פתרון תרגיל מספר 7 ־ דאסט

שם: מיכאל גרינבאום, **ת"ז:** 211747639

14 במאי 2019

1. **פתרון:**

(א) **צ"ל:** מספר הקודקודים המקסימלי בתתי־העצים הוא $\frac{2n}{3}$
הוכחה:
נוכיח באינדוקציה על מספר האיברים בעץ הכמעט שלם.
<u>בסיס:</u> $n=0$, נשים לב כי יש 0 קודקודים ולכן יש לכל היותר $0 = 0 \cdot \frac{2}{3}$
$n=1$, נשים לב כי יש קודקוד שורש ו־0 פנימיים, למספר הקודקודים המקסימלי בתתי־העצים הוא $0 \leq \frac{2}{3} \cdot 1$,
<u>צעד:</u> נניח שהטענה נכונה ל־$k < n$ ונוכיח ל־n,
כל תתי עץ שהם הבנים של השורש, הם גם כן עצים כמעט שלמים מגובה קטן מ־n,
הערה: מספר הקודקודים של הבנים הוא לכל היותר $\frac{n-1}{2}$,
לכן מהנחת האינדוקציה על תתי עץ שהם הבנים של השורש, מתקיים שמספר הקודקודים המקסימלי בתתי־העצים הוא $\frac{2}{3} \cdot \frac{n-1}{2}$.
לכן מספר הקודקודים המקסימלי בתתי־העצים הוא

$$\frac{2}{3} \cdot \frac{n-1}{2} + 1 = \frac{n+2}{3} \overset{n\geq 2}{\leq} \frac{2n}{3}$$

כלומר קיבלנו כי הטענה נכונה עבור n, כנדרש
כלומר לכל ערך בעץ כמעט שלם, הראנו שלכל איבר בגובהו מתקיים שמספר הקודקודים המקסימלי בתתי־העצים הוא $\frac{2n}{3}$

מ.ש.ל.א.☺

(ב) **צ"ל:** $T(n) \leq T\left(\frac{2}{3}n\right) + c$
הוכחה:
תחילה נשים לב שיש מספר סופי של פעולות על האיבר הראשון, נסמן מספר פעולות זה ב־c,
וב־t את כמות הפעולות שמתבצע על כל איבר ב־$while-loop$, אזי זמן הריצה קצר מזמן הריצה על המסלול הארוך ביותר בעץ,
נשים לב כי $t \leq c$ כי הפעולות t מתבצעות גם על האיבר הראשון שמספר הפעולות עליו הוא c,
לכן

$$T(n) = c + \sum_{x \text{ in longest path and not root}} t \leq c + \left(c + \sum_{x \text{ in longest path and not root or its child}} t\right) = c + T\left(\frac{2}{3}n\right)$$

$$\Rightarrow \boxed{T(n) \leq T\left(\frac{2}{3}n\right) + c}$$

מהיות $T(n) \leq c + T\left(\frac{2}{3}n\right)$,
וממשפט האב מתקיים כי עבור $q = \frac{2}{3} \leq 1$ מתקיים $f(n) \in \Theta(\log(n))$, נסיק כי $T(n) \in O(\log(n))$,
וגם ראינו כי כאשר האיבר המינימלי בשורש, האלגוריתם יורד $\log(n)$ פעמים, לכן

$$\log(n) \leq T(n) \Rightarrow T(n) \in \Omega(\log(n))$$

לכן מתקיים $\boxed{T(n) = \log(n)}$

מ.ש.ל.ב.☺

(ג) **צ"ל:** יש לכל היותר $\frac{n}{2^{h-d}}$ איברים בגובה d
הוכחה:
נוכיח באינדוקציה על גובה העץ,
<u>בסיס:</u> $d=1$, נשים לב שגובה 1 ויש לכל היותר איבר אחד ו$1 \geq \frac{n}{2^h} = \frac{n}{2^{h-d}}$, כנדרש
<u>צעד:</u> נניח שהטענה נכונה לd ונוכיח ל$d+1$,
מהנחת האינדוקציה יש לכל היותר $\frac{n}{2^{h-d}}$ איברים בגובה d ,
לכל איבר יש לכל היותר 2 ילדים, לכן בגובה $d+1$ יש לכל היותר $\frac{n}{2^{h-(d+1)}} = 2 \cdot \frac{n}{2^{h-d}}$ איברים,
כלומר הטענה נכונה ל$d+1$, כנדרש

מ.ש.ל.ג.☺

2. **פתרון:**

(א) **צ"ל:** איפה נמצא איבר מינימלי, ופסאודו קוד
הוכחה:
<u>טענה:</u> האיבר המינימלי הוא תמיד נמצא עלה
נניח בשלילה שהאיבר המינימלי הוא לא נמצא באף עלה, נסמן מסלול אליו מהשורש ב$(v_1, \ldots, v_n)$,
מהיות v_n לא עלה, קיים לו בן שנסמן בv_{n+1} אזי מתקיים $v_{n+1} \leq v_n$,
אם $v_n = v_{n+1}$, נפעיל את אותו הטיעון על v_{n+1} ובסוף נגיע לעלה, ומשם נקבל סתירה לכך שהאיבר לא נמצא בעלה
אם $v_{n+1} < v_n$ נקבל סתירה למינימליות v_n,
לכן האיבר המינימלי תמיד יהיה בעלה וראינו שבערימה העלים הם ה$log\,(n)+1$ איברים האחרונים, לכן נוכל לעבוד עם האלגוריתם הבא

```
Min_tree(A):
    return min(A[n-log(n)-1:n])
```

ראינו בהרצאות הראשונות שיעילות min היא $\Theta\,(n)$ ואנחנו רצים על $\log\,(n)+1$ איברים,
לכן יעילות Min_tree היא $\boxed{\Theta\,(\log\,(n))}$

מ.ש.ל.א.☺

(ב) **צ"ל:** הצעה למבנה חדש שמקיים פונקציות בזמן הנדרש
הוכחה:
<u>הצעה:</u> מבנה חדש הוא ערימה עם משתנה שמכיל את האיבר המינימלי. עתה נראה זמני ריצה שנדרשו

i. Min : נשים לב שהפונקציה מחזירה את התכונה min ולכן זה $O\,(1)$, כנדרש
ii. Max : הפונקציה עושה בדיוק מה שעשתה פעם ולכן זה $O\,(1)$, כנדרש
iii. $Insert$: הפונקציה עושה בדיוק מה שעשתה פעם, עד כדי עדכון השדה מינימום להיות המינימלי בין האיבר החדש למינימלי, ולכן זה $O\,(\log\,(n)) + O\,(1) = O\,(\log\,(n))$, כנדרש
iv. $Delete$: הפונקציה עושה בדיוק מה שעשתה פעם, אם הפונקציה הוציאה את המינימלי, אז היא קוראת לMin_tree מהסעיף הקודם ושומרת את הערך בשדה, לכן זמן הריצה הוא $O\,(\log\,(n)) + O\,(\log\,(n)) = O\,(\log\,(n))$, כנדרש
v. Extract_Max : הפונקציה עושה בדיוק מה שעשתה פעם, אם הפונקציה הוציאה את המינימלי, אז היא קוראת לMin_tree מהסעיף הקודם ושומרת את הערך בשדה, לכן זמן הריצה הוא $O\,(\log\,(n)) + O\,(\log\,(n)) = O\,(\log\,(n))$, כנדרש
vi. Extract_Min : הפונקציה תחילה מוצאת את האיבר המינימלי בין העלים $O\,(\log\,(n))$, לאחר מכן הפונקציה קוראת ל$delete(Min)$, ו$delete$ היא ביעילות $O\,(\log\,(n))$, לכן זמן הריצה הוא $O\,(\log\,(n)) + O\,(\log\,(n)) = O\,(\log\,(n))$, כנדרש

מ.ש.ל.ב.☺

3. **צ"ל:** הצעה למבנה חדש שעובד טוב לחציון
הוכחה:
<u>הצעה:</u> מבנה חדש 2 ערימות, ערימת מקסימום, ערימת מינימום,

- חישוב חציון: אם יש $2n$ איברים, נחזיר את הממוצע של המקסימלי בערימת המקסימום והמינימלי בערימת המינימום. אם יש $2n+1$ איברים, נחזיר את האיבר המקסימלי אם יש יותר איברים בערימת המקסימום, או את האיבר המינימלי בערימת המינימלי. (חישוב זה נעשה ב$O\left(1\right)$)
- הוספה: נחשב את החציון, אם המספר גדול שווה החציון, נוסיף לערימת המינימום, אחרת לערימת המקסימום. אם הפרש האיברים בין הערימות גדול נוציא איבר המקסימלי מהערימת המקסימום או מינימלי מערימת המינימום מהערימה עם היותר האיברים ונוסיף אותה לערימה השנייה. זה $O\left(\log\left(n\right)\right)+O\left(\log\left(n\right)\right)+O\left(1\right)=O\left(\log\left(n\right)\right)$
- הוצאה: נחשב את החציון, נוציא איבר מהערימה המתאימה, ולאחר מכן אם הפרש האיברים בין הערימות גדול נוציא איבר המקסימלי מהערימת המקסימום או מינימלי מערימת המינימום מהערימה עם היותר האיברים ונוסיף אותה לערימה השנייה. זה $O\left(\log\left(n\right)\right)+O\left(\log\left(n\right)\right)+O\left(1\right)=O\left(\log\left(n\right)\right)$

מ.ש.ל.☺

4. **פתרון:**

(א) **צ"ל:** הוכחת נכונות אלגוריתם

הוכחה:

תחילה נשים לב שאיננו משנים את המבנה, ולכן מהיות העץ התחיל כעץ כמעט שלם הוא יישאר כעץ כמעט שלם.
עתה בשביל להוכיח שהאלגוריתם שומר על ערימה נראה כי לכל אב מתקיים $key\geq key\left(left\right)$ וגם $key\geq key\left(right\right)$,
נשים לב שהעץ בהתחלה, לפני עדכון הערך הוא ערימה, לכן לכל קודקוד מתקיים $keyA_i\geq key\left(left\right)$ וגם $keyA_i\geq key\left(right\right)$,
לאחר עדכון הערך, מתקיים $\text{new\_key}\geq keyA_i\geq\max\left\{key\left(right\right),key\left(left\right)\right\}$,
לאחר בקודקוד $A\left[i\right]$ מתקיימת התכונה הנדרשת,
עתה נשים לב שכל קודקוד שומר על התכונה כי $key\geq key\left(left\right)$ וגם $key\geq key\left(right\right)$, פרט אולי לאב של $A\left[i\right]$ כי יכול להיות שמתקיים $\text{new\_key}>key\left(parent\left(A\left[i\right]\right)\right)\geq A\left[i\right]$.
עתה נוכיח בשמורת לולאה כי כל הקודקודים של העץ מקיימות את התכונה פרט אולי לאב של $A\left[i\right]$,

- <u>בסיס:</u> ראינו כי לאחר עדכון הערך, קודקוד שומר על התכונה כי $key\geq key\left(left\right)$ וגם $key\geq key\left(right\right)$, פרט אולי לאב של $A\left[i\right]$
- <u>צעד:</u> נשים לב שמהנחה מתקיים שהאב של $key\left(parent\left(A\left[i\right]\right)\right)\geq key\left(parent\left(A\left[i\right]\right).child\left(notA\left[i\right]\right)\right)$, לכן, $\text{new\_key}>key\left(parent\left(A\left[i\right]\right)\right)$, נבצע החלפה ואז הקודקוד $parent\left(A\left[i\right]\right)$ יקיים את התכונה, וגם $\text{new\_key}>key\left(parent\left(A\left[i\right]\right)\right)\geq key\left(parent\left(A\left[i\right]\right).child\left(notA\left[i\right]\right)\right)$, ולכן גם $A\left[i\right]$ יקיים את התכונה, לכן האיבר היחיד שעלול לא לקיים את התכונה הוא האב של $A\left[i\right]$, לאחר העדכון, כנדרש

לאחר שמורת הלולאה, יתקיים: $A\left[i\right]$ הוא השורש ולכן אין לו אב שמפר את התכונה ולכן כל קודקוד מקיים את התכונה הנדרשת ולכן העץ הוא ערימה. אחרת $key\left(parent\left(A\left[i\right]\right)\right)\geq A\left[i\right]$, ולכן האב של $A\left[i\right]$ מקיים את התכונה, ולכן כל איבר מקיים את התכונה, ולכן העץ הוא ערימה, כנדרש

מ.ש.ל.א.☺

(ב) **צ"ל:** יעילות

הוכחה:

נשים לב שמתקיים במקרה הגרוע המערך ממוין ואז increase_key מבצע $\Theta\left(\log\left(n\right)\right)$ פעולות

$$T\left(n\right)=\Theta\left(1\right)+\sum_{i=1}^{n}\Theta\left(\log\left(i\right)\right)=\Theta\left(1+\sum_{i=1}^{n}\log\left(i\right)\right)=\Theta\left(\sum_{i=1}^{n}\log\left(i\right)\right)=$$
$$=\Theta\left(\log\left(\prod_{i=1}^{n}i\right)\right)=\Theta\left(\log\left(n!\right)\right)$$

נשים לב כי

$$\log\left(n!\right)\geq\log\left(\prod_{i=\frac{n}{2}}^{n}i\right)\geq\log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right)=\frac{n}{2}\cdot\log\left(\frac{n}{2}\right)=\frac{n}{2}\cdot\left(\log\left(n\right)-1\right)$$
$$\Rightarrow\log\left(n!\right)=\Omega\left(\frac{n}{2}\cdot\left(\log\left(n\right)-1\right)\right)=\Omega\left(n\log\left(n\right)\right)$$

וגם

$$\log(n!) \leq \log(n^n) = n \cdot \log(n) \Rightarrow \log(n!) = O(n\log(n))$$

לכן $\log(n!) = \Theta(n\log(n))$, נציב ונקבל

$$\boxed{T(n) = \Theta(\log(n!)) = \Theta(n\log(n))}$$

מ.ש.ל.ב.☺

5. **פתרון:**

(א) **צ"ל:** $\frac{3}{4}$ מהאיברים בגובה לפחות $\frac{\log n}{2}$

הוכחה:

נסמן $n = 2^h - 1 + t$, כאשר $1 \leq t \leq 2^h$, נשים לב כי $h + 1 \geq \log(n) \geq h$,

נשים לב שבגובה h יש 2^h איברים,

נשים לב שבגובה $h-1$ יש 2^{h-1} איברים,

נשים לב שבגובה $h+1$ יש t עלים, לכן כמות האיברים בגובה $h, h+1, h-1 \geq \left\lfloor \frac{\log(n)}{2} \right\rfloor$ הוא

$$\begin{aligned}\frac{\text{number of elements in last layers}}{n} &= \frac{2^{h-1}+2^h+t}{2^h-1+t} \\ &= \frac{2^h+2^{h-1}}{2^h-1+t} \geq \frac{2^h+2^{h-1}}{2^h-1+2^h} \\ &\frac{2^h+2^{h-1}}{2^{h+1}-1} \geq \frac{2^{h-1}+2^h}{2^{h+1}} \geq \frac{3}{4}\end{aligned}$$

לכן קיבלנו כי כמות האיברים בגובה $h, h+1, h-1$ הוא לפחות $\frac{3}{4}$ מכל האיברים, כנדרש

מ.ש.ל.א.☺

(ב) **צ"ל:** יעילות Ω מיון ערימות

הוכחה:

נשים לב כי בהתחלה בונים $MaxHeap$ וראינו שזה לוקח $\Theta(n)$,

לאחר מכן, ראינו כי לפחות $\frac{3}{4}n$ בגובה $\frac{\log(n)}{2}$,

לכן מהיות האחרונים הם הכי קטנים (הוראה בשאלה 1), כמות הפעולות שתבצע $heapify$ יהיה לפחות $\frac{\log(n)}{2}$ עבור כל איבר מה$\frac{3}{4}n$ האחרונים, לכן

$$\begin{aligned}&T(n) \geq \Theta(n) + \frac{3}{4}n \cdot \frac{\log n}{2} \Rightarrow T(n) = \Omega\left(\Theta(n) + \frac{3}{4}n \cdot \frac{\log n}{2}\right) = \Omega\left(\frac{3}{4}n \cdot \frac{\log n}{2}\right) = \Omega(n\log(n)) \\ &\Rightarrow \boxed{T(n) = \Omega(n\log(n))}\end{aligned}$$

מ.ש.ל.ב.☺

6. **צ"ל:** אלגוריתם למציאת 10 מקסימליים ביעילות $O(1)$ והוכחת נכונות

הוכחה:

```
def find10Max(tree):
    arr = []
    findAllPossible(tree, arr, 10)
    sort(arr)
    return arr[-10:]

def findAllPossible(tree, arr, n):
    if n == 0:
        return
    if tree == None:
        return
    arr.append(tree.value)
    findAllPossible(tree, arr.left, n-1)
    findAllPossible(tree, arr.right, n-1)
```

תחילה נוכיח נכונות,
נוכיח זאת ב2 שלבים, תחילה נוכיח נכונות $findAllPossible$, ששומרת במערך את כל האיברים של עץ עד הגובה הn,
נוכיח באינדוקציה על n,
בסיס: עבור $n=0$, מתקיים שאין איברים עד הגובה הn ולכן הוא לא יחזיר כלום כנדרש,
צעד: נניח שהטענה נכונה ל$n-1$ לכל עץ ונראה שהיא נכונה לn ,
נשים לב שכמות האיברים בגובה הn שווה לשורש ועוד $n-1$ בתת עץ ימני ועוד $n-1$ בתת העץ השמאלי,
מהנחת האינדוקציה, הקריאות הרקורסיביות מטפלות במקרים הללו $n-1$ בתת עץ ימני ועוד $n-1$ בתת העץ השמאלי,
ונשאר לטפל בשורש, ולכן הפונקציה מוסיפה אותו למערך.
עתה נוכיח את נכונות $find10Max$,
נשים לב ש$tree$ הוא ערימה ולכן $key \geq \max\{left.key, right.key\}$, לכן ה10 המקסימליים יהיו בגובה 10 לכל היותרמקיום תנאי זה לכל קודקוד
עתה, מנכונות $findAllPossible$, הוא מקבל מערך של כל האפשרויות ל10מקסימליים,
ממיין אותו, ואז המקסימליים הם האחרונים, ומחזיר את ה10 האחרונים שהם המקסימליים במערך
נשים לב ש$findAllPossible$ מבצע מספר סופי של פעולות לכל קודקוד שנסמנו בc, ועובר על לכל היותר 2^{11} קודקודים,
לכן $findAllPossible$ הוא ביעילות $O\left(c\cdot2^{11}\right)=O\left(1\right)$,
עתה $find10Max$, קורא ל$findAllPossible$ שמחזיר מערך גודל 2^{11} לכל היותר, לאחר מכן ממיין אותו שלוקח לכל היותר $c\cdot2^{11}\cdot\log\left(2^{11}\right)$ פעולות ולבסוף מחחזיר את ה10 האחרונים שלוקח $c\cdot10$ פעולות, לכן היעילות של האלגוריתם הוא

$$O\left(1\right)+O\left(c\cdot2^{11}\cdot\log\left(2^{11}\right)\right)+O\left(c\cdot10\right)=O\left(1\right)+O\left(1\right)+O\left(1\right)=O\left(1\right)$$

כנדרש

מ.ש.ל.☺