עצים ועצי חיפוש

חומר קריאה לשיעור זה

Chapter 5.5- Trees (91 - 97)

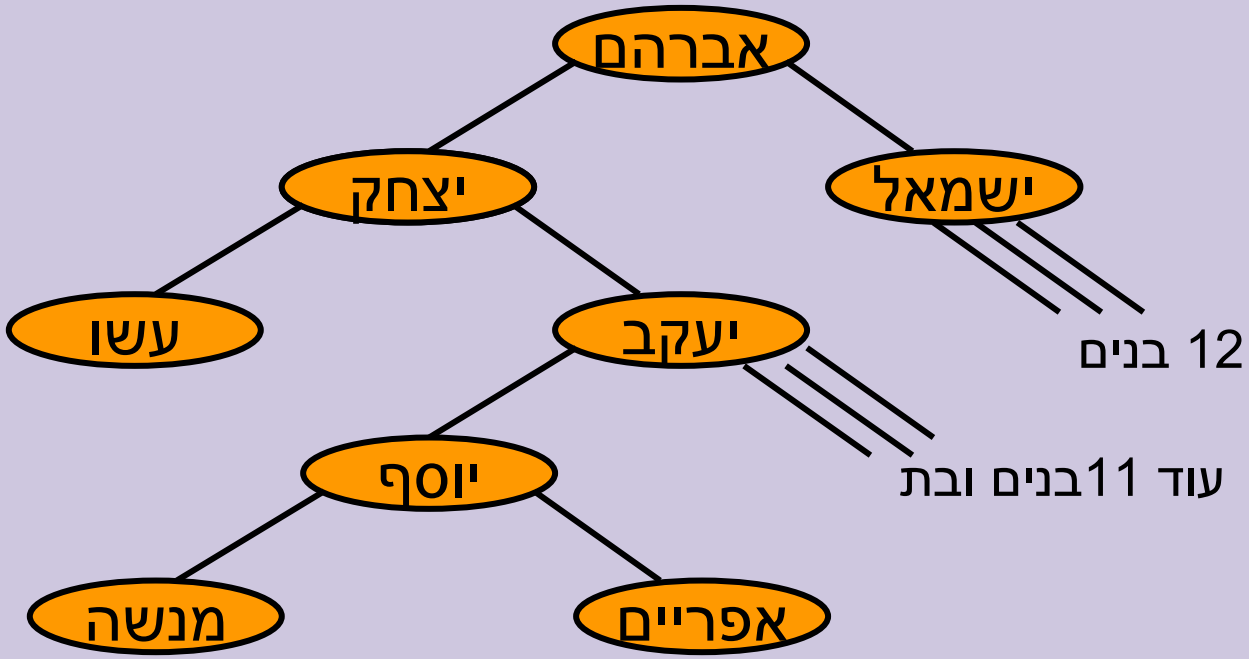
Chapter 13- Binary Search Trees (244 - 262)

©Geiger & Itai, 2001

עצים

דוגמאות

1. אִילָן יִחֻסִין

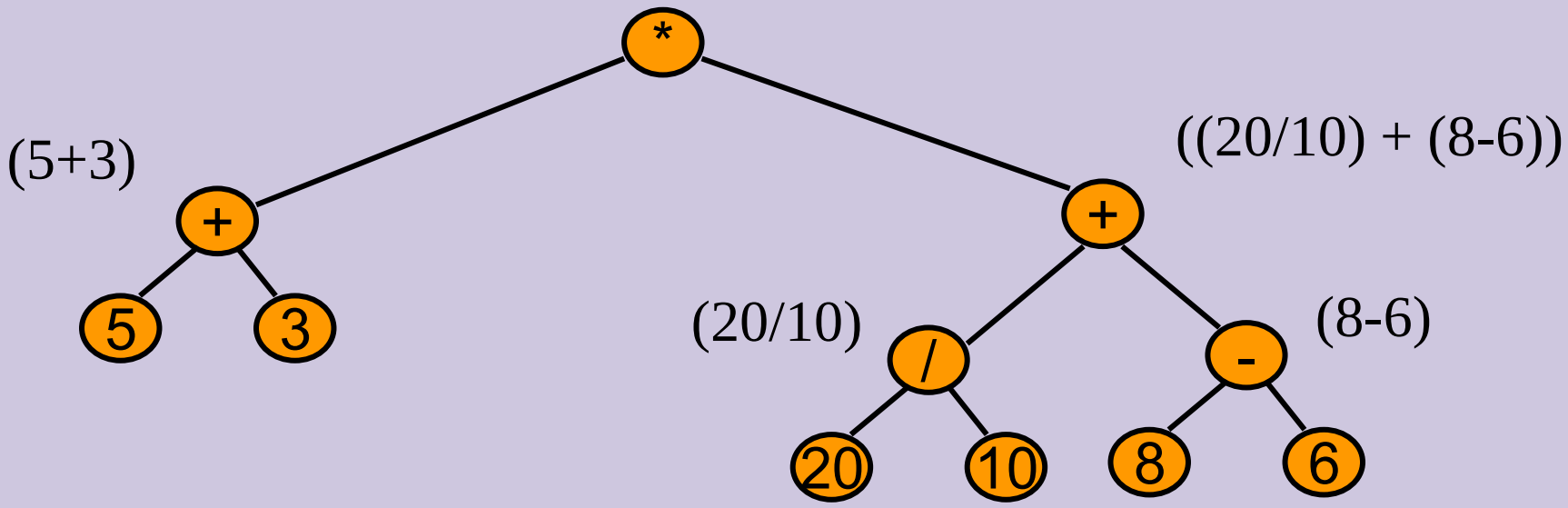


©cs,Technion

עצים

2. ביטויים אריתמטיים

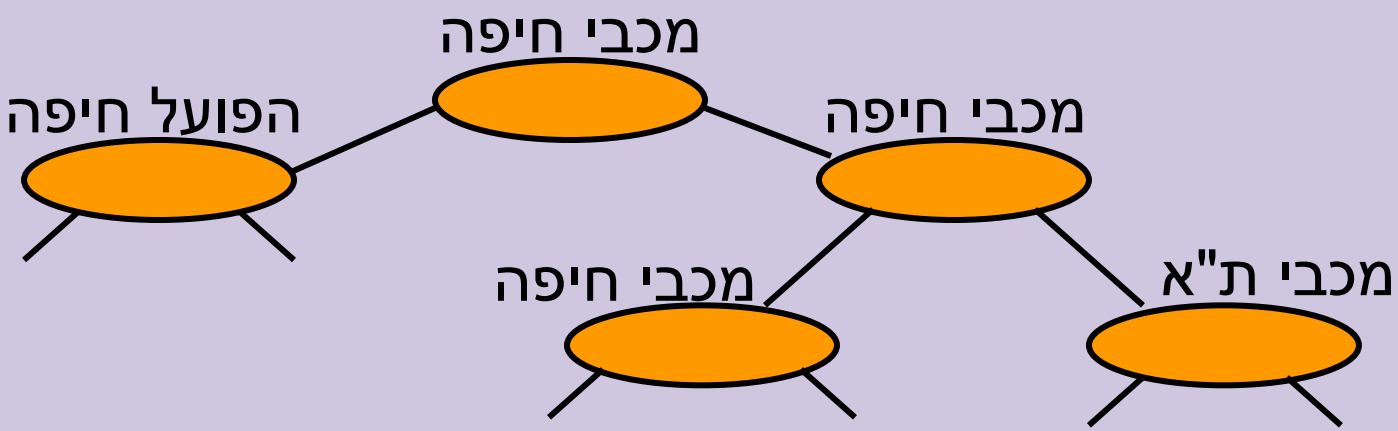
(5+3) * ((20/10) + (8-6))



©cs,Technion

עצים

3. עץ מנצחים (גביע)



4. מבנה היררכי

צה"ל
חיל הים
חיל האוויר
חיל היבשה
שריון
תותחנים
...
חי"ר
חטיבת צנחנים
חטיבת הנח"ל
גדוד 856
...
גדוד 934
מחלקה 1
מחלקה 2
כיתה 1
כיתה 2
אלון אבוטבול

©cs,Technion

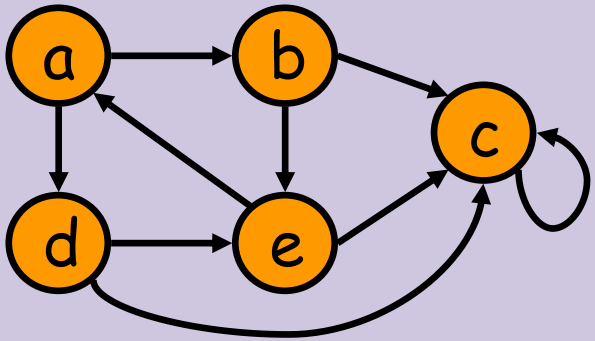
גרפים מכוונים (Directed Graphs)

גרף מכוון הוא זוג (V,E) המורכב מקבוצת צמתים V

וקבוצת קשתות $E \subseteq V \times V$.

$V = \{a,b,c,d,e\}$

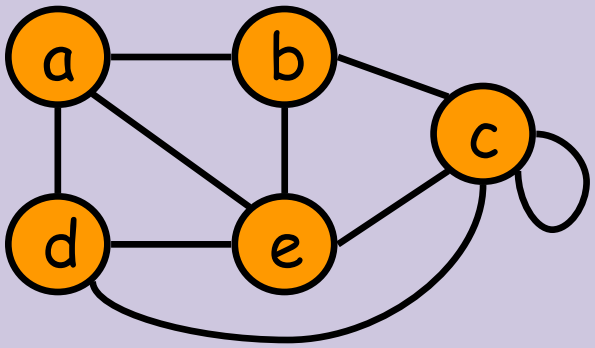
$E = \{(a,b),(a,d),(b,c),(b,e),(c,c),(d,c),(d,e),(e,a),(e,c)\}$



נסמן $n = |V|$ וכן $m = |E|$. בדוגמא: $n = 5$, $m = 9$.

גרפים לא-מכוונים (Undirected Graphs)

גרף לא-מכוון הוא זוג (V,E) המורכב מקבוצת צמתים V וקבוצת קשתות E
קשת ב-E היא קבוצה בת שני איברים מתוך V. קשת מסומנת ע"י (i,j)
(במקום הסימון המדויק יותר $\{i,j\}$).



חוג עצמי

V = {a,b,c,d,e}

E = {(a,b), (a,d), (a,d) ,(b,c),(b,e),(c,c),(d,c),(d,e),(e,c) }

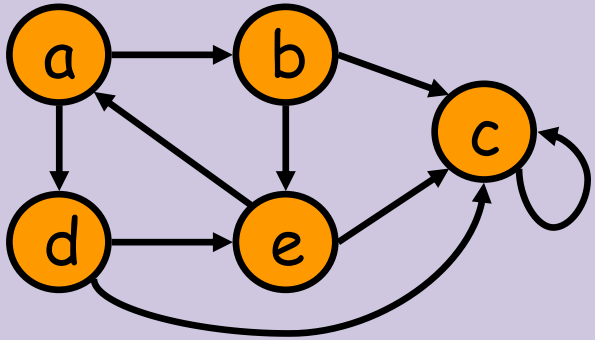
נסמן $n = |V|$ וכן $m = |E|$. בדוגמא: $n = 5$, $m = 9$.

מספר הקשתות m קטן בכל גרף מ- n^2 .

©cs,Technion

הגדרות לגרפים מכוונים (Directed Graphs)

$V = \{a,b,c,d,e\}$

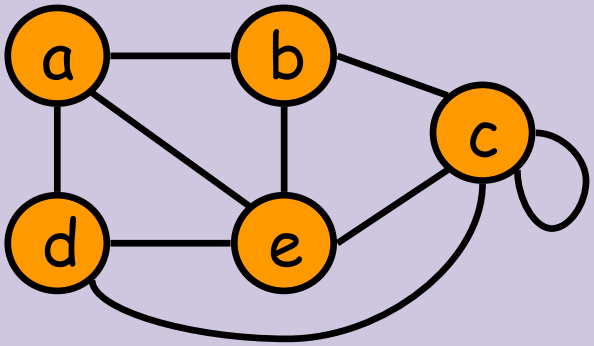


$E = \{(a,b),(a,d),(b,c),(b,e),(c,c),(d,c),(d,e),(e,a),(e,c)\}$

מסלול (מכוון) בגרף (מכוון) (V,E) הוא סדרת צמתים $(v_1,v_2,..,v_k)$ כך שלכל זוג צמתים עוקבים בסדרה, (v_i,v_{i+1}) היא קשת ב-E.

המסלול נקרא **מעגל** (מכוון) אם $v_1=v_k$ (לדוגמא, (a,d,e,a)).

גרף התשתית של גרף מכוון G הוא גרף לא-מכוון עם אותם צמתים כמו ב- G ואותם קשתות כמו ב-G אך ללא כוון. לדוגמא:

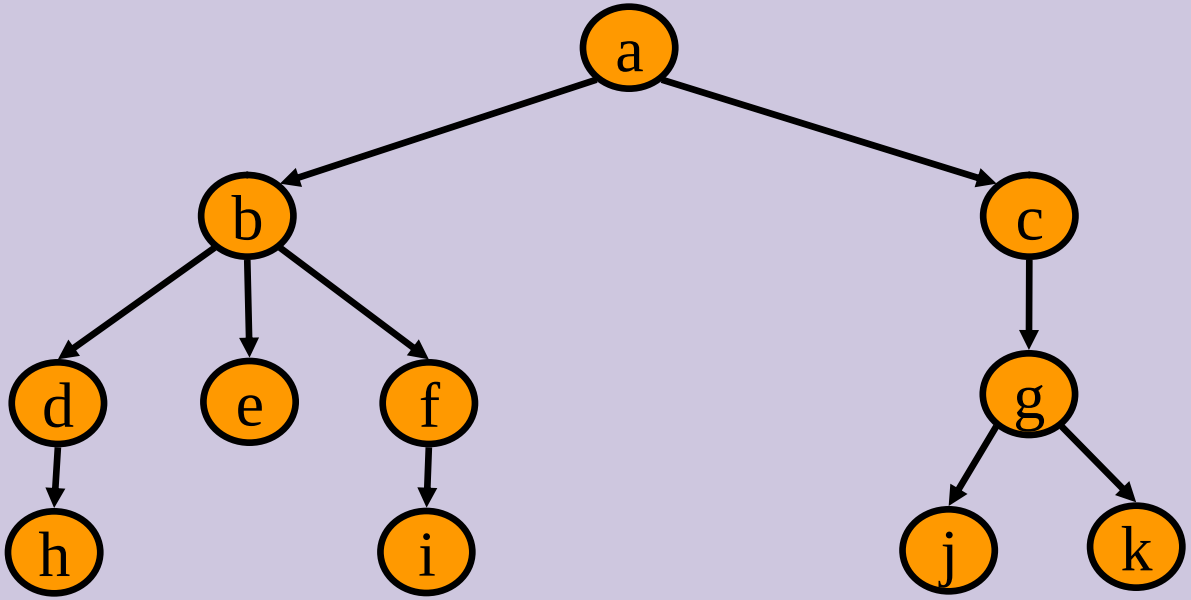


עצים מכוונים

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**.

הגדרות	דוגמאות
v **בן** של u אם קיימת קשת מצומת u לצומת v.	f בן של b
u **אב** של v אם v בן של u.	b אב של e



©cs,Technion

עצים מכוונים

הגדרות

v **צֶאֱצָא** של u אם קיים מסלול מכוון מצומת u ל- v.

u **אָב קַדְמוֹן** של v אם v צאצא של u.

תת-עץ של G ששורשו v הוא עץ מכוון שצמתיו הם v עצמו וכל הַצֶּאֱצָאִים 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.

דוגמאות

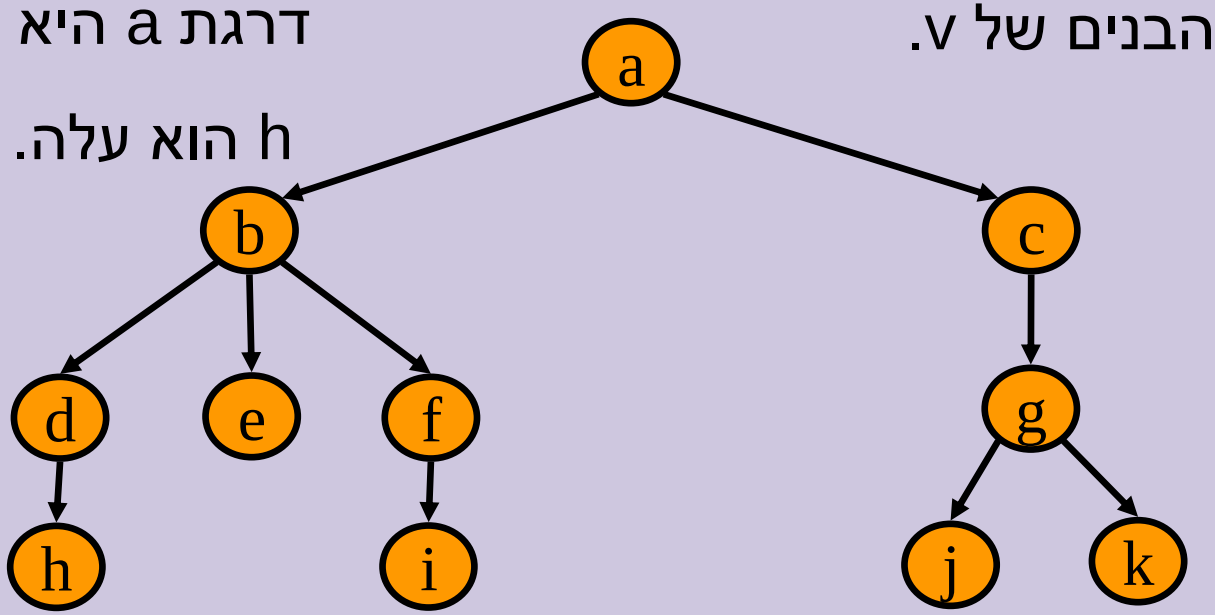
g צאצא של a

b אב-קדמון של h

תת העץ ששורשו g מכיל 3 צמתים ושתי קשתות.

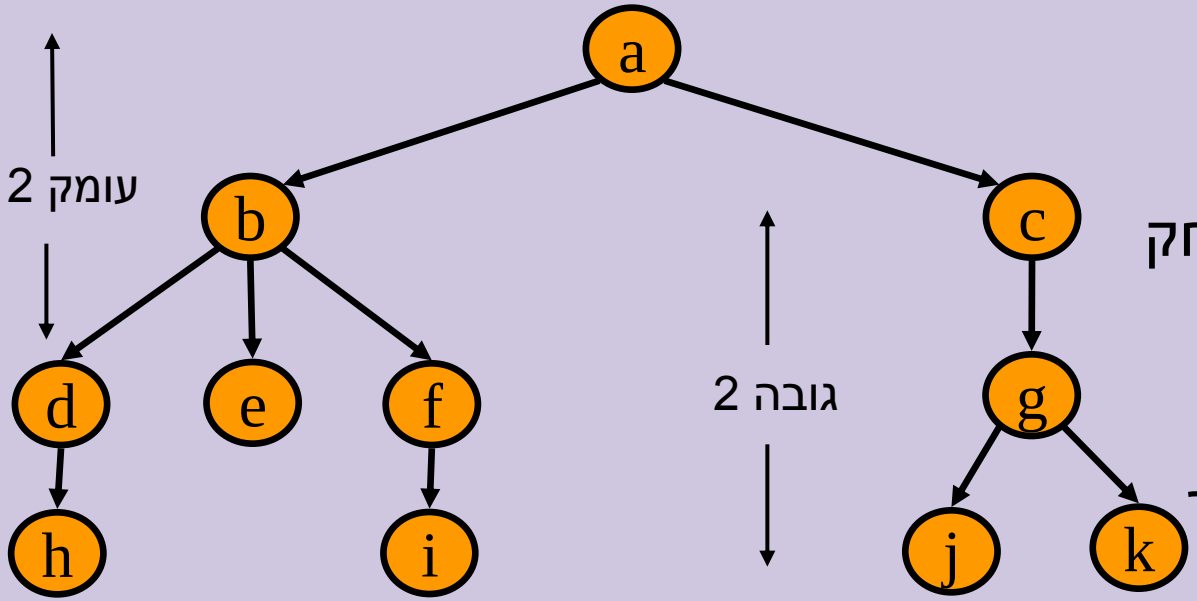
דרגת a היא 2.

h הוא עלה.



©cs,Technion

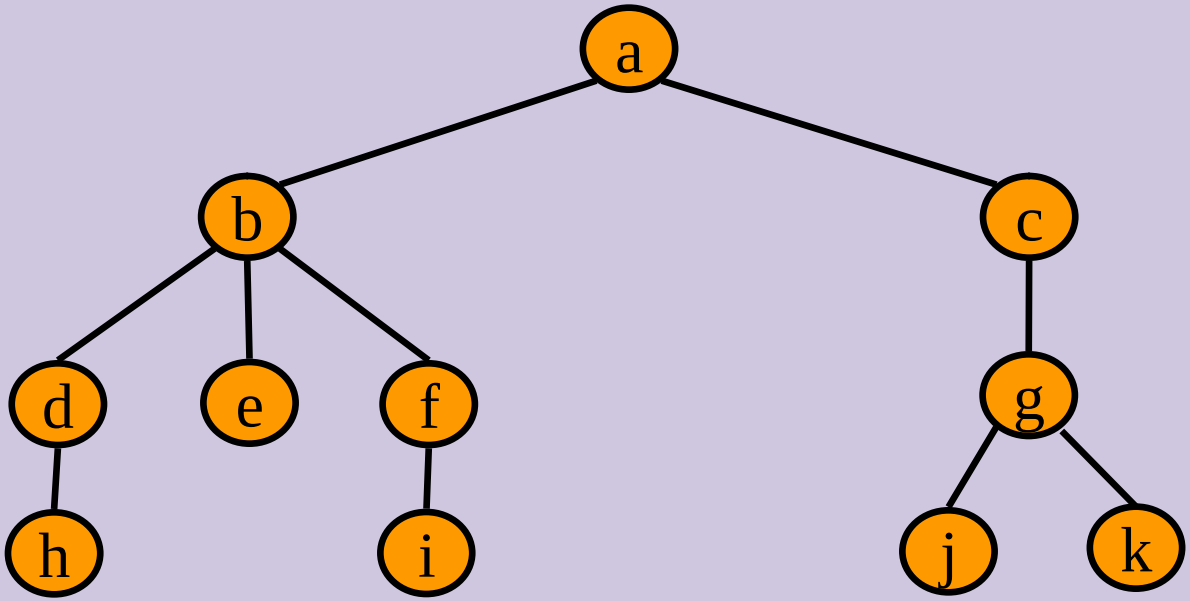
עצים מכוונים



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.



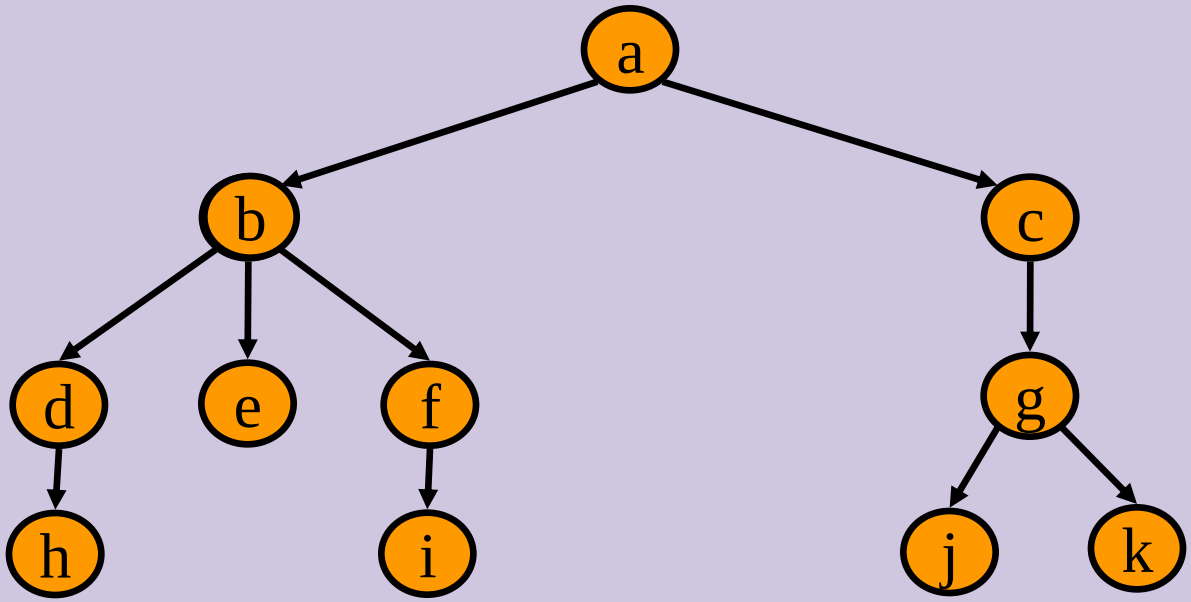
הערה: לעיתים נשמיט את החצים מתוך הבנה שכוון הקשתות כלפי מטה. כמו כן לרוב נאמר עץ במקום עץ מכוון.

©cs,Technion

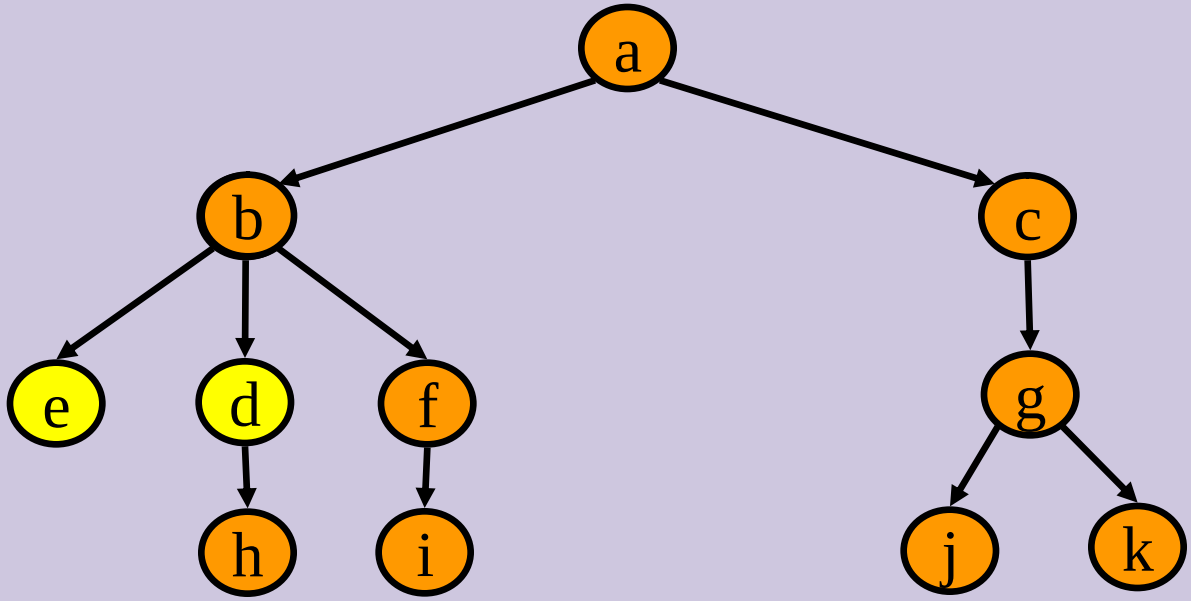
עצים מסודרים

עץ מסודר הוא עץ מכוון

שבו הבנים של כל צומת מסודרים (משמאל לימין).



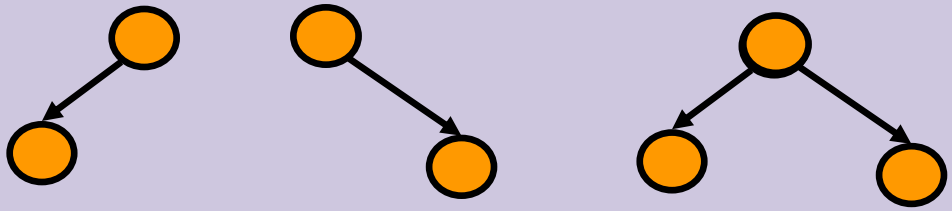
למשל עץ זה שונה מהעץ העליון בגלל שסדר הבנים השתנה.



©cs,Technion

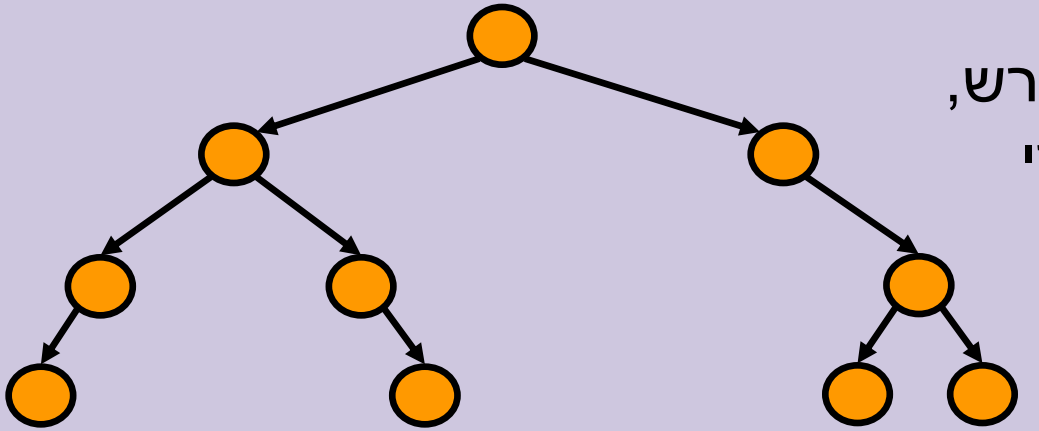
עצים בינריים

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



הגדרה רקורסיבית: עץ בינרי הוא מבנה

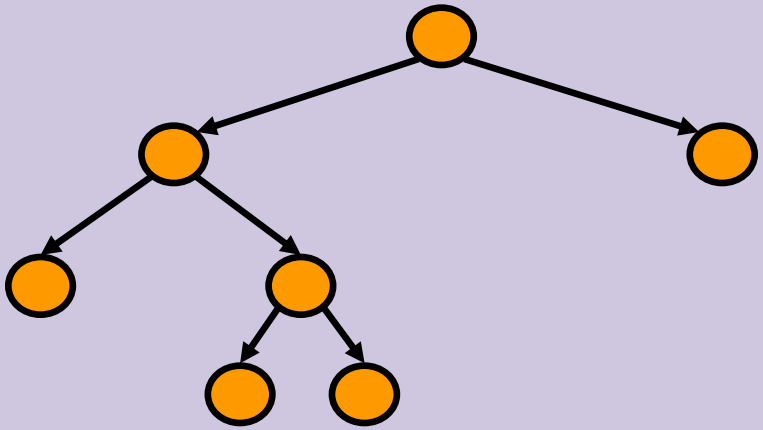
1. ריק (ללא צמתים), או
2. מורכב משלושה חלקים: צומת הנקרא שורש, עץ בינרי הנקרא תת-עץ שמאלי, ועץ בינרי הנקרא תת-עץ ימני.



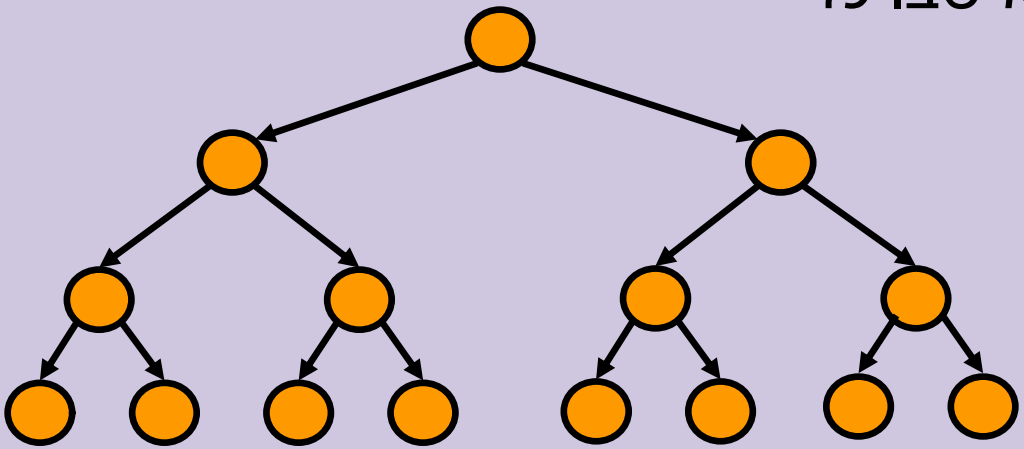
©cs,Technion

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.



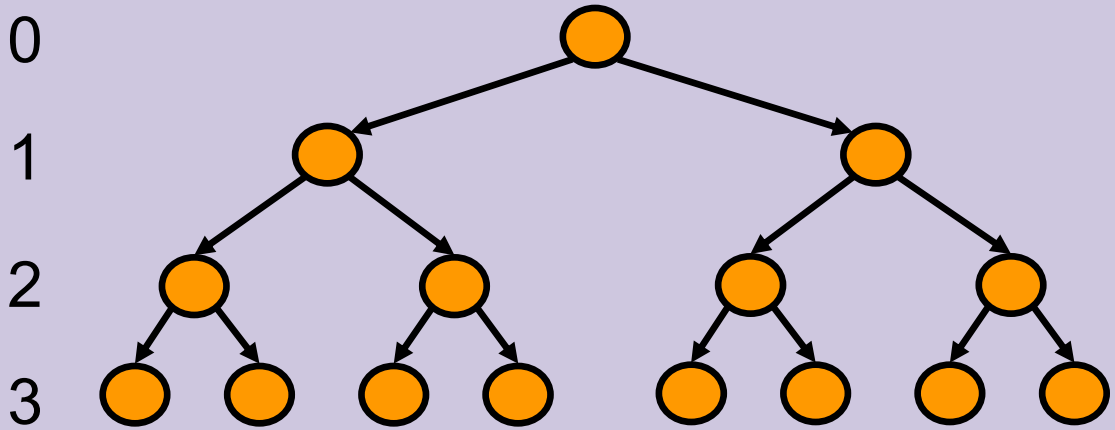
עץ בינרי כמעט שלם: עץ בינרי שלם שהוצאו ממנו עלים ("מצד ימין").

©cs,Technion

תכונות עצים בינריים שלמים

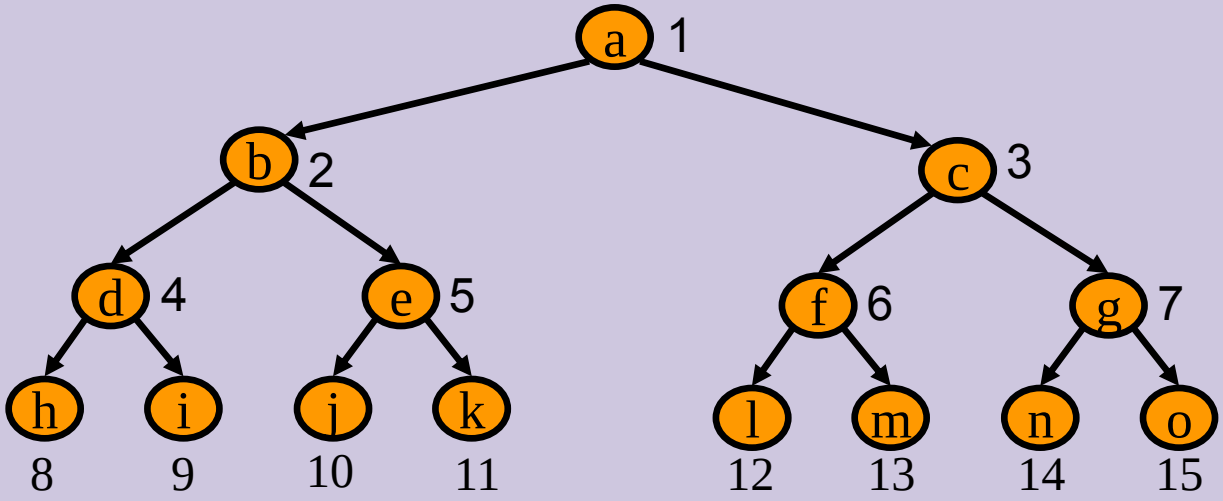
בעץ בינרי שלם בעל n צמתים, L עלים, וגובה h:

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: $n - L = 2^h - 1 = L - 1$



©cs,Technion

מימוש "מערכי" לעץ בינרי שלם



בן שמאלי של צומת i נמצא ב- $2i$

בן ימני של צומת i נמצא ב- $2i+1$

אבא 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

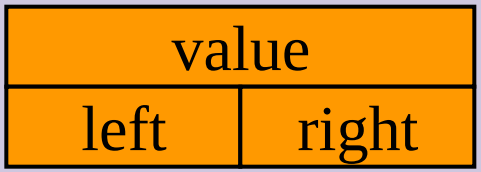
יעיל רק עבור עצים שלמים או כמעת שלמים !

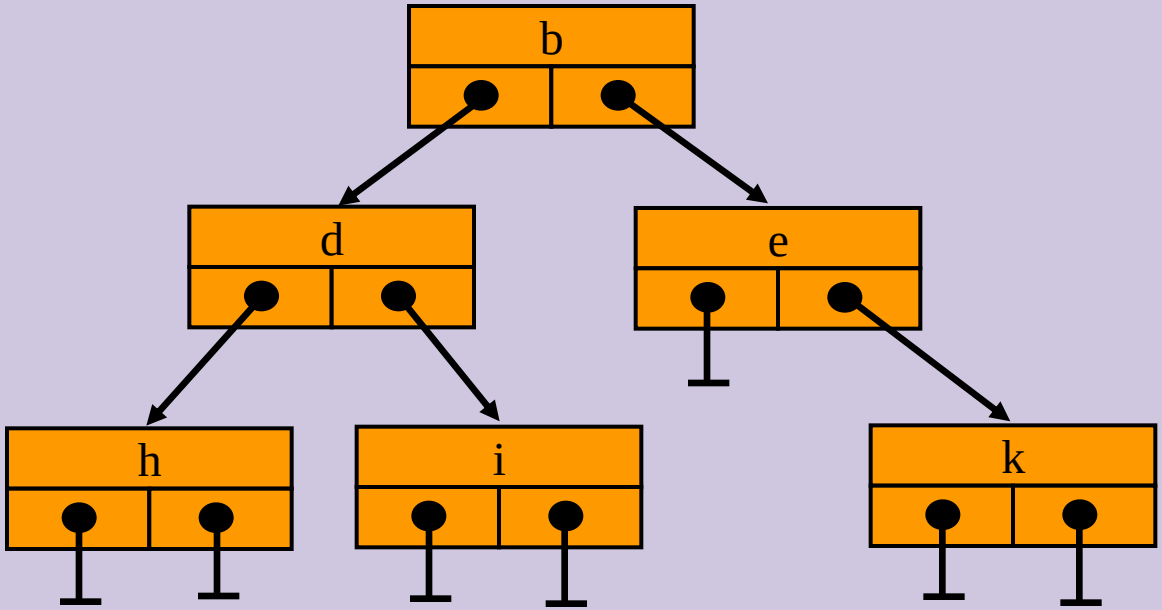
©cs,Technion

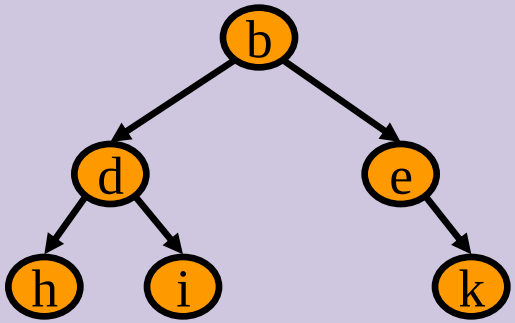
מימוש באמצעות מצביעים

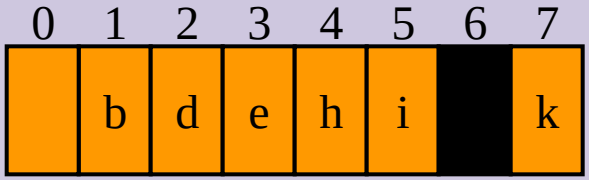
```
typedef struct node {
  DATA_Type value;
  struct node *left, *right;
  }  NODE;
```

מבנה צומת:



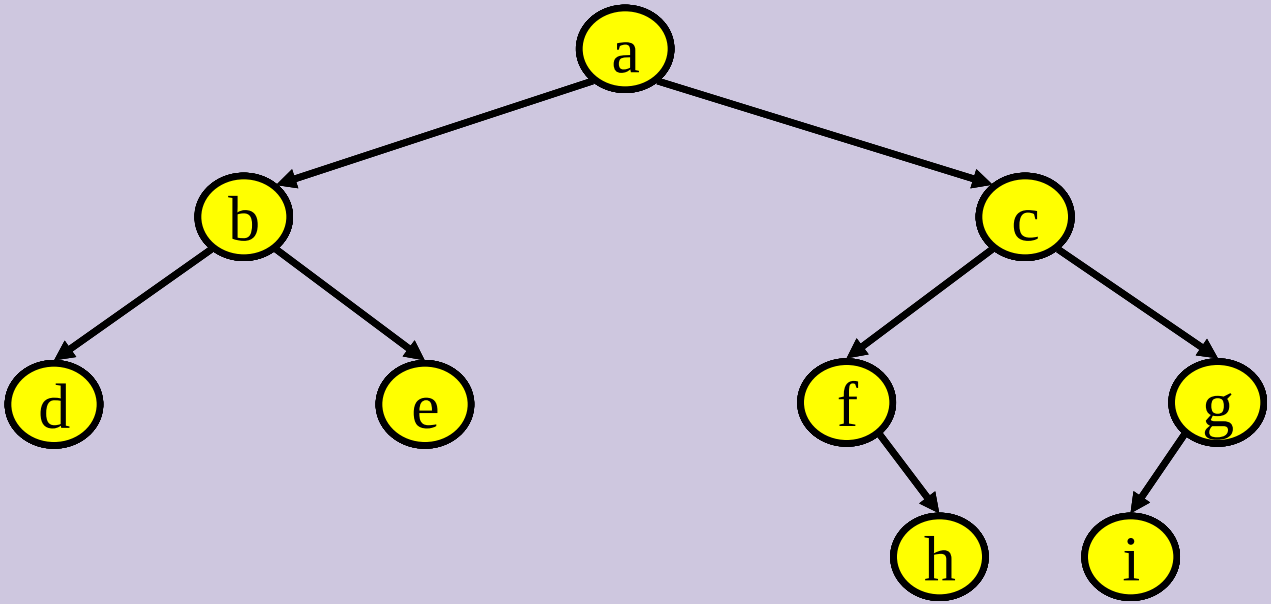






©cs,Technion

סִיּוּרִים בעצים בינריים



preorder

בַּקֵּר בשורש

סַיֵּר בתת העץ השמאלי

סַיֵּר בתת העץ הימני

a b d e c f h g i

postorder

סַיֵּר בתת העץ השמאלי

סַיֵּר בתת העץ הימני

בַּקֵּר בשורש (חשוב ביטויים אריתמטיים)

d e b h f i g c a

inorder

סַיֵּר בתת העץ השמאלי

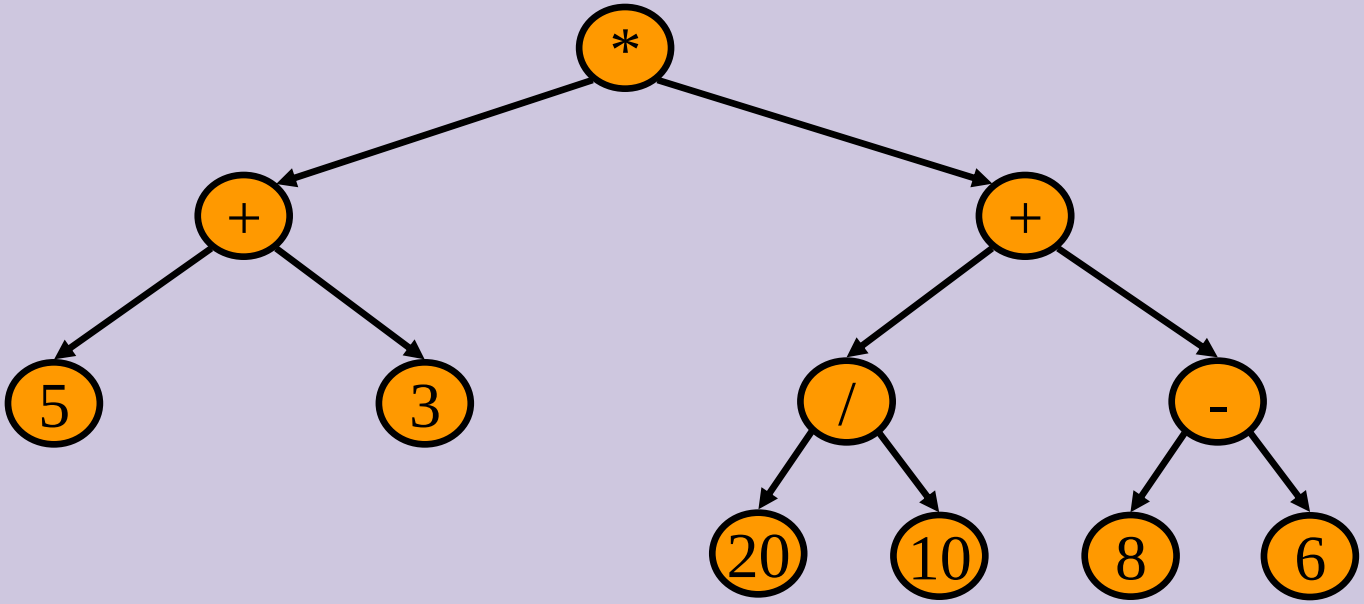
בַּקֵּר בשורש

סַיֵּר בתת העץ הימני

d b e a f h c i g

©cs,Technion

חישוב וְהַמָּרָה של ביטויים אריתמטיים



<u>inorder</u>

סייר בתת העץ השמאלי

בקר בשורש (הדפס)

סייר בתת העץ הימני

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

infix

<u>postorder</u>

סייר בתת העץ השמאלי

סייר בתת העץ הימני

בקר בשורש (חשב את ערך הביטוי או הדפס).

5 3 + 20 10 / 8 6 - + *

postfix

ביטויי postfix נקראים **כתיב פולני**

ע"ש שם הלוגיקאי הפולני Łukasievicz

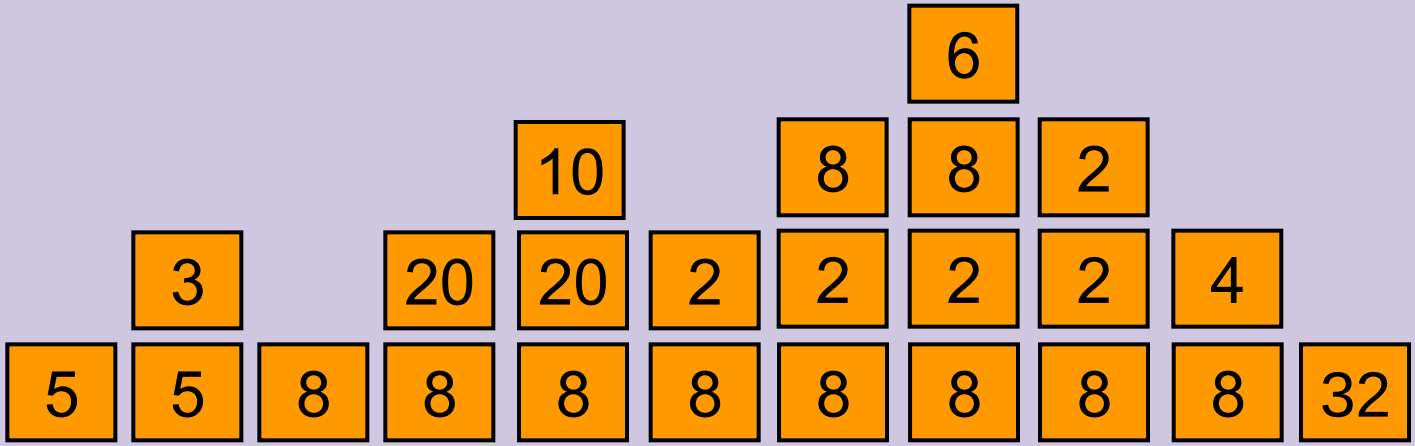
בביטוי postfix אין צורך בסוגריים: לכל ביטוי postfix יש לכל היותר פרוק יחיד!

©cs,Technion

חישוב ביטוי postfix באמצעות מחסנית

אלגוריתם לחישוב ביטוי postfix

1. התחל עם מחסנית ריקה.
2. עבור על כל ביטוי משמאל לימין:
3. אם האיבר הבא הוא אופרנד (מספר) - הכנס אותו למחסנית.
4. אם הוא פעולה – הפעל את הפעולה על שני האיברים שבראש המחסנית והכנס את התוצאה למחסנית.



postfix: 5 3 + 20 10 / 8 6 - + *

©cs,Technion

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"

מבנה צומת:

value	
left	right

הערה: ע"י החלפת שורה #2 עם שורה #3 נקבל מימוש של סיור inorder.

תרגיל I: יש לשנות תוכנית זו כך שתחשב ערך של ביטוי אריתמטי.

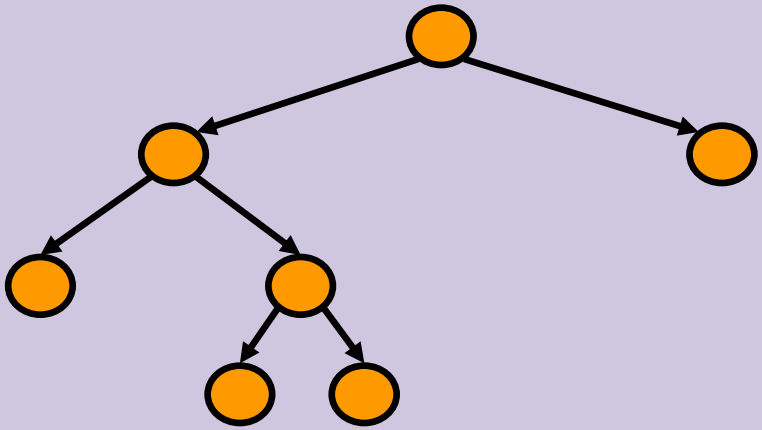
תרגיל II: יש לכתוב תוכנית זו ללא רקורסיה (תוך שימוש במחסנית).

©cs,Technion

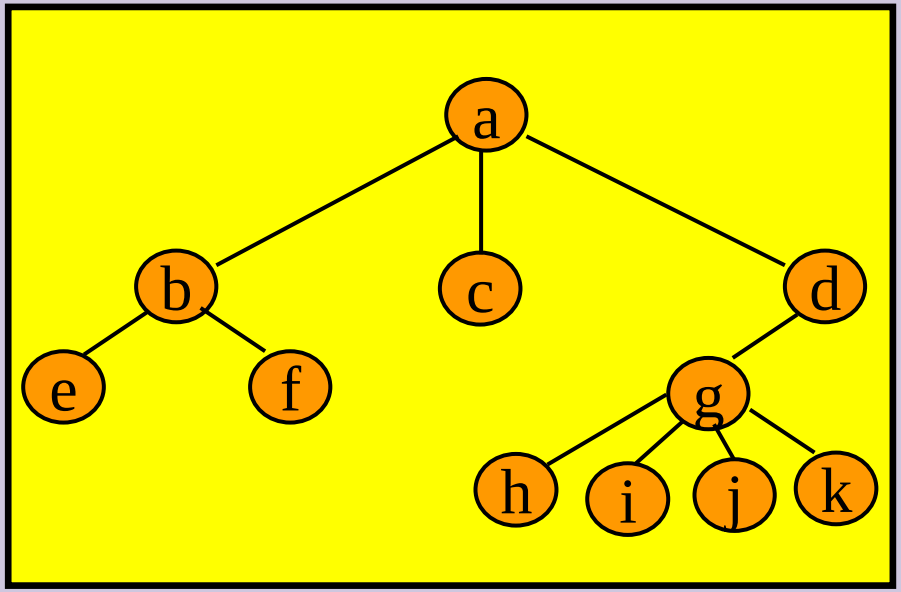
דוגמא לשימוש חביב

פונקציה רקורסיבית לחישוב גובה העץ (דוגמא לסיור postorder):

```
int height (NODE *T)
{           int L,R;
            if (T = = NULL)  return  –1
            else {
                        L = height(T → left);
                        R = height(T → right);
                        return 1 + max(L,R) ;
                  }
}
```



מימושים של עצים מסודרים

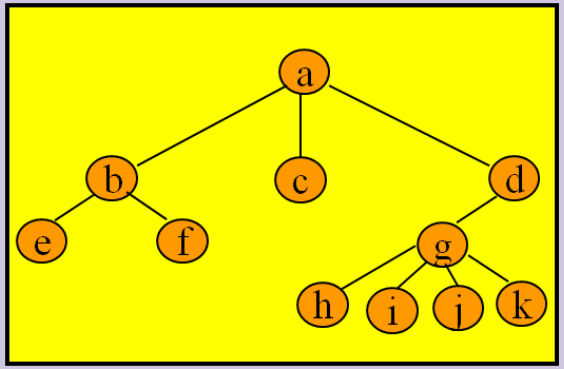


אם לכל צמת דרגה ≤ d:

value(s)			
child[0]	child[1]	…	child[d-1]

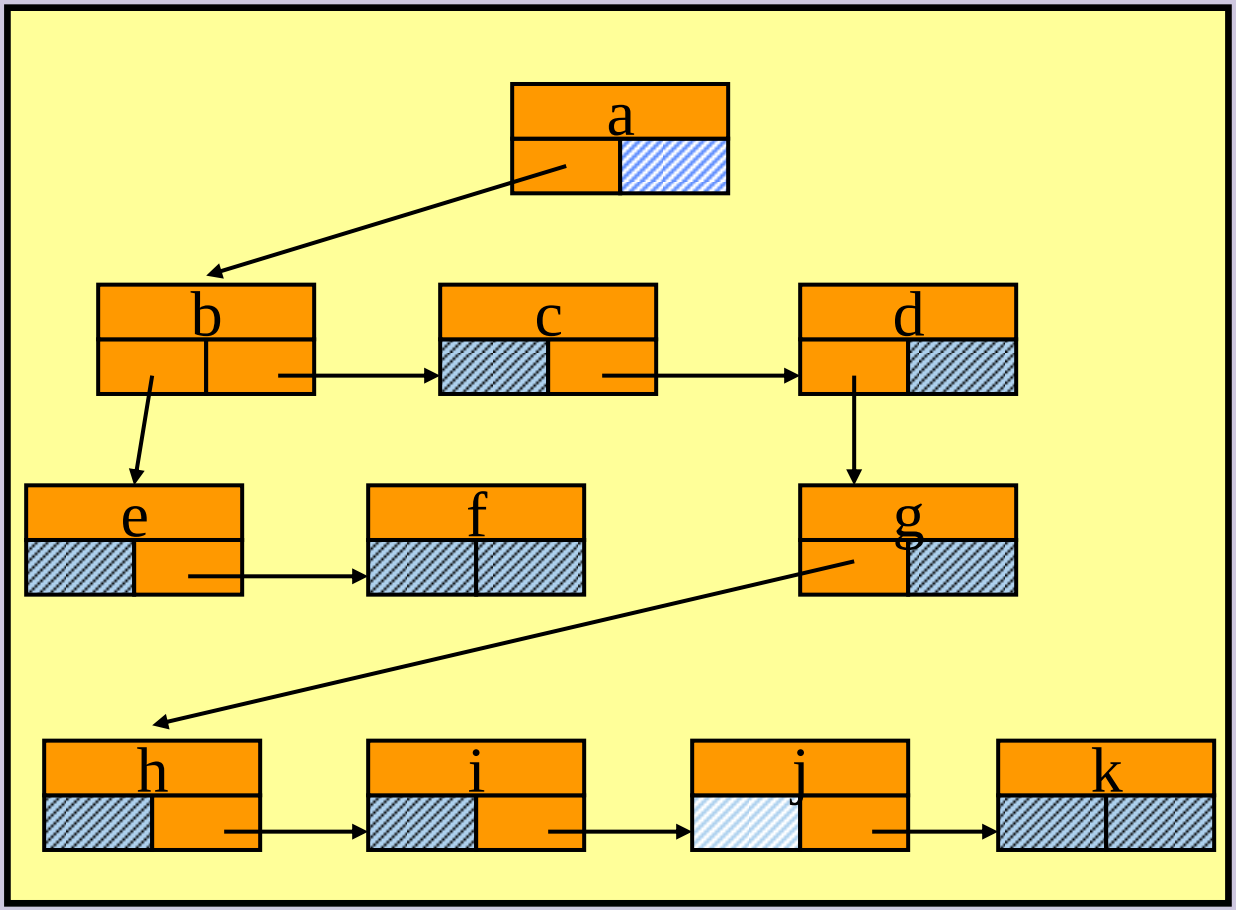
©cs,Technion

מימושים של עצים מסודרים



נתן לייצג עץ מדרגה כלשהי ע"י עץ בינרי:

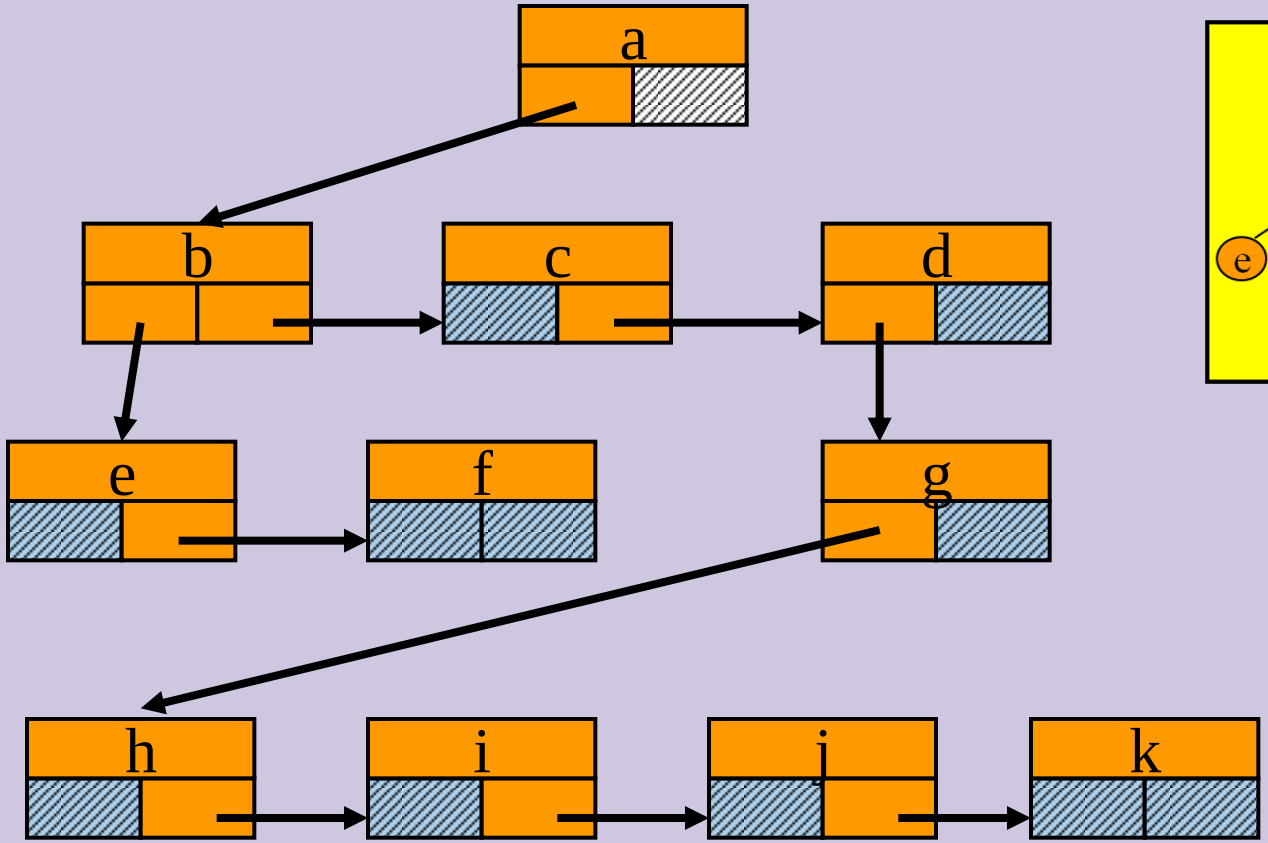
value(s)	
first-child	next-brother

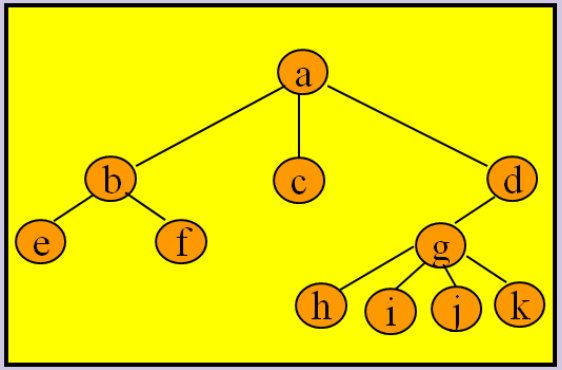


©cs,Technion

עצים מסודרים

מה הקשר בין גובה העץ המקורי וגובה העץ הבינרי ?





$h_{new} \le d\, h_{original}$, כאשר d הוא מספר הבנים המקסימלי בעץ המקורי.

©cs,Technion

מילון (Dictionary)

מילון מאחסן אוסף של רְשׁוּמוֹת מהטיפוס (מפתח, אינפורמציה).

המפתח שונה (בד"כ) מרשומה לרשומה.

אוסף המפתחות האפשריים מסומן ב-U. לדוגמא: מספרים שלמים.

פעולות:

- **אתחול:** יצירת מילון ריק. create(D)
- **חיפוש:** החזר מצביע לרשומה ב- D שמפתחה x או NULL. find(D,x)
- **הוספה:** הוסף ל- D רשומה שמפתחה x. insert(D,x,info)
- **הוצאה:** סלק מ- D רשומה שמפתחה x. delete(D,x)

כללים:

- x שייך לקבוצת המפתחות U.
- כל x מופיע לכל היותר פעם אחת במילון (בדר"כ).

©cs,Technion

מילון, מבנה חיפוש ועצי חיפוש

פעולות נוספות כאשר מוגדר סדר על U (למשל כאשר מפתח הוא מספר):

מינימום: החזר את המפתח המינימלי ב- D. min(D)

עוקב: החזר מצביע לאיבר במילון D בעל המפתח הקטן ביותר שגדול מ- x. next(D,x)

מילון + מינימום + עוקב נקרא = מבנה חיפוש

מטרה: לבצע את כל הפעולות בזמן O(log n) (במקרה הגרוע ביותר) כאשר n הוא מספר המפתחות הנמצאים במילון בזמן ביצוע הפעולה.

עצי חיפוש: היא משפחה של מימושים למבנה חיפוש.

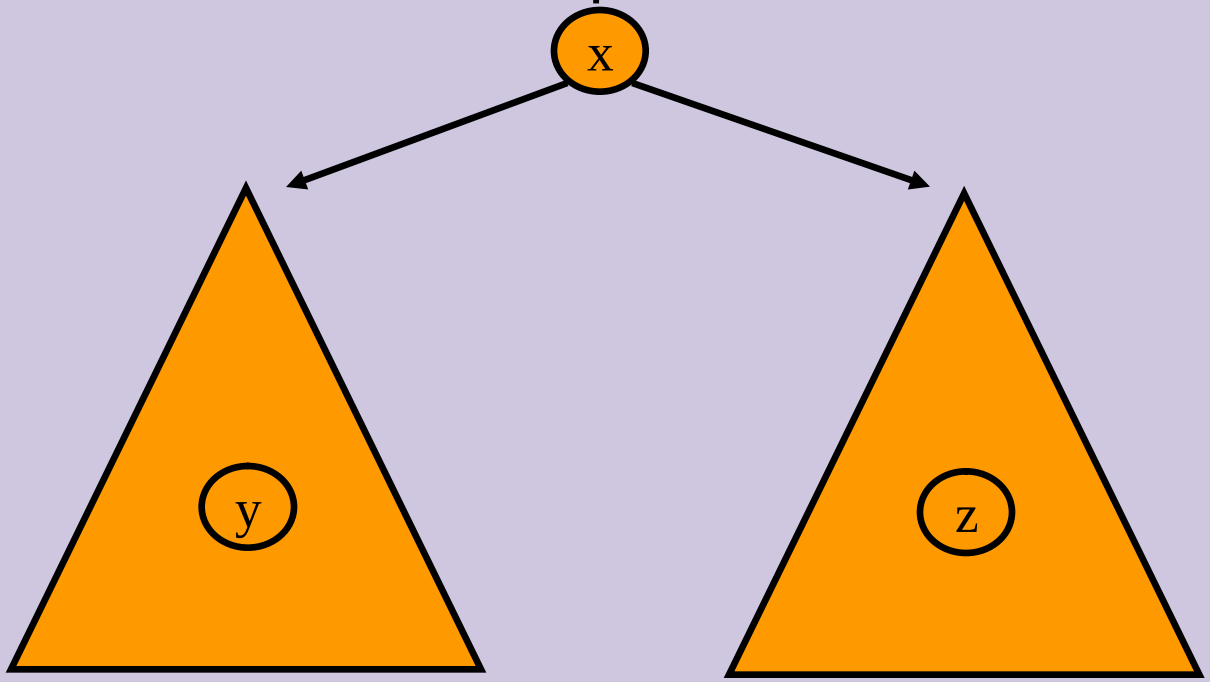
©cs,Technion

עץ בינרי כעץ חיפוש

נשתמש בעץ בינרי מכוון.

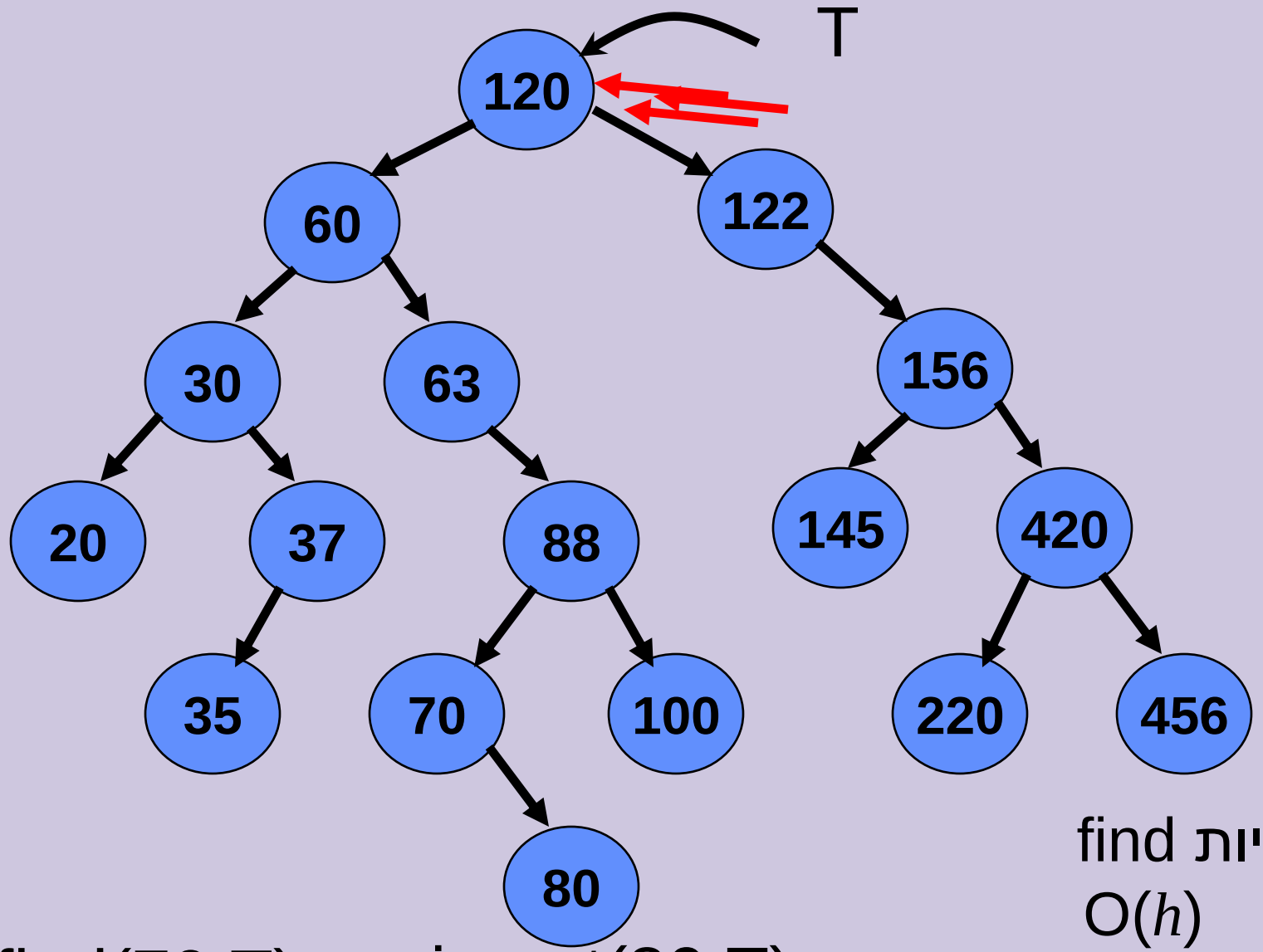
בכל צומת נאחסן רשומה אחת מתוך המילון (או מפתח ומצביע לאינפורמציה של הרשומה).

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.



$$y < x < z$$

©cs,Technion



find(70,T)

find(80,T)

insert(80,T)

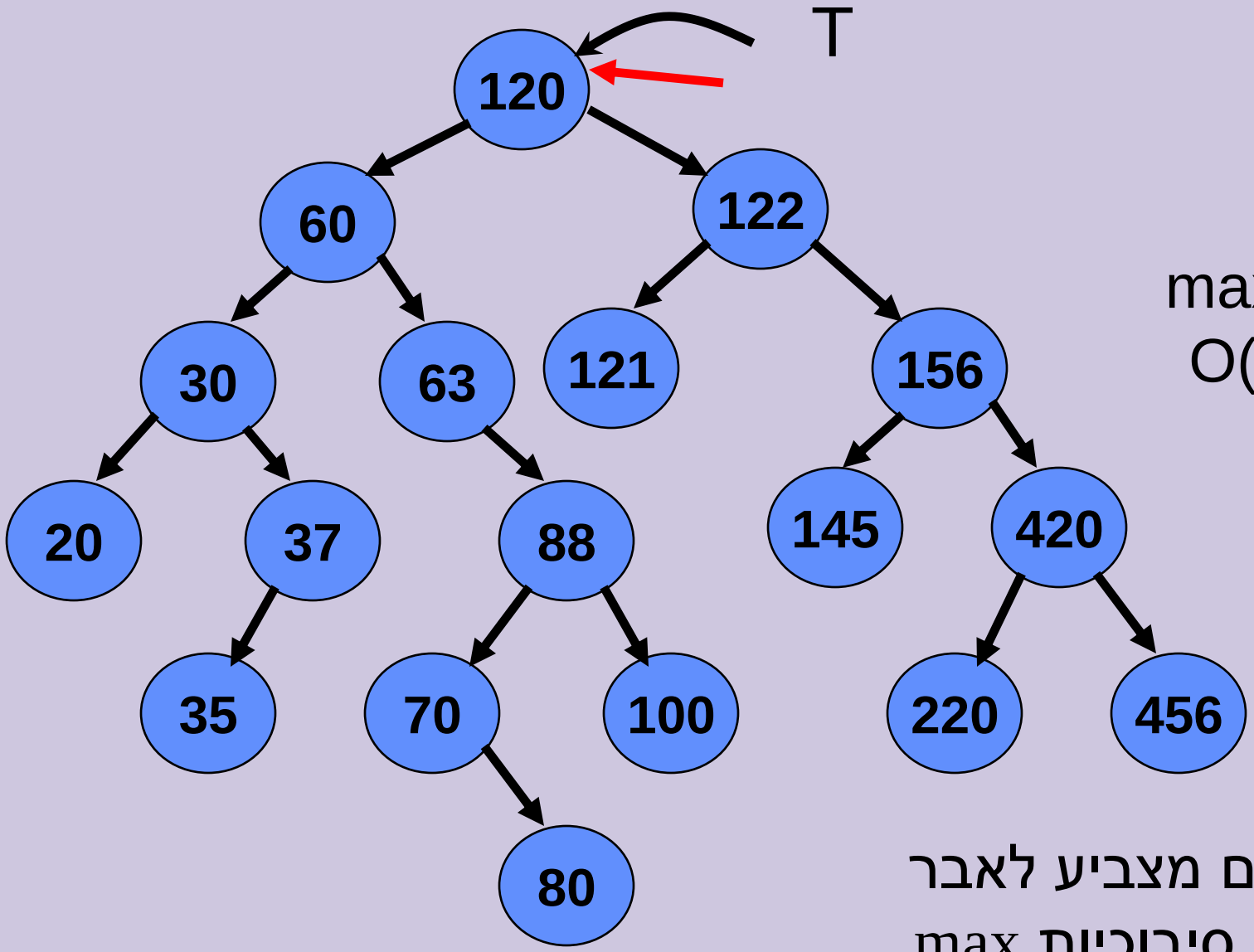
סיבוכיות find

O(h)

סיבוכיות insert

O(h)

©cs,Technion



סיבוכיות max
O(h)

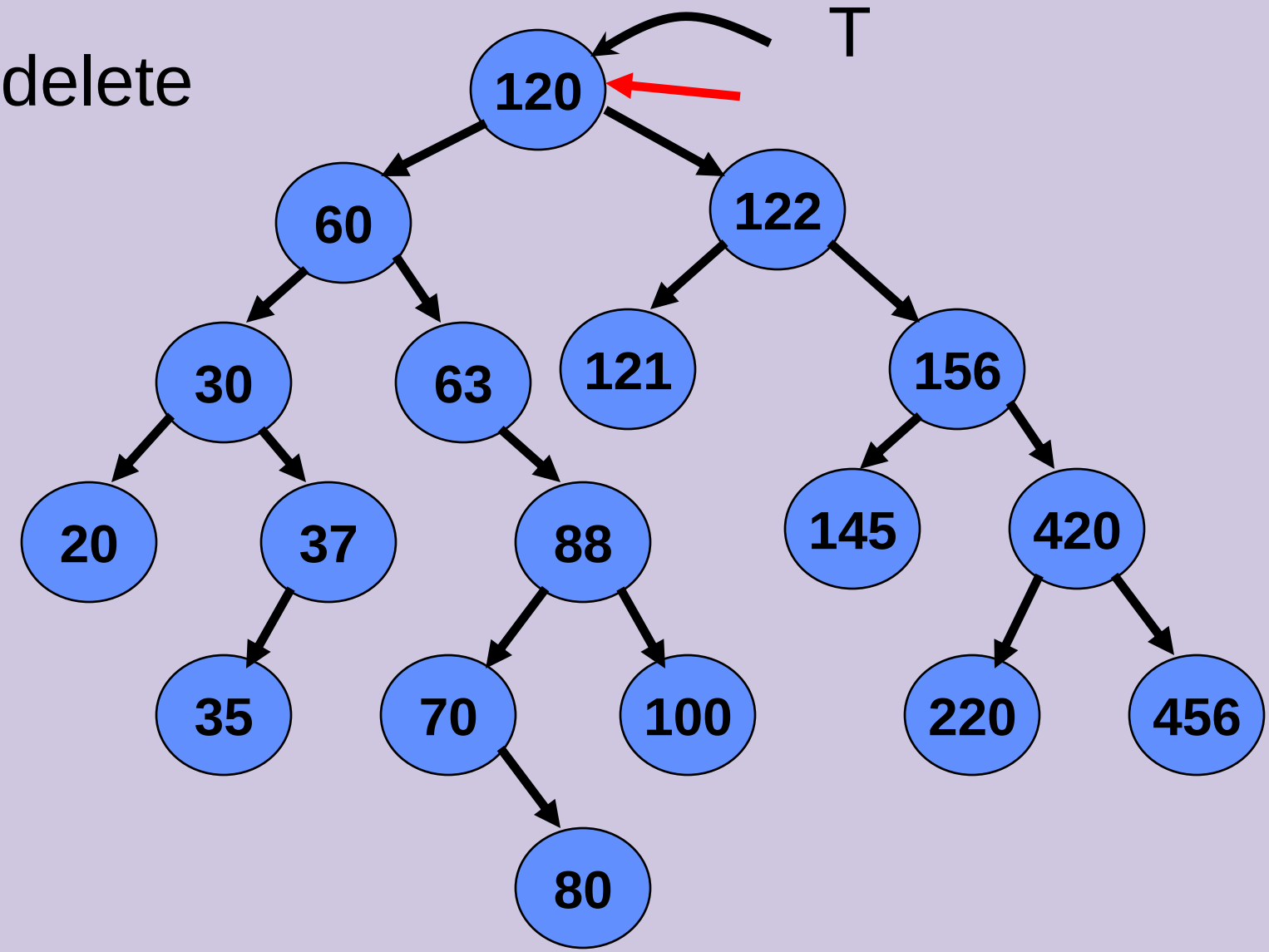
אם שומרים מצביע לאבר המכסימלי סיבוכיות max
O(1)

insert(121,T) max(T)

©cs,Technion

delete

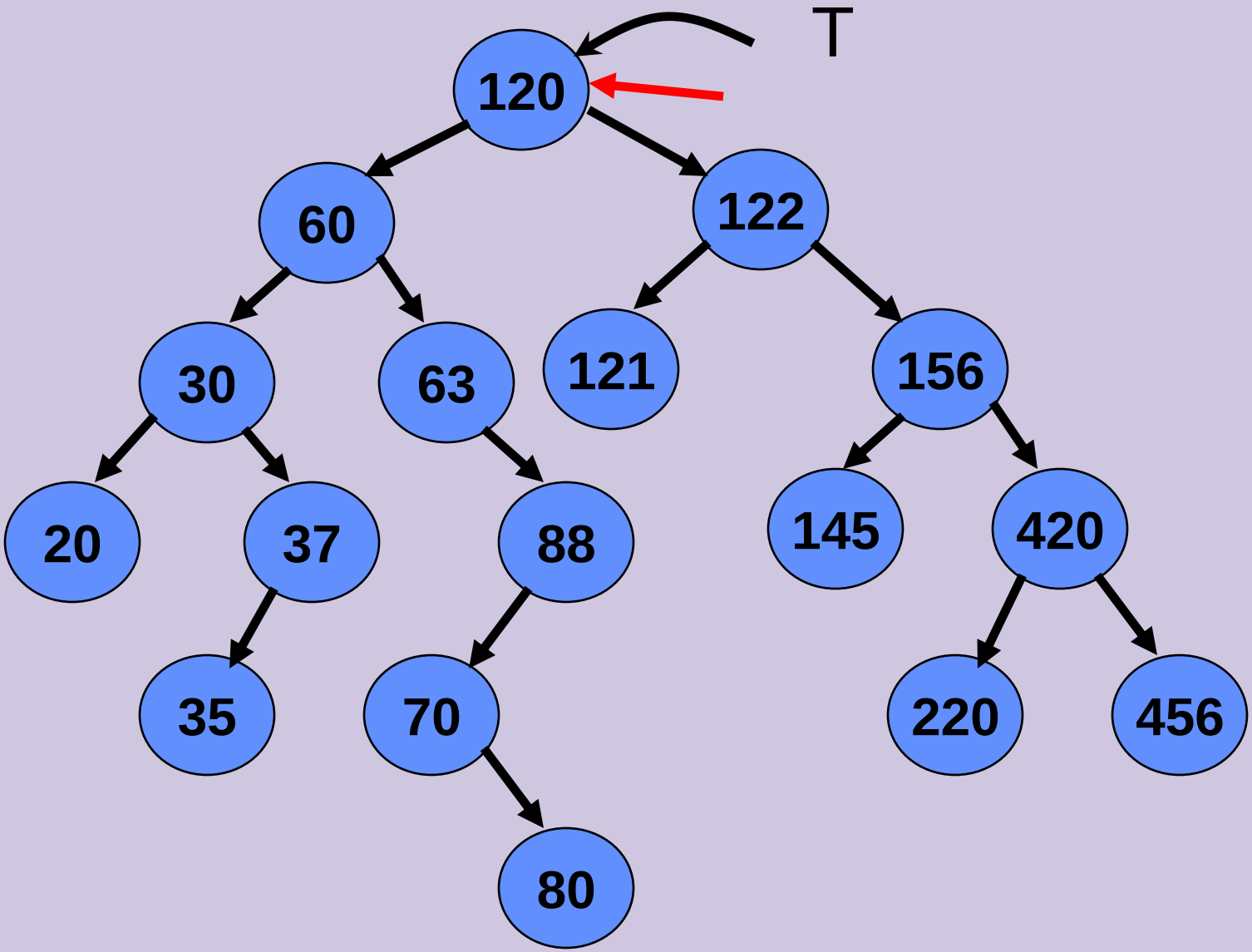
מקרא 1



delete(100,T)

©cs,Technion

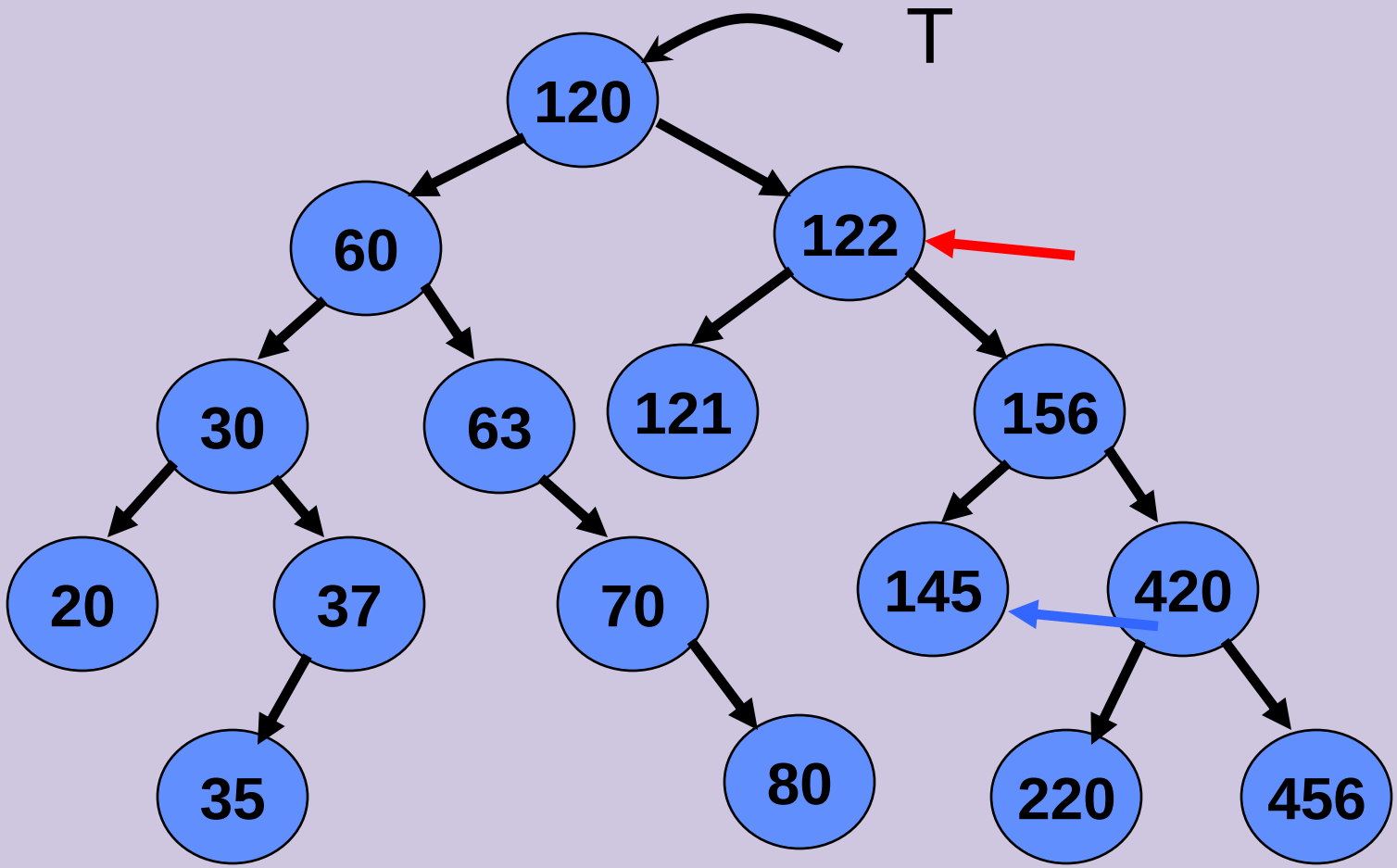
מקרא 2



delete(88,T)

©cs,Technion

מקרא 3



סיבוכיות delete

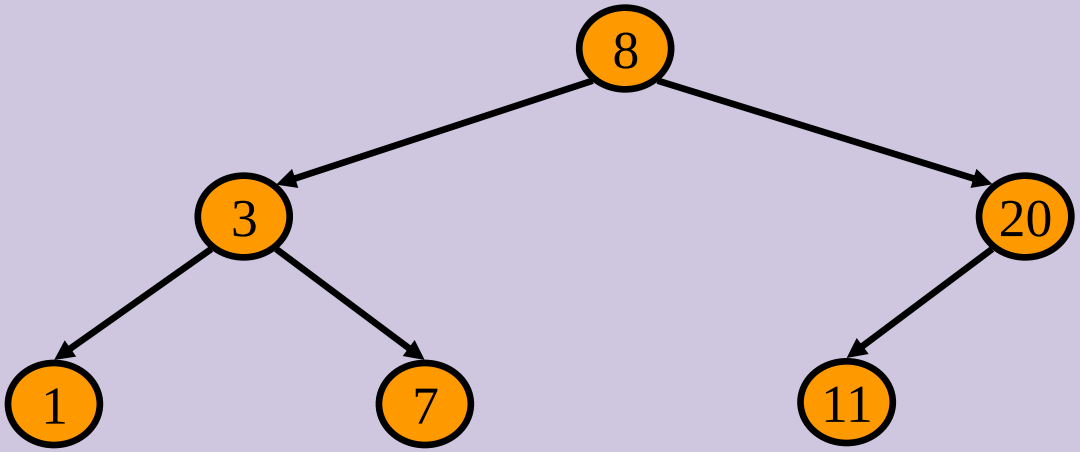
O(h)

delete(122,T)

©cs,Technion

עץ בינרי כעץ חיפוש

הערה: בציור מופיעים רק המפתחות ולא הרשומות במלואן.



אלגוריתם החיפוש: find(T,x)

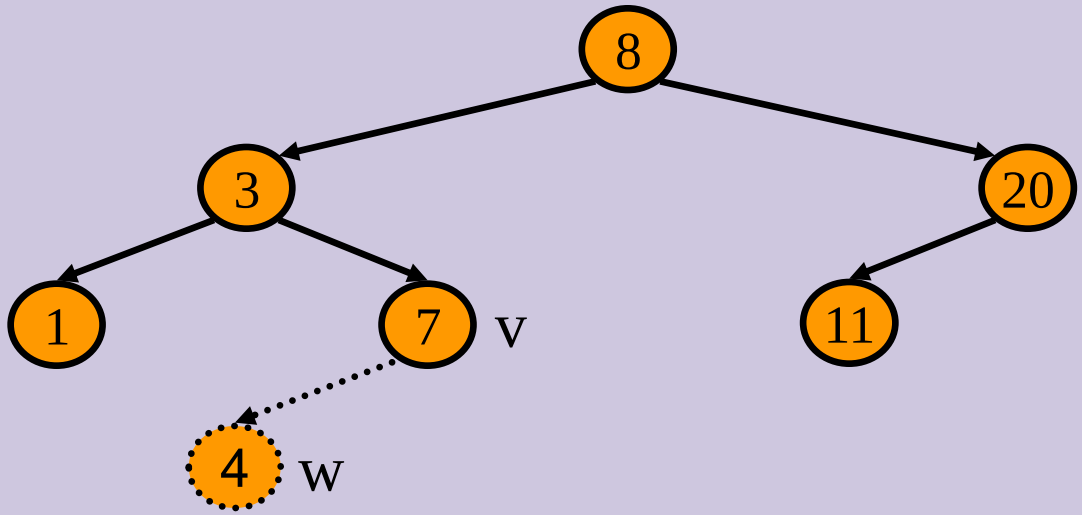
1. אם T ריק, דווח ש-x לא בעץ.
2. יהי y הערך שבשורש.
3. אם x = y, החזר מצביע לצומת המחזיק את x.
4. אם x <y, המשך את החיפוש בתת העץ השמאלי של T.
5. אחרת (כאשר x > y), המשך את החיפוש בתת העץ הימני של T.

©cs,Technion

הכנסה בעץ חיפוש

אלגוריתם הכנסה: 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.



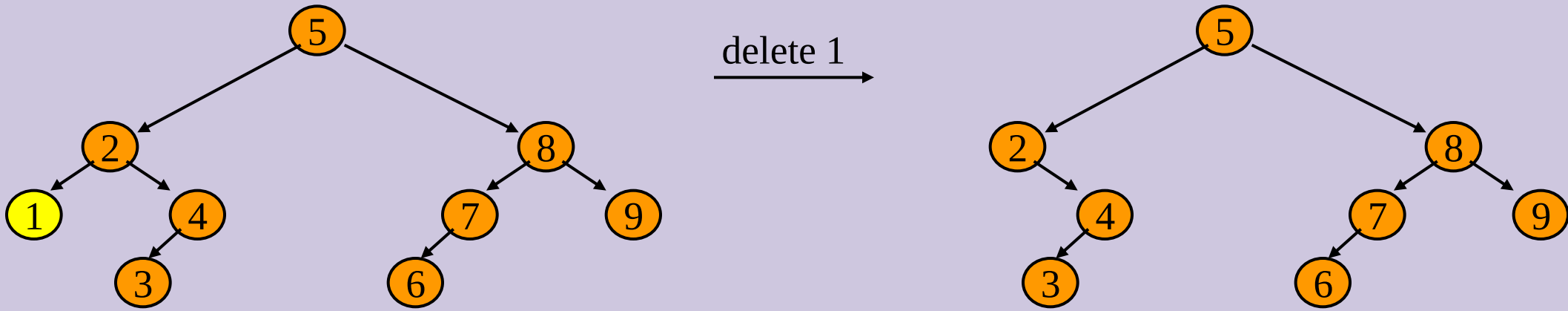
insert(T,4)

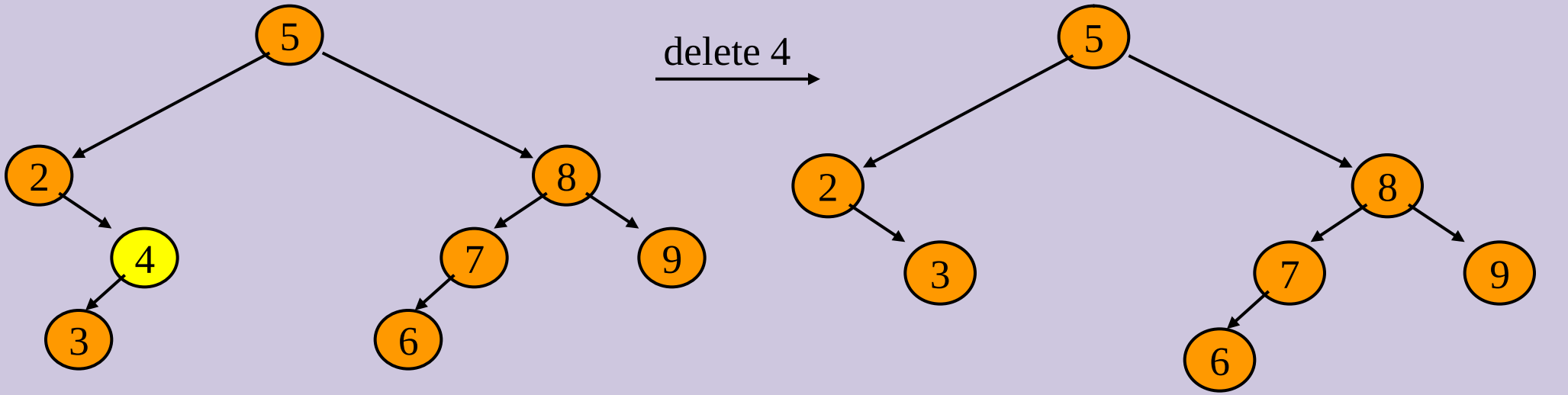
©cs,Technion

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.

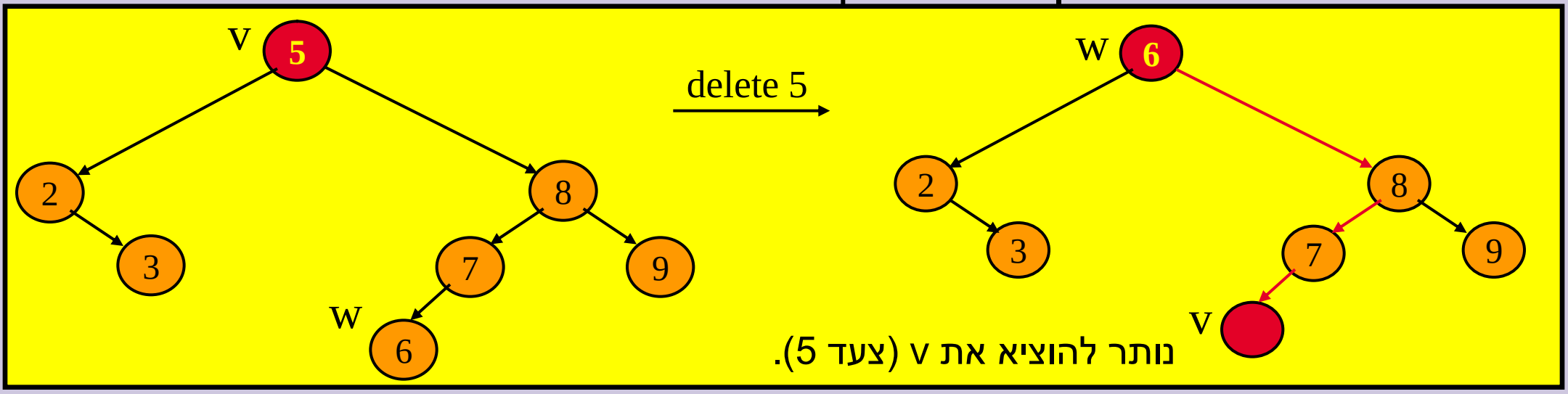




הוצאה מעץ חיפוש

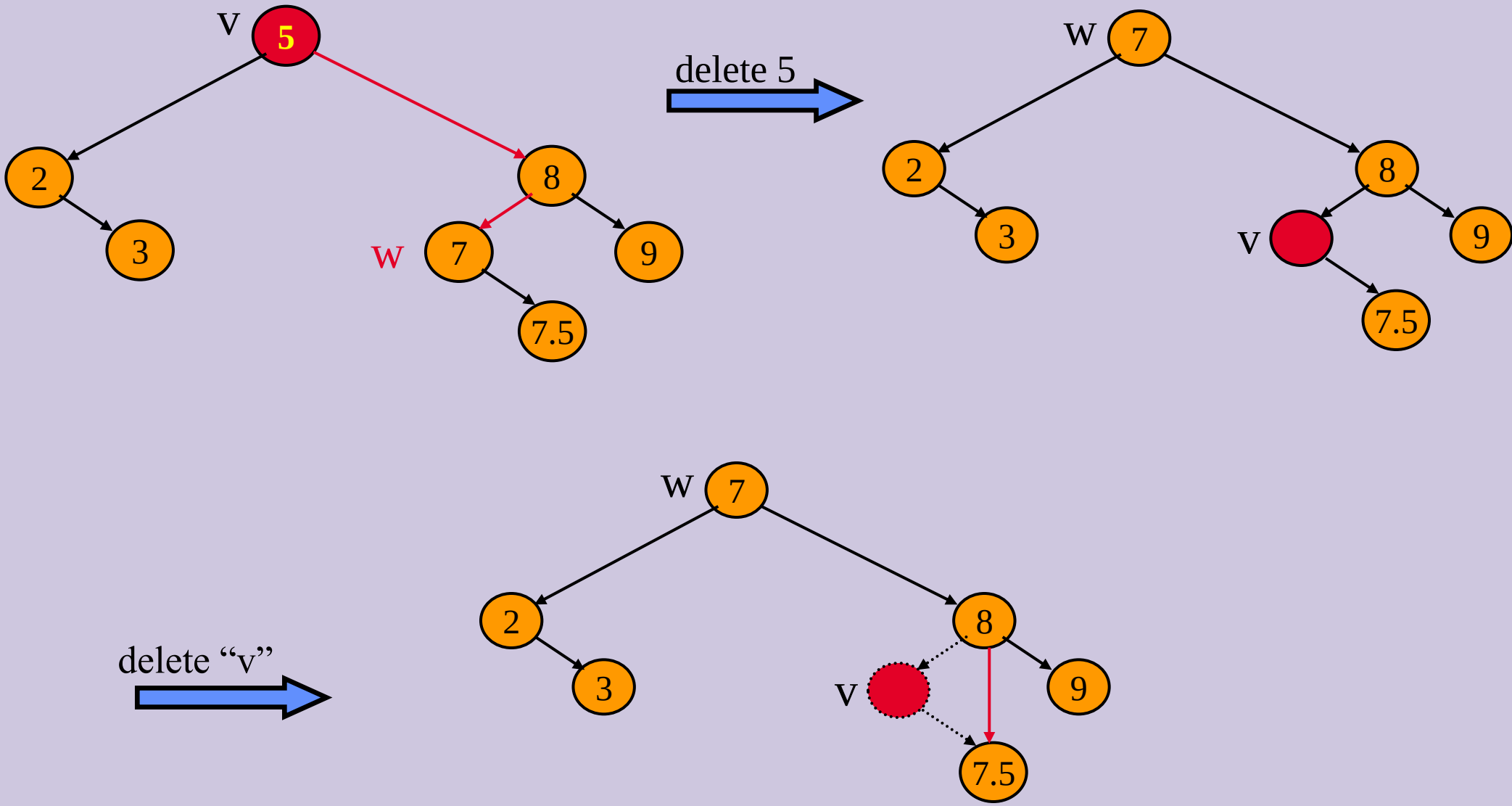
<u>אלגוריתם הוצאה</u>. יהי v צומת בעץ המיועד להוצאה.

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.



נותר להוציא את v (צעד 5).

דוגמא נוספת



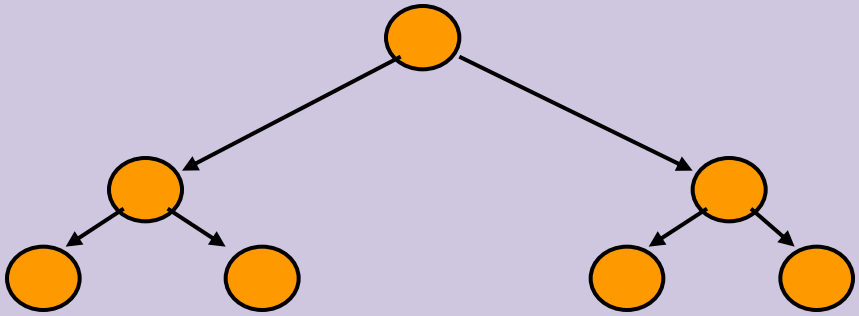
©cs,Technion

סיבוכיות הפעולות:

פעולה	function	סיבוכיות
אתחול	create(D)	$O(1)$
חיפוש	find(D,x)	$O(h)$
הוספה	insert(D,x,info)	$O(h)$
הוצאה	delete(D,x)	$O(h)$
מינימום	min(D)	$O(h) \rightarrow O(1)$
עוקב	next(D,x)	$O(h) \rightarrow O(1)$

©cs,Technion

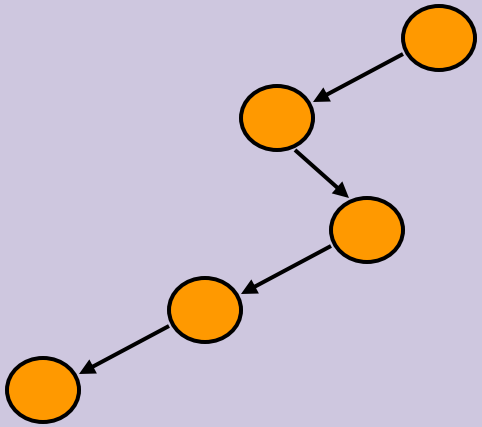
נתוח זמנים



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. $h = n - 1$

ומהו 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.

הגובה הממוצע מוגדר כדלקמן:

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

ניתן להראות שהגובה הממוצע הוא $O(\log n)$

כלומר בממוצע כל הפעולות מתבצעות בזמן $O(\log n)$.

ההוכחה (עמודים 254-258 בספר הלימוד) מושמטת.

נבחן טענה דומה אך קלה יותר להוכחה:

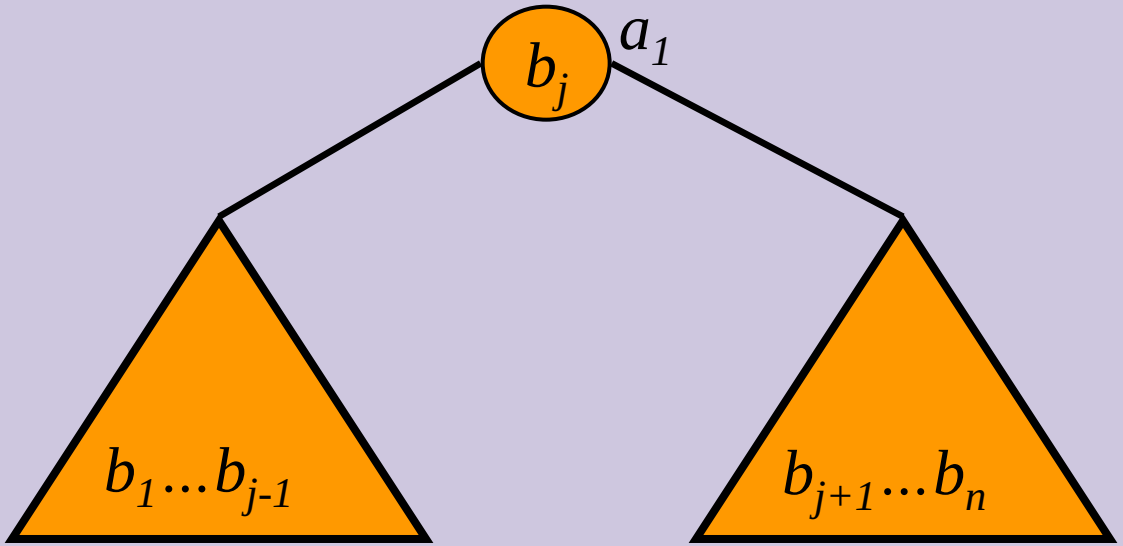
זמן בניה ממוצע של עץ חיפוש בינרי הוא $O(n \log n)$.

©cs,Technion

זמן בניה צפוי של עץ חיפוש בינרי

נחשב את זמן בנית עץ אקראי המתקבל מהכנסת פרמוטציה אקראית $a_1 \ldots a_n$ לעץ ריק.

נניח שסדר האיברים הוא $b_1 \ldots b_n$



נסמן ב- $T(n)$ את מספר ההשוואות הממוצע הדרוש לבניית עץ בן n צמתים.

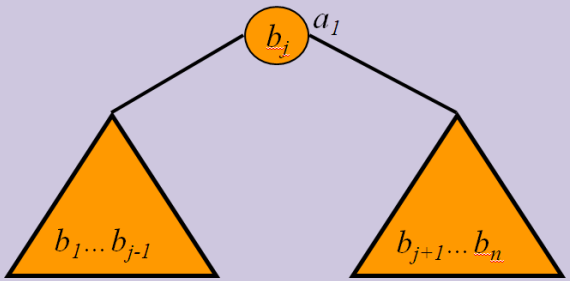
לפיכך $T(j-1)$ הוא מספר ההשוואות הממוצע הדרוש לבניית עץ בן $j-1$ צמתים

ו- $T(n-j)$ הוא מספר ההשוואות הממוצע הדרוש לבניית עץ בן $n-j$ צמתים.

משוואת ההפרשים המתאימה:

$$T(n) = \frac{1}{n}\sum_{j=1}^{n}\left[(n-1) + T(j-1) + T(n-j)\right] \qquad (T(0) = 0)$$

פתרון משואת ההפרשים



$$T(n) = \frac{1}{n}\sum_{j=1}^{n}\left[(n-1) + T(j-1) + T(n-j)\right] \qquad (T(0) = 0)$$

$$T(n) = \frac{1}{n}\sum_{j=1}^{n}\left[(n-1) + T(j-1) + T(n-j)\right] = (n-1) + \frac{2}{n}\sum_{j=1}^{n-1} T(j)$$

$$nT(n) = n(n-1) + 2\sum_{j=1}^{n-1} T(j)$$

$$-$$

$$(n-1)T(n-1) = (n-1)(n-2) + 2\sum_{j=1}^{n-2} T(j)$$

$$nT(n) - (n-1)T(n-1) = 2(n-1) + 2T(n-1)$$

$$\boxed{nT(n) = 2n - 2 + (n+1)T(n-1)}$$

©cs,Technion

פתרון משואת ההפרשים

$$nT(n) = 2n - 2 + (n+1)T(n-1)$$

$$\frac{T(n)}{n+1} = \frac{2}{n+1} - \frac{2}{n(n+1)} + \frac{T(n-1)}{n} < \frac{2}{n+1} + \frac{T(n-1)}{n}$$

נגדיר

$$g(n) = \frac{T(n)}{n+1}$$

$$\Rightarrow \quad g(n) < \frac{2}{n+1} + g(n-1)$$

$$g(n) < \frac{2}{n+1} + g(n-1) < \frac{2}{n+1} + \frac{2}{n} + g(n-2) < \cdots < 2H_{n+1} = O(\log n)$$

לכן

$$\boxed{\frac{T(n)}{n+1} = O(\log n) \Rightarrow T(n) = O(n \log n)}$$

©cs,Technion

עוד על המספר ההרמוני

$$H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + 0.57721 \ldots + o\left(\frac{1}{n}\right) = O(\ln n)$$

קבוע אוילר

על קבוע אוילר:

$$\gamma = \lim_{n \to \infty} \left(H_n - \ln n\right)$$

אוילר, שהגדיר את הקבוע, ציין כי הוא "ראוי להתייחסות רצינית"

$$\gamma = 0.5772156649 \quad 0153286060 \quad 6512090082 \quad 402431042 \quad \ldots$$

לא ידוע אם הוא רציונלי (אם כן, יש לו יותר מ-$10^{242,080}$ ספרות במכנה)

©cs,Technion