# מבני נתונים

MADE BY:TAL KATZ, OMER STERN,YUVAL COHN,AMIT SHAFRAN

1

**תור קדימויות** מוגדר כמבנה נתונים $S$ התומך בשתי הפעולות הבאות:

$\text{INSERT}(z,S)$ : הכנסת המפתח $z$ למבנה $S$;

$\text{DELETE}-\text{MIN}(S)$ : מציאת, החזרת ומחיקת המפתח המינימלי מהמבנה $S$.

כידוע, ניתן לממש תור קדימויות בערימה בינרית.

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

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

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

2

הצע מבנה נתונים, שבאמצעותו ניתן לממש כל אחת מהפקודות הבאות בסיבוכיות הזמן המבוקשת. n מציין את מספר האיברים במבנה.

| פקודה | סיבוכיות | משמעות |
|---|---|---|
| build (S) | $O(n \cdot \lg n)$ | אתחל מבנה S מרשימה של n איברים |
| insert (S, x) | $O(\lg n)$ | הכנס את x ל-S |
| find (S, x) | $O(\lg n)$ | מצא את x ב-S |
| max(S) | $O(1)$ | הדפס איבר מקסימלי ב-S |
| min (S) | $O(1)$ | הדפס איבר מינימלי ב-S |

3

א. הראו כיצד ניתן לממש שתי מחסניות באמצעות מערך אחד $A[1..n]$ ; הפעולות $PUSH$ ו- $POP$ צריכות להתבצע בזמן $O(1)$.

ב. הראו כיצד ניתן לממש תור באמצעות שתי מחסניות;
נתחו את זמן הריצה של הפעולות על התור.

4

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: $O(\lg n)$.

5 הציעו מבנה נתונים $S$ התומך בפעולות הבאות בזמנים הנדרשים:

$\mathrm{SEARCH}(S,k)$: חיפוש אחרי המפתח $k$ במבנה $S$;

$\mathrm{INSERT}(S,k)$: הכנסת המפתח $k$ למבנה $S$;

$\mathrm{DELETE}(S,z)$: מחיקת האיבר שאליו מצביע $z$ מהמבנה $S$;

$\mathrm{MEDIAN}(k_1,k_2)$: החזרת החציון של תת-קבוצת המפתחות $\{k : k_1 \le k \le k_2\}$ של $S$.

כל אחת מהפעולות צריכה להתבצע בזמן $O(\lg n)$.

6 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 $n$.

7 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 $S$.

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.

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

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

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

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

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

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

9 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-$n$).

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

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

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

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

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

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

11 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.

12 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.

13

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.

---

14

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 האיבר 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.

---

15

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: $O(n\cdot\lg n)$;

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

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

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

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

---

16

נתון מספר טבעי $m>0$ (קבוע).

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

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

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

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

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.

17

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: $O(n\cdot\lg n)$;

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

DEL-MEDIAN($S$): מחיקת החציון של המבנה $S$; זמן הריצה: $O(\lg n)$;

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.

18

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- $S$):

BUILD($S$): בניית המבנה $S$ מסדרה של $N$ מפתחות לא בהכרח שונים; זמן הריצה: $O(N\cdot\lg n)$;

INSERT($S,k$): הכנסת המפתח $k$ (מופע חדש או חוזר שלו) למבנה $S$; זמן הריצה: $O(\lg n)$;

MAX-FREQ($S$): החזרת המפתח בעל השכיחות (כפילות) המכסימלית במבנה $S$; זמן הריצה: $O(1)$;

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.

19

20

# פתרונות מבני נתונים

<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 – $O(h)$.

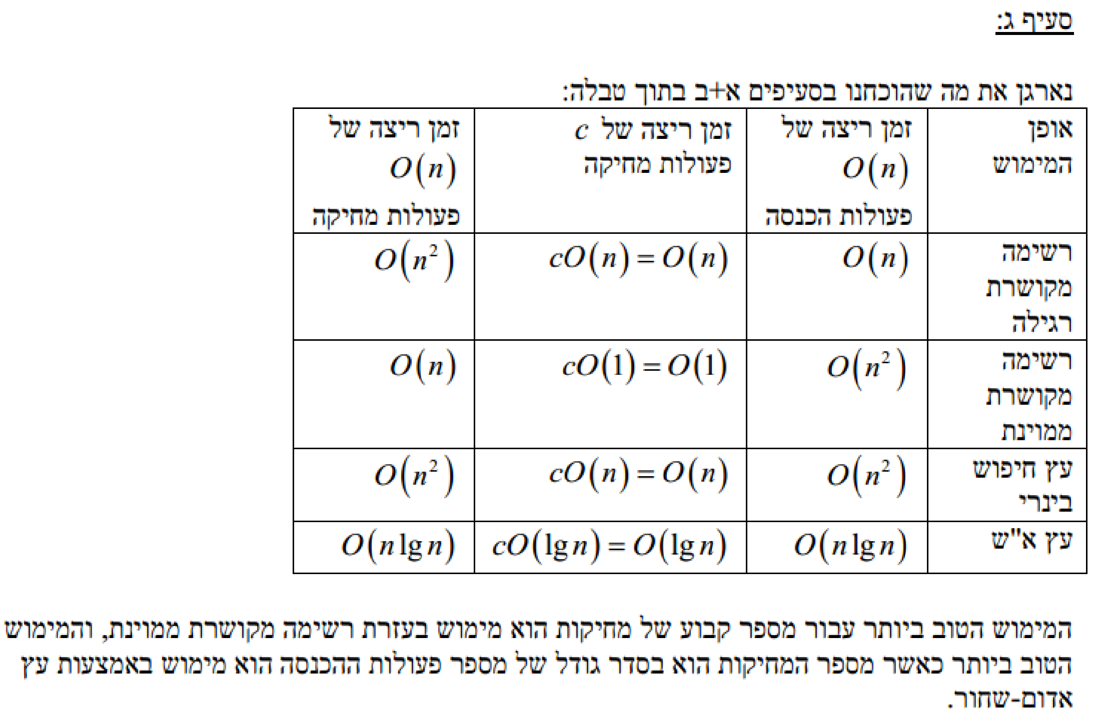
( $h$ מציין את גובה העץ)

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 $O(\lg n)$.

# 1 המשך



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 ממוינת, והמימוש הטוב ביותר כאשר מספר המחיקות הוא בסדר גודל של מספר פעולות ההכנסה הוא מימוש באמצעות עץ אדום-שחור.

# 2

נממש עץ א"ש עם מצביע לאיבר המינימאלי והאיבר המקסימאלי (האיבר הימני והשמאלי ביותר בעץ בהתאמה.

איתחול מבנה הנתונים דורש בניה של עץ א"ש שהיא ע"פ הידוע כמיון מערך.

הכנסת איבר למבנה הנתונים היא הכנסת איבר לעץ א"ש ועדכון המינימום או המקסימום במידת הצורך (אם האיבר החדש קטן מהמינימום או גדול מהמקסימום)

חיפוש של איבר הוא חיפוש של איבר בעץ א"ש

מציאת מקסימום או מינימום תבוצע ע"י קריאה למצביע המתאים.

3 א. נממש 2 מחסניות על מערך בצורה הבאה.
נשים מצביע בתחילת המערך למחסנית S פעולה Push תכניס איבר למערך במקום של המצביע ותזיז את המצביע לתא הבא במערך. אם המצביע מצביע על null אז נכניס את האיבר למקום הראשון במערך ונצביע על המקום השני.
פעולת POP תבצע מחיקה אם המצביע לא null תמחק את האיבר במיקום הקטן ב1 ממיקום המצביע ותשנה את המצביע למיקום שנמחק. אם המיקום שנמחק הוא האיבר הראשון במערך נצביע על null.
נממש את המחסנית T בצורה אנלוגית על סוף המערך
נשים מצביע בסוף המערך למחסנית T פעולה Push תכניס איבר למערך במקום של המצביע ותזיז את המצביע לתא הקודם במערך. אם המצביע מצביע על null אז נכניס את האיבר למקום האחרון במערך ונצביע על המקום האחד לפני אחרון.
פעולת POP תבצע מחיקה אם המצביע לא null נמחק את האיבר במקום הגדול ב1 מהמקום שעליו המצביע ואם מחקנו את האיבר במקום האחרון במערך אז נצביע על null.

ב. נממש תור C בעזרת 2 מחסניות A,B.
פעולת Push בתור תבצע פעולה Push למחסנית A.
פעולת POP בתור תבצע פעולת POP במחסנית B אם B לא ריקה.
אם B ריקה אז נעביר את כל האיברים שב A לB (נהפוך את המחסנית) ונבצע POP במחסנית B.
ניתקל במצב גלישה כאשר ננסה להכניס יותר מהקיבולת בA (בהנחה שA וB באותו הגודל)
מצב חמיקה אם ננסה לבצע POP כאשר התור ריק (כלומר 2 המחסניות ריקות)
פעולת הPush מבצעת תמיד בסיבוכיות קבועה של $O(1)$
פעולת הPop תבצע במקרה הגרוע (שבוא מחסנית A מלאה וB ריקה) 2n פעולות push וPOP על המחסניות ועוד פעולת pop על B לאחר מכן ובסך הכל $O(2n+1)=O(n)$

4 מבנה הנתונים $S$ יהיה מורכב מערמת מכסימום $H_1$ וערמת מינימום $H_2$.

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: $O(n)$.

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: $O(\lg n)$.

MEDIAN$(S)$: החזרת מפתח השורש של $H_1$; זמן הריצה: $O(1)$.

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: $O(\lg n)$.

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: $O(\lg n)$.

5

נשתמש בעץ אדום-שחור מורחב (עץ ערכי מיקום) $T$.

SEARCH$(S,k)$ : חיפוש רגיל בעץ אדום-שחור ; זמן הריצה : $O(\lg n)$.

INSERT$(S,k)$ : הכנסה רגילה בעץ ערכי מיקום ; זמן הריצה : $O(\lg n)$.

DELETE$(S,z)$ : מחיקה רגילה בעץ ערכי מיקום ; זמן הריצה : $O(\lg n)$.

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 $S$.

6

(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) : בניית המבנה מורכבת מ- N פעולות הכנסה, כל אחת בזמן $O(\lg n)$ (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-עץ, אלא את מספר הרשומות שבו.

SEARCH$(S,k_0,k_1)$ : מחפשים בעץ את המפתח $k_0$; אם מצאנו, מבצעים חיפוש ליניארי ברשימה המקושרת אחר המפתח $k_1$; זמן הריצה: $O(\lg n + m)$;

EXTENDED-OS(S,j) : משנים את השגרה OS-SELECT כך שתעבוד עם מספר הרשומות; בכל צומת שאליו הגענו, חייבים לספור את הרשומות; זמן הריצה: $O(m \cdot \lg n)$;
אם מחזיקים בכל צומת שדה המכיל את מספר הרשומות, זמן הריצה יורד ל- $O(\lg n+m)$.

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

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 השנייה. 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.

8 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.

9 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.

10

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.

11

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.

12

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).

13 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.

14 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.

15 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.

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

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

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

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

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

16 מבנה הנתונים $S$ מורכב ממערך של $m$ מצביעים ל- $m$ עצים אדומים-שחורים $T_0,T_1,\dots,T_{m-1}$ וממערך של $m$ מצביעים ל- $m$ ערמות מכסימום $H_0,H_1,\dots,H_{m-1}$. 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- $T_r$.

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 $O(n)$.

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

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

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

17 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).

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 $O(n \cdot \lg n)$.

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 $O(\lg n)$.

DEL-MEDIAN($S$): מחיקת שורש ערמת המינימום; העברת איבר בין שתי הערמות, לפי הצורך; מחיקת האיברים המקביליים מהעץ ומהמחסנית. זמן ריצה $O(\lg n)$.

MIN-OLDEST($S,t$): מציאת ערך המיקום ה- $t$ בעזרת OS-SELECT($T,t$); מעבר אל המחסנית ובחירת $\min[z]$.

18 מבנה הנתונים $S$ מורכב מעץ אדום-שחור מורחב $T$ (עץ ערכי מיקום) ומעץ אדום-שחור רגיל $F$. כל צומת $z$ ב- $T$ מכיל את שדה $freq[z]$ (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- $T$.

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 $O(\lg n)$.

BUILD$(S)$: מבצעים $N$ פעולות הכנסת מפתח; זמן הריצה הכולל $O(N \cdot \lg n)$.

MAX-FREQ$(S)$: עוברים מהצומת בעל המפתח המכסימלי ב- $F$ אל המפתח המקביל ב- $T$; מחזירים את מפתח הצומת; זמן ריצה $O(1)$.

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 $O(\lg n)$.

19