מבני נתונים ומבוא לאלגוריתמים

שיעור הכנה למבחן סמסטר 2003ב

מבחן 2000 ב' מועד 93 שאלה 5

תהי $<p_1, p_2, \ldots, p_n>$ סדרה של n פעולות על מבנה נתונים כלשהו.

כל פעולה מוגדרת או "גדולה" או "קטנה"; ידוע ש- p_1 היא פעולה גדולה ו- p_2 היא פעולה קטנה. זמן הריצה של כל פעולה קטנה הוא תמיד קבוע. זמן הריצה של הפעולה הגדולה הראשונה (p_1) הוא קבוע, אך זמן הריצה של כל פעולה גדולה אחרת הוא פי שניים מזה של הפעולה הגדולה הקודמת. בין כל שתי פעולות גדולות עוקבות נמצאות תמיד מספר קבוע של פעולות קטנות.

מהי העלות לשיעורין של הפעולות בסדרה?

פתרון

ידוע כי בין כל שתי פעולות גדולות עוקבות נמצאות תמיד מספר קבוע של פעולות קטנות. כך לכל היותר בין כל שתי פעולות גדולות יש רק פעולה קטנה אחת. במקרה זה, זמן הביצוע של n פעולות הוא:

$\sum_{i=0}^{n/2} 2^i = \frac{2^{n/2+1}-1}{2-1} = 2^{n/2+1}-1$, כלומר אקספוננציאלי. לפיכך, גם אם נחלק זמן זה ב- n נקבל כי העלות לשיעורין של הפעולות הוא עדיין אקספוננציאלי. כלומר זמן הריצה לשיעורין הוא $O\left(2^n/n\right)$.

פתרון בחינה מסמסטר 1998 ב' מועד 93

שאלה 1

בהינתן קבוצה S של n מספרים ומספר ממשי נוסף z:

א. תארו אלגוריתם שזמן ריצתו $\Theta(n\lg n)$, הקובע האם קיימים ב- S שני איברים שהפרשם בדיוק z.

ב. תארו אלגוריתם שזמן ריצתו $\Theta(n^2)$, הקובע האם קיימים ב- S שלושה איברים שסכומם בדיוק z.

פתרון

א.

נעביר את הקבוצה S למערך A.

נמיין את המערך בעזרת מיון-מיזוג.

לכל $A[i]$ כאשר $1 \le i \le n$

    נחפש את האיבר $z+A[i]$ בעזרת חיפוש בינרי

    אם מצאנו נחזיר "כן"

נחזיר "לא"

זמן ריצה: העברת S למערך בן n איברים – n פעולות. מיון בעזרת מיון-מיזוג $\Theta(n\lg n)$, חיפוש בינרי $\Theta(\lg n)$ המבוצע במקרה הגרוע n פעמים. אז נקבל: $\Theta(n)+\Theta(n\lg n)+n\cdot\Theta(\lg n)=\Theta(n\lg n)$.

ב.

נעביר את הקבוצה S למערך A. ממיינים את המערך A בעזרת האלגוריתם מיון-מיזוג.

לכל איבר w של A, נחפש שני איברים שונים x, y של A, המקיימים את התנאים:

$x + y = z - w,\ y \neq w,\ x \neq w$.

```
TRIPPLES(A)
    n ← length[A]
    MERGE-SORT(A)
    for k ← 1 to n
        do found ← CHECK-SUM(A, z – A[k])
            if found = true and i ≠ k and j ≠ k
                then return true
    return false

CHECK-SUM(A, z)
    i ← 1, j ← n
    while i < j
        do if A[i] + A[j] < z
            then i ← i + 1
        else if A[i] + A[j] > z
            then j ← j – 1
        else
            return true
    return false
```

נכונות השגרה $CHECK\text{-}SUM$:

מערך הקלט ממוין.

אם $A[p] + A[q] < z$, אזי $A[i] + A[q] < z$ לכל $i < p$.

אם $A[p] + A[q] > z$, אזי $A[p] + A[j] > z$ לכל $j > q$.

נניח שקיים זוג אינדקסים (p, q), $p < q$, המקיים $A[p] + A[q] = z$ (אם קיימים יותר מזוג אחד כזה, נבחר אותו זוג שעבורו p מינימלי ו- q מקסימלי).

בזמן הרצת השגרה, מגיעים למצב שבו זוג אינדקסים (i, j) מקיים את התנאי $i = p$, או את התנאי $j = q$.

במקרה הראשון, $j > q$, לכן $A[i] + A[j] > z$ והפעולה הבאה תהיה הפחתת האינדקס j. אחר כך יתקיים $A[i] + A[j] \geq z$.

במקרה השני, $i < p$, לכן $A[i] + A[j] < z$ והפעולה הבאה תהיה קידום האינדקס i. אחר כך יתקיים $A[i] + A[j] \leq z$.

בכל מקרה, השגרה תגיע למצב שבו $A[i] + A[j] = z$ $(i = p, j = q)$.

זמן הריצה של השגרה הוא $\Theta(n)$.

במקרה הגרוע, האלגוריתם מבצע פעם אחת את מיון-מיזוג ו- n פעמים את השגרה $CHECK\text{-}SUM$. לכן זמן הריצה שלו יהיה $\Theta(n^2)$.

שאלה 2

בהינתן רשימה של n תת-קטעים של $[0, 1]$

כתבו אלגוריתם יעיל הקובע האם קיימת נקודה ב- $[0, 1]$ שאינה שייכת לאף אחד מ- n תת-הקטעים.

מהי סיבוכיות האלגוריתם?

פתרון

a – נקודות ההתחלה, b – נקודות הסיום, n – מספר תת-הקטעים

נמיין (נניח במיון-מיזוג) את $2n$ הנקודות לתוך $x[1..2n]$ – לכל קטע יש נקודת התחלה ונקודת סיום, ויש n תת-קטעים – כלומר סה"כ $2n$ נקודות שנכניס למערך x (נקודת התחלת קטע קטנה יותר מנקודת סיום אותו הקטע).

כמו כן לכל נקודה נשמור האם היא נקודת התחלה של קטע או נקודת סיום.

כלומר לכל נקודה $x[i]$: $END\text{-}POINT(x[i]) \leftarrow true$ אם ורק אם הנקודה היא נקודת סיום-קטע.

```
SEGMENTS(a, b, n)
    Sort the n points into x[1..2n]                      ראו הערה לעיל
    for each point x[i]: END-POINT(x[i]) ← true          ראו הערה לעיל
    iff the point was a segment's ending point

    if x[0] > 0 or x[2n] < 1
        then return true
    c ← 0                                                מונה הסופר כמה קטעים נחתכים בנקודה
    for i ← 1 to 2n -1
        do if END-POINT(x[i])                            הסתיים קטע ולכן נפחית את המונה
            then c ← c – 1
            if c = 0                                     נקודה שלא נחתכת באף קטע כלומר לא שייכת
                then return true
        else c ← c + 1                                   מדובר בנקודת התחלה למנות שוב
    return false
```

זמן ריצה: $\Theta(n \lg n)$

שאלה 3

בהינתן שתי רשימות של מספרים, אחת בת m איברים והשניה בת n איברים (m ו- n משתנים בלתי-תלויים):

א. תארו אלגוריתם הקובע האם קיים איבר משותף לשתי הרשימות;

ב. תארו אלגוריתם משופר הפותר את אותה בעיה ושזמן ריצתו טוב יותר מאשר $\Theta(\max\{m \lg m, n \lg n\})$.

פתרון

לשני הסעיפים: נעביר את שתי הרשימות למערכים A ו- B בגדלים n ו- m בהתאמה.

א.

נמיין את שני המערכים בעזרת מיון מיזוג.

לכל $A[i]$ כאשר $1 \le i \le n$

נחפש את $A[i]$ במערך B בעזרת חיפוש בינרי

אם מצאנו, נחזיר "כן"

נחזיר "לא"

ניתוח זמן ריצה: מיון A לוקח $\Theta(n\lg n)$, מיון B לוקח $\Theta(m\lg m)$, מעבר במקרה הגרוע על כל איברי A, כלומר n פעמים וביצוע חיפוש בינרי, כלומר $O(n\lg n)$.

סה״כ זמן ריצה: $\Theta(n\lg n)+\Theta(m\lg m)+O(n\lg n)=\Theta(\max\{m\lg m, n\lg n\})$.

ב.

למעשה אין צורך למיין את שני המערכים – למטרות חיפוש בינרי מספיק מערך אחד ממוין.

נניח בלי הגבלת הכלליות כי $n > m$. אז נמיין את B בלבד בעזרת מיון מיזוג – סה״כ זמן ריצה $\Theta(m\lg m)$. לכל $A[i]$ כאשר $1\leq i\leq n$

נחפש את $A[i]$ במערך B בעזרת חיפוש בינרי

אם מצאנו, נחזיר ״כן״

נחזיר ״לא״

במקרה הגרוע אין איברים משותפים ולכן נעבור על כל איברי A, כלומר סה״כ זמן ריצה של חיפוש בינרי יהיה $O(nlgm)$. מכיוון ש- $n > m$ מתקבל כי $O(nlgn) > O(nlgm)$ ולכן אלגוריתם זה עדיף על פני האלגוריתם מסעיף א׳.

שאלה 4

נתונות שתי ערימות A_1, A_2 בגודל n_1, n_2. נניח ש- $n_1 \geq n_2$ ושכל איבר של A_1 גדול מכל איבר של A_2.

א. הסבירו איך למזג את שתי הערימות לתוך ערימה את בזמן ריצה $O(n_1)$;

ב. הסבירו למה דרוש התנאי $n_1 \geq n_2$;

ג. איך ובאיזה זמן ריצה אפשר למזג את שתי הערימות אם התנאי $n_1 \geq n_2$ מופר.

פתרון

א. נבנה מערך חדש בגודל שני המערכים המכילים את הערימות A_1, A_2. גודל המערך יהיה n_1+n_2. נעתיק את איברי A_1 קודם ואז את איברי A_2. מכיוון שנתון כי כל איבר של A_1 גדול מכל איבר של A_2 נקבל כבר ערימה. זמן הריצה הוא רק זמן ההעתקה שהוא $O(n_1+n_2)$ ומכיוון שנתון כי $n_1 \geq n_2$ אז זמן הריצה הוא $O(n_1)$ כמבוקש.

ב. אם לא היה התנאי $n_1 \geq n_2$ זמן ההעתקה היה $O(n_1+n_2)$ שיכול להיות כי $n_2 > n_1$ ואז זמן הבניה לבדו היה לוקח $O(n_2) > O(n_1)$ בניגוד לדרישות סעיף א׳ שהמיזוג יעשה בזמן $O(n_1)$ בלבד.

ג. במידה והתנאי $n_1 \geq n_2$ מופר, נבנה מערך חדש בגודל שני המערכים המכילים את הערימות A_1, A_2. גודל המערך יהיה n_1+n_2. נעתיק את איברי A_1 קודם ואז את איברי A_2, ונבצע בניית ערימה. זמן הריצה הוא $O(n_1+n_2)$.

שאלה 5

א. נראו כיצד ניתן לממש תור באורך n באמצעות מערך $A[1..n]$. במקום מיקומם של הראש ושל הזנב, המימוש הזה יכלול את מיקומו של הראש ואת אורך התור. הפעולות *ENQUEUE* ו- *DEQUEUE* צריכות להתבצע בזמן $O(1)$.

ב. הראו כיצד ניתן לממש מחסנית באמצעות שני תורים; נתחו את זמן הריצה של הפעולות על המחסנית.

פתרון

א. נדרוש משתנה *len* שיאותחל ב– 0 וייצג את אורך התור. כאשר $len = 0$ התור ריק וכאשר $len = n$ התור מלא. כדי להתייחס למערך נצטרך את *tail*. בהתחלה *tail* יאותחל להיות $head + len$. כאשר בכל הוספת איבר לתור, ומכיוון שתור הוא מעגלי, נבצע את הבדיקה $tail > n$. אם הוספנו איבר ועברנו את גודל המערך, נצטרך לחזור לתחילת המערך ואז נעדכן את *tail* להיות $tail - n$.

ב. מימוש מחסנית ע״י שתי תורים שגודל כל אחד מהם כגודל המחסנית המבוקשת, ההנחה בפתרון הזה היא שניתן להחליף את המצביעים לאובייקטים שקיבלנו באופן שישפיע גם מחוץ לפונקציה, אם ההנחה הזו שגויה ניתן במקום הקריאה ל *exchange* לבצע לולאה שתחליף את תוכן שני התורים

```
Push (Q1, Q2, x)
    Enqueue (Q1,x)

Pop (Q1, Q2)
    if (QueueEmpty(Q1) = true)
        error "underflow"
    else
        x←Dequeue(Q1)
        while(QueueEmpty(Q1)=false)
            Enqueue(Q2,x)
            x← Dequeue(Q1)
        exchange Q1↔Q2          //מעתה ואילך החלפנו את המצביעים לאובייקטים
        return x
```

שאלה 6

נניח ש- S ו- T הינן קבוצות בעלות m ו- n איברים בהתאמה. בחרו במבנה נתונים המאפשר את יישום השגרה *intersection(S, T)* בזמן ריצה $O((m + n)lgm)$.

פתרון

יש לשים לב כי עלינו לממש פעולות מילוניות (הכנסה, מחיקה וחיפוש).

את S ו- T נכניס לשני עצים אדומים שחורים A ו- B בהתאמה. עבור כל איבר בעץ B נחפש האם קיים בעץ A, ואם התשובה היא חיובית – נכניס את האיבר לעץ שלישי $A \cap B$. מכיוון שהפעולות המילוניות על עץ אדום-שחור הן ביחס לגובה העץ נקבל כי זמן הריצה של ההכנסה ל- A ו- B הוא $O(mlgm)$ ו- $O(nlgn)$ בהתאמה. מעבר (סריקה) של כל איברי העץ B הוא $O(n)$ וכל חיפוש ב- A הוא $O(lgm)$. במקרה הגרוע כל איבר של B נמצא ב- A ולכן ההכנסה לעץ החיתוך תעשה על m איברים כלומר בזמן $O(mlgm)$.

סה״כ זמן ריצה: $O(mlgm) + O(nlgn) + O(n) + O(nlgm) + O(mlgm) = O((m + n)lgm)$.

שאלה 1

בהינתן סדרה של n מספרים שונים מאפס

$$a_1, a_2, \ldots, a_n$$

יש לסדר את המספרים מחדש, כך שכל המספרים השליליים יבואו לפני כל המספרים החיוביים; זה חייב להתבצע בעזרת זכרון נוסף בגודל קבוע בלבד.

כתבו אלגוריתם המבצע את הדרוש בזמן $O(n)$.

שאלה 2 (15 נק' לסעיף א', 5 נק' לסעיף ב')

א. בהינתן ערימה A בת n איברים וערך כלשהו z, כתבו שגרה FIND-IN-HEAP (A , z) המחפשת את z ב-A; השגרה תחזיר את האינדקס של איבר שערכו z (אם קיים כזה) או NIL במקרה הנגדי. השגרה חייבת לנצל את תכונת הערימה כדי לשפר את זמן החיפוש.
האם אפשר להוריד את שיעור גידול פונקצית החיפוש (מתחת ל- $\Theta(n)$) או רק את מקדם הפונקציה?

ב. הציגו דוגמא של קלט שעבורו זמן הריצה של השגרה הינו $\Theta(\lg n)$.

שאלה 3 (10 נקודות לכל סעיף)

נתונה השגרה הבאה:

```
SORT (A , n)
    do
        t ← A[1]
        for j ← 2 to n
            do  if  A[j-1]>A[j]
                    then t ← A[j-1]
                         A[j-1] ← A[j]
                         A[j] ← t
    until  t=A[1]
```

א. תארו (במלים) את פעולת האלגוריתם (מה מתבצע בשלב הראשון, השני, האחרון; מהו תנאי העצירה).

ב. מצאו חסם אסימפטוטי הדוק עבור זמן הריצה במקרה הגרוע ביותר.

שאלה 4 (ניקוד: 1 (2) ; 2 (3) ; 3 (3) ; 4 (1) ; 5 (1) ; 6 (10))

מבנה הנתונים "קבוצה" ממומש בצורה של מערך $S[1..N]$, בעל N איברים השונים זה מזה וממויין בסדר עולה; נכתוב |S| כדי לציין את מספר האיברים של S.

עליכם להוכיח שאפשר ליישם את כל אחת מן הפעולות הבאות בזמן הנתון:

1. MEMBER (x , S) (האם x שייך ל-S) - בזמן $\Theta(\lg|S|)$;
2. ADD (x , S) (הכנסת x לתוך S) - בזמן $\Theta(|S|)$;
3. DELETE (x , S) (מחיקת x מתוך S) - בזמן $\Theta(|S|)$;
4. MIN (S) (מציאת המינימום של S) - בזמן $\Theta(1)$;
5. MAX (S) (מציאת המקסימום של S) - בזמן $\Theta(1)$;
6. SUBSET (S_1 , S_2) (האם S_1 תת-קבוצה של S_2) - בזמן $\Theta(\min(|S_2|,|S_1|\cdot\lg(|S_2|))$.

שאלה 5 (10 נק' לכל סעיף)

נתונה סדרה של מספרים ממשיים $(a_0, a_1, a_2, \ldots, a_n)$. נסמן:

$$m = \min\{a_i \ |i = 0, \ldots, n\}$$
$$M = \max\{a_i \ |i = 0, \ldots, n\}$$

א. הוכיחו שקיימים בסדרה שני איברים x ו-y כך ש-

$$|x - y| \le \frac{M - m}{n}$$

ב. כתבו אלגוריתם המוצא את שני האיברים כמתואר בסעיף הקודם; זמן הריצה חייב להיות $O(n)$.

שאלה 6

נתון מערך A של מספרים שלמים, באורך בלתי-מוגבל. n האיברים הראשונים ממויינים בסדר עולה (לא יורד); כל האיברים האחרים מכילים את הערך ∞.

כתבו אלגוריתם הפותר את בעיית חיפוש ערך סופי כלשהו z (ז"א z שונה מ-∞) בין n האיברים הראשונים (יוחזר האינדקס של אחד האיברים שערכם z, או הערך 0 אם z לא נמצא במערך); זמן הריצה של האלגוריתם חייב להיות $O(\lg n)$.

הערה: n הוא משתנה שערכו **אינו ידוע מראש**.

סוף!

מבני נתונים ומבוא לאלגוריתמים (20407)

פתרון מועד 91 - סמסטר 1999א

שאלה 1

השגרה *Partition* מארגנת מחדש מערך סביב *pivot* מסוים. מכיוון שנתון כי 0 אינו איבר במערך ומבקשים כי המספרים השליליים יהיו לפני החיוביים – אך לא מבקשים מיון, כל שעלינו לעשות הוא לשנות בשגרה *Partition* את השורה הראשונה ל- $x \leftarrow 0$ (כמובן שזה אומר כי 0 הוא ה- *pivot*). זמן הריצה של השגרה *Partition* הוא $O(n)$ כנדרש.

שאלה 2

א.

FIND-IN-HEAP(A, z)
    $n \leftarrow heap\_size[A]$
    return HEAP-SEARCH(A, 1, n, z)

HEAP-SEARCH(A, l, r, z)
    If z > A[l] ▷ *z* גדול מהשורש
        then return NIL
    if z = A[l] ▷ *z* הוא השורש
        then return l
    else
        return HEAP-SEARCH(A, Left(l), r)
        return HEAP-SEARCH(A, Right(l), r)

במקרה הגרוע *z* אינו קיים בערימה, אך הוא קטן מכל איבריה ונאלץ לסרוק את כל הערימה. סה״כ זמן ריצה $O(n)$. אין אפשרות להוריד את שיעור פונקצית הגידול מכיוון ש- $n/2$ העלים הם חסרי סדר.

ב.

עבור הערימה $< 16, 14, 10, 8, 7, 9, 3, 2, 4, 1 >$ (הערימה המודגמת בעמוד 132 בספר) ועבור $z = 2$ נקבל כי גובה העץ הוא $h = 4$. אנחנו הולכים לאורך מסלול יחיד – המסלול השמאלי ביותר ומגיעים לעלה *2*. זמן החיפוש $O(lgn)$.

שאלה 3

א. האלגוריתם *SORT* מבצע מיון בועות. בסוף כל הרצה של לולאת ה- *for*, המספר הגדול ביותר יגיע למקומו (בהרצה ראשונה – הגדול ביותר, בהרצה השניה – השני הגדול ביותר וכן הלאה). הלולאה החיצונית ממשיכה לרוץ עד אשר נעשה מעבר אחד בו לא היה שינוי במערך – כלומר המערך כבר ממוין.

ב. במקרה הגרוע ביותר המערך ממוין בסדר הפוך, ואז לולאת ה- *for* רצה כל פעם *n - 1* פעמים. במקרה הגרוע, מכיוון שיש *n* איברים ובכל מעבר של הלולאה, רק האיבר הכבד ביותר יגיע למקומו הנכון, הלולאה תרוץ *n* פעמים. סה״כ זמן ריצה $\Theta\left(n\left(n-1\right)\right)=\Theta\left(n^2\right)$.

שאלה 4

א.

מכיוון שנתון כי *S* הוא מערך ממוין בסדר עולה:

```
MEMBER(x, S)
        Return BINARY-SEARCH(S, x)
```

זמן הריצה של חיפוש בינרי הוא $\Theta\left(\lg|S|\right)$ כמבוקש.

ב.

מכיוון שמדובר במערך, שהוא מבנה נתונים סטטי, עלינו להכניס את האיבר החדש למקומו, "לדחוף" את האיברים ממיקום זה תא אחד קדימה ו"לזרוק" את הנתון האחרון.

```
ADD(x, S)
        i ← 1
        while S[i] < x and i ≤ |S|                 ▷ חיפוש הערך הראשון המקיים S[i] > x
                do i ← i + 1
        if i = |S|                                 ▷ x צריך להיכנס כאיבר אחרון
                then S[|S|] ← x
        else
                while i ≤ |S|
                        do exchange A[i] ↔ x       ▷ הכנסת x למיקומו הנכון ועדכון שאר המערך
                           i ← i + 1
```

ישנן 2 לולאות *while*. לולאה ראשונה עוברת על $|S|$ - *k* איברים (עד *x*) והשניה על השאר. אין פה מקרה גרוע, בכל מקרה עוברים על כל המערך. לכן זמן הריצה הוא ליניארי $\Theta\left(|S|\right)$.

ג.

מכיוון שמדובר במערך, שהוא מבנה נתונים סטטי, אין באמת מחיקת תא. במקום האיבר המועמד למחיקה, נפעפע את הבאים אחריו תא אחד קודם, ובתא האחרון נשים את הערך *(-1)* שעל פי הנתונים אינו איבר במערך, ונעדכן את גודל המערך.

```
DELETE(x, S)
        i ← 1
        while S[i] < x and i ≤ |S|                 ▷ חיפוש הערך הראשון המקיים S[i] > x
                do i ← i + 1
        if i = |S|
                then S[|S|] ← (- 1)
                     |S| ← |S| - 1
        else
                while i ≤ |S| - 1
                        do exchange S[i] ↔ S[i + 1]
                           i ← i + 1
                S[i] ← (-1)
                |S| ← |S| - 1
```

ישנן 2 לולאות *while*. לולאה ראשונה עוברת על $|S|$ - *k* איברים (עד *x*) והשניה על השאר. אין פה מקרה גרוע, בכל מקרה עוברים על כל המערך. לכן זמן הריצה הוא ליניארי $\Theta\left(|S|\right)$.

ד.

מכיוון שנתון כי *S* הוא מערך ממוין בסדר עולה:

```
MIN(S)
    return S[1]
```

וברור כי זמן הריצה הוא קבוע כלומר $\Theta(1)$.

ה.

מכיוון שנתון כי *S* הוא מערך ממוין בסדר עולה:

```
MAX(S)
    return S[|S|]
```

וברור כי זמן הריצה הוא קבוע כלומר $\Theta(1)$.

ו.

```
SUBSET(S1, S2)
    for i ← 1 to |S1|
        do if (BINARY-SEARCH(S2, S1[i]) = NIL)
            then return false
    return true
```

במקרה הגרוע, התשובה חיובית ואז נעבור על כל איברי *S1* ונבצע |*S1*| מעברים בלולאה בסה"כ. לכל איבר נעשה חיפוש בינרי ב- *S2*. לכן סה"כ זמן ריצה יהיה $O\left(|S_1|\lg|S_2|\right)$.

שאלה 5

א.

יהיו $M = a_0^{'}, a_1^{'}, ..., a_n^{'} = m$ אותם מספרים מהסדרה בסדר ממוין. נניח בשלילה כי לכל $a_i^{'}, a_{i+1}^{'}$ מתקיים $\left|a_{i+1}^{'} - a_i^{'}\right| > \frac{M-m}{n}$, אז $M-m = \sum_{i=1}^{n} a_i^{'} - a_{i-1}^{'} \boxed{>} \sum_{i=1}^{n} \frac{M-m}{n} = M-m$ וזו סתירה.

ב.

```
CLOSE(A, i, j)
    if (j – 1) = 1
        then return {A[i], A[j]}
    p ← Select(i, j, ⌊(j−i)/2⌋
    m ← min(A, i, j)
    M ← max(A, i, j)
    if p < m + ⌊(j−i)/2⌋ (M−m)/(j−i)
        then return CLOSE(A, i, i + ⌊(j−i)/2⌋)
    else
        return CLOSE(A, i + ⌊(j−i)/2⌋, j)
```

קריאת הפעלה *CLOSE(A, 1, n)*.

נוכיח נכונות:

עבור $n = 2$ טריוויאלי. (השורה הראשונה תיתן "כן" ולכן נחזיר את שני האיברים הקיימים).

עבור $n > 2$:

$i = 0$, $j = n$, p הוא האיבר במקום ה- $\left\lfloor n/2 \right\rfloor$ ונוכיח באופן כללי עבור p, ערך מיקום k.

נניח כי $p < m + k \cdot \frac{M-m}{n}$. אז נקבל מהרקורסיה שני ערכים x ו- y כך ש- $|x-y| < \frac{p-m}{n}$. אבל עם

הצבת p נקבל $\frac{M-m}{n} = \frac{m + k \cdot \frac{M-m}{n} - m}{k} < \frac{p-m}{k}$ ומהרקורסיה השניה נקבל

$\frac{M-m}{n} \cdot k - m - M \le \frac{M-p}{n-k} < |x-y|$.

שאלה 6

אלגוריתם לחיפוש סוף מערך המספרים (המקום ה- n):

כל עוד לא עברנו את הערך z

קפוץ בקפיצות אינדקס של 2^i

את החיפוש של z נעשה בחיפוש בינרי (נתון כי המערך ממוין).

הלולאה תרוץ במקרה הגרוע על $2n - 2$ תאים בקפיצות של 2^i. זמן ריצה $O(lgn)$.

חיפוש z מתבצע בעזרת חיפוש בינרי. זמן ריצה $O(lgn)$. סה"כ זמן ריצה של האלגוריתם $O(lgn)$.

לשים לב: במקרה הגרוע z אינו נמצא כלל במערך, הקפיצה הלפני אחרונה מביאה אותנו לתא ה- $(n - 1)$

והקפיצה הבאה תיקח אותנו לתא ה- $2^i = 2(n-1) = 2n-2$.

20407 - מבני נתונים ומבוא לאלגוריתמים

שאלה 1

נניח שהסדרה $(a_1, a_2, \ldots, a_n)$ של מספרים ממשיים מקיימת את התנאי

$$a_1 < \ldots < a_m > \ldots > a_n$$

עבור m כלשהו, $1 \le m \le n$.

עליך לכתוב אלגוריתם למציאת ערך אינדקס m בזמן $\Theta(\cancel{n})$.
lgn

שאלה 2 (10 נק' לכל סעיף)

בהינתן סדרה S של מספרים ממשיים ומספר ממשי נוסף z, $z \neq 0$:

א. בהנחה ש-S ממויינת, כתבו אלגוריתם שזמן ריצתו $\Theta(n)$, הקובע האם קיימים ב-S שני איברים שמכפלתם בדיוק z.

ב. ההנחה ש-S איננה ממויינת, כתבו אלגוריתם שתוחלת זמן ריצתו $\Theta(n)$, הקובע האם קיימים ב-S שני איברים שמכפלתם בדיוק z.

אזהרה: שימו לב, הסדרה יכולה להכיל מספרים חיוביים ושליליים.

שאלה 3

נתונה סדרה של n תת-קטעים

$$[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n]$$

של הקטע [0, 1].

עליך לכתוב אלגוריתם המחזיר מספר ממשי x, כך ש-x שייך למספר מקסימלי של קטעים $[a_i, b_i]$.

הערה: האלגוריתם חייב לרוץ בזמן $O(n \cdot \lg n)$ במקרה הגרוע.

שאלה 4 (10 נק' לכל סעיף)

מערך $A[1..n]$ נקרא ״כמעט ממוין עם שגיאה בגודל k $(k < n)$״ אם $A[j] \ge A[i]$ לכל j,i המקיימים $j - i > k$; במלים אחרות, המערך לא חייב להיות ממויין, אבל כל שני אברים הנמצאים בסדר הפוך לא יכולים להיות רחוקים זה מזה יותר מ-k מקומות.

א. איך אפשר לשנות את האלגוריתם מיון-מהיר כך שיהפוך כל קלט לפלט כמעט ממויין עם שגיאה בגודל k? האלגוריתם החדש חייב להיות יעיל יותר מאשר האלגוריתם המקורי. מהו זמן הריצה האסימפטוטי של האלגוריתם החדש במקרה **הטוב** היותר?

ב. נצמצם את האלגוריתם מיון-הכנסה לקלטים כמעט ממויינים עם שגיאה בגודל k. מהו זמן הריצה של האלגוריתם במקרה **הגרוע** ביותר?

שאלה 5 (15 נק' סעיף א', 5 נק' סעיף ב')

בהינתן מערך $A[1..n]$, אנו מבקשים להחזיר בסדר ממויין (לא יורד) את k האיברים הקטנים ביותר של A (n ו-k משתנים בלתי-תלויים, $1 \le k \le n$).

א. איך אפשר לשנות את שגרת האלגוריתם מיון-ערימה כך שהיא תפתור את הבעיה? האלגוריתם המוצע חייב לרוץ בזמן הסימפטוטי טוב יותר מאשר האלגוריתם המקורי. מהו הזמן הזה במקרה הגרוע?

ב. עבור אלו ערכים של k (כפונקציה של n) מתקבל פתרון בסיבוכיות $\Theta(n)$?

שאלה 6

הוכיחו שזמן הריצה של השגרה BUILD-HEAP הינו $O(n)$, תוך שימוש בשיטת החיובים (ניתוח לשיעורין).

סוף!

מבני נתונים ומבוא לאלגוריתמים (20407)

פתרון מועד 93 - סמסטר 1999א

שאלה 1

```
Find_Index(a, l, r)
        j ← ⌊(l+r)/2⌋
        if a[j-1] < a[j] and a[j] >a[j+1]
                then return j
        else if a[j] > a[j+1]
                then Find_Index(a, l, j)
        else
                Find_Index(a, j, r)
```

עבור קלט בן n איברים, בכל פעם נעשות מקסימום 2 בדיקות עבור ערכי הקלט (בשני תנאים – עלות $O(1)$) ועושים רקורסיה על מחצית מגודל הקלט. לכן זמן הריצה הוא $\Theta(n \lg n)$.

שאלה 2

א.

```
Mult(A, z)
        i ← 1
        j ← length[A]
        while i < j
                do if A[i] * A[j] = z
                        then return true
                else if A[i] * A[j] > z
                        then j ← j – 1
                else
                        i ← i + 1
        return false
```

במקרה הגרוע לא קיימים ב- A שני מספרים שמכפלתם שווה בדיוק ל- z, ו- z עצמו מקיים $z < A[1] * A[2]$ או $z > A[n-1] * A[n]$. במקרים אלו, i או j (תלוי מה מקיים z) יעברו על כל המערך בעוד j או i בהתאמה ישארו במקום. כלומר לולאת ה- *while* תתבצע n פעמים ולכן זמן הריצה של השגרה *Mult* הוא $\Theta(n)$.

שאלה 1 (10 נק' לכל סעיף)

א. צייר דוגמא של עץ אדום-שחור בעל 4 צמתים שחורים ו-4 צמתים אדומים.

ב. האם אפשר לבנות עץ אדום-שחור בעל 3 צמתים שחורים ו-6 צמתים אדומים? התשובה חייבת להיות מנומקת היטב.

שאלה 2 (סעיף א' - 5 נק'; סעיף ב' - 15 נק')

נתון מערך $A[1..n]$ המכיל n איברים שונים זה מזה. זוג אינדקסים (i, j) נקרא היפוך אם $i<j$ ו-$A[i]>A[j]$.

א. איזה מערך של איברים מן הקבוצה $\{1,2,...,n\}$ מכיל את המספר הגבוה ביותר של היפוכים? כמה היפוכים הוא מכיל?

ב. כתוב אלגוריתם המחשב את מספר ההיפוכים בתמורה כלשהי של n איברים בזמן $\Theta(n \cdot \lg n)$ במקרה הגרוע.

רמז: פתרון אפשרי מבוסס על הרחבה של מיון-מיזוג.

שאלה 3 (כל סעיף 10 נק')

א. נתבונן בשגרת החלוקה PARTITION (A , p , r), כפי שהיא מתוארת בספר הלימוד (עמ' 143). מחליפים את השורה ה-1 בשורה

$x \leftarrow A[r]$

האם האלגוריתם מיון-מהיר עדיין פועל כהלכה? הסבר.

ב. נתבונן בשגרת החלוקה RANDOMIZED-PARTITION (A, p, r), כפי שהיא מתוארת בספר הלימוד (עמ' 150). מוחקים את השורה ה-2; בשגרה PARTITION (A, p, r) מחליפים את השורה ה-1 בשורה

$x \leftarrow A[i]$

האם האלגוריתם מיון-מהיר האקראי עדיין פועל כהלכה? הסבר.

שאלה 4 (כל סעיף 10 נק')

א. איבר של מערך $A[1..n]$ נקרא איבר רוב, אם ערכו מופיע במערך יותר מאשר $n/2$ פעמים. כתוב אלגוריתם שזמן הביצוע שלו ליניארי, הקובע האם קיים במערך איבר רוב.

ב. כתוב אלגוריתם שזמן הביצוע שלו ליניארי, הקובע האם קיים במערך איבר שערכו מופיע יותר מאשר $n/3$ פעמים.

שאלה 5 (20 נקודות)

הצע מבנה נתונים S שבאמצעותו ניתן לממש את כל אחת מהפעולות הבאות בסיבוכיות המבוקשת:

1. INSERT (k, R, S) - הכנס ל-S את הרשומה R בעלת המפתח k;
2. DELETE (k, S) - מחק מ-S רשומה כלשהי בעלת המפתח k;
3. FIND (k, S) - מצא ב-S רשומה כלשהי בעלת המפתח k;
4. MODE (k, S) - החזר את ערך מפתח בעל שכיחות גבוהה ביותר.

את הפעולות DELETE, INSERT ו-FIND, יש לבצע בזמן $O(\lg n)$, כאשר n הוא מספר המפתחות השונים ב-S (מספר הרשומות יכול להיות הרבה יותר גדול מ-n). את הפעולה MODE יש לבצע בזמן $O(1)$.

הערה: ניקוד חלקי (15 נקודות) יינתן עבור מימוש שבו זמני הביצוע של כל הפעולות הם $O(\lg n)$.

שאלה 6 (20 נקודות)

סדרה של n פעולות מתבצעת על מבנה נתונים. עלות הפעולה ה-i היא i יחידות אם i היא חזקה מדויקת של 3; אחרת, העלות היא 1.
השתמש בשיטת הצבירה לקביעת העלות לשיעורין של כל פעולה.

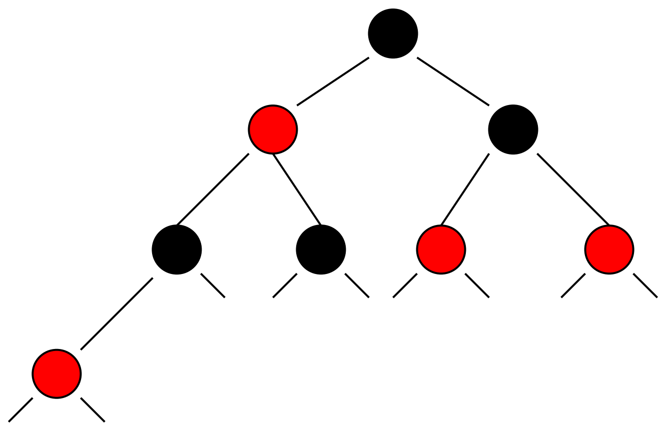
סוף!

מבני נתונים ומבוא לאלגוריתמים

פתרון מועד ב׳ מסמסטר א1999

שאלה 1

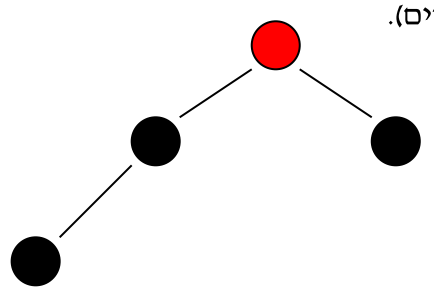
א. דוגמה לעץ אדום-שחור בעל 4 צמתים שחורים ו-4 צמתים אדומים :



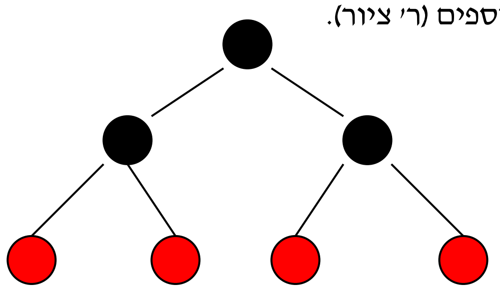
ב. אי אפשר לבנות עץ אדום-שחור בעל 3 צמתים שחורים ו-6 צמתים אדומים.

נוכיח זאת באמצעות בדיקת כל המקרים האפשריים עבור הצומת ושני בניו :

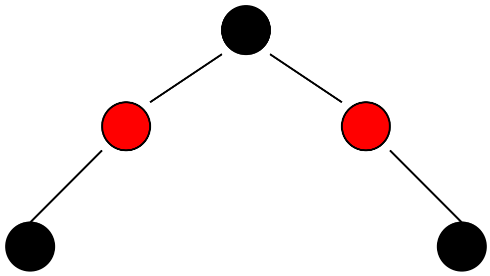
מקרה ראשון: שורש אדום. במקרה זה שני בניו של השורש צריכים להיות שחורים.
הצומת השחור השלישי חייב להיות בנו של אחד משני הצמתים השחורים (ר׳ ציור), וברור שהתנאי הרביעי לא יתקיים (למשל, כל המסלולים הפשוטים מהשורש לצאצאים עלים לא יכילו אותו מספר של צמתים שחורים).



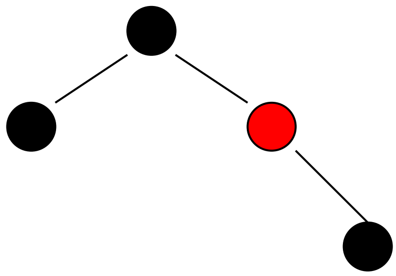
מקרה שני: שורש שחור ושני בנים שחורים. במקרה זה לא ניתן להוסיף לעץ 6 צמתים אדומים מבלי להוסיף גם צמתים שחורים נוספים (ר׳ ציור).



מקרה שלישי: שורש שחור ושני בנים אדומים. במקרה זה צריך להוסיף לעץ עוד שני צמתים שחורים. ברור שאחד מהם יהיה בתת-עץ השמאלי של השורש והשני בתת-עץ הימני של השורש (ר' ציור). ברור שהתנאי הרביעי לא יתקיים (למשל, כל המסלולים הפשוטים מהבן השמאלי של השורש לצאצאים עלים לא יכילו אותו מספר של צמתים שחורים).



מקרה רביעי: שורש שחור שאחד מבניו אדום והשני שחור. במקרה זה צריך להוסיף לעץ עוד צומת שחור אחד. ברור שאפשר להוסיף אותו רק כבן של הצומת האדום (ר' ציור), כי אחרת התנאי הרביעי לא יתקיים עבור השורש. אבל, במקרה זה התנאי הרביעי לא יתקיים עבור הצומת האדום.



שאלה 2

א. המערך $[n, \ldots, 2, 1]$ מכיל את המספר הגבוה ביותר של היפוכים, מפני שכל זוג אינדקסים הוא היפוך. מספר ההיפוכים הוא $\frac{1}{2}n(n-1)$.

ב. הרעיון הוא לספור את ההיפוכים תוך כדי ביצוע השגרה MERGE.

(תיאור של השגרה מופיע בעמודים 10-11 בספר הלימוד, ובתרגיל 1.3-2 בספר היה צריך לכתוב את הפסידו-קוד של השגרה.)

באופן כללי, בעת ביצוע השגרה MERGE משווים איברים מהתת-מערך השמאלי לאיברים בתת-מערך הימני, ומציבים את הקטן מביניהם בתת-מערך הממוזג. כאשר משווים איבר $A[i]$ הנמצא בתת-מערך השמאלי לאיבר $A[j]$ הנמצא בתת-מערך הימני יש שתי אפשרויות:

אם $A[i] \leq A[j]$ אז זוג האינדקסים (i, j) אינו מהווה היפוך.

לעומת זאת, כאשר $A[i] > A[j]$ אז זוג האינדקסים (i, j) מהווה היפוך. יתרה מזאת, גם זוג האינדקסים (x, j) לכל $i + 1 \leq x \leq q$ מהווה היפוך.

נוסיף לאלגוריתם מיון-מיזוג משתנה גלובלי בשם count שיאותחל לאפס.

כאשר $A[i] > A[j]$ נוסיף לשגרה MERGE את השורה: $\text{count} \leftarrow \text{count} + q - i + 1$.

מכיוון שהפעולה שהוספנו מתבצעת בזמן קבוע, זמן הריצה של מיון-מיזוג לא משתנה.

שאלה 3

א. האלגוריתם מיון-מהיר אינו פועל כהלכה לאחר השינוי, מפני שאם $A[r]$ הוא איבר המקסימום במערך, האלגוריתם ייכנס ללולאה אינסופית.

(הערה: השאלה מופיעה במדריך הלמידה!)

ב. האלגוריתם מיון-מהיר האקראי אינו פועל כהלכה לאחר השינוי מסיבה דומה.

כלומר, האלגוריתם ייכנס ללולאה אינסופית אם יתקיימו שני התנאים הבאים:

1. $A[r]$ הוא איבר המקסימום במערך.
2. בכל פעם שתתבצע קריאה ל-RANDOMIZED-PARTITION יוגרל בשורה 1 של השגרה הערך r.

שאלה 4

א. האלגוריתם מתבסס על כך, שאם יש במערך איבר רוב, אז הוא חייב להיות החציון:

MAJORITY-MEMBER (A, n)

$x \leftarrow$ SELECT (A, 1, n, $\lceil n/2 \rceil$)

$i \leftarrow 1$, count $\leftarrow 0$

while $i \le n$ and count $\le n/2$

do if $A[i] = x$

then count $\leftarrow$ count + 1

if count $> n/2$

then return YES

else return NO

ב. הרעיון דומה מאד לסעיף א', אבל הפעם צריך למצוא את ערכי המיקום ה- $\lceil n/3 \rceil$ וה- $\lceil 2n/3 \rceil$.

כלומר, מוצאים באמצעות SELECT את ערך המיקום ה- $\lceil n/3 \rceil$ ובודקים אם הוא מופיע במערך יותר מאשר $n/3$ פעמים. אם כן - מחזירים YES.

אחרת - מוצאים באמצעות SELECT את ערך המיקום ה- $\lceil 2n/3 \rceil$ ובודקים אם הוא מופיע במערך יותר מאשר $n/3$ פעמים. אם כן - מחזירים YES. אחרת - מחזירים NO.

שאלה 5

מבנה הנתונים המוצע: עץ אדום-שחור בעל n צמתים + ערימת מקסימום בגודל n.

לכל אחד מ-n המפתחות השונים זה מזה יהיה "נציג" גם בעץ וגם בערימה.

כל צומת בעץ יורכב משלושה שדות:

1. המפתח
2. מצביע לרשימה מקושרת של הרשומות בעלות אותו מפתח
3. מצביע לצומת המתאים בערימה

הסדר היחסי של האיברים בעץ ייקבע עפ"י שדה המפתח.

כל איבר בערימה יורכב משני שדות:

1. המפתח
2. שכיחות המפתח (כלומר, מספר הרשומות בעץ בעלות אותו מפתח).

הסדר היחסי של האיברים בערימה ייקבע עפ"י השדה של שכיחות המפתח (כלומר, בראש הערימה יימצא המפתח בעל השכיחות המכסימלית).

הפעולות השונות יבוצעו באופן הבא:

INSERT (k, R, S) - נחפש את המפתח k בעץ. אם הוא כבר קיים בעץ, נכניס את הרשומה R לראש הרשימה המקושרת ונגדיל ב-1 את השכיחות של הצומת המתאים בערימה (ייתכן שיהיה צורך לתקן את הערימה). אם המפתח k לא קיים בעץ, נוסיף לעץ צומת חדש בעל המפתח k ונוסיף צומת חדש לערימה בעל שכיחות 1.

DELETE (k, S) - נמצא את המפתח k בעץ ונמחק את הרשומה שבראש הרשימה המקושרת. צריך כמובן גם להחסיר 1 מהשכיחות של הצומת המתאים בערימה. אם הרשומה שנמחקה הייתה הרשומה היחידה בעץ בעלת המפתח k, אז צריך גם למחוק את הצומת בעל המפתח k מהעץ ומהערימה.

FIND (k, S) - נחפש בעץ את הצומת בעל המפתח k, ונחזיר את הרשומה הנמצאת בראש הרשימה המקושרת שבצומת.

MODE (S) - נחזיר את הערך הנמצא בשדה המפתח של הצומת שבראש הערימה.

שאלה 6

השאלה כמעט זהה לשאלה 2 (סעיף ב) בפרק י"ב במדריך הלמידה.

מקבלים שהעלות לשיעורין של כל פעולה היא $\frac{5n-1}{2n}$; כלומר, היא קטנה מ- $2\frac{1}{2}$.

שאלה 1 (10 + 10 נק')

מצא חסמים אסימפטוטיים הדוקים עבור $T(n)$ בכל אחת מנוסחאות הנסיגה שלהלן:

א. $T(1) = c \geq 1 \;\; ; \;\; T(n) = 2T(n/2) + (\lg n)^8$

ב. $T(0) = 0 \;\; ; \;\; T(n) = 2T(n-1) + 1$

שאלה 2 (10 + 10 נק')

נתונים שני מערכים $A[1..n]$ ו- $B[1..n]$. נתבונן בשגרה הבאה:

```
ZSORT (A)
    for i ← 1 to n
      do x ← 1
         for j ← 1  to n
           do  if  A[j] >  A[i]
                  then   x ← x + 1
         B[x] ← A[i]
    for i ← n  downto 1
      do  A[n - i + 1] ← B[i]
```

א. הוכח שהשגרה ממיינת נכון את A.

ב. מהי סיבוכיות הזמן של השגרה? הסבר.

הערה: האיברים של A שונים זה מזה.

שאלה 3 (8 + 6 + 6 נק')

נתבונן באלגוריתמים מיון-הכנסה, מיון-מהיר, מיון-ערימה.
איזה משלושת האלגוריתמים הוא היעיל ביותר כאשר:

א. הקלט ממוין מראש בסדר עולה.

ב. הקלט ממוין מראש בסדר יורד.

ג. היעילות נמדדת באמצעות מספר ההחלפות של איברים בלבד (ז"א ההשוואות לא נספרות).

התשובות חייבות להיות מנומקות היטב.

שאלה 4 (10 + 10 נק׳)

נתון עץ בינרי כלשהו T עם ערכים מספריים (שלמים או ממשיים) המאוחסנים בצמתים.

א. כתוב אלגוריתם יעיל הבודק האם T הוא עץ חיפוש בינרי.
זמן הריצה הדרוש הינו $O(n)$, כאשר n הוא מספר הצמתים של T.

ב. כתוב אלגוריתם יעיל הבודק האם T הוא ערימה. זמן הריצה הדרוש הינו $O(n)$.

שאלה 5 (10 + 10 נק׳)

א. חשב את הגובה המינימלי ואת הגובה המקסימלי של עץ אדום-שחור בן 7 מפתחות.

ב. צייר שני עצים אדומים-שחורים בני 7 צמתים, כאשר גובהו של הראשון מינימלי וגובהו של השני מקסימלי.

שאלה 6

תן פתרון בעזרת שיטת התכנון הדינמי לחישוב המקדמים הבינומיים $\binom{n}{k}$, לכל זוג של שלמים n, k, כאשר $0 \le k \le n$.
מהי סיבוכיות הזמן והמקום של האלגוריתם?

סוף!

מבני נתונים ומבוא לאלגוריתמים
פתרון מועד 99 מסמסטר ב2000

שאלה 1

א. נשתמש במשפט האב. בנוסחה זו: $a = 2$, $b = 2$, $n^{\lg_b a} = n^1 = n$, $f(n) = (\lg n)^8$.

$f(n) = (\lg n)^8 = O(n^{1-\varepsilon})$ (למשל, עבור $\varepsilon = 0.5$)

ולכן לפי מקרה 1 במשפט האב $T(n) = \Theta(n)$.

ב. נשתמש הפעם בשיטת האיטרציה:

$$T(n) = 2T(n-1) + 1 = 2[2T(n-2) + 1] + 1 = 4T(n-2) + 3 = 4[2T(n-3) + 1] + 3 =$$
$$= 8T(n-3) + 7 = ... = 2^i T(n-i) + 2^i - 1 = ... = 2^n T(0) + 2^n - 1 = 2^n - 1$$

(מי שמכיר את בעיית מגדלי האנוי, אולי שם לב שנוסחת הנסיגה מתארת את זמן הריצה של האלגוריתם הרקורסיבי לפתרון הבעיה.)

שאלה 2

א. אופן הפעולה של השגרה מזכיר את מיון-מנייה.

בשלב הראשון, השגרה מחשבת לכל איבר *A[i]* את מספר האיברים **הגדולים** ממנו במערך (ועוד אחד), ומציבה מספר זה במשתנה *x*. לאחר מכן *A[i]* מוצב במקום ה-*x* במערך *B*.

ברור שהמערך *B* המתקבל הוא ממוין בסדר יורד, מפני שבמקום *B[1]* יהיה האיבר הגדול ביותר ב-*A*, במקום *B[2]* יהיה האיבר השני בגודלו ב-*A* וכו'.

בשלב השני, השגרה מעתיקה את איברי *B* חזרה למערך *A* בסדר הפוך, ולכן מקבלים ב-*A* מערך ממוין.

ב. סיבוכיות הזמן של השגרה היא $O(n^2)$, מפני שהשגרה מכילה שתי לולאות מקוננות, שכל אחת מהן מתבצעת בדיוק *n* פעמים.

שאלה 3

א. **קלט ממוין בסדר עולה:** מיון-הכנסה יהיה היעיל ביותר, מפני שזמן הריצה שלו יהיה $O(n)$.

זמן הריצה של מיון-מהיר במקרה זה יהיה $O(n^2)$ וזמן הריצה של מיון-ערמה יהיה $O(n \lg n)$.

ב. **קלט ממוין בסדר יורד:** מיון-ערמה יהיה היעיל ביותר, מפני שזמן הריצה שלו יהיה $O(n \lg n)$.

זמני הריצה של מיון-הכנסה ושל מיון-מהיר יהיו $O(n^2)$.

ג. **כאשר סופרים רק החלפות:** במקרה של קלט ממוין בסדר עולה, מיון-הכנסה יהיה עדיין היעיל ביותר, מפני שבמהלך המיון לא תתבצע אף החלפה.

במקרה של קלט ממוין בסדר יורד, מיון-מהיר (!) יהיה היעיל ביותר, מפני שיתבצעו $n/2$ החלפות (תתבצע החלפה אחת בכל רמה זוגית של עץ הרקורסיה.)

במיון-הכנסה יתבצעו $O(n^2)$ החלפות, ובמיון-ערמה יתבצעו $O(n \lg n)$ החלפות.

שאלה 4

א. אלגוריתם איטרטיבי הבודק אם עץ בינרי T הוא **עץ חיפוש בינרי**:

CHECK-BST (T)

$x \leftarrow$ *TREE-MINIMUM (T)*

$i \leftarrow 1$, $bst \leftarrow$ *TRUE*

while $i \le n - 1$ *and* $bst =$ *TRUE*

  do $y \leftarrow$ *TREE-SUCCESSOR (x)*

    if $key[x] \le key[y]$

      then $i \leftarrow i + 1$

        $x \leftarrow y$

    else $bst \leftarrow$ *FALSE*

return bst

לפי שאלה *13.2-4* בספר, זמן הריצה של האלגוריתם הוא $\Theta(n)$.

ב. אלגוריתם רקורסיבי הבודק אם עץ בינרי T הוא **ערימה**:

CHECK-HEAP (T)

if $T =$ *NIL*

  then return TRUE

  else if (*left*[*T*] $\neq$ *NIL and left*[*T*] > *key*[*T*]) *or* (*right*[*T*] $\neq$ *NIL and right*[*T*] > *key*[*T*])

    then return FALSE

    else return CHECK-HEAP (left[T]) and CHECK-HEAP (right[T])

בכל צומת בעץ מתבצעות שתי קריאות רקורסיביות ולכן זמן הריצה הוא $\Theta(n)$.

שאלה 5

א. הגובה של עץ אדום-שחור המכיל שבעה צמתים פנימיים יהיה מינימלי

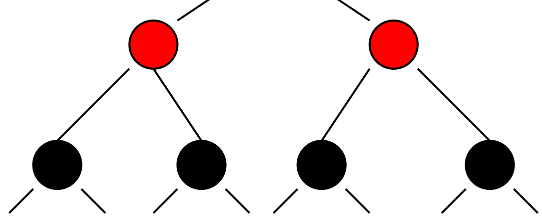
כאשר העץ הוא עץ בינרי **שלם**. גובה העץ במקרה זה הוא $\lfloor \lg n \rfloor = \lfloor \lg 15 \rfloor = 3$

(n מציין כאן את מספר כל צמתי העץ, כולל העלים.)

לפי למה 14.1, חסם עליון על הגובה של עץ אדום-שחור המכיל שבעה צמתים פנימיים

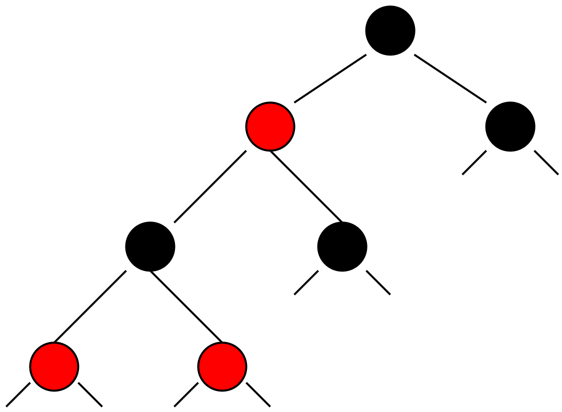
הוא: $2\lg(n + 1) = 2\lg(7 + 1) = 6$.

ב. דוגמה לעץ אדום-שחור המכיל שבעה צמתים פנימיים שגובהו מינימלי:



גובה העץ: 3

דוגמה לעץ אדום-שחור המכיל שבעה צמתים פנימיים שגובהו מקסימלי:



גובה העץ: 4

שאלה 6

המקדמים הבינומיים מקיימים את הזהות הבאה (ר' שאלה 6.1-7 מהספר):

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

כלומר, כדי לדעת את הערך של $\binom{i}{j}$, מספיק לדעת את הערכים של $\binom{i-1}{j}$ ושל $\binom{i-1}{j-1}$.

אלגוריתם תכנון דינמי לחישוב המקדמים הבינומיים ישתמש בטבלה בגודל $(n+1) \times (k+1)$, כאשר הערך שיהיה רשום בתא (i, j) בטבלה יהיה $\binom{i}{j}$.

האלגוריתם ימלא את הטבלה בצורה שיטתית, שורה אחר שורה, בהתאם לזהות שלעיל:

COMB (n, k)

עמודה 0
```
for i ← 0  to  n
  do  C[i, 0] ← 1
```

שורה 0
```
for j ← 1  to  k
  do  C[0, j] ← 0
```

```
for i ← 1  to  n
  do for j ← 1  to  k
       do  C[i, j] ← C[i –1,  j] + C[i –1,  j –1]
```

סיבוכיות הזמן של האלגוריתם הוא $\Theta(n \cdot k)$, מפני שלחישוב כל ערך בטבלה נדרש זמן קבוע.

גם סיבוכיות המקום של האלגוריתם היא, כמובן, $\Theta(n \cdot k)$.

נראה דוגמה לטבלה המתקבלת בעת חישוב $\binom{6}{4}$:

$n\backslash k$	0	1	2	3	4
0	1	0	0	0	0
1	1	1	0	0	0
2	1	2	1	0	0
3	1	3	3	1	0
4	1	4	6	4	1
5	1	5	10	10	5
6	1	6	15	20	15

כפי שניתן לראות, הטבלה שקיבלנו מזכירה מאוד את **משולש פסקל**.
(במשולש פסקל המקדמים הבינומיים מסודרים באופן דומה, ו- $0 \leq k \leq n$)

שאלה 1

מצא חסמים אסימפטוטיים הדוקים עבור נוסחאות הנסיגה שלהלן :

(10 נק') א. $T(n) = 25T(n/5) + 125n^2$

(15 נק') ב. $T(n) = 16T(n^{1/4}) + (\lg n)^4 \cdot \lg\lg n$

התעלם משאלת שלמותם של הערכים ומשאלת תנאי השפה.

שאלה 2

נתון מערך $A[1..n]$ של מספרים. נגביל את האלגוריתם מיון-הכנסה ל- $(k-1)$ האיטרציות הראשונות של הלולאה החיצונית. כידוע, נקבל את התת-מערך $A[1..k]$ ממוין בסדר עולה (או לא-יורד).

(6 נק') א. כתוב שגרה (בפסֵידוקוד) המנצלת את תוצאת המיון החלקי כדי לבצע חיפוש של הערך z במערך $A[1..n]$.

(7 נק') ב. נסמן $m = n - k$ ונתייחס אל m כאל משתנה בלתי-תלוי ב-n $(0 \le m \le n)$. מהו זמן הריצה של השגרה שבסעיף א' במקרה הגרוע כפונקציה של n ושל m?

(6 נק') ג. עבור אילו ערכים של m כפונקציה של n (בסימון אסימפטוטי) ניתן לבצע את פעולת החיפוש בזמן $O(\lg n)$?

(6 נק') ד. כמה פעולות חיפוש (כפונקציה של n) ניתן לבצע בזמן כולל $O(n^2)$ כאשר :

$m = O(\lg n)$; $m = O(n/\lg n)$; $m = O(n)$?

שאלה 3

נתון מערך $A[1..n]$; נבחר איבר ציר x עבור שגרת החלוקה PARTITION ונסמן ב-m את מספר ההופעות של הערך x במערך A.

(7 נק') א. כתוב שגרת חלוקה חדשה M-PARTITION, המחלקת את המערך A לשלושה תת-מערכים המכילים : הראשון, את כל האיברים הקטנים מ-x; השני, את כל ההופעות של x; השלישי, את כל האיברים הגדולים מ-x. זמן הריצה של השגרה יישאר $O(n)$.

(3 נק') ב. כתוב גרסה חדשה של מיון-מהיר M-QUICKSORT, המשתמשת בגרסה החדשה של שגרת החלוקה.

(5 נק') ג. כתוב נוסחת נסיגה עבור זמן הריצה של M-QUICKSORT. אילו פתרונות מתקבלים במקרה הגרוע ובמקרה הטוב? תן הסבר קצר.

(10 נק') ד. כתוב נוסחת נסיגה עבור זמן הריצה של M-QUICKSORT, בהנחה שבכל שלב של הרקורסיה מתקבל ערך של m שווה בקירוב ל- $n/2$.

אילו פתרונות מתקבלים במקרה הגרוע ובמקרה הטוב?

הערה: מותר להתעלם מבעיית שלמותם של הביטויים השונים.

שאלה 4

נתונה טבלת גיבוב $T[1..m]$; כל תא של הטבלה מצביע אל מערך בגודל n. ברצוננו ליישם בכל אחד מ-m המערכים ערימה בינרית. ברצוננו להכניס לטבלת הגיבוב סדרה של n מפתחות בשתי שיטות.

(6 נקי) א. בשיטה הראשונה מכניסים את כל n המפתחות למבנה הנ״ל; אחרי שכולם הוכנסו, בונים את m הערימות. כתוב שגרה המבצעת את הפעולות האלה.

(6 נקי) ב. מהו זמן הריצה של השגרה בסעיף א׳, במקרה הגרוע?
האם יתכן שמתקבל זמן ריצה טוב יותר במקרה הממוצע?

(6 נקי) ג. בשיטה השניה מכניסים את n המפתחות למבנה הנ״ל; אחרי כל הכנסת מפתח, מתקנים את הערימה המתאימה. כתוב שגרה שמבצעת את הפעולות האלה.

(7 נקי) ד. מהו זמן הריצה של השגרה בסעיף ג׳, במקרה הגרוע ובמקרה הממוצע?

שאלה 5

נתון עץ אדום-שחור T המכיל n מפתחות מתוך הסדרה $S = \langle 0, 1, ..., 2^k - 1 \rangle$; נניח שמתקיים $n = O(2^k)$. כל מפתח מאוחסן בתוך מערך בינרי בן k סיביות. פעולת השוואה בין המפתחות מבצעת k פעולות השוואה בין סיביות (כל אחת בזמן קבוע).

(9 נקי) א. מהם זמני הריצה במקרה הגרוע של הפעולות הבאות (כפונקציות של n ושל k): הכנסה לעץ T של מפתח מהסדרה S; חיפוש מפתח מ-S בעץ T ; מחיקת מפתח מ-T, בהינתן מצביע לצומת?

(8 נקי) ב. נניח עכשיו שמתקיים $n = 2^k$. מוסיפים לכל מפתח בעץ יחידה אחת (פעולת קידום כל המפתחות).
תאר (במילים) את אלגוריתם הקידום.

לתשומת לבך : המפתח $2^k - 1$ מקודם ל-0 בעץ החדש.

(8 נקי) ג. מהו זמן הריצה של פעולת קידום כל המפתחות?

סוף!

מבני נתונים ומבוא לאלגוריתמים – פתרון מועד 89 מסמסטר א2002

שאלה 1

א. $\lg_b a = 2$ ולכן, לפי מקרה 2 של משפט האב, מקבלים ש- $T(n) = \Theta(n^2 \lg n)$.

ב. נשתמש בהחלפת משתנים. נסמן $m = \lg n$ ונקבל:

$$T(n) = T(2^m) = 16T(2^{m/4}) + m^4 \cdot \lg m$$

עתה נסמן $S(m) = T(2^m)$ ונקבל את נוסחת הנסיגה החדשה:

$$S(m) = 16S(m/4) + m^4 \cdot \lg m$$

כדי שנוכל להשתמש במקרה 3 של משפט האב, יש להוכיח רגולריות של $f(m) = m^4 \cdot \lg m$.

כלומר, צריך להראות שקיים קבוע $c < 1$ כך שמתקיים $a \cdot f(m/b) \le c \cdot f(m)$

עבור $b = 4$, $a = 16$:

$$16 f(\frac{m}{4}) \le c \cdot f(m)$$

$$16 (\frac{m}{4})^4 \cdot \lg \frac{m}{4} \le c \cdot m^4 \lg m$$

$$\lg(\frac{m}{4}) \le 16c \cdot \lg m$$

$$\lg m - 2 \le 16c \cdot \lg m$$

ואפשר לבחור

$$c = \frac{1}{16}$$

כעת ניתן ליישם את מקרה 3 של משפט האב ונקבל ש- $S(m) = \Theta(m^4 \cdot \lg m)$.

נחזור מ- $S(m)$ ל- $T(n)$: $T(n) = T(2^m) = S(m) = \Theta(m^4 \cdot \lg m) = \Theta(\lg^4 n \cdot \lg\lg n)$

שאלה 2

א. שגרה לחיפוש הערך z במערך $A[1..n]$:

```
FIND-VALUE (A, n, z)
if   A[1] ≤ z ≤ A[k]
    then   i ← BINARY-SEARCH (A, 1, k, z)
             if  i > 0
                then return A[i]
i ← LINEAR-SEARCH (A, k + 1, n, z)
if  i > 0
   then return A[i]
return 0
```

ב. במקרה הגרוע יתבצעו גם החיפוש הבינרי וגם החיפוש הלינארי, ולכן זמן הריצה יהיה:
$T(m,\ n) = O(\lg k + m) = O(\lg(n-m)+m)$.

ג. עבור $m = O(\lg n)$.

ד. $T(m,n) = O(\lg n) \qquad \Leftarrow \qquad m = O(\lg n)$

ולכן ניתן לבצע במקרה זה $O(\frac{n^2}{\lg n})$ פעולות חיפוש בזמן כולל של $O(n^2)$.

$$m = O(\frac{n}{\lg n}) \quad \Rightarrow \quad T(m,\ n) = O(\lg n + \frac{n}{\lg n}) = O(\frac{n}{\lg n})$$

ולכן ניתן לבצע במקרה זה $O(n \cdot \lg n)$ פעולות חיפוש בזמן כולל של $O(n^2)$.

$$m = O(n) \quad \Rightarrow \quad T(m,n) = O(n)$$

ולכן ניתן לבצע במקרה זה $O(n)$ פעולות חיפוש בזמן כולל של $O(n^2)$.

שאלה 4

א. צריך לבצע HASH-INSERT n פעמים, ולאחר מכן לקרוא ל- BUILD-MAX-HEAP m פעמים:

```
for  i ← 1 to n
   do  read (k)
        HASH-INSERT (T[h(k)], k)          ▸ insert to the array at index h(k)
for  i ← 1 to m
   do BUILD-MAX-HEAP (T[i])
```

ב. במקרה הגרוע כל n האיברים יוכנסו לאותו מערך.

הכנסת האיברים לטבלה: $O(n)$

בניית הערימה: $O(n)$

סה"כ: $O(n)$

במקרה הממוצע יהיו m ערימות בעלות $\frac{n}{m}$ איברים כל אחת.

הכנסת האיברים לטבלה: $O(n)$

בניית הערימות: $m \cdot O(\frac{n}{m}) = O(n)$

סה"כ: $O(n)$

ג. במקרה זה צריך לקרוא לשגרה MAX-HEAP-INSERT n פעמים:

```
for i ← 1 to n
   do  read (k)
        MAX-HEAP-INSERT (T[h(k)], k)
```

ד. במקרה הגרוע כל n האיברים יוכנסו לאותו מערך.

הכנסת איבר בודד: $O(1) + O(\lg n) = O(\lg n)$

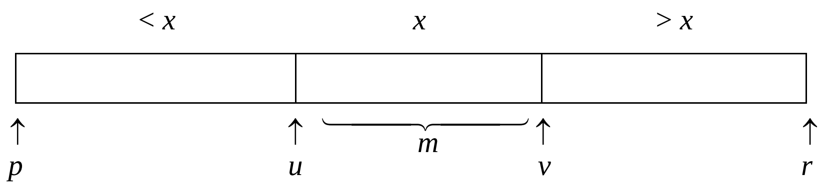
סה"כ: $n \cdot O(\lg n) = O(n \cdot \lg n)$

במקרה הממוצע יהיו m ערימות בעלות $\frac{n}{m}$ איברים כל אחת.

הכנסת איבר בודד: $O(1) + O(\lg \frac{n}{m}) = O(\lg \frac{n}{m})$

סה"כ: $n \cdot O(\lg \frac{n}{m}) = O(n \cdot \lg \frac{n}{m})$

שאלה 3



א. הרעיון הוא לקרוא קודם כל ל- PARTITION כש-x הוא איבר החלוקה.

לאחר מכן נתקן את שני התת-מערכים $A[p..q-1]$ ו- $A[q+1..r]$, כך שב- $A[p..q-1]$ כל האיברים השווים ל-x יהיו בצד ימין, וב- $A[q+1..r]$ כל האיברים השווים ל-x יהיו בצד שמאל.

```
M-PARTITION (A, p, r)
q ← PARTITION (A, p, r)
i ← p – 1  ,  j ← q                        ▸ fixing the left sub-array
while i < j
    do repeat  j ← j – 1
         until    A[j] ≠ x
         repeat   i ← i + 1
         until    A[i] = x
         if  i < j
              then exchange (A[i], A[j])
u ← j
i ← q  ,  j ← r + 1                        ▸ fixing the right sub-array
while i < j
    do repeat  j ← j – 1
         until    A[j] = x
         repeat   i ← i + 1
         until    A[i] ≠ x
         if  i < j
              then exchange (A[i], A[j])
v ← i
return u, v
```

ב. הגרסה החדשה של מיון-מהיר:

```
M-QUICKSORT (A, p, r)
if  p < r
   then   u, v  ← M-PARTITION (A, p, r)
             M-QUICKSORT (A, p, u)
             M-QUICKSORT (A, v, r)
```

ג. נסמן ב- n_L וב- n_R את גודל התת-מערך השמאלי וגודל התת-מערך הימני, בהתאמה.

כלומר: $n_L + n_R = n - m$

נוסחת הנסיגה: $T(n) = T(n_L) + T(n_R) + O(n)$

המקרה הגרוע: $m = 1,\ n_L = 1,\ n_R = n - 2$ (ללא הגבלת הכלליות)

$T(n) = T(n-2) + O(n) = O(n^2)$

המקרה הטוב: $m = n,\ n_L = 0,\ n_R = 0$

$T(n) = O(n)$

ד. כאשר $m \cong \frac{n}{2}$ אז גם $n_L + n_R \cong \frac{n}{2}$.

המקרה הגרוע: $n_L = 1, \quad n_R \cong \frac{n}{2}$ (ללא הגבלת הכלליות)

$T(n) = T(\frac{n}{2}) + O(n) = O(n)$

המקרה הטוב: $n_L = n_R \cong \frac{n}{4}$

$T(n) = 2T(\frac{n}{4}) + O(n) = O(n)$

שאלה 5

א. גובה העץ הוא $O(\lg n)$ והזמן הנדרש לביצוע פעולת השוואה בין מפתחות הוא $O(k) = O(\lg n)$.

לפיכך:

הזמן הנדרש לביצוע הכנסה הוא $O(\lg^2 n)$;

הזמן הנדרש לביצוע חיפוש הוא $O(\lg^2 n)$;

הזמן הנדרש לביצוע מחיקה הוא $O(\lg n)$.

(בעת מחיקה לא מתבצעות פעולות השוואה בין מפתחות!)

שאלה 1

(10 נק׳) א. פתור את נוסחת הנסיגה (פתרון אסימפטוטי הדוק):

$$\begin{cases} T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2 \\ T(1) = 1 \end{cases}$$

(10 נק׳) ב. נתון מערך $A[n]$ של מספרים ממשיים.

כתוב אלגוריתם המוצא בין כל ההפרשים $|A[i] - A[j]|$, $1 \le i \ne j \le n$, את השלישי הקטן ביותר. סיבוכיות האלגוריתם תהיה $O(n \cdot \lg n)$. הוכח.

שאלה 2

הצע מבנה נתונים המבצע את הפעולות הבאות בזמנים הנדרשים:

$BUILD(S)$: בניית המבנה S מתוך סדרה של n מפתחות; זמן: $O(n)$;

$INSERT(S, z)$: הכנסת המפתח z לתוך המבנה S; זמן: $O(\lg n)$;

$DEL\text{-}MIN\text{-}MAX(S)$: מחיקת האיבר המינימלי והאיבר המקסימלי מהמבנה S; זמן: $O(\lg n)$;

$DEL\text{-}THIRDS\ (S)$: מחיקת ערך המיקום ה-$\lfloor n/3 \rfloor$ וערך המיקום ה-$\lceil 2n/3 \rceil$ מהמבנה S;

זמן: $O(\lg n)$;

הערה: המבנה S יכול להיות מורכב ממבנים פשוטים יותר.

שאלה 3

(13 נק׳) א. נתונה סדרה של n תת-קטעים של $[0,1]$:

$[a_i, b_i]$, $0 \le a_i < b_i \le 1$, $i = 1,2,\dots, n$.

ברצוננו להפוך את סדרת התת-קטעים לסדרה חדשה של תת-קטעים שיהיו זרים זה לזה; כל קטע חדש הוא איחוד של כמה תת-קטעים מקוריים.

כתוב אלגוריתם למציאת הקטעים החדשים; זמן הביצוע הנדרש: $O(n \cdot \lg n)$.

(7 נק׳) ב. פתור את נוסחת הנסיגה (פתרון אסימפטוטי הדוק):

$$\begin{cases} T(n) = 4T(n/2) + n + \lg n \\ T(1) = 0 \end{cases}$$

שאלה 4

בהינתן $n < 2^m$ מספרים בעלי m סיביות, הראה כיצד ניתן למצוא מספר נוסף, השונה מכל המספרים הנתונים, בזמן $O(n)$. הנח שהשוואת שני מספרים בעלי m סיביות מתבצעת בזמן קבוע.

שאלה 5

הצע מבנה נתונים המבצע את הפעולות הבאות בזמנים הנדרשים:

$BUILD(S)$: בניית המבנה מתוך סדרה של n מפתחות; זמן: $O(n \cdot \lg n)$;

$INSERT(S, z)$: הכנסת המפתח z לתוך המבנה S; זמן: $O(\lg n)$;

$OLD\ (S,k)$: החזרה (ללא מחיקה) בסדר ממוין, של k המפתחות שנכנסו הראשונים ל-S;
זמן: $O(\min(n, k \lg k))$;

$NEW\ (S,k)$: החזרה (ללא מחיקה) בסדר ממוין, של k המפתחות שנכנסו האחרונים ל-S;
זמן: $O(\min(n, k \lg k))$.

הערה: המבנה S יכול להיות מורכב מכמה מבנים פשוטים יותר.

שאלה 6

נתון מונה בינרי בן k סיביות התומך בפעולת הקידום $INCREMENT$. ידוע שהעלות לשיעורין של פעולה זו היא $O(1)$.

(10 נק') א. הראה שאם מוסיפים למונה את פעולת הנסיגה $DECREMENT$ (חיסור 1), העלות לשיעורין של שתי הפעולות עלולה להגיע ל- $O(k)$.

(10 נק') ב. מה תהיה העלות לשיעורין של הפעולות בהנחה שהפעולה $DECREMENT$ מופעלת רק כאשר ערך המונה אי-זוגי? הוכח.

סוף!

מבני נתונים ומבוא לאלגוריתמים (20407)

פתרון מועד 90 - סמסטר 2002ב

שאלה 3

א.

ראשית נעביר את איברי הסדרה למערך. זמן ריצה $\Theta(n)$.

נמיין את המערך לפי תחילת הקטעים בעזרת מיון מיזוג. זמן ריצה $\Theta(n\lg n)$.

<u>כעת נאחד את הקטעים:</u>

נסרוק את המערך:

בכל שלב נחזיק את הקצה הימני (סוף הקטע המאוחד)

אם הקטע הבא מתחיל לפני נקודה זו

אז מדובר באיחוד – נמשיך

אחרת

נתחיל קטע מאוחד חדש

זמן הריצה של הסריקה: סריקת המערך בעל n האיברים $\Theta(n)$.

סה"כ זמן ריצה של האלגוריתם: $\Theta(n)+\Theta(n\lg n)+\Theta(n)=\Theta(n\lg n)$ כמבוקש.

ב.

$$T(n)=4T\left(n/2\right)+n+\lg n$$

$n^{\lg 4}=n^2, \quad f(n)=n+\lg n=O(n), \quad b=2, \quad a=4$

נבחר $\varepsilon=1/2$. שיטת האב מקרה 1 $\quad T(n)=\Theta(n^2)$

שאלה 1

נתון מערך $A[m+n]$, כאשר m ו-n משתנים בלתי-תלויים זה בזה. נתונה השגרה הבאה:

```
What (A, m, n)
    if  n = 1
        then return  A[m + 1]
    a1 ← What (m, ⌊n/2⌋)
    a2 ← What (m + ⌊n/2⌋, ⌈n/2⌉)
    if  a1 < a2
        then return  a1
        else return  a2
```

(10 נקי) א. מה מבצעת השגרה? הסבר.

(10 נקי) ב. כתוב נוסחת נסיגה עבור זמן הריצה של השגרה. פתור את נוסחת הנסיגה.

שאלה 2

נתונים מערך $A[n]$ של מספרים ממשיים ומספר ממשי נוסף z.

(5 נקי) א. כתוב שגרה (בפסֵידוקוד) למציאת אינדקס k $(1 \le k \le n)$ כך שהאיבר $A[k]$ יהיה מינימלי בין כל האיברים $A[i]$ המקיימים $z \le A[i]$; זמן הריצה הנדרש: $O(n)$.

(7 נקי) ב. כתוב שגרה (בפסֵידוקוד) לפתרון אותה בעיה, בהנחה הנוספת שהמערך A ממוין בסדר עולה (לא יורד); זמן הריצה הנדרש: $O(\lg n)$.

(8 נקי) ג. פתור את נוסחת הנסיגה:

$$\begin{cases} T(n) = T(n-5) + 2n \quad , \quad n \ge 5 \\ T(n) = 0 \quad , \quad n < 5 \end{cases}$$

שאלה 3

נתונה השגרה $MED3$, המוצאת את ערכי המיקום ה- $\lfloor n/3 \rfloor$ וה-$\lceil 2n/3 \rceil$ במערך בגודל n בזמן לינארי ($MED3$ פועלת כקופסה שחורה ולא ידוע שום דבר נוסף עליה).

(13 נק') א. כתוב אלגוריתם שמבצע קריאות ל- $MED3$ והמוצא את ערך המיקום ה- k (k נתון, $1 \le k \le n$) בזמן לינארי. הוכח את זמן הריצה.

(7 נק') ב. האם ניתן לכתוב אלגוריתם שרץ בזמן לינארי והמוצא את **כל** ערכי המיקום (מהמינימום ועד למקסימום) באמצעות קריאות ל- $MED3$? הוכח או הפרך.

שאלה 4

נתונה **ערימת מינימום** $H[n]$: לכל $i > 1$, $H[Parent(i)] \le H[i]$.

מכל איבר בערימה (פרט לשורש) מחסירים את ערך אביו. מתקבל מערך $D[n]$.

(8 נק') א. באיזה סדר עלינו להחסיר את האבות כך שיתאפשר שחזור הערימה המקורית **ללא** שימוש בזיכרון נוסף? כתוב שגרה לבנית המערך D ושגרה לשחזור המערך H.

(12 נק') ב. איך מתבצעת פעולת $INSERT(H,k)$ (הכנסת המפתח החדש k לערימה H) אם משתמשים בצורה D של הערימה? האם זמן הריצה נשמר? הוכח.

שאלה 5

נתונה קבוצה של N רשומות, כאשר כל רשומה R מכילה שני מפתחות מספריים: $key0[R]$ ו-$key1[R]$. יהי n מספר המפתחות $key0$ השונים זה מזה המופיעים ב-N הרשומות (N ו-n הם משתנים בלתי-תלויים זה בזה, $n \le N$).

(12 נק') א. הצע מבנה נתונים, המבוסס על עץ אדום-שחור, המאפשר את ביצוע הפעולות הבאות בזמנים הנדרשים (במקרה הגרוע):

$BUILD(S)$: בניית המבנה S ; זמן: $O(N \cdot \lg n)$;

$SEARCH(S,k)$: חיפוש רשומה כלשהי R, המקיימת $key0[R] = k$, במבנה S; זמן: $O(\lg n)$;

$INSERT(S, k_0, k_1)$: הכנסת רשומה כלשהי R, המקיימת $key0[R] = k_0$, $key1[R] = k_1$, למבנה S; זמן: $O(\lg n)$;

$DELETE\ (S,p)$: מחיקת הרשומה R, שאליה מצביע p, מהמבנה S; זמן: $O(\lg n)$;

$OS(S,i)$: מציאת ערך המיקום ה-i בסדרת n המפתחות השונים $key0$; זמן: $O(\lg n)$;

$INORDER\ (S)$: הדפס את כל הרשומות במבנה S בסדר ממוין לפי $key0$; זמן: $O(N)$.

תאר כל פעולה באופן מלא.

(8 נק') ב. נניח כעת שלכל ערך מפתח $key0$, כל המפתחות $key1$ שונים זה מזה ומספרם לכל היותר m. הסבר איך ניתן לבצע באופן יעיל את הפעולות הבאות:

$SEARCH\ (S, k_0, k_1)$: חיפוש הרשומה R המקיימת את התנאים $key0[R] = k_0$, $key1[R] = k_1$;

$EXTENDED\text{–}OS(S, j)$: מציאת ערך המיקום ה-j בסדרת כל N הרשומות במבנה; לצורך זה נגדיר $R \le R'$ אם ורק אם $key0[R] < key0[R']$ או אם $key0[R] = key0[R']$ וגם $key1[R] \le key1[R']$.

לכל פעולה תן את זמן הריצה האסימפטוטי (במקרה הגרוע) כפונקציה של m ושל n.

שאלה 6

(12 נק׳) א. נתון עץ חיפוש בינרי T בעל N מפתחות; כל מפתח הוא מחרוזת המכילה m תווים לכל היותר. פעולת ההשוואה בין מחרוזות מבוססת על הסדר הלקסיקוגרפי ומבצעת מספר השוואות בין תווים כמספר התווים במחרוזת הקצרה יותר ועוד אחת. נתבונן בפעולות הבאות:

SEARCH (T,s): חיפוש המחרוזת s בעץ T;

INSERT (T,s): הכנסת המחרוזת s לעץ T;

DELETE (T,p): מחיקת האיבר שאליו מצביע p מהעץ T.

ידוע שאחרי כל השוואה בין המחרוזת t שבעץ לבין המחרוזת s, מתבצעת השגרה *LCS-LENGTH*(s,t), המחשבת את אורך התת-מחרוזת המשותפת הארוכה ביותר של s ושל t (עליך להתייחס אל ״מחרוזת״ כאל ״סדרה״ ואל ״תת-מחרוזת״ כאל ״תת-סדרה״).

כל שגרה תחזיר את הערך המקסימלי המתקבל מכל הקריאות לשגרה LCS-LENGTH.

מהם זמני הריצה האסימפטוטיים של שלוש הפעולות כפונקציות של m ושל N?

הוכח כל טענה.

(8 נק׳) ב. פתור את נוסחת הנסיגה הבאה:

$$\begin{cases} T(n) = 2T\left(\sqrt{n}\right) + \lg n \\ T(1) = 0 \end{cases}$$

הפתרון יינתן כחסם אסימפטוטי הדוק של $T(n)$.

סוף!

מבני נתונים ומבוא לאלגוריתמים (20407)

פתרון מועד 92 - סמסטר 2002ב

שאלה 1

(א) השגרה מחזירה את המינימום בתת-המערך *A[m+1...m+n]*.

(ב) הפרמטר *m* נשאר ללא שינוי; מתקבלת נוסחת הנסיגה
$$T(n)=\begin{cases}\Theta(1) & n=1\\ 2T\left(n/2\right)+\Theta(1) & n>1\end{cases}$$
שפתרונה (לפי שיטת האב מקרה 1) הוא $T(n)=\Theta(n)$.

שאלה 2

א.

```
LINEAR-MIN(A, z)
    min ← ∞, k ← 0
    for i ← 1 to n
        do if A[i] ≥ z and min > A[i]
            then k ← i
                 min ← A[i]
    if k > 0
        then return k
        else return "not found"
```

השגרה רצה בזמן *O(n)*.

ב.

```
BINARY-MIN(A, z)
    if  A[n] < z
        then return "not found"
    low ← 1, high ← n
    while low ≤ high
        do if A[low] ≥ z
            then return low
          if A[high] < z
            then return high + 1
          mid ← (low + high) / 2
          if A[mid] = z
            then return mid
            else if A[mid] > z
                then high ← mid – 1
                else low ← mid + 1
```

השגרה פועלת בדומה לחיפוש הבינרי ורצה בזמן *O(lgn)*.

ג.

נשתמש בשיטת האיטרציה:

$$T(n)=T(n-1)+2n=T(n-10)+4n-10=...=T(n-5k)+2kn-2\cdot 5\cdot\sum_{i=1}^{k-1} i$$

עבור $k=\left\lfloor n/5\right\rfloor$: $T(n)=T\left(n-5\cdot\left\lfloor n/5\right\rfloor\right)+2n\cdot\left\lfloor n/5\right\rfloor-5\cdot\left\lfloor n/5\right\rfloor\left(\cdot\left\lfloor n/5\right\rfloor-1\right)=\Theta(n^2)$

שאלה 3

א.

נכתוב אלגוריתם רקורסיבי. הקריאה לשגרה *MEDS3* מחזירה את ערכי המיקום ה- $\left\lfloor n/3 \right\rfloor$ וה- $\left\lceil 2n/3 \right\rceil$.

– אם $k = \left\lfloor n/3 \right\rfloor$ או $k = \left\lceil 2n/3 \right\rceil$, סיימנו; אחרת

– אם $k < \left\lfloor n/3 \right\rfloor$, מבצעים חלוקה סביב ערך המיקום ה- $\left\lfloor n/3 \right\rfloor$ וקוראים ל- *MEDS3* עבור השליש הראשון של המערך; אחרת,

– אם $k > \left\lceil 2n/3 \right\rceil$, מבצעים חלוקה סביב ערך המיקום ה- $\left\lceil 2n/3 \right\rceil$ וקוראים ל- *MEDS3* עבור השליש האמצעי של המערך.

נוסחת הנסיגה של הרקורסיה: $T(n) = T(n/3) + \Theta(n)$.

פתרון הנוסחה (שיטת האב, מקרה 3, הפונקציה $f(n) = n$ רגולרית) הוא: $T(n) = \Theta(n)$.

ב.

לו יכולנו לכתוב אלגוריתם המחזיר את <u>כל</u> ערכי המיקום, היינו מקבלים מיון של המערך, דבר שלא ניתן לבצע בזמן ליניארי.

שאלה 4

א.

במעבר מ- H ל- D חייבים לרוץ מהעלים אל השורש:

$for\ i \leftarrow n\ downto\ 2$

$do\ H[i] \leftarrow D[i] \leftarrow H[i] - H[Parent(i)]$

בשיחזור של H מתוך D חייבים לרוץ בכיוון ההפוך:

$for\ i \leftarrow 2\ to\ n$

$do\ H[i] \leftarrow D[i] + H[Parent(i)]$

השורש לא משתנה: $D[1] = H[1]$.

ב.

מוסיפים את המפתח החדש לסוף המערך. מסמנים את המסלול מהאיבר החדש כלפי מעלה, עד השורש (ניתן להשתמש במערך של מצביעים בגודל $\lceil \lg n \rceil$); זמן: $O(lgn)$.

יורדים מהשורש לאורך המסלול המסומן ומשחזרים את הערכים המקוריים של H (לכל איבר במסלול וגם לבן השני שלו, אם הוא קיים); זמן: $O(lgn)$.

מבצעים את תיקון הערימה בעליה על המסלול המסומן; זמן $O(lgn)$.

עולים לאורך המסלול המסומן ומחשבים את ההפרשים (הערכים של D) עבור כל איבר על המסלול (וגם עבור אחיו, אם הוא קיים); זמן: $O(lgn)$.

שאלה 5

א.

מבנה הנתונים S יהיה עץ אדום-שחור, עם רשימה מקושרת מחוברת לכל צומת (הצומת מכיל מצביע אל ראש הרשימה); מוסיפים לכל צומת בעץ שדה *size* שמכיל את מספר הצמתים שבעץ המושרש בצומת.

BUILD(S) : בנית המבנה מורכבת מ- N פעולות הכנסה, כל אחת בזמן $O(lgn)$ (ראו למטה);

SEARCH(S, k) : מחפשים בעץ את המפתח k. אם מצאנו, בוחרים את הרשומה R שבראש הרשימה בצומת;

INSERT(S, K_0, K_1) : מבצעים פעולת הכנסה לעץ; אם המפתח K_0 כבר נמצא, מוסיפים את R לראש הרשימה בצומת; אחרת, יוצרים צומת חדש בעל המפתח K_0 ובונים רשימה חדשה שבראשה R.

DELETE(S, p) : מוחקים את הרשומה R מהרשימה; אם R היתה הרשומה היחידה, מוחקים את הצומת מהעץ;

OS(S, i) : השגרה *OS-SELECT(S, i)* נותנת לנו את ערך המיקום ה- i.

INORDER(S) : מבצעים סריקה תוכית; בכל צומת של העץ, מדפיסים את כל הרשומות שברשימה המקושרת; סה"כ N רשומות.

ב.

מאחסנים בשדה *size* לא את מספר הצמתים שבתת-העץ, אלא את מספר הרשומות שבו.

SEARCH(S, K_0, K_1) : מחפשים בעץ את המפתח K_0. אם מצאנו, מבצעים חיפוש ליניארי הרשימה המקושרת אחר המפתח K_1, זמן הריצה : $O(lgn + m)$;

EXTENDED-OS(S, j) : משנים את השגרה *OS-SELECT* כך שתעבור עם מספר רשומות; בכל צומת שאליו הגענו, חייבים לספור את הרשומות; זמן הריצה : $O(mlgn)$.

אם מתחזקים בכל צומת שדה המכיל את מספר הרשומות, זמן הריצה יורד ל- $O(lgn + m)$.

שאלה 6

א.

כל פעולת השוואה בין s לבין מפתח t בעץ מתבצעת בזמן $O(m)$. הפעולה *LCS-LENGTH(s, t)* מתבצעת בזמן $O(m^2)$.

SEARCH(T, s) : מבצעת $O(N)$ פעולות השוואה בין s לבין מפתחות בעץ; זמן הריצה הוא :

$O\left(mN + m^2N\right) = O\left(m^2N\right)$.

INSERT(T, s) : כמו במקרה החיפוש; זמן הריצה הוא $O\left(m^2N\right)$.

DELETE(T, p) : לא מבצעת פעולות השוואה בין מפתחות בעץ; זמן הריצה : $O(N)$.

ב.

מחליפים $n = 2^m$ $\left(m = \lg n\right)$; מתקבלת נוסחת הנסיגה :

$$S\left(m\right) = T\left(2^m\right) = 2T\left(2^{m/2}\right) + m = 2S\left(m/2\right) + m$$

פתרון הנוסחה (לפי שיטת האב, מקרה 2) : $S(m) = O(mlgm)$ ומזה נובע הפתרון : $T(n) = O(lgnlglgn)$.

שאלה 1

תור קדימויות מוגדר כמבנה נתונים S התומך בשתי הפעולות הבאות:

$\textsc{Insert}(z,S)$: הכנסת המפתח z למבנה S;

$\textsc{Delete-Min}(S)$: מציאת, החזרת ומחיקת המפתח המינימלי מהמבנה S.

כידוע, ניתן לממש תור קדימויות בערימה בינרית.

(7 נק') א. איך ניתן לממש תור קדימויות בעזרת
- רשימה מקושרת רגילה;
- רשימה מקושרת ממוינת;
- עץ חיפוש בינרי;
- עץ אדום-שחור?

(6 נק') ב. מהו זמן הריצה של כל פעולה בכל מימוש?

(7 נק') ג. איזה מימוש עדיף אם
- מספר פעולות המחיקה הוא קבוע;
- מספר פעולות המחיקה הוא בסדר גודל של מספר פעולות ההכנסה?

שאלה 2 (20 נקודות)

נתון עץ בינרי T. הרמה ה- d של העץ מוגדרת כאוסף כל הצמתים הנמצאים בעומק d (יחסית לשורש). לדוגמא: הרמה 0 מכילה את השורש, הרמה 1 מכילה את הבנים של השורש ; הרמה $d+1$ מכילה את כל הבנים של הצמתים שברמה d.

סריקה ברמות של העץ T היא פעולה על העץ המחזירה את הצמתים של כל רמה משמאל לימין, החל מהרמה 0 ועד הרמה המכסימלית.

כתבו אלגוריתם המבצע סריקה ברמות של העץ T בזמן $O(n)$ (n הוא מספר הצמתים בעץ).

הסבירו מדוע האלגוריתם שהצעתם פועל נכון.

שאלה 3 (20 נקודות)

נתונה סדרה של n תת-קטעים

$$[a_1,b_1],[a_2,b_2],\ldots,[a_n,b_n]$$

של הקטע $[0,1]$.

כתבו אלגוריתם המחשב את מספר הזוגות (i,j), $1\le i,j\le n$ המקיימים את התנאי $b_i < a_j$.

(במילים אחרות, אנו רוצים לדעת כמה זוגות של תת-קטעים זרים זה לזה קיימים בסדרה.)

האלגוריתם חייב לרוץ בזמן $O(n\lg n)$ במקרה הגרוע.

שאלה 4

בהינתן מערך S של n מספרים ממשיים שונים זה מזה ומספר ממשי נוסף z:

(10 נק') א. כתבו אלגוריתם שזמן ריצתו $\Theta(n\lg n)$, הסופר את מספר הזוגות (x, y) של איברים ב-S המקיימים את התנאי $x + y \le z$.

(10 נק') ב. כתבו אלגוריתם שזמן ריצתו $\Theta(n^2)$, הסופר את מספר השלשות (u, v, w) של איברים ב-S המקיימות את התנאי $u + v + w = z$.

הערה: מותר להתעלם מההבדל בין זוג סדור / לא סדור ובין שלשה סדורה / לא סדורה (כלומר, מותר לספור את הזוג (x, y) וגם את הזוג (y, x); בדומה עבור שלשות).
אין חובה לכתוב פסידוקוד.

שאלה 5

נניח שממשים ערימות בינריות בעצים (בעזרת מצביעים) במקום במערכים.
ברצוננו למזג שתי ערימות שלמות: H_a בת $2^a - 1$ צמתים ו-H_b בת $2^b - 1$ צמתים, כך שהתוצאה תהיה גם היא ערימה בינרית בת $n = 2^a + 2^b - 2$ צמתים.

(6 נק') א. כתבו אלגוריתם למיזוג שתי הערימות עבור המקרה $a = b$, שרץ בזמן $O(\lg n)$.

(6 נק') ב. כתבו אלגוריתם למיזוג שתי הערימות עבור המקרה $|a - b| = 1$ שרץ בזמן $O(\lg n)$.

(8 נק') ג. כתבו אלגוריתם למיזוג שתי הערימות עבור a, b כלשהם, שרץ בזמן $O(\lg^2 n)$.

הערה: אין חובה לכתוב פסידוקוד.

שאלה 6

נתונים n פריטים. משקלו של הפריט ה-i הוא w_i (w_i שלם). בהינתן מספר שלם נוסף W, ברצוננו לקבוע אם קיימת תת-קבוצה של הפריטים, כך שמשקל הפריטים בתת-קבוצה הוא **בדיוק** W.
(במקרה שקיימת תת-קבוצה כזו, ברצוננו לדעת את הרכבה).
למשל, כאשר $W = 10$ ומשקלי הפריטים הם $\{5, 2, 6, 3\}$, ניתן להגיע למשקל של 10 ע"י בחירת הפריטים שמשקליהם 5,2,3.

(12 נק') א. כתבו אלגוריתם תכנון דינמי לפתרון הבעיה, שרץ בזמן $O(nW)$.

(8 נק') ב. הריצו את האלגוריתם שכתבתם על הדוגמא לעיל, והראו כיצד האלגוריתם מוצא את הפתרון לבעיה.

סוף!

מבני נתונים ומבוא לאלגוריתמים – פתרון מועד 87 מסמסטר א2003

תשובה 1

<u>סעיפים א+ב:</u>

נתאר את המימוש של שתי הפעולות וננתח את המימוש שהצענו עבור כל אחד מהמבנים המבוקשים

I. רשימה מקושרת רגילה:

$\textsc{Insert}(z,L)$ – נשתמש בהכנסה רגילה לראש רשימה. סיבוכיות הזמן של הפעולה היא $O(1)$. (פרק 10, עמוד 172)

$\textsc{Delete-Min}(L)$ – נבצע את שתי הפעולות הבאות:

1. נעבור על הרשימה ונמצא את המינימום – $O(n)$.
2. נמחק את המינימום מהרשימה – $O(1)$.

בסה"כ זמן הריצה של הפעולה הוא $O(n)$.

II. רשימה מקושרת ממוינת:

$\textsc{Insert}(z,L)$ – נבצע את שתי הפעולות הבאות:

1. נעבור על הרשימה עד שנמצא את המקום המתאים – $O(n)$.
התנאי לכך שהמקום המתאים הוא אחרי p הוא:
$$(key[p] \le z) \wedge (key[next[p]] \ge z)$$
2. נכניס את z אחרי המקום שמצאנו – $O(1)$. (אין להכנסה כזו מימוש בספר, אבל ניתן לממש אותה בדומה לשגרה $\textsc{List-Insert}$ כאשר מחליפים כל מופע של $head[L]$ בצומת שאחריו רוצים להכניס את האיבר החדש.)

בסה"כ זמן הריצה של הפעולה הוא $O(n)$.

$\textsc{Delete-Min}(L)$ – המינימום נמצא בראש הרשימה, ולכן צריך למחוק אותו ולהחזיר אותו בסה"כ – $O(1)$.

III. עץ חיפוש בינרי:

$\textsc{Insert}(z,L)$ – הכנסה רגילה לעץ חיפוש בינרי – $O(h)$.

$\textsc{Delete-Min}(L)$ – חיפוש מינימום בעץ בינרי ומחיקתו – $O(h)$.

(h מציין את גובה העץ)

מכיוון שבמקרה הגרוע $h = O(n)$, הרי שזמן הריצה הוא $O(n)$ עבור שתי הפעולות.

IV. עץ אדום-שחור:

מכיוון שעץ אדום-שחור הוא עץ חיפוש בינרי שגובהו $h = O(\lg n)$, ומכיוון שלכל הפעולות שבהן השתמשנו ב-III יש פעולות אנלוגיות בעץ אדום-שחור, הרי שמימוש שתי הפעולות אנלוגי וזמן הריצה הוא $O(\lg n)$.

סעיף ג:

נארגן את מה שהוכחנו בסעיפים א+ב בתוך טבלה:

אופן המימוש	זמן ריצה של $O(n)$ פעולות הכנסה	זמן ריצה של c פעולות מחיקה	זמן ריצה של $O(n)$ פעולות מחיקה
רשימה מקושרת רגילה	$O(n)$	$cO(n)=O(n)$	$O(n^2)$
רשימה מקושרת ממוינת	$O(n^2)$	$cO(1)=O(1)$	$O(n)$
עץ חיפוש בינרי	$O(n^2)$	$cO(n)=O(n)$	$O(n^2)$
עץ א"ש	$O(n\lg n)$	$cO(\lg n)=O(\lg n)$	$O(n\lg n)$

המימוש הטוב ביותר עבור מספר קבוע של מחיקות הוא מימוש בעזרת רשימה מקושרת רגילה, והמימוש הטוב ביותר כאשר מספר המחיקות הוא בסדר גודל של מספר פעולות ההכנסה הוא מימוש באמצעות עץ אדום-שחור.

תשובה 2

נשתמש בתור עזר Q שיכיל את האיבר הבא לסריקה על-פי רמות:

$\text{BFS}(T)$

1 $\text{Enqueue}(Q, root[T])$
2 **while** not $\text{IsEmpty}(Q)$
3 **do** $p \leftarrow \text{Dequeue}(Q)$
4 **if** $left[p] \neq \text{Nil}$
5 **then** $\text{Enqueue}(Q, left[p])$
6 **if** $right[p] \neq \text{Nil}$
7 **then** $\text{Enqueue}(Q, right[p])$
8 print $key[p]$

האלגוריתם מכניס לתור את שורש העץ, ולאחר מכן מבצע לולאה המסתיימת כאשר התור מתרוקן. בכל איטרציה של הלולאה מוציאים מהתור את האיבר שבראש התור, מדפיסים אותו ומכניסים לתור את שני בניו (אם הם קיימים), הבן השמאלי ואחריו הבן הימני.

נוכיח את נכונות האלגוריתם.

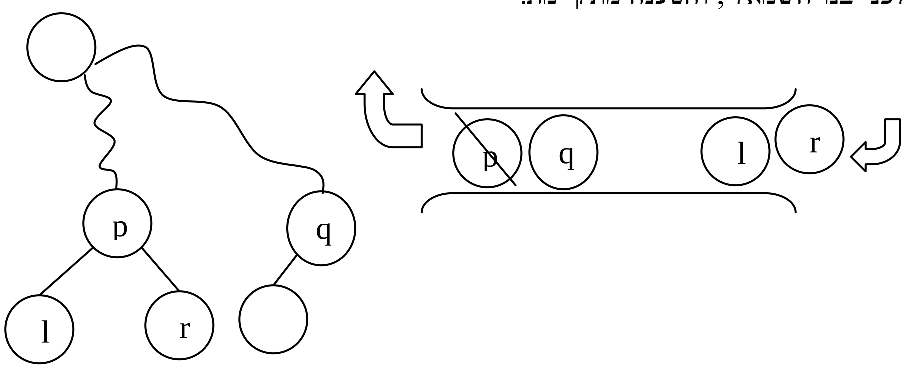
טענה: לפני כל איטרציה של הלולאה בשורה 2, התור מכיל את הצמתים מהצומת שבראש התור ועד לצומת שלפני בנו השמאלי, בסדר המוגדר על-ידי סריקה לפי רמות.

נוכיח את הטענה באינדוקציה על מספר האיטרציות.

לפני האיטרציה הראשונה, התור מכיל אך ורק את השורש והטענה מתקיימת.

נניח כעת כי הטענה מתקיימת לפני איטרציה i והתור מכיל את צמתי העץ בסריקה לפי רמות החל בצומת שבראש התור (להלן p) ועד לצומת שלפני בנו השמאלי בסריקה לפי רמות. נפריד לשני מקרים:

1. p אינו הצומת האחרון ברמה שלו. נסמן ב- q את האיבר הנמצא בתור אחרי p . עפ"י הנחת האינדוקציה, זהו האיבר הנמצא מימין ל- p בעץ. אחרי הוצאת p מהתור מכניסים לתור את שני בניו. על-פי הנחת האינדוקציה, התור לפני הוצאת p הכיל את כל הצמתים בעץ שבין p לבין הצומת שלפני בנו השמאלי של p (בסדר המוגדר על-ידי סריקה לפי רמות). כעת, בניו של p נמצאים בעץ משמאל לבניו של q (ראו באיור), ולכן לפני האיטרציה הבאה התור יכיל את כל הצמתים בעץ שבין q לבין הצומת שלפני בנו השמאלי, והטענה מתקיימת.



2. p הוא הצומת האחרון ברמה שלו. במקרה זה בניו של p הם האחרונים ברמה שלהם. נסמן ב-q את האיבר הנמצא בתור אחרי p. עפ"י הנחת האינדוקציה, זהו האיבר הראשון ברמה הבאה בעץ והתור מכיל את כל הצמתים בעץ שבין p לבין הצומת שלפני בנו השמאלי של p (בסדר המוגדר על-ידי סריקה לפי רמות). שני בניו של q הם הראשונים ברמה שמתחת לבנים של p, ולכן לאחר הוצאת p והכנסת שני בניו התור יכיל את כל הצמתים בעץ שבין q לבין הצומת שלפני בנו השמאלי. כלומר, הטענה מתקיימת.

קיבלנו שאיברי התור מסודרים בדיוק בסדר המוגדר ע"י סריקת לפי רמות. מכיוון שהאיברים מוצאים מהתור ומודפסים זה אחר זה, האלגוריתם מבצע סריקה ברמות של העץ.

נשים לב שכל צומת בעץ מוכנס לתור פעם אחת ומוצא מהתור פעם אחת, ולכן זמן הריצה של האלגוריתם הוא $O(n)$, כנדרש.

תשובה 3

נניח שכל הנקודות שונות זו מזו ולכל i $a_i < b_i$.

תיאור האלגוריתם:

1. נבנה מערך בן $2n$ תאים, כאשר כל תא מכיל שני שדות – $value$ ו-$type$. השדה $value$ יכיל את ערך הנקודה, והשדה $type$ יכיל את הערך a-type אם הנקודה היא מסוג a_i ואת הערך b-type אם היא מסוג b_i.
2. נמיין את המערך לפי השדה $value$ (למשל, באמצעות מיון-ערמה).
3. נעבור על המערך משמאל לימין ונשמור את מספר הנקודות מסוג b_i שחלפנו על פניהן במשתנה $BCounter$. בכל פעם שנגיע לנקודה מסוג a_i, נוסיף את $BCounter$ למספר הזוגות של תת-קטעים זרים, מפני שהנקודה a_i גדולה מכל הנקודות מסוג b_i שלפניה.

להלן האלגוריתם:

```
CountDisjoint(A, B)
1   n ← length[A]
2   for i ← 1 to n
3       do   value[C[i]] ← A[i]
4            type[C[i]] ← a-type
5            value[C[n+i]] ← B[i]
6            type[C[n+i]] ← b-type
7   HeapSort(C)
8   BCounter ← 0
9   PairCounter ← 0
10  for i ← 2 to n−1
11      do if type[C[i]] = b-type
12          then Bcounter ← Bcounter + 1
13          else PairCounter ← PairCounter + Bcounter
14  return PairCounter
```

ננתח את סיבוכיות זמן הריצה של האלגוריתם:

זמן הריצה של לולאת ה-for בשורה 2 הוא $O(n)$, מכיוון שבכל איטרציה של הלולאה מבוצע מספר קבוע של פעולות.

זמן הריצה של שורה 8 הוא:

$$O(2n\lg 2n) = O(n\lg 2 + n\lg n) = O(n\lg n)$$

זמן הריצה של לולאת ה-for בשורה 10 הוא גם-כן $O(n)$.

בסה"כ סיבוכיות זמן הריצה של האלגוריתם היא:

$$O(n) + O(n\lg n) + O(n) = O(n\lg n)$$

תשובה 4

<u>סעיף א:</u>

נשתמש בגרסה שונה מעט של חיפוש בינרי: במקרה שהאיבר שמחפשים אינו נמצא במערך הממוין, אז השגרה תחזיר את האינדקס של האיבר הכי גדול שקטן ממנו. אם אין כזה, היא תחזיר 0. זמן הריצה של גרסה זו הוא גם-כן $O(\lg n)$.

נתאר את האלגוריתם הדרוש:

1. נמיין את S באמצעות מיון אופטימלי. (למשל מיון-ערמה)
2. עבור כל איבר $S[i]$ ב-S, נחפש את $z-S[i]$ בתוך S באמצעות שגרת החיפוש המורחבת, ונסכום את כל האינדקסים שקיבלנו.
3. נחזיר את הסכום.

נסביר מדוע האלגוריתם מבצע את הדרוש וננתח את סיבוכיות הזמן שלו:
שגרת החיפוש תחזיר את האינדקס הגדול ביותר j שעבורו $S[j]\le z-S[i]$. כלומר, $S[j]+S[i]\le z$. מכיוון שלכל $j^*\le j$ מתקיים $S[j^*]\le S[j]$, נקבל שיש בדיוק j איברים ב-S שעבורם $S[j]+S[i]\le z$. אם נסכום עבור כל i את האינדקסים הללו, נקבל את התוצאה הדרושה.

הערה: בספירה יכולים להיות גם זוגות מהצורה (j,j). ניתן להרחיב את האלגוריתם כך שיטפל במקרה קצה זה בלי להשפיע על הסיבוכיות

סיבוכיות הזמן של המיון בשלב 1 היא $\Theta(n\lg n)$.

בשלב 2 קוראים לשגרת העזר n פעמים, ולכן סיבוכיות הזמן של שלב 2 היא $\Theta(n\lg n)$.

בסה"כ סיבוכיות הזמן של האלגוריתם היא $\Theta(n\lg n)$, כנדרש.

<u>סעיף ב:</u>

ראשית, נתאר שגרה שסופרת במערך ממוין A את מספר הזוגות (a,b) המקיימים $a+b=c$ עבור c נתון.

1. נצביע על שני קצות המערך.
2. נבצע עד אשר שני המצביעים נפגשים:
   2.1. אם סכום האיברים שמצביעים עליהם גדול מ-c, נזיז את המצביע הימני שמאלה.
   2.2. אם הסכום קטן מ-c, נזיז את המצביע השמאלי ימינה.
   2.3. אם הסכום שווה ל-c, נגדיל את המונה ונזיז את המצביע הימני שמאלה.

נסביר את פעולת השגרה וננתח את סיבוכיותה:
אם הסכום גדול מ-c: כל איבר הנמצא מימין למצביע הימני גדול יותר מהאיבר שעליו מצביע המצביע הימני, ולכן לא ייתכן שהסכום שלו ושל האיבר שעליו מצביע המצביע השמאלי יהיה שווה ל-c. לכן מזיזים את המצביע הימני שמאלה.
בצורה דומה מטפלים במקרה שהסכום קטן מ-c.
אם הסכום שווה ל-c, אז מצאנו זוג אחד, וממשיכים לסרוק את המערך בחיפוש אחר זוגות נוספים.
(הבחירה להזיז את המצביע הימני שמאלה היא שרירותית.)

ניתוח האלגוריתם פשוט: בסה"כ עוברים על כל איבר במערך פעם אחת פעם אחת – עם המצביע הימני או עם השמאלי. מכיוון שבכל איטרציה מבצעים מספר קבוע של פעולות, הרי שסיבוכיות האלגוריתם היא $\Theta(n)$, כאשר $n = length[A]$.

וכעת, נתאר אלגוריתם לבעיה הנתונה:

1. נמיין את המערך באמצעות מיון אופטימלי. (למשל מיון-ערמה)
2. עבור כל איבר $S[i]$, נספור את מספר הזוגות במערך שסכומם $z - S[i]$ ונסכם את כל המספרים הללו.
3. נחזיר את הסכום שקיבלנו.

שוב, יהיו זוגות לא חוקיים שספרנו, אבל ניתן לטפל במקרי קצה אלו בלי לשנות את מהות האלגוריתם

כעת ננתח את סיבוכיות זמן הריצה:
שלב 1 – $\Theta(n \lg n)$.
שלב 2 – $\Theta(n^2)$.
שלב 3 – $O(1)$
בסה"כ סיבוכיות זמן הריצה היא $\Theta(n^2)$, כנדרש.

תשובה 5

סעיף א:

1. ניקח את העלה הכי ימני ב- H_b ונוציא אותו מהערמה.
   נשים לב, שניתן להגיע לאיבר האחרון ב-H_b בזמן $O(b)$. (כיצד?)
2. נשריש בו את H_a כתת-עץ שמאלי, ואת H_b כתת-עץ ימני.
3. "נערמם" את הערמה שקיבלנו החל מהשורש (כלומר, נקרא לשגרה MAX-HEAPIFY עם האינדקס 1).

נשים לב שאחרי שלב 2 מתקבל עץ בינרי כמעט שלם, שכן H_a, H_b היו עצים שלמים.
כמו-כן, H_a, H_b היו ערמות חוקיות, ולכן לאחר הערמום נקבל ערמה חוקית.

סיבוכיות שלב 1 היא $O(\lg n)$, סיבוכיות שלב 2 היא $O(1)$ וסיבוכיות שלב 3 היא $O(\lg n)$.
בסה"כ סיבוכיות האלגוריתם היא $O(\lg n)$.

סעיף ב:

נניח ללא הגבלת הכלליות כי $a = b + 1$.
1. הפעם, ניקח את העלה הכי ימני ב- H_a ונוציא אותו מהערמה.
2. נשריש בו את H_a כתת-עץ שמאלי, ואת H_b כתת-עץ ימני.
3. נערמם את הערמה שקיבלנו מהשורש.

שוב, נשים לב שאחרי שלב 2 מתקבל עץ בינרי כמעט שלם. זה נובע מכך שהגובה של H_a גדול ב-1 מהגובה של H_b. לכן, אחרי שמסירים את העלה הכי ימני מ- H_a ו"מחברים" את שתי הערמות באופן שתואר לעיל, מתקבל עץ כמעט שלם שגובהו גדול ב-1 מהגובה של H_a. לאחר שנבצע ערמום, נקבל ערמה חוקית שמהווה מיזוג של שתי הערמות המקוריות.

ניתוח זמן הריצה:
שלב 1 – $O(\lg n)$.
שלב 2 – $O(1)$.
שלב 3 – $O(\lg n)$.

לכן בסה"כ נקבל שזמן הריצה הוא $O(\lg n)$.

סעיף ג:

שוב, נניח ללא הגבלת הכלליות כי $a \geq b$.

אם $a = b$, נשתמש בסעיף א.

אם $a = b+1$, נשתמש בסעיף ב.

אחרת נפעל באופן הבא:

מהשורש של H_a נפנה שמאלה $a-b$ פעמים.

מגיעים לשורש של ערמה בגובה b. נסמן אותה ב- A_1. נמזג את A_1 עם H_b באמצעות האלגוריתם מסעיף א'. הערמה הממוזגת (להלן, M) מחליפה את A_1 ב- H_a.

נסמן ב-x את האיבר הנמצא בשורש של M. נשים לב, ש-H_a הוא עץ כמעט שלם בגובה $a+1$, אולם ייתכן ש-x הגיע מ-H_b ולכן ייתכן שהאבות הקדמונים של x ב-H_a קטנים ממנו (כמובן, ייתכן גם שהם קטנים מאיברים נוספים שהגיעו מ-H_b).

כדי להפוך את H_a לערמה חוקית, נעלה עם x במעלה הערמה, עד שנגיע לאיבר שאינו קטן מ-x.

בכל שלב בלולאה נחליף את x עם אביו, ולאחר מכן נמצא מקום מתאים בערמה עבור האב.

מציאת המקום המתאים עבור האב תתבצע באופן הבא: נשווה בין האב לבין בנו השמאלי, ונחליף ביניהם אם הבן השמאלי גדול יותר. כאשר נגיע עם האב לשורש של M נקרא ל- MAX-HEAPIFY.

סיבוכיות הזמן: ל-x יש $a-b$ אבות הקדמונים בערמה ולכן נצטרך לחזור על התהליך $a-b$ פעמים לכל היותר. הזמן הדרוש למציאת מקום בערמה עבור כל אב קדמון של x הוא $O(\lg n)$, כגובה הערימה.

לפיכך, סיבוכיות הזמן של האלגוריתם היא: $(a-b)\cdot O(\lg n) = O(\lg^2 n)$

(כדי לראות את הדברים בצורה מוחשית, כדאי לצייר דוגמה. למשל, עם $a=5$ ו- $b=3$.)

מבנה נתונים ומבוא לאלגוריתמים

שיעור הכנה למבחן סמסטר 2004א

פתרון מבחן מסמסטר 2003 א' מועד 95

שאלה 1

מצאו חסמים אסימפטוטיים הדוקים עבור הפונקציות $T(n)$ הנתונות ע"י נוסחאות הנסיגה הבאות:

א. $$T(n)=\begin{cases}1 & n=1\\ 3T\left(n/2\right)+n & n>1\end{cases}$$

ב. $$T(n)=\begin{cases}1 & n=2\\ 2T\left(\sqrt{n}\right)+\lg n & n>2\end{cases}$$

פתרון

א. $$T(n)=3T\left(n/2\right)+n$$

$a=3,\quad b=2,\quad f(n)=n,\quad n^{\lg 3}\approx n^{1.58}$

נבחר $\varepsilon=\frac{\lg 3-1}{2}$. שיטת האב מקרה 1 $$T(n)=\Theta\left(n^{\lg 3}\right)$$

ב. $$T(n)=2T\left(\sqrt{n}\right)+\lg n$$

נסמן $m=\lg n$ ונקבל $$T\left(2^m\right)=2T\left(2^{\frac{m}{2}}\right)+m$$

נסמן $S(m)=T\left(2^m\right)$ ונקבל $$S(m)=2S\left(\frac{m}{2}\right)+m$$

$a=2,\quad b=2,\quad f(m)=m,\quad m^{\lg 2}=m^1=m$

שיטת האב מקרה 2 $$S(m)=\Theta(m\lg m)$$

נחזור ל- $T(n)$: $$S(m)=\Theta(m\lg m)\Rightarrow T\left(2^m\right)=\Theta(m\lg m)\Rightarrow T(n)=\Theta(\lg n\lg\lg n)$$

שאלה 2

תור קדימויות מוגדר כמבנה נתונים S התומך בשתי הפעולות הבאות:

INSERT(S, x): הכנסת האיבר x למבנה S

EXTRACT-MIN(S): מציאת המפתח המינימלי במבנה S, הוצאתו מ- S והחזרתו.

א. איך ניתן לממש תור רגיל (מדיניות *FIFO*) על ידי תור קדימויות?

מהם זמני הריצה של שתי הפעולות אם תור הקדימויות ממומש על ידי ערימה?

ב. איך ניתן לממש מחסנית (מדיניות *LIFO*) על ידי תור קדימויות?

מהם זמני הריצה של שתי הפעולות אם תור הקדימויות ממומש על ידי ערימה?

פתרון

א. לכל איבר נצמיד מפתח שיהיה מונה עולה. הכנסה תהיה הכנסה רגילה עם מונה מתאים. מחיקה תתבצע ע"י *EXTRACT-MIN* שהוא המפתח המינימלי – האיבר הראשון שהוכנס – ועל כן תשמר מדיניות *FIFO* כנדרש. זמני ריצה – נממש באמצעות ערימת מינימום ולכן זמני הריצה של הפעולות הכנסה ומחיקה: $\Theta(\lg n)$.

ב. בדומה לסעיף א' רק שהמפתח יהיה מונה יורד, ואז האיבר המינימלי יהיה האיבר שהוכנס אחרון. זמני הריצה נשמרים.

שאלה 3

נתבונן בשגרת החלוקה *PARTITION(A, p, r)* כפי שהיא מתוארת בספר הלימוד.

א. נניח שבמקום לבחור את האיבר הראשון *A[p]* כאיבר ציר, החלטנו לבחור איבר כלשהו *A[q]*, כאשר מתקיים $p < q < r$. האם השגרה עדיין תפעל נכון? נמקו.

ב. הראו (בעזרת דוגמה נגדית) שאם בוחרים שאיבר הציר את האיבר האחרון *A[r]*, אזי השגרה לא תפעל נכון.

ג. כיצד ניתן לשנות את שגרת החלוקה, כך שהיא כן תפעל נכון כאשר בוחרים את *A[r]* כאיבר ציר? כתבו את השגרה החדשה בפסידוקוד.

פתרון

א. אם נבחר איבר פנימי כלשהו כאיבר הציר, השגרה עדיין תפעל נכון. במקרה זה, בדומה לשגרה *RANDOMIZED-PARTITION*, המצביעים i ו- j לא ייעצרו מיד על האיבר הראשון או האחרון, כי תמיד איבר הציר מקיים שהוא גדול או שווה ל- i וגם קטן או שווה ל- j. אז לא ייווצר מצב בו נחזיר מערך אחד מלא ושני ריק.

ב. נבחר במערך *<1, 2, 3, 4>*. איבר היצר שייבחר הוא $A[r] = 4$. לפי האלגוריתם i ירוץ עד שייעצר על *A[r]*. מצד שני j ירוץ אף הוא וייעצר מיד ב- *A[r]*. לפיכך תחזיר השגרה את הערך $p = r$, מה שגורם לכך שחלוקת המערך היא מערך אחד בגודל r ומערך שני ריק – ולפיכך נכנס לרקורסיה אינסופית.

ג. כדי לשנות את שגרת החלוקה, כך שהיא כן תפעל נכון כאשר בוחרים את *A[r]* כאיבר ציר, כל שעלינו לעשות הוא ראשית להחליף את השורה הראשונה בשגרה ל- $x \leftarrow A[r]$ (כי החלטנו שאיבר הציר עכשיו יהיה *A[r]* במקום *A[p]*) ואז נוסיף את השורה $A[p] \leftrightarrow A[r]$, כלומר החלפת ערך המשתנים. בצורה זו אמנם כביכול בחרנו כאיבר ציר את האיבר האחרון, אך החלפת המשתנים תחזיר אותנו לבחירת האיבר הראשון כאיבר ציר כפי שהשגרה פועלת.

שאלה 4

נתונים טבלת גיבוב *T[1..m]*, סדרה של *n* מפתחות מספריים $< k_1, k_2, ..., k_n >$, ופונקצית גיבוב כלשהי *h*.

נתבונן בתהליך הבא:

1. מכניסים את *n* המפתחות לטבלת הגיבוב על ידי שימוש בפונקציה *h* (עם שרשור במקרי התנגשויות)
2. ממיינים כל אחת מהרשימות המקושרות (באמצעות מיון-מיזוג או מיון-ערימה)
3. משרשרים את הרשימות המקושרות (לפי סדר התאים בטבלה *T*).

א. מהי תוחלת זמן הריצה של תהליך זה?

ב. מהו זמן הריצה של התהליך במקרה הגרוע?

ג. האם בכל מקרה מתקבל מיון של סדרת המפתחות $< k_1, k_2, ..., k_n >$? אם לא, באיזה מקרה מתקבלת סדרה ממוינת?

נמקו כל תשובה.

פתרון

א. הכנסת *n* המפתחות לטבלת הגיבוב – $\Theta(1)$ לכל הכנסה, יש *n* מפתחות. לכן זמן הריצה הכולל הוא $\Theta(n)$ במקרה הממוצע.

מיון כל רשימה מקושרת על ידי מיון-מיזוג – בכל תא יש בממוצע $\Theta\left(n/m\right)$ איברים ואז מיון כל רשימה לוקחת $\Theta\left(\left(n/m\right)\lg\left(n/m\right)\right)$. יש *m* רשימות (*T[1..m]*) ולכן זמן המיון הממוצע הכולל הוא: $\Theta\left(m\left(n/m\right)\lg\left(n/m\right)\right) = \Theta\left(n\lg\left(n/m\right)\right)$.

שרשור הרשימות המקושרות – יש *m* תאים בטבלה. עלינו לעבור על כל אחד מהתאים, ולשרשר את כל *n* האיברים - סה"כ זמן ריצה במקרה הממוצע הוא *O(n + m)*.
סה"כ תוחלת זמן הריצה של האלגוריתם:

$\Theta(n) + \Theta\left(n\lg\left(n/m\right)\right) + O(n+m) = \Theta\left(n\lg\left(n/m\right) + m\right)$ (לא ידוע אם *n > m*).

ב. במקרה הגרוע כל המפתחות מגובבים לאותו ערך ואז:

הכנסת *n* המפתחות לטבלת הגיבוב לאותו ערך – $\Theta(n)$ במקרה הגרוע.

מיון רשימה מקושרת אחת בת *n* איברים על ידי מיון-מיזוג – *O(nlgn)* במקרה הגרוע.

שרשור הרשימות המקושרות – *O(n)* במקרה הגרוע.

סה"כ תוחלת זמן הריצה של האלגוריתם: *2O(n) + O(nlgn) = O(nlgn)*.

ג. ככ

שאלה 5

נתונים קבוצה *S* של *n* מספרים שונים זה מזה ומספר טבעי *m*, $m \le n$.

כתבו אלגוריתם שזמן ריצתו *O(n)*, המוצא את *m* המספרים ב- *s* בהם הקרובים ביותר לחציון (התחתון) של *S*.

הערה: הקרבה נמדדת על פי הערך המוחלט של ההפרש.

מבני נתונים ומבוא לאלגוריתמים (20407)

פתרון מועד 89 - סמסטר 2004א

שאלה 1

א.

$$T(n)=9n\left(n/3\right)+n^2\lg n$$

$n^{\log_3 9}=n^2,\quad f(n)=n^2\lg n,\quad b=3,\quad a=9$

שיטת האב מקרה מורחב $T(n)=\Theta\left(n^2\lg^2 n\right)$

ב.

$$T(n)=\frac{9}{\sqrt{n}}T\left(\sqrt{n}\right)+\frac{1}{n}\lg^2 n\lg\lg n$$

נסמן $T(n)=n^kU(n)$ $\quad n^kU(n)=\frac{9}{\sqrt{n}}\left(\sqrt{n}\right)^kU\left(\sqrt{n}\right)+\frac{1}{n}\lg^2 n\lg\lg n$

נבדוק את ערכו של k: $\quad n^k=9n^{-1/2}n^{k/2}\Rightarrow k=-1/2+k/2\Rightarrow 2k=k-1\Rightarrow k=-1$

נחזור לנוסחה המקורית $\quad T(n)=\frac{9}{\sqrt{n}}T\left(\sqrt{n}\right)+\frac{1}{n}\lg^2 n\lg\lg n$

נסמן $T(n)=\frac{U(n)}{n}$ $\quad \frac{U(n)}{n}=\frac{9}{\sqrt{n}}\cdot\frac{U\left(\sqrt{n}\right)}{\sqrt{n}}+\frac{1}{n}\lg^2 n\lg\lg n$

נכפיל את שני צידי המשוואה ב- n: $\quad U(n)=9U\left(\sqrt{n}\right)+\lg^2 n\lg\lg n$

נסמן $m=lgn$: $\quad U\left(2^m\right)=9U\left(2^{m/2}\right)+m^2\lg m$

נסמן $S(m)=U\left(2^m\right)$: $\quad S(m)=9S\left(m/2\right)+m^2\lg m$

$m^{\lg 9}\approx m^{3.169},\quad f(m)=m^2\lg m,\quad b=2,\quad a=9$

נבחר $\varepsilon=\frac{\lg 9-2}{2}$, שיטת האב מקרה 1 $\quad S(m)=\Theta\left(m^{\lg 9}\right)$

נחזור ל- $T(n)$ ע"י הצבות הפוכות:

$$S(m)=\Theta\left(m^{\lg 9}\right)\Rightarrow U\left(2^m\right)=\Theta\left(m^{\lg 9}\right)\Rightarrow U(n)=\Theta\left(\lg^{\lg 9} n\right)\Rightarrow\frac{U(n)}{n}=\Theta\left(\frac{\lg^{\lg 9} n}{n}\right)\Rightarrow$$

$$\Rightarrow T(n)=\Theta\left(\frac{\lg^{\lg 9} n}{n}\right)$$

שאלה 1

נתון מערך P באורך $m+n$ של מספרים שלמים. ידוע ש-m המספרים הראשונים שייכים לתחום $\left[1..n^2\right]$ ו-n המספרים האחרונים שייכים לתחום $\left[1..m^3\right]$.

תארו אלגוריתם למיון המערך P בזמן $O(m+n)$.

אין צורך לכתוב פסֵידוקוד.

שאלה 2

נתון אלגוריתם מיון M הפועל באופן הבא:

בהינתן מערך A באורך n

- מוצאים את החציון שלו (בעזרת האלגוריתם האופטימלי);
- מבצעים חלוקה סביב החציון;
- ממיינים את החלק השמאלי בעזרת מיון-ערמה;
- מפעילים את האלגוריתם M על החלק הימני של החלוקה באופן רקורסיבי.

(5 נק') א. הסבירו מדוע האלגוריתם M ממיין נכון את המערך.

(15 נק') ב. כתבו נוסחת נסיגה עבור האלגוריתם M; פתרו את נוסחת הנסיגה.

(5 נק') ג. השוו את האלגוריתם M לאלגוריתמי המיון הידועים (מיון-מיזוג, מיון-ערמה, מיון-מהיר) מבחינת הביצועים במקרה הגרוע ובמקרה הממוצע.

שאלה 3

נתון מערך A המכיל N מספרים. ידוע שבין N המספרים קיימים רק k ערכים שונים זה מזה $(0<k<N)$; כלומר, חלק מהערכים מופיעים ב-A יותר מפעם אחת. נסמן ב- $m(a)$ את השכיחות של a ב-A; כלומר, $m(a)$ הוא מספר הפעמים שהערך a מופיע במערך A.

(12 נק') א. כתבו שגרה למציאת איבר a במערך כך שהערך $m(a)$ יהיה מכסימלי; זמן הריצה הנדרש הוא $\Theta(N\cdot\lg k)$.

(13 נק') ב. נתון בנוסף ערך מספרי z.

כתבו שגרה למציאת שני איברים a ו-b במערך כך שיתקיים התנאי $m(a)\cdot a+m(b)\cdot b=z$; זמן הריצה הנדרש הוא $\Theta(N\cdot\lg k)$.

אין צורך לכתוב פסֵידוקוד.

שאלה 4

נתון אלגוריתם M הפועל באופן הבא:

בהינתן מערך A של n מספרים

– מוצאים את החציון של A (בעזרת האלגוריתם האופטימלי);

– מבצעים את חלוקת המערך סביב החציון;

– בונים ערמת מינימום מתוך $\left\lceil \frac{n}{2} \right\rceil$ האיברים הגדולים;

– מפעילים את האלגוריתם M באופן רקורסיבי על $\left\lfloor \frac{n}{2} \right\rfloor$ האיברים הקטנים.

נקרא למבנה המתקבל S.

(5 נק׳) א. כתבו את האלגוריתם M בפסֵידוקוד.

(5 נק׳) ב. הוכיחו שזמן הריצה שלו לינארי.

(15 נק׳) ג. הראו שעל מבנה הנתונים S ניתן לבצע את הפעולות הבאות:

FIND-OS (S, i): מציאת ערך המיקום ה- $\left(\left\lfloor \frac{n}{2^i} \right\rfloor + 1\right)$ של S $\left(1 \le i \le \lfloor \lg n \rfloor\right)$;

זמן הריצה $O(1)$;

DEL-OS (S,i): מחיקת ערך המיקום ה- $\left(\left\lfloor \frac{n}{2^i} \right\rfloor + 1\right)$ של S $\left(1 \le i \le \lfloor \lg n \rfloor\right)$;

זמן הריצה $O(\lg^2 n)$;

INSERT (S, z): הכנסת מפתח חדש z למבנה S; זמן הריצה $O(\lg^2 n)$.

אין צורך לכתוב פסֵידוקוד.

המשך הבחינה בעמוד הבא

שאלה 5

(12 נק׳) א. הציעו מבנה נתונים S שבאמצעותו ניתן לבצע את הפעולות הבאות בזמנים הנדרשים:

INSERT (S, z) : הכנסת המפתח החדש z למבנה S; זמן ריצה: $O(n)$;

DELETE (S, p) : מחיקת האיבר שאליו מצביע p מהמבנה S; זמן ריצה: $O(n)$;

MIN-GAP (S) : החזרת זוג המפתחות x ו-y עבורם $| x - y |$ מינימלי; זמן ריצה $O(1)$;

n מציין את מספר האיברים במבנה S.

(13 נק׳) ב. הציעו מבנה נתונים S שבאמצעותו ניתן לבצע את הפעולות הבאות בזמנים הנדרשים:

BUILD (L, S) : בניית המבנה S מתוך רשימת מפתחות נתונה L באורך n; זמן ריצה: $O(n)$;

INSERT (S, z) : הכנסת המפתח החדש z למבנה S; זמן ריצה: $O(\lg n)$;

DEL-MED (S) : מחיקת החציון מהמבנה S, אם n אי-זוגי, או מחיקת שני החציונים מהמבנה S, אם n זוגי; זמן ריצה: $O(\lg n)$;

DEL-MIN2 (S) : מחיקת ערך המיקום השני מהמבנה S; זמן ריצה: $O(\lg n)$.

אין צורך לכתוב פסֵידוקוד.

בהצלחה !

2005 ב'

1 $\textsc{Radix-Sort} + \textsc{Counting-Sort}$ (בדומה לשאלה ז-18, עמ' 126 במדריך למידה).

2 א מחלקים את המערך לשני חלקים ככה שכל איבר בחלק הראשון קטן שווה לכל איבר בחלק השני. ממיינים את החלק הראשון עם מיון ערמה ואת החלק השני רקורסיבית ויוצא ששני החלקים ממויינים.

2 ב נוסחת הנסיגה היא

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n) + \Theta\left(\frac{n}{2}\lg\frac{n}{2}\right) = T\left(\frac{n}{2}\right) + \Theta(n\lg n)$$

אפשר לפתור אותה עם שיטת האב מקרה 3 כאשר $c = \frac{1}{2}$. מקבלים $T(n) = \Theta(n\lg n)$.

2 ג במקרה הגרוע מיון מהיר הוא היחיד שרץ ב-$\Theta(n^2)$. במקרה הממוצע כולם רצים ב-$\Theta(n\lg n)$.

3 א בונים עץ אדום שחור מכל הערכים **השונים** ב-A. בגלל שיש k ערכים שונים, יהיו בעץ k צמתים ולכן הגובה שלו הוא $O(\lg k)$. N הכנסות לעץ כזה לוקחות $O(N\lg k)$. לכל צומת x בעץ נשמור שדה $size$ שמכיל את מספר הפעמים שהמפתח של x מופיע בקלט - $m(key[x])$. שומרים בצד זוג מספרים: המספר שהופיע הכי הרבה עד עכשיו וכמות הפעמים שהופיע. בכל הכנסה בהתאם לשדה ה-$size$ של הצומת המוכנס מעדכנים את זוג השדות.

3 ב משתמשים בעץ מסעיף א' כדי לבנות עץ אדום-שחור נוסף שמכיל את המכפלה $key[x] \cdot size[x]$ לכל צומת x בעץ המקורי. בכל צומת שומרים רשימה מקושרת $keys$ של המפתחות.

סורקים את העץ שבנינו ולכל צומת x מחפשים את $y = z - key[x]$. אם $key[y] \neq key[x]$ אז מחזירים את האיברים הראשונים ב-$keys[x], keys[y]$. אם $key[y] = key[x]$ אז צריך לוודא שיש יותר מאיבר אחד ב-$keys[x]$, אחרת האיבר שמצאנו הוא אותו אחד.

זמן ריצה: $O(N\lg k) + O(k\lg k) + O(k) = O(N\lg k)$.

4 ב גובה עץ הרקורסיה הוא $\lg n$ כאשר עלות הרמה ה-k היא $O(\frac{n}{2^k})$:

$$\sum_{k=0}^{\lg n} \frac{n}{2^k} = O(n)$$

4 ג אחרי החלוקה המערך מחולק לשני חצאים ככה שכל איבר בחצי הראשון קטן מכל איבר בחצי השני. בחצי השני בונים ערמת מינימום ולכן האיבר הראשון שם הוא המינימום בחצי השני והוא גדול מכל האיברים בחצי הראשון, כלומר הוא ערך המיקום ה-$\lfloor \frac{n}{2^1} \rfloor + 1$...

$\textsc{Find-Os}$ - מחזירים את $A[\lfloor \frac{n}{2^i} \rfloor + 1]$.

$\textsc{Del-Os}$ - מוציאים את השורש של הערמה שמתחילה ב-$\lfloor \frac{n}{2^i} \rfloor + 1$.
לכל $k = i+1, \ldots, \lfloor \lg n \rfloor$ מוציאים את שורש הערמה שמתחילה ב-$\lfloor \frac{n}{2^k} \rfloor + 1$ ומכניסים אותו לערמה שמתחילה ב-$\lfloor \frac{n}{2^{k-1}} \rfloor + 1$ - סוג של "בעבוע". כל הפעולות על הערמות לוקחות $O(\lg n)$ ויש $\lg n$ ערמות, סה"כ מקבלים $O(\lg^2 n)$.

5 א מתחזקים מערך ממויין (או רשימה דו מקושרת) עם זוג אינדקסים בשביל $\textsc{Min-Gap}$. אחרי שמכניסים איבר במקום המתאים בודקים אם המרחק מהמפתח המוכנס לזה שמשמאלו או ימינו קטן מה-$\textsc{Min-Gap}$ הנוכחי, אם כן מעדכנים בהתאם. אם האינדקס של האיבר שמוחקים הוא אחד מאלה שיוצרים את ה-$\textsc{Min-Gap}$ אז רצים על כל הזוגות הסמוכים במערך ומוצאים את ה-$\textsc{Min-Gap}$ החדש (זה לינארי).

5 ב המבנה מורכב משלוש ערמות: את $\lceil \frac{n}{2} \rceil$ האיברים **הקטנים** מחזיקים פעמיים בערמות מינימום ומקסימום. כל זוג איברים זהים בכל ערמה מצביעים אחד לשני. כל פעם שמבצעים פעולה כלשהי (הזזה, מחיקה) על איבר בערמה אחת, מתבצע עדכון למצביע (או מחיקה) בערמה השניה. את שאר $\lfloor \frac{n}{2} \rfloor$ האיברים **הגדולים** מחזיקים בערמת **מינימום**. בשורשי הערמות יושבים החציונים.

Build - מוצאים את החציון עם Select ב-$O(n)$ ובונים את שתי הערמות.

Insert - משווים את המפתח שרוצים להכניס לשורש ערמת המינימום של חצי האיברים הגדולים ולשורש ערמת המקסימום של חצי האיברים הקטנים כדי לדעת לאיזו צריך להכניס. אחרי הכנסה צריך שהערמות ישמרו על גודלם היחסי, לכן במקרה הצורך מוציאים איברים מראש ערמה אחת ומכניסים לשניה. כל הפעולות לוקחות $O(\lg n)$.

Del-Med - החציונים נמצאים בשורשים של ערמת המינימום והמקסימום. מוחקים את האיברים המתאימים ומאזנים את הערמות שיהיו בגדלים המתאימים אחרי המחיקה.

Del-Min2 - ערך המיקום השני נמצא ברמה השנייה בערמת המינימום של חצי האיברים הקטנים. מוחקים אותו וגם את האיבר שאליו הוא מצביע בערמת המקסימום ומאזנים את הערמות במידת הצורך.

שאלה 1

הציעו מבנה נתונים S שבאמצעותו ניתן לבצע את הפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים של S):

$\mathrm{INSERT}(S, k)$: הכנסת איבר חדש בעל המפתח k למבנה S; זמן הריצה : $O(\lg n)$;

$\mathrm{DELETE}(S, x)$: מחיקת האיבר שאליו מצביע x מהמבנה S; זמן הריצה : $O(\lg n)$;

$\mathrm{PAIR\text{-}SUM}(S, z)$: מציאת שני איברים ב-S כך שסכום המפתחות שלהם הינו z;
זמן הריצה : $O(n)$;

$\mathrm{SUM}(S, k)$: החזרת סכום כל המפתחות ב-S שערכם לא עולה על k; זמן הריצה : $O(\lg n)$;

$\mathrm{MAX2}(S)$: החזרת המפתח השני בגודלו במבנה S; זמן הריצה : $O(1)$.

שאלה 2

נתון מערך A באורך n של מספרים ממשיים **חיוביים**, שונים זה מזה.

ברצוננו למצוא שני אינדקסים $1 \le i, j \le n$, כך שמתקיים התנאי $(A[i])^2 = A[j]+1$.

(13 נק׳) א. כתבו שגרה למציאת שני האינדקסים, שזמן ריצתה $O(n \lg n)$ במקרה הגרוע.

(12 נק׳) ב. כתבו שגרה למציאת שני האינדקסים, ש**תוחלת** זמן ריצתה $O(n)$.

שאלה 3

(12 נק׳) א. נתונה סדרה של n מספרים. כתבו אלגוריתם שזמן ריצתו לינארי, המוצא וממיין את p האיברים הקטנים ביותר של הסדרה. ידוע לנו כי $p \le n/\lg n$.

(13 נק׳) ב. נתונה סדרה של n מספרים שלמים בתחום $[n..n^2+n-1]$. כתבו אלגוריתם שזמן ריצתו לינארי, הממיין את סדרת המספרים.

שאלה 4

נתון מספר שלם חיובי **קבוע** c.

נבנה גרסה של האלגוריתם מיון-מיזוג הפועלת באופן הבא :

(1) המערך מחולק ל-c חלקים באורך $\lfloor n/c \rfloor$ או $\lceil n/c \rceil$ כל אחד ; על כל חלק מופעלת גרסה זו של מיון-מיזוג באופן רקורסיבי ;

(2) c החלקים ממוזגים כדי לקבל מערך ממוין.

(5 נקי) א. הראו כיצד ניתן לבצע את מיזוג c החלקים בזמן לינארי.

(10 נקי) ב. כתבו את נוסחת הנסיגה עבור המקרה הגרוע של האלגוריתם (הגרסה החדשה של מיון-מיזוג).

(10 נקי) ג. פתרו את נוסחת הנסיגה והשוו בין זמני הריצה האסימפטוטיים של שתי הגרסאות של מיון-מיזוג (הגרסה מספר הלימוד והגרסה מהשאלה הזאת).

שאלה 5

הציעו מבנה נתונים S שבאמצעותו ניתן לממש את כל אחת מהפעולות הבאות בסיבוכיות המבוקשת :

INSERT (k, R, S) : הכנסת רשומה R בעלת המפתח k למבנה S ; זמן הריצה : $O(\lg n)$;

DELETE (k, S) : מחיקת רשומה כלשהי בעלת המפתח k מהמבנה S ; זמן הריצה : $O(\lg n)$;

FIND (k, S) : מציאת רשומה כלשהי בעלת המפתח k במבנה S ; זמן הריצה : $O(\lg n)$;

MODE (k, S) : החזרת ערך המפתח בעל השכיחות הגבוהה ביותר ; זמן הריצה : $O(1)$.

הערה: n הוא מספר המפתחות השונים ב-S (מספר הרשומות יכול להיות הרבה יותר גדול מ-n).

ב ה צ ל ח ה !

2006 ב'

1 עץ אדום שחור עם שדה sum בכל צומת שמכיל את סכום כל המפתחות בתת העץ המושרש בצומת זה. תיחזוק השדה הוא סטנדרטי. בנוסף שומרים שני שדות בצד $max, max2$ שיחזיקו את המפתח המקסימלי וקודמו (בשביל Max2).

Insert - בודקים אם הצומת גדול מהמקסימום, אם כן max הוא $max2$ החדש ומשימים את הערך שמוכנס ל-max. אם הערך המוכנס בין max ל-$max2$ הטיפול דומה.

Delete - אם מוחקים את המקסימום אז $max2$ הופך להיות max ו-$max2$ החדש הוא ה-Predecessor של $max2$ הישן. סה"כ כל הפעולות פה לוקחות $O(\lg n)$.

Pair-Sum - מנהלים שתי סריקות תוכיות במקביל: אחת רגילה ואחת הפוכה (הולכים ימינה לפני שהולכים שמאלה). הראשונה תתן לנו את איברי העץ בסדר עולה והשנייה בסדר יורד. בכל שלב סוכמים את שני הצמתים הנוכחיים. אם הסכום שווה ל-z סיימנו. אם הוא קטן מקדמים את הסריקה הראשונה, אחרת את השנייה. מפסיקים אם הסריקות מצביעות לאותו צומת או אם המפתח של הראשונה גדול מהמפתח של השניה. האלגוריתם הזה רץ ב-$O(n)$ ודורש שתי מחסניות בגודל $O(\lg n)$.

פתרון אחר יותר פשוט שדורש $O(n)$ מקום זה פשוט לסרוק תוכית את העץ ולשים אותו במערך ולרוץ עם שני מצביעים מההתחלה והסוף...

Sum - שומרים מונה בצד שיכיל את הסכום המבוקש. עבור כל צומת, אם המפתח קטן מ-k אז הצומת הנוכחי וכל תת העץ השמאלי שלו צריכים להיסכם, אז מוסיפים למונה את המפתח הנוכחי ואת שדה ה-sum של הבן השמאלי. ממשיכים רקורסיבית לבן הימני. אם המפתח של הצומת הנוכחי גדול מ-k, אז הצומת הנוכחי וכל התת עץ הימני לא מעניינים, ממשיכים רקורסיבית שמאלה.

2 א ממיינים את המערך ב-$\Theta(n\lg n)$. לכל איבר x במערך מחפשים בינארית את x^2-1. סה"כ n חיפושים בעלות $\Theta(\lg n) \Leftarrow \Theta(n\lg n)$.

2 ב מכניסים לטבלת גיבוב (שומרים גם את האינדקס המקורי של האיבר) ולכל איבר בטבלה מחפשים אותו דבר כמו בסעיף הקודם. סה"כ n חיפושים בעלות $\Theta(1) \Leftarrow \Theta(n)$.

3 א מוצאים עם Select את האיבר ה-p וקוראים ל-Partition כשהוא ה-$pivot$. אז ממיינים עם מיון מיזוג/ערימה את התת מערך שכל איבריו קטנים מ-p. סה"כ יוצא:

$$O(p\lg p) = O(\frac{n}{\lg n}(\lg n - \lg\lg n)) = O(n(1-\underbrace{\frac{\lg\lg n}{\lg n}}_{\downarrow \atop 0})) = O(n)$$

3 ב Radix-Sort + Counting-Sort (בדומה לשאלה ז-18, עמ' 126 במדריך למידה).

4 א ממזגים את כל c הקטעים בבת אחד כמו במיון מיזוג (אלא שכאן יש לנו c קטעים במקום 2). צריך לעבור על n איברים ובכל שלב לבצע $c = O(1)$ בדיקות (כדי למצוא את המינימום מבין c החלקים), סה"כ יוצא $\Theta(n)$.

4 ב + ג פותרים את הנוסחה $T(n) = c \cdot T(\frac{n}{c}) + \Theta(n)$ עם שיטת האב מקרה 2. מקבלים $T(n) = \Theta(n \lg n)$ ולכן אין הבדל בין שתי הגרסאות.

5 עץ אדום-שחור שבכל צומת בנוסף למפתח שומרים רשימה מקושרת של רשומות (שומרים גם את גודל הרשימה). בנוסף שומרים בכל צומת שדה $majority$ שמכיל את מצביע לצומת עם הרשימה הכי ארוכה בתת העץ שמושרש בצומת זה.

Mode - מחזירים את $key[majority[root[S]]]$.

Insert - מתחילים כרגיל. כשמגיעים לצומת מוסיפים את הרשומה לרשימה המקושרת ומגדילים את הגודל ב-1. מטיילים במעלה העץ ומעדכנים את שדה ה-$majority$ בכל צומת במידת הצורך.

Delete - מתחילים כרגיל. כשמגיעים לצומת, אם הרשימה גדולה מ-1 פשוט מוחקים את ראש הרשימה ומקטינים את שדה הגודל. אחרת מוחקים את הצומת. מטיילים במעלה העץ ומעדכנים את שדה ה-$majority$ שיכיל את הצומת שאורך הרשימה שלה מקסימלי מבין הצומת הנוכחי, השמאלי והימני.

אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.
חובה להוכיח (או להסביר) כל טענה.

שאלה 1

נתון עץ חיפוש בינרי T ; נסמן ב- n את מספר האיברים ב- T.

(10 נקודות)

א' כתבו אלגוריתם למציאת שני צמתים x ו- y ב- T, המקיימים את התנאי $key[x]+key[y]=2\cdot key[root[T]]$. זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n)$.

(15 נקודות)

ב' נניח עכשיו שהעץ T מאוזן.

לכל שני צמתים x ו- y ב- T, נסמן ב- $p(x,y)$ את האב הקדמון המשותף הנמוך ביותר של x ו- y ב- T (כלומר, x נמצא בתת-עץ השמאלי של $p(x,y)$ ו- y נמצא בתת-עץ הימני של $p(x,y)$, או להיפך).

כתבו אלגוריתם למציאת שני צמתים x ו- y ב- T, המקיימים את התנאי $key[x]+key[y]=2\cdot key[p(x,y)]$. זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n\cdot \lg n)$.

רמז: שימו לב שמספר הרמות ב- T הוא $\Theta(\lg n)$.

שאלה 2

פתרו את נוסחת הנסיגה הבאה:

$$\begin{cases} T(1)=\Theta(1) \\ T(n)=4\cdot T\left(\sqrt[8]{n}\right)+\sqrt[3]{\lg n}\cdot\left(\sqrt[3]{\lg n}+\left(\lg\lg n\right)^3\right) \end{cases}$$

שאלה 3

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים במבנה):

$\mathrm{BUILD}(S)$: בניית המבנה S מתוך סדרה של n מפתחות; זמן הריצה: $O(n)$;

$\mathrm{MIN}(S)$: החזרת הערך המינימלי של S ; זמן הריצה: $O(1)$;

$\mathrm{DEL\text{-}MEDIAN}(S)$: מחיקת חציון המפתחות של S ; זמן הריצה: $O(\lg n)$;

$\mathrm{OS\text{-}MED7}(S)$: החזרת ערך המיקום ה-$(n/2+7)$ של S ; זמן הריצה: $O(1)$.

הערה: מבנה הנתונים S יכול להיות מורכב מכמה מבני נתונים יסודיים.

שאלה 4

(15 נקודות)

(1) נתון מערך $A[1..n]$ המקיים את התנאי הבא: אם $A[i] > A[j]$, $1 \le i < j \le n$, אזי $j = i+1$.

מהם זמני הריצה (ההדוקים) של האלגוריתמים מיון-הכנסה, מיון-מיזוג, ומיון-מהיר בהפעלתם על המערך A ? הוכיחו את טענותיכם.

(10 נקודות)

(2) נתון תור קדימויות מינימום (למשל, ערמת מינימום) שבאמצעותו ניתן לבצע את שתי הפעולות INSERT (הכנסת איבר חדש) ו-EXTRACT-MIN (מחיקת המינימום).

הוכיחו שקיימת סדרה של $2n$ פעולות (INSERT ו- EXTRACT-MIN) כך שלפחות אחת מפעולות אלה רצה בזמן $\Omega(\lg n)$.

הערה: אין קשר בין שני הסעיפים.

שאלה 5

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים במבנה):

$\text{SEARCH}(S,k)$: חיפוש במבנה S אחר המפתח k ; זמן הריצה: $O(\lg n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k למבנה S ; זמן הריצה: $O(\lg n)$;

$\text{MAX}(S)$: החזרת המפתח המכסימלי של המבנה S ; זמן הריצה: $O(1)$;

$\text{DELETE-MAX}(S)$: מחיקת האיבר בעל המפתח המכסימלי מהמבנה S ; זמן הריצה: $O(\lg n)$;

$\text{DELETE-OLD}(S,t)$: מחיקת האיבר ה- t הוותיק ביותר מהמבנה S ; זמן הריצה: $O(\lg n)$.

כתבו בפסֵידוקוד את השגרה $\text{DELETE-OLD}(S,t)$.

הערה: מבנה הנתונים S יכול להיות מורכב מכמה מבני נתונים יסודיים.

בהצלחה !

2009 א' מועד 87

1 א מנהלים שתי סריקות תוכיות במקביל: אחת רגילה על תת העץ השמאלי של השורש ואחת הפוכה על תת העץ הימני של השורש (הולכים ימינה לפני שהולכים שמאלה). הראשונה תתן לנו רשימה של מפתחות בסדר עולה והשנייה בסדר יורד. בכל שלב סוכמים את שני הצמתים הנוכחיים. אם הסכום שווה ל-$2 \cdot key[root[T]]$ סיימנו. אם הוא קטן מקדמים את הסריקה הראשונה, אחרת את השנייה. האלגוריתם רץ ב-$O(n)$ ודורש שתי מחסניות בגודל $O(\lg n)$.

1 ב מריצים את האלגוריתם מסעיף א' לפי רמות (קודם הצמתים בעומק 0, אחרי זה 1, 2...). לשורש יש $\Theta(n)$ עבודה, לכל אחד מהבנים $\Theta(\frac{n}{2})$ וכך הלאה...

זמן הריצה הוא: $T(n) = 2 \cdot T(\frac{n}{2}) + \Theta(n) = \Theta(n \lg n)$.

2 מציבים $m = \lg n$ ומשתמשים בשיטת האב מקרה 2: $\Theta(\lg^{2/3} n \cdot \lg\lg n)$.

3 נניח שהכוונה ב-Os-Med7 היא לערך המיקום ה-$\lceil \frac{n}{2} \rceil + 7$. נשים את $\lceil \frac{n}{2} \rceil$ האיברים הקטנים בערמת מקסימום. מתוך $\lfloor \frac{n}{2} \rfloor$ האיברים הגדולים, את 6 הקטנים מהם נשמור במערך ממויין ואת כל השאר בערמת מינימום. תחזוק המערך הזה לוקח $\Theta(1)$ כי גודלו קבוע.

Build - מוצאים את החציון (Select) ומכניסים את הערכים למבנים המתאימים. שומרים במשתנה בצד את המינימום.

Min - מחזירים את המשתנה ששמרנו בצד.

Del-Median - מוציאים את המקסימום מהערמה הראשונה. שומרים על איזון בין כל שלושת המבנים ע"י העברת איברים מאחד לשני במידת הצורך.

Os-Med7 - מחזירים את שורש הערמה השניה.

5 האיברים יוכנסו לעץ אדום שחור. בנוסף, כל איבר שמוכנס יוכנס גם לעץ ערכי מיקום כאשר המפתח בכל צומת יהיה מספר רץ. הצמתים יכילו מצביעים אחד לשני.

לדוגמה נניח שהכנסנו כבר 5 איברים, ועכשיו מכניסים איבר עם מפתח x: מכניסים את x לעץ הראשי. מכניסים לעץ ערכי מיקום את המפתח 6 ומעדכנים את שני הצמתים שהוכנסו שיצביעו אחד לשני.

Insert - כמו שתואר בפסקה למעלה. מקדמים את המספר הרץ.

Max - מתוחזק במשתנה בצד, (מעדכנים אותו בכל הכנסה/מחיקה מהעץ בעלות של $\Theta(\lg n)$).

Delete-Max - מוחקים את Max ואת הצומת שהוא מצביע אליו בעץ ערכי מיקום.

Delete-Old - מחפשים את ערך המיקום ה-t בעץ ערכי מיקום. מוחקים אותו ואת הצומת שאליו הוא מצביע.

אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.

חובה להוכיח (או להסביר) כל טענה.

שאלה 1

נתון מערך של מספרים $A[1..n]$.

כתבו אלגוריתם למציאת שלושה אינדקסים $1 \le i < j < k \le n$ כך שמתקיים $A[i] + A[k] = 2 \cdot A[j]$. זמן הריצה הנדרש הינו $\Theta(n^2)$.

שאלה 2

פתרו את נוסחת הנסיגה הבאה:

$$\begin{cases} T(1) = \Theta(1) \\ T(n) = 8n\sqrt{n} \cdot T\left(\sqrt{n}\right) + n^3 \cdot \lg^3 n \end{cases}$$

רמז: $U(n) = T(n)/n^3$.

שאלה 3

הציעו מבנה נתונים S שמפתחותיו n שלמים חיוביים, התומך בפעולות הבאות בזמנים הנדרשים:

$\text{BUILD}(S)$: בניית המבנה S מתוך סדרה של n שלמים חיוביים; זמן הריצה: $O(n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k למבנה S; זמן הריצה: $O(\lg n)$;

$\text{MEDIAN}(S)$: החזרת חציון המפתחות; זמן הריצה: $O(1)$;

$\text{ODD-MEDIAN}(S)$: החזרת חציון המפתחות האי-זוגיים; זמן הריצה: $O(1)$;

$\text{EVEN-MEDIAN}(S)$: החזרת חציון המפתחות הזוגיים; זמן הריצה: $O(1)$;

$\text{DEL-MEDIAN}(S)$: מחיקת חציון המפתחות מתוך המבנה S; זמן הריצה: $O(\lg n)$;

$\text{DEL-ODD-MEDIAN}(S)$: מחיקת חציון המפתחות האי-זוגיים מתוך המבנה S; זמן הריצה: $O(\lg n)$;

$\text{DEL-EVEN-MEDIAN}(S)$: מחיקת חציון המפתחות הזוגיים מתוך המבנה S; זמן הריצה: $O(\lg n)$;

$\text{INCREASE}(S,p)$: קידום ב-1 של מפתח האיבר שאליו מצביע p; זמן ריצה: $O(\lg n)$.

הערה: מבנה הנתונים S יכול להיות מורכב מכמה מבני נתונים יסודיים.

שאלה 4

נתונים מערך $A[1..n]$ של מספרים ממשיים ומספר טבעי m המקיים $2m < n$.

כתבו אלגוריתם הבודק האם **קיים** ב-A איבר z המקיים את שני התנאים הבאים :

(1) A מכיל לפחות $n - 2m$ איברים קטנים מ- z ;

(2) z מופיע יותר מ- m פעמים ב-A .

זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n)$.

שאלה 5

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים במבנה) :

$\mathrm{BUILD}(S)$: בניית המבנה S מתוך סדרה נתונה של n איברים; זמן הריצה : $O(n \cdot \lg n)$;

$\mathrm{SEARCH}(S,k)$: חיפוש במבנה S אחר המפתח k ; זמן הריצה : $O(\lg n)$;

$\mathrm{INSERT}(S,k)$: הכנסת המפתח k למבנה S ; זמן הריצה : $O(\lg n)$;

$\mathrm{DELETE}(S,p)$: מחיקת האיבר שאליו מצביע p מהמבנה S ; זמן הריצה : $O(\lg n)$;

$\text{DECREASE-UPTO}(S,k,d)$: הקטנה בערך $d > 0$ של כל המפתחות במבנה S שערכיהם קטנים מ- k או שווים ל- k ; זמן הריצה : $O(\lg n)$.

כתבו בפסֵידוקוד את השגרה $\text{DECREASE-UPTO}(S,k,d)$.

בהצלחה !

2009 א' מועד 88

1 מתחילים במיון המערך. לכל $1 < j < n$ רצים עם שני אינדקסים, אחד בראש המערך והשני בסופו ובודקים אם הסכום שלהם שווה ל-$2 \cdot A[j]$. במידה וכן אז סיימנו. אם הוא קטן מקדמים את האינדקס הראשון. אם הוא גדול מזיזים את האחרון אחד אחורה. מקדמים את j כאשר אחד האינדקסים מגיע אליו ומתחילים את התהליך שוב.

המיון הראשוני לוקח $\Theta(n \lg n)$. בכל שלב זוג האינדקסים משמאל ומימין ל-j עוברים על $\Theta(n)$ איברים, ו-j גם כן, לכן זמן הריצה הכולל הוא $\Theta(n^2)$.

2 משתמשים ברמז ומקבלים לאחר הצבה $U(n) = 8 \cdot U(\sqrt{n}) + \lg^3 n$. עושים עוד הצבה $S(m) = U(2^m)$, $m = \lg n$ ומקבלים $S(m) = 8 \cdot S(\frac{m}{2}) + m^3$. את המשוואה האחרונה פותרים עם שיטת האב מקרה 2 ומקבלים $S(m) = \Theta(m^3 \lg m)$. אחרי חזרה לפונקציה המקורית מקבלים $T(n) = \Theta(n^3 \lg^3 n \lg \lg n)$.

3 המבנה מורכב מ-6 ערמות: מקסימום H_L ומינימום H_H ($L = low, H = high$) מכילות בהתאמה את חצי האיברים הקטנים והגדולים במבנה. מקסימום H_{E_L} ומינימום H_{E_H} מכילות את חצי האיברים הזוגיים הקטנים והגדולים ואותו דבר לאי-זוגיים ב-H_{O_L}, H_{O_H}. ארבעת הערמות האחרונות מכילות איברים שכבר הופיעו בשתי הערמות הראשונות, לכן כדי לשמור על סנכרון ביניהם, כל זוג איברים זהים יצביעו אחד לשני וכאשר אחד מהם מוזז/נמחק ההעתק שלו יעודכן בהתאם.

Build - מוציאים את החציונים המתאימים עם Select ומכניסים למבנים המתאימים את האיברים שקטנים/גדולים מהם.

Median - מחזירים את השורש של H_L.

Odd-Median - מחזירים את השורש של H_{O_L}.

Even-Median - מחזירים את השורש של H_{E_L}.

במחיקות למיניהן מוחקים את השורש של הערמה המתאימה (וגם את ההעתק שהוא מצביע אליו) ואז "מאזנים" מחדש כל זוג ערמות שישמרו על היחס ביניהם ע"י הוצאת איבר מערמה אחת והכנסה שלו לערמה השניה.

Increase - מקדמים את המפתח ומתקנים את הערמה (Max-Heapify). מסירים את האיבר מהערמה של הזוגיים או אי-זוגיים ומכניסים לערמה המתאימה בהתאם לזוגיותו ולגודל (משווים לשורש של הערמה).

4 המועמדים האפשריים לתנאי הראשון הם כל האיברים שערך המיקום שלהם גדול מ-$n-2m$. כדי שאיבר כלשהו יהיה מועמד לתנאי השני, הוא צריך להיות ערך המיקום ה-$n-m$ או ה-n. אם כך מוצאים את ערך המיקום ה-$n-m$ עם Select ועושים עוד מעבר על המערך כדי למנות את מספר המופעים שלו. אם הוא מופיע יותר מ-m פעמים אז סיימנו, אחרת עושים אותו דבר לערך המיקום ה-n. סה"כ קוראים פעמיים ל-Select ועושים עוד שני מעברים על המערך לכן זמן הריצה הוא $\Theta(n)$.

5 עץ אדום-שחור כאשר בכל צומת שומרים שדה מספרי $accum$ שיאותחל ל-0 כברירת מחדל. לכל צומת x בעץ מחברים את $accum[x]$ לכל המפתחות בתת העץ שמושרש ב-x.

למימוש חיפוש, הכנסה ומחיקה ראה שאלה 3 ב' בממ"ן 17.

```
procedure DECREASE-UPTO(S, k, d)
    n ← root[S]
    while n ≠ nil do
        if key[n] ≤ k then
            accum[n] ← accum[n] − d
            if right[n] ≠ nil then
                accum[right[n]] ← accum[right[n]] + d
            n ← right[n]
        else
            n ← left[n]
```

אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.
חובה להוכיח (או להסביר) כל טענה.

שאלה 1

נתון מערך $A[1..n]$ של מספרים שלמים המקיימים את התנאי $0 \le A[i] < 65536$, $i = 1,...,n$.
כתבו אלגוריתם למציאת שני אינדקסים $i, j \in \{1,...,n\}$ כך שיתקיים $A[i] + A[j] = 65536$;
זמן הריצה הנדרש הינו $O(n)$.

שאלה 2

נתונים במישור n מלבנים. המלבן ה- i מורכב מכל הנקודות (x, y) המקיימות את התנאים
$0 \le x \le x_i$, $0 \le y \le y_i$, $i = 1,...,n$.
כתבו אלגוריתם לחישוב שטח איחוד כל n המלבנים; זמן הריצה הנדרש הינו $O(n \cdot \lg n)$.

שאלה 3

נתון מערך $A[1..n]$ של מספרים שלמים. כתבו אלגוריתם שזמן ריצתו **לינארי**, המחזיר את מספר האיברים במערך השווים לחציון.

שאלה 4

הציעו מבנה נתונים S, התומך בפעולות הבאות בזמנים הנדרשים:

BUILD(S): בניית המבנה S מתוך סדרה בת n מפתחות; זמן הריצה: $O(n \cdot \lg n)$;

INSERT(S,k): הכנסת המפתח k לתוך המבנה S; זמן הריצה: $O(\lg n)$;

DELETE-OLD(S): מחיקת האיבר הוותיק ביותר מתוך המבנה S; זמן הריצה: $O(\lg n)$;

COUNT-MIN-OLD(S): החזרת מספר המפתחות במבנה הקטנים ממפתח האיבר הוותיק ביותר (או שווים לו); זמן הריצה: $O(\lg n)$.

הערה: מבנה הנתונים S יכול להיות מורכב מכמה מבני נתונים פשוטים יותר; n מציין את מספר האיברים במבנה.

שאלה 5

הציעו מבנה נתונים S, התומך בפעולות הבאות בזמנים הנדרשים:

BUILD(S): בניית המבנה S מתוך סדרה בת n מפתחות; זמן הריצה: $O(n)$;

INSERT(S,k): הכנסת המפתח k לתוך המבנה S; זמן הריצה: $O(\lg n)$;

DELETE-MAX(S): מחיקת האיב ר בעל המפתח המכסימלי מתוך המבנה S; זמן הריצה: $O(\lg n)$;

SWITCH-OLD-NEW(S): החלפת המפתחות בין האיבר הוותיק ביותר לבין האיבר החדש ביותר במבנה S; זמן הריצה: $O(\lg n)$.

הערה: מבנה הנתונים S יכול להיות מורכב מכמה מבני נתונים פשוטים יותר; n מציין את מספר האיברים במבנה.

בהצלחה!

2009 ב' מועד 91

1 ממיינים את המערך עם Counting-Sort ורצים עם שני אינדקסים מההתחלה ומהסוף. אם הסכום שלהם גדול מ-65535 מזיזים את האינדקס השני אחד אחורה. אם הסכום קטן אז הראשון זז אחד קדימה. מסיימים כשהסכום שווה או שהאינדקסים נפגשים/עברו אחד את השני.

2 ממיינים את הזוגות לפי x. אם יש יותר מזוג נקודות אחד עם אותו ערך x, לוקחים את האחד עם ערך ה-y המקסימלי.

שומרים במשתנה $area$ את השטח שנצבר עד כה. האלגוריתם רץ על הזוגות הממויינים תוך שהוא שומר במשתנה $max = (x, y)$ את הנקודה עם ה-y המקסימלי. המלבן ה-i מטופל בצורה הבאה:

- אם $y_i \leq y_{i-1}$ אז $area += (x_i - x_{i-1}) \cdot y_i$.
- אחרת אם $y_i > max_y$ אז $area = x_i \cdot y_i$ (מעדכנים את max).
- אחרת $area = x_i \cdot max_y - (x_i - max_x) \cdot max_y$.

3 מוצאים את החציון עם Select ורצים על המערך וסופרים כמה איברים שווים לו.

4 עץ אדום שחור עם שדה $size$ בכל צומת ומחסנית (שממומשת עם רשימה דו-מקושרת) של מצביעים לצמתים בעץ. כשמכניסים צומת לעץ דוחפים למחסנית מצביע לצומת.

Delete-Old - האיבר הוותיק ביותר יהיה בסוף המחסנית, מוחקים אותו משני מבני הנתונים.

Count-Min-Old - מחפשים בעץ את המפתח של האיבר שבסוף המחסנית k. מתחילים בשורש העץ: לכל צומת x, אם $key[x] > k$ אז כל האיברים שמימין ל-x לא רלוונטים והולכים רקורסיבית שמאלה. אם $key[x] \leq k$ אז כל האיברים שמשמאל ל-x קטנים מ-k. מוסיפים את $size[left[x]] + 1$ למונה ששומרים בצד וממשיכים רקורסיבית ימינה.

5 ערמת מקסימום ומחסנית (שממומשת עם רשימה דו-מקושרת) של מצביעים לערמה. כל איבר בערמה ישמור מצביע לאיבר המתאים לו במחסנית ויעדכן אותו כשהוא מוזז.

Switch-Old-New - האיבר הראשון במחסנית הוא החדש ביותר והאחרון הוא הוותיק ביותר. מעדכנים את המפתח של אחד מהם עם המפתח השני (שומרים את הערך הישן בצד) וקוראים ל-Max-Heapify ואז מטפלים בשני (צריך לעשות את זה אחד אחרי השני כי השגרה מניחה שתת העץ שמושרש בצומת שמתקנים הוא ערמה תקינה).

מימוש שאר השגרות הוא סטנדרטי בתוספת עדכון המצביעים במקומות המתאימים ותחזוק המחסנית.

אפשר להשתמש בכל עובדה או תוצאה הנמצאת בספר הלימוד או במדריך הלמידה, ללא הוכחה או הסבר. חובה להוכיח או להסביר כל טענה אחרת.
אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.

שאלה 1

נתונה ערמת מינימום H בת n איברים. נניח שכל שורה (רמה) בעץ בערמה ממוינת משמאל לימין, בסדר לא יורד.

א' (10 נקודות) כתבו אלגוריתם לא רקורסיבי המבצע חיפוש אחר ערך נתון z בין איברי הערמה; האלגוריתם יבצע את החיפוש באמצעות קריאה לחיפוש בינרי על כל אחת מהשורות. יש לכתוב את האלגוריתם בפסֵידוקוד.

ב' (10 נקודות) נתחו את זמן הריצה של האלגוריתם.

ג' (5 נקודות) הוכיחו את נכונות האלגוריתם.

פתרון:

א' גובה הערמה הוא $\lfloor \lg n \rfloor$; מספר השורות הינו $1+\lfloor \lg n \rfloor$. בכל שורה מבצעים חיפוש בינרי.

```
HEAP-LEVEL-SEARCH(H, n, z)
  l ← 0
  while l < ⌊lg n⌋
     do i ← BINARY-SEARCH(A, 2^l, 2^(l+1) − 1, z)
        if i ≠ NIL
           then return i
           else l ← l + 1
  i ← BINARY-SEARCH(A, 2^l, n, z)
  if i ≠ NIL
     then return i
  else return NIL
```

ב' זמן הריצה הינו $\left(1+\lfloor \lg n \rfloor\right)\cdot O(\lg n) = O(\lg^2 n)$.

ג' שמורת הלולאה היא: לפני האיטרציה l ($0 \le l \le \lfloor \lg n \rfloor$), אם האיבר z נמצא בערמה, אזי הוא נמצא באחת השורות $l,\dots,\lfloor \lg n \rfloor$.

שאלה 2

ענו לשאלות הבאות ונמקו את תשובותיכם :

א' (13 נקודות) מהו זמן הריצה של האלגוריתם מיון-מהיר על המערך $A=[2,3,...,n,1]$?

ב' (12 נקודות) מהו זמן הריצה של האלגוריתם מיון-מהיר על המערך $A=[1,n,...,3,2]$?

הערה: מדובר באלגוריתם מיון-מהיר המוצג בספר הלימוד.

פתרון:

א' בקריאה הראשונה, איבר הציר 1 מתחלף עם האיבר הראשון 2 ; מתקבל התת-מערך $A=[3,...,n,2]$. התהליך חוזר על עצמו $n-1$ פעמים ; לכן, זמן הריצה של האלגוריתם הינו $\Theta(n^2)$.

ב' בקריאה הראשונה, איבר הציר 2 מתחלף עם האיבר השני n ; מתקבל התת-מערך $A=[n-1,...,3,n]$. בקריאה השניה, אין שינויים במערך ; מתקבל התת-מערך $A=[n-1,...,3]$. מערך זה הוא ממוין בסדר יורד ; לכן, זמן הריצה של האלגוריתם הינו $\Theta(n^2)$.

שאלה 3

נתון עץ חיפוש בינרי T ; נסמן ב- n את מספר הצמתים וב- h את גובה העץ. נסיר מהעץ צומת כלשהו y ; המבנה $T-\{y\}$ מתפרק ל- m ($1 \le m \le 3$) עצי חיפוש בינריים.

א' (10 נקודות) תארו אלגוריתם לאיחוד m העצים הנ"ל לעץ חיפוש בינרי אחד U ; זמן הריצה הנדרש : $O(h)$.

ב' (5 נקודות) באלו מקרים הגובה של U קטן מ- h ?

ג' (10 נקודות) מהו הגובה המכסימלי האפשרי של U כפונקציה של h ? באלו מקרים מתקבל גובה זה?

הערה: שימו לב שלצומת y שמוסר מהעץ T יכולים להיות שני בנים ולכן אי אפשר להשתמש באלגוריתם המתואר בספר הלימוד למחיקת צומת מעץ חיפוש בינרי.

פתרון:

א' יהא x הבן השמאלי של y ויהא z הבן הימני של y . אם אביו p של y קיים, מחברים את z כבן של p במקומו של y . אחר-כך, אם z קיים, מחברים את x כבנו השמאלי של הצומת המינימלי בתת-עץ המושרש ב- z . אם z אינו קיים, מחברים את x כבן של p במקומו של y .

כל התהליך מתבצע בזמן $O(h)$.

ב' הגובה של U קטן מ- h במקרים הבאים : כל מסלול באורך h מהשורש לעלה מסתיים בעלה של התת-עץ המושרש ב- z ; או, z לא קיים וכל מסלול כזה מסתיים בעלה של התת-עץ המושרש ב- x .

ג' הגובה המכסימלי האפשרי של U כפונקציה של h הינו $2h-2$. זה קורה כאשר y הוא השורש של העץ, גובהו של כל אחד מהתת-עצים המושרשים ב-x וב-z הוא $h-1$, והצומת המינימלי בתת-עץ המושרש ב-z הוא עלה נמוך ביותר.

שאלה 4

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים במבנה):

$\text{BUILD}(S)$: בניית המבנה S מסדרה של n מפתחות; זמן הריצה: $O(n \cdot \lg n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k לתוך המבנה S; זמן הריצה: $O(\lg n)$;

$\text{DELETE-MAX}(S)$: מחיקת האיבר בעל המפתח המכסימלי מהמבנה S; זמן הריצה: $O(\lg n)$;

$\text{DELETE-OLD}(S,t)$: מחיקת האיבר ה-t הוותיק ביותר מהמבנה S ($1 \le t \le n$); זמן הריצה: $O(\lg n)$;

$\text{ADD-TO-NEW}(S,d)$: הוספת הערך d לאיבר שנכנס האחרון למבנה S; זמן הריצה: $O(\lg n)$.

פתרון:

מבנה הנתונים S מורכב מעץ אדום-שחור T ומעץ ערכי מיקום W. כל אחד מ-n המפתחות מופיע בעץ T; זמן ההכנסה שלו משמש במפתח בעץ W. שני הצמתים המקבילים מחוברים ביניהם באמצעות שני מצביעים.

$\text{BUILD}(S)$: בניית כל אחד משני העצים אורכת זמן $O(n \cdot \lg n)$.

$\text{INSERT}(S,k)$: הכנסת המפתח k לכל אחד משני העצים אורכת זמן $O(\lg n)$; הוספת המצביעים – זמן קבוע.

$\text{DELETE-MAX}(S)$: מחפשים את המפתח המכסימלי ב-T באמצעות $\text{TREE-MAXIMUM}(T)$; מחיקת האיבר בעזרת $\text{RB-DELETE}(T,\bullet)$ ומחיקת מקבילו בעזרת $\text{RB-DELETE}(W,\bullet)$; זמן ריצה $O(\lg n)$.

$\text{DELETE-OLD}(S,t)$: מחפשים את המפתח ה-t הקטן ביותר ב-W באמצעות $\text{OS-SELECT}(W,t)$; מחיקת האיבר בעזרת $\text{RB-DELETE}(W,\bullet)$ ומחיקת מקבילו בעזרת $\text{RB-DELETE}(T,\bullet)$; זמן ריצה $O(\lg n)$.

$\text{ADD-TO-NEW}(S,d)$: מחפשים את המפתח המכסימלי ב- W באמצעות
$\text{TREE-MAXIMUM}(W)$; מחיקת הצומת ומקבילו משני העצים ; הוספת הערך d למפתח ;
הכנסה מחדש של שני הצמתים לשני העצים ; זמן ריצה $O(\lg n)$.

שאלה 5

נתון מספר טבעי $m > 0$ (קבוע).
הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים במבנה) :

$\text{BUILD}(S)$: בניית המבנה S מסדרה **ממוינת** של n מפתחות שלמים ; זמן הריצה : $O(n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k לתוך המבנה S ; זמן הריצה : $O(\lg n)$;

$\text{DELETE-MIN}(S,r)$: מחיקת האיבר בעל המפתח המינימלי k המקיים $k \bmod m = r$ מהמבנה S ($0 \le r < m$) ; זמן הריצה : $O(\lg n)$;

$\text{MODE}(S,r)$: החזרת המפתח k בעל השכיחות המכסימלית, המקיים $k \bmod m = r$ ($0 \le r < m$);זמן הריצה : $O(1)$.

הערות: המפתחות במבנה הם מספרים שלמים. המבנה S יכול להיות מורכב מכמה מבני נתונים פשוטים יותר.

פתרון:

מבנה הנתונים S מורכב ממערך של m מצביעים ל- m עצים אדומים-שחורים $T_0, T_1, ..., T_{m-1}$ וממערך של m מצביעים ל- m ערמות מכסימום $H_0, H_1, ..., H_{m-1}$. כל צומת בעץ T_r מכיל מצביע לצומת מקביל בערמה H_r , וגם בכיוון ההפוך ($0 \le r < m$). המפתח ב- H_r הוא השכיחות של המפתח ב- T_r .

$\text{BUILD}(S)$: מחלקים את סדרת המפתחות ל- m תת-סדרות ממוינות (לפי שאריות המפתחות).
מכל תת-סדרה בונים את העץ T_r ואת הערמה H_r בהתאמה. זמן הריצה הכולל $O(n)$.

$\text{INSERT}(S,k)$: מחשבים $r = k \bmod m$; מכניסים את המפתח ל- T_r ול- H_r ; זמן ריצה $O(\lg n)$.

$\text{DELETE-MIN}(S,r)$: מוחקים את האיבר המינימלי מהעץ T_r ואת המקביל שלו מהערמה H_r ;
זמן ריצה $O(\lg n)$.

$\text{MODE}(S,r)$: מוצאים את שורש הערמה H_r ומחזירים את מפתח המקביל שלו ב- T_r .

אפשר להשתמש בכל עובדה או תוצאה הנמצאת בספר הלימוד או במדריך הלמידה, ללא הוכחה או הסבר. חובה להוכיח או להסביר כל טענה אחרת.

אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.

שאלה 1

נתונות שתי רשימות של מספרים ממשיים, S בת m איברים ו- T בת n איברים; בנוסף, נתון מספר ממשי z.

כתבו אלגוריתם הקובע האם קיימים $x \in S$, $y \in T$, כך שמתקיים $x + y = z$. זמן הריצה הנדרש של האלגוריתם הינו $\Theta((m+n)\cdot \lg(\min(m,n)))$.

פתרון:

נעתיק את שתי הרשימות למערכים (באותם שמות S ו- T).

נניח כי $m < n$. ממיינים את המערך S (מיון מיזוג או מיון ערמה); זמן ריצה $\Theta(m \cdot \lg m)$. לכל איבר $y \in T$ נבצע חיפוש בינרי במערך S אחר הערך $x = z - y$; זמן ריצה $\Theta(n \cdot \lg m)$. סה"כ $\Theta((m+n)\cdot \lg m)$.

אם $m > n$, דומה.

שאלה 2

נתון מערך של מספרים $A[1..n]$. ידוע שקיימים אינדקסים p, q, $1 < p < q < n$, כך שמתקיימים התנאים $A[1] > ... > A[p-1] > A[p] = ... = A[q] < A[q+1] < ... < A[n]$ (כלומר, התת-מערך $A[1..p]$ ממוין בסדר יורד והתת-מערך $A[q..n]$ ממוין בסדר עולה).

כתבו שגרת פסֵידוקוד המבצעת חיפוש אחר האינדקסים p, q והמחזירה אותם; זמן הריצה הנדרש הינו $O(\lg n)$.

פתרון:

נכתוב שתי השגרות, הראשונה למציאת האינדקס p :

```
FIND-FIRST-INDEX(A)
i ← 1
j ← length[A]
while j − i > 1
   do m ← (i + j) / 2
      if A[m − 1] > A[m] and A[m] = A[m + 1]
         then return m
      if A[m − 1] > A[m]
         then i ← m
         else j ← m
```

השגרה השניה, למציאת האינדקס q, דומה.

שאלה 3

א' (12 נקודות) נתון מערך $A[1..n]$ של מספרים שלמים המקיימים את התנאים
$n^2 \le A[i] \le n^3 + n^2 - 1$ לכל $i = 1,...,n$.

כתבו שגרה למיון המערך בזמן לינארי.

ב' (13 נקודות) נתון מערך $Q[1..n]$ של מספרים ממשיים; ידוע שלא קיימים במערך יותר מ-
$\lg^2 n$ ערכים שונים זה מזה.

הראו שניתן למיין את המערך בזמן $O(n \cdot \lg\lg n)$.

פתרון:

א' בונים את המערך $B[1..n]$: $B[i] = A[i] - n^2$. אז מתקיים $0 \le B[i] \le n^3 - 1$ לכל $i = 1,...,n$.
ניתן לייצג כל איבר של B כמספר בן שלוש ספרות בבסיס n. משתמשים במיון בסיס מעל מיון-מניה.

ב' בונים את העץ האדום-שחור בהתחשב בכפילויות של המפתחות. מתקבל עץ בגודל $O(\lg^2 n)$ שגובהו $O(\lg(\lg^2 n)) = O(\lg\lg n)$.

שאלה 4

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים במבנה):

BUILD(S): בניית המבנה S מסדרה של n מפתחות; זמן הריצה: $O(n \cdot \lg n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k לתוך המבנה S ; זמן הריצה : $O(\lg n)$;

$\text{MEDIAN}(S)$: החזרת החציון של המבנה S ; זמן הריצה : $O(1)$;

$\text{MIN-OLDEST}(S,t)$: החזרת המפתח המינימלי בין t המפתחות הותיקים ביותר במבנה; זמן הריצה : $O(\lg n)$.

הערה: המבנה S יכול להיות מורכב מכמה מבני נתונים פשוטים יותר.

פתרון:

מבנה הנתונים S מורכב מערמת מינימום $H_{\min}$, ערמת מכסימום $H_{\max}$, עץ אדום-שחור מורחב T (עץ ערכי מיקום) ומחסנית P. ערמת המינימום $H_{\min}$ מכילה את $\lceil n/2 \rceil$ המפתחות הקטנים היותר של המבנה, ערמת המכסימום $H_{\max}$ מכילה את $\lfloor n/2 \rfloor$ המפתחות הגדולים יותר של המבנה. העץ T והמחסנית P מכילים כל אחד כל האיברים. המפתחות של T הם זמני ההכנסה. כל איבר בעץ T קשור לאיבר המקביל במחסנית P ולאיבר המקביל באחת הערמות באמצעות מצביעים דו-כיווניים. כל איבר z במחסנית מכיל מצביע $\min[z]$ אל האיבר בעל המפתח המינימלי הקודם לו (או לעצמו).

$\text{BUILD}(S)$: מציאת החציון בעזרת האלגוריתם SELECT ; חלוקת המערך בעזרת השגרה PARTITION ; בניית ערמת המינימום מהמפתחות הקטנים ובניית ערמת המכסימום מהמפתחות הגדולים; בניית המחסנית באופן סדרתי, בהוספת השדה $\min[z]$ (אם המפתח הנכנס הוא הקטן ביותר, אז המצביע מופנה לעצמו; אחרת, הוא מופנה אל המינימום עבור האיבר הקודם; עד עכשיו, זמן ריצה לינארי. בונים את העץ T בזמן $O(n \cdot \lg n)$.

$\text{INSERT}(S,k)$: הכנסה לעץ T ולמחסנית P, כרגיל, עם הוספת שדה המינימום; האיבר שנכנס לעץ T מקבל מפתח גדול ב-1 מהקודם. הכנסה לאחת הערמות לפי המקרה (מפתח קטן מהחציון או גדול ממנו); העברת איבר בין שתי הערמות, לפי הצורך. זמן ריצה $O(\lg n)$.

$\text{DEL-MEDIAN}(S)$: מחיקת שורש ערמת המינימום; העברת איבר בין שתי הערמות, לפי הצורך; מחיקת האיברים המקביליים מהעץ ומהמחסנית. זמן ריצה $O(\lg n)$.

$\text{MIN-OLDEST}(S,t)$: מציאת ערך המיקום ה- t בעזרת $\text{OS-SELECT}(T,t)$; מעבר אל המחסנית ובחירת $\min[z]$.

שאלה 5

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים (n מציין את מספר המפתחות השונים ב- S ; N מציין את המספר הכולל של מפתחות ב- S):

$\text{BUILD}(S)$: בניית המבנה S מסדרה של N מפתחות (לא בהכרח שונים זה מזה); זמן הריצה: $O(N \cdot \lg n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k למבנה S (k יכול להיות מפתח חדש או מפתח שכבר קיים ב- S); זמן הריצה: $O(\lg n)$;

$\text{MAX-FREQ}(S)$: החזרת המפתח בעל השכיחות (כפילות) המכסימלית במבנה S ;זמן הריצה: $O(1)$;

$\text{SMALLER-KEYS}(S,k)$: החזרת מספר המפתחות במבנה S , הקטנים מהערך הנתון k (לכל מפתח סופרים את הכפילות שלו); זמן הריצה: $O(\lg n)$.

הערה: המבנה S יכול להיות מורכב מכמה מבני נתונים פשוטים יותר.

פתרון:

מבנה הנתונים S מורכב מעץ אדום-שחור מורחב T (עץ ערכי מיקום) ומעץ אדום-שחור רגיל F.
כל צומת z ב- T מכיל את השדה $freq[z]$ (השכיחות של המפתח) ואת השדה $size[z]$ (מספר המפתחות בתת-עץ המושרש ב- z, כולל כפילויות). כל צומת בעץ T מכיל מצביע לצומת מקביל בעץ F, וגם בכיוון ההפוך. המפתח ב- F הוא השכיחות של המפתח ב- T.

$\text{INSERT}(S,k)$: מחפשים את המפתח k בעץ T;

אם הוא נמצא (בצומת z), מוסיפים 1 לשדה $freq[z]$; מוסיפים 1 לשדה $size[z]$ בכל האבות הקדמונים של z; מוסיפים 1 למפתח המקביל ב- F, מוחקים את הצומת ומכניסים אותו מחדש;
אם הוא לא נמצא, יוצרים צומת חדש ב- T בעל מפתח k וכפילות 1 ויוצרים צומת חדש ב- F בעל מפתח 1; אם יש צורך לבצע סיבובים ב- T, משתמשים בנוסחאות

$$size[y] \leftarrow size[x]$$
$$size[x] \leftarrow size[left[x]] + size[right[x]] + freq[x]$$

מוסיפים 1 לשדה $size[z]$ בכל האבות הקדמונים של z; זמן ריצה $O(\lg n)$.

$\text{BUILD}(S)$: מבצעים N פעולות הכנסת מפתח; זמן הריצה הכולל $O(N \cdot \lg n)$.

$\text{MAX-FREQ}(S)$: עוברים מהצומת בעל המפתח המכסימלי ב- F אל המפתח המקביל ב- T; מחזירים את מפתח הצומת; זמן ריצה $O(1)$.

$\text{SMALLER-KEYS}(S,k)$: מבצעים את פעולת החיפוש בעץ T אחר המפתח k; בכל צומת z לאורך מסלול החיפוש אוגרים את המספר $size[left[z]]$ (אם $left[z]$ קיים; מחזירים את סכום שדות אלה; זמן הריצה $O(\lg n)$.

בהצלחה!

אפשר להשתמש בכל עובדה או תוצאה הנמצאות בספר הלימוד או במדריך הלמידה, ללא הוכחה או הסבר. חובה להוכיח או להסביר כל טענה אחרת.
אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.

שאלה 1

נתונה נוסחת הנסיגה

$$T(n) = a \cdot T\left(\sqrt{n}\right) + \lg^2 n$$

כאשר $a > 0$ הינו פרמטר ממשי.
פתרו את נוסחת הנסיגה עבור הערכים האפשריים השונים של a.

פתרון:

נשתמש בשיטת החלפת המשתנים: $m = \lg n,\ \ n = 2^n$. מתקבלת נוסחת הנסיגה

$$T(2^m) = a \cdot T\left(2^{m/2}\right) + m^2$$

אחרי הסימון $S(m) = T\left(2^m\right)$, מתקבלת הנוסחה

$$S(m) = a \cdot S\left(m/2\right) + m^2$$

נשתמש בשיטת האב. ההפרדה למקרים השונים מתבצעת לפי הערך $\log_b a = \lg a$.

1) אם $a > 4$, אזי $\lg a > 2$, אנחנו במקרה 1, ומתקבל הפתרון $S(m) = \Theta\left(m^{\lg a}\right)$;
2) אם $a = 4$, אזי $\lg a = 2$, אנחנו במקרה 2, ומתקבל הפתרון $S(m) = \Theta\left(m^2 \cdot \lg m\right)$;
3) אם $a < 4$, אזי $\lg a < 2$, אנחנו במקרה 3, ומתקבל הפתרון $S(m) = \Theta\left(m^2\right)$.

עבור נוסחת הנסיגה המקורית, מתקבלים הפתרונות

1) אם $a > 4$, $T(n) = \Theta\left(\lg^{\lg a} n\right) = \Theta\left(a^{\lg\lg n}\right)$;
2) אם $a = 4$, $T(n) = \Theta\left(\lg^2 n \cdot \lg\lg n\right)$;
3) אם $a < 4$, $T(n) = \Theta\left(\lg^2 n\right)$.

שאלה 2

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים:

$\text{BUILD}(L,S)$: בניית המבנה S מתוך רשימה נתונה L בת n מפתחות; זמן הריצה: $O(n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח החדש k למבנה S; זמן הריצה: $O(\lg n)$;

$\text{MEDIAN}(S)$: החזרת חציון המפתחות של S; זמן הריצה: $O(1)$;

$\text{DEL-MEDIAN}(S)$: מחיקת חציון המפתחות של S; זמן הריצה: $O(\lg n)$;

$\text{DECREASE-KEY}(S,p,d)$: הקטנת המפתח שאליו מצביע p בערך d; זמן הריצה: $O(\lg n)$.

הערה: המבנה S יכול להיות מורכב מכמה מבנים בסיסיים.

פתרון:

מבנה הנתונים S יהיה מורכב מערמת מכסימום H_1 וערמת מינימום H_2.

$\text{BUILD}(L,S)$: מעבירים את הרשימה L למערך; מפעילים את האלגוריתם SELECT למציאת החציון, אחר-כך מבצעים חלוקה סביב החציון. מ-$\lceil n/2 \rceil$ האיברים הקטנים יותר בונים את ערמת המכסימום H_1 ומ-$\lfloor n/2 \rfloor$ האיברים הגדולים יותר בונים את ערמת המינימום H_2 (משתמשים בשתי הגרסאות של BUILD-HEAP); זמן הריצה: $O(n)$.

$\text{INSERT}(S,k)$: אם המפתח k קטן מהחציון (או שווה לו), מכניסים לערמה H_1; אחרת, מכניסים אותו לערמה H_2. אחר-כך, אם מספר האיברים ב-H_1 גדול מ-$\lceil n/2 \rceil$, מוחקים את השורש של H_1 ומעבירים אותו ל-H_2; אחרת, אם מספר האיברים ב-H_2 גדול מ-$\lfloor n/2 \rfloor$, מוחקים את השורש של H_2 ומעבירים אותו ל-H_1; זמן הריצה: $O(\lg n)$.

$\text{MEDIAN}(S)$: החזרת מפתח השורש של H_1; זמן הריצה: $O(1)$.

$\text{DEL-MEDIAN}(S)$: מחיקת השורש של H_1; אחר-כך, אם מספר האיברים ב-H_2 גדול מ-$\lfloor n/2 \rfloor$, מוחקים את השורש של H_2 ומעבירים אותו ל-H_1; זמן הריצה: $O(\lg n)$.

$\text{DECREASE-KEY}(S,p,d)$: אם p מצביע אל איבר של H_1, מקטינים את המפתח בערך d ומפעילים את השגרה MAX-HEAPIFY; אחרת (p מצביע אל איבר של H_2), מקטינים את המפתח בערך d ובודקים אם הוא עדיין גדול או שווה לשורש של H_1. אם כן, מפעילים את השגרה המקבילה ל- HEAP-INCREASE-KEY עבור ערמות מינימום; אחרת, מחליפים את האיבר עם השורש של H_1, מתקנים את H_1 באמצעות הפעלת השגרה MAX-HEAPIFY ומתקנים את H_2 באמצעות הפעלת השגרה MIN-HEAPIFY. זמן הריצה: $O(\lg n)$.

שאלה 3

א' (5 נקודות) נתון המערך $[3,0,2,4,5,8,7,6,9]$ כפי שהוא נראה **אחרי** ביצוע שגרת החלוקה PARTITION.

אילו מהאיברים שלו היו יכולים לשמש כאיבר הציר בשגרת החלוקה? נמקו את תשובתכם.

ב' (20 נקודות) שגרת החלוקה PARTITION מופעלת על המערך $A[1..n]$ ויוצרת את המערך $B[1..n]$.

נתון מערך הפלט $B[1..n]$. כתבו אלגוריתם, יעיל ככל שאפשר, למציאת כל האיברים שהיו יכולים לשמש כאיבר הציר בשגרת החלוקה. נתחו את זמן הריצה שלו.

פתרון:

א' אחרי ביצוע החלוקה, כל איברי המערך הקטנים מאיבר הציר חייבים להימצא לפניו, וכל איברי המערך הגדולים ממנו חייבים להימצא אחריו. האיברים $< 4;5;9 >$ מקיימים את הנדרש.

ב' סורקים את המערך B משמאל לימין; כל איבר $B[i]$ שהוא מכסימום בתת-מערך $B[1..i]$ מועתק למערך עזר L. סורקים את המערך B מימין לשמאל; כל איבר $B[i]$ שהוא מינימום בתת-מערך $B[i..n]$ מועתק למערך עזר R. מחזירים את כל האיברים הנמצאים גם ב- L וגם ב- R. מכיוון שהמערך L ממוין בסדר עולה והמערך R ממוין בסדר יורד, אפשר למצוא את האיברים האלה בזמן $O(n)$. זמן הריצה הכולל: $O(n)$.

שאלה 4

הציעו מבנה נתונים S התומך בפעולות הבאות בזמנים הנדרשים:

$\text{SEARCH}(S,k)$: חיפוש אחרי המפתח k במבנה S;

$\text{INSERT}(S,k)$: הכנסת המפתח k למבנה S;

$\text{DELETE}(S,z)$: מחיקת האיבר שאליו מצביע z מהמבנה S;

$\text{MEDIAN}(k_1,k_2)$: החזרת החציון של תת-קבוצת המפתחות $\{k : k_1 \le k \le k_2\}$ של S.

כל אחת מהפעולות צריכה להתבצע בזמן $O(\lg n)$.

פתרון:

נשתמש בעץ אדום-שחור מורחב (עץ ערכי מיקום) T.

$\text{SEARCH}(S,k)$: חיפוש רגיל בעץ אדום-שחור ; זמן הריצה : $O(\lg n)$.

$\text{INSERT}(S,k)$: הכנסה רגילה בעץ ערכי מיקום ; זמן הריצה : $O(\lg n)$.

$\text{DELETE}(S,z)$: מחיקה רגילה בעץ ערכי מיקום ; זמן הריצה : $O(\lg n)$.

$\text{MEDIAN}(k_1,k_2)$: בעזרת השגרה OS-RANK מוצאים את המיקום r_1 של k_1 ואת המיקום r_2 של k_2 ; בעזרת השגרה OS-SELECT מוצאים את הערך שדירוגו $\lfloor (r_1+r_2)/2 \rfloor$; זמן הריצה : $O(\lg n)$.

הערה : הפתרון מסתמך על ההנחה שהמפתחות k_1 ו- k_2 נמצאים במבנה S.

שאלה 5

נתון עץ אדום-שחור T ; נסמן ב- ln את גודל התת-עץ השמאלי של השורש וב- rn את גודל התת-עץ הימני של השורש.

האם היחס $ln < 128 \cdot rn$ מתקיים תמיד? הוכיחו או הביאו דוגמה נגדית.

הערה: גודל תת-עץ (בעץ אדום-שחור) הינו מספר הצמתים הפנימיים שלו.

פתרון:

נבנה את העץ האדום-שחור הבא : התת-עץ הימני הוא עץ שלם בגובה h ; כל הצמתים שלו שחורים. התת-עץ השמאלי הוא עץ שלם בגובה $2h$; הצמתים שלו צבועים לסירוגין, כל הצמתים ברמות הזוגיות שחורים, כל הצמתים ברמות האי-זוגיות אדומים. העץ המתקבל הינו עץ אדום-שחור חוקי.

גודל התת-עץ הימני הוא $rn = 2^h - 1$; גודל התת-עץ השמאלי הוא $ln = 2^{2h} - 1$.

אם מתקיים התנאי $ln < 128 \cdot rn$, אז $2^{2h} - 1 < 128 \cdot (2^h - 1) < 2^{h+7} - 1$; כלומר, $h < 7$.

מסקנה : התנאי הנתון לא מתקיים תמיד.

אפשר להשתמש בכל עובדה או תוצאה הנמצאת בספר הלימוד או במדריך הלמידה, ללא הוכחה או הסבר. חובה להוכיח או להסביר כל טענה אחרת.
אין צורך לכתוב פסֵידוקוד, אלא אם נדרש במפורש.

שאלה 1

נתונה קבוצה $P=\{p_1,...,p_n\}$ של נקודות במישור הממשי. נתונים בנוסף שני מספרים ממשיים a ו- b.

כתבו אלגוריתם למציאת שתי נקודות $p_i, p_j \in P$, $i \neq j$, $p_i=(x_i,y_i)$, $p_j=(x_j,y_j)$, המקיימות את התנאי $|ax_i+by_i|=|ax_j+by_j|$ (או קביעה שאין שתי נקודות כאלה). זמן הריצה הנדרש של האלגוריתם הוא $O(n\cdot\lg n)$.

פתרון:

נבנה את המערך $C[1..n]$; נגדיר $C[i]=|ax_i+by_i|$, $i=1,...,n$ (כאשר $p_i=(x_i,y_i)$). ממיינים את המערך C באמצעות אלגוריתם מיון אופטימלי; אחר-כך, בודקים, בזמן לינארי, אם קיימים במערך C שני ערכים בעלי ערך זהה. זמן הריצה הכולל הינו $O(n\cdot\lg n)$.

שאלה 2

נתונים מערך $A[1..n]$ של מספרים ממשיים ומספר טבעי k המקיים את התנאי $3k<n$.

כתבו אלגוריתם הבודק האם **קיים** איבר z ב- A המקיים את שני התנאים הבאים:

(1) A מכיל לפחות $n-3k$ איברים קטנים מ- z ;

(2) z מופיע יותר מ- k פעמים ב- A.

זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n)$.

פתרון:

אילו המערך A היה ממוין, כל המופעים של z היו מופיעים ברצף והתחלתו של הרצף אחרי המיקום $n-3k$; רצף כזה, באורך גדול מ- k, חייב לכלול את המיקום $n-2k$ או את המיקום $n-k$.

לכן, האלגוריתם המוצע יפעל בצורה הבאה:

נמצא את ערך המיקום ה- $n-2k$ בעזרת האלגוריתם SELECT ; נבדוק אם ערך זה מופיע k פעמים לפחות במערך A ; אם כן, ערך זה הוא הערך z הנדרש; אחרת,

נמצא את ערך המיקום ה- $n-k$ בעזרת האלגוריתם SELECT ; נבדוק אם ערך זה מופיע k פעמים לפחות במערך A ; אם כן, ערך זה הוא הערך z הנדרש; אחרת, הערך הנדרש לא קיים.

שאלות חזרה

שאלה 1

מצאו פתרון אסימפטוטי הדוק עבור נוסחת הנסיגה הבאה (α הוא פרמטר ממשי חיובי) :

$$T(n) = 27T(n/3) + 2n^{\alpha}\log^{\alpha+1} n + 120\log n$$
$$T(1) = \Theta(1)$$

שאלה 2

א. נשנה את שגרת החלוקה של האלגוריתם מיון-מהיר באופן הבא : בהינתן מערך A באורך n, נבחר כאיבר ציר את האיבר ה**מינימלי** בתת-מערך $A[\lfloor n/4 \rfloor + 1..n - \lfloor n/4 \rfloor]$; נחזור על פעולה זו בכל קריאה רקורסיבית.
מהו זמן הריצה של האלגוריתם מיון-מהיר במקרה הזה? הוכיחו את טענתכם.

ב. נשנה את שגרת החלוקה של מיון-מהיר באופן דומה, אך הפעם נבחר כאיבר ציר את ה**חציון** של התת-מערך $A[\lfloor n/4 \rfloor + 1..n - \lfloor n/4 \rfloor]$.
מהו זמן הריצה של האלגוריתם מיון-מהיר במקרה הזה? הוכיחו את טענתכם.

שאלה 3

נתונה מטריצה של מספרים שלמים עם m שורות ו- n עמודות, שכל שורה בה ממוינת בסדר עולה.
אנו מעוניינים באלגוריתם להדפסת כל איברי המטריצה בסדר ממוין (עולה).
לדוגמא, בהינתן המטריצה הבאה :

$n = 5$, $m = 3$

3	5	12	13	50
1	4	8	8	14
4	5	10	19	20

הפלט של האלגוריתם יהיה : 1 3 4 4 5 5 8 8 10 12 13 14 19 20 50

א. להלן פתרון אחד לבעייה. נתחו את סיבוכיות הזמן והמקום הנוסף שלו (הניחו כי m הוא חזקה של 2).

- ממזגים כל שתי שורות עוקבות. מקבלים רצפים ממויינים באורך 2 שורות כל אחד.
ממזגים כל שני רצפים עוקבים (כלומר כל שתי שורות עם שתי השורות הבאות). מקבלים רצפים ממויינים באורך 4 שורות כל אחד.
כך ממשיכים, עד שכל המטריצה ממויינת (איברי כל שורה קטנים מאיברי השורה שמתחתיה).
- נעבור על המטריצה שורה שורה ונדפיס את כל האיברים.

ב. תארו אלגוריתם לפתרון הבעיה, הפועל בסיבוכיות זמן $O(m \cdot n \cdot \log m)$, ומשתמש ב- $O(m)$ זיכרון נוסף (כלומר בנוסף למטריצה הנתונה).

שאלה 4

סופר-מערך (super-array) הוא מבנה נתונים דינמי אשר מאותחל להכיל *n* אלמנטים ולתמוך בפעולות הבאות:

- **SET** (i,x) – מציבה את הערך *x* במקום ה-*i*-י במערך.
- **GET** (i) – מחזירה את ערכו של האיבר ה-*i*-י במערך.
- **SETALL** (x) – מציבה את הערך x בכל איברי המערך.

הצע מבנה נתונים למימוש המבנה סופר-מערך בצורה כזו שכל פעולה תדרוש O(1) זמן, והאתחול יתבצע בזמן ליניארי, במקרה הגרוע.

שאלה 5

אנו מעוניינים לתחזק מבנה נתונים המייצג סדרת מספרים טבעיים.

ממשו את הפעולות הבאות בסיבוכיות זמן O(log n) במקרה הגרוע כאשר n הוא מספר האיברים הנוכחי במבנה.

- Init() - אתחול סדרה ריקה
- Element(i) - החזרת את האיבר ה-i בסדרה
- Sum(i,j) - החזרת סכום איברי הסדרה בין המקום ה-i ל- j (כולל)
- Insert(i,a) - הוספת המספר הטבעי a כך שיהיה מיד אחרי האיבר ה- i בסדרה
- Delete(i) - מחיקת האיבר ה- i של הסדרה.

הסבירו את אופן מימוש הפעולות והעמידה בדרישות הסיבוכיות.

פתרון שאלה 3

סעיף א׳

זמן: $\Theta(m \cdot n \cdot log\ m)$

זיכרון נוסף: $\Theta(nm)$

סעיף ב׳

1. נאתחל ערימת מינימום מהעמודה הראשונה של המטריצה (נעתיק את איברי עמודה זו למערך ונבצע BUILD-MIN-HEAP). המפתח לערימה הוא הערך של האיבר במטריצה, אבל נשמור בנוסף בכל איבר בערימה גם את השורה ואת העמודה ממנו הוא מגיע.

2. כעת, כל עוד לא הודפסו $m \cdot n$ איברים:
   2.1 נדפיס את הערך שבשורש הערימה
   2.2 נשלוף אותו ע״י EXTRACT-MIN. נניח שהאיבר שנמחק הגיע משורה i ועמודה j
   2.3 אם $j<n$-1 (כלומר יש עוד איברים שלא הודפסו בשורה i) נכניס לערימה את האיבר משורה i ומעמודה j+1

ניתוח סיבוכיות:

זמן:

- שלב 1 לוקח $\Theta(m)$ כפי שנלמד על בניית ערימה.
- נשים לב שגודל הערימה נשאר חסום ע״י m לאורך ביצוע האלגוריתם (בכל צעד בשלב 2 מכניסים (אולי) איבר רק לאחר שמוציאים איבר אחר).
- בשלב 2 מתבצעים $m \cdot n$ צעדים, שכל אחד כולל הדפסה ($\Theta(1)$), הוצאת שורש מערימה ($\Theta(\log m)$), והכנסת איבר לערימה ($\Theta(\log m)$).

לכן בסה״כ האלגוריתם לוקח $\Theta(m + m \cdot n \cdot log\ m) = \Theta(m \cdot n \cdot log\ m)$.

זיכרון:

כאמור, גודל הערימה חסום ע״י m, ולכן זוהי סיבוכיות הזיכרון הנוסף של האלגוריתם.

פתרון שאלה 4

נשתמש במערך A כדי לאחסן את איברי סופר-המערך, במערך עזר B, ובשני משתנים: $count$ ו-$value$. המשתנה $count$ מונה את מספר הפעמים שהפעולה SETALL התבצעה. המשתנה $value$ ישמש לשמירת ערכה האחרון של פעולת SETALL. בזמן תהליך האתחול אנו מאתחלים את כל ערכיו של B להיות -1, את המשתנה $count$ להיות 0, ואת $value$ להיות -1. הפעולות המבוקשות ימומשו בדרך הבאה:

- **SET** (i,x) : $A[i] \leftarrow x,\ B[i] \leftarrow count$.
- **GET** (i) : **if** $B[i] = count$ **then return** $A[i]$, **otherwise return** $value$.
- **SETALL** (x) : $count \leftarrow count + 1,\ value \leftarrow x$.

פתרון שאלה 5

הרעיון הוא להשתמש **בעץ ערכי מיקום**. הנקודה הטריקית היא שהאיברים אינם מסודרים בעץ לפי המפתחות שלהם אלא לפי **מיקומם בסדרה**. יש לשמור בכל צומת v, בנוסף לאינפורמציה הרגילה בעצי ערכי מיקום, גם את **סכום כל המפתחות שנמצאים בתת העץ** של הצומת, בשדה שייקרא sum[v].

Init()– בניית עץ ערכי מיקום אדום שחור ריק T.

Element(i) - קרא ל- OS-SELECT(T,i).

Insert(i,a) - נמצא את האיבר במקום ה-i-י בסדרה ע"י OS-SELECT(T,i) ונתלה צומת עם מפתח a כך שיהיה האיבר שבא אחריו בסיור inorder: כלומר, אם אין לו בן ימני אזי נחבר את a כבנו הימני, אחרת, נרד לבנו הימני ונמשיך שמאלה ככל שנוכל ושם נתלה את a כבן שמאלי. לאחר מכן נתקן את העץ כרגיל (כמו בהכנסה לעץ אדום שחור).

Delete(i) - נמצא את האיבר למחיקה ע"י OS-SELECT(T,i), ונמחק אותו כרגיל.

Sum(i,j) - נממש את Less(i) שמחזירה את סכום איברי הסדרה שמיקומם קטן או שווה i (איך?), ובעזרת Less(i) קל לפתור את Sum(i,j):

return Less(T,j) - Less(T,i) + key[Element(i)]

יש להסביר כיצד מתחזקים את שדה sum בעת שינויים בעץ, וכיצד מממשים את Less.

אפשר להשתמש בכל עובדה או תוצאה המופיעה בספר הלימוד או במדריך הלמידה, ללא הוכחה או הסבר. חובה להוכיח או להסביר כל טענה אחרת.
אין צורך לכתוב פסידוקוד, אלא אם הדבר נדרש במפורש.

שאלה 1

פתרו את נוסחת הנסיגה שלהלן:

$$\begin{cases} T(1)=\Theta(1) \\ T(n)=(9/4)\cdot T\left(\sqrt[3]{n^2}\right)+\sqrt[4]{\lg^3 n}\cdot(\lg\lg n)^3\cdot\left(\sqrt[4]{\lg^5 n}+(\lg\lg n)^5\right) \end{cases}$$

שאלה 2

נתונה רשימת קלט בעלת n איברים. קוראים את האיברים ברשימה בזה אחר זה.

(15 נק') א. עומד לרשותנו זיכרון מחשב שגודלו $O(k)$, כאשר $k<n$. הסבירו כיצד ניתן למצוא בזמן $O(n+k\lg k)$ את k האיברים הגדולים ברשימה (בסדר ממוין).

(10 נק') ב. נניח עתה כי לרשותנו זיכרון בגודל $O(n)$ לקליטת האיברים. הסבירו כיצד ניתן למצוא את k האיברים הגדולים ביותר בזמן $O(n)$.

שאלה 3

הציעו מבנה נתונים S המבצע את הפעולות שלהלן בזמנים הנדרשים (n מסמן את המספר הכולל של איברים במבנה ו- m מסמן את מספר האיברים השונים זה מזה):

$\text{BUILD}(L,S)$: בניית המבנה S מרשימה **ממוינת** L בת n איברים; זמן הריצה: $\Theta(n)$;

$\text{INSERT}(S,k)$: הכנסת מפתח k לתוך המבנה S; זמן הריצה: $\Theta(\lg m)$;

$\text{DELETE-LAST}(S,k)$: מחיקת העותק שנכנס אחרון של המפתח k מהמבנה S (אם הוא קיים); זמן ריצה: $\Theta(\lg m)$;

$\text{MAX-FREQ}(S)$: החזרת השכיחות המקסימלית של איבר במבנה S; זמן ריצה: $\Theta(1)$.

הערה: מבנה הנתונים S יכול להיות מורכב מכמה מבנים יסודיים.

שאלה 4

צומת בעץ בינרי נקרא מאוזן אם גובה התת-עץ השמאלי שווה לגובה התת-עץ הימני.

בהינתן עץ בינרי T בן n צמתים, נניח שכל הצמתים בעץ מאוזנים פרט לשורש: ההפרש בין גובה הבן השמאלי שלו וגובה הבן הימני שלו הינו 2.

(10 נק') א. מהו המספר n של צמתים כפונקציה של הגובה h של העץ T?

(15 נק') ב. הראו כיצד ניתן לאזן את השורש בעזרת סיבוב אחד או שני סיבובים. איך משתנה גובה העץ? (אין חובה לשמור על איזון הצמתים האחרים.)

שאלה 5

הציעו מבנה נתונים S שבאמצעותו ניתן לבצע את הפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים של S):

SEARCH(S,k): חיפוש אחר המפתח k במבנה S; זמן הריצה: $O(\lg n)$;

INSERT(S,k): הכנסת איבר חדש בעל המפתח k למבנה S; זמן הריצה: $O(\lg n)$;

DELETE(S,p): מחיקת האיבר שאליו מצביע p מהמבנה S; זמן הריצה: $O(\lg n)$;

DECREASE(S,k,d): הורדת הערך $d>0$ מכל המפתחות של S הקטנים מ-k; זמן הריצה: $O(\lg n)$;

INCREASE(S,k,d): הוספת הערך $d>0$ לכל המפתחות של S הגדולים מ-k; זמן הריצה: $O(\lg n)$.

2014 א' – מועד 86.

1. נפתור בעזרת הצבה ושיטת האב:

$$T(n) = \left(\frac{9}{4}\right) T\left(\sqrt[3]{n^2}\right) + \sqrt[4]{\lg^3 n}\ (lglgn)^3\ (\sqrt[4]{\lg^5 n} + (lglgn)^5)$$

נציב: $m = lgn, 2^m = n$

$$T(2^m) = \frac{9}{4} T\left(2^{\frac{2m}{3}}\right) + m^{\frac{3}{4}} \lg^3 m \left(m^{\frac{5}{4}} + \lg^5 m\right)$$

נציב פעם שניה: $S(m) = T(2^m)$

$$S(m) = \frac{9}{4} S\left(\frac{2m}{3}\right) + m^{\frac{3}{4}} \lg^3 m \left(m^{\frac{5}{4}} + \lg^5 m\right)$$

$$S(m) = \frac{9}{4} S\left(\frac{2m}{3}\right) + m^2 \lg^3 m + m^{\frac{3}{4}} \lg^8 m$$

נשתמש בשיטת האב: $a = \frac{9}{4}, b = \frac{3}{2}, \log_b a = \log_{\frac{3}{2}} \frac{9}{4} = 2$

$$f(m) = O(m^2 \log^3 m)$$

ולכן לפי מקרה 2 המורחב:

$$\mathrm{S(m)} = O(m^2 \log^4 m)$$

נציב חזרה ונקבל:

$$T(n) = O(\lg^2 n\,(lglgn)^4)$$

.2

a. ניצור מערך בגודל 2k, ונכניס אליו כל פעם K איברים מתוך הרשימה.
בפעם הראשונה יש K מספרים, אז אין צורך לעשות כלום. מהפעם השניה יש K2 איברים ובכל הכנסה, נעשה SELECT על מנת למצוא את ערך המיקום K ונעשה מסביבו PARTITION. מכיוון שיש 2k איברים במערך, כל פעולה כזאת תקח O(k) זמן ריצה.
אחרי כל חלוקה, יהיו לנו בסוף הרשימה את K המספרים הגדולים עד עתה, ואת ה-K הבאים נכניס במקום הקטנים ונחזור כך עד סוף הרשימה.
בסה"כ נבצע את הפעולה הזאת n/k פעמים, ולכן כל פעולה ההכנסה ומציאת ה- K הגדולים יקח לנו O(n) זמן.
לאחר מכן, נמיין את K הגדולים ב- klgk ובסה"כ: O(n + klgk)

b. קוראים את כל הרשימה למערך בגודל n, עושים SELECT על הערך מיקום n-k, ומבצעים סביבו PARTITION. סה"כ O(n).

3. נשתמש בעץ אדום שחור כשהצמתים הם לפי המפתחות, בכל צומת יש מחסנית כדי להכניס איברים עם אותו מפתח ושדה count שייצג את מספר האיברים באותו צומת. ובערימת מקסימום לייצוג השכיחויות. בכל צומת יוחזק מצביע לאיבר המתאים בערימה.
BUILD – נבנה עץ אדום שחור מרשימה ממוינית לוקח, זמן ריצה - O(n), כל איבר שהמפתח שלו כבר קיים, יוכנס לתוך המחסנית ויעדכן את ה- count. לאחר בניית העץ,נסרוק את m הצמתים ונכניס אותם למערך עזר עם מצביעים דו כיווניים בין הצומת לאיבר במערך כאשר הערך במערך הוא ה- count (השכיחות), ניצור ערימת מקסימום ממערך זה.
זמן ריצה O(n+m) אבל מכיוון ש- n>m, סה"כ נקבל O(n).
INSERT – מכיוון שיש לנו m צמתים, יקח לנו lgm זמן למצוא את הצומת המתאים. עדכון המחסנית, השדה count והאיבר המתאים בערימה לוקח O(1). תיקון הערימה ע"י "גלגול כלפי מעלה" במידת הצורך – lgm.
אם המפתח לא קיים, נוסיף צומת חדש לעץ עם count = 1, נוסיף אותו גם כאיבר חדש לעריצה ומצביע דו כיווני בין הצומת לערימה.
סה"כ זמן ריצה O(lgm) – מציאת איבר או הוספה ותיקון הערימה או הוספת איבר לערימה.
DELETE-LAST – מציאת הצומת עם המפתח (lgm), POP למחסנית ועדכון הcount (O1), ע"י המצביע נוריד גם 1 מהאיבר המתאים בערימה ונבצע heapify.
אם ע"י המחיקה, הגענו ל count = 0 נמחוק את האיבר מהעץ ומהערימה ונתקן - ובסה"כ O(lgm).
MAX-FREQ – החזרת השורש של ערימת המקסימום O(1).

4.

a. גובה העץ T הוא h, גובה התת עץ הימני הוא h-1 ומהנתון אנו יודעים שגובה התת עץ השמאלי הוא h-3.
ידוע לנו גם שהתת עצים הימני והשמאלי הם מאוזנים וכך גם כל הבנים שלהם ולכן הם עצים שלמים. מספר הצמתים כפונקציה של גובה בעץ שלם הוא: $n = 2^{h-1} - 1$
ולכן מספר הצמתים בעץ הימני הוא: $2^h - 1$ ומספר הצמתים בתת עץ השמאלי הוא $2^{h-2} - 1$, כמובן שצריך להוסיף את הצומת של השורש ובסה"כ מספר הצמתים בעץ T כפונקציה של h הוא: $n = 2^h + 2^{h-2} - 1$.

5. מבנה הנתונים יהיה עץ אדום שחור עם 2 שדות נוספים D ו- I. ה-D יסמן את הערך שצריך להוריד מכל הצמתים שקטנים ממנו, ו- I יסמן את הערך שצריך להוסיף לכל הצמתים הגדולים ממנו.
DECREASE – סורקים את העץ כלפי מטה מהשורש, אם מפתח הצומת קטן מ- K, נעדכן את הערך D הנוכחי שלו ל- D=D-d, ונמשיך לסרוק את התת עץ הימני. אם הוא גדול או שווה ל-K, נמשיך לסרוק את התת עץ השמאלי. סה"כ אפשר להגיע לגובה העץ – O(lgn).
INCREAE – שוב נסרוק מהשורש, הפעם אם המפתח קטן מ- K נמשיך לסרוק את התת עץ הימני ואם הוא גדול או שווה ל – K, נעדכן את הערך I של הצומת ל- I=I+d.
זמן ריצה – O(lgn).
SEARCH – נשתמש ב-2 משתני עזר d ו – i שיואתחלו ב-0. בכל צומת שמגיעים, מוסיפים את ערכי D, I של הצומת ל- d ו- i העזרים. נבצע השוואה: key[x]+d+i מול K שקיבלנו בשגרה כאשר X הוא הצומת הנוכחי. אם K קטן מהתוצאה נמשיך ללכת על התת עץ השמאלי לא לפני שנבטל את עדכון ה- i שלנו ע"י i=i-I , אם גדול אז נמשיך לסרוק את התת עץ השמאלי ונשווה ונבטל את עדכון d ע"י d = d – D. אם שווה, מצאנו. סה"כ זמן ריצה הוא שוב גובה העץ – logn.
INSERT – הכנסה רגילה לעץ א"ש, רק שבכל צומת בדרך אם פונים ימינה נוסיף את הערך D ל-D שבאיבר שלנו, ואם פונים שמאלה נוסיף את הערך i.

אפשר להשתמש בכל עובדה או תוצאה המופיעה בספר הלימוד או במדריך הלמידה, ללא הוכחה או הסבר. חובה להוכיח או להסביר כל טענה אחרת.

אין צורך לכתוב פסֵידוקוד, אלא אם הדבר נדרש במפורש.

שאלה 1

נתונות שתי קבוצות של מספרים S ו- T בעלות m ו- n איברים בהתאמה.

הציעו מבנה נתונים המאפשר לממש את השגרה $\mathrm{XOR}(S,T)$, המחזירה את ההפרש הסימטרי $S\Delta T=S\cup T-S\cap T$ בזמן $O\left(\left(m+n\right)\cdot\lg\left(\min\left(m,n\right)\right)\right)$. תארו את השגרה ונתחו את זמן ריצתה.

פתרון:

מעבירים את הקבוצה S למערך באורך m ואת הקבוצה T למערך באורך n.

נניח כי $m\le n$ ולכן $\min(m,n)=m$. ממיינים את המערך S באמצעות מיון אופטימלי, בזמן $O\left(m\cdot\lg m\right)$. עבור כל איבר של T, מבצעים חיפוש בינרי ב- S ; זמן ריצה $O\left(n\cdot\lg m\right)$. כל איבר של T שלא נמצא ב- S יוצא לפלט ; בסוף, מצרפים לפלט גם את איברי S. זמן ריצה כולל $O\left(\left(m+n\right)\cdot\lg m\right)$.

שאלה 2

כתבו גרסה של מיון-הכנסה, כאשר הקלט נשמר במחסנית. זמן הריצה הנדרש עדיין $\Theta\left(n^2\right)$.
מותר להשתמש בשתי מחסניות עזר.

פתרון:

נסמן ב- S את מחסנית הקלט וב- T , U את שתי מחסניות העזר. מעבירים את $n-1$ האיברים העליונים של S ל- T. לכל i , $i=1,\ldots,n-1$, מתבצע התהליך הבא: מעבירים כל האיברים הנמצאים ב- S למחסנית U ; אחר-כך, מעבירים את האיברים מ- U חזרה ל- S , ביחד עם האיבר העליון של T, כאשר בדרך מתבצע מיזוג איבר זה בתוך הסדרה הממוינת. זמן הריצה של כל השגרה הינו $\Theta\left(n^2\right)$.

שאלה 3

האם קיים עץ אדום-שחור המכיל:

(5 נק׳) א. 3 צמתים שחורים ו-4 צמתים אדומים;

(5 נק׳) ב. 4 צמתים שחורים ו-3 צמתים אדומים;

(5 נק׳) ג. 5 צמתים שחורים ו-2 צמתים אדומים;

(10 נק׳) ד. 6 צמתים שחורים וצומת אחד אדום.

בכל מקרה תנו דוגמה, או הוכיחו שעץ כזה לא קיים.

הערה: הכוונה לצמתים פנימיים בלבד.

פתרון:

א׳ השורש ושני בניו שחורים; לכל בן של השורש, שני בנים אדומים.

ב׳ השורש ובן אחד שחורים, הבן השני אדום. שני בניו של הבן השחור, אדומים; שני בניו של הבן האדום, שחורים.

ג׳ השורש שחור, שני בניו אדומים; לכל בן של השורש, שני בנים שחורים.

ד׳ לא ניתן לבנות עץ כזה (אילו היה אפשר לצבוע את השורש באדום, זה היה נותן לנו פתרון).

נתאר מצב כללי, כאשר יש לנו עץ אדום-שחור בעל גובה השחור b, ובתוכו צומת אדום יחיד (שלא יכול להיות השורש). במצב זה, כל מסלול מהשורש לעלה, שלא עובר דרך הצומת האדום, מכיל b צמתים פנימיים; המסלולים האחרים, אלה שכן עוברים דרך הצומת האדום, מכילים כל אחד $b+1$ צמתים פנימיים. נסמן ב- c את מספר הצמתים הפנימיים במסלול מהצומת האדום לעלה.

המספר הכולל של צמתים פנימיים בעץ הינו $n = 2^{b+1} - 1 + 2^{c}$.

אם $n = 7$, אזי $2^{b+1} + 2^{c} = 8$; מכיוון ש- $b > c$, למשוואה הזאת אים פתרון בשלמים.

שאלה 4

הציעו מבנה נתונים S שבאמצעותו ניתן לממש כל אחת מהפעולות הבאות בסיבוכיות המבוקשת:

$\text{SEARCH}(S,k)$: חיפוש אחר המפתח k במבנה S ; זמן הריצה: $O(\lg n)$;

$\text{INSERT}(S,k)$: הכנסת המפתח k למבנה S ; זמן הריצה: $O(\lg n)$;

$\text{DELETE}(S,k)$: מחיקת עותק כלשהו של המפתח k מהמבנה S ; זמן הריצה: $O(\lg n)$;

$\text{MODE}(S)$: החזרת ערך המפתח בעל השכיחות הגבוהה ביותר; זמן הריצה: $O(1)$;

$\text{MARK}(S,t)$: החזרת ערך המפתח בעל רישום הזמן ה- t הקטן ביותר; זמן הריצה: $O(\lg n)$.

הערות: n הוא מספר המפתחות השונים ב- S ; אחרי כל הכנסת עותק של המפתח k, רישום הזמן של k משתנה (לפי זמן ההכנסה של העותק החדש), כלומר, רישום הזמן של מפתח הוא זמן ההכנסה של העותק החדש ביותר שלו.

שאלה 5

מבנה נתונים S מורכב מערמת מינימום ראשית MH בגודל m ומ- m ערמות מינימום H_i, $i=1,...,m$. לכל $i=1,...,m$, האיבר $MH[i]$ מכיל מצביע אל שורש הערמה H_i. לכל $i=1,...,m$, ערך השורש $H_i[1]$ משמש גם כמפתח ב- $MH[i]$. נסמן ב- n את הגודל המכסימלי בין כל הערמות H_i, $i=1,...,m$.

א' כתבו את שגרת הכנסת מפתח חדש עבור המבנה S (הכנסה ל- MH או לאחת הערמות H_i).

ב' כתבו את שגרת מחיקת המינימום עבור המבנה S (מחיקה מ- MH או מאחת הערמות H_i).

לכל אחת משתי הפעולות, נתחו את זמן הריצה כפונקציה של m ו- n.