# מבחן מספר 1: סמסטר 2024ב׳, 1 ביולי 2024, שאלון 450, מועד 82

**עליכם לענות על ארבע מבין חמש השאלות. משקל כל שאלה 25 נקודות.**

**שאלה 1**

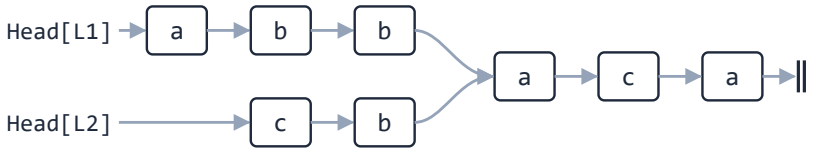
א. פתרו את נוסחת הנסיגה הבאה: $T(n) = 2T(n-1) + 1$

ב. תהיינה $f, g : \mathbb{N} \to \mathbb{R}^+$. נגדיר ש-$f(n) = O_l(g(n))$ אם קיים $\varepsilon > 0$ שעבורו $f(n) = O\left(g(n)\log^{\varepsilon}(n)\right)$.
הוכיחו או הפריכו: אם $f(n) = O_l(g(n))$ וגם $g(n) = O_l(h(n))$ אז $f(n) = O_l(h(n))$. יש להוכיח/להפריך על פי ההגדרה.

**שאלה 2**

א. כתבו אלגוריתם המקבל זוג מערכים בגדלים $m, n$ ומחזיר מערך המכיל את כל האיברים המשותפים לשני המערכים. ניתן להניח כי אף אחד מן המערכים אינו מכיל את אותו מפתח יותר מפעם אחת (כלומר בכל מערך בפני עצמו אין כפולים אך ייתכנו ערכים המשותפים לשני המערכים). זמן הריצה המבוקש הוא:
$\Theta\left(\max(m,n)\cdot\log(\min(m,n))\right)$.

ב. נגדיר מבנה של **רשימה דו-ראשית** כמורכב משתי רשימות (חד-מקושרות) המתחברות לרשימה אחת. שני מקטעי הראש אינם בהכרח שווים בגודלם (ראו דוגמה באיור).




כתבו אלגוריתם המקבל כקלט מצביעים לרשימה דו-ראשית (כגון $head[L1]$, $head[L2]$ באיור), ובונה רשימה חד-מקושרת רגילה, המכילה איברים עם אותם מפתחות כמו ברשימה הדו-ראשית ובאותו מספר מופעים. (בדוגמת האיור הרשימה שתיווצר צריכה לכלול שלושה איברים עם מפתח a, שלושה עם b, ושניים עם c). אין לשנות את הרשימה הדו-ראשית (או להוסיף לה שדות). אין חשיבות לסדר האיברים ברשימה הרגילה שהאלגוריתם בונה. זמן הריצה המבוקש הוא לינארי באורך הרשימה המתקבלת.

**שאלה 3**

נתונה טבלה (מטריצה) בגודל $m \times n$ שאיברי כל שורותיה ממוינים משמאל לימין ואיברי כל עמודותיה ממוינים מלמעלה למטה. לדוגמה:

| | | | | |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 |
| 15 | 25 | 35 | 45 | 52 |
| 27 | 29 | 37 | 48 | 55 |
| 32 | 33 | 39 | 58 | 60 |

א. כתבו שגרה המקבלת כקלט מספר $K$ ומחזירה את מיקומו בטבלה (או 0 אם הוא לא קיים). זמן הריצה הנדרש: $\Theta(m+n)$.

ב. כתבו שגרה המקבלת כקלט מספר $K$, מכניסה אותו במקום האיבר המקסימלי בטבלה, ומתקנת את הטבלה כך ששורותיה ועמודותיה יקיימו את התכונה שלעיל. זמן הריצה הנדרש: $\Theta(m+n)$.

הדרכה: חשבו היכן נמצא המקסימום. נסו להחליפו ולתקן ע"י הזזת איברים מתאימים.

תזכורת: בשני הסעיפים עליכם להסביר את נכונות האלגו' (לאו דווקא באופן פורמלי לחלוטין).

**שאלה 4**

נתון מערך של $m$ מספרים **שלמים ושונים זה מזה** בתחום $[0 \dots n^5/\log n]$, $m \leq n$.

בנוסף נתון עץ חיפוש בינארי $T$ ובו $n$ צמתים (לא עץ אדום שחור).

כתבו אלגוריתם הבודק בזמן $\Theta(n)$ האם **כל** האיברים במערך מופיעים בעץ.

**שאלה 5**

תכננו מבנה נתונים $S$ שבאמצעותו ניתן לבצע את הפעולות הבאות בסיבוכיות זמן ריצה של $\Theta(\log n)$, כאשר $n$ מציין את מספר האיברים במבנה. הניחו כי כל האיברים במבנה **שונים** זה מזה.

- $insert(S,k)$ – הכנסת איבר בעל המפתח $k$ ל-$S$.
- $find(S,k)$ – החזרת איבר בעל המפתח $k$ ב-$S$, אם קיים. אחרת, מחזירה $NIL$.
- $inversions(S,k)$ – אם קיים במבנה איבר בעל המפתח $k$, השגרה תחזיר את מספר ההיפוכים עבור $k$.

הגדרת היפוך: כל איבר בעל מפתח $l$ שהוכנס לפני $k$ (כלומר לפני האיבר שערך המפתח שלו הוא $k$) והמקיים $l > k$.

במילים אחרות, היפוך הוא זוג-איברים שנמצאים במבנה, אשר הוכנסו אליו בסדר הפוך ליחס-הערכים של מפתחותיהם; כלומר – האיבר בעל המפתח הגדול הוכנס לפני האיבר בעל המפתח הקטן.

# תשובות למבחן מספר 1: סמסטר 2024ב׳, 1 ביולי 2024, שאלון 450, מועד 82

**תשובה לשאלה 1:**

סעיף א

**האלגוריתם: פתיחת הנוסחה (איטרציה) ואז הוכחה באינדוקציה (שיטת ההצבה, הספר עמ׳ 53–57)**

1. מציבים את הנוסחה בתוך עצמה כמה פעמים, עד שרואים את התבנית אחרי $k$ צעדים.
2. בוחרים $k$ שמגיע לתנאי ההתחלה $T(1)$, ומחשבים את הסכום שנוצר.
3. מוכיחים את הנוסחה שקיבלנו באינדוקציה (שיטת ההצבה), ומסיקים את הסדר $\Theta$.

> **שימו לב:** שיטת האב (הספר עמ׳ 63) **לא** מתאימה כאן – היא מטפלת רק בנוסחאות מהצורה $T(n) = aT(n/b) + f(n)$, ופה מקטינים ב-1 ($n-1$) ולא מחלקים ב-$b$.

**שלב ראשון – פותחים את הנוסחה.** נסמן $T(1) = c$ (קבוע). מציבים פעם אחר פעם:

$$T(n) = 2T(n-1) + 1 = 2\big(2T(n-2)+1\big) + 1 = 4T(n-2) + 2 + 1 = 8T(n-3) + 4 + 2 + 1$$

ואחרי $k$ הצבות מקבלים את התבנית:

$$T(n) = 2^k\, T(n-k) + \big(2^{k-1} + \cdots + 2 + 1\big) = 2^k\, T(n-k) + 2^k - 1$$

(סכום של טור הנדסי: $1 + 2 + \cdots + 2^{k-1} = 2^k - 1$.)

**שלב שני – מגיעים לתנאי ההתחלה.** בוחרים $k = n-1$, ואז $T(n-k) = T(1) = c$:

$$T(n) = 2^{n-1}c + 2^{n-1} - 1 = 2^{n-1}(c+1) - 1$$

**שלב שלישי – הוכחה באינדוקציה (שיטת ההצבה).** נוכיח ש-$T(n) = 2^{n-1}(c+1) - 1$ לכל $n \geq 1$:

- **בסיס:** $n = 1$: $2^0(c+1) - 1 = c = T(1)$.
- **צעד:** נניח שהנוסחה נכונה עבור $n-1$. אז $T(n) = 2T(n-1) + 1 = 2\big(2^{n-2}(c+1) - 1\big) + 1 = 2^{n-1}(c+1) - 2 + 1 = 2^{n-1}(c+1) - 1$.

**שלב רביעי – הסדר.** $c+1$ הוא קבוע חיובי, ולכן $T(n) = \frac{c+1}{2} \cdot 2^n - 1 = \Theta(2^n)$: לכל $n \geq 2$ מתקיים $\frac{c+1}{4} \cdot 2^n \leq T(n) \leq \frac{c+1}{2} \cdot 2^n$ (הגדרת $\Theta$, הספר עמ׳ 38).

**נקבל אפוא את התשובה הסופית:**

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

סעיף ב

**תבנית: הוכחה לפי ההגדרה (הגדרת $O$, הספר עמ' 38–39)**

1. כותבים כל נתון לפי ההגדרה, **עם קבועים משלו** ($c_1, n_1, \varepsilon_1$ ו-$c_2, n_2, \varepsilon_2$).
2. משלבים את שני אי-השוויונות לאי-שוויון אחד.
3. מציגים קבועים מפורשים ($c, n_0, \varepsilon$) שמקיימים את ההגדרה של המסקנה.

**הטענה נכונה.** נוכיח לפי ההגדרה.

**שלב ראשון – הנתונים לפי ההגדרה.**

- $f(n) = O_l(g(n))$: קיים $\varepsilon_1 > 0$ ש-$f(n) = O(g(n)\log^{\varepsilon_1} n)$ כלומר קיימים $c_1 > 0$ ו-$n_1$ כך שלכל $n \geq n_1$: $0 \leq f(n) \leq c_1\, g(n)\log^{\varepsilon_1} n$.
- $g(n) = O_l(h(n))$: קיים $\varepsilon_2 > 0$, וקיימים $c_2 > 0$ ו-$n_2$ כך שלכל $n \geq n_2$: $0 \leq g(n) \leq c_2\, h(n)\log^{\varepsilon_2} n$.

**שלב שני – משלבים.** נבחר $n_0 = \max(n_1, n_2, 2)$. לכל $n \geq n_0$ שני אי-השוויונות מתקיימים, ו-$\log n \geq 1 > 0$, ולכן מותר להכפיל את השני ב-$c_1 \log^{\varepsilon_1} n > 0$:

$$0 \leq f(n) \leq c_1\, g(n)\log^{\varepsilon_1} n \leq c_1 \cdot c_2\, h(n)\log^{\varepsilon_2} n \cdot \log^{\varepsilon_1} n = c_1c_2\ h(n)\log^{\varepsilon_1+\varepsilon_2} n$$

**שלב שלישי – הקבועים.** עם $c = c_1c_2 > 0$, $n_0 = \max(n_1, n_2, 2)$ ו-$\varepsilon = \varepsilon_1 + \varepsilon_2 > 0$ מתקיים $f(n) = O(h(n)\log^{\varepsilon} n)$, וזה בדיוק $f(n) = O_l(h(n))$.

**נקבל אפוא את התשובה הסופית:**

הטענה נכונה: $f(n) = O_l(h(n))$ עם $\varepsilon = \varepsilon_1 + \varepsilon_2$ ו-$c = c_1c_2$.

## תשובה לשאלה 2:

סעיף א

| INTERSECTION(A, B) | |
|---|---|
| **הרעיון:** נמיין את המערך **הקצר**, ונחפש בו בחיפוש בינרי כל איבר של המערך הארוך. | |
| **מה נתון:** בתוך כל מערך אין מפתח כפול (ערכים משותפים לשני המערכים מותרים). | |
| **הכלים:**<br>• $A$ – המערך הקצר, $k = \min(m, n)$ איברים<br>• $B$ – הארוך, $M = \max(m, n)$ איברים<br>• $C$ – מערך התוצאה, ו-$c$ – מספר האיברים בו. | |
| **האלגוריתם:** | **זמן** |
| 1. אם $A$ ארוך מ-$B$ – מחליפים ביניהם, כך ש-$A$ הוא הקצר (שורות 1–2). | $O(1)$ |
| 2. ממיינים את $A$ במיון ערמה (שורה 3 ; הספר עמ' 113). | $\Theta(k \lg k)$ |

| | |
|---|---|
| 3. עוברים על איברי $B$ אחד-אחד, ומחפשים כל אחד ב-$A$ בחיפוש בינרי (תרגיל 2.3-5, הספר עמ' 31); אם הוא נמצא – מוסיפים אותו לסוף $C$ ב-$O(1)$ (שורות 4–8). | $M \cdot \Theta(\lg k)$ |
| סה"כ סיבוכיות הזמן: $O(1) + \Theta(k \lg k) + \Theta(M \lg k) = \Theta(M \lg k) = \Theta\left(\max(m,n) \cdot \log(\min(m,n))\right)$, כי $k \le M$ | |

בפסאודו-קוד:

```
INTERSECTION(A, B)
1  if length[A] > length[B]
2     then exchange A <-> B            ▷ now A is the shorter: k = length[A]
3  HEAPSORT(A)                         ▷ Θ(k lg k)
4  c <- 0
5  for i <- 1 to length[B]             ▷ M times
6      do if BINARY-SEARCH(A, B[i]) ≠ NIL      ▷ Θ(lg k)
7            then c <- c + 1
8                 C[c] <- B[i]
9  return C[1..c]
```

**נכונות:** איבר נמצא בחיתוך אם ורק אם הוא מופיע בשני המערכים. עוברים על **כל** איבר של $B$ ובודקים אם הוא ב-$A$, ולכן כל איבר משותף נכנס ל-$C$ ורק הם. כי אין כפולים בתוך כל מערך, כל איבר משותף נכנס פעם אחת בלבד.

> **שימו לב:** חשוב למיין דווקא את הקצר. מיון של הארוך עולה $\Theta(M \lg M)$, ואם $M$ גדול הרבה יותר מ-$k$ זה יותר מהמבוקש.

סעיף ב

| COPY-TWO-HEADED(h1, h2) | |
|---|---|
| **הרעיון:** נמצא את הצומת שבו שתי הרשימות מתחברות, ואז נעתיק לרשימה חדשה את כל $L_1$ ורק את החלק של $L_2$ שלפני צומת החיבור. | |
| **מה נתון:** אסור לשנות את הרשימה הדו-ראשית או להוסיף לה שדות; ייתכן שאין חלק משותף (אז צומת החיבור הוא $NIL$). $t$ – אורך החלק המשותף, ו-$N = l_1 + l_2 - t$ – אורך הרשימה שנבנה. | |
| **הכלים:**<br>• $l_1, l_2$ – אורכי שתי הרשימות<br>• $x, y$ – מצביעים שרצים על $L_1$ ועל $L_2$<br>• $join$ – צומת החיבור<br>• $L$ – הרשימה החדשה. | |
| **האלגוריתם:** | **זמן** |
| 1. מודדים את אורך כל רשימה במעבר אחד עליה – השגרה LENGTH (שורה 1). | $\Theta(l_1 + l_2)$ |
| 2. מקדמים את המצביע של הרשימה הארוכה בהפרש האורכים – עכשיו $x$ ו-$y$ רחוקים אותו מספר צמתים מהסוף (שורות 2–4). | $O(\lvert l_1 - l_2 \rvert)$ |
| 3. מקדמים את $x$ ואת $y$ יחד, צעד אחר צעד, עד שהם מצביעים על אותו צומת – זה צומת החיבור (שורות 5–6). | $O(\min(l_1, l_2))$ |

| | |
|---|---|
| 4. עוברים על $L_1$ מתחילתה ועד סופה, ולכל צומת מוסיפים לראש $L$ צומת חדש עם אותו מפתח – LIST-INSERT, $O(1)$ לצומת (שורות 7–9; הספר עמ' 172). | $\Theta(l_1)$ |
| 5. עוברים על $L_2$ מתחילתה עד צומת החיבור (לא כולל), ומוסיפים כך גם את צמתיה (שורות 10–11). | $\Theta(l_2 - t)$ |
| סה"כ סיבוכיות הזמן: $\Theta(l_1 + l_2) = \Theta(N)$, כי $N \le l_1 + l_2 = N + t \le 2N$ ($t \le l_1 \le N$) | |

בפסאודו-קוד:

```
LENGTH(x)
1  l <- 0
2  while x ≠ NIL
3       do l <- l + 1
4          x <- next[x]
5  return l

COPY-TWO-HEADED(h1, h2)
1  l1 <- LENGTH(h1);  l2 <- LENGTH(h2)
2  x <- h1;  y <- h2
3  while l1 > l2  do  x <- next[x];  l1 <- l1 - 1
4  while l2 > l1  do  y <- next[y];  l2 <- l2 - 1
5  while x ≠ y    do  x <- next[x];  y <- next[y]     ▷ x = the join (or NIL)
6  join <- x
7  L <- an empty list
8  for p <- h1, while p ≠ NIL, p <- next[p]
9       do LIST-INSERT(L, a new node with key[p])     ▷ all of L1
10 for p <- h2, while p ≠ join, p <- next[p]
11      do LIST-INSERT(L, a new node with key[p])     ▷ only the head part of L2
12 return L
```

**נכונות:** החלק המשותף הוא סוף של שתי הרשימות, ולכן אחרי שורות 3–4 שני המצביעים רחוקים באותו מספר צמתים מהסוף, ושורה 5 מביאה אותם **ביחד** לצומת המשותף הראשון. שורות 8–11 מעתיקות כל צומת של המבנה בדיוק פעם אחת: את החלק המשותף רק דרך $L_1$, ואת ראש $L_2$ בנפרד. בדוגמה: $a, b, b, a, c, a$ מ-$L_1$ ועוד $c, b$ – שלושה $a$, שלושה $b$ ושני $c$.

**נקבל אפוא את התשובה הסופית:**

מוצאים את צומת החיבור בעזרת האורכים ($\Theta(l_1 + l_2)$), ומעתיקים את $L_1$ כולה ואת ראש $L_2$ – זמן $\Theta(N)$.

**תשובה לשאלה 3:**

זו **טבלת יאנג** (בעיה 3-6, הספר עמ' 119–120): כל שורה ממוינת משמאל לימין וכל עמודה מלמעלה למטה.

סעיף א

| YOUNG-SEARCH(Y, m, n, K) | |
|---|---|
| **הרעיון:** נחפש מהפינה הימנית-העליונה; בכל השוואה נמחקת שורה שלמה או עמודה שלמה שבוודאות אין בהן $K$. | |
| **מה נתון:** כל שורה ממוינת משמאל לימין וכל עמודה מלמעלה למטה. | |
| **הכלים:**<br>• $i$ – השורה הנוכחית, $j$ – העמודה הנוכחית. | |
| **האלגוריתם:** | **זמן** |
| 1. מתחילים ב-$Y[1,n]$ – הגדול בשורה שלו והקטן בעמודה שלו (שורה 1). | $O(1)$ |
| 2. כל עוד לא יצאנו מהטבלה משווים את $Y[i,j]$ ל-$K$: אם שווים – מחזירים $(i,j)$; אם $Y[i,j] > K$ – כל העמודה מכאן ולמטה גדולה מ-$K$, ולכן זזים **שמאלה**; אם $Y[i,j] < K$ – כל השורה מכאן ושמאלה קטנה מ-$K$, ולכן יורדים **למטה**. כל סבב מגדיל את $i$ או מקטין את $j$ – לכל היותר $m+n-1$ סבבים (שורות 2–7). | $(m+n-1)\cdot O(1)$ |
| 3. יצאנו מהטבלה – $K$ לא בה, מחזירים 0 (שורה 8). | $O(1)$ |
| סה"כ סיבוכיות הזמן: $O(1)+(m+n-1)\cdot O(1)=\Theta(m+n)$ במקרה הגרוע | |

בפסאודו-קוד:

```
YOUNG-SEARCH(Y, m, n, K)
1  i <- 1;  j <- n
2  while i ≤ m and j ≥ 1
3       do if Y[i, j] = K
4             then return (i, j)
5          if Y[i, j] > K
6             then j <- j - 1        ▷ column j (from row i down) is all > K
7             else i <- i + 1        ▷ row i (from column j left) is all < K
8  return 0
```

**דוגמה:** חיפוש $K=37$ – $50>37$ שמאלה, $40>37$ שמאלה, $30<37$ למטה, $35<37$ למטה, $37$ – נמצא במקום $(3,3)$:

| | | | | |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 |
| 15 | 25 | 35 | 45 | 52 |
| 27 | 29 | 37 | 48 | 55 |
| 32 | 33 | 39 | 58 | 60 |

וחיפוש $K = 44$: 50 שמאלה, 40 למטה, 45 שמאלה, 35 למטה, 37 למטה, 39 למטה – יצאנו מהטבלה, מחזירים 0:

| | | | | |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 ← | 50 |
| 15 | 25 | 35 ← | 45 ↓ | 52 |
| 27 | 29 | 37 ↓ | 48 | 55 |
| 32 | 33 | 39 ↓ | 58 | 60 |

**נכונות:** בכל צעד נמחקת שורה או עמודה שבוודאות אין בה $K$ (לפי סדר המיון), ולכן אם $K$ בטבלה – לא "דילגנו" עליו, ונגיע אליו.

סעיף ב

| YOUNG-REPLACE-MAX(Y, m, n, K) | |
|---|---|
| **הרעיון:** נשים את $K$ במקום המקסימום, בפינה הימנית-התחתונה, ונעלה אותו למעלה או שמאלה עד שהסדר מתוקן. | |
| **מה נתון:** הטבלה ממוינת בשורות ובעמודות. המקסימום הוא $Y[m,n]$: הוא הגדול בשורה $m$, וכל איבר קטן או שווה לאיבר שמתחתיו ולאיבר שמימינו – עד $Y[m,n]$. | |
| **הכלים:**<br>• $i,j$ – המקום הנוכחי של $K$<br>• $up$ – השכן שמעליו, $left$ – השכן שמשמאלו ($-\infty$ אם אין שכן כזה). | |
| **האלגוריתם:** | **זמן** |
| 1. כותבים $K$ ב-$Y[m,n]$ (שורות 1–2). | $O(1)$ |
| 2. בכל סבב משווים את $K$ לשכן שמעליו ולשכן שמשמאלו: אם $K$ גדול או שווה לשניהם – עוצרים; אחרת מחליפים את $K$ עם **הגדול** מבין השניים, ו-$i$ או $j$ קטן ב-1. לכל היותר $(m-1)+(n-1)$ החלפות (שורות 3–10). | $(m+n-2)\cdot O(1)$ |
| סה"כ סיבוכיות הזמן: $O(1)+(m+n-2)\cdot O(1)=\Theta(m+n)$ במקרה הגרוע | |

בפסאודו-קוד:

```
YOUNG-REPLACE-MAX(Y, m, n, K)
1  i <- m;  j <- n
2  Y[i, j] <- K
3  loop
4     up   <- Y[i-1, j]  if i > 1, else -∞
5     left <- Y[i, j-1]  if j > 1, else -∞
6     if K ≥ up and K ≥ left
7        then return                    ▷ K is in place
8     if up ≥ left
9        then exchange Y[i, j] <-> Y[i-1, j];  i <- i - 1
10       else exchange Y[i, j] <-> Y[i, j-1];  j <- j - 1
```

**למה עם הגדול מבין השניים?** נניח ש-$up \geq left$ ומחליפים עם $up$: הוא יורד למקום של $K$, ושם הוא גדול גם מ-$left$ (שמשמאלו עכשיו) וגם ממה שמעליו לפני כן – כלומר הסדר סביבו נשמר. כל שאר הטבלה לא השתנתה,

והבעיה היחידה שאולי נשארה היא בין $K$ לשכניו החדשים מעליו ומשמאלו – וזה בדיוק מה שהלולאה בודקת שוב.

**דוגמה:** $K = 26$ במקום 60. השכנים 55 (למעלה) ו-58 (משמאל): הגדול 58, שמאלה. אחר כך 48 (למעלה) מול 39: למעלה. אחר כך 45 מול 37: למעלה. אחר כך 40 מול 35: למעלה. בשורה 1 אין שכן מעל, ו-$30 > 26$: שמאלה. עכשיו $20 < 26$ – עוצרים ב-$(1, 3)$:

| | | | | |
|---|---|---|---|---|
| 10 | 20 | 26 ← | 30 | 50 |
| 15 | 25 | 35 | 40 ↑ | 52 |
| 27 | 29 | 37 | 45 ↑ | 55 |
| 32 | 33 | 39 | 48 | 58 ← |

**נקבל אפוא את התשובה הסופית:**

א. חיפוש מהפינה הימנית-העליונה, שורה או עמודה בכל צעד – $\Theta(m + n)$. ב. $K$ במקום $Y[m, n]$ והחלפה עם השכן הגדול מעליו/משמאלו – $\Theta(m + n)$.

**תשובה לשאלה 4:**

| ALL-IN-TREE(A, m, T, n) | |
|---|---|
| **הרעיון:** נהפוך את העץ ואת המערך לשתי סדרות ממוינות – בלי מיון השוואות – ונעבור עליהן במקביל. | |
| **מה נתון:** במערך $m \le n$ מספרים **שלמים ושונים** בתחום $[0, n^5/\log n]$; $T$ עץ חיפוש בינרי עם $n$ צמתים, לא בהכרח מאוזן. | |
| **הכלים:**<br>• $S[1..n]$ – מפתחות העץ לפי הסדר<br>• $i$ – המקום ב-$A$, $j$ – המקום ב-$S$. | |
| **האלגוריתם:** | **זמן** |
| 1. סריקה תוכית של $T$; כל מפתח שמודפס נכתב לסוף $S$, ולכן $S$ ממוין – לכל עץ חיפוש, גם לא מאוזן (שורה 1; משפט 12.1, הספר עמ' 214–215). | $\Theta(n)$ |
| 2. ממיינים את $A$ במיון בסיס בבסיס $n$: המספרים קטנים מ-$n^6$, ולכן יש להם לכל היותר 6 ספרות, וכל ספרה – מיון מנייה עם $k = n$ (שורה 2; למה 8.3, הספר עמ' 143). | $\Theta(6(m + n))$ |
| 3. עוברים על $A$ לפי הסדר; לכל $A[i]$ מקדמים את $j$ כל עוד $S[j] < A[i]$. אם $j$ עבר את סוף $S$ או $A[i] - S[j] \ne A[i]$ לא בעץ, ומחזירים "לא". $j$ רק מתקדם (שורות 3–8). | $\Theta(m + n)$ |
| 4. עברנו על כל $A$ – מחזירים "כן" (שורה 9). | $O(1)$ |
| סה"כ סיבוכיות הזמן: $\Theta(n) + \Theta(6(m + n)) + \Theta(m + n) = \Theta(n)$, כי $m \le n$ | |

בפסאודו-קוד:

```
ALL-IN-TREE(A, m, T, n)
1  S <- INORDER-TREE-WALK(root[T])        ▷ S[1..n] sorted, Θ(n)
2  RADIX-SORT(A, d = 6) in base n          ▷ COUNTING-SORT per digit, Θ(6(m + n))
3  j <- 1
4  for i <- 1 to m
```

```
5      do while j ≤ n and S[j] < A[i]
6            do j <- j + 1
7         if j > n or S[j] ≠ A[i]
8            then return FALSE              ▷ A[i] is not in T
9  return TRUE
```

**נכונות:** שני המערכים ממוינים, ו-$j$ רק מתקדם. כשמגיעים ל-$A[i]$, כל מפתחות $T$ שלפני $S[j]$ קטנים מ-$A[i]$, ולכן אם $A[i]$ בעץ – הוא בדיוק $S[j]$. ה-$A[i]$ הבא גדול יותר (המספרים שונים), כך שאין צורך לחזור אחורה.

**שימו לב:** מיון השוואות של המערך (למשל מיון ערמה) עולה $\Theta(m \lg m)$, וכש-$m$ קרוב ל-$n$ זה יותר מ-$\Theta(n)$ (החסם התחתון למיון השוואות, הספר עמ' 137). לכן משתמשים בכך שהמספרים **שלמים ובתחום חסום** – מיון בסיס. גם חיפוש של כל איבר בעץ לא עובד: העץ אינו מאוזן, וגובהו עלול להיות $n$.

**נקבל אפוא את התשובה הסופית:**

סריקה תוכית של $T$ ($\Theta(n)$), מיון בסיס של המערך בבסיס $n$ ($6 \le$ ספרות, $\Theta(n)$), ומעבר מיזוג אחד ($\Theta(n)$) – סך הכול $\Theta(n)$.

**תשובה לשאלה 5:**

**תבנית: הרחבת מבנה נתונים (סעיף 14.2, הספר עמ' 258–259)**

1. **בוחרים מבנה בסיס:** עץ אדום-שחור – $insert$ ו-$find$ ב-$\Theta(\lg n)$.
2. **בוחרים מידע נוסף:** בכל צומת $x$ – $size[x]$ (גודל תת-העץ, כמו בעץ ערכי מיקום) ו-$inv[x]$ (מספר ההיפוכים של המפתח שלו).
3. **מוודאים שאפשר לתחזק אותו:** $size$ מתעדכן לאורך מסלול ההכנסה ובכל סיבוב ב-$O(1)$ (הספר עמ' 257); $inv[x]$ נקבע פעם אחת, ברגע ההכנסה, ולא משתנה אחר כך.
4. **כותבים את הפעולות החדשות.**

**הרעיון המרכזי:** ההיפוכים של $k$ הם המפתחות שהוכנסו **לפני** $k$ והם **גדולים** ממנו. מפתחות שיוכנסו אחריו לא משפיעים עליו. לכן מספיק לספור, **ברגע ההכנסה** של $k$, כמה מהמפתחות שכבר במבנה גדולים ממנו:

$$inv[k] = \underbrace{size[root]}_{\text{כמה כבר במבנה}} - \underbrace{r}_{\text{כמה מהם קטנים מ-}k}$$

את $r$ מחשבים תוך כדי הירידה במסלול ההכנסה, כמו ב-OS-RANK (הספר עמ' 255): כל פעם שיורדים **ימינה** מצומת $y$, כל המפתחות בתת-העץ השמאלי של $y$ ו-$y$ עצמו קטנים מ-$k$.

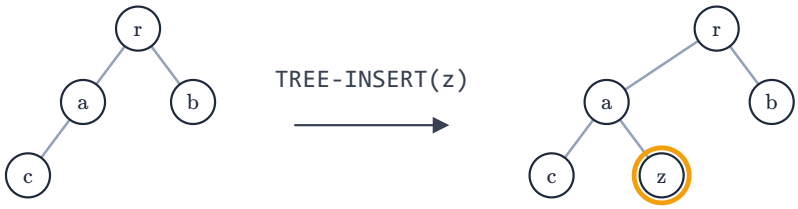
**המבנה:**

| המבנה $S$ – עץ אדום-שחור, ובכל צומת $x$: | |
|---|---|
| `root[S]` | שורש העץ; $size[root[S]]$ = מספר האיברים $n$. |
| `key[x]` | המפתח. |
| `color[x]` | אדום / שחור – שומר על גובה $\leq 2\lg(n+1)$ (למה 13.1). |
| `left[x], right[x], p[x]` | הבן השמאלי, הבן הימני והאב. |
| `size[x]` | מספר הצמתים בתת-העץ של $x$ – כמו בעץ ערכי מיקום (הספר עמ' 253). |
| `inv[x]` | מספר ההיפוכים של $key[x]$ – נקבע בהכנסה ולא משתנה. |

**שמירת מבנה הנתונים של העץ האדום-שחור – נמחיש את התיקונים בכל שינוי במבנה, כדלקמן:**

לפני כל פעולה השדות $size$ ו-$inv$ נכונים בכל הצמתים. כמו בתחזוקת $size$ בספר (סעיף 14.1, עמ' 256–257), הכנסה לעץ אדום-שחור עוברת שני שלבים, ובכל שלב מתקנים רק את הצמתים שהשתנו:

**שלב 1 – הכנסת הצומת** (כמו TREE-INSERT): הצומת החדש $z$ נתלה כעלה, ומשתנה רק המסלול מ-$z$ אל השורש.




תיקון לאחר ההכנסה:

1. $size[z]=1$; $inv[z]$ נקבע בהכנסה (שורה 8) ולא תלוי במבנה העץ – אף פעולה לא משנה אותו.
2. לכל צומת $w$ במסלול מהשורש אל $z$: $size[w] \leftarrow size[w]+1$ – בירידה עצמה (שורות 1–6), $O(\lg n)$.

**שלב 2 – RB-INSERT-FIXUP:** צביעה מחדש לא משנה את השדות, כי הם לא תלויים בצבע. סיבוב משנה רק את שני הצמתים שעל הקשת שסביבה מסתובבים (הספר עמ' 257, איור 14.2):




תיקון לאחר סיבוב שמאלה:

1. $size[x] \leftarrow size[y]$ – תת-העץ של $y$ הוא עכשיו כל תת-העץ שהיה של $x$ (שורה 13 בספר).
2. $size[x] \leftarrow size[\alpha]+size[\beta]+1$ (שורה 14 בספר).

שאר הצמתים לא משתנים: בתת-העץ של כל אחד מהם אותה קבוצת צמתים כמו לפני הסיבוב (תרגיל 14.2-4, הספר עמ' 260). סיבוב ימינה – סימטרי.

**נקבל אפוא כי כל התיקונים עולים $O(1)$ לצומת ו-$O(\lg n)$ בסך הכול, ולכן ההכנסה נשארות בזמן $O(\lg n)$ (משפט 14.1, הספר עמ' 259).**

| INSERT(S, k) | |
|---|---|
| **הרעיון:** ברגע ההכנסה של $k$ נספור כמה מהמפתחות שכבר במבנה גדולים ממנו, ונשמור את המספר בצומת החדש. | |
| **מה נתון:** כל המפתחות שונים. מפתח שיוכנס אחרי $k$ אינו היפוך של $k$, ולכן $inv$ של צומת נקבע פעם אחת ולא משתנה. | |
| **הכלים:**<br>• $S$ – עץ אדום-שחור עם $size$ בכל צומת (עץ ערכי מיקום)<br>• $y$ – מצביע שיורד בעץ<br>• $r$ – מונה המפתחות הקטנים מ-$k$<br>• $z$ – הצומת החדש. | |
| **האלגוריתם:** | **זמן** |
| 1. יורדים מהשורש במסלול ההכנסה של $k$: אם $k < key[y]$ – שמאלה; אחרת $y$ וכל תת-העץ השמאלי שלו קטנים מ-$k$, ולכן מוסיפים ל-$r$ את $size[left[y]] + 1$ ויורדים ימינה – כמו OS-RANK (שורות 1–6; הספר עמ' 255). גובה העץ $\leq 2\lg(n+1)$. | $\Theta(\lg n)$ |
| 2. יוצרים את $z$ עם $key[z] = k$ ו-$size[z] = 1$, ו-$inv[z] \leftarrow size[root[S]] - r$: כל המפתחות שבמבנה פחות אלה שקטנים מ-$k$ (שורות 7–8). | $O(1)$ |
| 3. מכניסים את $z$ ב-RB-INSERT, ומעדכנים $size$: $+1$ לאורך המסלול, ותיקון של שני הצמתים בכל סיבוב ב-$O(1)$ (שורה 9; הספר עמ' 257, משפט 14.1). | $\Theta(\lg n)$ |
| סה"כ סיבוכיות הזמן: $\Theta(\lg n) + O(1) + \Theta(\lg n) = \Theta(\lg n)$ | |

בפסאודו-קוד:

```
INSERT(S, k)
1  r <- 0;  y <- root[S]                    ▷ r = the keys already in S that are < k
2  while y ≠ NIL
3      do if k < key[y]
4            then y <- left[y]
5            else r <- r + size[left[y]] + 1
6                 y <- right[y]
7  z <- a new node: key[z] <- k, size[z] <- 1
8  inv[z] <- size[root[S]] - r              ▷ the keys already in S that are > k
9  RB-INSERT(S, z) keeping size (+1 on the path, fixed in each rotation)
```

| FIND(S, k) | |
|---|---|
| **הרעיון:** חיפוש רגיל בעץ חיפוש. | |
| **האלגוריתם:** | **זמן** |
| 1. TREE-SEARCH מהשורש לפי $k$, בעץ שגובהו $\Theta(\lg n)$ (הספר עמ' 216). | $\Theta(\lg n)$ |
| סה"כ סיבוכיות הזמן: $\Theta(\lg n)$ | |

בפסאודו-קוד:

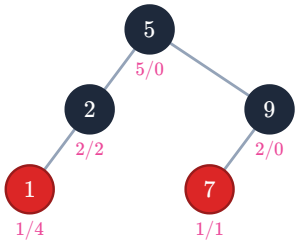
```
FIND(S, k)
1  return TREE-SEARCH(root[S], k)
```

| INVERSIONS(S, k) | |
|---|---|
| **הרעיון:** מוצאים את הצומת של $k$ ומחזירים את מספר ההיפוכים ששמור בו. | |
| **האלגוריתם:** | **זמן** |
| 1. TREE-SEARCH של $k$ (הספר עמ' 216). | $\Theta(\lg n)$ |
| 2. אם לא נמצא – מחזירים $NIL$; אחרת מחזירים $inv[x]$. | $O(1)$ |
| סה"כ סיבוכיות הזמן: $\Theta(\lg n) + O(1) = \Theta(\lg n)$ | |

בפסאודו-קוד:

```
INVERSIONS(S, k)
1  x <- TREE-SEARCH(root[S], k)
2  if x = NIL then return NIL
3  return inv[x]
```

**דוגמה:** מכניסים $5, 9, 2, 7, 1$. כשמכניסים את 2 כבר יש 2 מפתחות, ואף אחד מהם לא קטן מ-2: $inv = 2 - 0 = 2$ (ההיפוכים 5, 9). כשמכניסים את 1: $inv = 4 - 0 = 4$. בשרטוט – העץ האדום-שחור בסוף, ומתחת לכל צומת $inv / size$:




**סיבוכיות מקום:** $\Theta(n)$ – בכל צומת מספר קבוע של שדות.

> **שימו לב:** אין צורך לעדכן את $inv$ של צמתים אחרים בהכנסה – איבר שמוכנס **אחרי** $k$ אינו היפוך של $k$ לפי ההגדרה (הוא היפוך של עצמו, אם $k$ גדול ממנו – וזה נספר ב-$inv$ שלו).

**נקבל אפוא את התשובה הסופית:**

עץ אדום-שחור עם $size$ (עץ ערכי מיקום) ועם $inv[x]$, שנקבע בהכנסה: $inv = size[root] - r$, כאשר $r$ = מספר המפתחות הקטנים מ-$k$ שנספרו בירידה. כל פעולה ב-$\Theta(\lg n)$.