שאלה 1 חלק א' תיאורטי

השיטה Build-Max-Heap שלמדתם בכתה, יכולה להיות ממומשת גם בדרך אחרת – ע"י שימוש ב-Max-Heap-Insert.

נתונה השיטה הבאה:

BUILD-MAX-HEAP-BY-INSERTION*(A)*

1. *heap-size*[*A*] ← 1
2. **for** *i* ← 2 **to** *length*[*A*]
3. **do** MAX-HEAP-INSERT*(A,A[i])*

א. בהינתן מערך כלשהו A של מספרים, האם השיטות Build-Max-Heap(A) ו-Build-Max-Heap-By-Insertion(A) יבנו ערימות מקסימום זהות? אם כן הוכיחו, אחרת ספקו דוגמה נגדית.

ב. הראו כי במקרה הגרוע Build-Max-Heap-By-Insertion תרוץ בזמן **O(nlogn)** (בבניית ערימה בת n איברים).

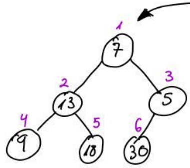
נבנה מערך לדוגמא:

A

7	13	5	9	18	30
1	2	3	4	5	6

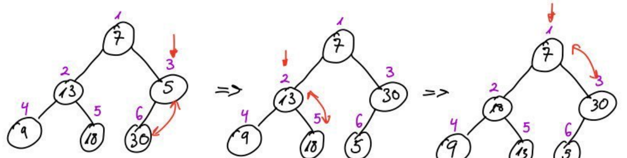
→ index

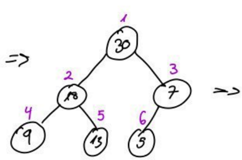
ערימה נראית כך:



נפעיל את הפונקציה שלמדנו בכיתה

Bulid-Max-HeaP(A)





ערימה החדשה אחרי שהפעלנו

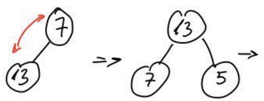
Bulid-Max-HeaP(A)

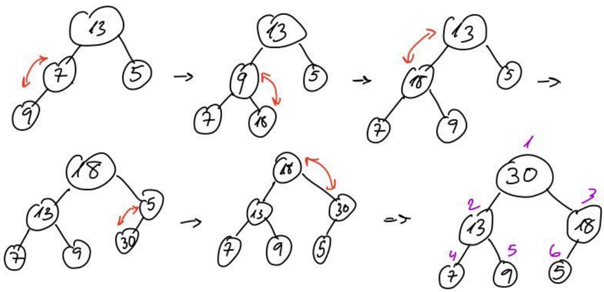
A

30	18	7	9	13	5
1	2	3	4	5	6

Max-Heap-insert ← מכניסה איבר חדש לערימה!
מקבלת מערך ומפתח! ניקח את המערך שיצרנו ונשתמש גם
כאן ונראה איזה ערימה זה ייצר

A	7	13	5	9	18	30
	1	2	3	4	5	6





A	30	13	18	7	9	5
	1	2	3	4	5	6

$\neq$

A	30	18	7	9	13	5
	1	2	3	4	5	6

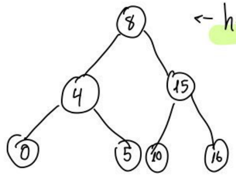
לכן ניתן לראות שני ערימות שונות!!!

ב) ניתן לראות שבפונקציה Build-Max-Heap-By-insertion
יש לולאת for כך במקרה הגרוע של זה היא תרוץ n
פעמים! ובתוך הלולאה for אנחנו מבצעים קריאה לפונקציה
Max-Heap-Insert שזה יעלה שזמן ריצה שלו הוא log(n)
לכן זמן הריצה של פונקציה Build-Max-Heap-By-insertion
$O(n\log n)$

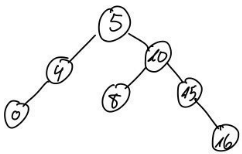
שאלה 3

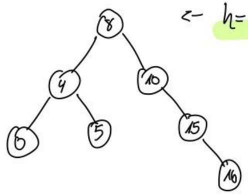
שרטטו עצי חיפוש בינאריים בגבהים 2, 3, 4, 5 עבור קבוצת המפתחות:
<0, 2, 5, 8, 10,15, 16>

במידה ויש מספר אפשרויות עבור גובה מסוים, ציירו אחת מהן.

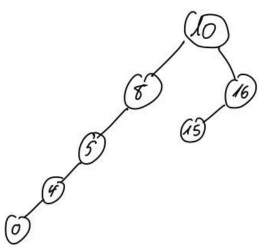


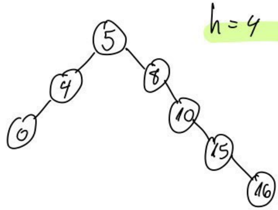
<- h=2 תקין





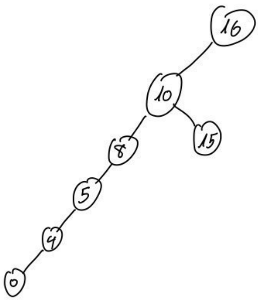
<- h=3 תקין

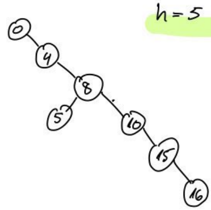




h = 4 תקין

נתון $h=5$





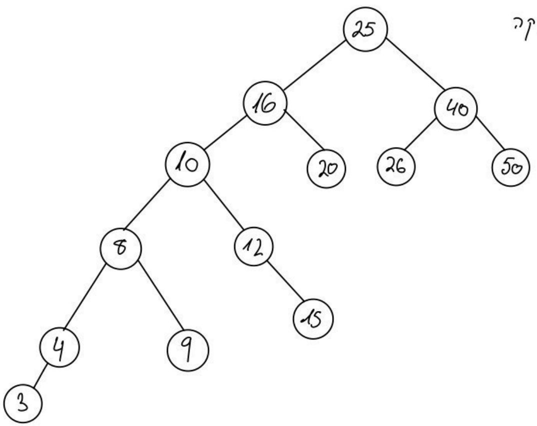
<u>שאלה 5</u>

בנו עץ חיפוש בינארי (BST) על ידי הכנסת הצמתים הבאים החל מעץ ריק (סדר הכנסת צמתים משמאל לימין):

25, 16, 10, 12, 8, 4, 9, 40, 15, 20, 3, 26, 50

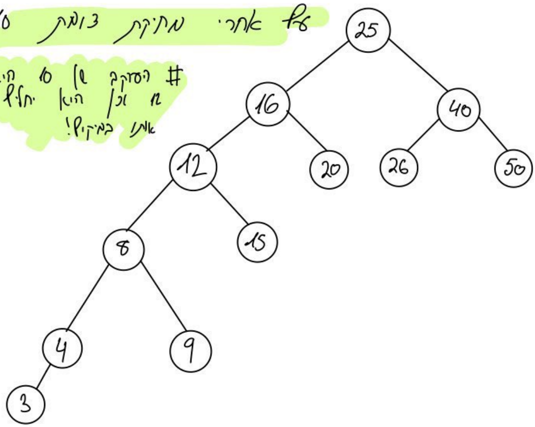
שרטטו עץ אחרי מחיקת צומת 10

עץ לפני מחיקה
צומת 10



עץ אחרי מחיקת צומת 10

העוקב של 10 היא 12 ולכן הוא יחליף אותו במיקום!



<u>שאלה 4</u>

כתבו גרסה רקורסיבית (פסאודו-קוד) של פונקציה TREE-INSERT (הכנסת איבר חדש לעץ חיפוש בינארי).

```
TREE_INSERT(node , z)

If (node = NULL)
     node <- z
elseif(z.key < node.key)
     TREE_INSERT(node.left. z)
else TREE_INSERT(node.right, z)
```

הוכיחו באינדוקציה שבערימה בת n איברים יש $\left\lceil \frac{n}{2} \right\rceil$ עלים.

הוכחה באינדוקציה שבערימה בת n איברים יש $\left\lceil \frac{n}{2} \right\rceil$ עלים

<u>בסיס</u>: נבדוק את מקרה הבסיס שלנו כאשר $n=1$ (צומת אחת)

אז $\left\lceil \frac{1}{2} \right\rceil = 1$ לכן מקרה מתקיים! (צומת אחת היא גם עלה)

<u>הנחת האינדוקציה:</u>

נניח שהטענה נכונה לכל n!! $\left\lceil \frac{n}{2} \right\rceil$

<u>צעד האינדוקציה</u>: נוכיח שהטענה נכונה עבור $n+1$ $\left\lceil \frac{n+1}{2} \right\rceil$

נחלק ל-2 מקרים:

<u>מקרה I</u>: n זוגי לכן לפי הנחת האינדוקציה מתקיים $\left\lceil \frac{n}{2} \right\rceil$
כאשר נוסיף איבר חדש לעץ וערימה מיוצגת על ידי עץ וכל הצמתים
מרוכזים בצד שמאל האיבר החדש ייכנס לאינדקס $2i+1$ ולכן בהכרח
יהיה עלה ואז מתקיים $\left\lceil \frac{n+1}{2} \right\rceil$

<u>מקרה II</u>: n אי זוגי, שוב לפי הנחת האינדוקציה מתקיים $\left\lceil \frac{n}{2} \right\rceil$
ומתקיים שעלים הם על מלא!! כי הרמות שלו
מלאים דומות יש 0 או 2 בנים! ולפי המשפט

עלים   צמתים פנימיים   אחד

$$\begin{cases} l(T) = m(T) + 1 \\ n = m(T) + l(T) \end{cases} \Rightarrow$$

סה"כ צמתים!

הצבה

$$\Rightarrow m(T) = l(T) - 1 \Rightarrow n = l(T) - 1 + l(T) \Rightarrow$$

$$n = 2l(T) - 1 \Rightarrow \boxed{l(T) = \frac{n+1}{2}} \longrightarrow$$

כמות עלים

מ.ש.ל.