

# עצים ועצי חיפוש

## עצים מכוונים

מקור הוא צומת שאף קשת אינה מצביעה אליו.

עץ מכוון הוא גרף מכוון ללא מעגלים (בגרף התשתית שלו) ואשר לו מקור אחד בלבד הנקרא שורש.

דוגמאות

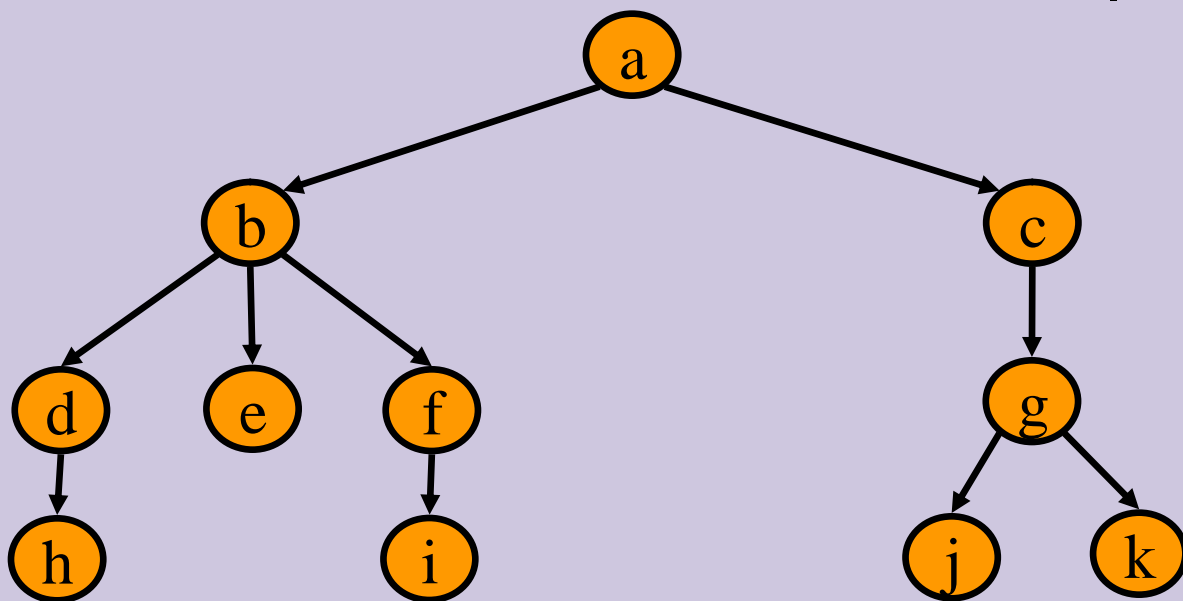
f בן של b

b אב של e

הגדרות

v בן של u אם קיימת קשת מצומת u לצומת v.

u אב של v אם v בן של u.



## עצים מכוונים

הגדרות

דוגמאות

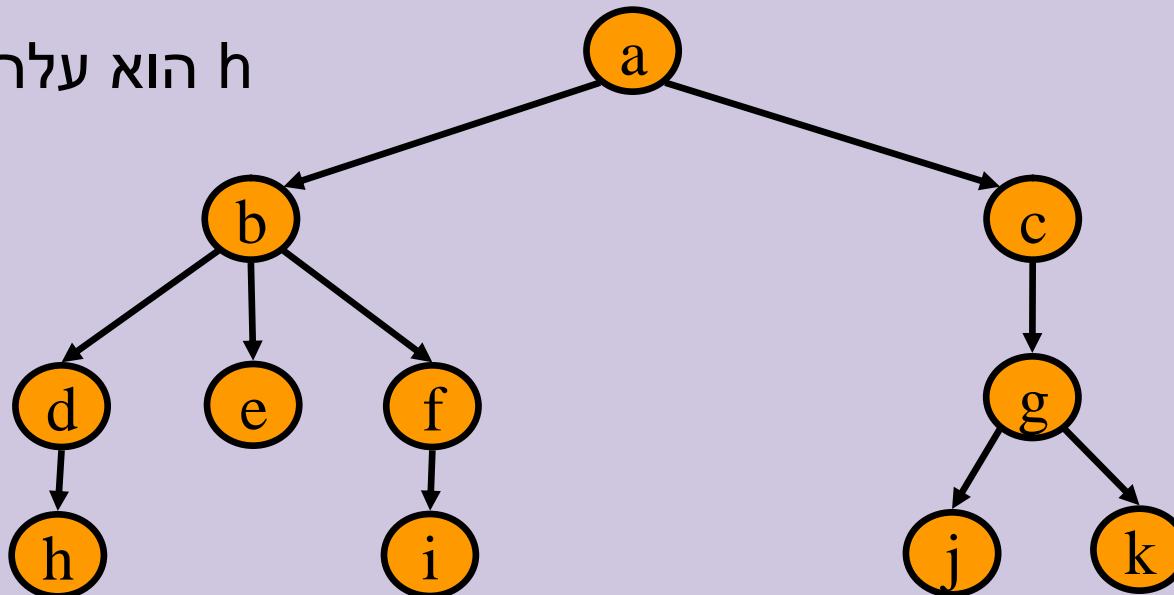
g צאצא של a

b אב-קדמון של h

תת-עץ של G ששורשו g מכיל 3 צמתים ושתים קשתות.

דרגת a היא 2.

h הוא עלה.



v צאצא של u אם קיים מסלול מכוון מצומת u ל- v.

u אב-קדמון של v אם v צאצא של u.

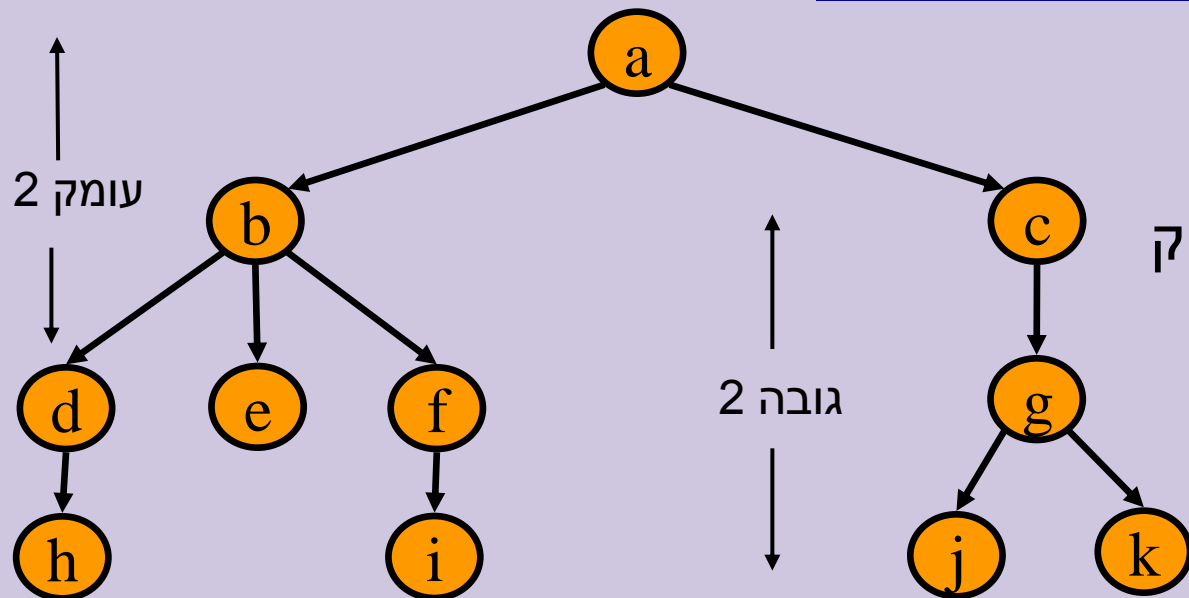
תת-עץ של G ששורשו v הוא עץ מכוון שצמתיו הם v עצמו וכל הצאצאים של v, והקשתות שלו הן הקשתות המחברות צמתים אלו ב-G.

דרגת צומת v היא מספר הבנים של v.

עלה הוא צמת ללא בנים.

צומת פנימי הוא צומת שאינו עלה.

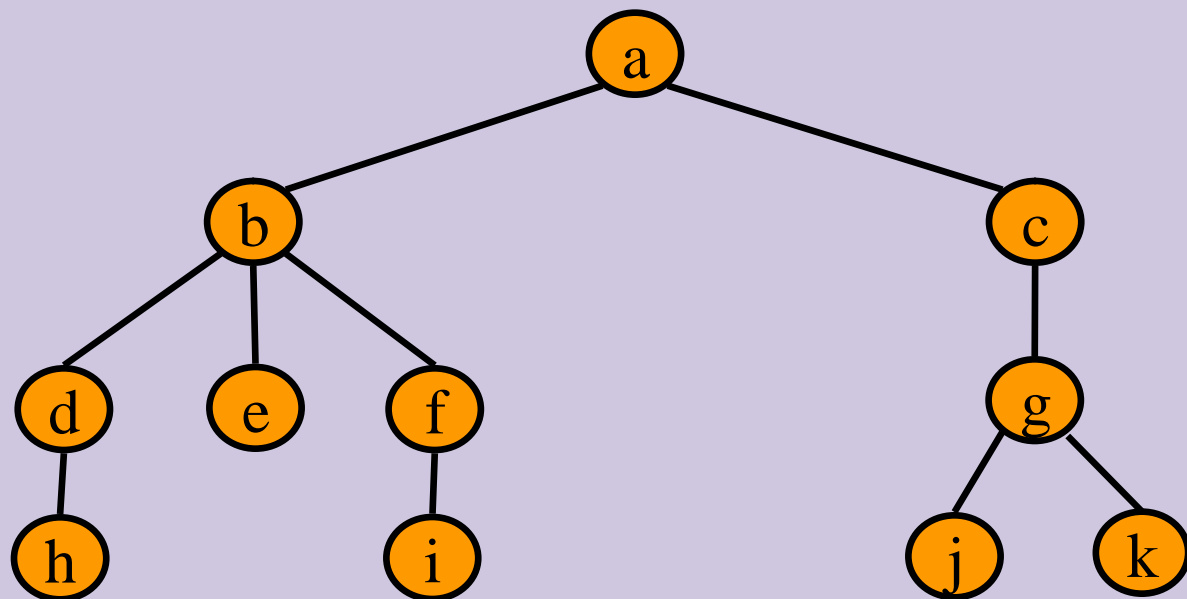
## עצים מכוונים



**עומק של צומת  $v$**  הוא מספר הקשתות משורש העץ אל  $v$  (המרחק מהשורש).

**גובה של צומת  $v$**  הוא מספר הקשתות מ- $v$  לצאצא הרחוק ביותר של  $v$  (עלה).

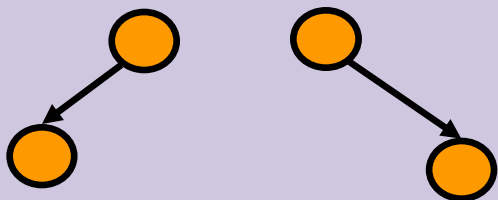
**גובה העץ** הוא הגובה של שורשו.



הערה: לעיתים נשמיט את החצים מתוך הבנה שכוון הקשתות כלפי מטה. כמו כן לרוב נאמר עץ במקום עץ מכוון.

## עצים בינריים

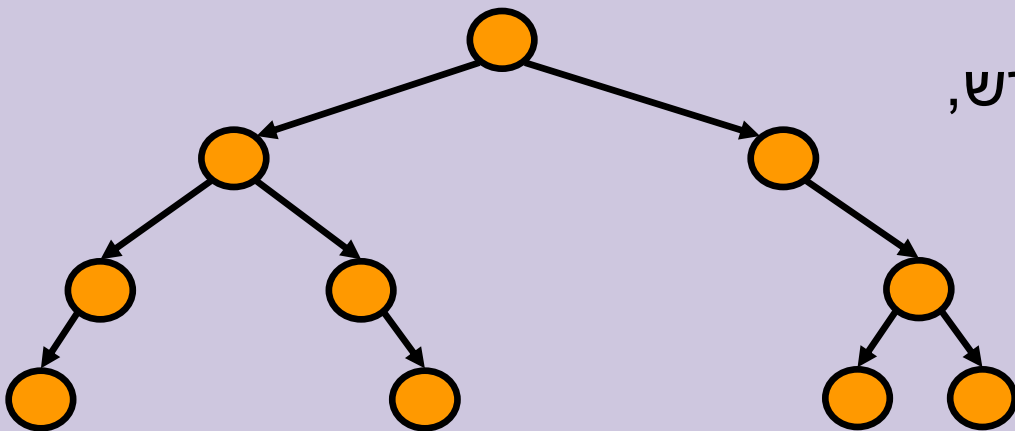
עץ בינרי: עץ שבו לכל צומת שאינו עלה יש בן שמאלי ו/או בן ימני.



הגדרה רקורסיבית: עץ בינרי הוא מבנה

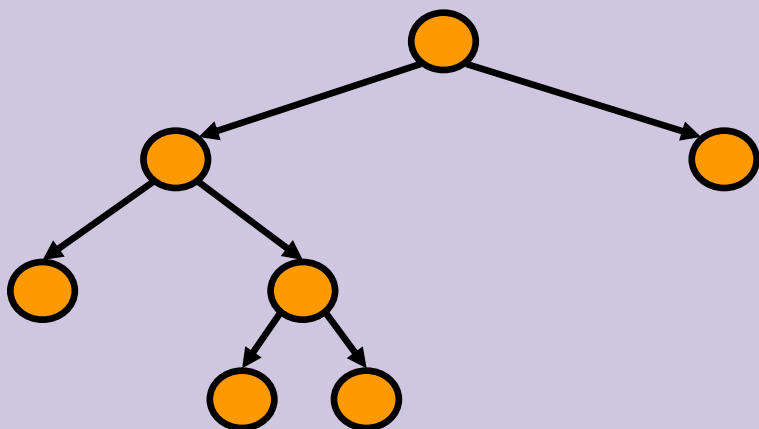
1. ריק (ללא צמתים), או

2. מורכב משלושה חלקים: צומת הנקרא שורש, עץ בינרי הנקרא תת-עץ שמאלי, ועץ בינרי הנקרא תת-עץ ימני.

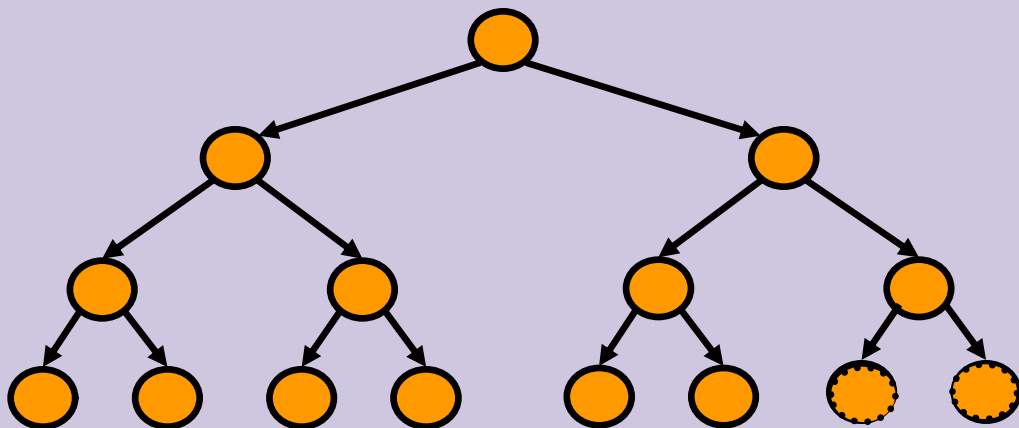


## עצים בינריים מלאים ושלמים

עץ בינרי מלא (full): עץ שבו לכל צומת פנימי 2 בנים.



עץ בינרי שלם (complete): עץ בינרי מלא שבו כל העלים באותו עומק.



עץ בינרי כמעט שלם: עץ בינרי שלם שהוצאו ממנו עלים ("מצד ימין").

## תכונות עצים בינריים שלמים

בעץ בינרי שלם בעל  $n$  צמתים,  $L$  עלים, וגובה  $h$ :

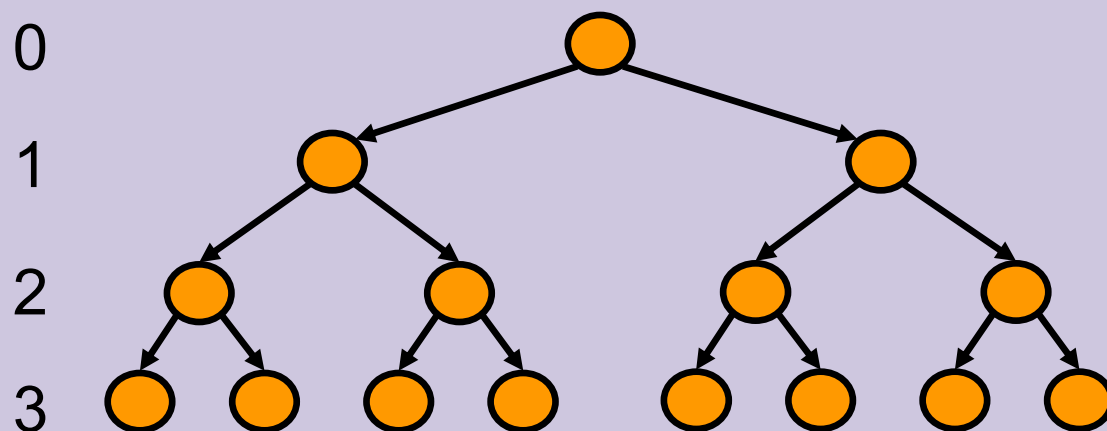
1. מספר הצמתים בעומק  $i$ :  $n_i = 2^i$

2. מספר העלים:  $L = n_h = 2^h$

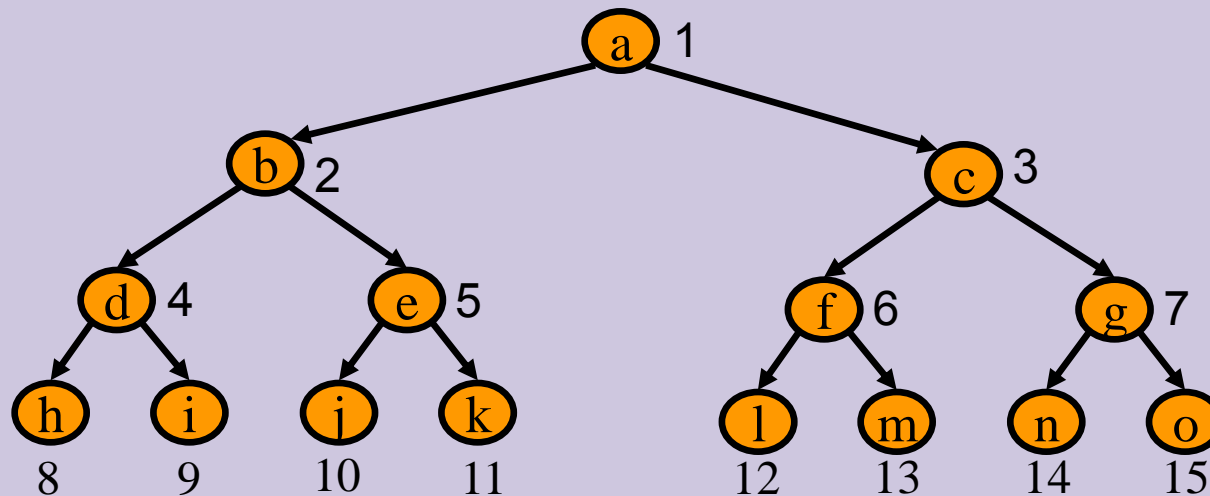
3. מספר הצמתים:  $n = \sum_{i=0}^h n_i = \sum_{i=0}^h 2^i = 2^{h+1} - 1$

4. הגובה:  $h = \log_2(n+1) - 1$

5. מספר הצמתים הפנימיים:  $n - L = 2^{h+1} - 1 - 2^h = 2^h - 1 = L - 1$



# מימוש "מערכי" לעצים בינריים



בן שמאלי של צומת  $i$   
נמצא ב-  $2i$

בן ימני של צומת  $i$  נמצא  
ב-  $2i+1$

אבא של של צומת  $i$   
נמצא ב-  $\lfloor i/2 \rfloor$

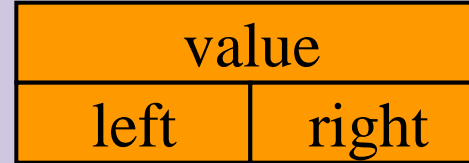
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o

**יעיל רק עבור עצים שלמים !**

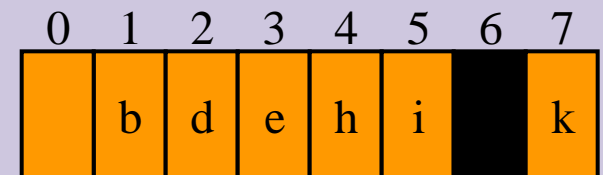
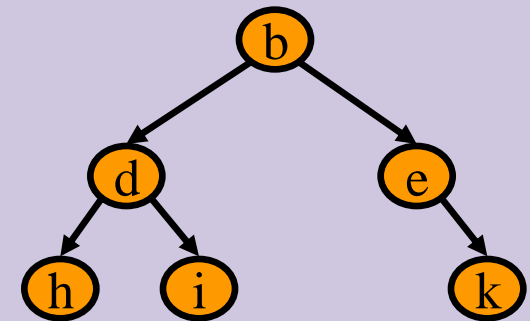
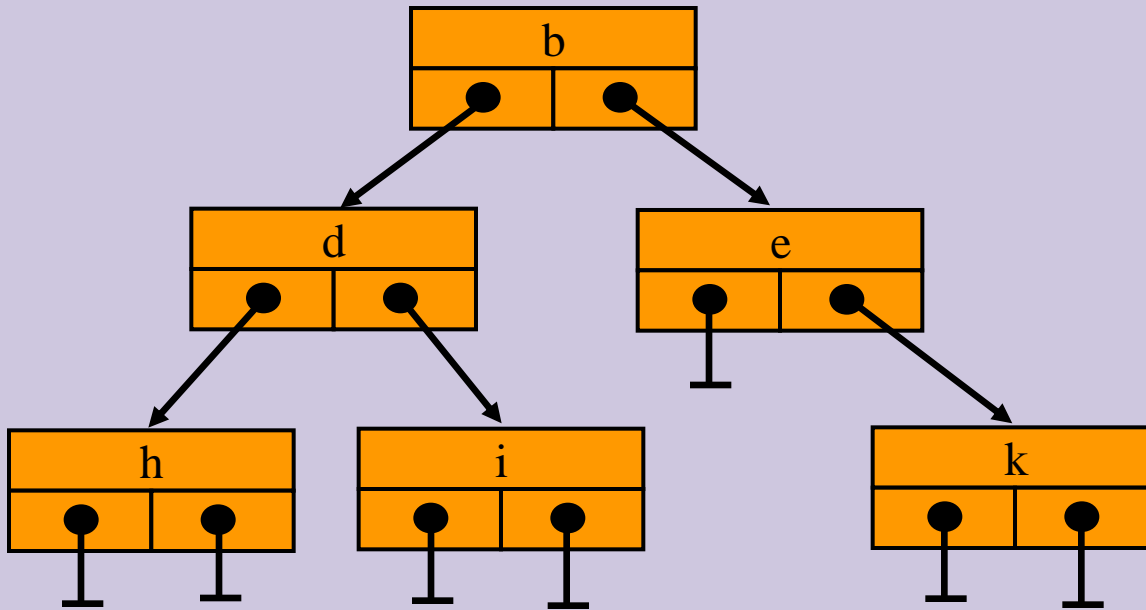


## מימוש באמצעות מצביעים

```
typedef struct node {
    DATA_Type value;
    struct node *left, *right;
} NODE;
```



מבנה צומת:



# Inorder - בעצים בינאריים

## inorder

סייר בתת העץ השמאלי

בקר בשורש

סייר בתת העץ הימני

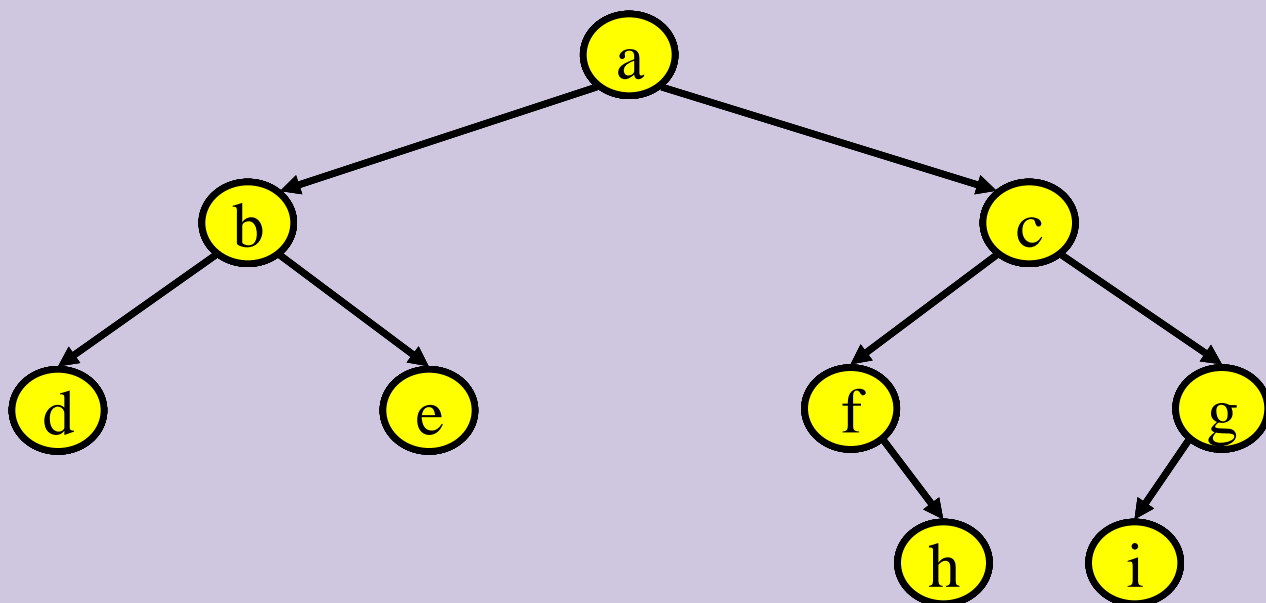
**d b e a f h c i g**

מימוש סיור inorder בעזרת רקורסיה

```
inorder (node *p) { // הפרמטר הוא מצביע לשורש
    if (!p) return; // תנאי נצירה p == NULL

    inorder (p -> left);
    do_something_with (p);
    inorder (p -> right);
}
```

## סיורים בעצים בינריים



### preorder

בקר בשורש

סייר בתת העץ השמאלי

סייר בתת העץ הימני

**a b d e c f h g i**

### inorder

סייר בתת העץ השמאלי

בקר בשורש

סייר בתת העץ הימני

**d b e a f h c i g**

### postorder

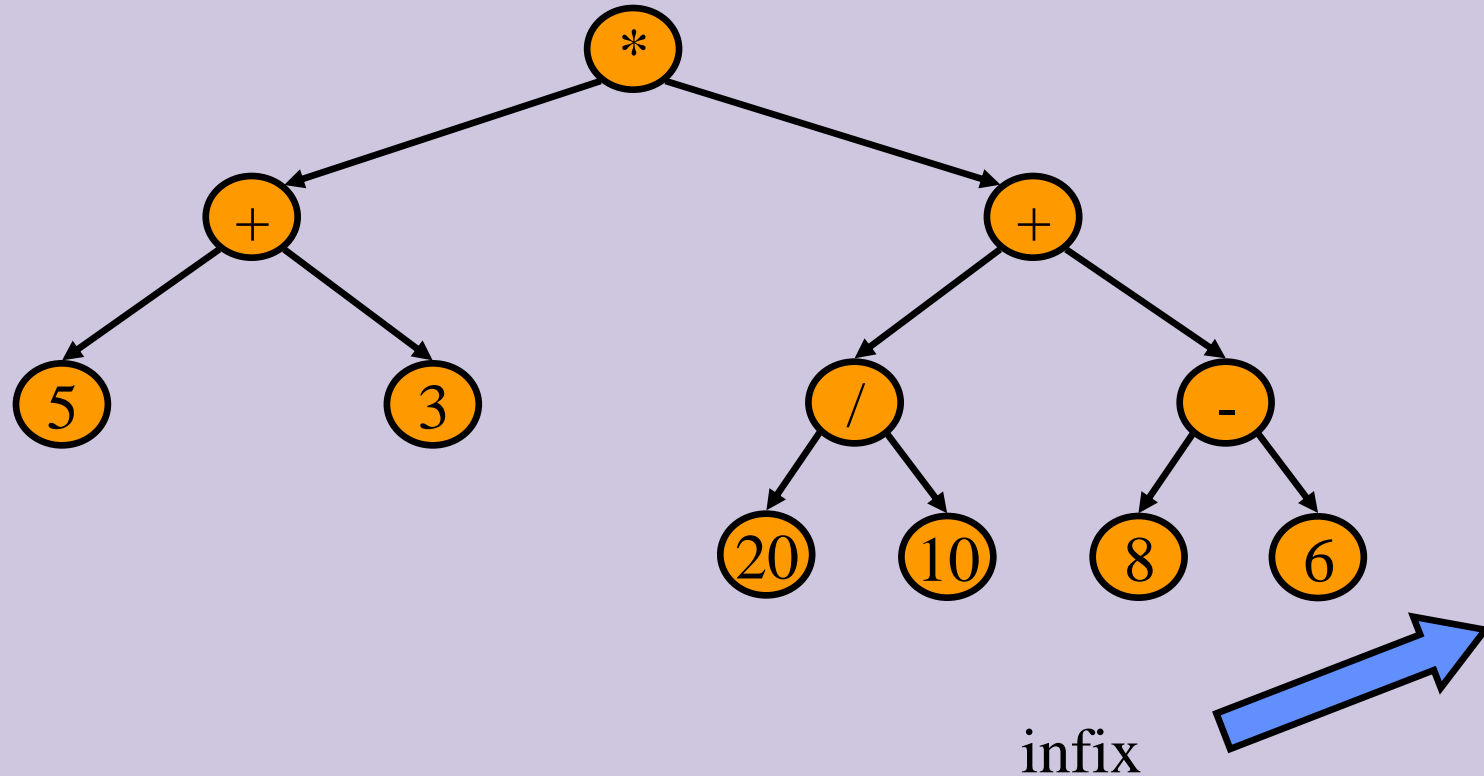
סייר בתת העץ השמאלי

סייר בתת העץ הימני

בקר בשורש (חשוב ביטויים אריתמטיים)

**d e b h f i g c a**

# חישוב והמרה של ביטויים אריתמטיים



inorder

סייר בתת העץ השמאלי

בקר בשורש (הדפס)

סייר בתת העץ הימני

$(5+3)*((20/10)+(8-6))$

# מימוש פרוצדורת postorder

```
void postorder (NODE *T)
{
if (T == NULL) return;

else {
    postorder( T → left);    /* 1*/
    postorder( T → right);   /* 2*/
    “visit”;                 /* 3*/
    return;}

}
```

נשתמש במימוש "מצביעי"

value	
left	right

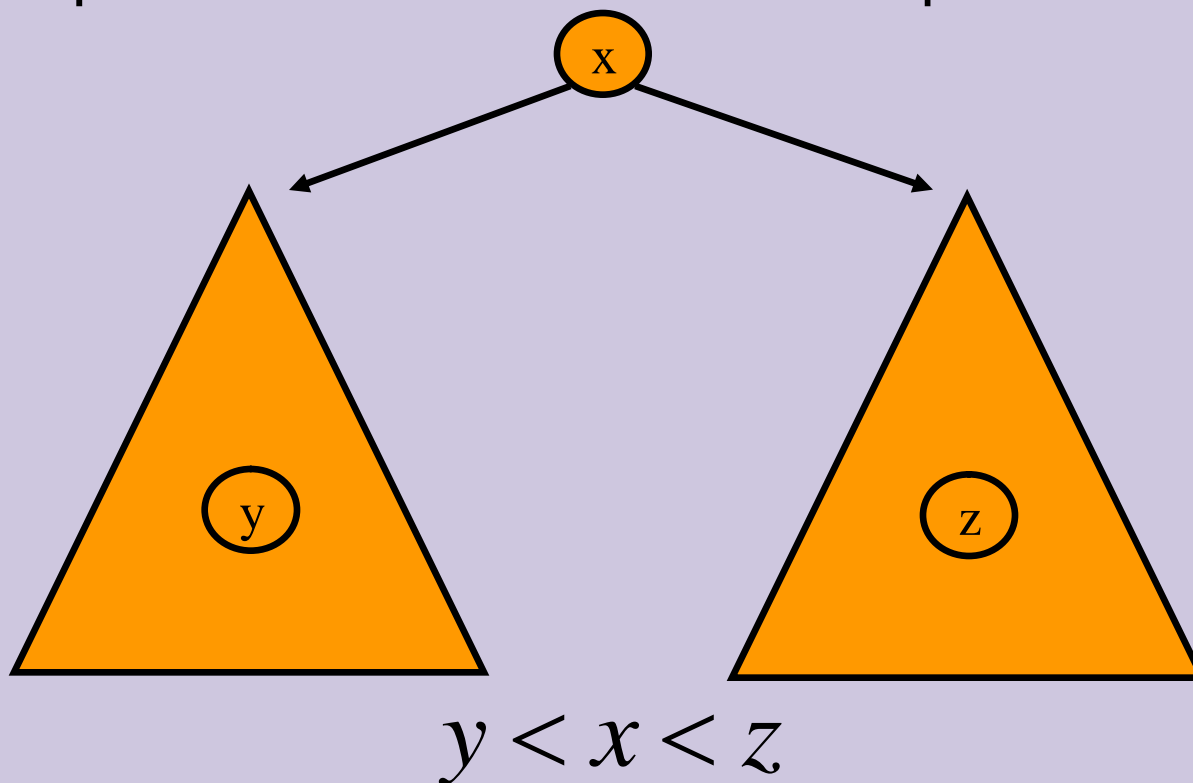
מבנה צומת:

## עץ בינרי כעץ חיפוש

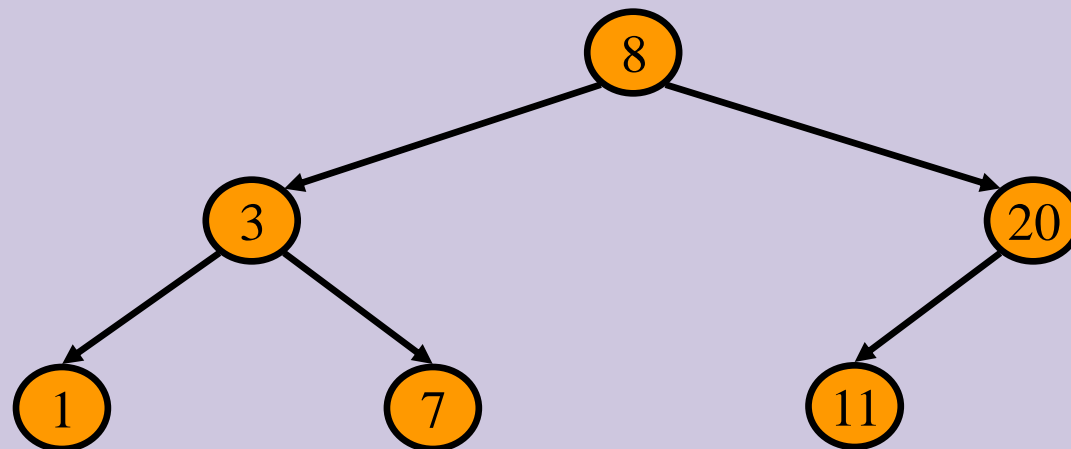
נשתמש בעץ בינרי מכוון.

בכל צומת נאחסן רשומה אחת מתוך מילון (או מפתח ומצביע לאינפורמציה של הרשומה).

נשמור על הכלל הבא: עבור צומת כלשהו בעל מפתח  $x$ , כל המפתחות בתת העץ השמאלי קטנים מ- $x$  וכל המפתחות בתת העץ הימני גדולים מ- $x$ .



## עץ בינרי כעץ חיפוש



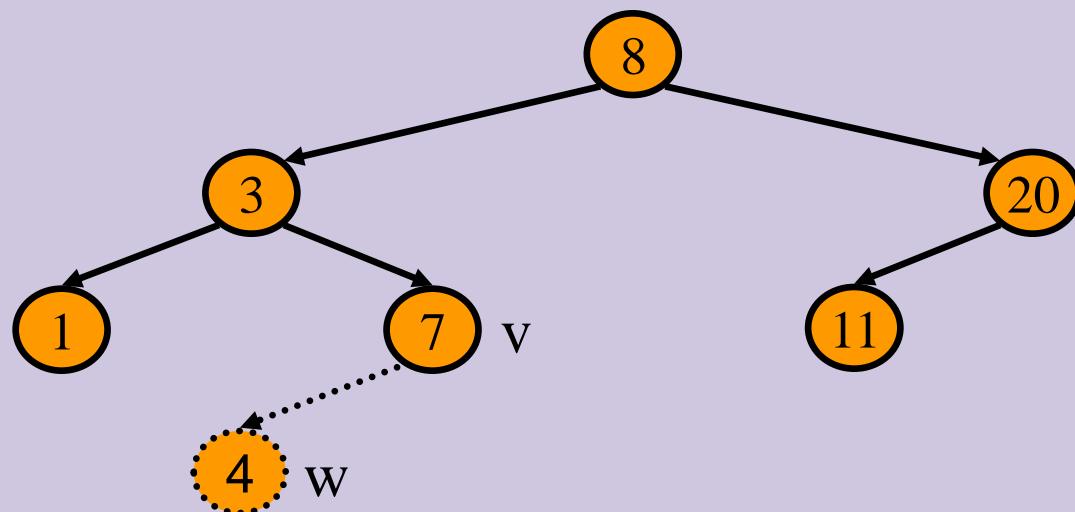
### אלגוריתם החיפוש: $\text{find}(T,x)$

1. אם  $T$  ריק, דווח ש- $x$  לא בעץ.
2. יהי  $y$  הערך שבשורש.
3. אם  $x = y$ , החזר מצביע לצומת המחזיק את  $x$ .
4. אם  $x < y$ , המשך את החיפוש בתת העץ השמאלי של  $T$ .
5. אחרת (כאשר  $x > y$ ), המשך את החיפוש בתת העץ הימני של  $T$ .

## הכנסה בעץ חיפוש

### אלגוריתם הכנסה: $\text{insert}(T,x)$

1. חפש את  $x$  בעץ החיפוש  $T$ .
2. אם  $x$  נמצא ב- $T$ , עצור ודווח.
3. יהי  $v$  הצומת האחרון במסלול החיפוש של  $x$  ויהי  $y$  המפתח שנמצא ב- $v$ .
4. אם  $x < y$ , הוסף צומת  $w$  עם מפתח  $x$  כבן שמאלי של  $v$ .
5. אחרת (כאשר  $x > y$ ), הוסף צומת  $w$  עם מפתח  $x$  כבן ימני של  $v$ .



$\text{insert}(T,4)$

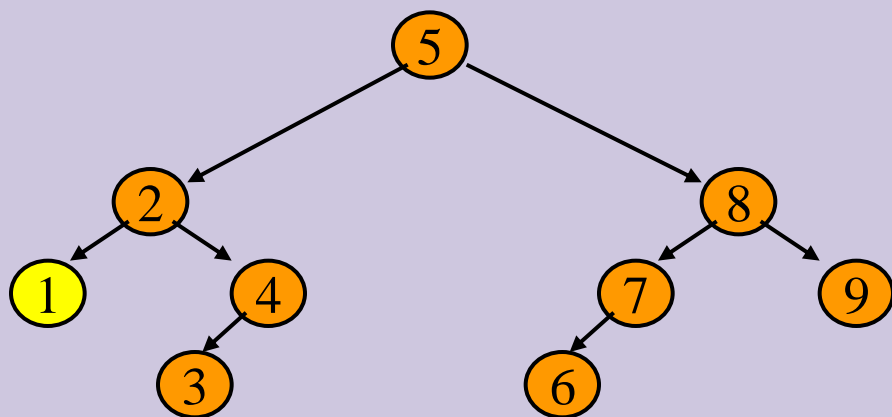


# הוצאה מעץ חיפוש – המקרים הקלים

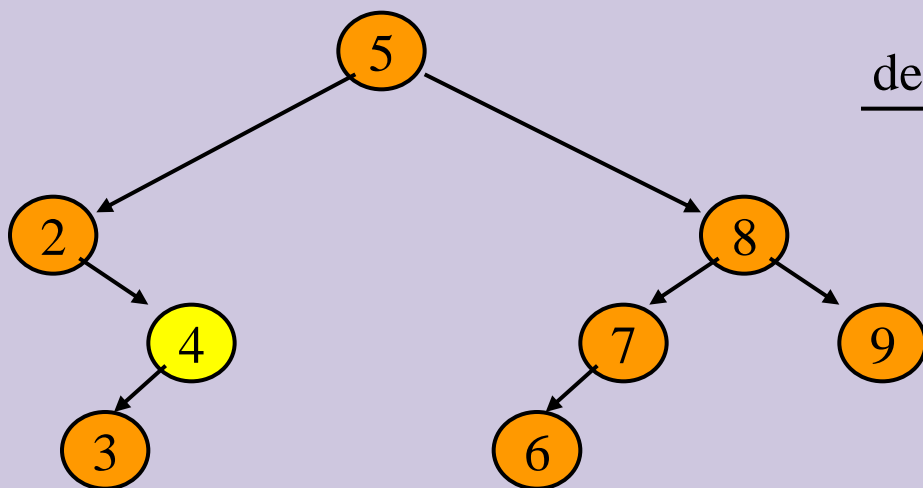
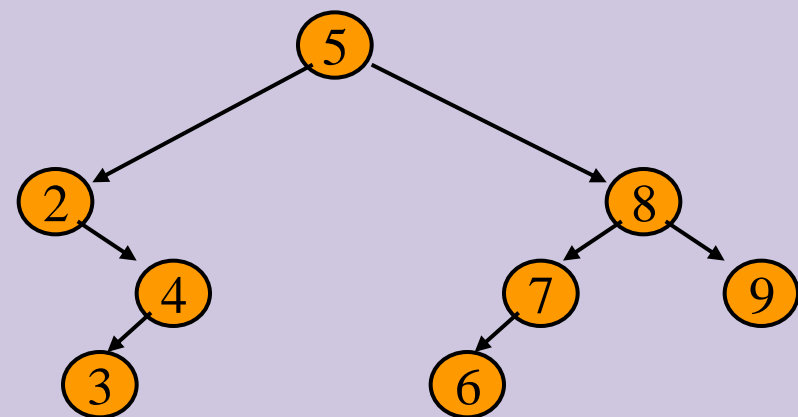
אלגוריתם הוצאה: יהי  $v$  צומת בעץ המיועד להוצאה.

1. אם  $v$  עלה, סלק אותו.

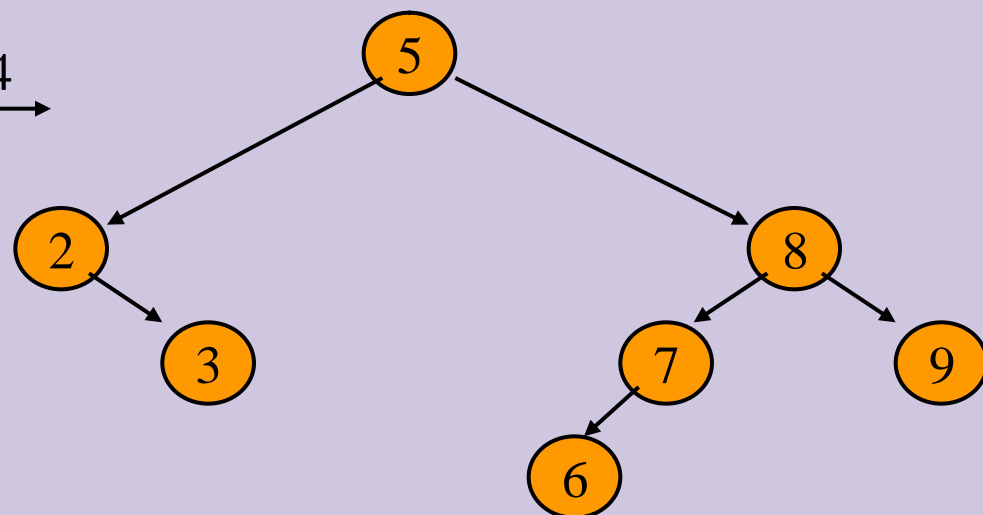
2. אם ל- $v$  בן יחיד, תן לאבא של  $v$  להצביע על הבן.



delete 1 →



delete 4 →



## הוצאה מעץ חיפוש

אלגוריתם הוצאה. יהי  $v$  צומת בעץ המיועד להוצאה.

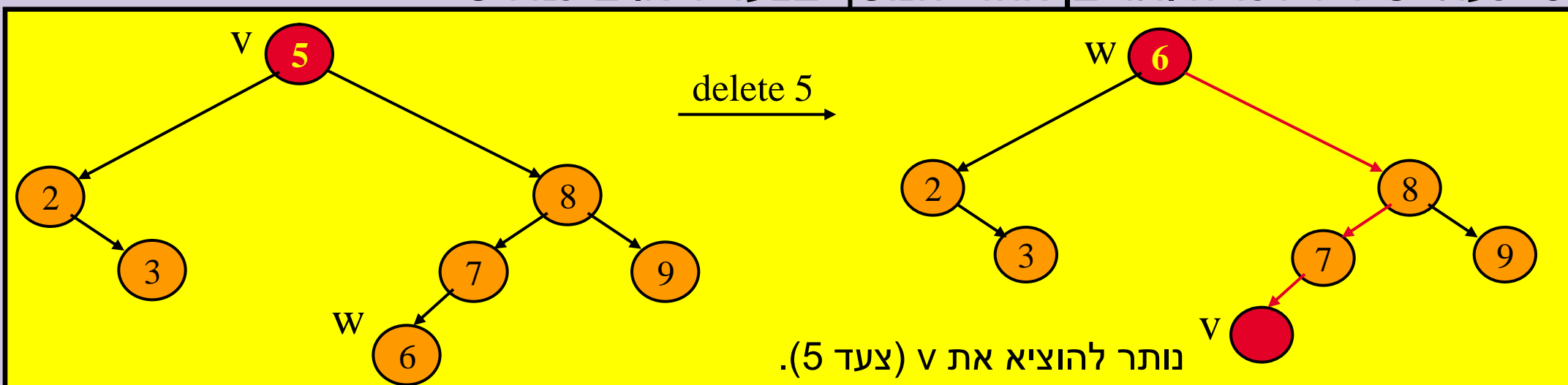
1. אם  $v$  עלה, סלק אותו.

2. אם ל- $v$  בן יחיד, תן לאבא של  $v$  להצביע על הבן.

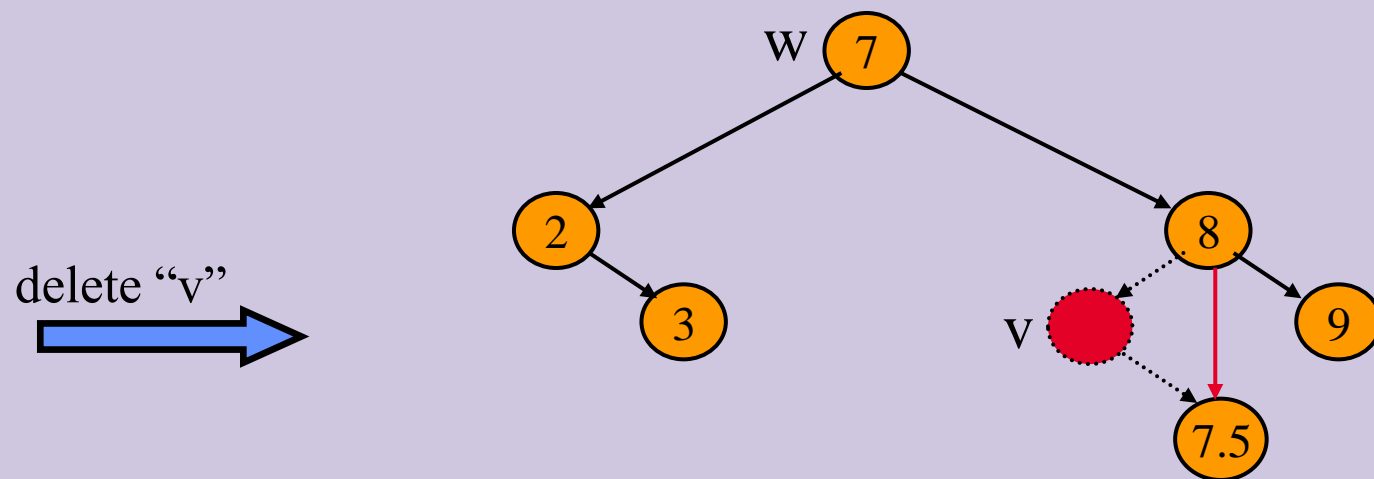
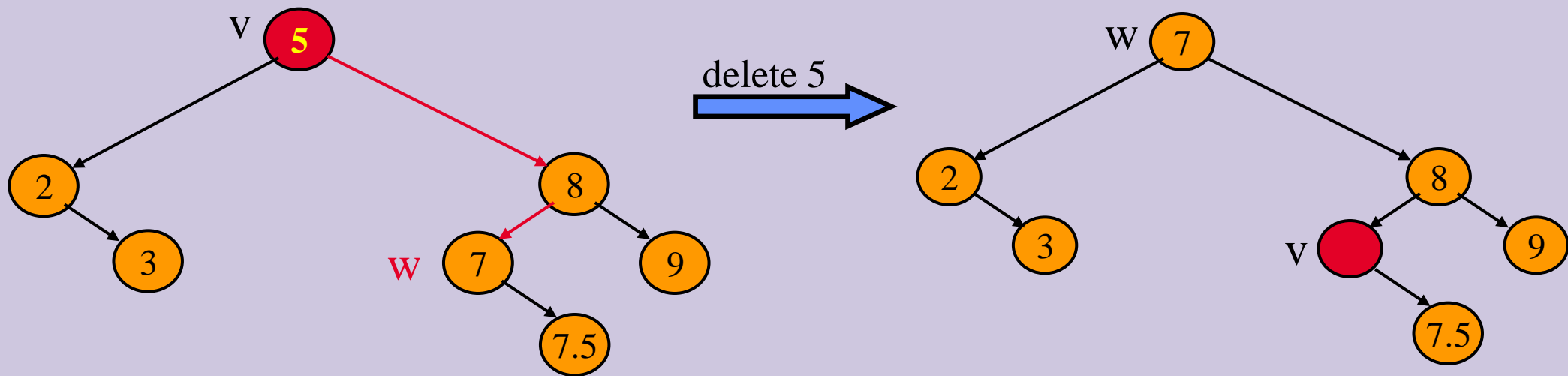
3. אחרת: יהי  $w$  הצומת העוקב ל- $v$  בסדר inorder. (זהו הצומת המכיל את הערך הבא אחרי הערך שב- $v$  כלומר הצומת המתקבל ע"י פניה אחת ימינה ואח"כ כל הדרך שמאלה. שימו לב שלצומת  $w$  בן אחד לכל היותר).

4. החלף בין צומת  $v$  וצומת  $w$ .

5. כעת יש ל- $v$  לכל היותר בן אחד. המשך בצעד 1 או 2 כנדרש.

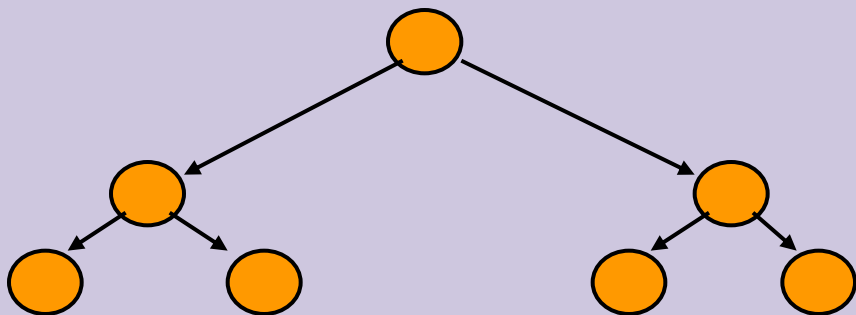


# דוגמא נוספת



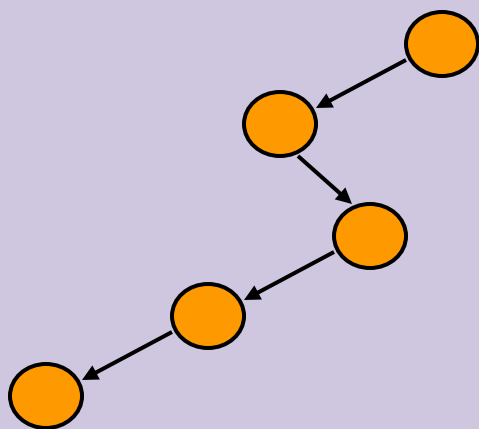
## נתוח זמנים

זמן חיפוש/הכנסה/הוצאה הוא לינארי בגובה העץ.



מהו גובה העץ?

מקרה טוב. עץ שלם.  $h = \lfloor \log n \rfloor$



מקרה גרוע. עץ הנראה כרשימה ליניארית.  $h = n - 1$

ומה הגובה הממוצע ?

## גובה ממוצע

ברור שצורת העץ נקבעת על פי סדר ההכנסה (למשל הסדר 1,2,3 יוצר שרשרת לעומת 2,1,3 שיוצר עץ מאוזן).

מספר אפשרויות (הסדרים) להכניס  $n$  צמתים לעץ הוא  $n!$ .

נסמן ב-  $h(i)$  את גובה העץ הנוצר בסדר ה- $i$ .

$$\bar{h} = \frac{1}{n!} \sum_{i=1}^{n!} h(i)$$

הגובה הממוצע מוגדר כדלקמן:

ניתן להראות שהגובה הממוצע שייך לקבוצה-  $O(\log n)$  כלומר בממוצע כל הפעולות מתבצעות בזמן  $O(\log n)$ .

זמן בניה ממוצע של עץ חיפוש בינרי הוא  $O(n \log n)$ .