מבני נתונים ומבוא לאלגוריתמים – פתרון מועד 89 מסמסטר 2002א

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

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

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

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

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

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 $i \leftarrow 1$ **to** n
  do **read** (k)
    HASH-INSERT $(T[h(k)], k)$ ▸ insert to the array at index $h(k)$
for $i \leftarrow 1$ **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 $i \leftarrow 1$ **to** n
  do **read** (k)
    MAX-HEAP-INSERT $(T[h(k)], k)$

ד. במקרה הגרוע כל n האיברים יוכנסו לאותו מערך.

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

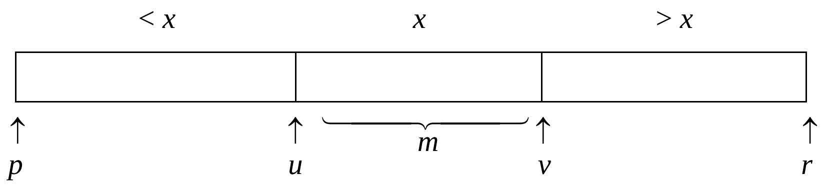
סה"כ: $n \cdot O(\lg n) = O(n \cdot \lg n)$

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!)