מבני נתונים ואלגוריתמים

עצים ומיוני השוואה

שיעורי בית- הערות

- יש להגיש את התרגיל במערכת submit עד השעה 23:55 בתאריך 19.11.
- יש לממש את הערימה בחלק השני לבד- אני אבדוק!
- יש להשתמש אך ורק במיון ערימה ויש לממש את המיון בעצמכם.
- תהיה בדיקת העתקה.
- יש לתכנת בpython3
- **לכתוב שם ות"ז!**

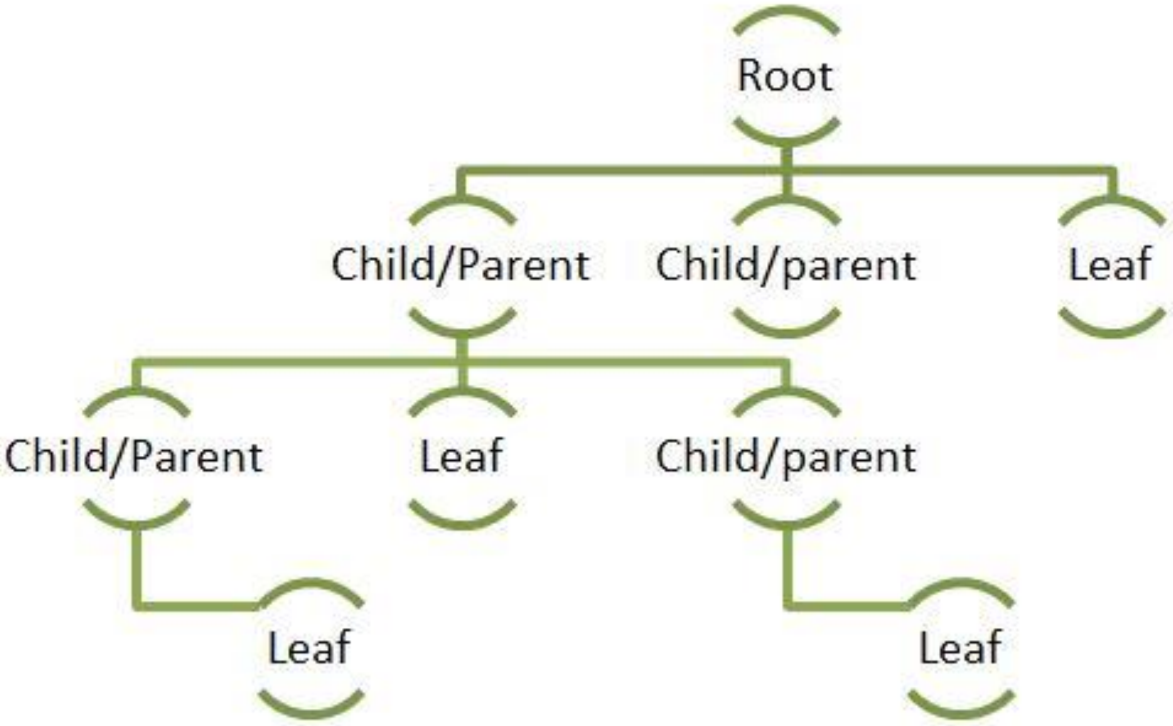
תזכורת–פסאדו קוד

פסאודו קוד תיאור מצומצם ולא רשמי לאלגוריתם של תוכנית מחשב. פסאודו קוד משתמש בקונבנציות של שפות תכנות, אך מיועד לקריאה של בני אדם ולא לקריאה על ידי מחשב.

עצים – הגדרות

- עץ: גרף קשיר ללא מעגלים.
- עלה: קודקוד ללא בנים.
- עומק של קודקוד: מרחקו מהשורש (=אורך המסלול הקצר ביותר מהשורש אליו), העומק של השורש הוא 0.
- גובה העץ : עומק העלה העמוק ביותר.
- צומת פנימי: קודקוד שאיננו עלה.
- רמה:הרמה ה i של העץ היא קבוצת הקודקודים בעומק i בעץ.

עצים – הגדרות

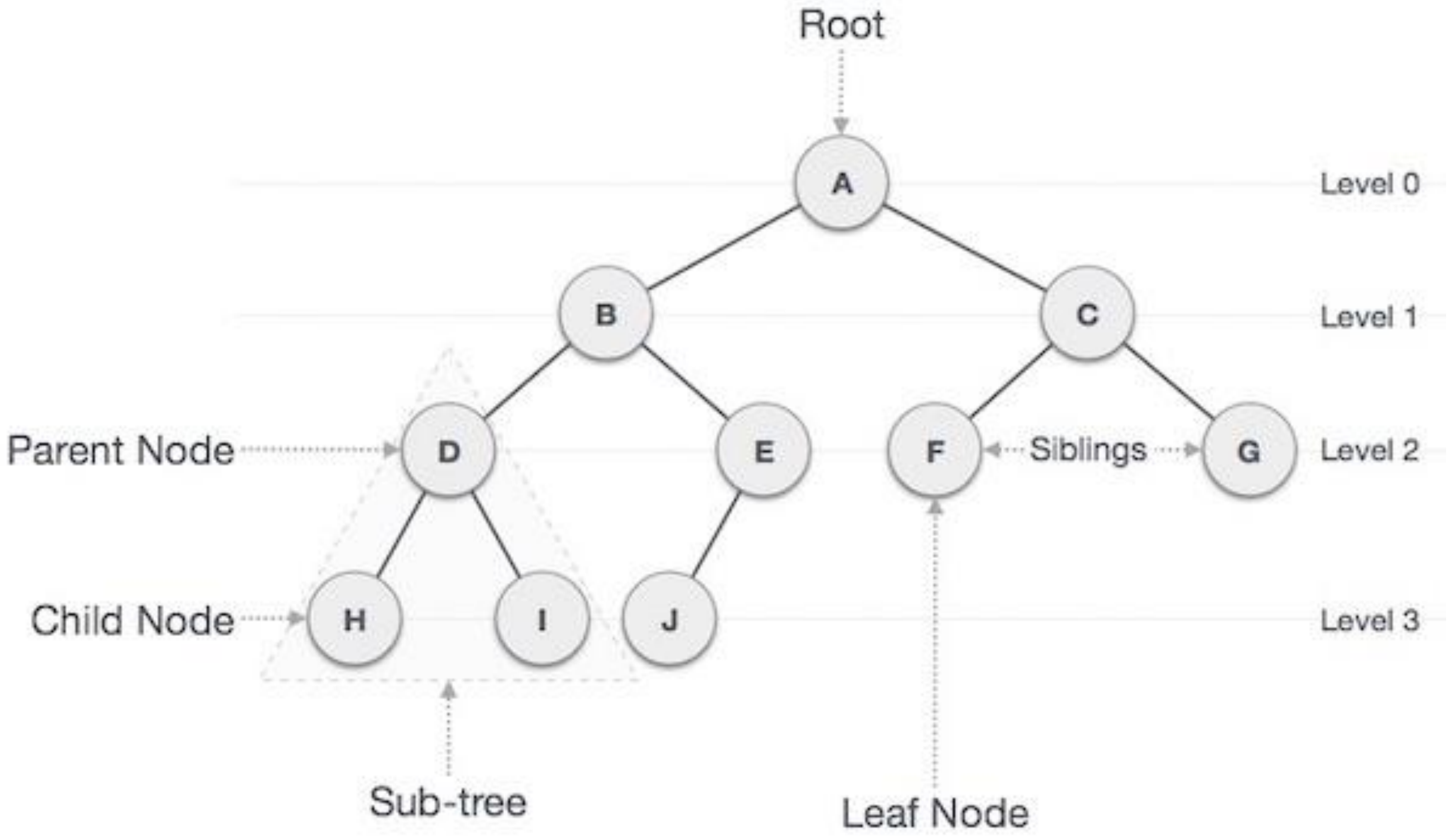


Root
Child/Parent
Child/parent
Leaf
Child/Parent
Leaf
Child/parent
Leaf
Leaf

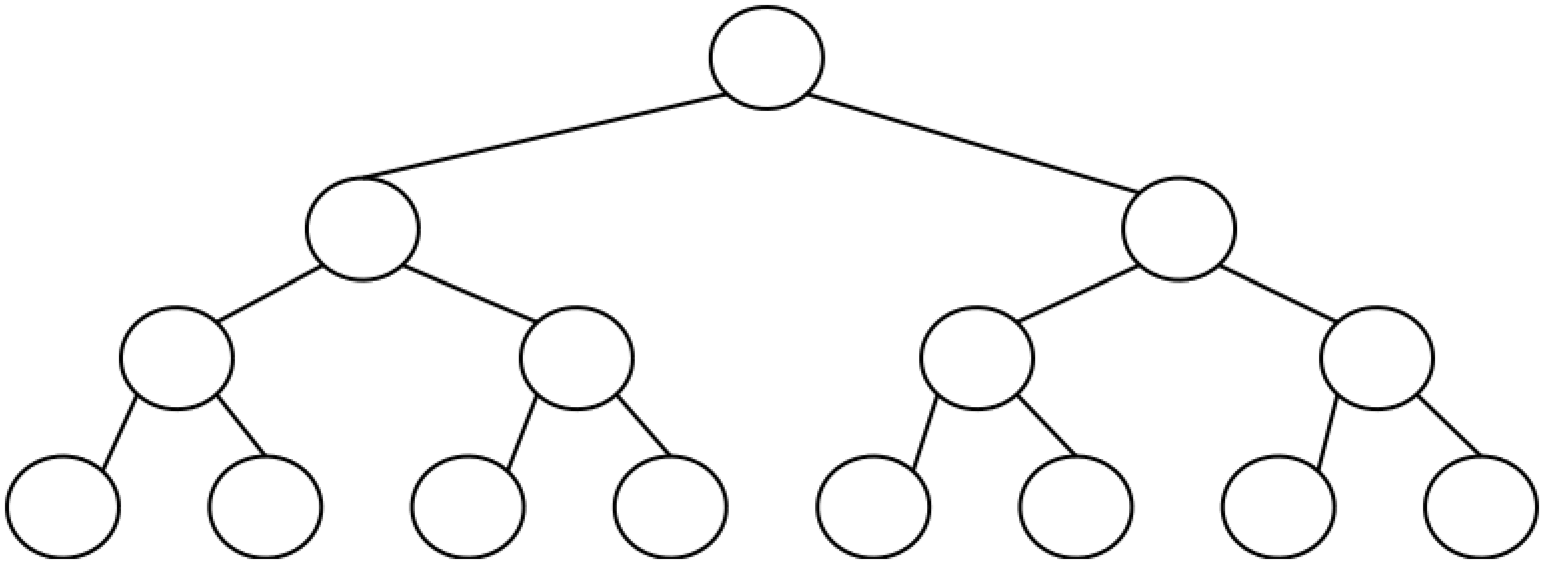
עצים- הגדרת

- עץ בינארי : עץ שבו לכל צומת יש לכל היותר 2 בנים.
- עץ בינארי מלא : עץ בינארי מלא בו לכל צומת פנימי יש בדיוק 2 עלים.
- עץ בינארי שלם: עץ בינארי שבו כל העלים באותו עומק.
- עץ בינארי כמעט שלם : עץ בינארי שכל הרמות פרט לרמה האחרונה מלאות, וכל הצמתים ברמה האחרונה מרוכזים בצד שמאל.

עצים– הגדרות


Root
A
Level 0
B
C
Level 1
Parent Node
D
E
F
Siblings
G
Level 2
Child Node
H
I
J
Level 3
Sub-tree
Leaf Node

דוגמה לעץ בינארי שלם



עצי חיפוש בינאריים

עץ בינארי שבו מתקיים הכלל הבא:

x צומת בעץ חיפוש בינארי.

אם y הוא צומת בתת-העץ השמאלי של x – מתקיים ש-$Y \leq X$.

אם y הוא צומת בתת-העץ הימני של x – מתקיים ש- $Y > X$

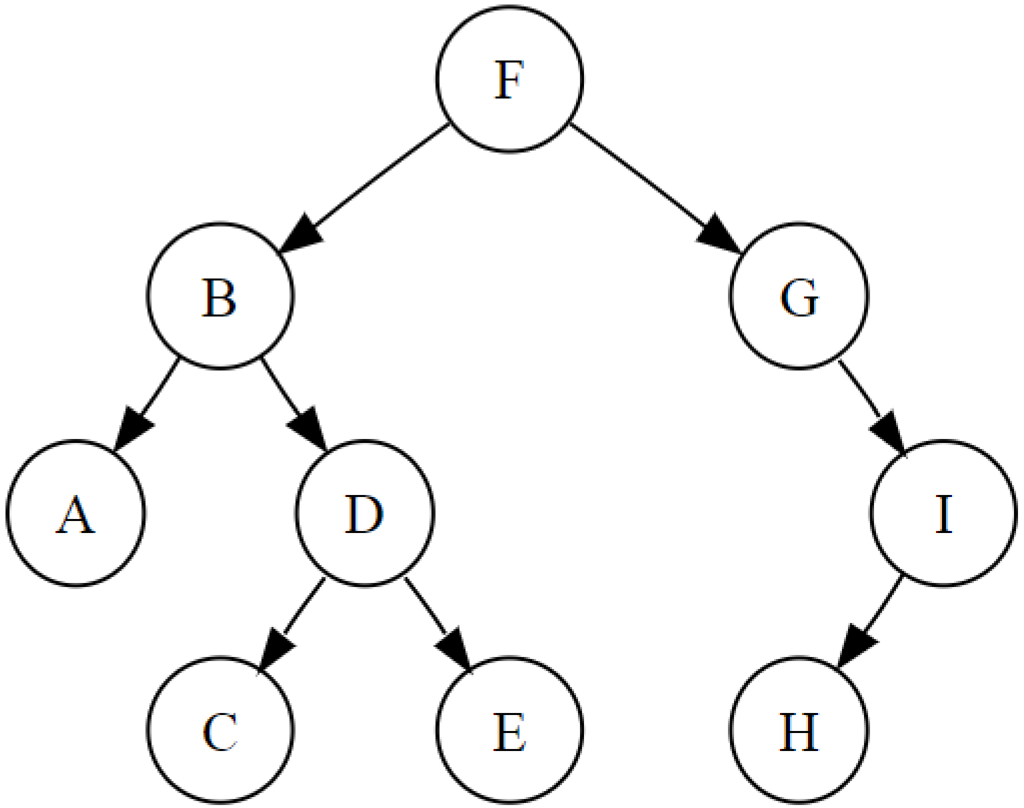
סריקות בעצים בינאריים

Post-order(x): Post-order(Left(x));Post-order(Right(x)); visit(x)

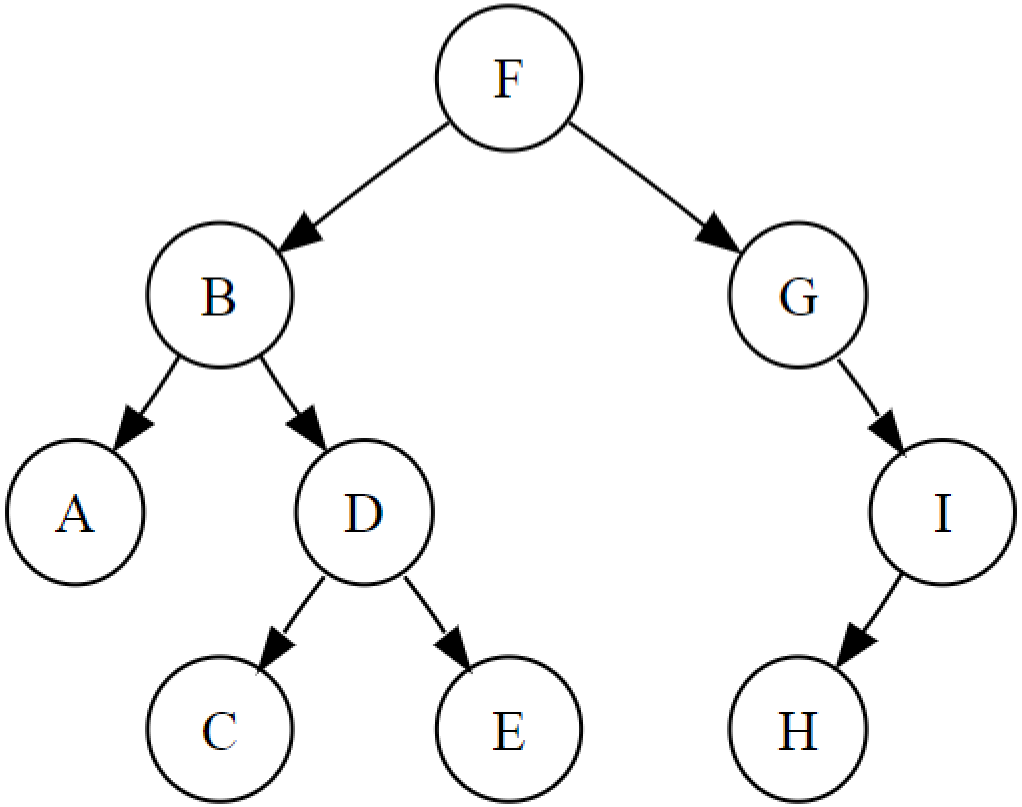
Pre-order(x): visit(x); Pre-order(Left(x)); Pre-order(Right(x))

In-order(x): In-Oder(Left(x)); visit(x); In-Order(Right(x))

דוגמה: (לקוח מוויקפדיה)



דוגמה: (לקוח מוויקפדיה)



Pre-Order: F B A D C E G I H
Post-Order: A C E D B H I G F
In-Order: A B C D E F G H I

תרגיל

יהי T עץ חיפוש בינארי.

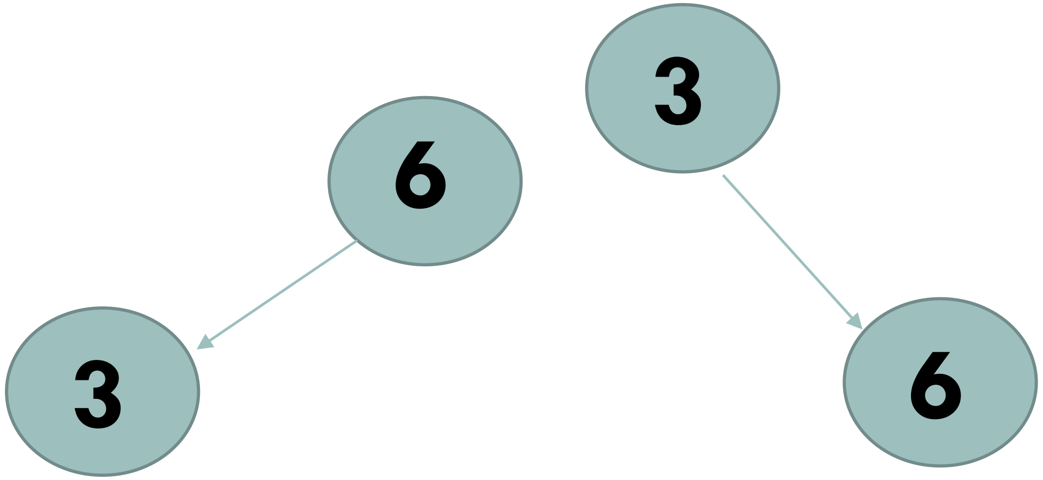
בהינתן סירקת In-order האם ניתן לשחזר את העץ?

תרגיל

יהי T עץ חיפוש בינארי.

בהינתן סירקת In-order האם ניתן לשחזר את העץ?

לא! דוגמה נגדית:



In-Order:3 6

תרגיל

יהי T עץ חיפוש בינארי.

בהינתן סירקת Pre-Order האם ניתן לשחזר את העץ?

תרגיל

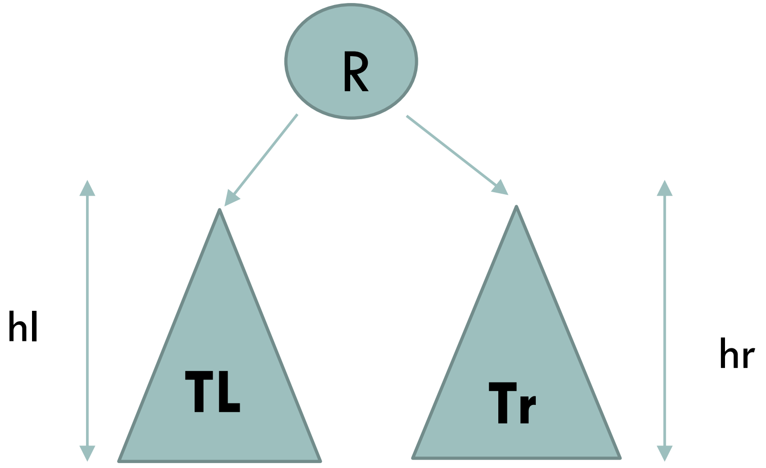
יהי T עץ חיפוש בינארי.

בהינתן סירקת Pre-Order האם ניתן לשחזר את העץ?

שב.

עצי AVL

- עצי חיפוש בינאריים.
- לכל קודקוד, ההפרש בין גבהי תת העץ הימני ותת העץ השמאלי הוא לכל היותר 1.
- נרצה להוסיף לכל צומת מקדם איזון (Balance Factor)
- $BF(v) = h_L(v) – h_R(v)$
- מקדמי איזון תקינים: ‎–1,1,0

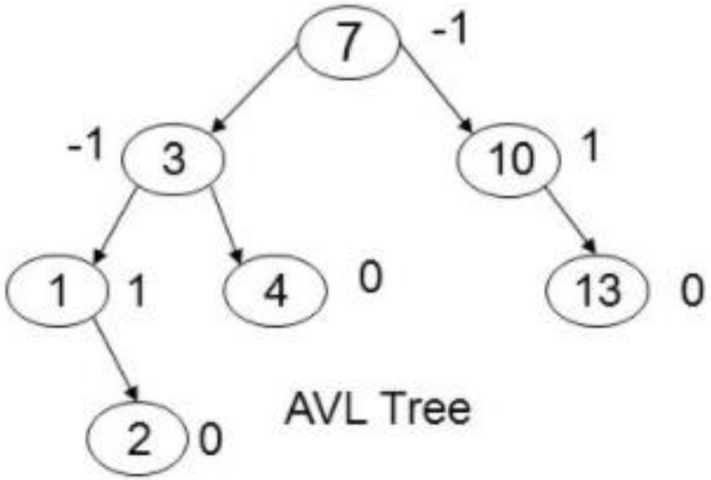


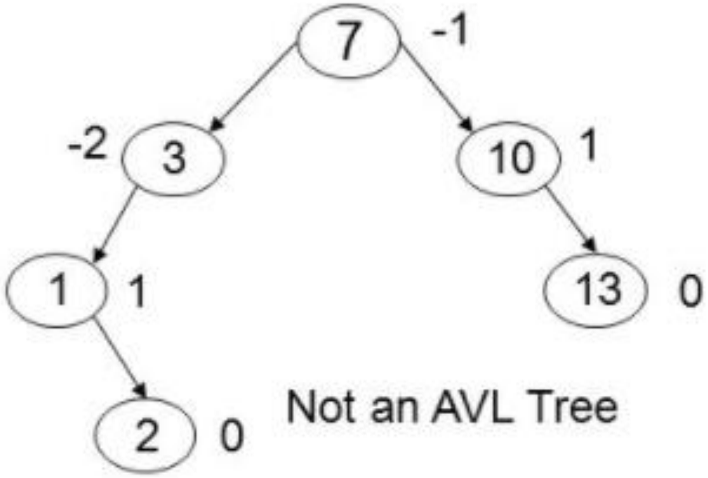
|hl-hr|<=1

עץ מאוזן

משפחה של עצים בינאריים תקרא מאוזנת אם כל עץ במשפחה המכיל n קודקודים גובהו הוא $O(\log n)$.

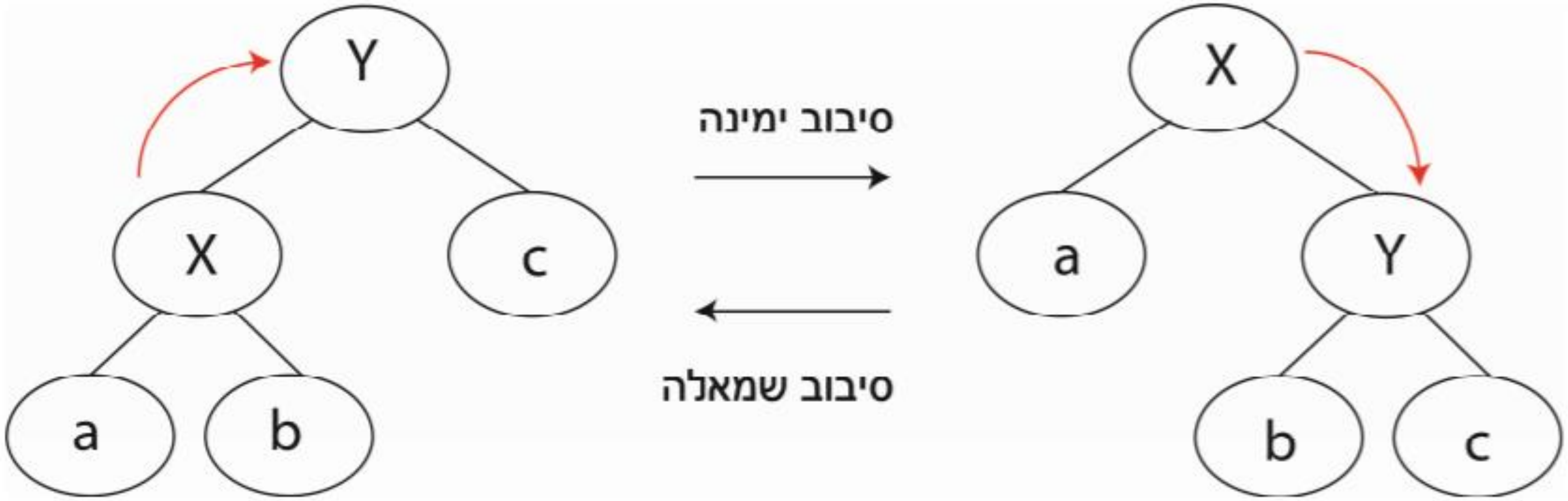
דוגמאות





לאחר הוספת/מחיקת צומת, ייתכן שיהיה צורך לאזן מחדש. לכן נבצע סיבובים:
(הצמתים בהם ייתכן שהופר האיזון הם לאורך מסלול ההוספה/מחיקה)

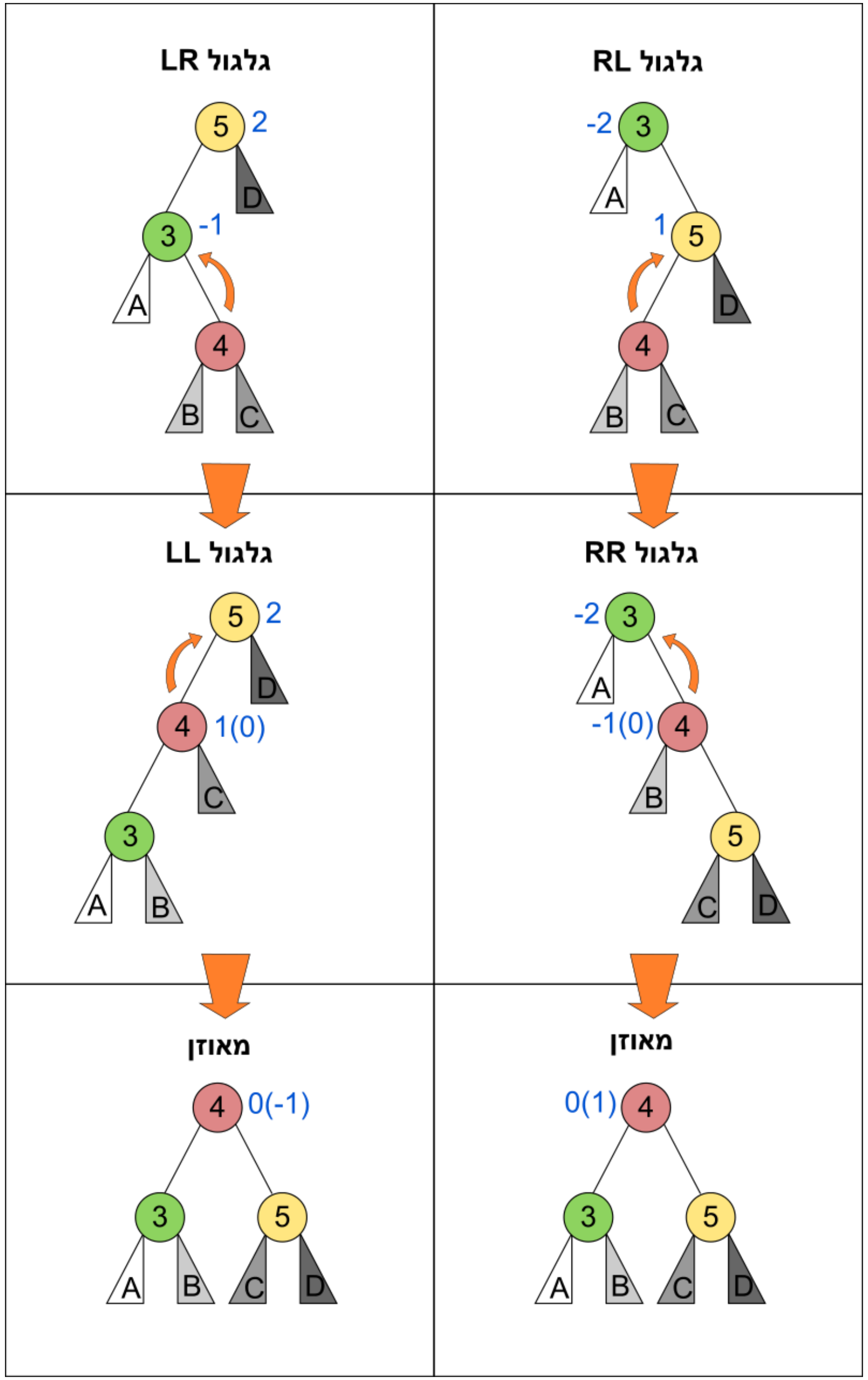
סיבוב פשוט:



סוג ומספר הסיבובים תלויים בצורה שבה הופר האיזון – עדכון מקדמי האיזון מתבצע לאחר הוספת/מחיקת צומת כאשר עולים חזרה במעלה העץ. - אם באים משמאל בעץ – מוסיפים 1+
- אם באים מימין בעץ – מחסירים 1–

סה"כ בכל מצב :
|BF(v)| <=2

דוגמה-וויקפדיה



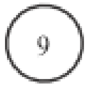
תרגיל

Show the AVL tree that results after **each** of the integer keys $9, 27, 50, 15, 2, 21$, and 36 are inserted, in that order, into an initially empty AVL tree. Clearly show the tree that results after each insertion, and make clear any rotations that must be performed.

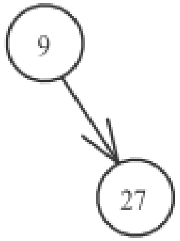
תרגיל

Show the AVL tree that results after **each** of the integer keys 9, 27, 50, 15, 2, 21, and 36 are inserted, in that order, into an initially empty AVL tree. Clearly show the tree that results after each insertion, and make clear any rotations that must be performed.

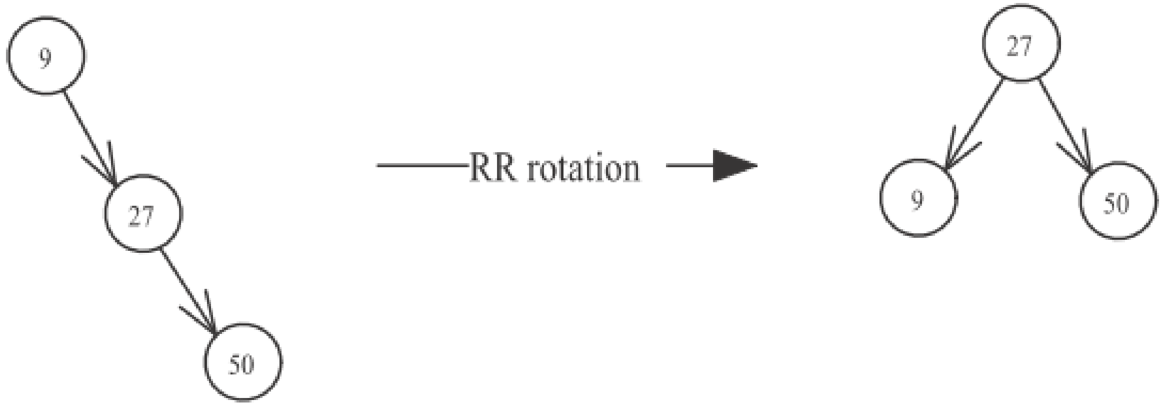
Insert 9



Insert 27

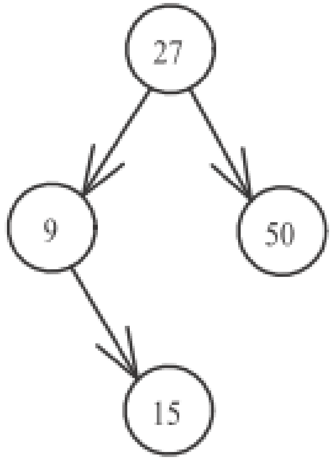


Insert 50

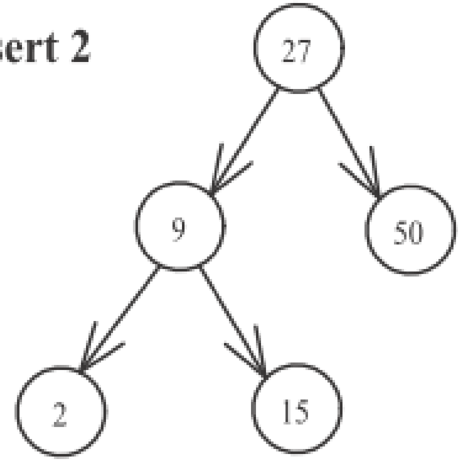


Shov
into
that

Insert 15

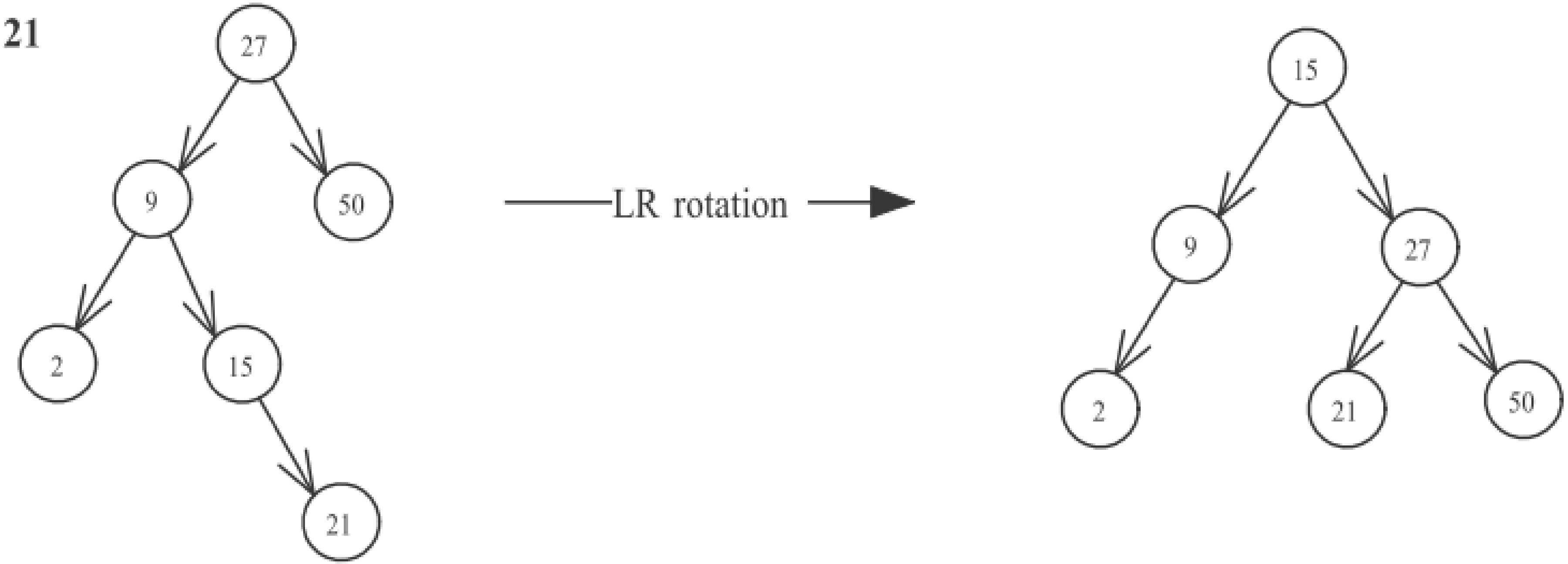


Insert 2



Insert 36

Insert 21



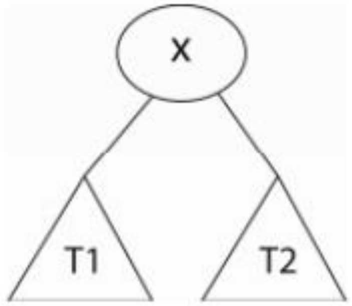
תרגיל

<u>תרגיל:</u>

נתונים 2 עצי AVL – T_1, T_2 וערך x כך ש-$T_1 < x < T_2$ (כלומר כל הערכים ב-T_1 קטנים ממש מ-x וכל הערכים ב-T_2 גדולים ממש מ-x). אחד את T_1 ו-T_2 לעץ AVL בצורה יעילה.

פתרון

- חשב את גובהם של 2 העצים $h(T_1), h(T_2)$. $O(logn_1+ logn_2)$
- אם $h(T_1)=h(T_2)$ החזר:

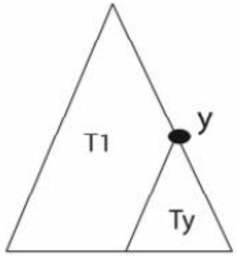


המשך פתרון

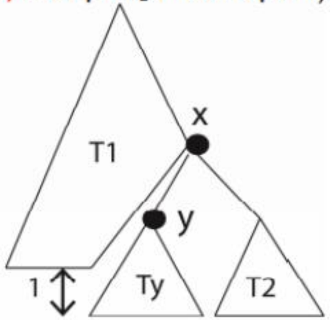
- אחרת:

אם $h(T_1)>h(T_2)$:

1) חפש את תת העץ הימני של T_1 שגובהו $h(T_2)$. נקרא לתת-העץ הזה T_y ושורשו y.
$O(logn_1- logn_2)$



2) החלף את y ב-x, והוסף את T_y כבן שמאלי ו-T_2 כבן ימני. $O(1)$



3) חזור מ-x עד לשורש ותקן לפי הצורך. $O(logn_1)$

אם $h(T_1)<h(T_2)$: כנ"ל הפוך.

סיבוכיות $O(logn_1+ logn_2)=O(log(n_1*n_2))$

עצי 2-3

- כל העלים באותה רמה.
- כל הערכים בעלים.
- בצמתים הפנימיים יש אינדקסים.
- לכל צומת יש 2 או 3 בנים.
- מספר העלים מקיים $2^h \leq L \leq 3^h$ (h- גובה העץ)
- גובה העץ : $\log_2 L \leq h \leq \log_3 L \leftarrow h = \Theta(\log L)$.
-

עצי 2-3

בצומת עם 2 בנים: יש אינדקס יחיד שגדול ממש מהערך המקסימלי בתת העץ השמאלי וקטן-שווה מהערך המינימלי בתת-העץ הימני.

בצומת עם 3 בנים: האינדקס הראשון גדול ממש מהערכים בתת-העץ השמאלי, קטן-שווה מהערכים בתת-העץ האמצעי, ואינדקס שני גדול ממש מהערכים בתת-העץ האמצעי וקטן- שווה מהערכים בתת-העץ הימני.

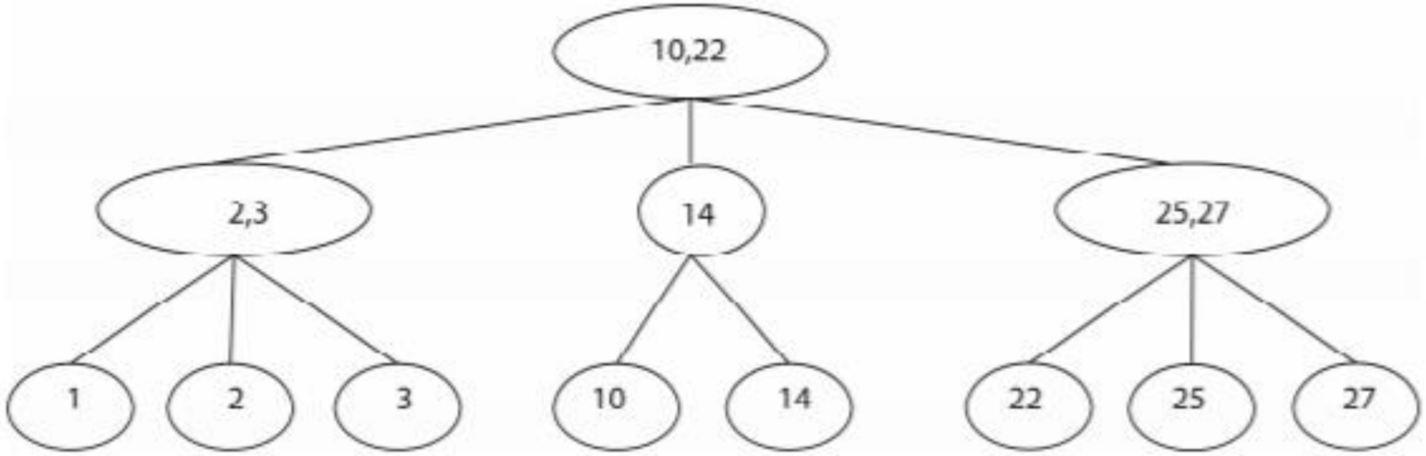
פעולות בעצי 2–3

פעולות בעץ 2-3:

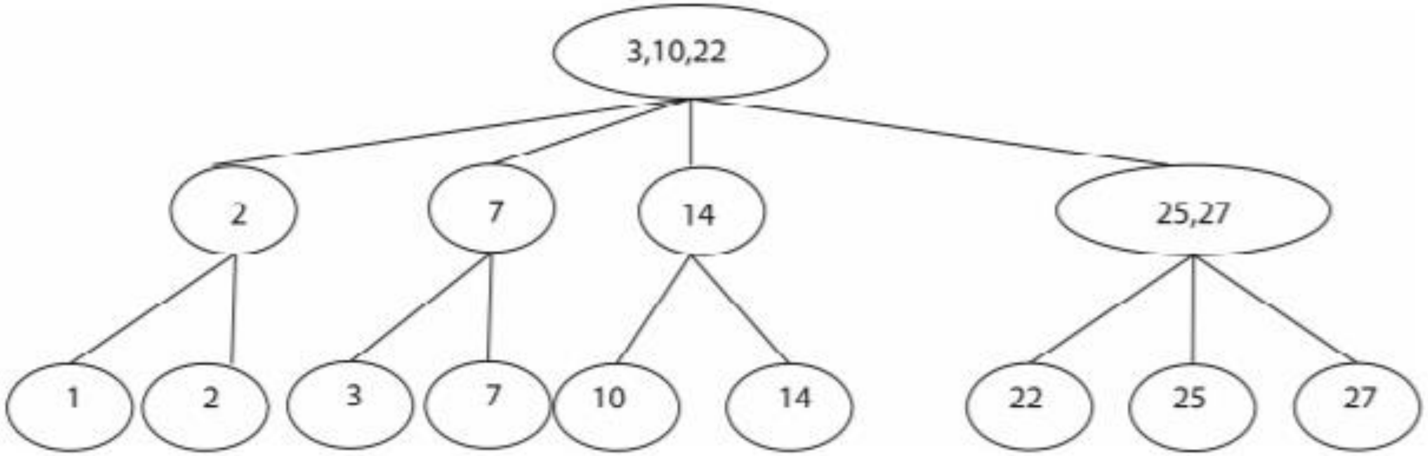
1) חיפוש - לפי הכללים הנ"ל
2) הוספה: חפש את הערך. הוסף אותו לצומת הפנימי האחרון אליו הגעת והוסף אינדקס
   2.1. אם יש 4 בנים – פצל והעלה אינדקס אמצעי
   2.2. חזור ל2.1 עבור האב
3) מחיקה: חפש את הערך. מחק את הערך ואת האינדקס משמאלו (או הכי שמאלי).
   3.1. אם נשארו 2 בנים – סיים.
   3.2. אחרת – אם לאב יש אח עם 3 בנים – השאל בן וסיים.
   3.3. אחרת – אחד את האב עם אח שלו ועדכן אינדקסים.
   3.4. חזור ל3.1 עבור האב.

דוגמא

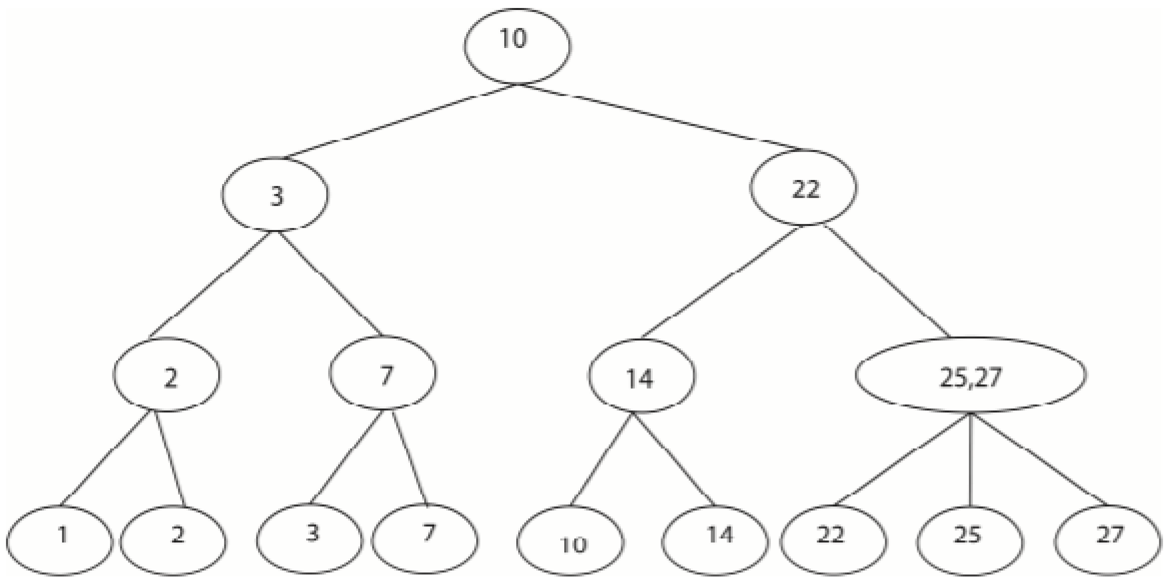
הוסף 7:



המשך

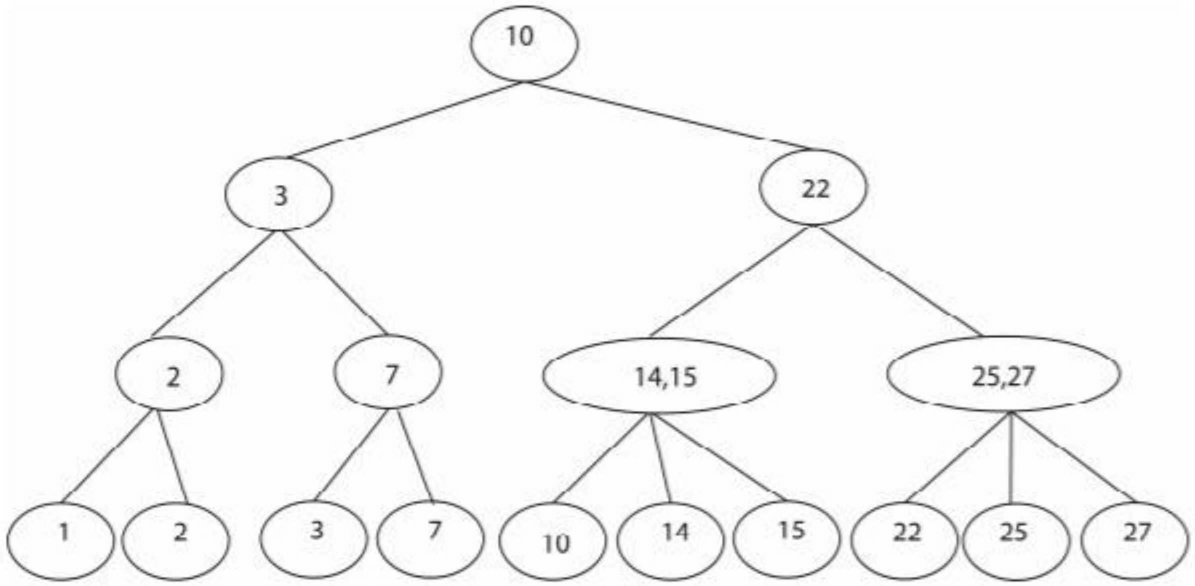


המשך



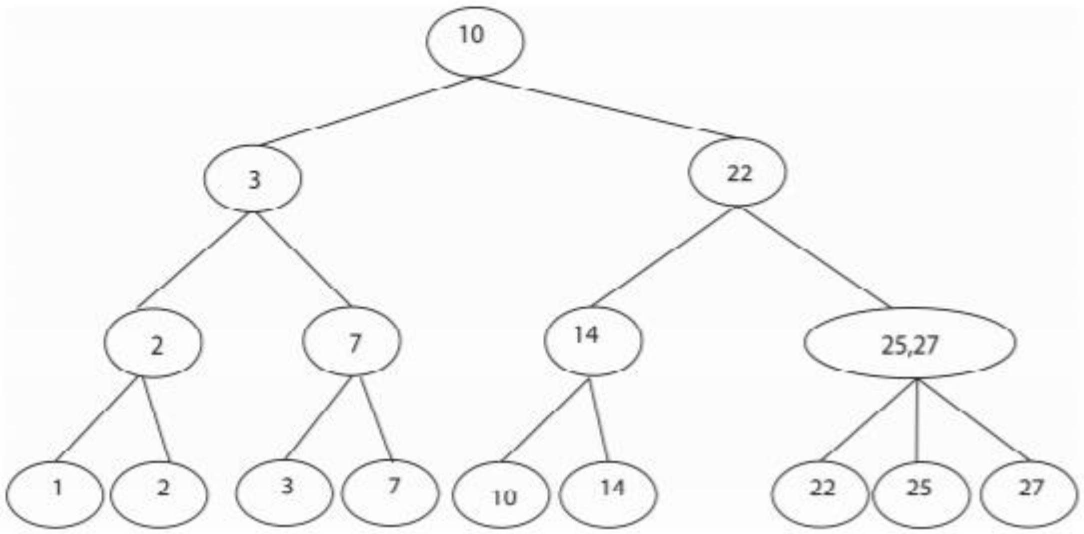
המשך

הוסף 15:



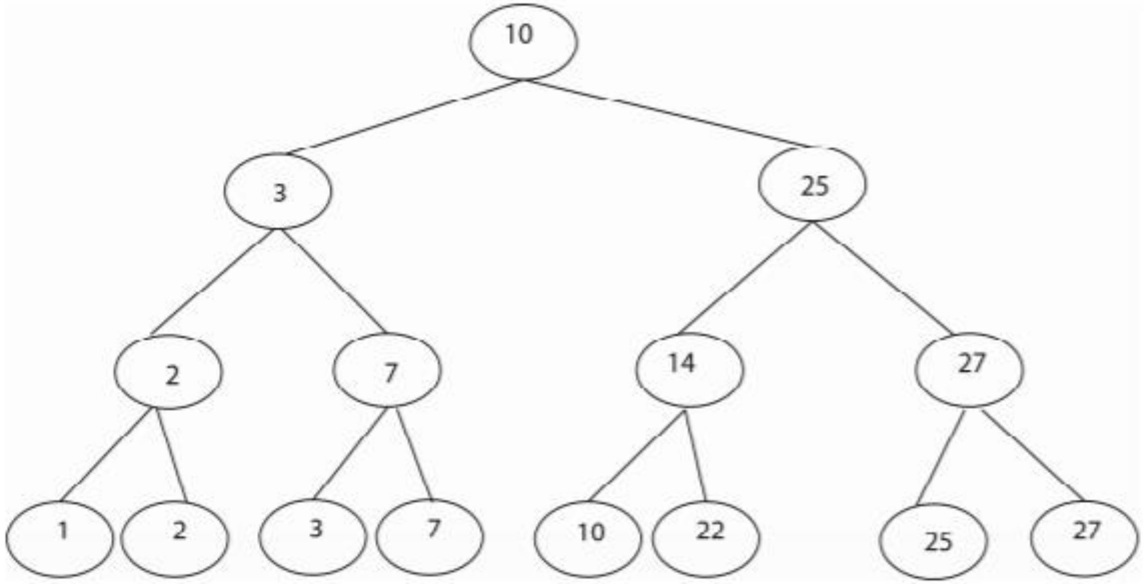
המשך

מחק 15:

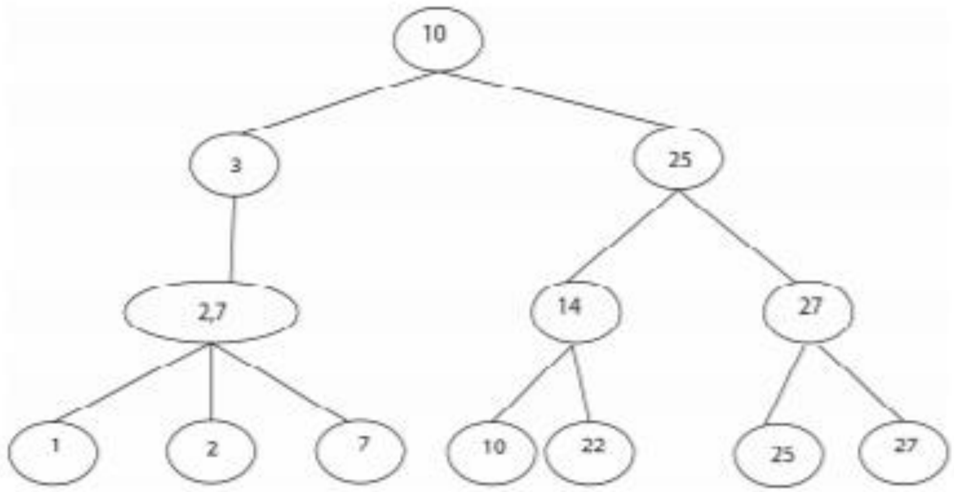


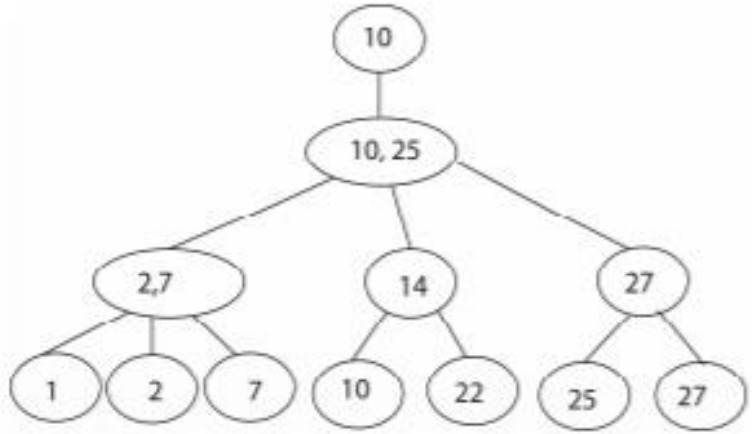
המשך

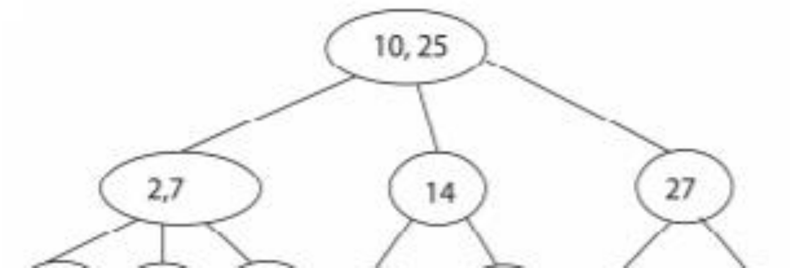
מחק 14:



המשך


10
3
25
2,7
14
27
1
2
7
10
22
25
27


10
10, 25
2,7
14
27
1
2
7
10
22
25
27


10, 25
2,7
14
27

תרגיל

תרגיל

כיצד תממשו מבנה נתונים שתומך ב:

- הכנסת זוגות מספרים (a, b) ב$O(\log n)$
- הוצאת זוגות מספרים (a, b) ב$O(\log n)$
- חיפוש לפי הקואורדינטה הראשונה ב$O(\log n)$
- חיפוש לפי הקואורדינטה השנייה ב$O(\log n)$
- מעבר על הזוגות ממוינים לפי הקואו' הראשונה\השניה ב$O(n)$

תרגיל

כיצד תממשו מבנה נתונים שתומך ב:

- הכנסת זוגות מספרים (a, b) ב$O(\log n)$
- הוצאת זוגות מספרים (a, b) ב$O(\log n)$
- חיפוש לפי הקואורדינטה הראשונה ב$O(\log n)$
- חיפוש לפי הקואורדינטה השנייה ב$O(\log n)$
- מעבר על הזוגות ממוינים לפי הקוא' הראשונה\השניה ב$O(n)$

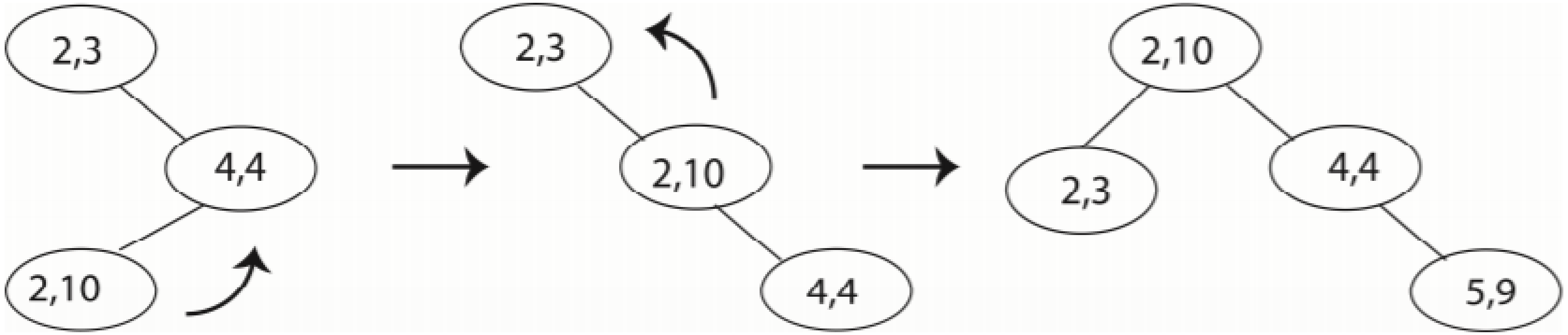
פתרון

נחזיק שני עצי חיפוש (AVL או $2-3$).
עץ אחד יהיה ממוין לפי הקואורדינטה השמאלית והשני לפי הימנית.

- הכנסה = הכנסה לשני העצים
- הסרה = הסרה משני העצים
- חיפוש - חיפוש בעץ המתאים לפי הקואורדינטה המבוקשת
- מעבר על האיברים ממויינים לפי קוא' שמאלית\ימנית - מעבר סדרתי על העץ המתאים.

דוגמה

דוגמא : (לפי הקואורדינטה הראשונה):
הוסף את הזוגות הבאים:
(2,3), (4,4), (2,10),(5,9).



מעבר סדרתי: *(2,3), (2,10), (4,4), (5,9)*

מיונים

מיוני השוואה
המקרה הכי טוב – $\Theta(n \log n)$

מיוני לא השוואה–
מיונים עם הנחות
נוספות על הקלט

מיון נקרא מיון יציב אם הוא שומר על הסדר של הנתונים לאחר המיון גם כשיש שני נתונים זהים.

מיוני השוואה

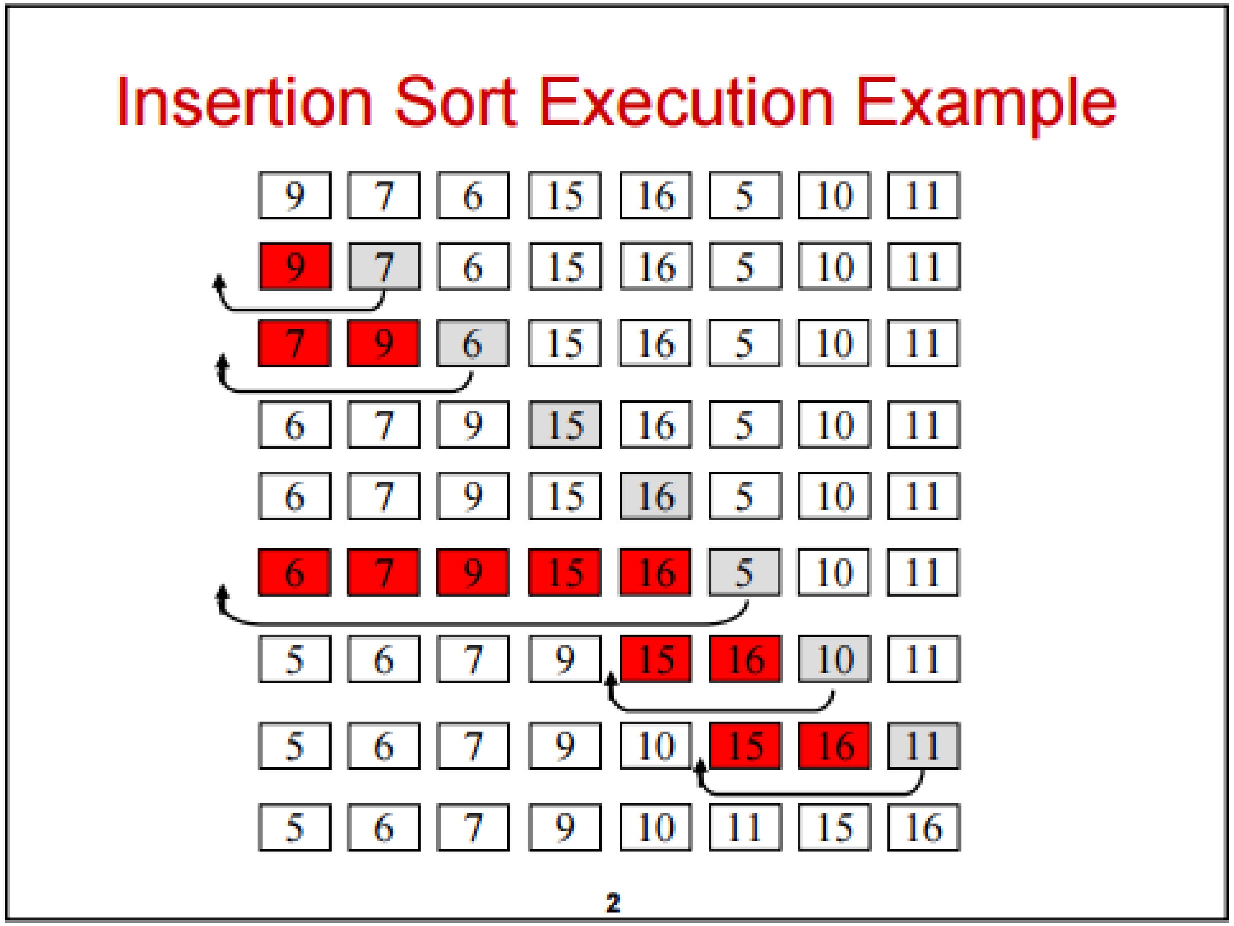
- מיון בועות $\Theta\left(n^2\right)$
- מיון Selection-sort $\Theta\left(n^2\right)$
- מיון Insertion-Sort $\Theta\left(n^2\right)$
- מיון Quick-sort $\Theta\left(n\log\left(n\right)\right)$
- מיון Merge-sort $\Theta\left(n\log\left(n\right)\right)$
- מיון heap-sort $\Theta\left(n\log\left(n\right)\right)$

INSERTION-SORT

אופן פעולה: מעבר על אברי המערך וכל איבר מועבר אחורה כל עוד קיימים איברים לפניו הגדולים ממנו.

פסאדו קוד: (וויקיפדיה)

```
for j ←1 to length(A)-1
    key ← A[ j ]
    > A[ j ] is added in the sorted sequence A[0, .. j-1]
    i ← j - 1
    while i >= 0 and A [ i ] > key
        A[ i + 1 ] ← A[ i ]
        i ← i - 1
    A [i + 1] ← key
```


Insertion Sort Execution Example
9 7 6 15 16 5 10 11
9 7 6 15 16 5 10 11
7 9 6 15 16 5 10 11
6 7 9 15 16 5 10 11
6 7 9 15 16 5 10 11
6 7 9 15 16 5 10 11
5 6 7 9 15 16 10 11
5 6 7 9 10 15 16 11
5 6 7 9 10 11 15 16
2

[קישור]

INSERTION - SORT

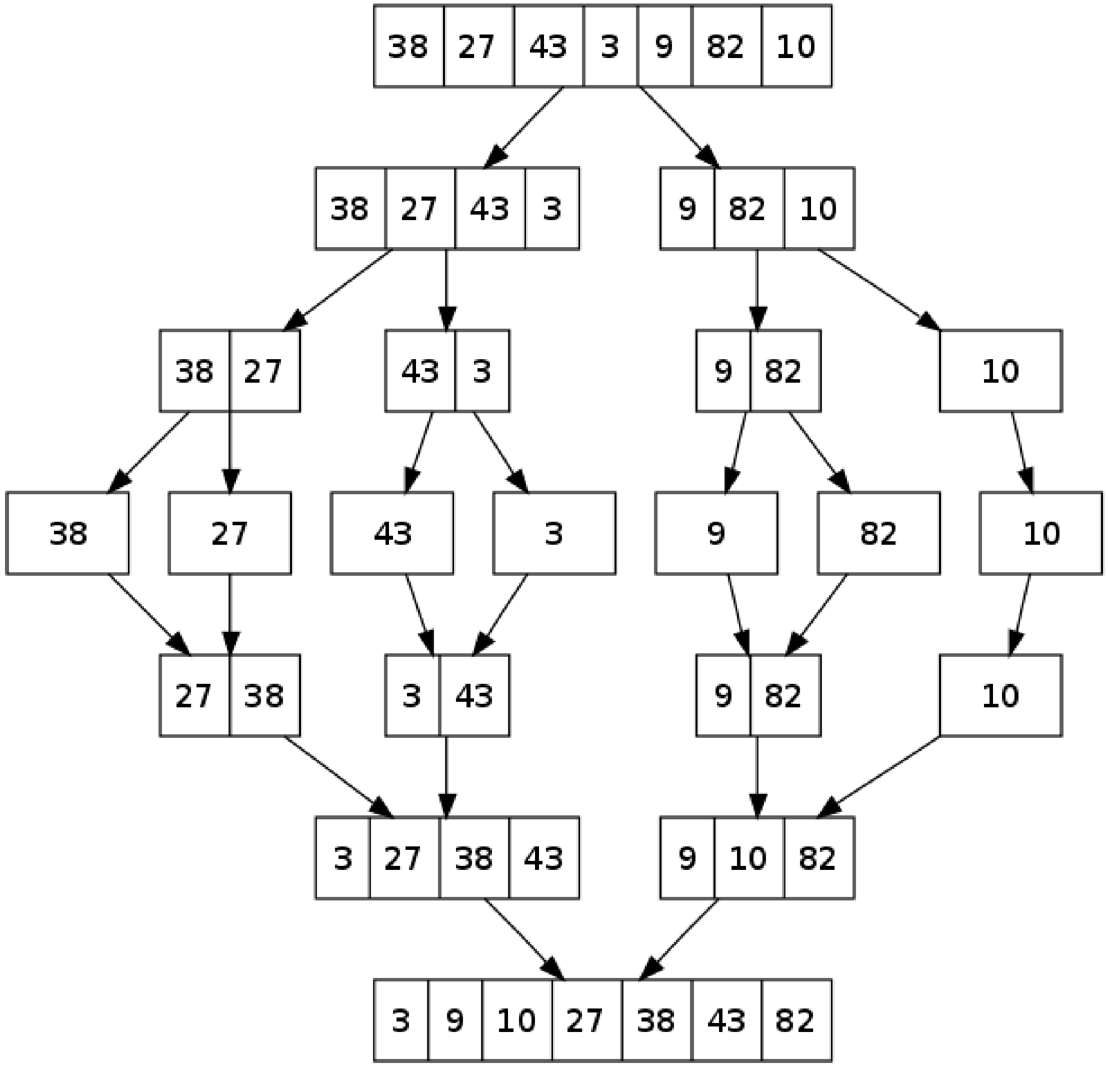
- מימוש פשוט
- יעיל עבור מערך (כמעט)ממוין $O(n)$ ועבור מערכים קטנים.במקרה של מערך כמעט ממוין – $O(nk)$
- לא דורש זיכרון נוסף $O(1)$
- יציב (תלוי מימוש)

MERGE-SORT

```
MergeSort (A)
        n ← length(A)
        if n > 1
                mid ← n/2
                A1 ← A [0 … mid-1]
                A2 ← A [mid … n-1]
                MergeSort(A1)
                MergeSort(A2)
                Merge(A1, A2, A)
```

```
Merge(A1, A2, A)
        ind1 = 0
        ind2 = 0
        for i ← 0 to i < length(A) ; i++
                if  ind2 = length(A2)
                        A[i] ← A1[ind1++]
                else if ind1 = length(A1)
                        A[i] ← A2[ind2++]
                else
                        if  A1[ind1] < A2[ind2]
                                A[i] ← A1[ind1++]
                        else
                                A[i] ← A2[ind2++]
```

MERGE-SORT



מקור:וויקיפדיה

:MergeSort אנליזת זמן ריצה עבור

MergeSort(A)	
n ← length(A)	1
if n > 1	1
mid ← n/2	1
A1 ← A [0 … mid-1]	n/2
A2 ← A [mid … n-1]	n/2
MergeSort(A1)	T(n/2)
MergeSort(A2)	T(n/2)
Merge(A1, A2, A)	n

:לכן את נוסחת חישוב זמן הריצה ניתן לרשום באופן רקורסיבי

$$T(n)=\begin{cases}\Theta(1) & n=1\\ 2T(n/2)+\Theta(n) & n>1\end{cases}$$

MERGE-SORT

- תמיד $O(n*log(n))$
- דרוש אקסטרה זיכרון – $O(n)$
- הפרד ומשול

מיון ערימה

1. הכנס את האיברים לערימה.
2. כל עוד הערימה לא ריקה:
   הוצא איבר מהערימה והוסף למערך הסופי.

מיון ערימה

סיבוכיות זמן: $O(n\log(n))$

סיבוכיות מקום: $O(n)$