

## הרצאה 6

התכנית  $\text{Inorder}(T)$  עוברת על כל הצמתים בעץ ומדפיסה את המפתחות שלהם. הסדר הוא: כשמגיעים לצומת  $x$ , קודם מדפיסים את תת העץ השמאלי של  $x$  אחר כך מדפיסים את  $x$  ואחר כך מדפיסים את תת העץ הימני של  $x$ .

כדי להדפיס את תת העץ השמאלי של  $x$  מבצעים קריאה רקורסיבית ל-  $\text{Inorder}(T_{\text{left}(x)})$

התכנית נראית כך:

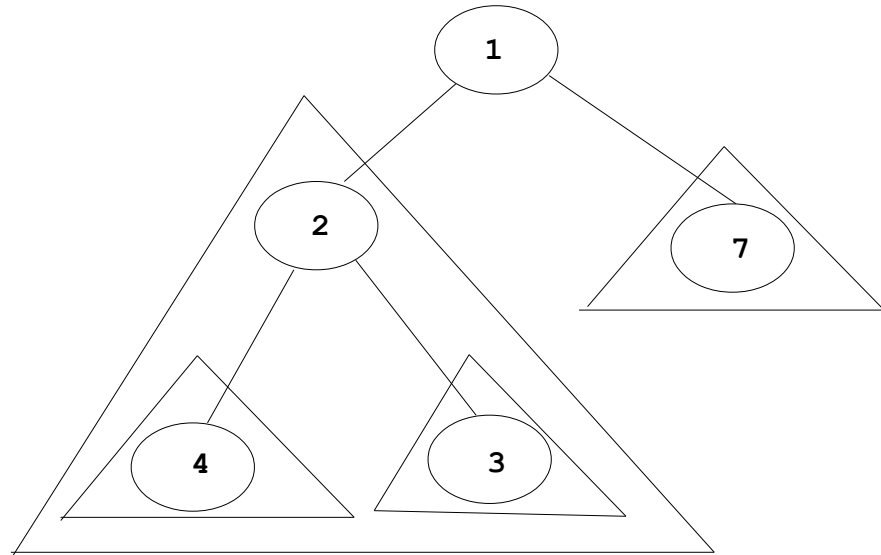
$\text{Inorder}(T)$

```
If (root(T)==NULL) return
x=root(T)
Inorder( $T_{\text{left}(x)}$ )
print key(x)
Inorder( $T_{\text{right}(x)}$ )
```

שימו לב ש-  $T_{\text{NULL}}$  מסמן עץ ריק.

חשוב להבין שקריאה רקורסיבית היא כמו קריאה לפונקציה אחרת והפקודה  $\text{print key}(x)$  תבצע רק אחרי שהקריאה ל-  $\text{Inorder}(T_{\text{left}(x)})$  (וכל הקריאות הרקורסיביות הנגזרות ממנה) תסתיים.

לדוגמה נבצע מעקב של התכנית הרקורסיבית על העץ שמתואר בציור הבא:



**Inorder( $T_1$ )**

**x=1**

**Inorder( $T_2$ )**

**x=2**

**Inorder( $T_4$ )**

**x=4**

**Inorder( $T_{\text{NULL}}$ )**

**return**

**print 4**

**Inorder( $T_{\text{NULL}}$ )**

**return**

**Inorder( $T_3$ )**

**x=3**

**Inorder( $T_{\text{NULL}}$ )**

**return**

**print 3**

**Inorder( $T_{\text{NULL}}$ )**

**return**

**print 1**

**Inorder( $T_7$ )**

**....**

לניתוח הסיבוכיות השיטה הקודמת שהראנו לניתוח סיבוכיות של קטע קוד רקורסיבית לא עובדת כי אנחנו לא יודעים מה הגודל של תת העץ השמאלי ושל תת העץ הימני.

אם גודל תת העץ השמאלי הוא  $k$  אז גודל תת העץ הימני הוא  $n-1-k$  ואז נוסחת הנסיגה יוצאת:

$$T(n) \leq C_1 + T(k) + T(n-1-k)$$

כאשר  $k$  מספר כלשהו, והפיתוח של הנוסחה הזו אינו אפשרי בשיטות שלמדנו.

לכן נשתמש בשיטה אחרת.

נסמן ב-  $C_1$  את כל הפעולות שמתבצעות בתכנית למעט הפעולות שמתבצעות בתכניות שנקראות על ידי התכנית

נשים לב שמספר הפעמים שנקרא לתוכנית Inorder הוא  $2n$

לדוגמה בעץ שבציור שלמעלה. התכנית Inorder( $T_1$ ) מבצעת שתי קריאות: אחת ל- Inorder( $T_2$ ) והשניה ל- Inorder( $T_7$ ) וכן הלאה מכל צומת יוצאות שתי קריאות. למשל מהצומת 4 יוצאות שתי קריאות ל- Inorder( $T_{NULL}$ )

ולכן בגלל שמכל צומת יוצאות 2 קריאות, ויש  $n$  צמתים אז סך כל הקריאות הוא  $2n$

בכל קריאה כזאת מבצעים לכל היותר  $C_1$  פעולות. ולכן סך הכל נקבל:

$$T(n) \leq C_1 + C_1 \cdot 2n = \theta(n)$$

ולכן הראנו ש-

$$T(n) = O(n)$$

לכוון השני:

$$T(n) \geq 2n = \theta(n)$$

ולכן הראנו ש-

$$T(n)=\Omega(n)$$

ולסיכום הראנו ש-

$$T(n)=\theta(n)$$

ברוב התכניות הרקורסיביות ניתוח הסיבוכיות דומה לניתוח הנ"ל.

התכנית הבאה מדפיסה עבור כל צומת בעץ את המפתח של הצומת ואת הגובה של הצומת:

h1(T)

```
if (root(T)==NULL) return 0
x=root(T)
y1=h1(Tleft(x))
y2=h1(Tright(x))
print key(x),max{y1,y2}+1
return max{y1,y2}+1
```

גורם איזון של צומת  $x$  מוגדר כהפרש בין הגובה של תת העץ השמאלי של  $x$  לגובה של תת העץ הימני של  $x$ .

הנוסחה לגורם איזון של צומת  $x$  היא:

$$\text{balance}(x)=h(T_{\text{left}(x)}) - h(T_{\text{right}(x)})$$

התכנית הבאה מדפיסה עבור כל צומת בעץ את המפתח של הצומת ואת גורם האיזון של הצומת:

b(T)

```
if (root(T)==NULL) return 0
x=root(T)
y1=b(Tleft(x))
y2=b(Tright(x))
print key(x),y1-y2
return max{y1,y2}+1
```

התכנית הבאה מדפיסה עבור כל צומת בעץ את המפתח של הצומת ואת מספר הצמתים שנמצאים בתת העץ ששורשו  $x$  ויש להם גורם איזון חיובי.

הפונקציה הזאת צריכה לקבל שני ערכים מהקריאה הרקורסיבית. (ערך אחד הוא הגובה וערך שני הוא מספר החיוביים). בתכניות הקודמות חזר רק ערך אחד (שהיה הגובה). כדי להחזיר שני ערכים השיטה היא שמחזירים ערך אחד למשל  $y$  ולערך הזה יש שני שדות למשל  $y.c$  ו- $y.h$ .

להלן התכנית:

P1(T)

```
if root(T)==NULL {y.c=0;y.h=0;return y}
x=root(T)
y1=P1(Tleft(x))
y2=P1(Tright(x))
y.c=y1.c+y2.c
if (y1.h > y2.h) {y.c++}
print key(x),y.c}
y.h=max{y1.h,y2.h}+1
return y
```

התכנית הבאה מדפיסה עבור כל צומת בעץ את המפתח של הצומת ואת סכום הצמתים שנמצאים במסלול מהצומת לשורש העץ.

להבדיל מהתכניות הקודמות, בתכנית הזאת צריך לקבל את האינפורמציה מלמעלה (ממי שקרא לפונקציה) ולא מלמטה (מהקריאה הרקורסיבית). השיטה היא להעביר לתכנית הרקורסיבית פרמטר נוסף שמכיל את האינפורמציה של מה שקורה למעלה. בדוגמה הנ"ל הפרמטר הנוסף הוא  $s$  שמציין את סכום המפתחות של הצמתים החל מהאבא של הצומת ועד לשורש העץ.

לתכנית הרקורסיבית צריך לקרוא פעם אחת מתכנית חיצונית שמעבירה בפרמטר הנוסף 0.

להלן התכנית החיצונית והתכנית הרקורסיבית.

P0(T)

P1(T,0)

P1(T,s)

```
if root(T)==NULL return
x=root(T)
print key(x),s+key(x)
P1(Tleft(x),s+key(x))
P1(Tright(x),s+key(x))
```

בעקרון יש 3 אפשרויות של פונקציות רקורסיביות:

(1) הפונקציה צריכה אינפורמציה רק מלמטה כדי להחליט על העץ הנוכחי.

(2) הפונקציה צריכה אינפורמציה רק מלמעלה כדי להחליט על העץ הנוכחי.

(3) התכנית צריכה אינפורמציה גם מלמעלה וגם מלמטה כדי להחליט על העץ הנוכחי.

כל התכניות פרט לאחרונה היו מסוג 1) התכנית האחרונה היתה מסוג 2)

לגבי תכנית מסוג 3) למשל התכנית הבאה תהיה מסוג 3:

כתוב תכנית שמדפיסה את כל הצמתים  $x$  בעץ שמקיימים את התנאי הבא: סכום המפתחות של הצמתים במסלול מ- $x$  אל

השורש גדול יותר מסכום המפתחות בתת העץ ששורשו x.  
להלן הקוד של התכנית הנ"ל :

P0(T)

P1(T,0)

P1(T,s)

```
if root(T)==NULL return 0
x=root(T)
y1=P1(Tleft(x),s+key(x))
y2=P1(Tright(x),s+key(x))
y=y1+y2+key(x)
if (s+key(x) > y) print key(x)
return y
```