Data Stractures $\sim$ Homework 2 $\sim$ *2025B*

Shahar Perets

April 22, 2025

הערות ומידע אישי

כל שאלה מופיעה בעמוד נפרד. ניתן אישור במודל להגיש את התרגיל שלא על גבי הקובץ המקורי בעבור מי שמקליד על המחשב.

שם: שחר פרץ

ת.ז.: 334558962

המשך בעמוד הבא

. . . . . (1) . . . . . .

1. נשנה את המימוש של המחסנית שראינו בכיתה להכפיל עצמה פי $(1+\alpha)$ בעבור $\alpha > 0$ כלשהו. נראה שזמן הריצה amortized הוא $O\left(\frac{1+\alpha}{\alpha}+1\right)$ בשיטת הפוטנציאל.

הוכחה. נתבונן בפונ' הפוטנציאל הבאה:

$$\Phi(M,n) = \begin{cases} \frac{(1+\alpha)}{\alpha}n - \frac{M}{\alpha} & n \geq \frac{M}{\alpha+1} \\ 0 & \text{else} \end{cases}$$

כאשר M גודל המערך הפנימי, ו־n גודל מבנה הנתונים (כמות האיברים בו). נסמן $\texttt{LI} =: \texttt{Insert-Last}(L,n)$.

נפריד למקרים. **אם גודל המערך לא משתנה**:

$$\begin{aligned}
\text{amort}(\texttt{IL}) &= \overbrace{\text{cost}(\text{op}(\texttt{IL}))}^{1} + \Phi(M,n+1) - \Phi(M,n) \\
&= 1 + \frac{(1+\alpha)(n+1)}{\alpha} - \frac{M}{\alpha} - \frac{(1+\alpha)n}{\alpha} + \frac{M}{\alpha} \\
&= \frac{\alpha+1+\alpha n+\alpha+n-\alpha n-n}{\alpha} + \cancel{\frac{-M+M}{\alpha}} \\
&= \frac{2\alpha+1}{\alpha} = 2 + \frac{1}{\alpha} = O\left(\frac{1}{\alpha}+1\right) = O\left(\frac{1+\alpha}{\alpha}+1\right)
\end{aligned}$$

אם גודל המערך משתנה, אזי בהכרח בפעולה הראשונה $M = n$ ובפעולה השנייה $M' = \alpha M$, ו־$n' = n+1$. הזמן של הפעולה השנייה שבצענו, שמשנה את גודל המערך, יהיה $M(\alpha+1)$ בעבור כתיבה מחדש של הכל.

$$\begin{aligned}
\text{amort}(\texttt{IL}) &= \overbrace{\text{cost}(\text{op}(\texttt{IL}))}^{M+1} + \Phi((\alpha+1)M, M+1) - \Phi(M,M) \\
&= (M+1) + \left(\frac{(1+\alpha)(M+1)}{\alpha} - \frac{(\alpha+1)M}{\alpha}\right) - \left(\frac{(1+\alpha)M}{\alpha} - \frac{M}{\alpha}\right) \\
&= M+1+\frac{1+\cancel{M-M}+\alpha+\cancel{\alpha M-\alpha M}}{\alpha} + \frac{\cancel{M-M}-M\alpha}{\alpha} \\
&= M+1+\frac{1+\alpha}{\alpha} - M = 1+\frac{1+\alpha}{\alpha} = O\left(\frac{1+\alpha}{\alpha}+1\right)
\end{aligned}$$

המקרה בו $n < \frac{M}{\alpha+1}$ לא רלוונטי, כי בתחילת המבנה הוא לא מתקיים מהצבה, ולאחר הגדלת מערך (אם נסמן ב־$\mathcal{M}$ את ה־array רגע לפני הגדלה) נבחין ש־$\frac{M}{\alpha+1} = \frac{\mathcal{M}(\alpha+1)}{\alpha+1} = \mathcal{M} < n$ (הזהות $\mathcal{M}(\alpha+1) = M$ נכונה כי הגדלנו את $\mathcal{M}$ פי $\alpha+1$ כדי לקבל את המערך מגודל M). זאת הסיבה שלאורך כל ההוכחה הצבתי תמיד במקרה הראשון של הפונקציה בלי הסבר – המקרה השני בו שוויה 0 לא ייתכן וקיים אך ורק כדי לשמור על $\Phi : \mathbb{N} \to \mathbb{R}$ ואי־שלילית.

סה"כ הראינו בשיטת הפוטנציאל שסיבוכיות הפעולות amortized במבנה להוסופה, הוא $O\left(\frac{1+\alpha}{\alpha}+1\right)$, כדרוש. ■

2. נשנה את המימוש כך שמערך בגודל k מתמלא, נקצה מערך חדש בגודל $\sqrt{k}$ תאים. נראה שזמן הריצה amortized הוא $\Theta(\sqrt{n})$.

הוכחה. לשם כך, נחסום אסימפטוטית את נוסחאת הנסיגה $T(i) = T(i-1) + \sqrt{T(i-1)}$ עם בסיס $T(1) = 1$. ננסה להבין כמה פעמים נצטרך לשנות את גודל המערך. נבחין כי הפונקציה $T(n)$ מתארת את גודל המערך אחרי n שינויי גודל, כי השינוי הגודל ה־i ידרוש ליצור מערך חדש מגודל $T(i-1) + \sqrt{T(i-1)}$. נוכיח באינדוקציה ש־$T(i) = O(n^2)$, באמצעות כך שנוכיח ש־$0.1n^2 \leq T(n) \leq n^2$ לכל $n \geq 2$.

בסיס: נציב $n = 2$ ונקבל ישירות ממקרה הבסיס של הנסיגה:

$$0.1n^2 = 0.1 \cdot 2^2 = 0.4 \leq T(n) = T(2) = T(1) + \sqrt{T(1)} = 1 + \sqrt{1} = 2 \leq 2^2 = n^2$$

צעד האינדוקציה בעבור חסם עליון:

$$\begin{aligned}
T(x) &= T(x-1) + \sqrt{T(x-1)} \\
&\leq (x-1)^2 + \sqrt{(x-1)^2} \quad \text{ה.א.} \\
&\leq x^2 - 2x + 1 + x - 1 \\
&\leq x^2 - x \leq x^2 \leq O(x^2)
\end{aligned}$$

צעד האינדוקציה בעבור חסם תחתון:

$$\begin{aligned}
T(x) &= T(x-1) + \sqrt{T(x-1)} \\
&\geq 0.1(x-1)^2 + \sqrt{0.1(x-1)^2} \qquad \text{כי } 0.3 \geq \sqrt{0.1} \\
&\geq 0.1(x^2-2x+1) + 0.3(x-1) \\
&\geq 0.1x^2 + 0.1x - 0.2 \\
&\geq 0.1x^2 \qquad \text{נכון לכל } x \geq 2
\end{aligned}$$

סה"כ ה.א. הוכחה ובהתאם להגדרת חסם אסימפטוטי, $T(x) = O(x^2) \wedge T(x) = \Omega(x^2)$ כלומר $T(x) = \Theta(x^2)$.

לכן, כדי להבין כמה פעמים עלינו להגדיל את גודל המערך כדי להגיע ל־n כלשהו, נשתמש ב־$T(i)$. נסמן את כמות הפעמים שהיה צריך להגדיל את המערך כדי להגיע לגודל n ב־k_n, או לפעמים פשוט k.

$$n = T(k) \implies n = \Theta(k_n^2) \implies \Theta(n) = k_n^2 \implies k_n = \Theta(\sqrt{n})$$

יהי רצף של n הוספות למערך, נסמן את הסיבוכיות הכוללת שלו amort. נבחין שבכל שלב מ־$i \in [k]$ ההגדלות של המערך, נצטרך "לבזבז" $\Theta(T(i))$ פעולות על יצירת המערך החדש מגודל $T(i)$ והעתקת התוכן של הישן אליו. פרט לכך, נשקיע ברצף הפעולות הזה n פעולות על כתיבת כל אחד מ־n האיברים החדשים. לא יהיו פעולות נוספות. נחסום את time הדוקות:

$$\begin{aligned}
\text{amort} &= n + \sum_{i=1}^{k} \Theta(T(i)) \qquad \Theta\left(\textstyle\sum_{i=m}^{p}\right) f(i) = \textstyle\sum_{i=m}^{p} \Theta(f(i)) \\
&= n + \Theta\left(\sum_{i=1}^{\Theta(\sqrt{n})} \overbrace{T(i)}^{\Theta(n^2)} \right) \qquad \textstyle\sum_{i=0}^{p} i^2 = \frac{p(p+1)(2p+1)}{6} \\
&= \Theta\left(n + \frac{\Theta(\sqrt{n})(\Theta(\sqrt{n})+1)(2\Theta(\sqrt{n})+1)}{6} \right) \\
&= \Theta(n + \Theta(\sqrt{n})^3) = \Theta\big(n + (\sqrt{n})^3\big) = \Theta(n + n\sqrt{n}) = \Theta(n\sqrt{n}) \quad \top
\end{aligned}$$

סה"כ, הסיבוכיות עבור רצף של n הוספות כלשהן על הרשימה תיארך $\Theta(n\sqrt{n})$, ובבמוצע לכל פעולה ידרש $\Theta(\sqrt{n})$ הוא החסם ה־amortized.

■

המשך בעמוד הבא

א. נראה שלא ניתן לממש מונה בינארי אינסופי שתומך בפעולות Increment ו־Decrement בזמן amortized קבוע לכל פעולה.

הוכחה. נניח בשלילה שקיים מונה כמתואר לעיל. יהי מספר N. נראה רצף פעולות עם זמן ריצה amortized של $\omega(N)$. נסמן לכל מספר k את $|k| = \lfloor \log k \rfloor$ מספר ספרותיו בייצוג בינארי. נחזור על Increment בדיוק $2^{|N|} - 1 - |N|$ פעמים, מה שיקח $O(|N|)$ זמן, כדי להגיע למספר $\underbrace{1111111}_{|N|+1 \text{ times}}$ הוא $|N| + (2^{|N|} - 1 - |N|) = 2^{|N|} - 1$. עתה, נחזור N פעמים על רצף של פעולות Incement ומיד לאחר כל כזה Decrement (כלומר מוסיף ונחסיר 1, N פעמים).

נחסום מלמטה את $2^{|N|} - 1 - |N|$ החיבורים הראשונים. בהנחה שכל חיבור יארך $\Omega(1)$ (מינימום), כל החיבורים האלו יחדיו יארכו $\Omega(2^{\lfloor \log N \rfloor} - \lfloor \log N \rfloor - 1) \overset{(1)}{=} \Omega(N - \log N - 1) = \Omega(N)$.

עתה נתבונן ב־N החיבור והחיסורים הנוספים. הגענו למספר ...111, נבחין מה קורה כאשר נחבר ואז נחסיר ממנו מספר:

$$1\underbrace{11111}_{|N|\text{ times}} \xrightarrow{+} 1\underbrace{00000}_{|N|\text{ times}} \xrightarrow{-} 1\underbrace{11111}_{|N|\text{ times}}$$

נבחין שבהתליך הזה, בהכרח, תחת כל פעולת חיבור או חיסור, נדרש מאיתנו להפוך $2|N|$ ביטים. לסיכום, כל זוג של חיבור ואז חיסור מעין זה – ידרוש $\Theta(2|N|)$ הפיכות, וסה"כ כל הרצף $\Theta(2|N| \cdot N)$.

נסכם: כל רצף הפעולות יארך:

$$\Omega(N) + \Theta(2|N| \cdot N) = \Omega(N) + \Theta(N \log N) = \Omega(N + N \log N) = \Omega(N \log N) = \omega(N)$$

וכמות הפעולות $\Theta(N - \log N - 1 + N) = \Theta(N)$ פעולות. סה"כ, בממוצע, כל פעולה תיארך amortized $\Omega\left(\frac{N \log N}{N}\right) = \omega(1)$, לכל מונה עם פעולות Increment ו־Decrement. בכך סיימנו להוכיח את הטענה שניתנה כרמז.

עתה נראה שלכל n טבעי קיים רצף פעולות באורך $O(n)$ כך ש־הסיבוכיות $\omega(n)$ ממבנה ריק: נתחיל מ־0 ונחבר ל־$0.5n$. משם, הראינו שקיים רצף פעולות ב־$\omega(0.5n) = \omega(n)$, באורך $O(n)$ (ניתן לחזור ולראות שאכן ביצעתי $O(n)$ פעולות לעיל). נחבר את שניהם ונקבל רצף פעולות בעלות של $O(n) + \omega(0.5N) = O(n) + \omega(n) = \omega(n)$ (במילים אחרות: המעבר מ־0 ל־$0.5n$ זניח אסימפטוטית ביחס לפעולות הכבדות שאפשר לעשות עם $O(n)$ פעולות נוספות). נחלק בכמות הפעולות ונקבל חסם amortized של $\omega(1)$ גם כאן. בכך הראינו ישירות לפי הגדרת חסם amoritzed שהסיבוכיות של כל מונה כזה היא $\omega(1)$ ולכן לא קיים מונה בסיבוכיות של $O(1)$ לכל הפעולות הנדרשות וסיימנו. ■

ב. נעבור לייצוג מספר המוגדר לפי רצף של $(t_i)_{i=0}^{k-1}$ ספרות הן $t_i \in -1, 0, +1$ כאשר המספר מוגדר לפי $\sum_{i=0}^{k-1} 2^i t_i$. נראה שכל רצף של פעולות Decrement ו־Increment יהיה בסיבוכיות amortized של $O(1)$.

הוכחה. מטעמי נוחות, נסמן $-1 =: -$, $+ =: 1$. מספר t_i כלשהו יקרא "הביט ה־i", ושינויו יקרא "הפיכה", וביט i יקרא "הפוך" אמ"מ $t_i \neq 0$. ניעזר בשיטת הבנק.

בכל הפיכת ביט מ־0 ל-+ או −, נשלם שני מטבעות – אחד בעבור הפיכת הביט ואחד נשמור מעל הביט ההפוך. נראה ששום פעולה לא תיקח יותר מטבעות ממה שנמצא בבנק (מעל הביטים ההפוכים) + 2.

פעולות ה־Decrement וה־Increment שתיהן מתוארות בצורה כזו שימשיכו להפוך את הביט הבא, אם בחיסור או חיבור בהתאמה נגיע למספר -2 או 2 בהתאמה, ובפרט תנאי הכרחי לכך הוא שהביט יהיה + או −, כלומר שכבר נהפך. על כן, כבר נמצא מטבע מעליו, ונוכל לשלמו כדי להופכו חזרה ל־0. האלגוריתם יגמר כאשר יגיע לביט ריק עם 0, שם יאלץ לשלם 2 מטבעות – 1 בעבור הפיכתו והשני בשביל להוסיף מטבע בבנק מעליו (כדי לעמוד בשמורה בפסקה לעיל).

סה"כ, בכל פעולה השקענו שני מטבעות בכל פעולה, ולכן הסיבוכיות amortized (כפונקציה של מספר הביטים הנהפכים = מספר הספרות המשתנות) היא $\Theta(2) = \Theta(1)$ כדרוש. ■

ג. עתה נממש מעל מונה רגיל פעולת Reset שתעבור על המערך ותהפוך את הביטים מ־0 ל־1 אם צריך, ומיקום הביט הכי שמאלי נשמר בשתנה עזר מחוץ למונה. נראה חסם amortized של $O(1)$.

הוכחה. נוכיח בשיטת הבנק. נחלק מטבעות באופן הבא: כל פעם שנהפוך ביט מ־0 ל־1, נשקיע 4 מטבעות – 1 בלהפוך אותו, ו־3 נשאיר "מעל" הספרה בשיוך אליה. במידת הצורך נשקיע פעולה נוספת בעדכון המיקום של הביט השמאלי ביותר. בפעולת Reset וב־Init נשקיע מטבע נוסף בביט היחיד והשמאלי ביותר במבנה. דוגמה לפעולות: (מס' המטבעות מעל כל ספרה)

$$\overset{32233}{10011} \xrightarrow{\texttt{Addition}} \overset{32322}{10100} \xrightarrow{\texttt{Reset}} \overset{2}{0}$$

כדי להראות שכל הפעולות מתבצעות ב־$O(1)$, נראה את השמורות הבאות:

- **שמורה 1.** "מעל" כל ביט במבנה שתוכנו 1 ישנם שלוש מטבעות. נובע ישירות מתיאור חלוקת המטבעות.
- **שמורה 2.** מעל כל ביט במבנה שתוכנו 0 ישנו מטבע אחד. (הערה: אם הביט נמצא משמאל לביט "הכי שמאלי", הוא לא ייחשב חלק מהמבנה). במקרה של איפוס יהיה ביט יחיד במבנה, הוא 0, ומתיאור חלוקת המטבעות מעליו יהיה מטבע אחד. נראה את נכונות שמורה 2 בעבור פעולת חיבור כאשר נתאר את צריכת המטבעות שלה בהמשך, ללא תלות בשמורה עצמה.

נראה שכל הפעולות מתבצעות ב־$O(1)$:

- **חיבור.** פעולת החיבור, באופן כמעט זהה למתואר בתרגול, תעבור על הביטים מהימני עד שתקיע לספרה 0 – בדרך היא עברה דרך אחדות בלבד, שמשמורה 1 יש עליהם 3 מטבעות, ו"לקחה" מטבע אחד בשביל המעבר עליהם. בכך סיימנו להראות את שמורה 2 – על האפסים שעברה היו 3 מטבעות מהם היא לקחה 1, וסה"כ נשארו שני מטבעות כדרוש.

  משום שעתה שמורה 2 הוכחה באופן בלתי תלוי לעצמה, נוכל להשתמש בה כדי להראות שהביט האחרון יוכל ללהיפך ל־1 – יש עליו שני מטבעות ונשתמש בהם כדי לעבור עליו ולהפוך אותו ל־1. נשקיע עוד שני מטבעות לשים מעליו כדי לשמור על שמורה 2.

  במקרה ויהיה צורך לעדכן את המונה של הביט השמאלי ביותר נשקיע בכך מטבע אחד. סה"כ השקענו לכל היותר 3 מטבעות, 2 מהם ב־0 האחרון שהפכנו ל־1, ו־1 במידת הצורך לעדכון הביט השמאלי ביותר. כלומר $3 = O(1)$ זמן.

- **איפוס.** במקרה של איפוס, נעבור על כל הביטים. אם נגיע ל־0, יש עליו משמורה 2 שני מטבעות, נשקיע אחד מהם בשביל לעבור מעליו. אם נגיע ל־1, יש מעליו 3 מטבעות משמורה 1 ונשקיע שניים מהם, אחד בלעבור עליו והשני בלהפוך אותו ל־0. לסיום נשקיע פעולה נוספת בלשמור את מיקום הביט הכי שמאלי החדש.

  השקענו זמן קבוע של 2 מטבעות/פעולות להוספה מעל הביט היחיד החדש, אחד בשביל איפוס הביט השמאלי, והמעבר על המספר והפיכת אחדותיו ל־0 התקיימה באמצעות מטבעות שכבר היו זמינים. גם כאן $3 = O(1)$.

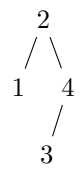
סה"כ, ללא תלות בפעולות שהיו לפניהם, חיבור ואיפוס ייקחו זמן amortized קבוע, וסה"כ כל הפעולות מתבצעות ב־$O(1)$ כדרוש. ■

המשך בעמוד הבא

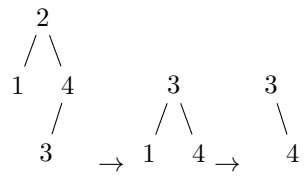
נוכיח ונפריך את הטענות הבאות:

א. **טענה.** פעולת המחיקה מעץ בינארי היא חלופית.

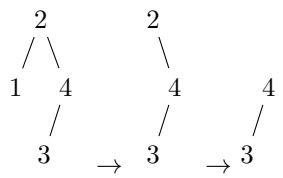
הפרכה. נתבונן בעץ הבא:



אם נסיר את 2 ואז את 1:



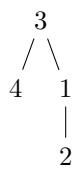
אם נסיר את 1 ואז את 2:



עצים שונים, בסתירה לטענה. ■

ב. **טענה.** לכל $u < v$ זוג צמתים בעץ בינארי, המסלול מ־u ל־v מסודר בסדר עולה.

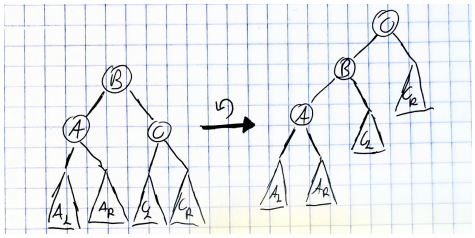
הפרכה. נתבונן בעץ הבא:



נבחר $u = 2,\ v = 4$ ואכן $u = 2 < 4 = v$. אך המסלול מ־u ל־v הוא $4, 3, 1, 2$ וזו איננה סדרה עולה, בסתירה לטענה. ■

ג. יהיו T_1, T_2 עצי חיפוש בינאריים בגודל n כך שסדרת המפתחות בשני העצים זהה. נראה שקיימת סדרה של $O(n)$ סיבובי קשתות אשר הופכת את T_1 ל־T_2.

הוכחה. נראה שכל עץ אפשר להעביר באמצעות $O(n)$ סיבובים לצורה של פס (כלומר לכל צומת בן יחיד). נבחין שהסיבוב הבא אפשרי:



ונחזור עליו (פורמלית באינדוקציה) עד שנקבל עץ מצורה של קו אחיד. משום שהגלגול הזה, במקרה הרע, יתבצע צומת עם שני עלים או (במקרה בו A ריק) על צומת עם עלה שמאלי בלבד, ויש n צמתים, אז כמות הגלגולים שידרשו לשם כך תהיה $O(n)$ במקרה הרע.

סה"כ ל־T_1 יש רצף של $O(n)$ פעולות $\alpha_1 \dots \alpha_k$ ע"מ להמיר אותו צורה של קו ישר עם הצומת הגבוהה ביותר למעלה, ובאופן דומה בעבור T_2 תהיה סדרה כזו $\beta_1 \dots \beta_n$. משום שצורה כזו יחידה, אזי $\alpha_1 \dots \alpha_k, \beta_n^{-1} \dots \beta_1^{-1}$ דרך להעביר את T_1 ל־T_2 (שכן המעבר בין α_k לבין β_k יתבצע על אותו העץ). זאת כאשר β_i^{-1} מייצג הגלגול שהופך את הגלגול β_i. סה"כ מצאנו רצף גלגולים באורך $2O(n) = O(n)$ כדרוש. ■

המשך בעמוד הבא

נגדיר עץ d־ארי להיות עץ בו לכל צומת לכל היותר d בנים, ונגדיר עץ d־ארי להיות נחמד אם לכל צומת שאינו עלה יש בדיוק d בנים.

בשאלה זו כאשר אדבר על בקיצור עץ d־ארי, הכוונה לעץ d־ארי נחמד.

1. **שאלה.** מהו מספר העלים בעץ d־ארי נחמד בעל n צמתים?

תשובה. נתייחס לעץ כאל גרף מכוון. נתבונן בעץ d־ארי נחמד כללי, ונסמן את מספר הצמתים שאינם עלים ב־k. עתה נבין כמה עלים ישנם בעץ. ניעזר בהכלה והפרדה. dk יהיה מספר הבנים שיש לכל אחד מהצמתים, וחסם עליון לכמות הצמתים בכל העץ. נבחין כי לא ספרנו שום צומת פעמים כי דרגת הכניסה של כל צומת היא 1 לכל היותר (כלומר לא יכול להיות שצומת נספרה כבת של שני צמתים שונים). נוסיף 1 כי לא ספרנו את השורש שאינו בן של אף צומת אחרת.

סה"כ, $n = dk + 1$. נחלץ את d:

$$n = dk + 1 \implies n - 1 = dk \implies k = \frac{n-1}{d}$$

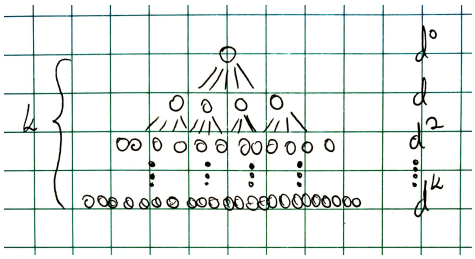
ניעזר בעקרון המשלים כדי למצוא את כמות הצמתים שהינם עלים. עולם דיון מגודל n הצמתים, ו־k שאינם עלים, סה"כ כמות העלים:

$$n - k = n - \frac{n-1}{d} = \boldsymbol{\frac{n(d-1)+1}{d}} = \mathscr{A}nswer$$

הערה: נבחין כי התשובה לא מוגדרת אמ"מ d לא מחלק את $n-1$, שכן אז ישנה כמות לא שלמה של עלים. במקרה כזה נסיק שלא קיים עץ d־ארי נחמד מתאים ל־n צמתים, מצב שייתכן. ■

2. **שאלה.** בהינתן עץ d־ארי נחמד בגובה h עם L עלים, הוכיחו שמתקיים $L \le d^h$.

תשובה. ננסה למצוא את העץ המקסימלי האפשרי בהינתן גובה h. נבחין כי אם העץ לא מאוזן, אזי ישנו צומת שבמרחק $h-i,\ i>0$ מהשורש, כלומר נוכל להוסיף לו צומת נוסף ולשמור על גובה h, ועל כן הוא אינו מקסימלי (מבחינת כמות הצמתים). לכן העץ המקסימלי מאוזן.



נבחין שבשכבה ה־i ישנם d^{i-1} צמתים (פורמלית, יש d^{i-1} במרחק i מהשורש) [הוכחה לטענה הזו בסוף ההוכחה]. לכן כמות הצמתים הכוללת היא:

$$n = \sum_{i=1}^{h} d^{i-1} = \sum_{i=0}^{h-1} d^i = \frac{d^h - 1}{d-1}$$

ולכן, מסעיף (א), כמות העלים בעץ כזה:

$$L = \frac{\frac{d^h-1}{d-1}(d-1)+1}{d} = \frac{d^h \cancel{-1+1}}{d} = d^{h-1}$$

שכאמור מהווה חסם עליון לכמות הצמתים בכל עץ d־ארי נחמד, וסה"כ החסם העליון לכמות הצמתים הוא $L \le d^{h-1} \le d^h$ כדרוש.

[הוכחה לכך שכמות הצמתים במרחק i מהשורש היא d^{i-1}: בסיס עץ עם צומת אחד, שכבה אחת ו־$d^{1-1} = d^0 = 1$ צמתים כדרוש, צעד נניח באינדוקציה על i ונוכיח על $i+1$, אזי לכל אחד מ־i הצמתים יש d בנים הם היחידים ממרחק $i+1$ (קשת נוספת בינהם לבין אביהם במרחק i), ומה.א. יש d^{i-1} צמתים במרחק i ולכן סה"כ ישנם $d \cdot d^{i-1} = d^{(i+1)-1}$ צמתים במרחק $i+1$ כדרוש] ■

המשך בעמוד הבא

סעיף א'

להלן פסאדו קוד (רקורסיבי) שבודק האם עץ בינארי הוא עץ חיפוש:

```
input  : Binary tree T
output: Whether is T a search tree

function RecSearch(node) → boolean, max, min is
    if node = null then
        return true, −∞, +∞
    end
    left ← node.left
    right ← node.right
    leftMax ← −∞
    rightMin ← +∞
    isSearch ← true
    if right ≠ null then
        recCall ← RecSearch(right)
        leftMax ← recCall[1]
        isSearch ← recCall[0] and isSearch and \\
            node.item > leftMax
    else if left ≠ null then
        recCall ← RecSearch(left)
        rightMin ← recCall[2]
        isSearch ← recCall[0] and isSearch and \\
            node.item < rightMin
    end
    return isSearch, max(node.item, leftMax), \\
        min(node.item, rightMin)
end
return RecSearch(T.root, −∞, +∞)
```

(הערה: \\ מציין מעבר שורה מפאת מקום מבלי לקטוע את ההצרה)

נוכיח שהסיבוכיות היא $O(n)$. נניח בשלילה שתקרא קריאה רקורסיבית פעמיים לאותו האיבר, אז משום שקריאות רקורסיבית מתבצעות אך ורק לצמתים המחוברים בצומת אחת לצומת האב, אזי אפשר להרכיב מעגל (לא מכוון) מן הצומת לעצמה, למרות שבעץ אין מעגלים כאלו וסתירה. יש n אפשרויות שונות לקלטי הפונקציה, הם צמתי העץ, הראינו כי לא נחזור על אותו הצומת פעמיים, וסה"כ יהיו לכל היותר n קריאות רקורסיביות. משום שבפנים כל הפעולות פעולות השוואה ושמירת מידע בזכרון (עד לכדי פעולות רקורסיביות שכבר ניתחנו), סה"כ הסיבוכיות היא $O(n)$.

סעיף ב'

הגיון: נשמור ב־mode את המצב הנוכחי (קורא שמאלה או קורא ימינה) וב־heads את הצמתים שעליהם עברנו ומצטרך לחזור אילהם. המצבים הם 0 כאשר זז שמאלה, 1 כאשר ימינה, ו־2 כאשר סיימנו לזוז ימינה ויש צורך לעבור לצומת האב הבא.

```
input  : T binary tree
output: nothing

heads ← Stack()    // Top, Pop & Push in O(1)
mode ← 0
Push(heads, T.root)
while Top(heads) ≠ 0 do
    if mode = 0 then                    // mode = 0
        left ← Top(heads).left
        if left ≠ null then
            Push(heads, left)
        else
            Print(Top(heads))
            mode ← 1
        end
    else if mode = 1 then               // mode = 1
        right ← Top(heads).right
        if right ≠ null then
            Push(heads, right)
            mode ← 0
        else
            Pop(heads)
            mode ← 2
        end
    else if mode = 2 then               // mode = 2
        Pop(heads)
        mode ← 1
    end
end
```

הסיבוכיות של האלגוריתם לינארית כי הוא מבצע ישירות מעבר על הצמתים in-order, בלי לעבור על שום צומת פעמיים.