Lecture 3 of Geiger & Itai's slide brochure
www.cs.technion.ac.il/~dang/courseDS

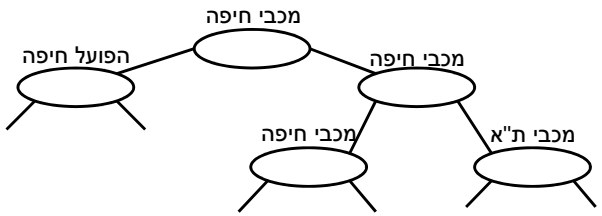
עצים ועצי חיפוש

חומר קריאה לשיעור זה

Chapter 5.5- Trees (91 – 97)
Chapter 13- Binary Search Trees (244 – 262)

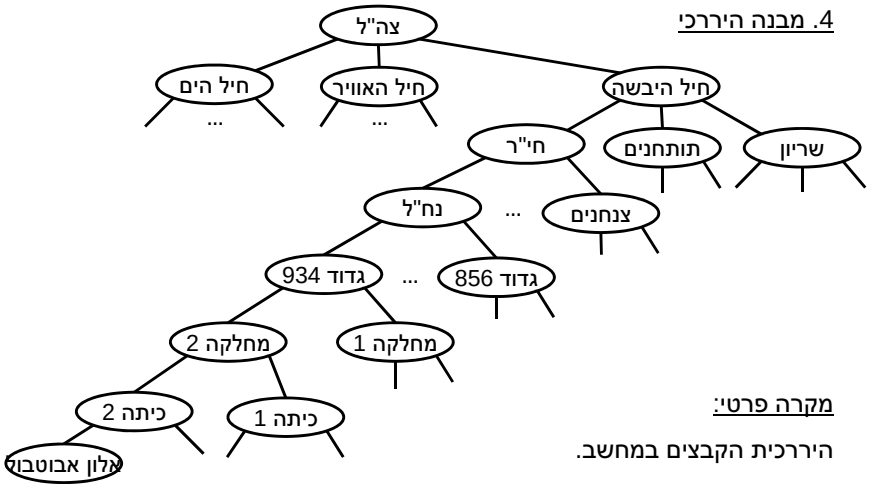
עצים

3. עץ מנצחים (גביע)



עצים

4. מבנה היררכי



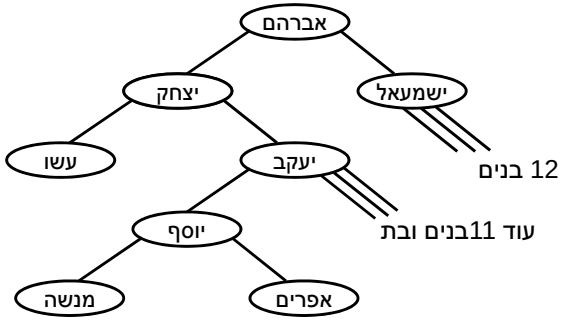
מקרה פרטי:

היררכית הקבצים במחשב.

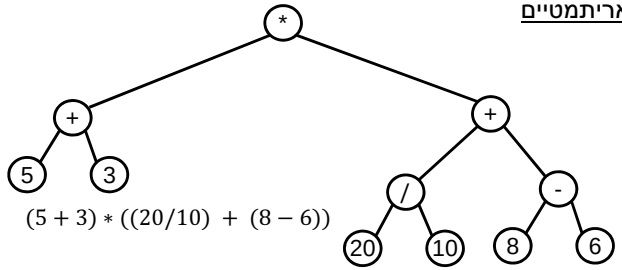
עצים

דוגמאות:

1. אילן יוחסין

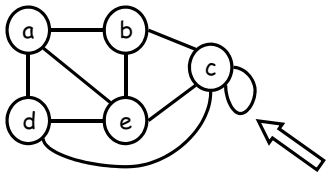


2. ביטויים אריתמטיים



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

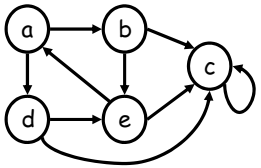
$V = \{a,b,c,d,e\}$

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

נסמן $n = |V|$ וכן $m = |E|$. בדוגמא: $n = 5, m = 9$.
מספר הקשתות m קטן בכל גרף מ- n^2 .

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: $n = 5, m = 9$.

<u>מסלול</u> (מכוון) בגרף (מכוון) (V,E) הוא סדרת צמתים $(v_1, v_2, \ldots, v_k)$ כך שלכל זוג צמתים עוקבים בסדרה, v_i, v_{i+1}, מתקיים כי (v_i, v_{i+1}) היא קשת ב-E.
המסלול נקרא **<u>מעגל</u>** (מכוון) אם $v_1 = v_k$ (לדוגמא, בגרף הנ"ל (a,d,e,a)).

<u>גרף התשתית</u> של גרף מכוון G הוא גרף לא-מכוון עם אותם צמתים כמו ב-G ואותן קשתות כמו ב-G אך ללא כוון. לדוגמא, הגרף בשקף הקודם הוא גרף התשתית של הגרף הנתון בשקף זה.

עצים מכוונים

<u>מקור</u> הוא צומת שאף קשת אינה מצביעה אליו.

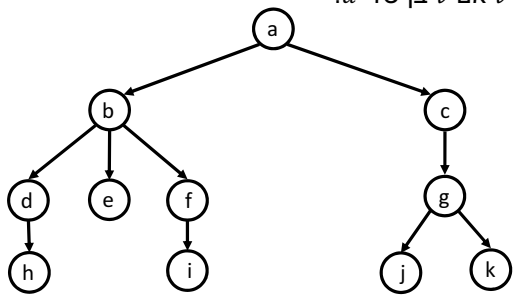
<u>עץ מכוון</u> הוא גרף מכוון ללא מעגלים (בגרף התשתית שלו) ואשר לו מקור אחד בלבד הנקרא **<u>שורש</u>**.

<u>הגדרות</u>

- v **בן** של u אם קיימת קשת מצומת u לצומת v.
- u **אב** של v אם v בן של u.

<u>דוגמאות</u>

f בן של b
b אב של e



עצים מכוונים

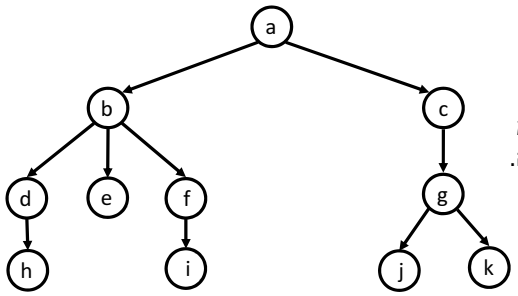
<u>הגדרות</u>

- v **<u>צאצא</u>** של u אם קיים מסלול מכוון מצומת u ל- v.
- u **<u>אב-קדמון</u>** של v אם v צאצא של u.
- **<u>תת-עץ של G ששורשו v</u>** הוא עץ מכוון שצמתיו הם v עצמו וכל הצאצאים של v, והקשתות שלו הן הקשתות המחברות צמתים אלו ב- G.
- **<u>דרגת צומת v</u>** היא מספר הבנים של v.
- **<u>עלה</u>** הוא צומת ללא בנים.
- **<u>צומת פנימי</u>** הוא צומת שאינו עלה.

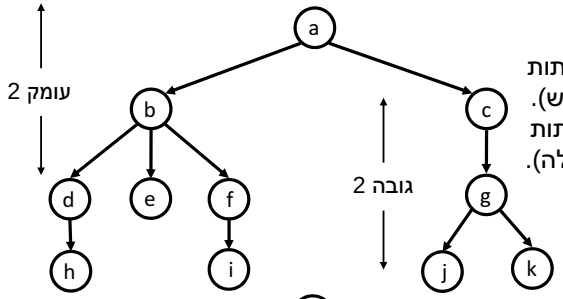
<u>דוגמאות</u>

g צאצא של a.
b אב-קדמון של h.
תת העץ ששורשו g מכיל 3 צמתים ושתי קשתות.

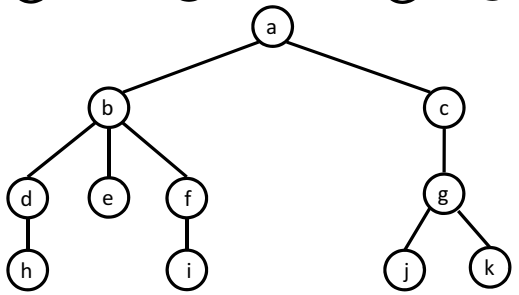
דרגת a היא 2.
h הוא עלה.



עצים מכוונים

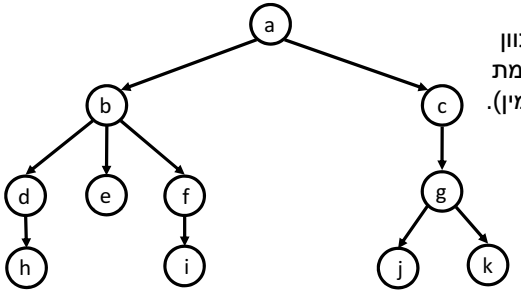


עומק של צומת v הוא מספר הקשתות משורש העץ אל v (המרחק מהשורש).
גובה של צומת v הוא מספר הקשתות מ-v לצאצא הרחוק ביותר של v (עלה).
גובה העץ הוא הגובה של שורשו.

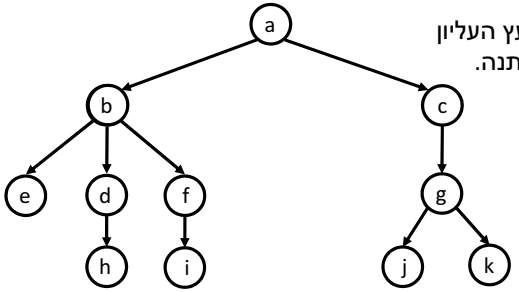


הערה: לעיתים נשמיט את החצים מתוך הבנה שכוון הקשתות כלפי מטה. כמו כן לרוב נאמר עץ במקום עץ מכוון.

עצים מסודרים



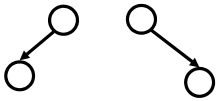
עץ מסודר הוא עץ מכוון שבו הבנים של כל צומת מסודרים (משמאל לימין).



למשל עץ זה שונה מהעץ העליון בגלל שסדר הבנים השתנה.

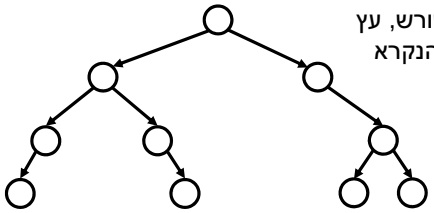
עצים בינריים

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

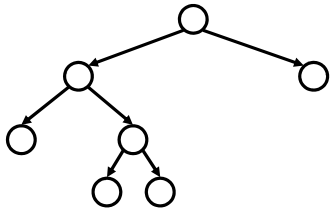


הגדרה רקורסיבית: עץ בינרי הוא מבנה

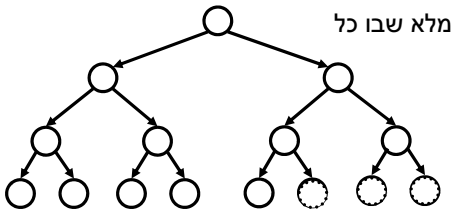
1. ריק (ללא צמתים), או
2. מורכב משלושה חלקים: צומת הנקרא שורש, עץ בינרי הנקרא תת-עץ שמאלי, ועץ בינרי הנקרא תת-עץ ימני.



עצים בינריים מלאים ושלמים



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



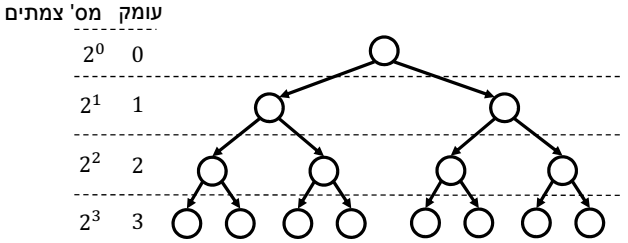
עץ בינרי שלם (complete): עץ בינרי מלא שבו כל העלים באותו עומק.

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

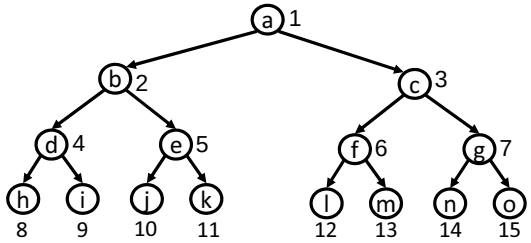
תכונות עצים בינריים שלמים

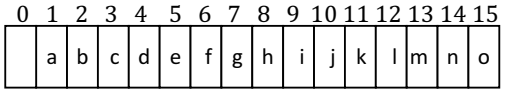
בעץ בינרי שלם בעל n צמתים, L עלים, וגובה h:

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



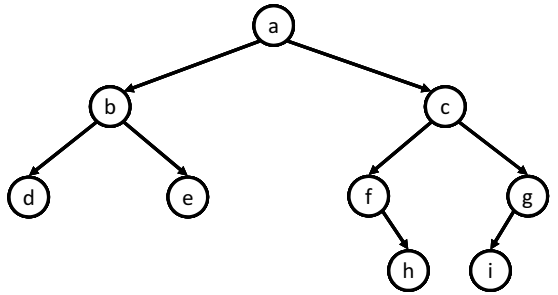
ייצוג לעצים בינריים שלמים





- בן שמאלי של צומת i נמצא ב- $2i$.
- בן ימני של צומת i נמצא ב- $2i+1$.
- אבא של צומת i נמצא ב- $\lfloor i/2 \rfloor$.

סיור בעצים בינריים



preorder
1. בקר בשורש
2. סייר בתת העץ השמאלי
3. סייר בתת העץ הימני

פלט: a b d e c f h g i

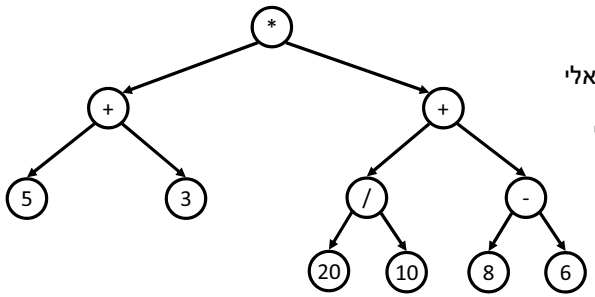
postorder
1. סייר בתת העץ השמאלי
2. סייר בתת העץ הימני
3. בקר בשורש (חשוב ביטויים אריתמטיים)

פלט: d e b h f i g c a

inorder
1. סייר בתת העץ השמאלי
2. בקר בשורש
3. סייר בתת העץ הימני

פלט: d b e a f h c i g

חישוב והמרה של ביטויים אריתמטיים



inorder
1. סייר בתת העץ השמאלי
2. בקר בשורש (הדפס)
3. סייר בתת העץ הימני

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

infix

postorder
1. סייר בתת העץ השמאלי
2. סייר בתת העץ הימני
3. בקר בשורש (חשב את ערך הביטוי או הדפס).

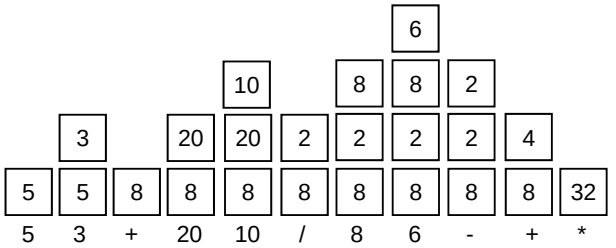
5 3 + 20 10 / 8 6 - + *

postfix

חישוב ביטוי postfix באמצעות מחסנית

אלגוריתם לחישוב ביטוי postfix

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



מימוש פרוצדורת postorder

מבנה הצומת:

value	
left	right

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

סיור Postorder:

```
void postorder (NODE *T){
    if (T = = NULL) return;
    postorder( T → left);      /*1*/
    postorder( T → right);     /*2*/
    "visit";                   /*3*/
}
```

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).

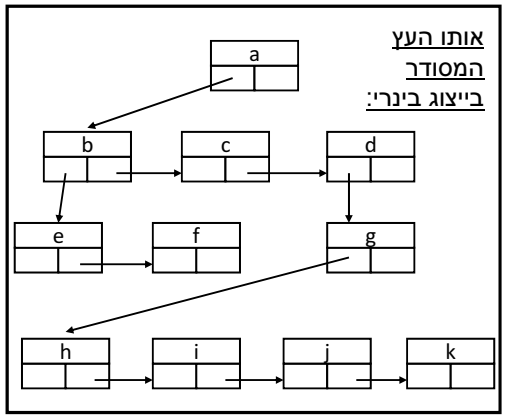
עצים מסודרים

אם לכל צמת דרגה $d \geq$:

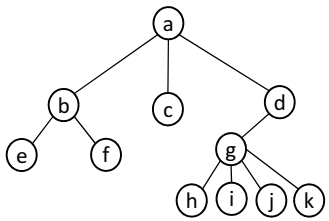
value(s)			
child[0]	child[1]	…	child[d-1]

value(s)	
first-child	next-brother

נתן לייצג עץ מדרגה כלשהי בצורה הבאה:



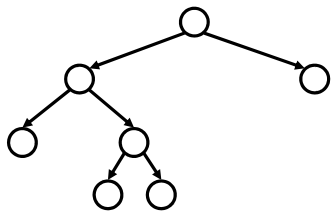
העץ המסודר:



מימוש עצים בינריים

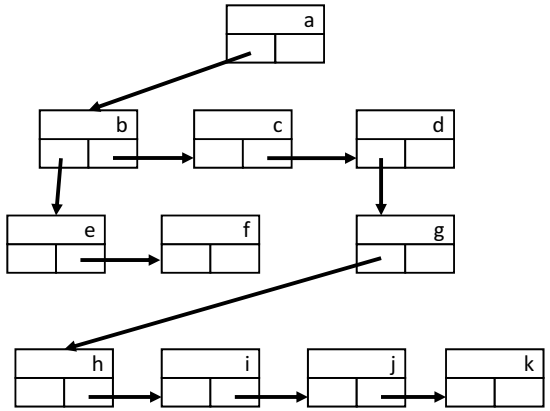
פונקציה רקורסיבית לחישוב גובה העץ (דוגמא לסיור postorder):

```
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) ;
        }
}
```



עצים מסודרים

מה הקשר בין גובה העץ המקורי, h_{old}, וגובה העץ הבינרי, h_{new} ?



$h_{new} \leq d \cdot h_{old}$ כאשר d הוא מספר הבנים המקסימלי בעץ המקורי.

מילון (Dictionary)

מילון מאחסן אוסף של רשומות מהטיפוס (מפתח, אינפורמציה). המפתח שונה (בד"כ) מרשומה לרשומה. אוסף המפתחות האפשריים מסומן ב-U. מפתחות אפשריים לדוגמא: מספרים שלמים.

מילון מוגדר ע"י הפעולות הבאות:

create(D) יצירת מילון ריק.

find(D,x) החזר מצביע לרשומה שמפתחה x ב- D (Null אם אין).

insert(D, $x, info$) הוסף ל-D רשומה שמפתחה x והמידע הנוסף שלה הוא $info$.

delete(D,x) הסר מ-D רשומה שמפתחה x.

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

מילון ועצי חיפוש

פעולות נוספות כאשר מוגדר סדר על U (למשל כאשר מפתחות הם מספר)

min(D) החזר את המפתח המינימלי ב-D.

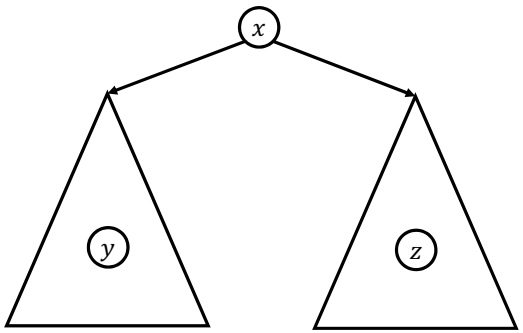
next(D,x) החזר מצביע לאיבר במילון D בעל המפתח הקטן ביותר גדול מ-x.

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 הנתונים "מילון" כאשר מוגדר סדר על קבוצת המפתחות U.

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.

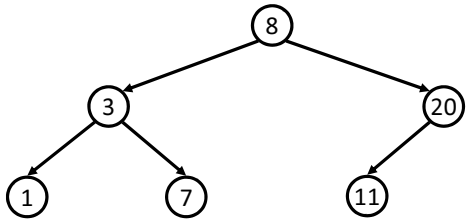


$y < x < z$

עץ בינרי כעץ חיפוש

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 $y < x$), המשך את החיפוש בתת העץ הימני של T.



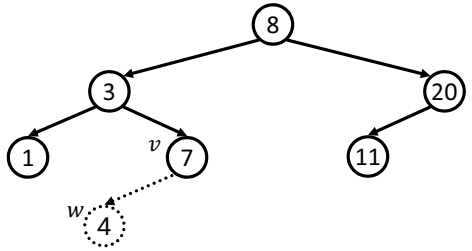
הערה: בציור מופיעים רק המפתחות ולא הרשומות במלואן.

הכנסה בעץ חיפוש

אלגוריתם הכנסה: insert$(T, x, info)$:

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 ומידע $info$ כבן שמאלי של v.
5. אחרת (כאשר $x > y$), הוסף צומת w עם מפתח x ומידע $info$ כבן ימני של v.

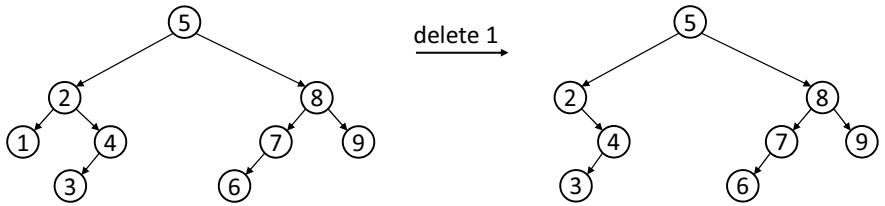
insert$(T, 4)$

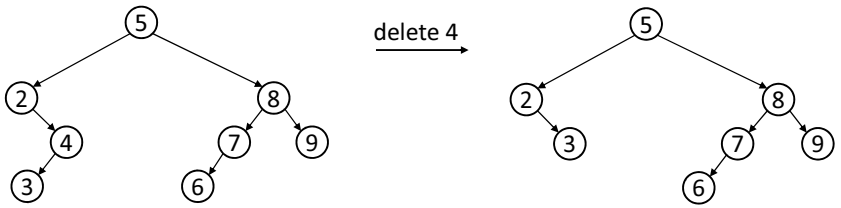


הוצאה מעץ חיפוש

אלגוריתם הוצאה: Delete(x): יהי v הצומת בעץ המיועד להוצאה.

1. אם v עלה, סלק אותו.
2. אם ל-v בן יחיד, תן לאבא של v להצביע על הבן.

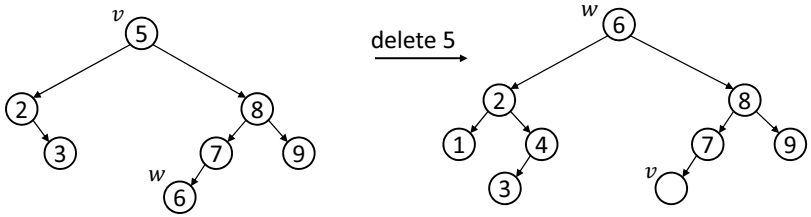




הוצאה מעץ חיפוש

אלגוריתם הוצאה: Delete(x): יהי v הצומת בעץ המיועד להוצאה.

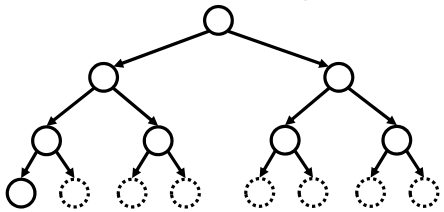
1. אם v עלה, סלק אותו.
2. אם ל-v בן יחיד, תן לאבא של v להצביע על הבן.
3. אחרת: יהי w הצומת העוקב ל-v בסדר inorder. (זהו הצומת המכיל את הערך הבא אחרי הערך שב- v. כלומר הצומת המתקבל ע"י פניה אחת ימינה ואח"כ כל הדרך שמאלה. שימו לב שלצומת w בן אחד לכל היותר).
4. החלף בין צומת v וצומת w.
5. כעת יש ל-v לכל היותר בן אחד. המשך בצעד 1 או 2 כנדרש.



נותר להוציא את v (צעד 5).

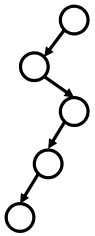
ניתוח זמנים

<u>זמן חיפוש/הכנסה/הוצאה הוא לינארי בגובה העץ.</u>



מהו גובה העץ?

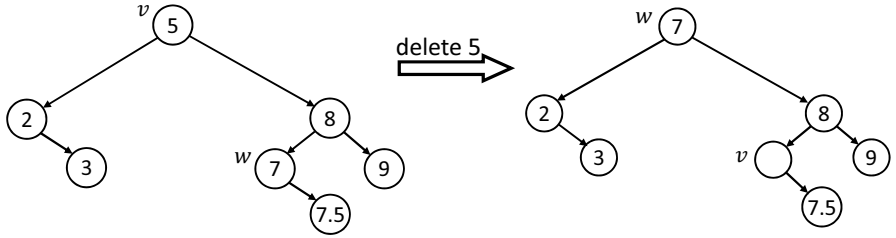
<u>מקרה טוב:</u>
עץ כמעט שלם מגובה h:
$n \geq 2^h$
לכן $\log_2 n \geq h$.

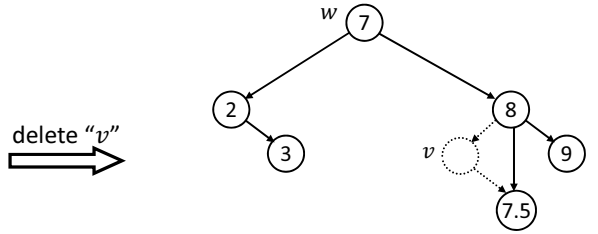


<u>מקרה גרוע:</u>
"שרוך" - עץ הנראה כרשימה ליניארית:
$h = n - 1$

ומהו הגובה הממוצע ?

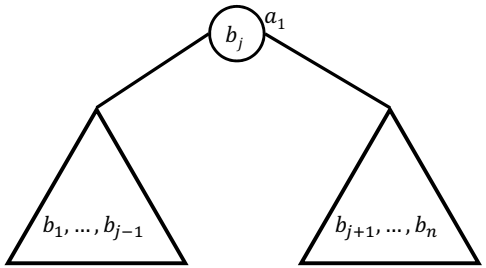
דוגמא נוספת





זמן בניה צפוי של עץ חיפוש בינרי

נחשב את זמן בנית עץ אקראי המתקבל מהכנסת פרמוטציה אקראית $a_1, \dots, a_n$ לעץ ריק.
נניח שסדר האיברים הוא $b_1 < b_2 < b_3 < \cdots < b_n$.



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

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

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.

<u>הגובה הממוצע מוגדר כדלקמן:</u>

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

ניתן להראות שהגובה הממוצע שייך לקבוצה $O(\log n)$ - כלומר בממוצע כל הפעולות מתבצעות בזמן $O(\log n)$.
ההוכחה (עמודים 258-254 בספר הלימוד) מושמטת. נבחן טענה דומה אך קלה יותר להוכחה: זמן בניה ממוצע של עץ חיפוש בינרי הוא $O(n \log n)$.

פתרון משוואת ההפרשים

הנוסחה שקיבלנו בשקף הקודם:

$$T(n) = \frac{1}{n}\sum_{j=1}^{n}[(n-1) + T(j-1) + T(n-j)] = n - 1 + \frac{2}{n}\sum_{j=0}^{n-1} T(j)$$

אם נכפול את שני האגפים ב-n, נקבל:

$$nT(n) = n(n-1) + 2\sum_{j=0}^{n-1} T(j)$$

באותו אופן, אם נציב $n-1$ בנוסחה, נקבל:

$$(n-1)T(n-1) = (n-1)(n-2) + 2\sum_{j=0}^{n-2} T(j)$$

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

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

פתרון משוואות ההפרשים (המשך)

בשקף הקודם קיבלנו: $\boxed{nT(n) = 2n - 2 + (n+1)T(n-1)}$

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

$$< \frac{2}{n+1} + \frac{2}{n} + \frac{T(n-2)}{n-1} < \cdots < 2\sum_{j=1}^{n}\frac{1}{j+1} + \frac{T(0)}{1}$$

$$= 2(H_{n+1} - 1) + 0 \le 2H_{n+1} = O(\log(n+1)) = O(\log n)$$

<u>תזכורת:</u> H_n נקרא המספר ההרמוני ה- n
והוא מקיים $H_n = \sum_{i=1}^{n} 1/i = O(\log n)$

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

לסיכום, נכפול את שני האגפים ב-$n+1$ ונקבל:

$$T(n) = O(n\log n)$$

המספר ההרמוני, H_n

<u>הערה:</u>
המספר ההרמוני, H_n, אותו ראינו בהרצאה 1, מקיים:
$H_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \ln(n) + 0.57721 \ldots + o\left(\frac{1}{n}\right)$
כאשר ... 0.57721 נקרא <u>קבוע אוילר (Euler)</u>.

נזכיר כי בהרצאה 1 ראינו חסמים (עליון ותחתון) לוגריתמיים על H_n - ראינו כי:

- $H_n \le 1 + \ln(n)$
- $H_n \ge \ln(n+1)$

לחסמים אלו טעות חיבורית (קטנה), אך בכל מקרה הם מראים כי מתקיים $H_n = \Theta(\log n)$.