# מעבדה 8-9

# נושא: עץ חיפוש בינארי מאוזן

תאריך הגשה לקבוצה של יום ב': 04.01.2021

תאריך הגשה לקבוצה של יום ד': 06.01.2021

(הגשה בזוגות)

**יש לקרוא היטב לפני תחילת העבודה!**

**עבודה נעימה!**

## מבוא:

במעבדה הנוכחית נממש עץ חיפוש בינרי מאוזן - עץ AVL.

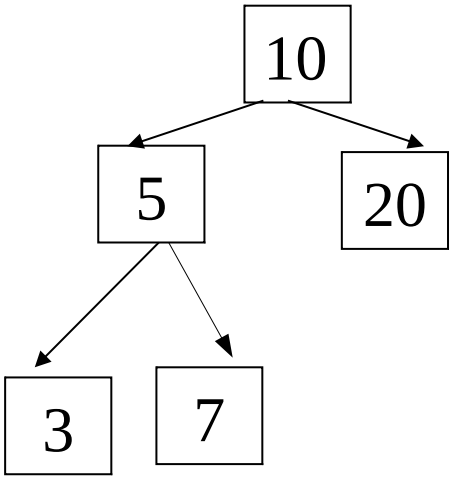
## תיאור:

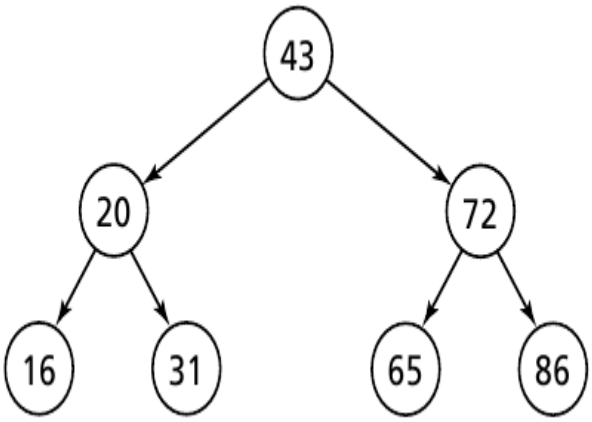
נחפש עץ חיפוש עבורו יתקיימו התנאים הבאים:

1. פעולות החיפוש, ההוספה והמחיקה יתבצעו בסדר גודל של ( $\log(n)$
2. העץ יהיה קל לתחזוקה.

**הגדרה 1 (עץ AVL):** עץ חיפוש בינרי בו **הפרש הגבהים** של תת-העץ השמאלי ותת-העץ הימני (לכל צומת פנימי) יהיה **1** לכל היותר ייקרא עץ AVL על שם Adelson-Velskii and Landis.

דוגמא לעצי AVL:

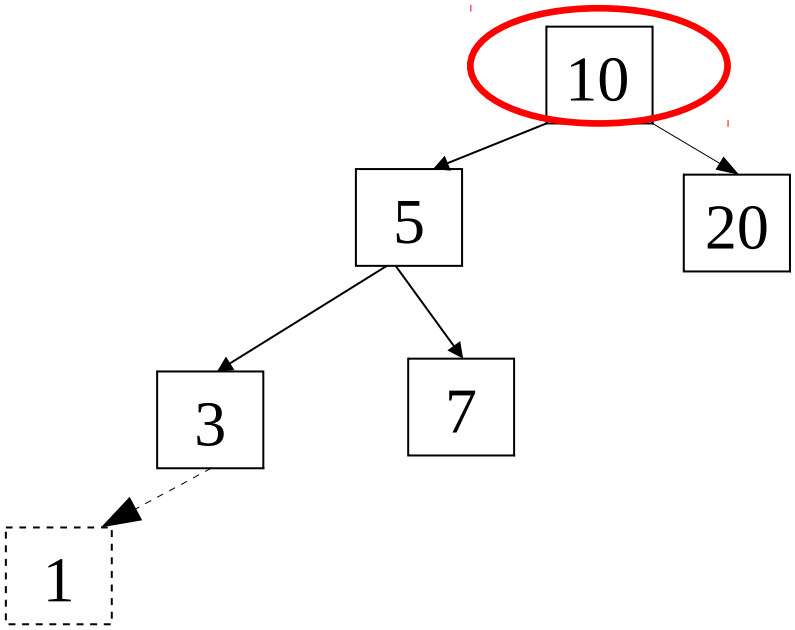


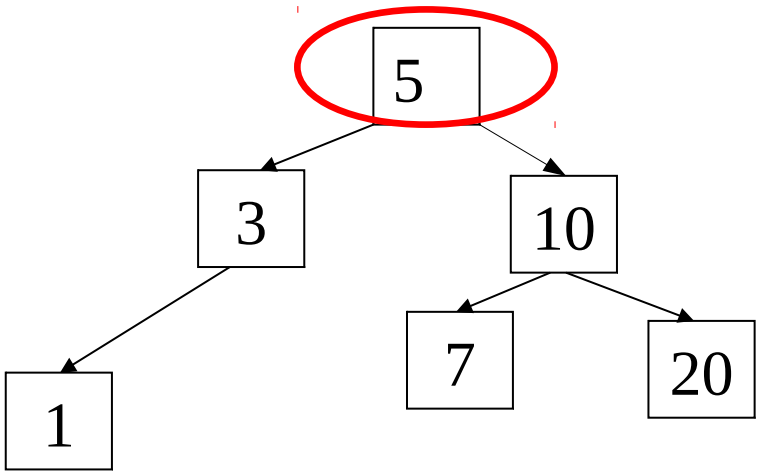
# הכנסת צומת לעץ AVL

ההכנסה נעשית באופן שגרתי לעצי חיפוש: הצומת החדש מתווסף בתור עלה. באופן כזה לא נפגעת תכונת החיפוש של העץ.
אולם, הכנסה עלולה לפגוע בתנאי האיזון. נסמן ב a - את הצומת הראשון במסלול מנקודת ההכנסה אל השורש, שבו יש הפרה של תנאי האיזון.

למשל, הכנסה של 1 גורמת להפרת האיזון בשורש (a הוא השורש):

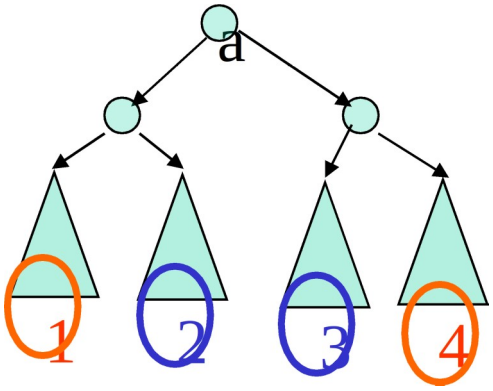



עלינו לתקן את האיזון בין תתי העצים של צומת זה (נראה כי זה מספיק).ניתן לעשות זאת אם נהפוך את 5 להיות שורש העץ ואת 7 לבן השמאלי של 10. נקבל:



## יש ארבעה מקרים, אך משיקולי סימטריה אפשר לדון רק בשני מקרים:

**א.** הכנסה "**חיצונית**". לתת-עץ שמאלי של בן שמאלי או לתת-עץ ימני של בן ימני. מקרים 1 ו-4
**ב.** הכנסה "**פנימית**". לתת-עץ ימני של בן שמאלי או לתת-עץ שמאלי של בן ימני. מקרים 2 ו-3




**פתרון הפרת האיזון:**

כאמור נסמן ב-a את הצומת הראשון במסלול מנקודת ההכנסה אל השורש, שבו יש הפרה של תנאי האיזון.

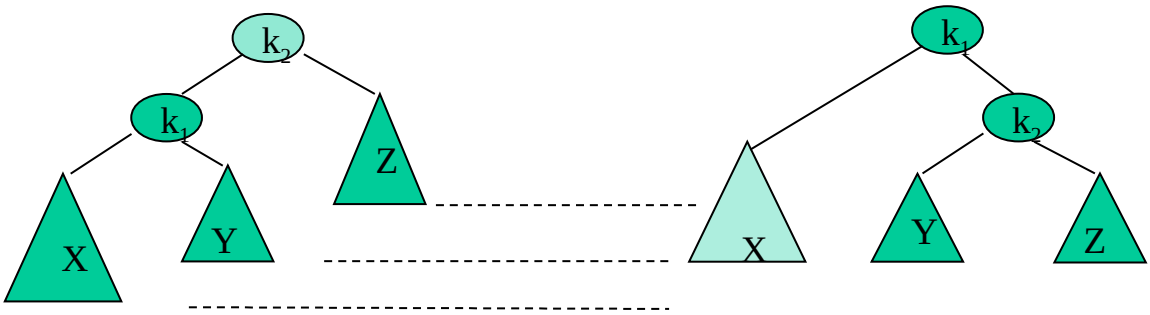
בהכנסה "חיצונית", (למשל: לתת-עץ השמאלי של הבן השמאלי של a) נראה שמספיק סיבוב אחד.

בהכנסה "פנימית", (למשל: לתת-עץ הימני של הבן השמאלי של a) נראה שיש צורך בשני סיבובים.

**הכנסה חיצונית (מקרים 1 ו 4)**

**רוטציה יחידה**

צד שמאל מתאר את המצב לפני התיקון (הרוטציה): אחרי ההוספה לתת-העץ X מופר האיזון. הצומת הראשון שאינו מאוזן מסומן ב- k2 . צד ימין מתאר את המצב לאחר התיקון.



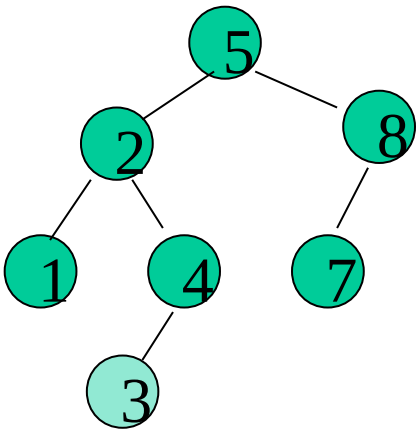

**הערה**: באיור לעיל הראנו רוטציה ימינה שפותרת את המקרה 1. תארו באיור דומה רוטציה שמאלה לפתרון מקרה 4.

**תרגיל עצמי 1**

הכנס מפתח 6 לתוך העץ הנתון.

יש להכניס את המפתח באופן רגיל ואז לבצע רוטציה יחידה.

מהם k1, k2, X,Y,Z במקרה זה?

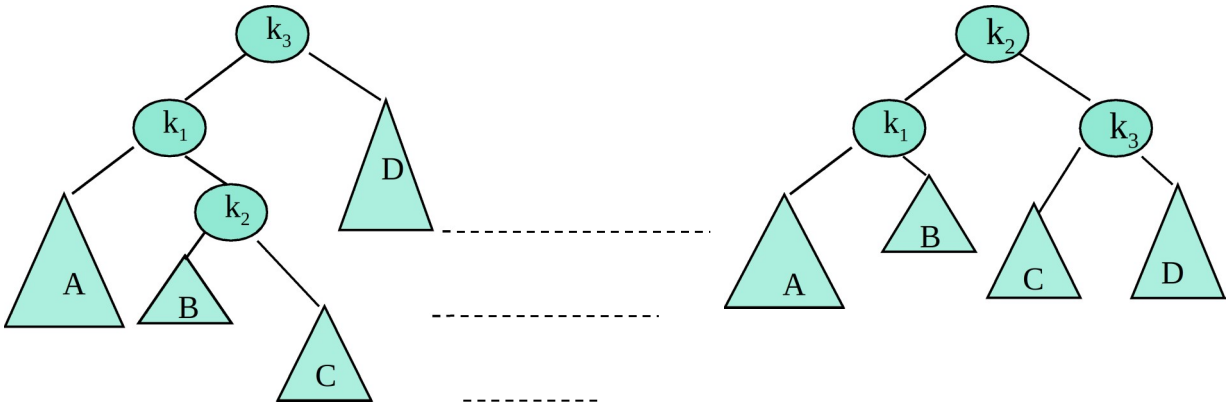


**תרגיל עצמי 2**

בנה עץ AVL אשר נוצר ע"י הכנסת המפתחות הבאים (משמאל לימין):

**3 2 1 4 5 6 7**

**הכנסה פנימית (מקרים 2 ו 3)**
**רוטציה כפולה**




מכניסים לתת-העץ הימני של הבן השמאלי, לכן הוא אינו ריק. לכן אפשר לתאר את העץ באמצעות 4 תתי-עצים (אולי ריקים) ועוד שלושה צמתים מקשרים.

$K_3$ הוא הצומת הראשון (מלמעלה) שבו מתגלה ההפרה. נניח בה"כ שהכנסנו ל-C ולכן הוא עמוק יותר.

1. $k_3$ לא יכול להישאר השורש ורוטציה עם $k_1$ לא פותרת את הבעיה, לכן נהפוך את $k_2$ להיות השורש.

2. הבנים שלו חייבים להיות: משמאל $k_1$ מימין $k_3$ המיקום של ארבעת תתי-העץ A עד D נקבע בהתאם.

3. עושים רוטציה שמאלה בין $k_2$ ל-$k_1$ ואח"כ רוטציה ימינה בין $k_2$ ל-$k_3$.

4. זה נקראה רוטציה שמאלה-ימינה.

**הערה:** המקרה האנלוגי – הכנסה לתת-עץ השמאלי של הבן הימני נפתר ע"י רוטציה ימינה-שמאלה.

**תרגיל עצמי 3**

לעץ שיצרתם בתרגיל עצמי 2 הכניסו את המפתחות הבאים (משמאל לימין):16,15,14,13,12,11,10,8,9

## אלגוריתם ההכנסה של צומת X לעץ AVL

1. אם העץ ריק, צור צומת חדש והכנס בו את X, השם 0 בשדה הגובה.

2. אחרת, אם X **קטן** מהמפתח של השורש, הכנס משמאל, בדוק אם יש הפרת איזון וטפל בה

3. אחרת אם X **גדול או שווה** מהמפתח של השורש, הכנס מימין, בדוק אם יש הפרת איזון וטפל בה

5. עדכן את גובה העץ והחזר אותו.

**טיפול בהפרת איזון:**

בהכנסה לתת-עץ שמאלי: אם X קטן מהמפתח של הבן השמאלי בצע סיבוב אחד, אחרת בצע סיבוב כפול.

בהכנסה לתת-עץ ימני: אם X גדול מהמפתח של הבן הימני בצע סיבוב אחד, אחרת בצע סיבוב כפול.

**נממש את עץ ה**חיפוש AVL **לפי השלבים הבאים:**

**שלב 1.**

ניצור מחלקה של **צומת בעץ** AVL כל איבר יכיל איבר (key), שני מצביעים וגובה תת-העץ שהצומת הנוכחי הוא השורש שלו. להלן המחלקה:

```
public class AVLTreeNode {
        private int key;
        private int height ;                    // Height of the sub-tree
        private AVLTreeNode left, right;
        public AVLTreeNode ( int elem, int height, AVLTreeNode leftPtr, AVLTreeNode rightPtr ){ }
        // ---Insert set/get methods  here ---//
} // class AVLTreeNode
```

**צור קובץ בשם AVLTreeNode.java וממש את המחלקה הנ"ל.**

**שלב 2.**

צור מחלקה המגדירה **עץ חיפוש בינארי מאוזן** (AVLTree). שים לב: עליך להוסיף מטודות **רקורסיביות** **private methods** הדרושות למימוש העץ.

**(תזכורת: הוספה של צומת מתבצעת רק כעלה)**

להלן המחלקה:

```
public class AVLTree{
private AVLTreeNode root;

public AVLTree ( ){ }
public void insert ( int  newElement ){ }
public AVLTreeNode  retrieve ( int searchKey ){ }
public void clear ( ){ }// Clear tree
public boolean isEmpty ( ){}
public boolean isFull ( ){}
```

```
public String toString ( ){} // Output the tree structure in Inorder
// Recursive partners of the public member methods --- Insert these methods here.
} // class AVLTree
```

**צור קובץ בשם AVLTree.java , וממש את המחלקה המופיעה לעיל.**

## שלב 3. בדיקות

| Command | Action |
|---|---|
| +key | Insert (or update) the element with the specified key. |
| ?key | Retrieve the element with the specified key and output it. |
| | |
| K | Output the keys in ascending order. |
| E | Report whether the tree is empty. |
| F | Report whether the tree is full. |
| C | Clear the tree. |
| Q | Quit the test program. |

א). צור קלאס חדש בשם **TestAVLTree .java** ובדוק בעזרתו את נכונות המחלקות שכתבת. בצע בדיקות .על המחלקה לתמוך בפעולות המופיעות בטבלה שלעיל (ניתן, המידת הצורך לשנות את הסימנים + ו ? ל ADD ו FIND).

## הגשה:

יש להגיש את הקבצים הבאים: **AVLTree.java, AVLTreeNode.java, TestAVLTree .java**

***עבודה נעימה!***