אוניברסיטת אריאל בשומרון

מבני נתונים

תרגול 5 – עצים מאוזנים

zvimints@gmail.com – צבי מינץ

היום

- הצגת הבעיה
- עצים מאוזנים
- עץ AVL
- מימוש
- תרגילים AVL
- עץ אדום-שחור
- תרגילים RBT

BBST

Balanced Binary Search Tree

- **עץ בינארי מאוזן** הוא עץ בינארי שכמות הרמות שלו היא **לוגריתמית** ביחס לכמות האובייקטים בעץ
- מספר הרמות שלו הוא $O(logn)$
- מאפשר ביצוע פעולות מהיר יותר

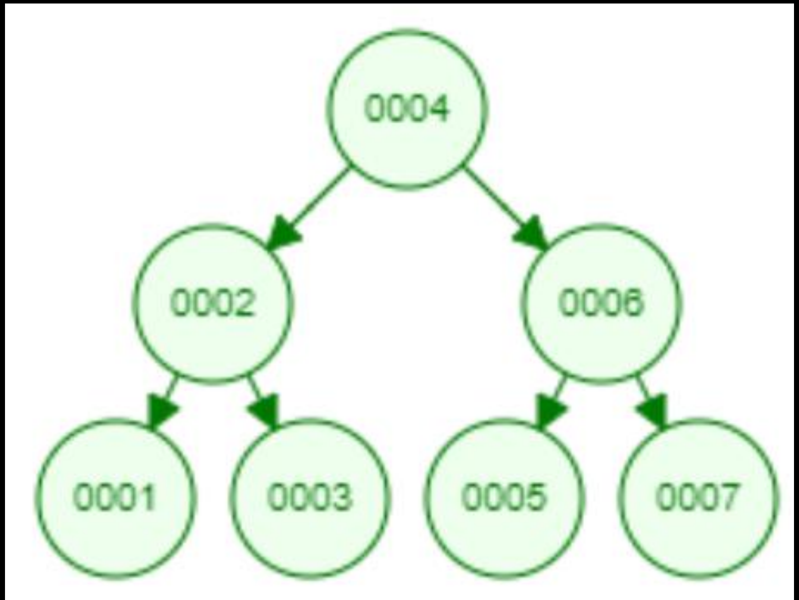
Complexity of BST

- כל הפעולות בעץ חיפוש בינארי לוקחים $\Theta(h)$ במקרה הרע, כאשר h הינו גובה העץ
- גובה העץ **האופטמלי** הינו $\lfloor \log_2 n \rfloor$

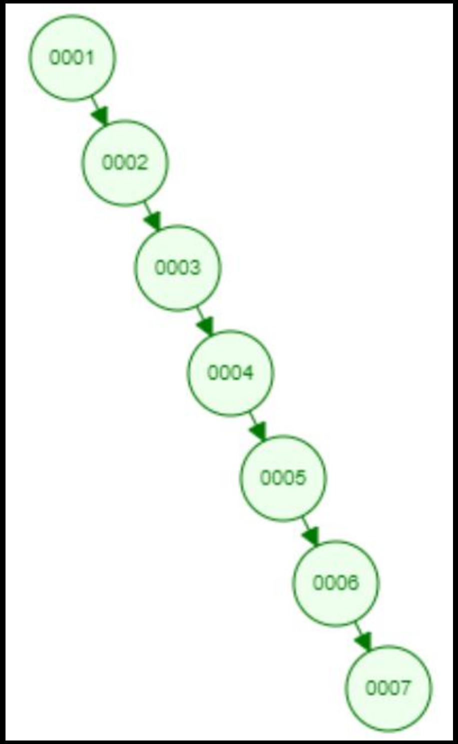
פעולה	ממוצע	מקרה הרע
הכנסה	$O(logn)$	$O(n)$
מחיקה	$O(logn)$	$O(n)$
חיפוש	$O(logn)$	$O(n)$

צבי מינץ – zvimints@gmail.com

Complexity of BST



[4,2,1,3,6,5,7]



[1,2,3,4,5,6,7]

Complexity of BBST

מקרה הרע	ממוצע	פעולה
$O(logn)$	$O(logn)$	הכנסה
$O(logn)$	$O(logn)$	מחיקה
$O(logn)$	$O(logn)$	חיפוש

Tree Rotations

AVL Trees (1962)

- עץ המאוזן **הראשון** בהיסטוריה שהמציאו אותו שני חוקרים רוסים בשנת 1962.
- עץ AVL הוא עץ בינארי מאוזן (**BBST**)

התכונה ששומרת על עץ AVL מאוזן היא ה-**Balanced Factor (BF)**

Height- Balance Property

לכל קודקוד פנימי בעץ, הפרש הגבהים של בניו הוא לכל היותר 1

$$\forall n$$

$$|height(n.right) - height(n.left)| \le 1$$

$$\in \{-1,0,1\}$$

Balance factor

$$BalanceFactor(n) \coloneqq H(n.right) - H(n.left)$$

$$BalanceFactor(n) \in \{-1,0,1\}$$

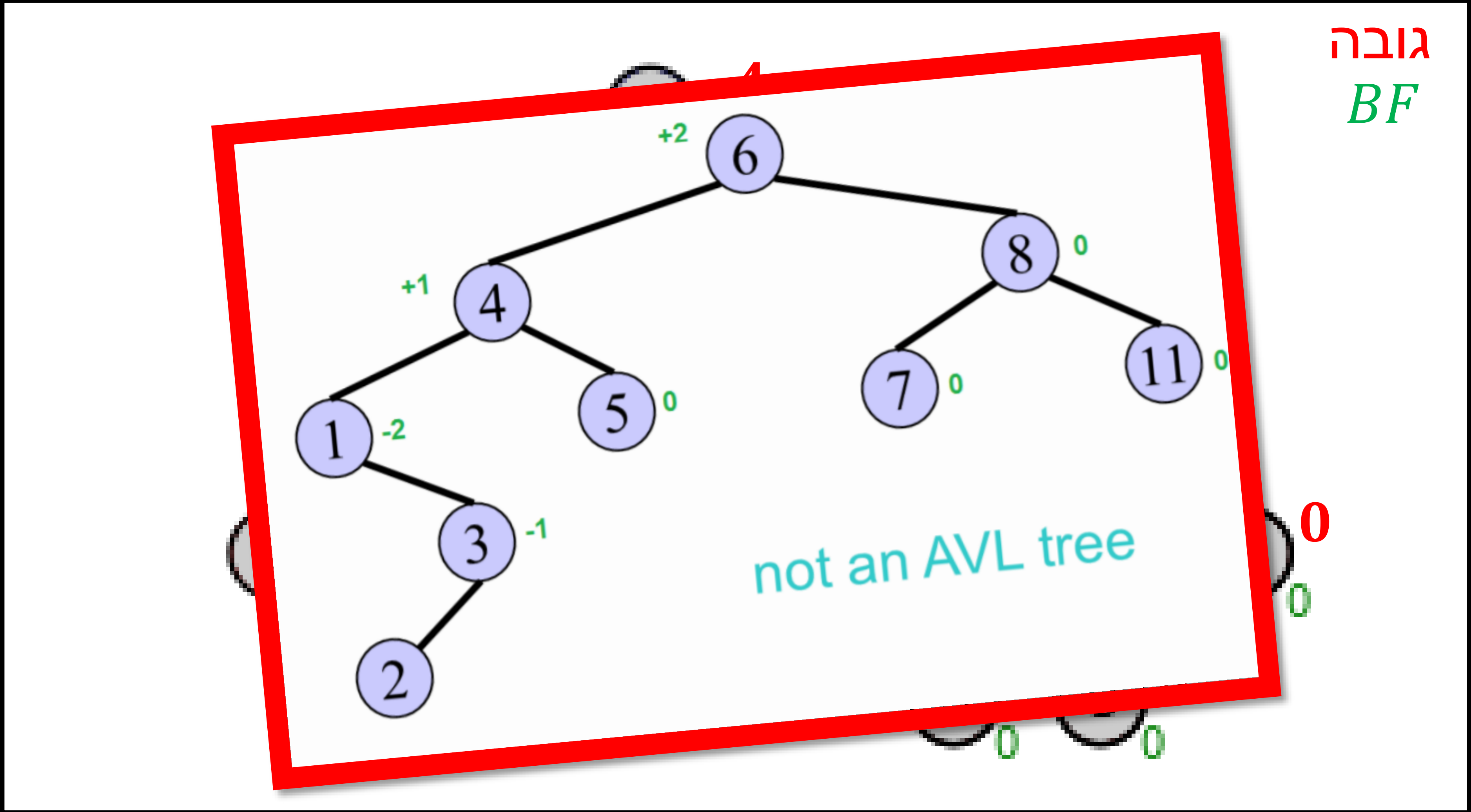
$$BalanceFactor(n) = 0 \quad - \quad balanced$$

$$BalanceFactor(n) = 1 \quad - \quad right\ heavy$$

$$BalanceFactor(n) = -1 \quad - \quad left\ heavy$$

zvimints@gmail.com – צבי מינץ

AVL Trees



AVL Trees (1962)

- עץ המאוזן **הראשון** בהיסטוריה שהמציאו אותו שני חוקרים רוסים בשנת 1962.
- עץ AVL הוא עץ בינארי מאוזן (**BBST**)

התכונה ששומרת על עץ AVL מאוזן היא ה-**Balanced Factor (BF)**

Height- Balance Property

לכל קודקוד פנימי בעץ, הפרש הגבהים של בניו הוא לכל היותר 1

$$if$$

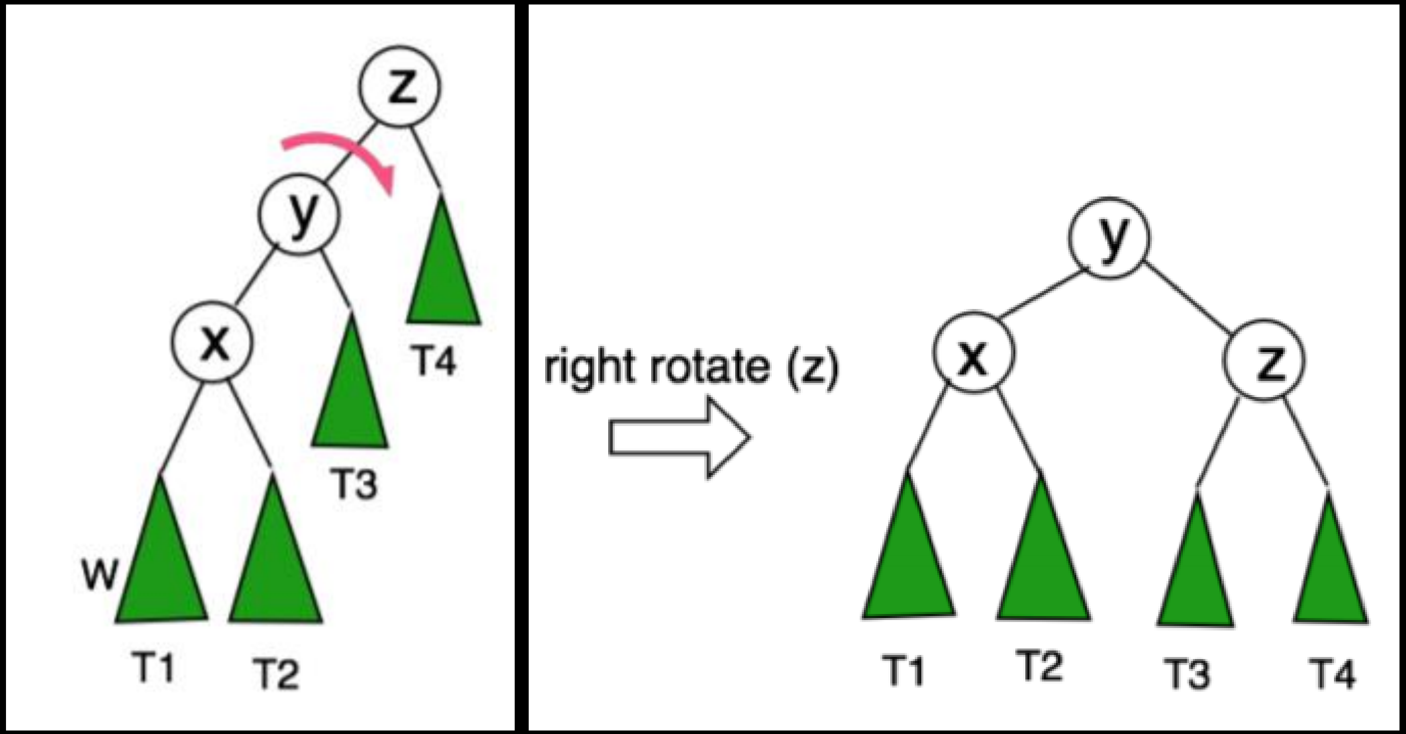
$$height(n.right) - height(n.left) \notin \{-1,0,1\}$$

$$\Rightarrow BF(n) = \pm 2$$

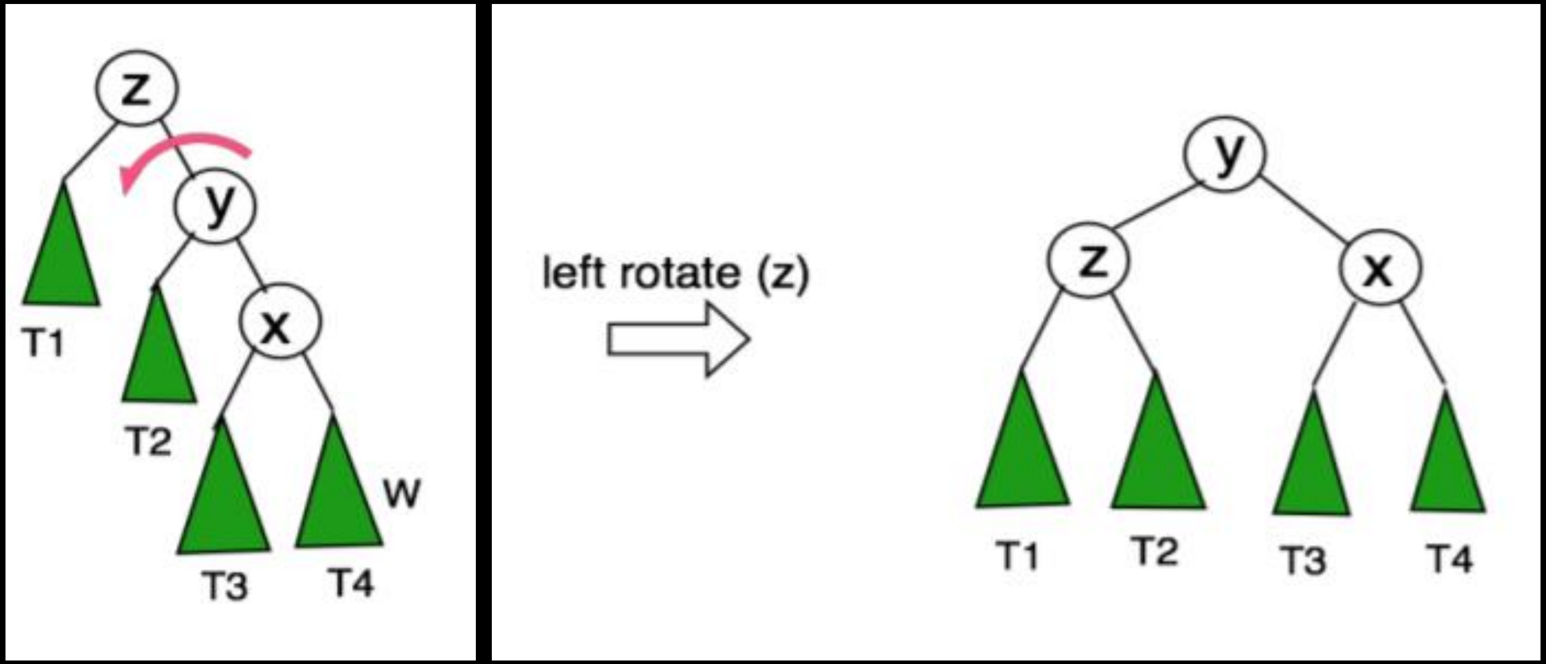
Tree Rotations

zvimints@gmail.com – צבי מינץ

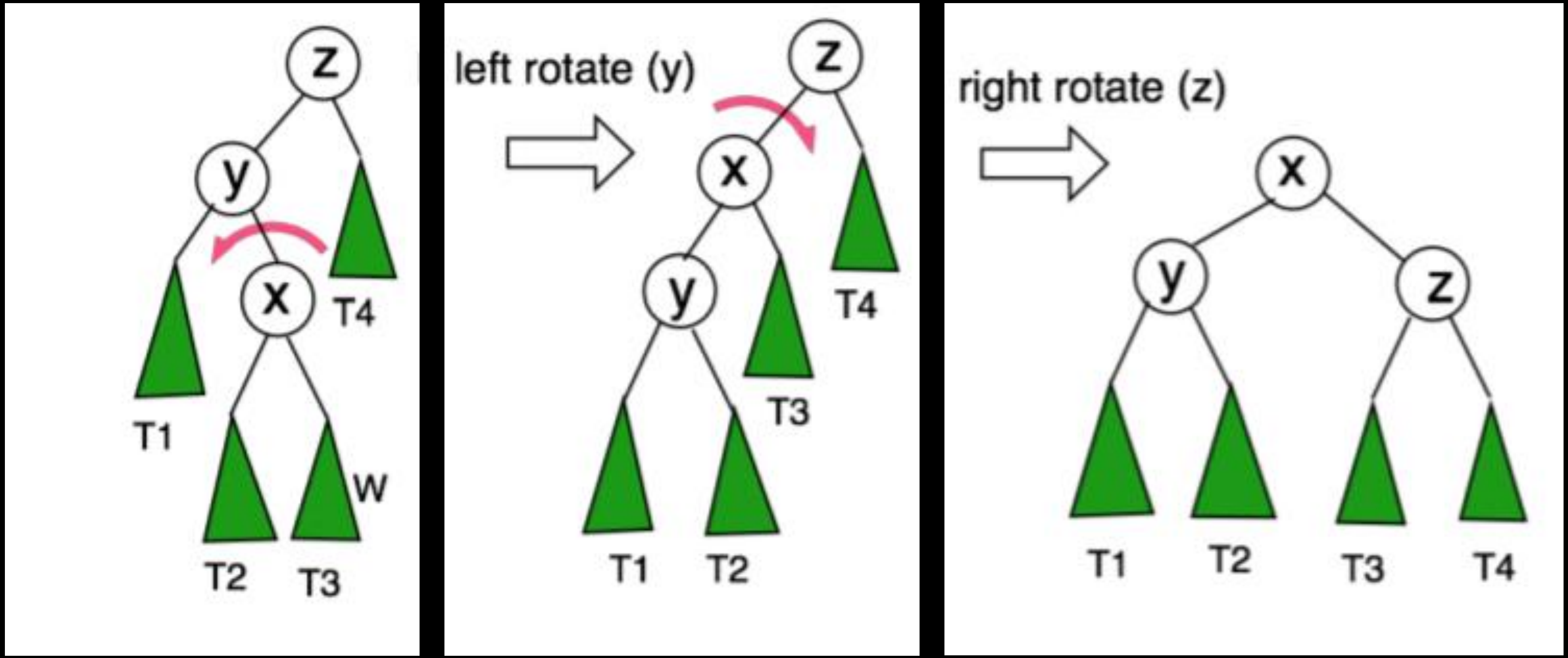
AVL Rotations
LL



AVL Rotations
RR

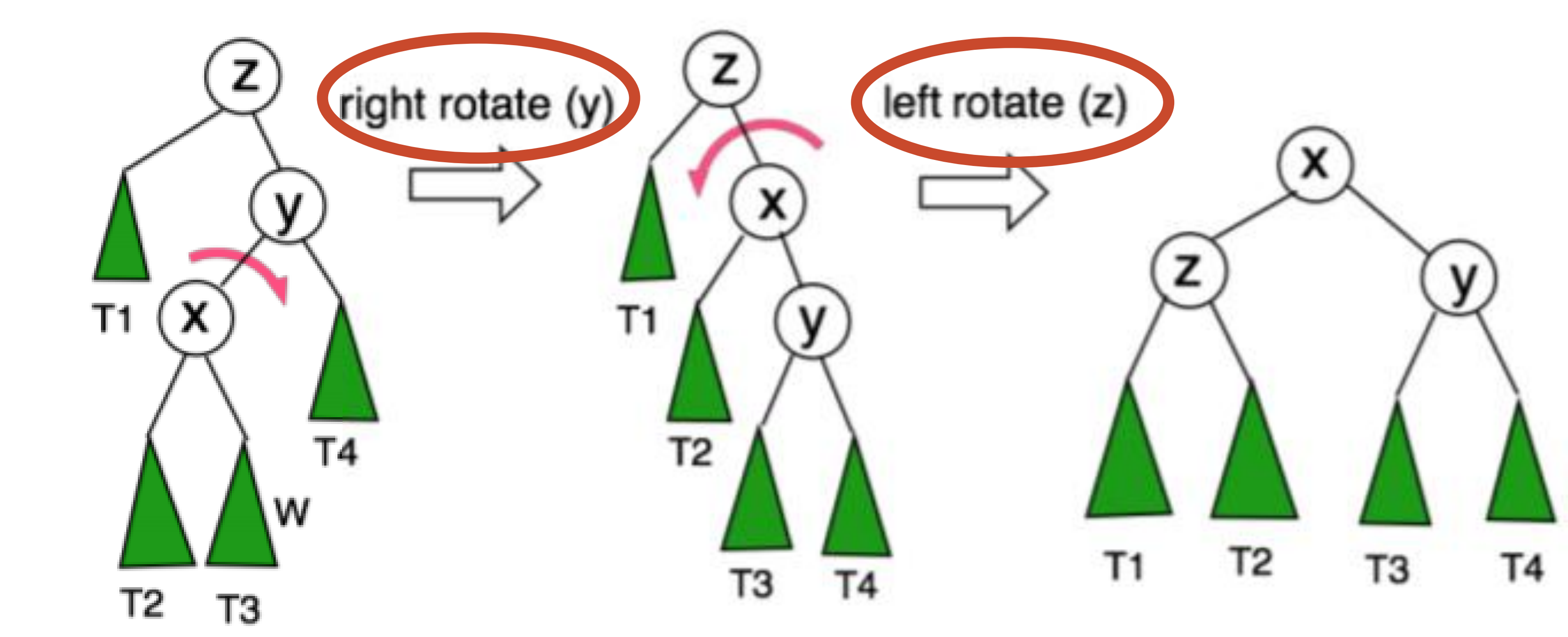


AVL Rotations
LR



zvimints@gmail.com – צבי מינץ

AVL Rotations
RL



AVL Rotations

Insertion

zvimints@gmail.com – צבי מינץ

אבחנות

- הצמתים היחידים שאולי הופר אצלם האיזון הם הצמתים לאורך המסלול הכנסה/הוצאה

- אם עבור צומת x במסלול הנ"ל גובה העץ ששורשו x לא השתנה אז גורמי האיזור בצמתים שמעליו במסלול גם כן לא השתנו

AVL Rotations
Remove

מחיקת איבר - ניתן להשתמש בטכניקה של מחיקת איבר מעץ בינארי רגיל.

אך צריך לבדוק שעכשיו העץ מקיים את חוקי עץ AVL, ולכן בודקים את האבות של כל צומת שנמחקה.

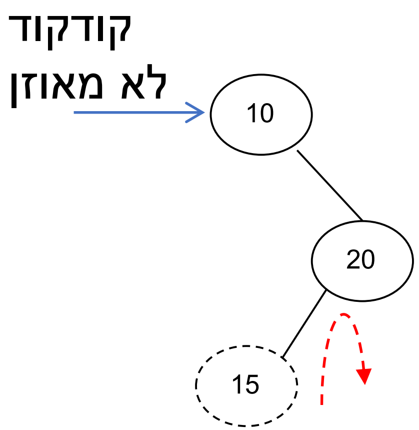
אם מוצאים אב קדמון שגובה ילדיו שונה ב-2, אז צריך לתקן את העץ. התיקון דומה למה שכבר הסברנו.

zvimints@gmail.com – צבי מינץ

שאלה 1

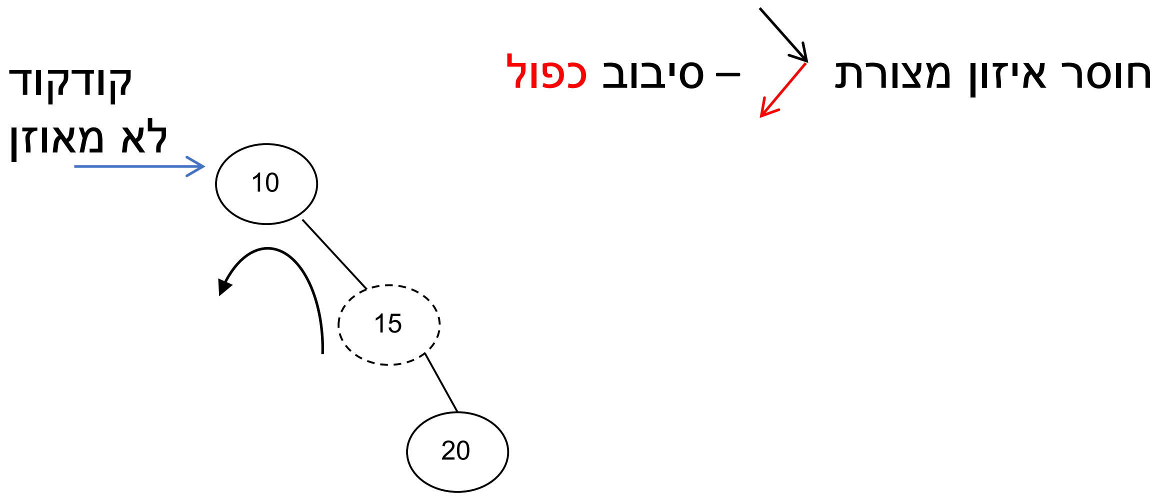
• הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19

חוסר איזון מצורת – סיבוב כפול



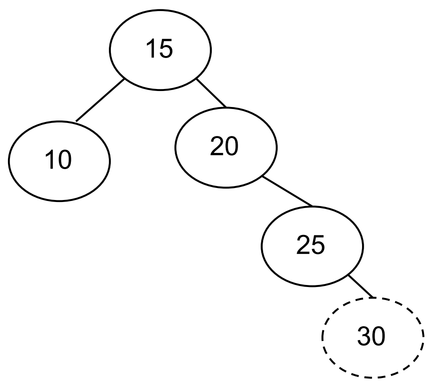
שאלה 1

- הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19



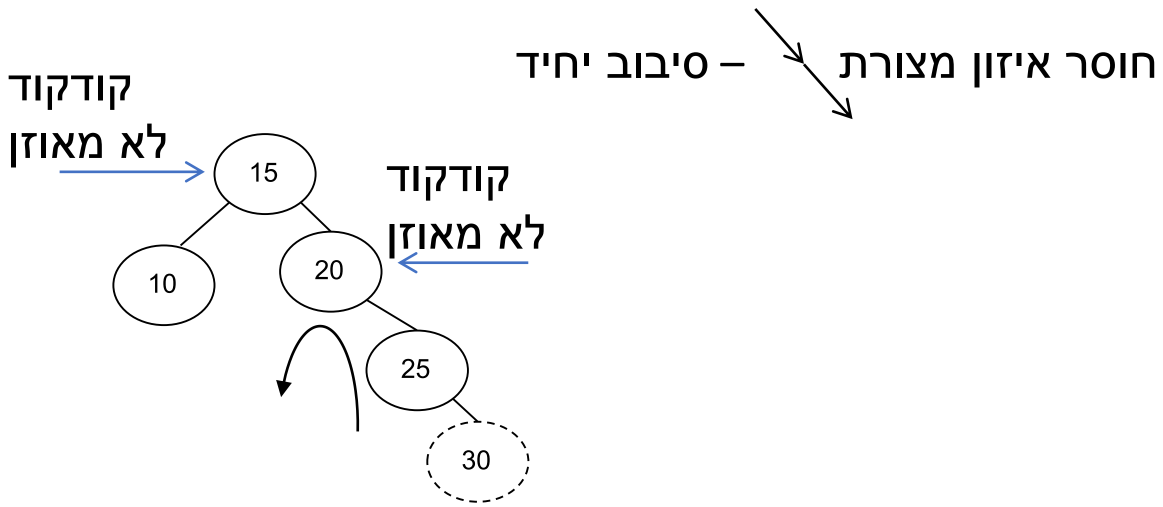
שאלה 1

• הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19



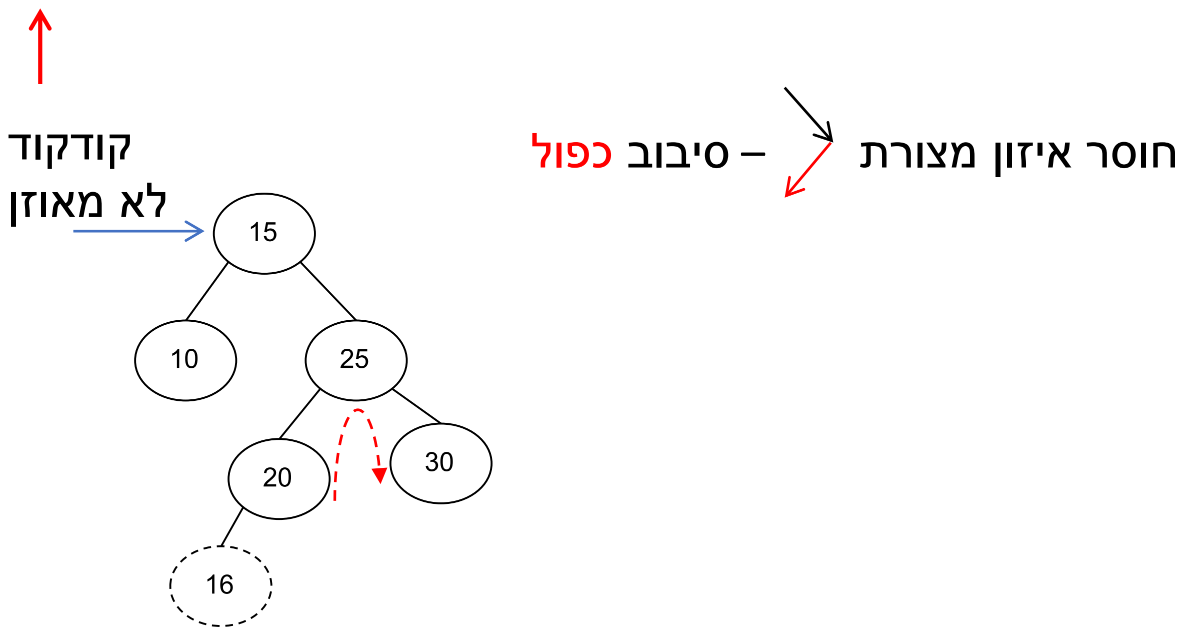
שאלה 1

- הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19



שאלה 1

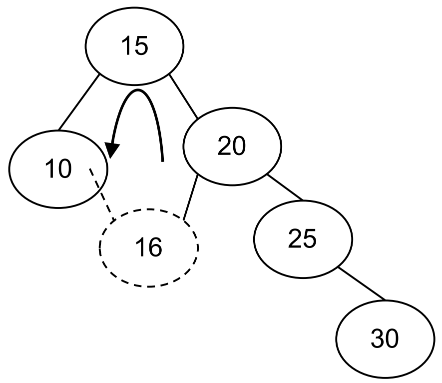
- הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19



שאלה 1

- הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19

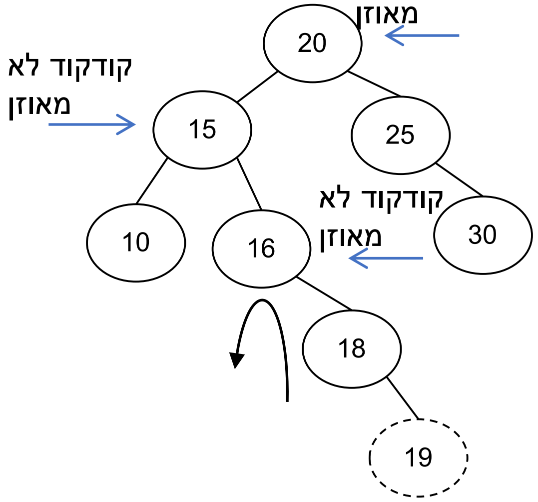
חוסר איזון מצורת ⟩ – סיבוב כפול



שאלה 1

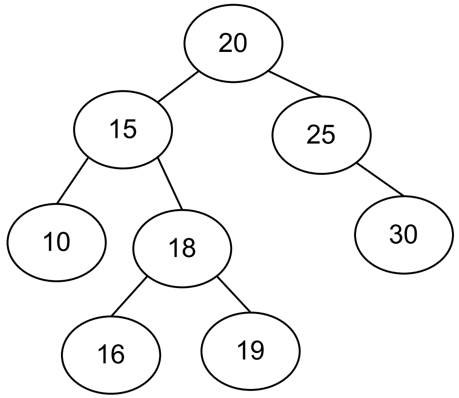
- הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19

חוסר איזון מצורת ↘ – סיבוב יחיד



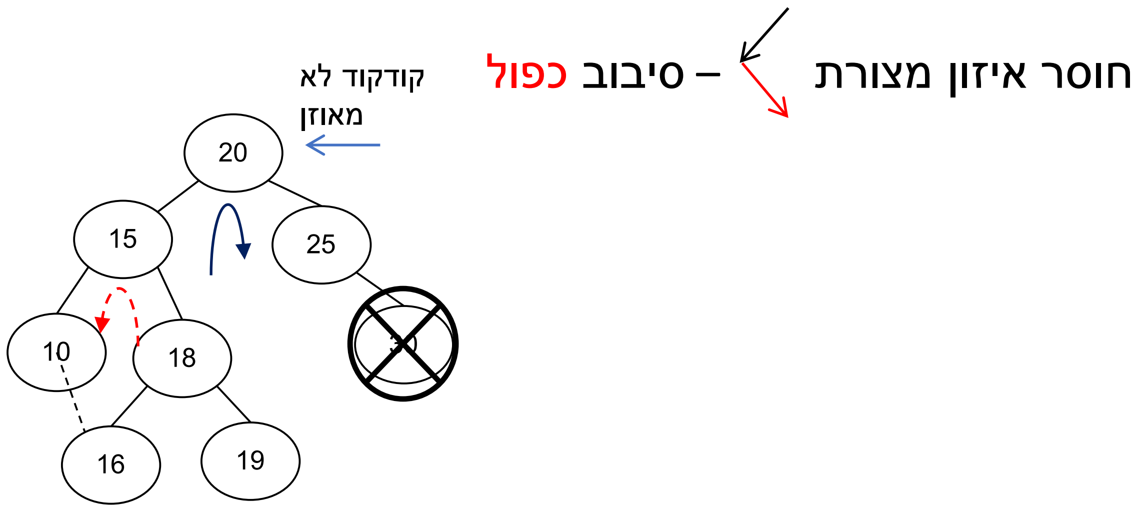
שאלה 1

- הכניסו את האיברים הבאים לפי הסדר לעץ AVL (העץ ריק בהתחלה): 10, 20, 15, 25, 30, 16, 18, 19



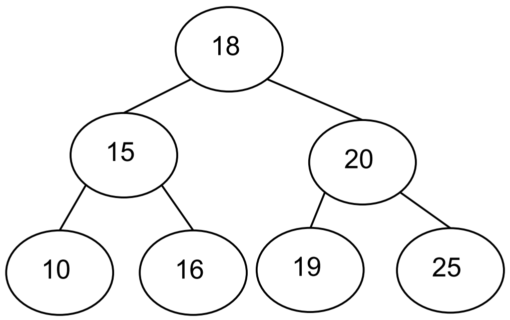
שאלה 1

• כעת הסירו את האיבר 30



שאלה 1

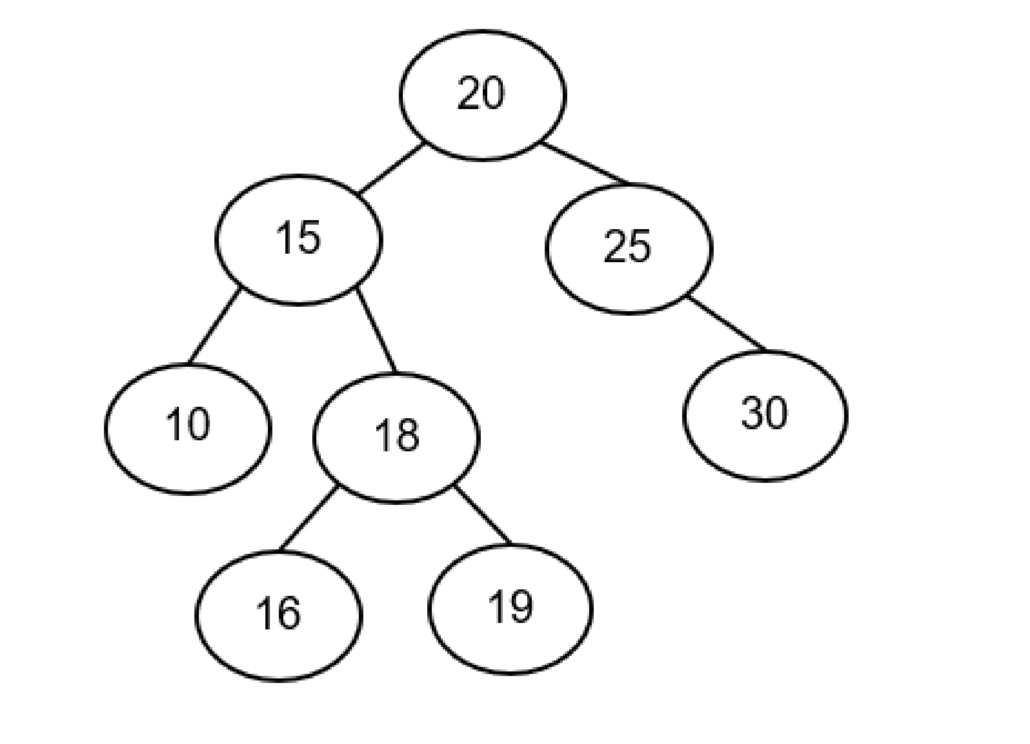
- כעת הסירו את האיבר 30

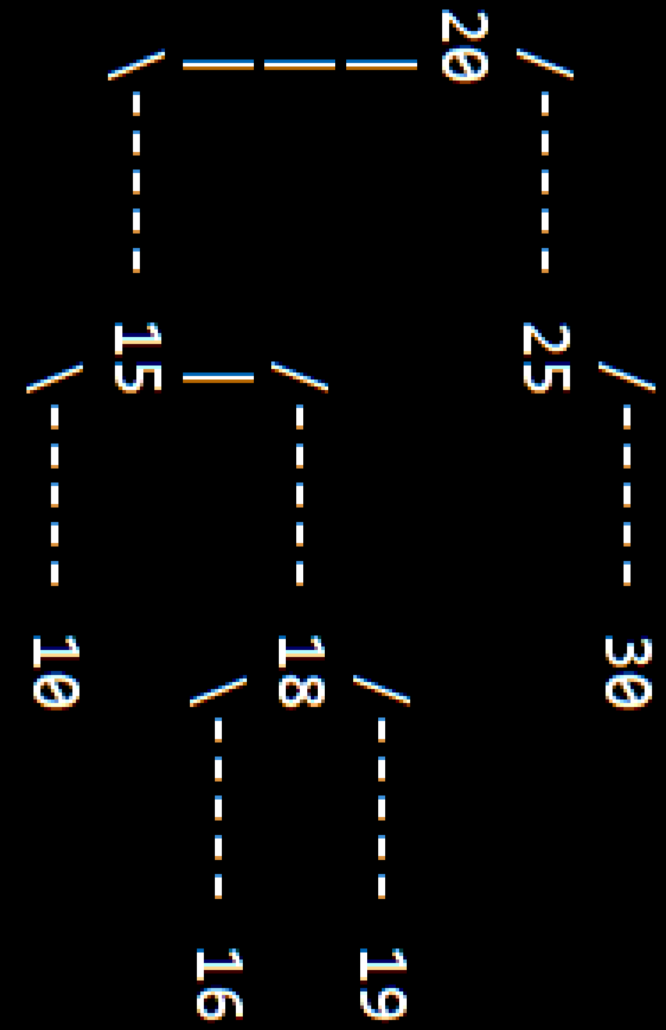


Code

zvimints@gmail.com – צבי מינץ

Output





אוניברסיטת אריאל בשומרון

AVL Height (Lemma)

גובה של עץ AVL עם n איברים הוא $O(\log n)$.

We can show that an AVL tree with n nodes has $O(\log n)$ height. Let N_h represent the minimum number of nodes that can form an AVL tree of height h.

If we know N_{h-1} and N_{h-2}, we can determine N_h. Since this N_h-noded tree must have a height h, the root must have a child that has height $h-1$. To minimize the total number of nodes in this tree, we would have this sub-tree contain N_{h-1} nodes. By the property of an AVL tree, if one child has height $h-1$, the minimum height of the other child is $h-2$. By creating a tree with a root whose left sub-tree has N_{h-1} nodes and whose right sub-tree has N_{h-2} nodes, we have constructed the AVL tree of height h with the least nodes possible. This AVL tree has a total of $N_{h-1}+N_{h-2}+1$ nodes (N_{h-1} and N_{h-2} coming from the sub-trees at the children of the root, the 1 coming from the root itself).

The base cases are $N_1 = 1$ and $N_2 = 2$. From here, we can iteratively construct N_h by using the fact that $N_h = N_{h-1} + N_{h-2} + 1$ that we figured out above.

Using this formula, we can then reduce as such:

$$N_h = N_{h-1} + N_{h-2} + 1 \quad (1)$$

$$N_{h-1} = N_{h-2} + N_{h-3} + 1 \quad (2)$$

$$N_h = (N_{h-2} + N_{h-3} + 1) + N_{h-2} + 1 \quad (3)$$

$$N_h > 2N_{h-2} \quad (4)$$

$$N_h > 2^{\frac{h}{2}}$$

$$\log N_h > \log 2^{\frac{h}{2}}$$

$$2 \log N_h > h$$

$$h = O(\log N_h) \quad (8)$$

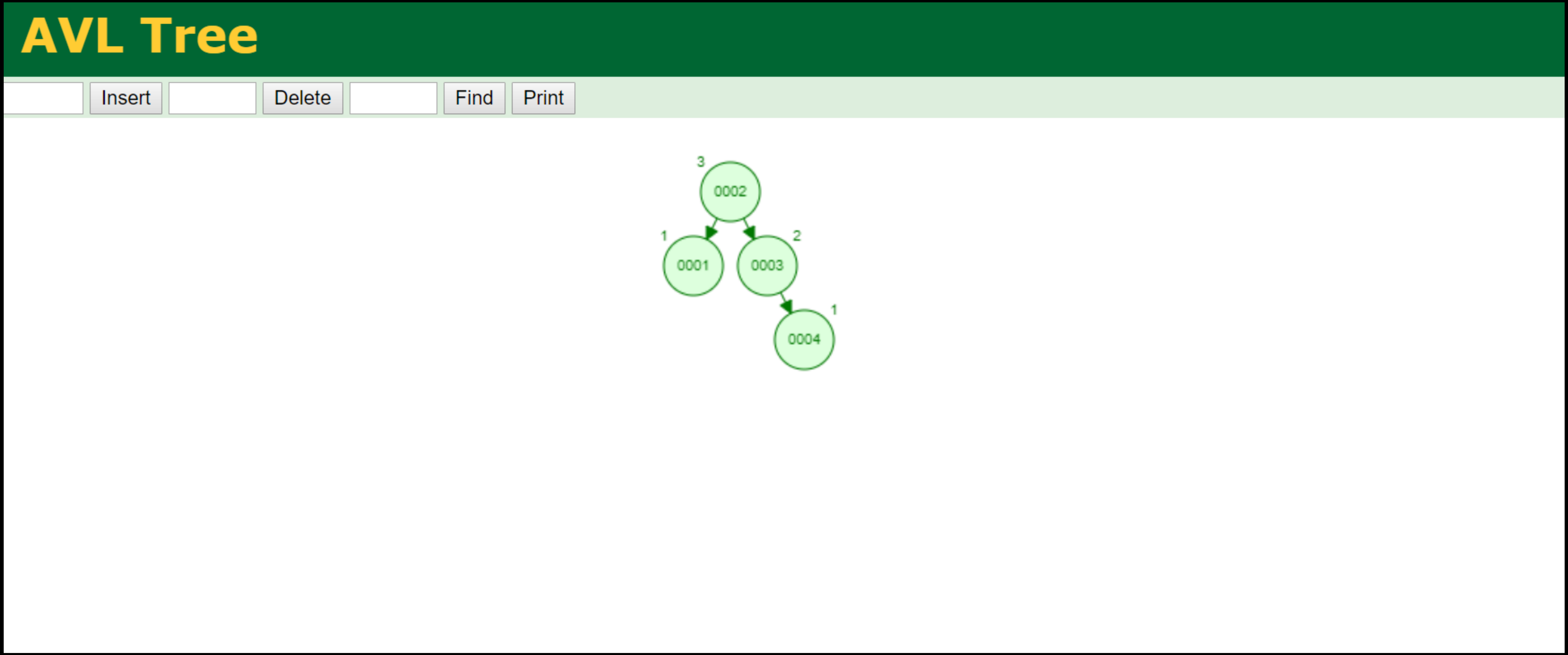
Showing that the height of an AVL tree is indeed $O(\log n)$.

כמות הקודקודים בעץ בגובה h הינה לפחות $2^{\frac{h}{2}} - 1$

If there are n nodes in AVL tree,
maximum height can't exceed $1.44 \cdot \log_2 n$.

https://courses.csail.mit.edu/6.006/spring11/rec/rec04.pdf

visualization

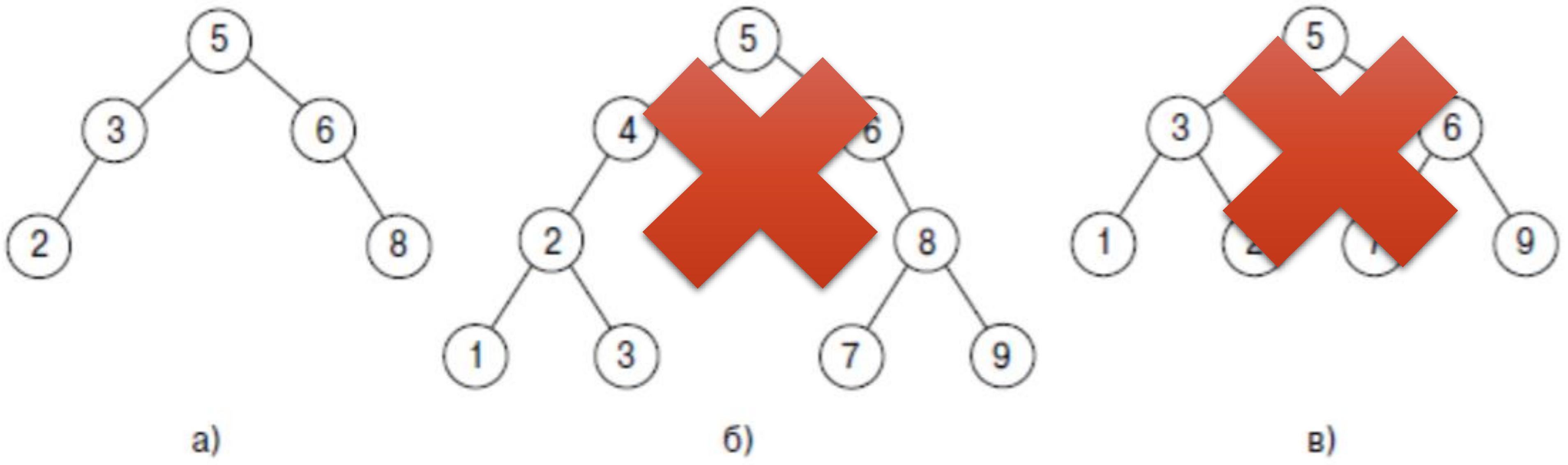


https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

שאלות

תרגיל 11 לעבודה עצמית עץ AVL

שאלה 1 בין העצים הבאים מהם עצים AVL?



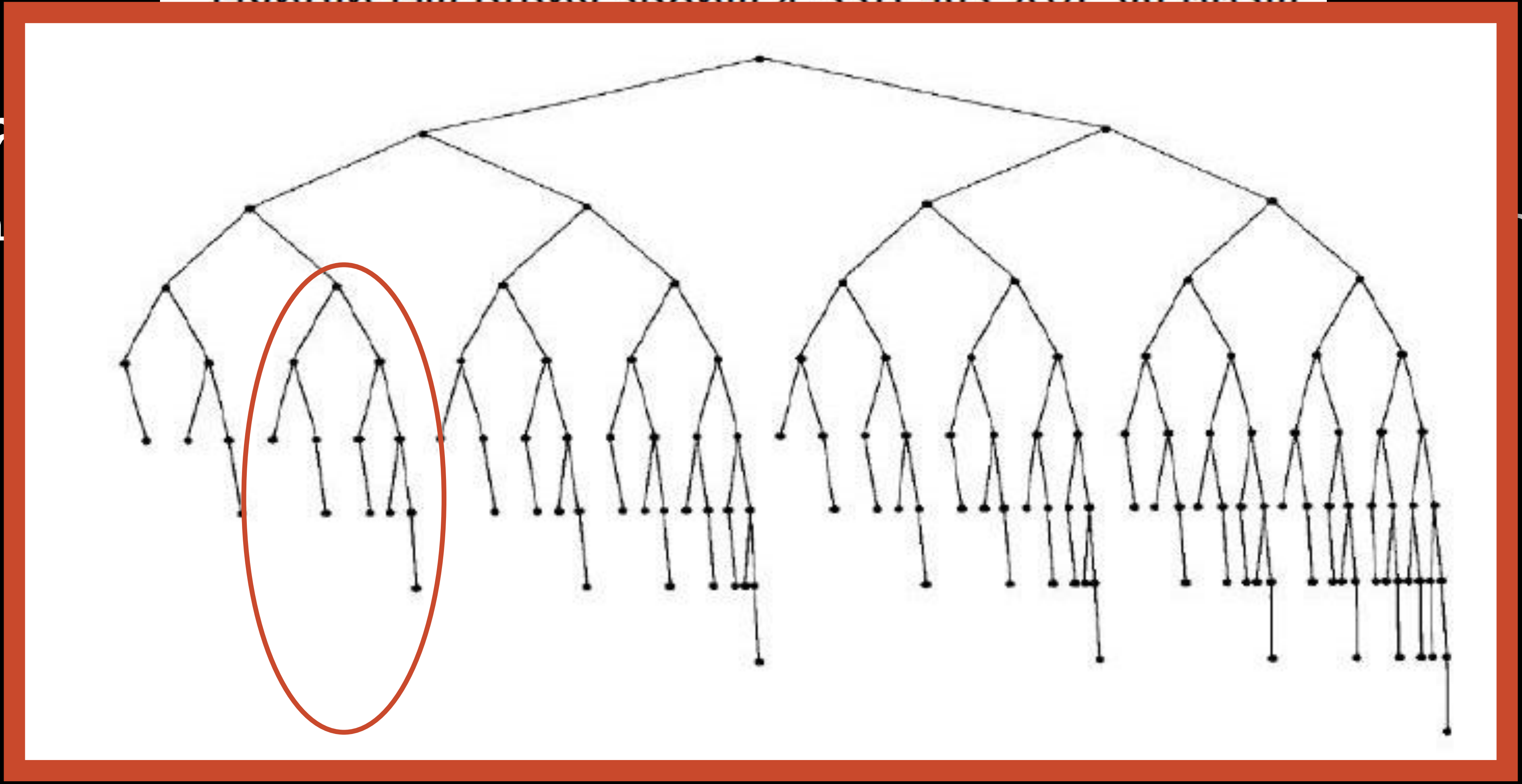
שאלה 3

נגדיר (h) ... h גובה

כדי שמספ... $h-2$ בה

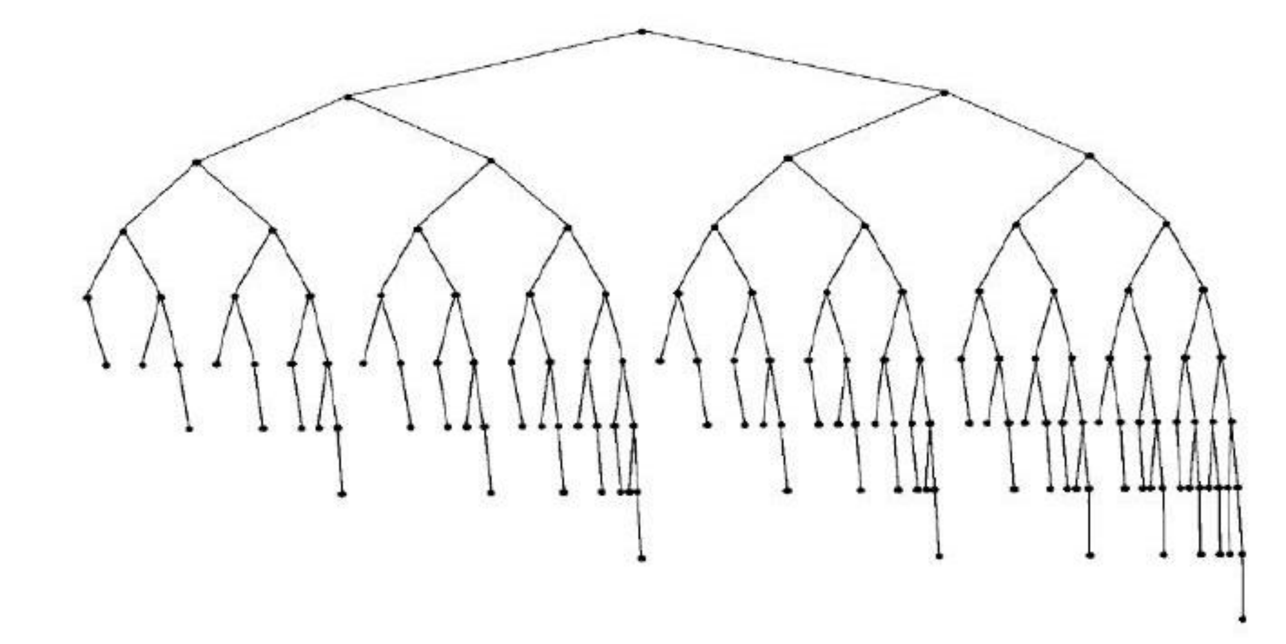
1

2



$$N(4) = N(3) + N(2) + 1 = 4 + 7 + 1 = 12$$

סעיף א' (12%): סמן נכון / **לא נכון** : בעץ AVL ההפרש בעומק בין שני עלים כלשהם הוא לכל היותר 1.



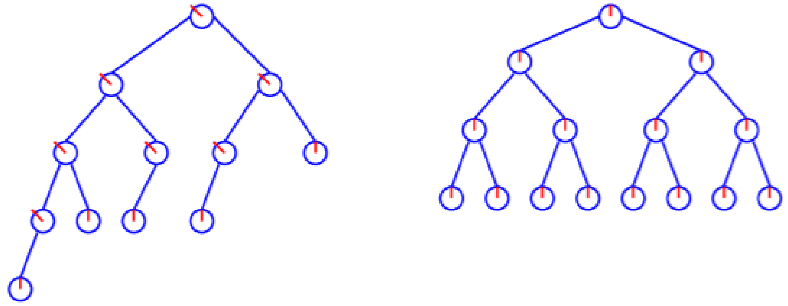
שאלה 6

האם הטענה הבאה נכונה: המפתח המינימאלי והמפתח המקסימאלי של עץ AVL נמצאים ברמה אחרונה או ברמה שלפני אחרונה.

zvimints@gmail.com – צבי מינץ

2. Prove or disprove: For all integers $h \geqslant 0$, any AVL tree of height $h+1$ contains strictly more nodes than any AVL tree of height h. **(6)**

Solution *False*: A Fibonacci tree of height four consists of only $F_{4+3} - 1 = F_7 - 1 = 13 - 1 = 12$ nodes, wherea full binary tree of height three contains $1 + 2 + 4 + 8 = 15$ nodes.



(a) Fibonacci tree of height 4 (b) Complete tree of height 3

AVL trees

(a) An insertion in an AVL with n nodes requires $\Theta(n)$ rotations.

False. Each insertion will require either no rotations, a single rotation, or a double rotation. So, the total number of rotations is in $\Theta(1)$.

https://leetcode.com/problems/balanced-binary-tree/submissions/

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as:

> a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

Example 1:

Given the following tree `[3,9,20,null,null,15,7]`:

```
    3
   / \
  9  20
    /  \
   15   7
```

Return true.

LeetCode

עבודה עצמית (15 דקות)

zvimints@gmail.com – צבי מינץ

https://leetcode.com/problems/balanced-binary-tree/submissions/

```
public static boolean isBalanced(Node root) {
    if(root == null)
    return true;
    else
    return (
            isBalanced(root.right) &&
            isBalanced(root.left) &&
            (Math.abs(height(root.right) - height(root.left)) <= 1)
            );
}

public static int height(Node root) {
    if(root == null)
        return -1;
    else
        return Math.max(height(root.left),height(root.right)) + 1;
}
```

האם ניתן לשכפל עץ AVL חוקי ע"י פעולות BST של הכנסה ומחיקה של ערכים (ללא סיבובים), כך שלאחר כל הכנסה\מחיקה העץ יישאר מאוזן?

פתרון: כן, נכניס את האיברים לפי סדר הרמות שלהם.

שאלה 5 (20 נקודות)

שאלה זו עוסקת בעץ AVL.

א. צייר עץ AVL לאחר הוספה של כל אחד מהאיברים הבאים (משמאל לימין): 5,6,7,1,2,3.
(סה"כ שישה ציורים).

ב. עץ Tri-AVL הוא עץ שמקיים את התכונה הבאה: לכל צומת יש לכל היותר שלשה ילדים, וההפרש בין הגבהים הוא לכל היותר 2. כלומר, אם גובה צומת מסוים הוא h, אז גובה הילדים שלו הוא בין h-1 ל-h-3.

האם עץ Tri-AVL הוא בהכרח מאוזן, כלומר האם גובהו O(log n)? אם כן הוכח, אחרת, הבא דוגמה נגדית.

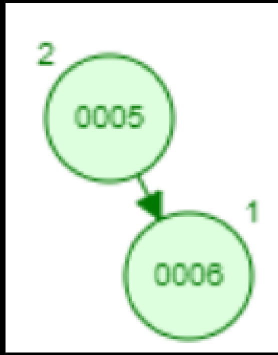
exam 6.8.17

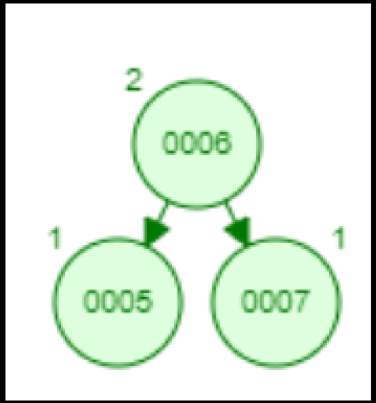
zvimints@gmail.com – צבי מינץ

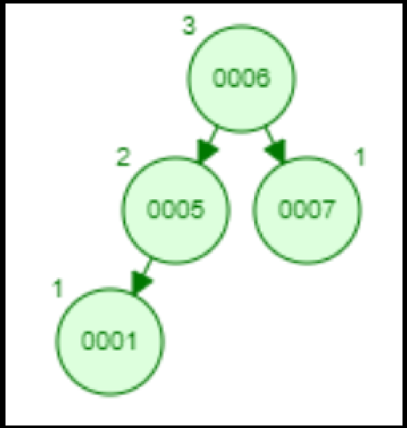
שאלה זו עוסקת בעץ AVL.

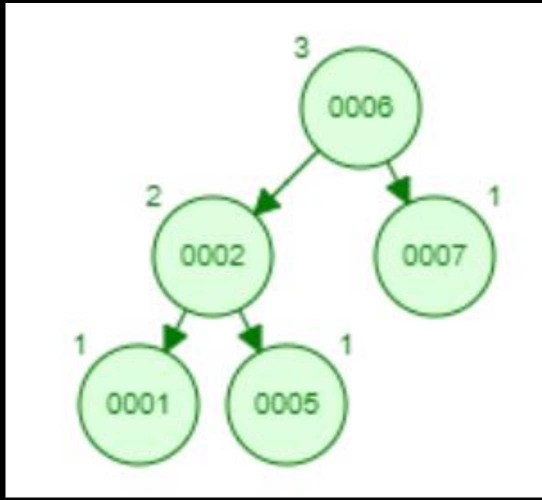
א. צייר עץ AVL לאחר הוספה של כל אחד מהאיברים הבאים (משמאל לימין): 5,6,7,1,2,3.

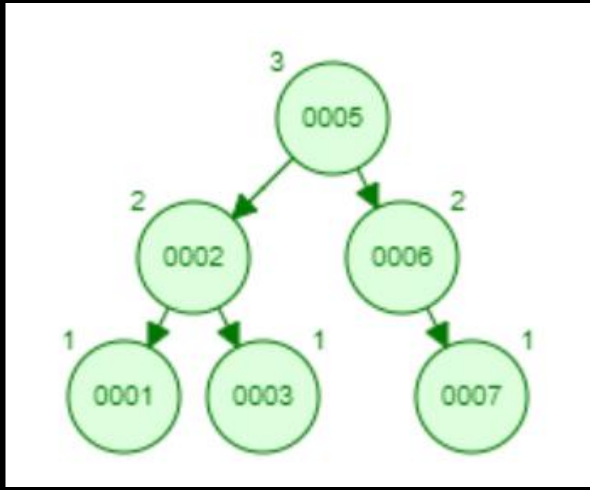
(סה"כ שישה ציורים).











exam 6.8.17

ב. עץ Tri-AVL הוא עץ שמקיים את התכונה הבאה: לכל צומת יש לכל היותר שלשה ילדים, וההפרש בין הגבהים הוא לכל היותר 2. כלומר, אם גובה צומת מסוים הוא h, אז גובה הילדים שלו הוא בין h-1 ל-h-3.

האם עץ Tri-AVL הוא בהכרח מאוזן, כלומר האם גובהו O(log n)? אם כן הוכח, אחרת, הבא דוגמה נגדית.

נסמן ב-n_h את מספר הקודקודים **המינמלי** בעץ בגובה h

ולכן:

$$n_h \geq 1 + n_{h-1} + n_{h-k}$$

$$n_h > 2n_{h-k}$$

$$n_h > 2^{\frac{h}{k}}$$

$$h < k\,log n_h$$

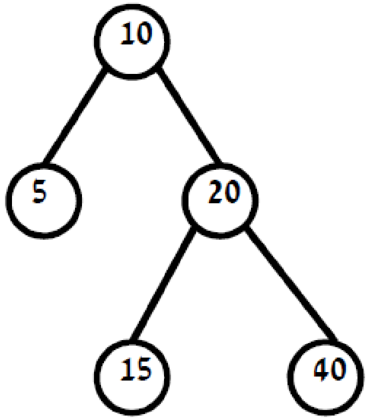
$$h = O(log n)$$

exam 6.8.17

zvimints@gmail.com – צבי מינץ

שאלה 4 (20 נקודות)

נתון העץ AVL הבא :



א. הכנס לעץ את המפתחות הבאים (משמאל לימין). לאחר כל הכנסה בדוק אם העץ נשאר מאוזן ואזן אותו במידת הצורך. עליך לפעול לפי האלגוריתמים שנלמדו בכתה. ציין את סוג הגלגולים שביצעת, ועל איזה קדקוד הופעלו.

80, 70, 90, 75, 78, 100,13

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

ב. סעיף זה אינו נגדיר עץ **BVL** כעץ חיפוש בינרי בו ההפרש בין גובה תת העץ השמאלי ותת העץ הימני של כל צומת הוא לכל היותר 2. הוכח כי גובה עץ BVL הוא O(log n) כאשר n מסמן את מספר הצמתים בעץ.

שקופית קודמת

SampleExam

Red Black Tree

zvimints@gmail.com – צבי מינץ

Red Black Tree

- עץ הינו מבנה נתונים מסוג עץ חיפוש בינארי מאוזן בקירוב.

- עץ אדום-שחור הוא מבנה נתונים **<u>מורכב יחסית</u>**, אך בשל היותו מאוזן הוא שומר על סיבוכיות זמן ריצה טובה.
הפעולות הבסיסיות "הכנסה", "הוצאה" ו"חיפוש" מתבצעות בזמן $O(logn)$ במקרה הגרוע ביותר (עבור עץ בעל n איברים)

Red Black Tree

- עץ הינו מבנה נתונים מסוג עץ חיפוש בינארי מאוזן בקירוב.

- עץ אדום-שחור הוא מבנה נתונים **<u>מורכב יחסית</u>**, אך בשל היותו מאוזן הוא שומר על סיבוכיות זמן ריצה טובה, יעילה ומעשית עבור הפעולות הבסיסיות "**הכנסה**", "**מחיקה**" ו"**חיפוש**" בזמן $O(logn)$ במקרה הגרוע ביותר (עבור עץ בעל n איברים)

- האיזון בעץ נשמר בגלל רוטציות או שינוי בצבעים

- לכל צומת בעץ יש שדה בולאני המכונה "צבע" בעל ערך אדום או שחור (0 או 1)

Red Black Tree

Properties

1. כל צומת בצבע אדום **__או__** שחור

2. השורש **__תמיד__** שחור

3. **__כל__** העלים (NIL) שחורים

4. שני ילדיו של צומת בצבע אדום הם __שניהם__ שחורים

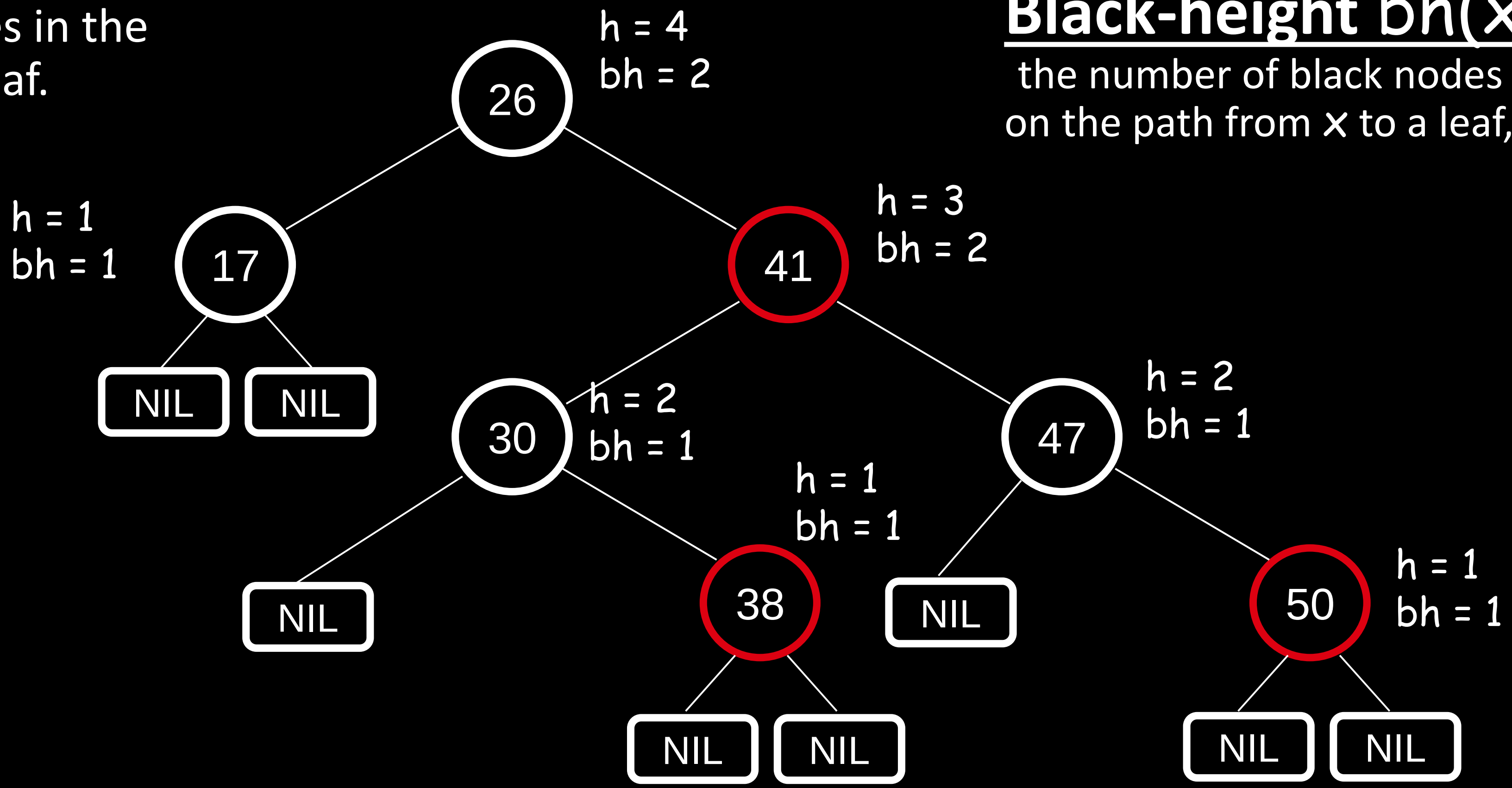
5. כל מסלול פשוט מצומת מסויימת לכל אחד מהצאצאים העלים שלו מכיל **__אותו מספר__** של צמתים שחורים

Red Black Tree

Definitions

Height of a node h(x):
the number of edges in the **longest** path to a leaf.

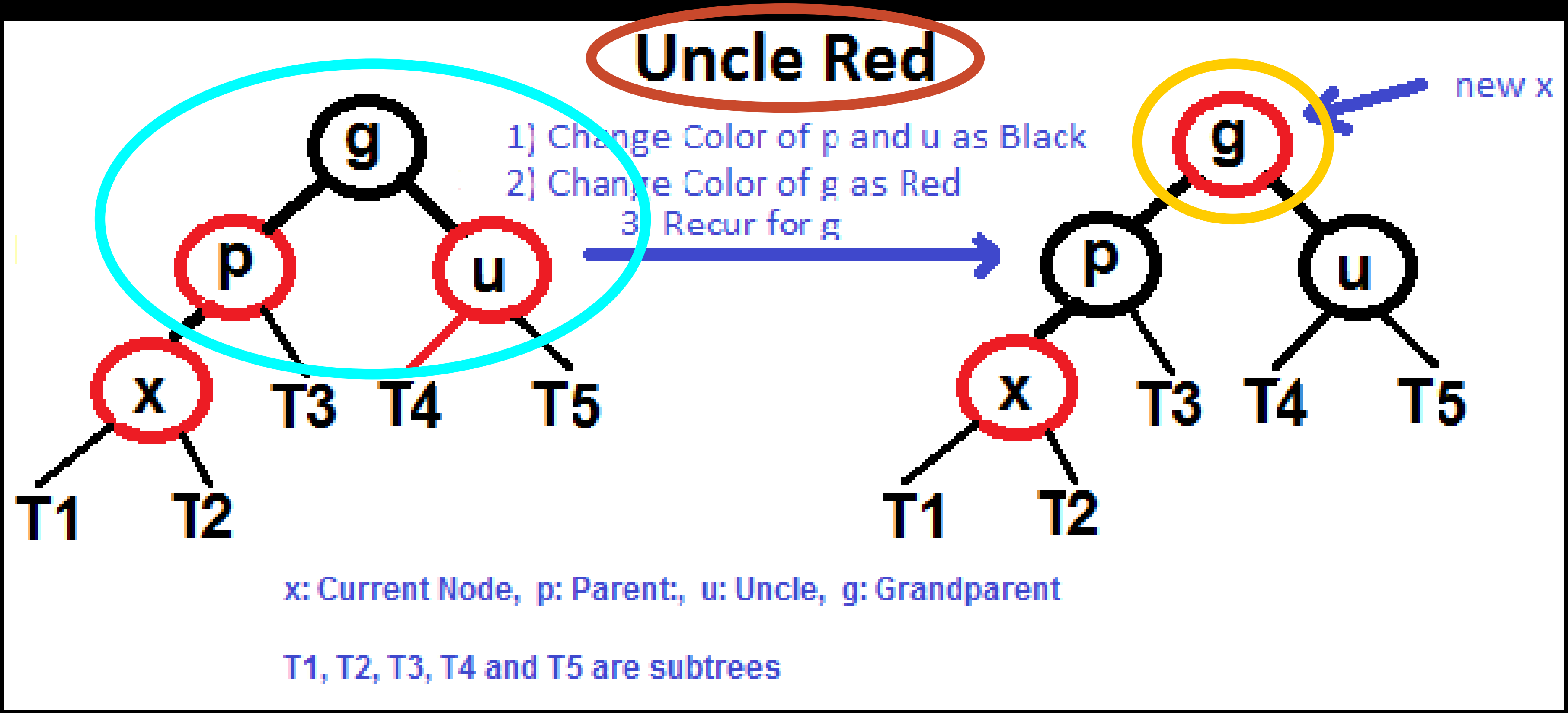
Black-height bh(x) of a node x:
the number of black nodes (including NIL) on the path from x to a leaf, not counting x.



zvimints@gmail.com – צבי מינץ

Red Black Tree

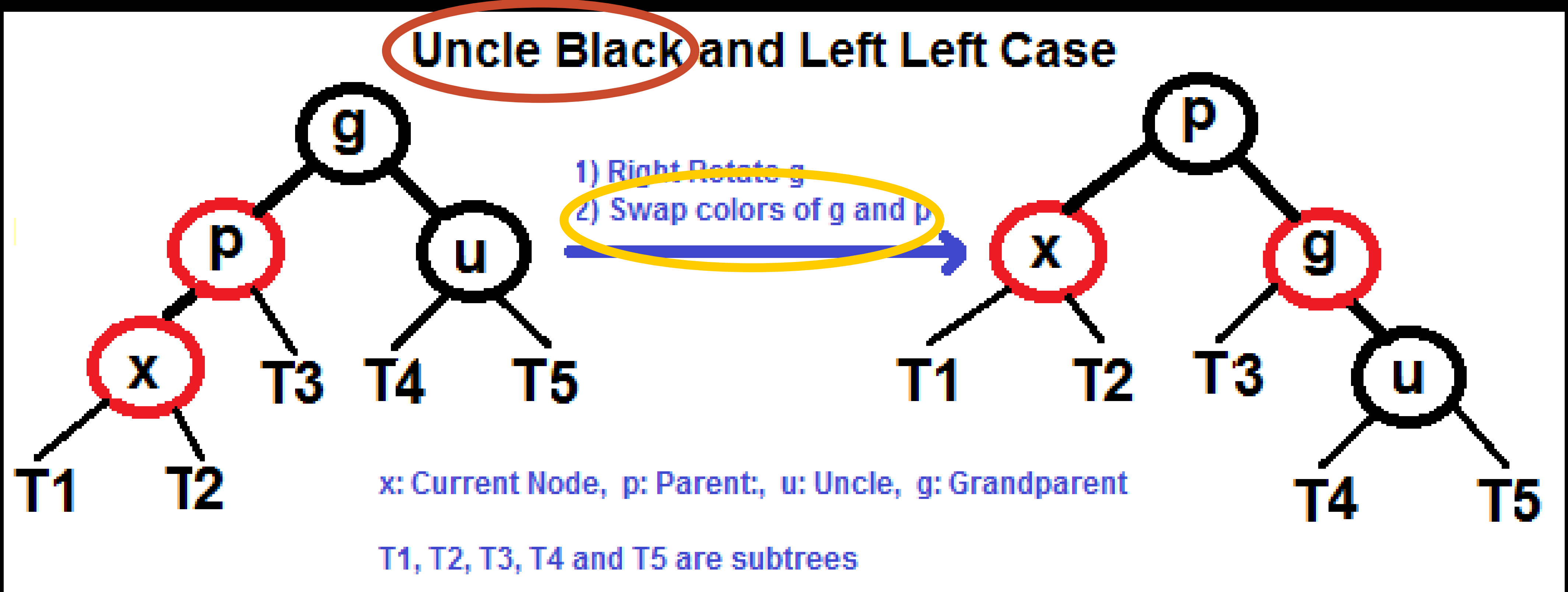
Insertion



https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Red Black Tree

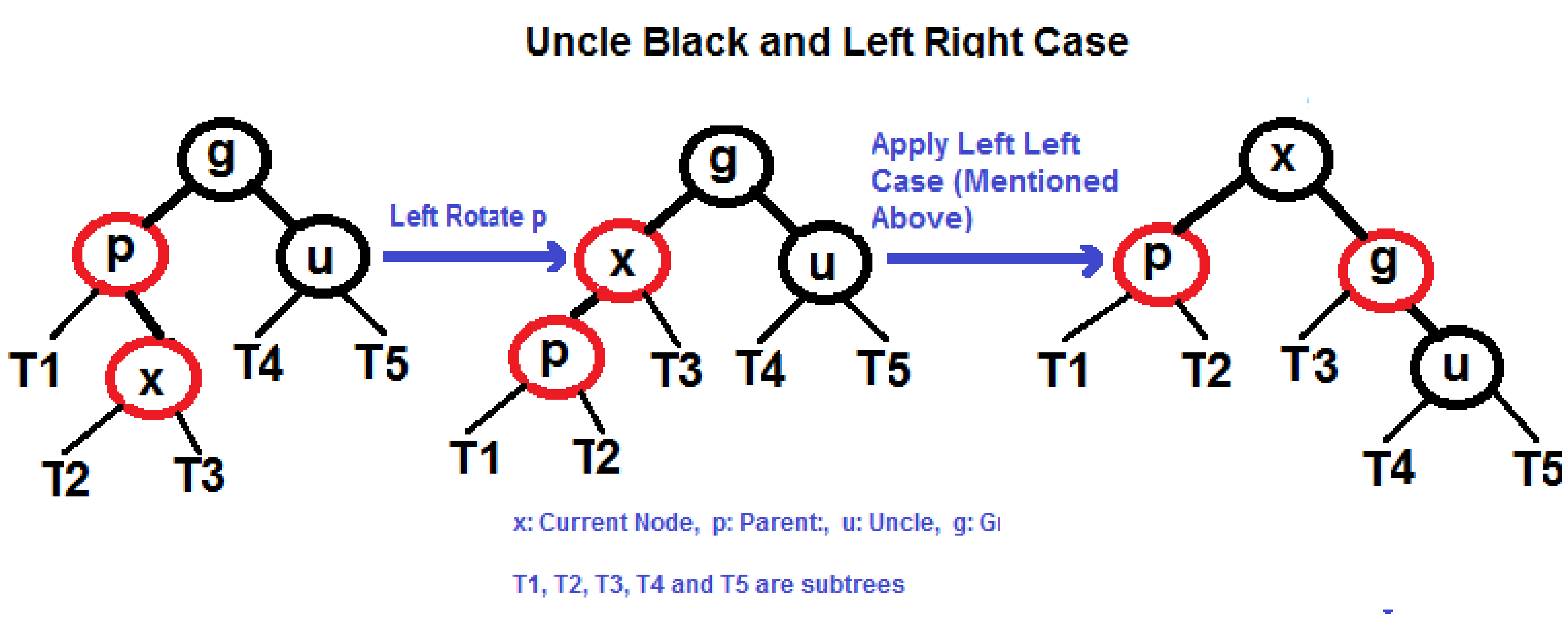
Insertion



zvimints@gmail.com – צבי מינץ https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Red Black Tree

Insertion

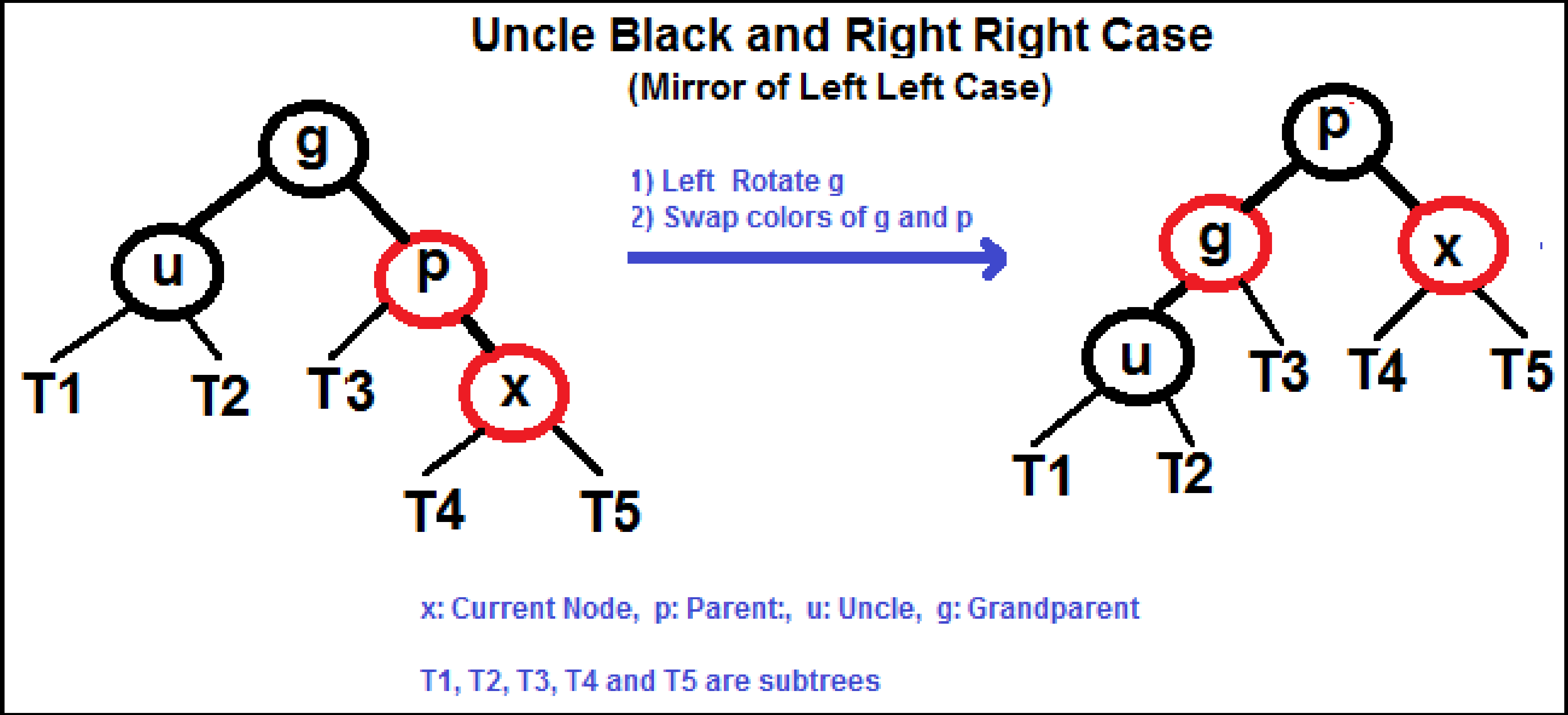


zvimints@gmail.com – צבי מינץ

https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Red Black Tree

Insertion

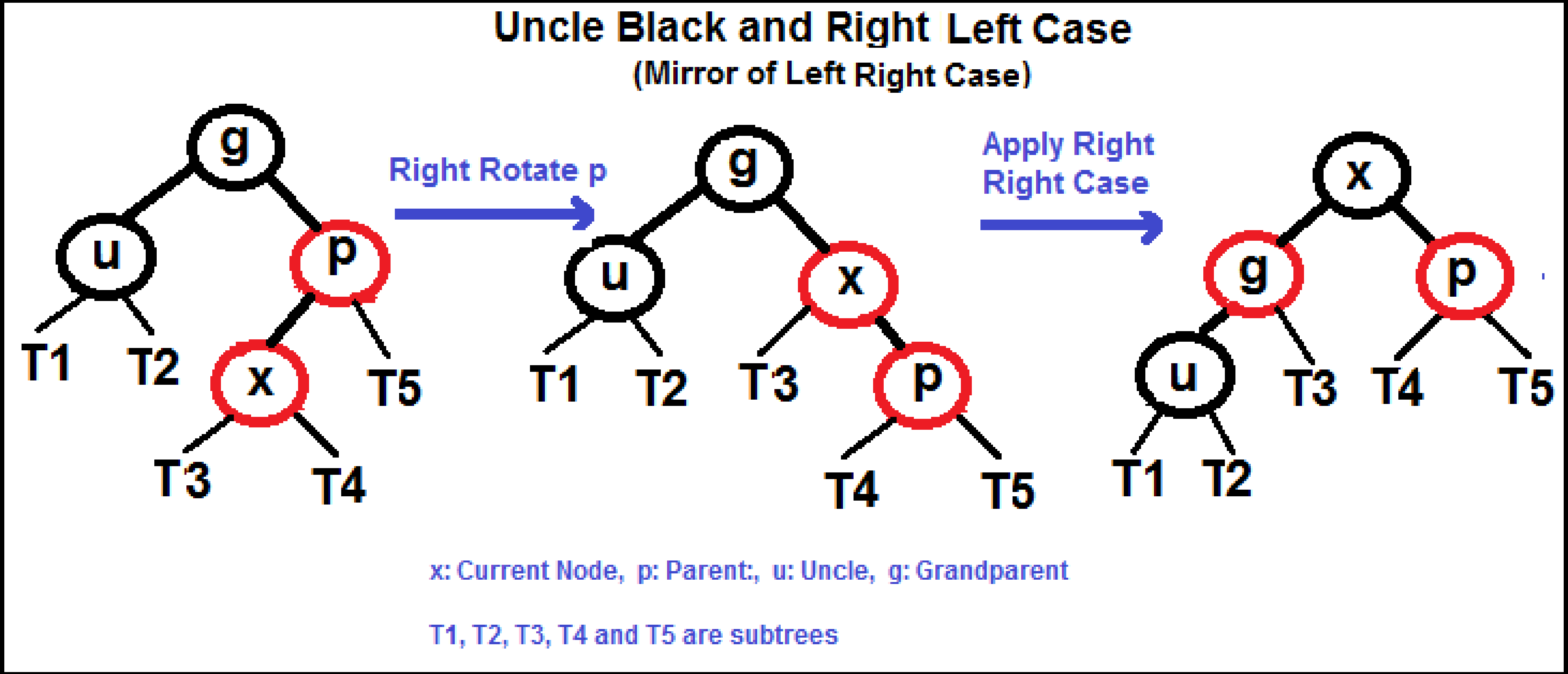


https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

zvimints@gmail.com – צבי מינץ

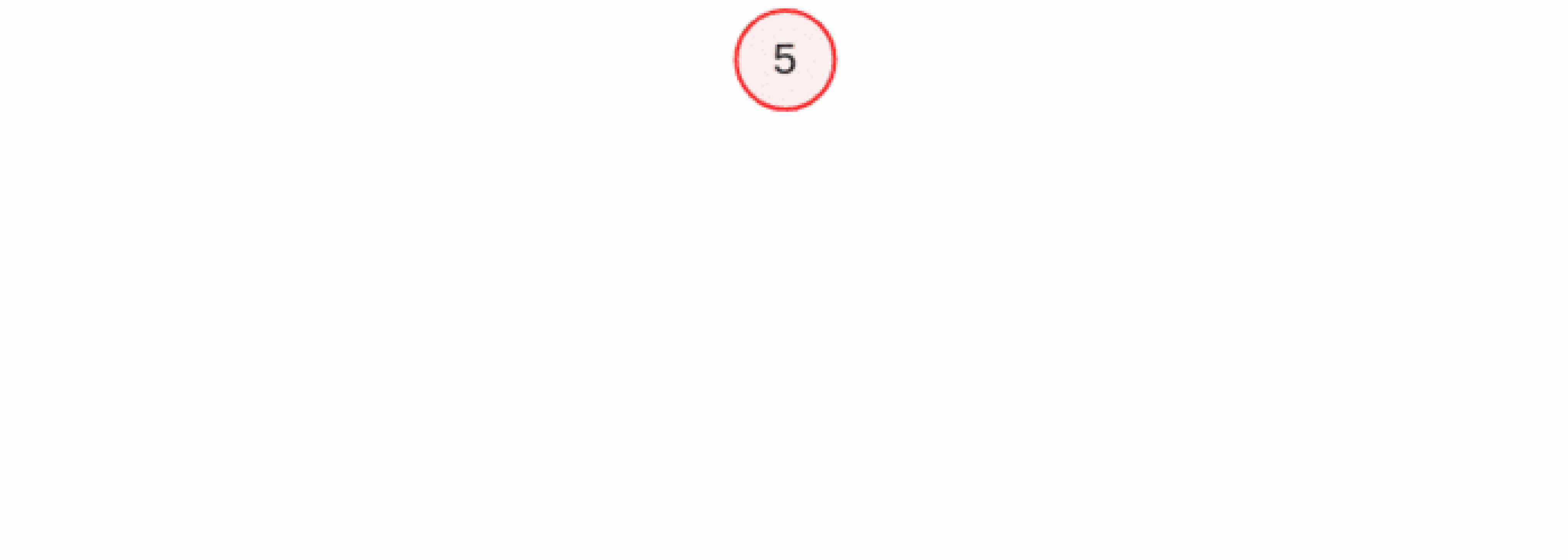
Red Black Tree

Insertion



Red Black Tree

Insertion Example



zvimints@gmail.com – צבי מינץ

Red Black Tree

Claim 1

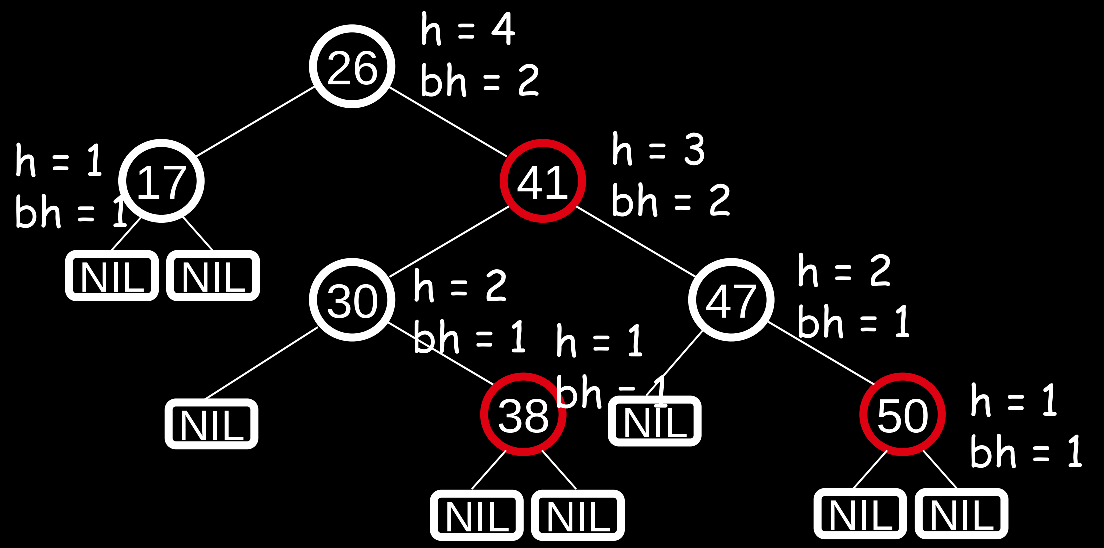
Any node x with height $h(x)$ has $bh(x) \geq \frac{h(x)}{2}$

Proof

4. שני ילדיו של צומת בצבע אדום הם שניהם שחורים

By property 4, at **<u>most</u>** $\frac{h}{2}$ **red** nodes on the path from the node to a leaf

Hence at least $\frac{h}{2}$ are **black**



Red Black Tree

Claim 2

The subtree rooted at any node x contains **at least** $2^{bh(x)} - 1$ internal nodes

Proof: By induction on **h[x]**

Basis: $h[x] = 0 \Rightarrow$

x is a leaf (NIL[T]) $\Rightarrow$

$bh(x) = 0 \Rightarrow$

\# of internal nodes: $2^0 - 1 = 0$

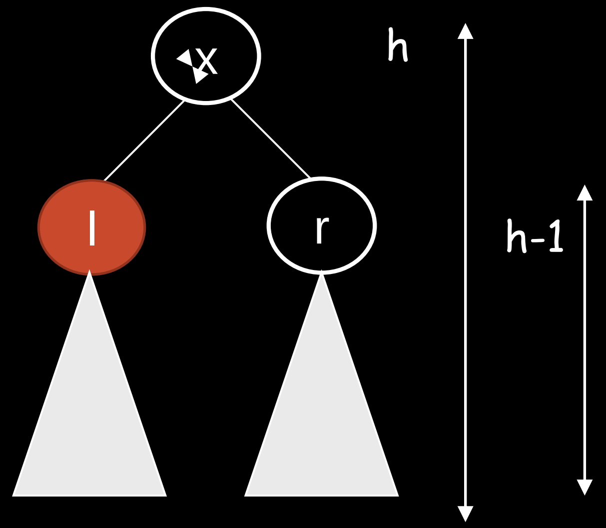
x

$\boxed{\text{NIL}}$

Red Black Tree

Claim 2

The subtree rooted at any node x contains **at least** $2^{bh(x)} - 1$ internal nodes



Inductive Hypothesis:

assume it is true for $h[x]=h-1$

Prove it for $h[x]=h$

internal nodes at x = internal nodes(L) + internal nodes(R)+ 1

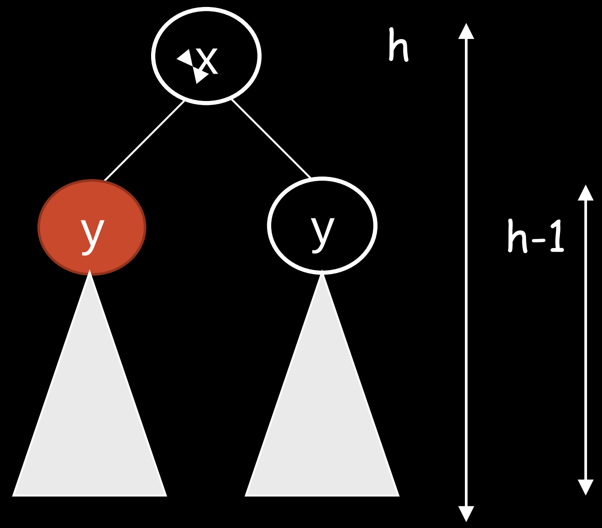
Using inductive hypothesis:

internal nodes at x $\geq (2^{bh(l)} - 1) + (2^{bh(r)} - 1) + 1$

Red Black Tree

Claim 2 (cont'd)

The subtree rooted at any node **x** contains **at least** $2^{bh(x)} - 1$ internal nodes



Let **bh**(x) = b, then any **child** y of x has:

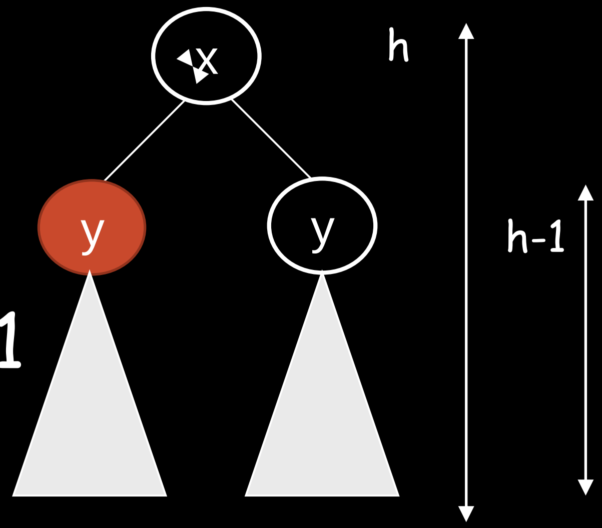
bh(y) = b (if the child is **red**), or
bh(y) = b - 1 (if the child is **black**)

$\Rightarrow bh(y) \geq bh(x)-1$

Red Black Tree

Claim 2 (cont'd)

The subtree rooted at any node x contains **at least** $2^{bh(x)} - 1$ internal nodes



$$\text{internal nodes}(x) \geq (2^{bh(l)} - 1) + (2^{bh(r)} - 1) + 1$$
$$\geq (2^{bh(x)-1} - 1) + (2^{bh(x)-1} - 1) + 1$$
$$= 2 \cdot (2^{bh(x)-1} - 1) + 1 =$$
$$= 2^{bh(x)} - 1 \quad \text{internal nodes}$$

Red Black Tree

Height of Red-Black-Trees (cont'd)

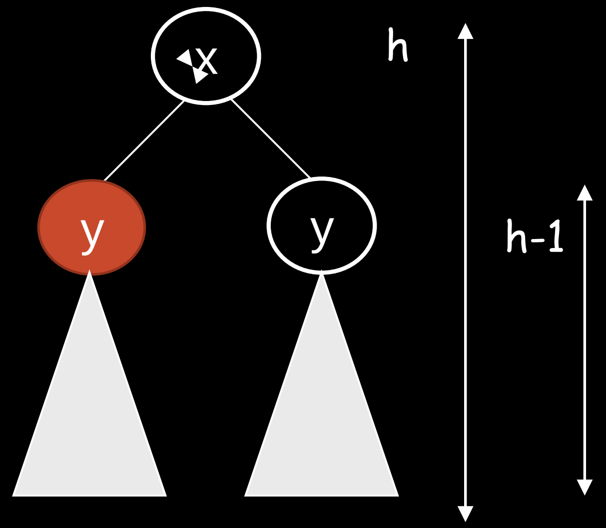
A red-black tree with N internal nodes has height at most $2\log(N+1)$.

Proof:

N   **bh(root)** = b   **height(root)** = h

$\geq 2^b - 1$ $\geq 2^{h/2} - 1$

number of internal nodes   Claim 2   Claim 1



Solve for h:

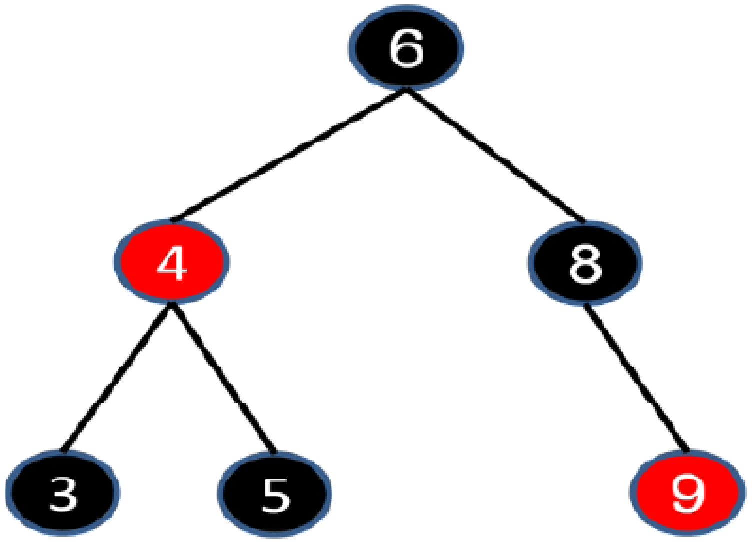
$$N + 1 \geq 2^{h/2}$$

$$\log(N + 1) \geq h/2 \Rightarrow$$

$$h \leq 2\log(N + 1)$$

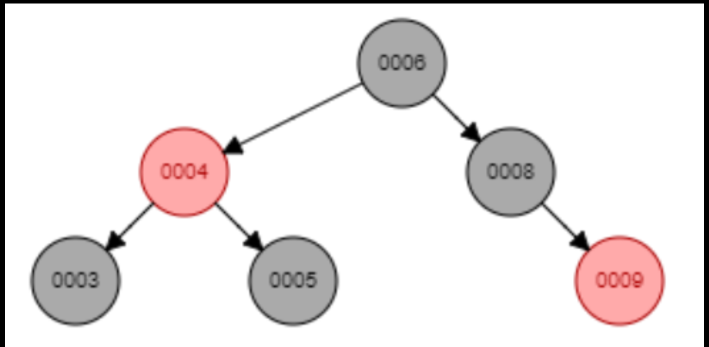
<u>שאלה 2 (20 נקודות)</u>

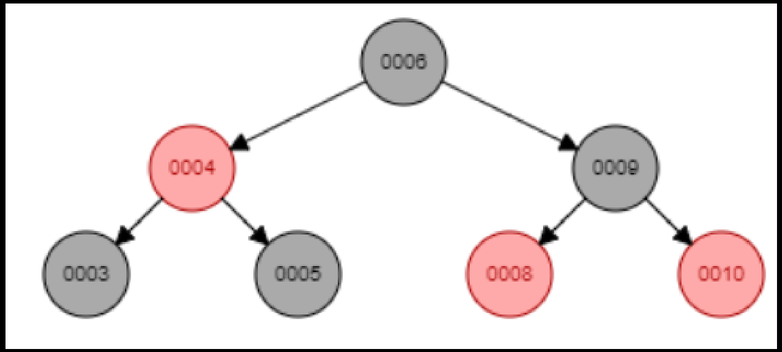
נתון עץ אדום-שחור. הוסף לעץ את האיברים הבאים (<u>משמאל לימין</u>): 10,7,2,1. סה"כ 4 ציורים. מהו זמן הריצה לכל הכנסה?

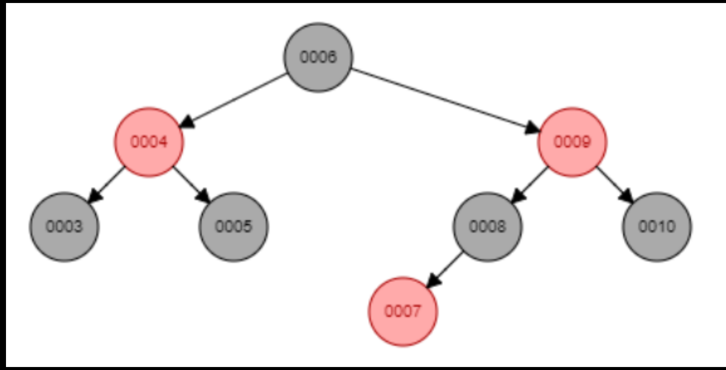


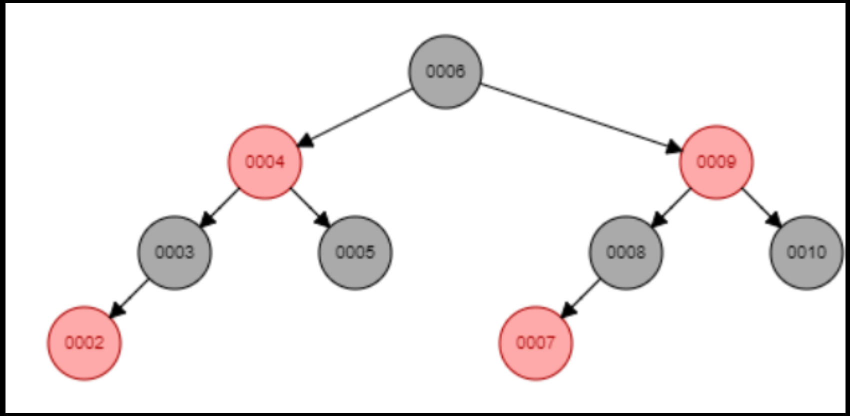
exam 9.7.17

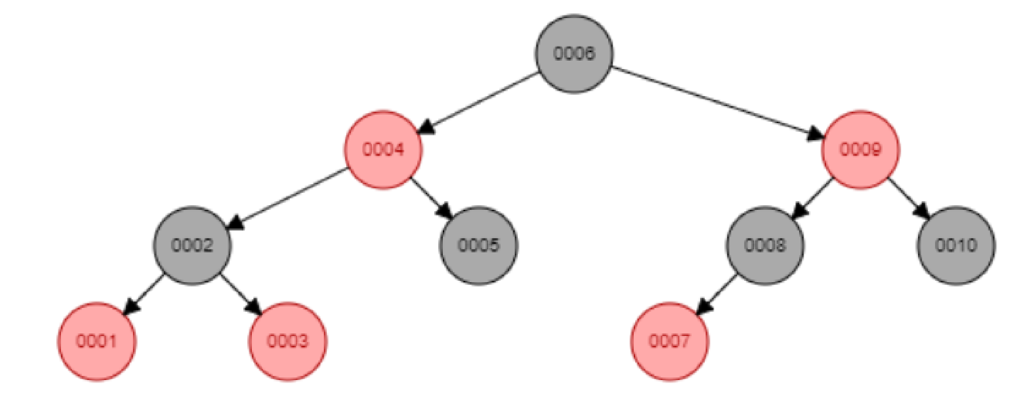
הוסף לעץ את האיברים הבאים (משמאל לימין): 10,7,2,1.











https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

exam 9.7.17

אז מה נעדיף? עץ AVL או עץ אדום שחור?

לעץ AVL יש יתרונות וחסרונות, הדחיסה היא גבוהה יותר וזה מתבטא בכך שלכל הוספת איבר / מחיקה יעלה לנו ביותר פעולות איזון כדי להגיע לגובה נמוך ביותר.

לעומת זאת בעץ אדום שחור אנו מבצעים פחות פעולות איזון כי אנו מאפשרים לעץ להיות **פחות** מאוזן
ולכן כל פעולת החיפוש תהיה במקרה הגרוע ביותר ארוכה יותר אבל ההוספה והסרה יקחו פחות זמן.

מבחינת סדרי גודל אין הבדל בין העצים אך אם אנו רוצים ביצועים אופטמליים כן אפשר להעדיף עץ אחד על פני האחר.
אם ברצוננו בעץ סטטי יחסית שאנו בונים פעם אחת ואז מריצים עליו חיפושים רבים אבל נעדיף AVL
אבל אם נעשה הרבה הסרות והוספות אז עדיף עץ אדום שחור.

צבי מינץ – zvimints@gmail.com