הוצאה כרגיל מעץ חיפוש בינארי $\Theta(h)$

לכל היותר גלגול אחד בכל רמה $\Theta(h)$

$\Theta(h) = \Theta(\log n)$

אנימציה

http://people.ksp.sk/~kuko/bak/index.html

שאלות חזרה

1. בקרו בקישור הבא, המציע סימולציה מצוינת של עצי AVL:
http://people.ksp.sk/~kuko/bak/index.html
בנו באמצעות הסימולציה את העץ המופיע בשקף 12.
בצעו כמה פעולות הכנסה ומחיקה על העץ - תחילה בעצמכם על נייר ואח"כ בעזרת הסימולציה לבדיקת תשובתכם.

2. הסבירו: כאשר גורם איזון של צומת משתנה בעקבות הכנסה מ- ±1 ל- 0 אז גובהו לא משתנה. האם ניתן לטעון זאת גם בעקבות הוצאה?

3. כפי שראינו, בעקבות הוצאת צומת מעץ AVL יכול להתקבל מצב (זמני, לפני התיקון), שבו יש צומת עם גורם איזון $+2$, ולו בן שמאלי עם גורם איזון 0.
הסבירו מדוע מצב כזה לא ייתכן בעקבות הכנסת צומת לעץ AVL.

4. רוצים לבנות עץ AVL עם השלמים בין 1 ל- 7. הציעו סדר הכנסה כזה, שלא יצריך ביצוע אף גלגול. הציעו סדר הכנסה שיצריך כמות מקסימלית של גלגולים.

5. מה צורתו של העץ המתקבל מהכנסת המספרים 1 עד 7 לפי הסדר לעץ AVL?

תשובות לשאלות חזרה

2. אם גורם האיזון של צומת v משתנה בעקבות הכנסה מ- $+1$ ל- 0, אז הצומת החדש הוכנס לתת-העץ הימני, וכעת גובהו שווה לגובהו של תת-העץ השמאלי. לכן הגובה של v לא משתנה. באופן סימטרי, הטענה נכונה גם עבור המצב השני.
בהוצאה המצב הפוך – דווקא שינוי של גורם האיזון מ- 0 ל- ±1 מעיד על אי-שינוי בגובה.

3. אם בעקבות הכנסה לצומת v יש גורם איזון $+2$, הדבר מעיד על כך שההכנסה בוצעה בתת-העץ השמאלי שלו. אם לבנו השמאלי של v גורם איזון 0, אז לפני ההכנסה הוא היה $+1$ או -1 (אם היה 0 אז הוא לא השתנה בעקבות ההכנסה, ולכן גם גורם האיזון של v גם לא היה אמור להשתנות).
לפי שאלה 2, גובהו של הבן השמאלי לא השתנה בעקבות ההכנסה, ולכן גם גורם האיזון של v לא היה יכול להשתנות.

4. ללא גלגולים, למשל (משמאל לימין): $4, 2, 6, 1, 3, 5, 7$
מקסימום גלגולים – סדר ממוין או ממוין הפוך (4 גלגולים).

5. עץ שלם. נסו להריץ את פעולות ההכנסה בעצמכם, ואח"כ בידקו בעזרת אחת האנימציות.

תרגילים

<u>תרגילים</u>

1. הציעו אלגוריתם אופטימלי, מבחינת סיבוכיות זמן, לבניית עץ AVL מרשימה נתונה של n איברים. הוכיחו כי האלגוריתם שלכם אופטימלי.

2. תארו אלגוריתם ליניארי לבניית עץ AVL ממערך <u>ממוין</u> A בעל n איברים.

3. נתונים שני עצי AVL, כל אחד בעל n צמתים.

   תארו אלגוריתם יעיל ככל האפשר למיזוג שני העצים לעץ AVL אחד.

4. נתונים n איברים כלשהם. לכל איבר יש מפתח, וידוע שישנם רק $\lfloor \log n \rfloor$ מפתחות שונים.

   הציעו אלגוריתם למיון האיברים ע"פ המפתחות שלהם, שרץ בזמן $\Theta(n \log\log n)$.

<u>פתרון 1</u>

נבחן תחילה את הפתרון הישיר: להכניס את האיברים בזה אחר זה לעץ AVL ריק בהתחלה.

סיבוכיות זמן:

$$\Theta(\sum_{i=1}^{n} \log i) = \Theta(\log n!) = \Theta(n \log n)$$

האם קיים פתרון טוב יותר?

נניח בשלילה שקיים פתרון שסיבוכיותו טובה יותר - $o(n\log n)$.

אז ניתן גם למיין בזמן $o(n\log n)$, ע"י בניית AVL כנ"ל ואז סיור inorder בזמן ליניארי, וזו סתירה לחסם התחתון למיון.

מכאן שהאלגוריתם הפשוט הנ"ל הוא גם אופטימלי מבחינת סיבוכיות זמן.

פתרון 2 – דרך א'

נקצה צומת לשורש, ובו נאחסן את המפתח במיקום האמצעי של המערך (שהוא גם החציון).

אח"כ נבנה שני תת-עצים באופן רקורסיבי ונחבר אותם לשורש.

```
Sorted-Array-2-AVL(A, p, r)  ► first call p=1, r=n
    if  p > r   return  nil
    Create  root                ► a new node
    mid ← ⌊(p+r)/2⌋
    key[root] ← A[mid]
    left[root] ←  Sorted-Array-2-AVL (A, p, mid-1)
    right[root] ←  Sorted-Array-2-AVL (A, mid+1, r)
    ► if AVL includes pointers to parents, update this info too
    return  root
```

נוסחת הנסיגה המתאימה לזמן הריצה היא $t(n) = 2t(n/2) + \Theta(1)$ ופתרונה האסימפטוטי $\Theta(n)$.

<u>פתרון 2 – דרך ב'</u>

נבנה עץ (כמעט) שלם, שהוא כידוע עץ AVL חוקי (גורם האיזון של כל הצמתים הוא 0 או 1).

נעשה זאת כך:

1. נבנה "שלד" של עץ (כמעט) שלם, ללא ערכים בצמתים, למשל ברקורסיה: נקצה צומת עבור השורש, ואח"כ נבנה רקורסיבית את תת-העץ השמאלי והימני, אם הם צריכים להיות קיימים (כלומר אם האינדקס של השורש שלהם אינו גדול מ- n)

```
Build-Empty-AVL(i, n)          ► first call i=1
    if  i > n   return  nil
    Create  root               ► a new node
    left[root] ← Build-Empty-AVL(2i, n)
    right[root] ← Build-Empty-AVL(2i+1, n)
    ► if AVL includes pointers to parents, update this info too
    return  root
```

2. נעבור על המערך, תוך כדי סיור inorder בעץ. הפעולה Visit תתורגם לכתיבה של האיבר מהמערך לצומת בעץ.

גם כאן סיבוכיות הזמן $\Theta(n)$.

פתרון 3

פתרון נאיבי: עוברים על כל אברי אחד העצים, ומכניסים אותם לעץ השני אחד אחד.

סיבוכיות: מכניסים n פעמים איבר לעץ AVL, שגודלו ההתחלתי n וגודלו הסופי $2n$.

כלומר כל הכנסה מתבצעת ב- $O(\log 2n)$ וב- $\Omega(\log n)$, כלומר ב- $\Theta(\log n)$.

סה"כ $\Theta(n\log n)$.

פתרון יעיל יותר:

- ✓ נבצע סיור inorder על כל אחד מהעצים ונכתוב את אבריו לתוך מערך.
- ✓ נמזג את שני המערכים (שימו לב שהמערכים ממוינים).
- ✓ נבנה מהמערך הממוזג, שגודלו $2n$, עץ AVL בשיטה המתוארת בשאלה קודמת.

סיבוכיות: כל השלבים רצים בזמן ליניארי, כלומר $\Theta(n)$.

פתרון 4

מבנה הנתונים:

נחזיק עץ AVL עם השינויים הבאים: בכל צומת בעץ יהיה מפתח, ורשימה מקושרת של האיברים בעלי מפתח זה.

האלגוריתם:

- נעבור על הקלט, ונכניס את האיברים בזה אחר זה לעץ עם השינויים הבאים:

1. עבור איבר x בעל מפתח k, אם לא קיים עדיין בעץ צומת עם המפתח k, נוסיף צומת כזה ונאתחל מצביע ממנו לרשימה מקושרת ריקה. אחרת אין צורך להוסיף צומת.
2. נכניס את x לראש הרשימה המקושרת של הצומת המתאים.

- לבסוף נסייר בעץ סיור inorder, כאשר בכל צומת נעבור על הרשימה שלו ונוציא לפלט את האיברים שנמצאים בו.

סיבוכיות זמן:

כידוע, הכנסה לעץ AVL מתבצעת בזמן ליניארי בגובהו. במקרה שלפנינו העץ הולך וגדל, אבל גודלו הסופי הוא $\lfloor \log n \rfloor$ צמתים, ולכן כל הכנסה תתבצע בזמן $O(\log\log n)$, וכל ההכנסות בזמן $O(n\log\log n)$. הסיור בסוף כולל המעברים על הרשימות יתבצע בזמן $\Theta(\log n + n) = \Theta(n)$.

סה"כ $O(n\log\log n)$.