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:

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

$\text{DELETE-MIN}(L)$ – 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:

$\text{INSERT}(z, L)$ – 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 LIST-INSERT כאשר מחליפים כל מופע של $head[L]$ בצומת שאחריו רוצים להכניס את האיבר החדש.)

בסה"כ זמן הריצה של הפעולה הוא $O(n)$.

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

III. עץ חיפוש בינרי:

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

$\text{DELETE-MIN}(L)$ – 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:

```
BFS(T)
1   ENQUEUE(Q, root[T])
2   while not ISEMPTY(Q)
3       do  p ← DEQUEUE(Q)
4           if left[p] ≠ NIL
5               then ENQUEUE(Q, left[p])
6           if right[p] ≠ NIL
7               then 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.

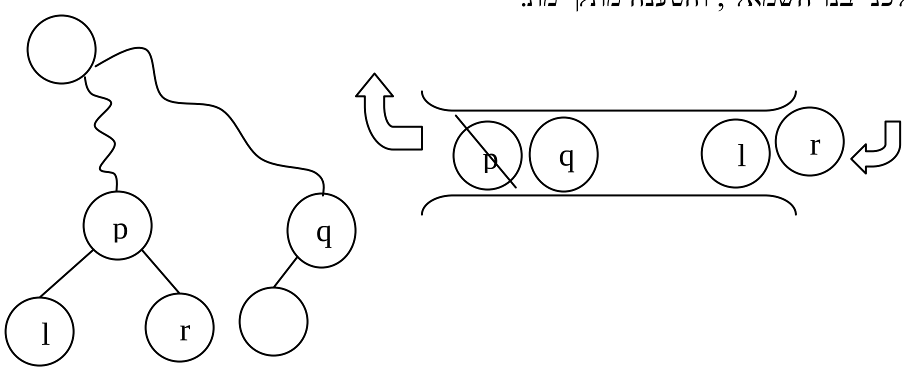
טענה: לפני כל איטרציה של הלולאה בשורה 2, התור מכיל את הצמתים מהצומת שבראש התור ועד לצומת שלפני בנו השמאלי, בסדר המוגדר על-ידי סריקה לפי רמות.

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

סעיף א:

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 $S[j] \leq z-S[i]$. כלומר, $S[j]+S[i] \leq z$. מכיוון שלכל $j^* \leq j$ מתקיים $S[j^*] \leq S[j]$, נקבל שיש בדיוק j איברים ב-S שעבורם $S[j]+S[i] \leq z$. 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.

סעיף ב:

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 שלx ב-H_a קטנים ממנו (כמובן, ייתכן גם שהם קטנים מאיברים נוספים שהגיעו מ-H_b).

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

בכל שלב בלולאה נחליף אתx 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 שלM 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 שלx 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- $b=3$.)