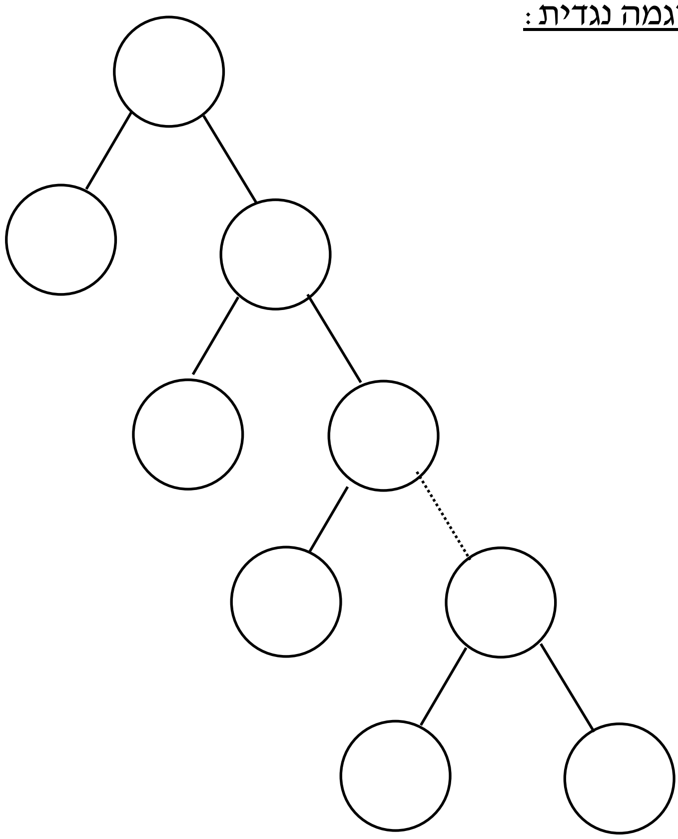
מבני נתונים – תרגיל 3

שאלה 1

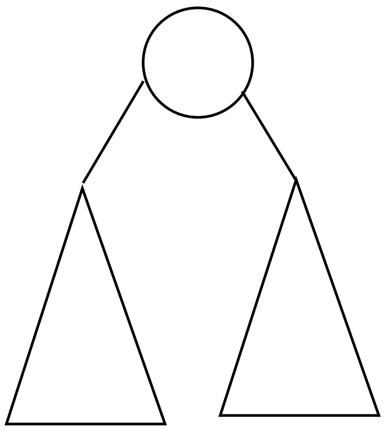
א. דוגמה נגדית:



ניתן לראות כי חוץ מהשורש כל 2 קודקודים מוסיפים 1 לגובה העץ, משמע גובה העץ ליניארי למספר הקודקודים. עבור n קודקודים גובה העץ הינו $\frac{n-1}{2}$ (n בהכרח אי זוגי) ולכן גובה העץ הינו $O(n)$ ולא $O(logn)$.

ב. הוכחה: נרצה ראשית להוכיח כי עבור כל תת עץ של עץ המקיים את התנאי הנתון מתקיים כי מספר הקודקודים בתת העץ השמאלי שלו שווה למספר הקודקודים בתת העץ הימני שלו.

יהי תת עץ של עץ המקיים את התנאי הנתון (אז הוא, תת העץ השמאלי ותת העץ הימני שלו מקיימים את הנתון):



נסמן את מספר הקודקודים של כל תת העץ הנ״ל ב 2^m-1, את מספר הקודקודים של תת העץ הימני 2^t-1 ושל תת העץ השמאלי ב 2^p-1. מספר הקודקודים של תת העץ הימני ועוד מספר הקודקודים של תת העץ השמאלי ועוד 1 עבור השורש שווה למספר הקודקודים של כל תת העץ הנ״ל. ולכן:

$$(2^t-1)+(2^p-1)+1=2^m-1$$

ומכאן ש: $2^t+2^p=2^m$ ולכן $2^t(1+2^{p-t})=2^m$ ולכן $1+2^{p-t}=2^{m-t}$.
קיבלנו כי 1 ועוד חזקה של 2 שווה לחזקה של 2 ולכן באופן ודאי המשוואה הנ״ל הינה $1+2^0=2^1$, ומכאן ש $p-t=0$ ולכן $p=t$. בנוסף ניתן לראות כי $m-t=1$ ומכאן ש $m=t+1=p+1$.

בסה״כ הוכחנו כי לכל עץ המקיים את התנאי בעל 2^k-1 קודקודים (העץ הינו תת עץ של עצמו ומכאן שגם מספר הקודקודים שלו הוא מהצורה הנתונה) מספר הקודקודים בתת העץ הימני שלו שווה למספר הקודקודים בתת העץ השמאלי שלו שווה ל $2^{k-1}-1$.

כעת נוכיח באינדוקציה כי לכל k טבעי, גובה כל עץ המקיים את התנאי ובעל $n=2^k-1$ קודקודים קטן שווה מ $\log(2^k-1)=\log n$.

<u>בסיס האינדוקציה:</u> *k=1* אז מספר הקודקודים בעץ כולו הינו $2^1-1=1$. גובה העץ הינו 0 הקטן שווה ל $\log(2^1-1)=log1=0$ ומכאן שבסיס האינדוקציה מתקיים.

<u>הנחת האינדוקציה:</u> יהי k טבעי. נניח כי גובה כל עץ בעל $2^{k-1}-1$ קודקודים המקיים את התנאי שגובהו קטן שווה מ $\log(2^{k-1}-1)$.

<u>צעד האינדוקציה:</u> יהי עץ המקיים את התנאי ובעל 2^k-1 קודקודים. צ״ל שגובהו קטן שווה מ $\log(2^k-1)$. נסתכל על השורש שלו:

מקרה 1: לשורש אין בנים. משמע העץ בעל קודקוד אחד, גובה העץ הינו 0 הקטן שווה ל $\log(2^1-1)=log1=0$ וסיימנו.

מקרה 2: לשורש בן אחד, אז באופן ברור תת העץ הימני של העץ כולו בעל יותר קודקודים מתת העץ השמאלי (או להפך) בסתירה למה שהוכחנו.

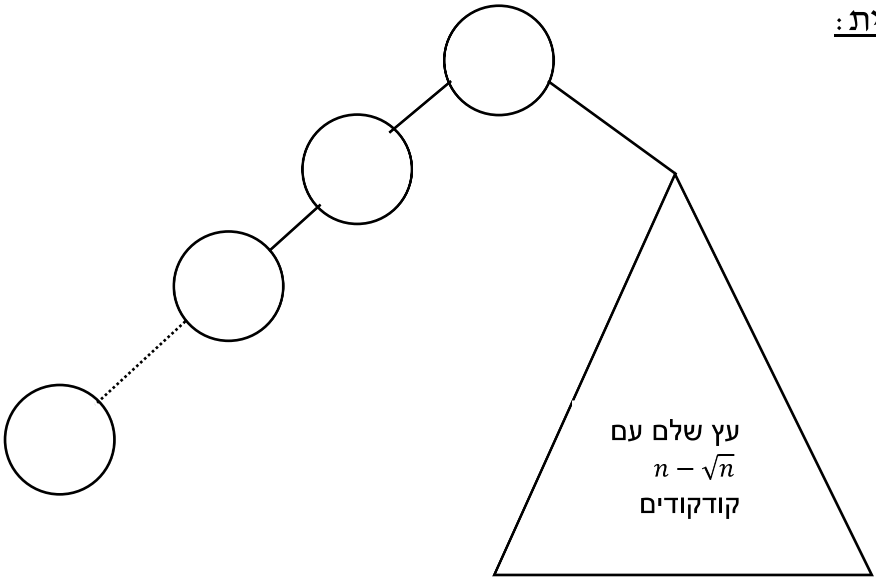
מקרה 3: לשורש שני בנים. לפי מה שהוכחנו מספר הקודקודים של שניהם שווה ל $2^{k-1}-1$. על פי הנחת האינדוקציה הגובה של שניהם קטן שווה מ $\log(2^{k-1}-1)$. מכאן ש גובה העץ כולו קטן שווה מ:

$$\log(2^{k-1}-1)+1=\log(2^{k-1}-1)+\log(2)=\log\left(2(2^{k-1}-1)\right)$$
$$=\log(2^k-2)\leq\log(2^k-1)$$

ובסה״כ ראינו כי גובה העץ הינו $O(\log(2^k-1))=O(logn)$,

כאשר $n=2^k-1$=מספר הקודקודים בעץ.

ג. <u>דוגמה נגדית</u>:



גובה תת העץ השמאלי הינו $\sqrt{\mathrm{n}}-1$ וגובה תת העץ הימני הינו $\log(n-\sqrt{n})$ ובסך הכל גובה כל העץ הינו $\sqrt{n}$ הגדול מ $\log(n)$. כל זאת בעוד שממוצע עומקי הצמתים הינו:

$$\leq \frac{0+[1+2+\cdots+\sqrt{n}]+(*)[(n-\sqrt{n})*\log(n-\sqrt{n})]}{n} = \frac{\frac{\sqrt{n}(1+\sqrt{n})}{2}+(n-\sqrt{n})*\log(n-\sqrt{n})}{n} \leq$$

$$\frac{\sqrt{n}(1+\sqrt{n})+(n-\sqrt{n})*\log(n)}{n} \leq \frac{\sqrt{n}((\#)\log(n)+\sqrt{n})+(n-\sqrt{n})*\log(n)}{n} = \frac{n+nlogn}{n} =$$

$$1+logn = O(logn)$$

בסה״כ הראינו עץ שממוצע העומק של קודקודיו הוא $O(logn)$ אך אינו מאוזן.

(*) העומק המקסימלי של עץ בינארי של בעל $n-\sqrt{n}$ קודקודים הוא $\log(n-\sqrt{n})$, לכן ממוצע העומקים בעץ הנ״ל קטן שווה מ $\log(n-\sqrt{n})$ ונכפיל במספר הקודקודים בעץ.

(#) עבור $n>1$

<u>שאלה 2</u>

א. נבנה שני AVL הממוינים בסדר In Order :

עץ R – כל קודקוד i מכיל שדה של r_i לפיו מתבצע המיון, ושדה של גודל תת העץ ששורשו הקודקוד i.

עץ B – כל קודקוד i מכיל שדה של b_i לפיו מתבצע המיון, ושדה של גודל תת העץ ששורשו הקודקוד i.

<u>הרעיון הכללי</u> : כאשר נדרש למצוא את מספר הקטעים שמכילים נקודה p כלשהי, נחשב את מספר הקודקודים בעץ R שה - r_i שלהם גדול ממש מ - p ועוד מספר הקודקודים בעץ B שה - b_i שלהם קטן ממש מ – p (שתי הקבוצות זרות משום שקטע לא יכול גם להתחיל אחרי נקודה וגם להסתיים לפני אותה הנקודה). את הסכום שקיבלנו נחסיר מ n והינה לנו מספר הקטעים שמכילים את p.

ראשית נפעיל פונקציה רקורסיבית על קודקודי העץ R אשר מתחילה בשורש. בכל קודקוד הפונקציה משווה את הערך p עם הערך r_i ופועלת כך :

- אם $r_i < p$ נפעיל את הפונקציה הרקורסיבית על הבן הימני – כל תת העץ השמאלי והקודקוד הנוכחי קטנים ממש בערכם מ p ויכולים לסמל קטע המכיל את p.
- אם $r_i = p$ נסכום את גודל תת העץ הימני ונפסיק את הפונקציה – כל הקודקודים בתת העץ הימני גדולים ממש בערכם מ p ושאר הקודקודים קטנים או שווים לו.
- אם $r_i > p$ נסכום את גודל תת העץ הימני פלוס אחד (עבור הקודקוד הנוכחי) ונפעיל את הפונקציה הרקורסיבית על הבן השמאלי - כל הקודקודים בתת העץ הימני והקודקוד הנוכחי גדולים ממש בערכם מ p ועלינו לבדוק את שאר הקודקודים.

*כאשר הגענו לעלה נפסיק את הפונקציה.

באופן סימטרי נפעיל פונקציה רקורסיבית על העץ B אשר סוכמת את מספר הקודקודים שה b_i שלהם קטן ממש מ p.

לבסוף נחסר מ n את סך הסכומים שצברנו בשתי הפונקציות וקיבלנו את מספר הקטעים שמכילים את נקודת השאילתה. מכיוון ששני העצים מאוזנים הגובה שלהם חסום ע"י $logn$, כל פונקציה מתוך השתיים שהפעלנו מבצעת מספר פעולות קבוע לכל היותר $logn$ פעמים כגובה העץ, ובסה"כ זמן הריצה של האלגוריתם הינו $\boldsymbol{O(logn)}$.

ניתן לראות כי זמן העיבוד המקדים הטוב ביותר שנוכל לקבל הינו זמן הריצה של בניית שני עצי AVL (עץ חיפוש בינארי מאוזן) שהוא $2O(nlogn) = \boldsymbol{O(nlogn)}$.

ב. 1. ראשית נחפש את הקודקוד בעל הגובה התחתון ביותר שנמצא בטווח (min) והקודקוד בעל הגובה העליון ביותר שנמצא בטווח (max). למציאת min נבצע חיפוש של הקצה התחתון של הטווח (y1) מהשורש. אם הגענו לקודקוד שערכו y1 או שאיננו יכולים להמשיך עוד בחיפוש סיימנו. אם הגענו לקודקוד שערכו קטן מy1, הקודקוד המבוקש הינו קודקוד אחד לפני. באופן דומה נחפש את max. בכל חיפוש כזה אנו עוברים על גובה העץ לכל היותר פעם אחת, בסך הכל עברנו על גובה העץ לכל היותר פעמיים ולכן החיפוש הנ"ל מתבצע ב $O(h)$. חשוב לציין כי את החיפוש הנ"ל נבצע במקביל, וכאשר "נפרדות דרכי החיפוש" נסמן את הקודקוד בו הן נפרדו כקודקוד x, שהוא האב הקדמון של min ו max. כעת, כאשר מצאנו את קצוות החיפוש שלנו ואת האב הקדמון שלהם נפעל באופן הבא:

a) <u>נבצע חיפוש של min מ x:</u> בכל קודקוד שנעבור בדרך (לא כולל x וכולל min) – אם הוא גדול שווה מ min נדפיס אותו ואת כל תת העץ הימני שלו. אם הוא קטן מ min לא נבצע דבר. b) <u>נדפיס את x.</u> c) <u>נבצע חיפוש של max מ x</u>: בכל קודקוד שנעבור בדרך (לא כולל x וכולל max) – אם הוא קטן שווה מ max נדפיס אותו ואת כל תת העץ השמאלי שלו. אם הוא גדול מ max לא נבצע דבר.

הפעולות הנ"ל מתבצעות ב O(h) (במקרה שעברנו על גובה כל העץ) וואו ב O(k) (במקרה שעברנו בדיוק על כל k הקודקודים). בסה"כ הדפסנו את כל הקודקודים שנמצאים בטווח ב **O(h+k)**.

2. נבנה עץ AVL ונסמנו T הממוין לפי גובה הקטעים. בנוסף, בדומה לסעיף א', כל קודקוד i ב T יקושר לשני עצי AVL אשר נסמנם R_i ו B_i:

R_i – עץ AVL המכיל את כל הקודקודים הנמצאים בתת העץ של T ששורשו הקודקוד ה i, וממוין לפי נקודת ההתחלה של הקטעים.

B_i – עץ AVL המכיל את כל הקודקודים הנמצאים בתת העץ של T ששורשו הקודקוד ה i, וממוין לפי נקודת הסיום של הקטעים.

<u>סיבוכיות זיכרון:</u> בעץ T n קודקודים שמלבד העצים המקושרים מכילים קבוע כלשהו = O(n). ברמה 0 מקושרים 2 עצים בעלי n קודקודים כל אחד המכילים קבוע כלשהו = O(2n)=O(n), ברמה 1 מקושרים 4 עצים בעלי $\frac{n-1}{2}$ קודקודים כל אחד המכילים קבוע כלשהו = $O\left(\frac{4(n-1)}{2}\right) = O(n)$, ברמה 2 מקושרים 8 עצים בעלי $\frac{n-3}{4}$ קודקודים כל אחד המכילים קבוע כלשהו = $O\left(\frac{8(n-3)}{4}\right) = O(n)$ וכו'. T הינו עץ AVL לכן מספר הרמות שבו (גובה העץ) = logn, וכל רמה תופסת O(n) מקום, מכאן שסיבוכיות הזיכרון הינה: $O(n) + logn * O(n) = \boldsymbol{O(nlogn)}$

כאשר נדרש למצוא את מספר הקטעים שנחתכים עם Q נחפש באופן זהה לסעיף ב.1 את הקודקודים בT אשר נמצאים בטווח בין הנקודה התחתונה לנקודה העליונה שלQ: נחפש את min, max ו x. כעת, כאשר מצאנו את קצוות החיפוש שלנו (מבחינת גובה) ואת האב הקדמון שלהם <u>נסכום מספר קטעים באופן הבא</u>:

a) <u>נבצע חיפוש של min מ x:</u> בכל קודקוד שנעבור בדרך (לא כולל x וכולל min) – אם הוא גדול שווה מ min והערך האופקי של Q נמצא בין נקודת ההתחלה לנקודת הסיום שלו נוסיף 1 לסכום. כל תת העץ הימני שלו תקין מבחינת הגובה, נפעיל עליו את האלגוריתם של סעיף א' (על עצי R_i ו B_i של השורש שלו) ונוסיף את הסכום

שקיבלנו לסכום שלנו. אם הוא קטן מ min לא נבצע דבר. b) <u>x :</u> אם הערך האופקי של Q נמצא בין נקודת ההתחלה לנקודת הסיום של x נוסיף 1 לסכום שלנו. c) <u>נבצע חיפוש של max מ x</u> : בכל קודקוד שנעבור בדרך (לא כולל x וכולל max) – אם הוא קטן שווה מ max והערך האופקי של Q נמצא בין נקודת ההתחלה לנקודת הסיום שלו נוסיף 1 לסכום שלנו. כל תת העץ השמאלי שלו תקין מבחינת הגובה, נפעיל עליו את האלגוריתם של סעיף א׳ (על עצי R_i ו B_i של השורש שלו) ונוסיף את הסכום שקיבלנו לסכום שלנו. אם הוא קטן מ min לא נבצע דבר. בסה״כ הסכום שסכמנו הינו מספר כל הקודקודים ב S שנחתכים עם Q.

<u>זמן ריצה :</u> אם h הינו הגובה שלT וT עץ AVL אז $h=logn$. בדומה לסעיף ב.1 למציאת min , max ו x לוקח $O(h)=O(logn)$. לאחר מכן בחיפוש של min מקודקוד x אנו עוברים לכל היותר ב logn קודקודים = O(logn), כשבכל אחד מהם מפעילים אלגוריתם שלפי סעיף א׳ לוקח O(logn). מכאן שהחיפוש הנ״ל לוקח O(logn)*O(logn)=$O(\log^2 n)$. באופן דומה בחיפוש של max מקודקוד x לוקח $O(\log^2 n)$. בסה״כ זמן הריצה של האלגוריתם הינו :

$$O(logn) + O(\log^2 n) + O(\log^2 n) = \boldsymbol{O(\log^2 n)}\ ,\boldsymbol{c = 2}$$

שאלה 3

מבנה הנתונים שלנו יהיו שני עצי AVL – D ו – G שכל קודקוד בהם שומר את העוקב והקודם שלו, וה location שלו. D עץ AVL שקודקודיו מייצגים את הגמדים וממוין על פי ה location. G עץ AVL שקודקודיו מייצגים את הענקים וממוין גם הוא על פי ה location.

<u>אתחול</u> נאתחל שני עצי AVL ריקים. ראינו בכיתה כי זה נעשה לכל היותר ב O(1).

<u>הכנסת גמד</u> – נכניס גמד חדש לעץ D ונעדכן מצביעים לעוקב ולקודם. ראינו בכיתה כי פעולת ה insert בעץ AVL נעשית לכל היותר ב O(logn) ופעולת עדכון המצביעים נעשית במספר פעולות קבוע ולכן לכל היותר ב O(1) ובסה"כ הפעולה נעשית ב O(logn)+O(1)=O(logn).

<u>הכנסת ענק</u> – נכניס ענק חדש לעץ G ונעדכן מצביעים לעוקב ולקודם. ראינו בכיתה כי פעולת ה insert בעץ AVL נעשית לכל היותר ב O(logn) ופעולת עדכון המצביעים נעשית במספר פעולות קבוע ולכן לכל היותר ב O(1) ובסה"כ הפעולה נעשית ב O(logn)+O(1)=O(logn).

<u>"האם מדברים"</u> – נחפש את L1 ו – L2 בעץ D. במידה ולפחות אחד מהם לא נמצא נחזיר הודעת שגיאה. ראינו בכיתה כי פעולת החיפוש בעץ AVL נעשית ב O(logn), בחיפוש שלנו היא נעשית פעמיים. נכניס את הגמד הנמצא במקום השמאלי מביניהם (נניח בה"כ L1) לעץ G. כאמור פעולה זו נעשית בלכל היותר O(logn). נתבונן בעוקב של L1 בעץ G, זהו הענק הקרוב ביותר ל L1 מימין ונסמן את מיקומו ב p. אם $P < L2$ אזי L1 אינו יכול לדבר עם L2 משום שיש ענק ביניהם, אחרת הם כן יכולים לדבר. ביצענו מספר פעולות קבועות הנעשות לכל היותר ב O(1). לבסוף לא נשכח למחוק את L1 מהעץ G. ראינו בכיתה כי גם פעולת המחיקה בעץ AVL נעשית לכל יותר ב O(logn). בסה"כ ביצענו פעולות הנעשות לכל היותר ב

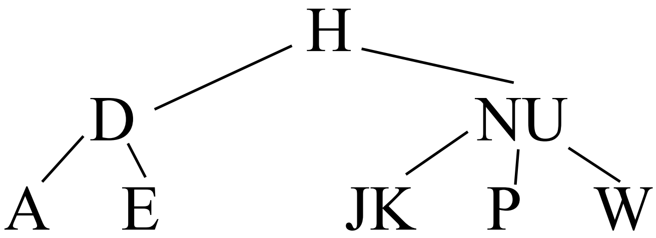
O(logn)+ O(logn)+ O(logn)+ O(1)+ O(logn)= O(logn)

<u>מחיקה</u> – נחפש האם קיים ענק בעץ G במיקום location. אם כן נמחק אותו ונעדכן את העוקב של הקודם שלו לעוקב שלו, ולהפך. אם לא נחפש האם קיים גמד בעץ D במיקום location. אם כן נמחק אותו ונעדכן את העוקב של הקודם שלו לעוקב שלו, ולהפך. אם לא נחזיר הודעת שגיאה. כאמור ראינו כי פעולות החיפוש והמחיקה בעץ AVL נעשות לכל היותר ב O(logn), ביצענו אותו מספר קבוע של פעמים ולכן כל פעולת המחיקה שלנו לקחה לכל היותר O(logn).

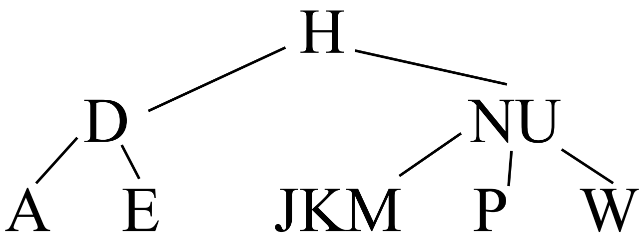
<u>"עם מי מדבר"</u> – נכניס איבר עם מיקום location לעץ G ונמצא את העוקב והקודם של האיבר שהוכנס, ונסמן את מיקומם ב r ו – l בהתאמה – אלו שני הענקים הקרובים ביותר לגמד משמאל ומימין (במידה ולא קיים עוקב נסמן את r באינסוף ו\או במידה ולא קיים קודם נסמן את l במינוס אינסוף). בדומה לסעיפים הקודמים זה נעשה ב O(logn). נחפש את הגמד עם המיקום location בעץ D – לכל היותר ב O(logn). נתקדם ונדפיס את הקודם של הגמד שמצאנו, נתקדם ונדפיס את הקודם גם של הגמד הזה וכו' עד שנגיע לגמד עם מיקום הקטן מ l או שלא נוכל להתקדם. באופן דומה נתקדם עם העוקבים עד שנגיע לגמד עם מיקום הגדול מ r או שלא נוכל להתקדם יותר. סה"כ עברנו בדיוק על k הגמדים הנמצאים בטווח ולכן זה נעשה לכל היותר ב O(k). לא נשכח למחוק את הגמד שהכנסנו לעץ G ב O(logn). בסה"כ בצענו מספר סופי של פעמים פעולות הלוקחות לכל היותר O(logn) ופעולה הלוקחת לכל היותר O(k) ולכן כל הפונקציה לקחה לכל היותר O(logn+k).

<u>שאלה 4:</u>

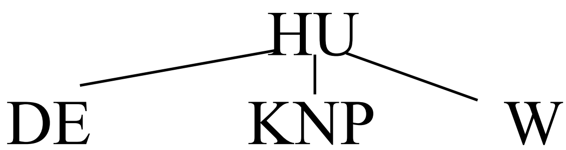
א. 1. לאחר הכנסת J :



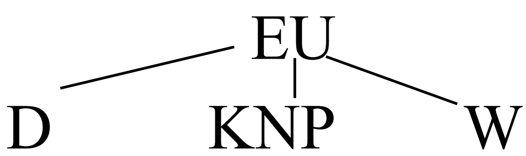
לאחר הכנסת M :



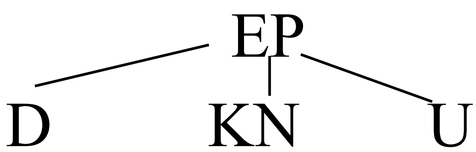
2. לאחר מחיקת A :



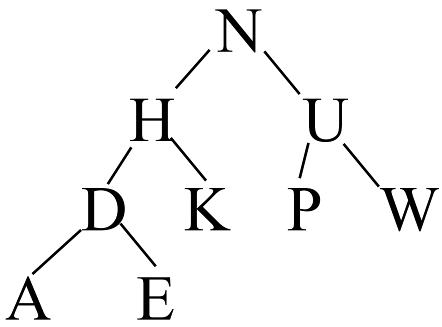
לאחר מחיקת H :



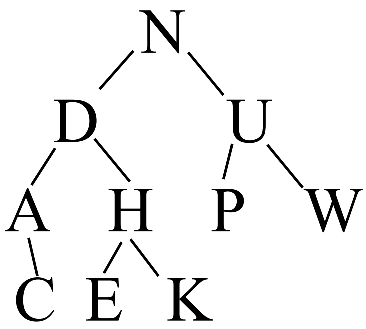
לאחר מחיקת W :



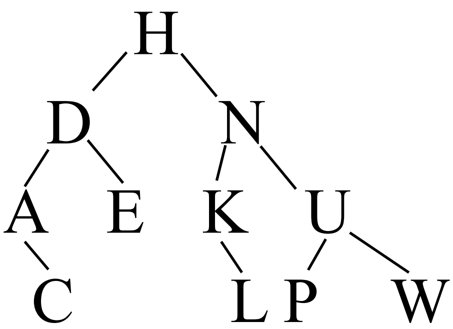
עץ AVL :



לאחר הכנסת C :



לאחר הכנסת L :



שאלה 5:

א. נבצע את פעולת החיפוש כפי שלמדנו בכיתה, רק שבכל קודקוד (שהוא בעצם מערך) נבצע חיפוש בינארי ולא ליניארי. כלומר, נבדוק האם הערך אותו אנו מחפשים נמצא באינדקס האמצעי שבשורש. במידה וכן פשוט נחזירו, במידה והוא גדול ממנו נבצע את אותה הפעולה על אמצע תת המערך שמתחיל מימין לאינדקס הנוכחי ונגמר בסוף המערך. במידה והוא קטן ממנו נבצע את אותה הפעולה באופן סימטרי על תת המערך השמאלי. נבצע זאת בכל קודקוד עד שנגיע לתת מערך בגודל 1. במידה והערך אותו אנו מחפשים גדול ממנו נמשיך לבן הימני, ובמידה וקטן ממנו נמשיך לבן השמאלי. אנו מבצעים לכל היותר מעבר 1 על גובה כל העץ, ובכל רמה יש לכל היותר 2t-1 איברים ואנו מבצעים בה חיפוש בינארי (הראינו הכיתה כי חיפוש בינארי על מערך גודל n הוא לכל היותר O(logn)) – ולכן כל החיפוש שלנו יתבצע לכל היותר ב O(h*log(2t-1))=O(logt*h).

ב. נוסיף לכל קודקוד שדה size המציין את מספר המפתחות בתת העץ אשר מתחיל באותו הקודקוד, ונוסיף למבנה שדה counter שבעזרתו נסכום גדלים של תתי עצים. נתחיל בשורש, ובכל קודקוד אליו נגיע נסרוק את כל המפתחות שלו החל מהשמאלי. אם גודל תת העץ השמאלי של המפתח בו אנו נמצאים ועוד counter קטן מ k בדיוק ב 1 משמע שאנו נמצא במפתח ה k-י. אם הוא קטן ממנו ביותר מ 1 נוסיף ל counter את ה size של תת העץ השמאלי ועוד אחד ונתקדם למפתח הבא. אם גודל תת העץ השמאלי של המפתח בו אנו נמצאים ועוד counter גדול או שווה ל k, משמע שהמפתח ה k-י נמצא בתת העץ השמאלי ונרד אל הבן השמאלי של המפתח בו אנו נמצאים, הלא הוא השורש של תת העץ השמאלי. אם הגענו בקודקוד מסוים למפתח האחרון שאין עוד מפתחות מימינו ועלינו להמשיך, נרד אל הבן הימני שלו ונמשיך בדיוק באותה הסריקה מהמפתח השמאלי שלו.

נשים לב שבזמן החיפוש אנו עוברים על גובה המבנה לכל היותר פעם אחת (לאחר שירדנו רמה לא נחזור שנית לרמה מעל), ובכל רמה אנו עוברים לכל היותר פעם אחת על כל המפתחות של קודקוד מסוים שמספרם חסום ע"י 2t-1. כלומר, זמן השאילתה הינו לכל היותר O(h*(2t-1))=O(t*h).