שאלות מבחנים

נתונים שני מערכים ממוינים a ו- b, כאשר a בגודל n, ו- b בגודל m
יש למצוא חיתוך, כ"א את כל האלמנטים המשותפים, כאשר:

(א) m ו- n באותו סדר גודל.

(ב) n גדול מאד מ- m, לדוגמא: $m < \log n$

פתרון:

(א) נניח שהמערכים ממוינים בסדר עולה. יהי c מערך נוסף שבו נשמור את קבוצת החיתוך, כאשר גודלו $\min\{m,n\}$ (c יכול להיות גם רשימה מקושרת ואז גודלה יהיה תלוי בגודל של קבוצת החיתוך).
נגדיר שני משתנים i ו- j שווים לאפס בהתחלה ותפקידם להצביע לאיבר ה- i-י במערך a ולאיבר ה- j-י במערך b.
עכשיו נבצע השוואת אלמנטים בשני מערכים באופן הבא:

1. אם הגענו לקצה של אחד המערכים, נסיים את השוואת מערכים ונחזיר את קבוצת החיתוך ב- c.
2. כל עוד $a[i] < b[j]$ וגם $i < n$, הגדל את i באחד.
3. כל עוד $b[j] < a[i]$ וגם $j < m$, הגדל את j באחד.
4. אם $a[i] = b[j]$ שמור את $a[i]$ במערך c (או ברשימה מקושרת, אם c היא רשימה מקושרת).
   הגדל את i ב- 1 ואת j ב- 1.
   חזור ל 1.

* זמן הריצה של האלגוריתם הוא $O(n)$ (בהנחה ש $n \approx m$), בגלל שאנו עוברים על שני מערכים וזה בזמן $O(n)$.

(ב) נבצע חיפוש בינארי עבור כל אלמנט של b בתוך a.
בעלות של חיפוש בינארי עבור m אלמנטים תהיה ב $O(m \log n)$
כלומר, אבל $m < \log n$ ולכן חיפוש בינארי עבור כל האלמנטים במקרה שלנו יקח $O(\log n)$ פעולות.

$O(\log^2 n)$

(2) תאר אלגוריתם המקבל מצביע לצומת בעץ בינארי ומחזיר מצביע לצומת הבא בסדר (א) Inorder

(ב) Postorder

הנחה: לכל צומת מחובר מצביע לאב.

סיבוכיות של האלגוריתם צריכה להיות $O(h)$, כאשר h גובה של העץ.

פתרון:

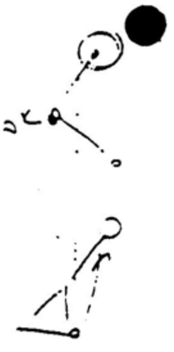
(א)

<u>אם לקודקוד יש בן ימני</u> אז האיבר הבא בסדר Inorder הוא קודקוד הכי שמאלי בתת-עץ של הבן הימני שלו.

<u>אחרת: אם קודקוד הוא בן שמאלי</u>, אז הבא בתור הוא אביו.

<u>אחרת: אם קודקוד הוא בן ימני</u>, אז הבא בתור הוא קודקוד האחרון שממנו הלכנו שמאלה במסלול מהשורש לקודקוד הנוכחי.

אם אין קודקוד כזה, אז הקודקוד שעליו קיבלנו מצביע הוא אחרון בסדר Inorder.



(ב)

<u>אם קודקוד הוא בן שמאלי</u>, אז הבא בתור הוא קודקוד הכי שמאלי בתת-עץ של אחיו הימני (אם לקודקוד אין אח ימני אז הבא בתור הוא אביו).

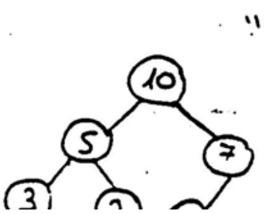
<u>אחרת: אם קודקוד הוא בן ימני</u>, אז הבא בתור הוא אביו.

<u>אחרת: אם קודקוד הוא לא בן שמאלי ולא בן ימני</u>, אז הוא שורש והוא אחרון בסדר.

(3) נתונה ערימה שממומשת ע"י מערך. יש לבדוק האם איבר מסוים m נמצא במסלול מהשורש לאיבר k.

הנחה: המיקום של k במערך ידוע, וגם ידועים את האינדקסים שלהם במערך.

ערימה:



$k=2 \quad m=5$

כן, 5 נמצא במסלול מהשורש ל-2.

[illegible]

לשורש אורך של המסלול בעץ שווה לגובה העץ לכל היותר, כלומר

ל-$\log n$ חיפוש בינארי על המסלול באורך $\log n$ יקח $O(\log(\log n))$

* אינדקס הצמתים שנמצאים במסלול החיפוש הם:

$i, i/2, i/4, \ldots, i/2^l, \ldots, 1$

כאשר i = אינדקס של צומת בעלת גובה K.

4. נתון מערך L בגודל m. ידוע ש-n התאים הראשונים בו מכילים מספרים שלמים ממוינים בסדר עולה, אבל n לא ידוע.
ידוע שבשאר התאים (החל מתא $n+1$) מכילים איזשהו ערך M השווה גדול (maxint).
תאר אלגוריתם שמקבל ערך x ומוצא מקום שלו בין n המקומות הראשונים במערך (או מחזיר -1 אם x לא נמצא שם) בזמן $O(\log n)$.
* זכור ש-n לא ידוע.

פתרון:

נבנה בצורה הדרגתית איברים במערך עד שנגיע לתא עם איבר שגדול מ-x ואז נפעיל חיפוש בינארי בחלק של המערך שצריך להכיל את x כמו באלגוריתם יותר של הרשימה הסופית, נפעיל בצורה כזו בשלבים, כלומר נשווה את x עם $L[1]$, אח"כ עם $L[2]$, אח"כ עם $L[4]$ ובשלב ה-k-י עם $L[2^k]$. אחרי k צעדים נגיע לערך שגדול מ-x (ב-$L[2^k]$ יכול להיות גם ערך M).

בשלב ה-k: $2^k \le 2n \Rightarrow 2^{k-1} \le n \Rightarrow k \le \log n + 1$, לכן, זמן הריצה של חלק זה של אלגוריתם הוא $O(\log n)$, ועכשיו נבצע חיפוש בינארי בקטע: $L[2^{k-1}+1], \ldots L[2^k-1]$,
זה יקח $k-1$ השוואות לכל היותר, שזה גם $O(\log n)$ ולכן
זמן הריצה של כל האלגוריתם הוא $O(\log n)$.

5. נתון עץ חיפוש בינארי מאוזן (לדוגמא עץ שחור-אדום).

~~יש לתאר אלגוריתם למציאת איבר ה-i בגודלו מתוך האיברים~~
~~הנמצאים בעץ~~

~~* לדוגמא ניתן להשתמש באלגוריתם כבסיס לפתרון של האיברים בעץ~~

הנחה:

נניח שרוצים לבצע מציאת איבר ה-i' הקטן ביותר בעץ זה רב פעמים,
כאשר בכל קריאה נוספת נרצה שזמן הריצה יהיה $O(\log n)$.

פתרון:

תחילה יש לעבור על כל העץ עם סריקה בצורת ה-Postorder (רק לפני
שנכנס לבן ימני נשמור את מס' הצמתים שבעץ שאותו בן שמאלי מייצג.
ובחזרה מבן ימני לשורש נשמור את מס' הצמתים שבתת-עץ ימני).
אחרי סריקה כזו נדע את מס' האיברים הנמצאים בתת-עץ ימני ושמאלי
של כל צומת (נשתמש בשני שדות נוספים עבור כל צומת).
ועכשיו כל פעם שנרצה למצוא איבר ה-i', נבצע דבר הבא:

יהי z מצביע לצומת של עץ, ל-z יש 3 שדות:

L $z \to a$ – מס' האיברים שבתת-עץ השמאלי.

R $z \to b$ – מס' האיברים שבתת-עץ הימני.

$z \to key$ – ערך המפתח שבו.

תחילה z שווה למצביע לשורש.

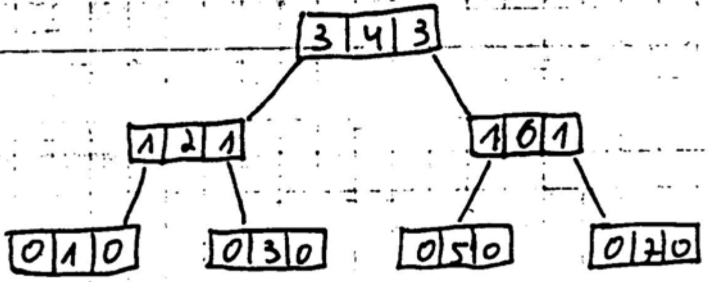
בצע:

1. אם $z \to a + 1 = i$, מחזירים את הערך ב-z הנוכחי.
2. אם $i \le z \to a$, הצב: $z = z \to left$ (כלומר z יצביע לבן השמאלי)
אחרת: $i = i - z \to a - 1$, אם $i \le z \to b$, הצב $z = z \to r$
אחרת, סיים את האלגוריתם עם כשלון.
חזור ל-1.

זמן החיפוש עבור איבר ה-i' כאשר כבר יודעים את מס' האיברים בתתי-עץ הימני ושמאלי של כל צומת, הוא $O(\log n)$.

תרגיל מבנה נתונים 21/2/99

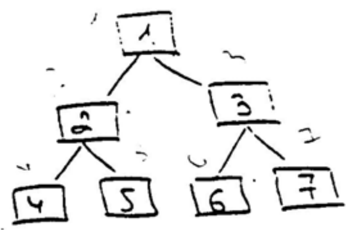
תרגול שאלה 2:



5 - 4 = 1

1 2 3 4 5 6 7

6. נתון עץ בינארי מלא ומאוזן המגודר ע"י פוינטרים. לכל צומת יש שני
פוינטרים המצביעים לבן שמאלי ובן ימני.
~~עץ הבנוי מספר הצמתים הוא 2^k-1 כאשר k הוא גובה העץ~~
לכל צומת פנימי יש בדיוק 2 בנים, כל העלים באותה רמה.
הצמתים של העץ הזה ממוספרים בצורה הבאה:
השורש מקבל את המספר 1.
לכל צומת i, הבן השמאלי שלו ממוספר כ- $2i$ והבן הימני
כ- $2i+1$
לדוגמא, העץ הבא הוא עץ כזה עבור $n=7$.



תאר אלגוריתם שבהינתן פוינטר לשורש root, ומספר x, מוצא
את הכתובת של הצומת שהמספר שלו הוא x (כלומר מצביע
לצומת עם המספר x).
הסיבוכיות של האלגוריתם צריכה להיות $O(\log x)$.
פתרון:
נסתכל על המסלול למטה.
בהינתן x, שארית החלוקה שלו ב- 2 נותנת את הפניה
האחרונה לצומת (נזכר איך בנינו אותו).
שארית 1 - פניה ימינה.
שארית 0 - פניה שמאלה.
ע"י חלוקות חוזרות ב 2 של x אפשר לשחזר את המסלול
(צריך רק להתחיל מהחלוקה האחרונה).
אם נשמור את השאריות החלוקה במחסנית, אז פשוט נעבור
על המסלול ונחזיר את המצביע לצומת שבו עצרנו.

(ג) תאר מבנה נתונים למערך אסוציאטיבי שעובדים בו מספרים float ושונים
של מערך רגיל N, שיכול להיות גדול (אין צורך לשים אותו במערך רגיל - כי זה
אסוציאטיבי) נקרא למערך בשם - V.
יש לבצע פעולות הבאות על המערך:

put(i,z) - השמת ערך z לתא i-י במערך
zero(i) - הצבת אפס לתא i
getfrom(i) - החזרת ערך שנמצא ב- i.

} $O(\log L)$ - ב

ב L - מספר האיברים השונים מ-0 במערך.

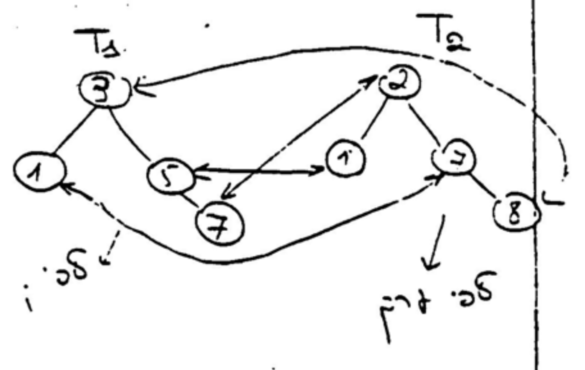
ובנוסף: print() - הדפסת רשומות {i, V[i]} בסדר עולה של V[i] שאינם אפסים, ב $O(L)$.

אתחול - ב-$O(1)$.

פתרון:
מבנה נתונים כזה שני עצי חיפוש כנ"ל מאוזנים (שחור-אדום לדוגמא)
עץ ראשון - עץ חיפוש לפי ערך (שומרים רק ערכים שונים מאפס)
עץ שני - עץ חיפוש לפי אינדקס i (עבור ערכים ששונים מאפס)
ובנוסף מקשרים בין צומת בעץ הראשון שמתאים לערך מסוים לבין
צומת בעץ השני עם i המתאים לערך (קישור בעזרת מצביעים)

דוגמא:

$V[1] = 7$
$V[3] = 8$
$V[5] = 1$
$V[7] = 2$



אתחול - יצירת 2 עצים ריקים.
put(i,z) - הוספת i ל-T_1 ואת z ל-T_2 ומקשרים ביניהם..

print() - עשיה Inorder ב- T_2 ונקבל את הערכים ממוינים

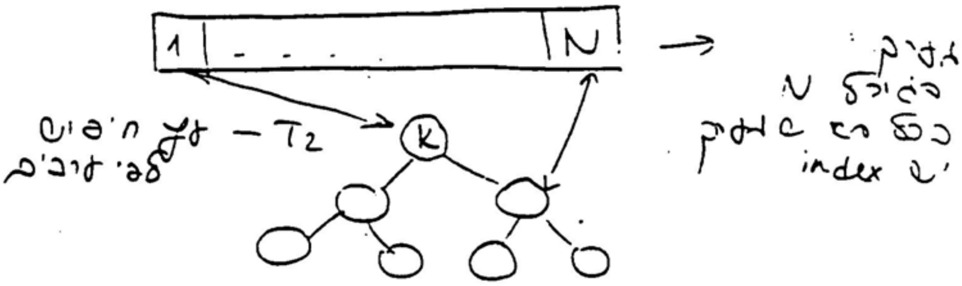
את [illegible] (ממוינים [illegible])

(ב)

כיצד תשתנה התשובה אם הפעולה get(i) חייבת להיות ב- O(1) והאתחול ב- O(N) (N = מספר האיברים במערך V).

פתרון:

בנוסף לעץ T יהיה מערך של מצביעים לכל האיברים ב- T_2.



(ג)

במחשב אישי המוגבל בזכרון יתכן שנכנס לקובץ קלט גדול מאוד. עבור סדרת הקלט בגודל n, יש להדפיס את m המספרים הכי קטנים בצורה ממוינת.

בנה מבנה נתונים וכתוב אלגוריתם שיבצע את הדרוש שיעמוד בסיבוכיות:

גודל הזכרון של המחשב הינו O(m)

סיבוכיות זמן - O(n log m)

פתרון:

בגלל המגבלה המקום בזכרון אנו ניקח כל פעם את ה-m-ים הבאים מהקלט ונמיין אותם למערך בגודל m שנקרא לו A. נשתמש בשני מערכים עזר נוספים B ו- C, בהתחלה שניהם ריקים. בכל שלב ניקח m-ים חדשים מהקלט, נכניס אותם למערך A ונמיין אותם, ונבצע מיזוג של Merge של A + B לתוך C ואת החצי של C נעביר ל B ונעבור לשלב הבא.

נסיים כאשר אין יותר m-יות, והתוצאה הינה ב- מערך C.

דוגמה:

$m = 3,\ n = 9$

[illegible]

A $\boxed{\ \ 1\ 0\ 8}$ $\boxed{\ }$ $\boxed{1\ 8\ 10}$ → $\boxed{1\ 8\ 10}$

B $\boxed{\text{ריק}}$ $\boxed{\ }$ $\boxed{1\ 8\ 10}$

C $\boxed{\text{ריק}}$ $\boxed{1\ 8\ 10}$ $\boxed{1\ 8\ 10}$

m מיון: →

7 6 5	→	5 6 7	→	5 6 7	→	
1 8 10		1 8 10		1 8 10		
1 8 10		1 8 10		1 5 6		

→	5 6 7	m - מיון: →	4 7 9	הכנסה →	4 7 9	A
	1 5 6		1 5 6		1 5 6	B
	1 5 6		1 5 6		1 4 5	C

↙ חזרה ל C

$$m(\underbrace{m\log m}_{\text{מיון}} + m) \approx O(n\log m)$$

מבני-נתונים

שאלות ממבחנים

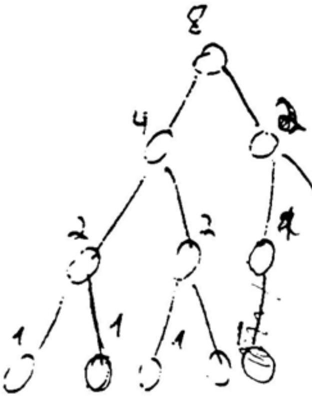
ב. תכנן מבנה נתונים המממש קבוצה של מספרים טבעיים עם הפעולות: insert, delete, search כמו בבעיית המילון ובנוסף הפעולה Find(k) אשר מקבלת מספר טבעי k ומחזירה את המספר ה-"k" בקבוצה S.

דהיינו: המספר ה-"k" בקבוצה S עם n איברים הוא מספר $a \in S$ כך שקיימים $k-1$ איברים ב-S קטנים מ-a, ו-$n-k-1$ איברים הגדולים מ-a.

כל הפעולות צריכות לקחת $O(\log n)$ בזמן.

פתרון

נבנה עץ חיפוש בינארי מאוזן (עץ אדום שחור) כאשר לכל צומת T node נוסיף שדה size שהוא מס' האיברים בתת עץ שמאלי של הצומת T (כולל הצומת עצמה):



למשל

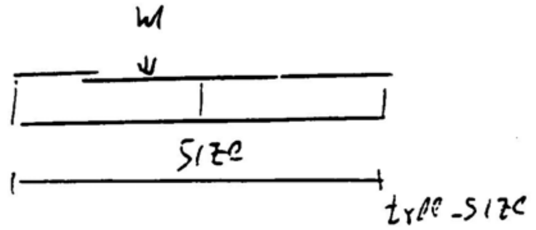
size	node

כאשר כל מס' מעל קודקוד מציין גודל size של אותו קודקוד.

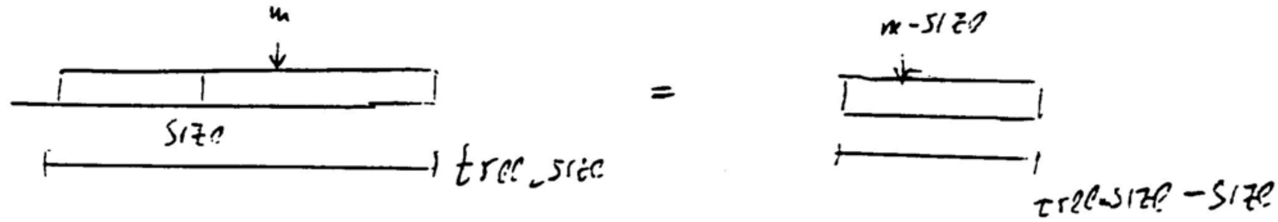
כאשר הפעולות הסטנדרטיות:

$O(\log n)$ { Insert(x), Delete(x) } לוקחות אותו זמן מכיוון שנבצע

כאשר m<R.size אזי להכניס איבר m בתת עץ שמאלי זה-
לחיפוש איבר m בכל העץ ולכן איבר m בכל העץ = איבר
m בתת עץ שמאלי:



כאשר R.size<m אזי חיפוש איבר m בכל העץ זהה לחיפוש איבר
m - R.size בתת עץ הימני.



ולכן ישירה נכונה, היא פועלת ב- O(h) כי היא יורדת לכל היותר
במסלול בעץ מהשורש לעלה ולכן סיבוכיות הפעולה היא O(lgn)

במערכת הפעלה מסויימת יש קבוצת תהליכים הממוספרים מ-1 עד n, כאשר n ידוע מראש, אך אינו קבוע. יש להכניס את התהליכים לתורי המתנה $Q_1, ..., Q_k$. כל תהליך יכול להמצא בתור אחד לכל היותר. כל התורים ריקים בהתחלה.

יש לממש את הפעולות הבאות: (סימון: p - מס׳ תהליך, Q - מצביע לתור)

- enqueue(Q, p) - הוסף תהליך p לזנב התור Q. הסיבוכיות הנדרשת O(log |Q|).
- dequeue(Q) - הוצא תהליך מראש התור Q החזר את מס׳ התהליך.
הסיבוכיות הנדרשת O(log |Q|).
- state_of_process(p) - החזר <Q, t> כאשר:
Q - התור בו התהליך p נמצא.
t - מספרו הסידורי של p בתור Q (מיספור התהליכים בתור הינו מהראש לזנב).
הסיבוכיות הנדרשת O(log |Q|).
- merge_queues(Q_1, Q_2) - שרשר את התור Q_2 לזנב של Q_1. סדר האיברים בשני התורים לא משתנה. אחרי הפעולה התור Q_2 ריק.
הסיבוכיות הנדרשת:

$$\text{lk}\begin{cases} O(|Q_1| \lg |Q_2|) \\ O(|Q_2| \lg |Q_1|) \end{cases}$$

א. תאר את מבנה הנתונים הנדרש במילים ובציור.

(5) <u>Dequeue(Q)</u> נגש לתור Q עפ"י מערך B לראש הערימה, נבצע
Delete_head(Q), מוציא את ראש הערימה ומחליף עם בן בעל מפתח
מינימלי! (לפי השגרה הידועה: Extract_min()), (ועדכון: Total_size --
בחזר מצביע חדש של שורש חדש של הערימה ב- B (ו- p-1) ברשימה A שבה
גם יתעדכן המצביע ל- NULL).

<u>Stat_O_p(p):</u>

נגש למערך A, עולה לראש הערימה - ישיר ב- $O(h)$
ונחזיר מצביע לאיבר (במערך B).

- נדפיס ב- $O(1)$ את השדה print(p.size) של p גודל הסדר של צומת p הנ"ל

<u>Merge_Q(Q1,Q2)</u>

נגש לערימה Q_1 ו- Q_2 (במערך B) בדוק לפי: Root.Q(size_total) מי מהערימות גדולה יותר, בה"כ נניח כי $|Q_2|<|Q_1|$.
אז לכל איבר $\forall p \in Q_2$ נבצע Q_1.Heap_Insert(p)
לפי מפתח קדימות נעדכן מפתחות קדימות במקביל ל- Q_1 ול- Q_2

- כך שעבור כל איברי Q_2 נעדכן גם במערך B אחרי הוצאת כל איברי Q_2 כי Q_2 כבר NULL.

<u>זמן ריצה לשגרה זו:</u> יש $|Q_2|$ הכנסות. כאשר כל הכנסה מתרחשת ב- $O(h) \mid h=\lg|Q_1|$ ולכן סה"כ זמן ריצה $O(|Q_1|\cdot\lg|Q_2|)$.