פתרון שאלה 4

ערמת המכסימום ממומשת כעץ בינרי T; ההנחה היא ש-T הוא עץ בינרי כמעט שלם (כל הרמות מלאות, פרט, אולי, לרמה האחרונה שהיא מלאה חלקית בצד השמאלי שלה). נסמן ב-$root[T]$ את שורש העץ וב-$last[T]$ את הצומת האחרון.

לכל איבר x ב-T, נסמן את הבן השמאלי שלו ב-$left[x]$, את הבן הימני שלו ב-$right[x]$ ואת ההורה שלו ב-$p[x]$. נתחיל בשגרה MAX-HEAPIFY:

MAX-HEAPIFY(x)
$l \leftarrow left[x]$
$r \leftarrow right[x]$
if $l \neq$ NIL and $key[l] > key[x]$
  then $largest \leftarrow l$
  else $largest \leftarrow x$
if $r \neq$ NIL and $key[r] > key[largest]$
  then $largest \leftarrow r$
if $largest \neq x$
  then exchange $key[x] \leftrightarrow key[largest]$
        MAX-HEAPIFY($largest$)

ונמשיך בשגרות ערמת המכסימום האחרות.

החזרת המפתח המכסימלי:

HEAP-MAXIMUM(T)
return $key[root[T]]$

הוצאת האיבר בעל המפתח המכסימלי :

```
HEAP-EXTRACT-MAX(T)
 1  if last[T] = NIL
 2     then error "heap underflow"
 3  max ← key[root[T]]
 4  nl ← last[T]
 5  key[root[T]] ← key[nl]
 6  while p[nl] ≠ NIL and nl = left[p[nl]]
 7     do nl ← p[nl]
 8  if nl ≠ root[T]
 9     then nl ← p[nl]
10          nl ← left[nl]
11  while right[n] ≠ NIL
12      do nl ← right[nl]
13  if left[p[last[T]]] = last[T]
14     then left[p[last[T]]] ← NIL
15     else right[p[last[T]]] ← NIL
16  last[T] ← nl
17  MAX-HEAPIFY(root[T])
18  return max
```

ההבדל לעומת השגרה HEAP-EXTRACT-MAX הרגילה נובע מכך שאין לנו גישה ישירה לצומת שלפני האחרון ועלינו למצוא אותו. לאחר מכן צריך להסיר מהעץ את הצומת האחרון ולעדכן את המצביע לצומת האחרון החדש. זה מתבצע באופן הבא :

ראשית, עולים מהצומת האחרון במעלה העץ כל עוד לא הגענו לשורש והצומת הנוכחי הוא בן שמאלי של אביו (שורות 6-7). כעת יש שתי אפשרויות :

אם לאחר ביצוע הלולאה בשורות 6-7 הגענו לשורש, זאת אומרת שהצומת האחרון היה הצומת היחיד בשורה האחרונה בערמה; במקרה זה הצומת האחרון החדש הוא הצומת האחרון בשורה הקודמת. כדי להגיע אליו צריך ללכת מהשורש ימינה עד הסוף (שורות 9-12).

לעומת זאת, אם לאחר ביצוע הלולאה בשורות 6-7 לא הגענו לשורש, אז הצומת האחרון החדש יהיה העלה הימני ביותר בתת-עץ השמאלי של אביו של הצומת הנוכחי ; לכן כדי להגיע אליו עולים לאביו של הצומת הנוכחי, פונים לבנו השמאלי ואז הולכים ימינה עד הסוף (שורות 9-12).

בשורות 13-15 מציבים NIL בשדה המתאים בצומת שהיה אביו של הצומת האחרון הקודם, ובשורה 16 מעדכנים את המצביע לצומת האחרון החדש.

העלאת ערך המפתח של האיבר x לערך נתון k :

HEAP-INCREASE-KEY(x, k)

if $k < key[x]$
  then error "new key is smaller than current key"
$key[x] \leftarrow k$
while $x \neq root[T]$ and $key[p[x]] < key[x]$
    do exchange $key[x] \leftrightarrow key[p[x]]$
       $x \leftarrow p[x]$

הכנסה לעץ של צומת חדש בעל מפתח k :

MAX-HEAP-INSERT(T, k)

1 $nl \leftarrow last[T]$
2 while $p[nl] \neq$ NIL and $nl = right[p[nl]]$
3   do $nl \leftarrow p[nl]$
4 if $nl \neq root[T]$
5   then $nl \leftarrow p[nl]$
6        $nl \leftarrow right[nl]$
7 while $left[nl] \neq$ NIL
8   do $nl \leftarrow left[nl]$
9 $left[nl] \leftarrow y$ $\triangleright$ new node
10 $left[y] \leftarrow$ NIL
11 $right[y] \leftarrow$ NIL
12 $last[T] \leftarrow y$
13 $key[y] \leftarrow -\infty$
14 HEAP-INCREASE-KEY$(last[T], k)$

ההכנסה מתבצעת בשני שלבים: בשלב הראשון (שורות 1-13) מוסיפים לעץ צומת חדש ומעדכנים את $last[T]$; בשלב השני קוראים לשגרה HEAP-INCREASE-KEY כדי למצוא את המקום המתאים עבור הצומת החדש.

מחיקת האיבר x :

HEAP-DELETE(T, x)

HEAP-INCREASE-KEY(x, ∞)
HEAP-EXTRACT-MAX(T)

זמן הריצה של כל השגָרות הנ״ל, פרט לשגרת החזרת המכסימום, הינו $O(\lg n)$.