תרגיל 6- מבני נתונים 67109:

שם: דן קורנפלד

חלק 1:

שאלה 1:

1. סעיף א:

מהגדרת עץ בינארי באופן רקורסיבי: עץ בינארי הוא עץ ריק (ללא צמתים), או עץ המורכב משורש ושני תתי-עצים בינאריים (ימני ושמאלי), לכן עבור כל עץ בינארי, כל תת עץ שלו מההגדרה יהיה עץ בינארי בעצמו.

נוכיח שכל תת עץ בינארי הוא כמעט שלם:

נגדיר את גובה העץ המקורי להיות h, ואת גובה התת עץ להיות h', כך ש $\Delta \geqslant 0, h' = h + \Delta$ (Δההפרש בין הגבהים).

נניח בשלילה שהתת עץ לא כמעט שלם, כלומר קיימת קומה $d < h'$ כך שהיא לא שלמה או שבקומה האחרונה העלים לא מיושרים לשמאל.

עבור קומה לא שלמה (שהיא לא האחרונה), קיים בגובה d פחות מ 2^d קודקודים, כלומר בעץ הכמעט מלא המקורי קיימת קומה שאינה מלאה (ואינה אחרונה). הסבר: נסמן $\Delta \geqslant 0, h' = h + \Delta$, הקומה $d + \Delta < h$ לא מלאה), לכן העץ לא כמעט מלא.

במידה וחסר קודקוד בקומה הh' שמתאימה לקומה הh (כלומר בקומת העלים- הקומה האחרונה), כך שחסר קודקוד והתת עץ לא מיושר שמאלה- בסתירה להנחה שהעץ המקורי הוא עץ כמעט מלא לכן המצב לא הגיוני, סתירה.

לכן עבור כל קודקוד בעץ בינארי כמעט שלם, כל תת עץ הוא גם עץ בינארי כמעט שלם.

2. סעיף ב:

נוכיח שלעץ בינארי יש לכל היותר 2^k קודקודים מעומק k באינדוקציה:

בסיס: עבור עץ בינארי בגובה 0 (עלה) אכן יש $2^0 = 1$ קודקודים

שלב: נניח את נכונות הטענה עבור $k-1$ ונוכיח עבור k.

צעד: לכל קודקוד יש לכל היותר 2 בנים. לכן, עבור הקומה ה $k-1$, יש לה לכל היותר 2^{k-1} קודקודים, ובמידה ונסתכל על המספר המקסימלי נניח שלכל קודקוד יש 2 בנים, כלומר $2 \cdot 2^{k-1} = 2^k$, כלומר לעץ בינארי יש לכל היותר 2^k קודקודים מעומק k.

לכן, מכיון שאנחנו מדברים על עץ בינארי כמעט מלא, אם העץ מלא יש 2^k קודקודים ואם לא, אז יש פחות מ2^k קודקודים, ולכן יש לכל היותר 2^k קודקודים מעומק k.

3. סעיף ג:

עבור עץ בינארי כמעט שלם מגודל n ומגובה h, נוכיח שמתקיים $2^h \leqslant n \leqslant 2^{h+1}-1$

עבור עץ בינארי כמעט שלם מגובה h, כל הקומות עד ל$h-1$ מלאות, ועבור הקומה הh או שהיא כמעט מלאה או שהיא מלאה.

נוכיח באינדוקציה שמספר הקודקודים של עץ בינארי מלא עד קומה h כלשהי הוא $2^{h+1}-1 = 2\cdot 2^h - 1$.

בסיס: עבור עץ בגובה 0 מתקיים ש$2\cdot 2^0 - 1 = 1$

שלב: נניח את נכונות הטענה עבור $h-1$ ונוכיח עבור h.

עבור מספר הקודקודים נחבר את מס הקודקודים עד הקומה $h-1$ כולל (כלומר $2\cdot 2^{h-1}-1$) ועוד הקומה ה h, כלומר 2^h לכן $2^h + 2\cdot 2^{h-1} - 1 = 2^h + 2^h - 1 = 2^{h+1}-1$

לכן: כל הקודקודים עד הקומה ה $h-1$ (כולל) הוא $2^h - 1$, אך מכיוון שהעץ בגובה h חייב להיות לפחות קודקוד 1 בגובה h כלומר העץ מכיל לכל הפחות 2^h קודקודים. במידה והעץ מלא בכל הקומה ה-h, העץ יכיל לפי הנוסחה $2^{h+1}-1$ קודקודים, לכן מתקיים שעבור עץ בינארי כמעט שלם מגודל n ומגובה h, מס הקודקודים הוא: $2^h \leqslant n \leqslant 2^{h+1}-1$

נוכיח שגובה העץ הוא בין $log_2 n - 1$ **לבין** $log_2 n + 1$

מהאי-שוויון שהוכחנו מתקיים: $2^h \leqslant n \leqslant 2^{h+1}-1$ נחלק את האי שוויון ל-2:

$$2^h \leqslant n \overset{log_2}{\Longrightarrow} h \leqslant log_2(n)$$

$$n \leqslant 2^{h+1}-1 \Longrightarrow n+1 \leqslant 2^{h+1} \overset{log_2}{\Longrightarrow} log(n+1) \leqslant h+1 \Longrightarrow log(n+1)-1 \leqslant h$$

$$log(n)-1 \leqslant log(n+1)-1 \leqslant h \leqslant log_2(n) \leqslant log_2(n)+1$$

כמבוקש $log(n)-1 \leqslant h \leqslant log(n)+1$

4. סעיף ד:

נסתכל על מספר אפשרויות לצורת העץ, לאחר שנאבחן את הצורה האופטימלית לכך שגודל תת העץ יהיה הגדול ביותר ביחס לכל העלים נבחן את גודלו.

(a) עבור עץ שלם כלשהו, גודל 2 תתי העצים יהיו $\left\lfloor \frac{n-1}{2} \right\rfloor$ כלומר מלבד השורש עצמו, גודל תתי העצים יהיה שווה.

(b) עבור עץ שלם עד לקומה ה-$h-1$, ולאחר מכן מוסיפים לתת העץ השמאלי קודקודים בגובה h עד שהתת עץ השמאלי מלא, והתת עץ הימני לא מכיל כלל איברים בגובה ה-h, כלומר היחס מהתת סעיף הקודם משתנה כך שבתת עץ השמאלי יש יותר קודקודים מהתת עץ הימני, עד שהתת עץ השמאלי בגובה ה h מלא.

(c) עבור עץ שלם עד לקומה ה-$h-1$, כמו כן התת עץ השמאלי מלא ומוסיפים קודקודים לתת עץ הימני, במצב זה היחס בין הקודקודים יורד עד ליחס שווה בין תתי הענפים, כאשר העץ מלא (כמו (a))

לכן המצב בו תת העץ השמאלי מלא בכל קודקודיו בגובה h וכאשר לתת העץ הימני אין כלל קודקודים בגובה h זהו המצב שבו היחס יהיה הגדול ביותר.

בגובה $h-1$ יהיו סה"כ לפי ההוכחה בתחילת השאלה $2^h - 1$ קודקודים, כך כ-חצי מהם בתת עץ

הימני וכ-חצי מהם בתת עץ השמאלי. בנוסף, הוספת הקודקודים בגובה ה h עבור תת העץ השמאלי בלבד הוא כמו הוספת כל הקודקודים עבור עץ בגובה $h-1$ כלומר 2^{h-1} קודקודים.

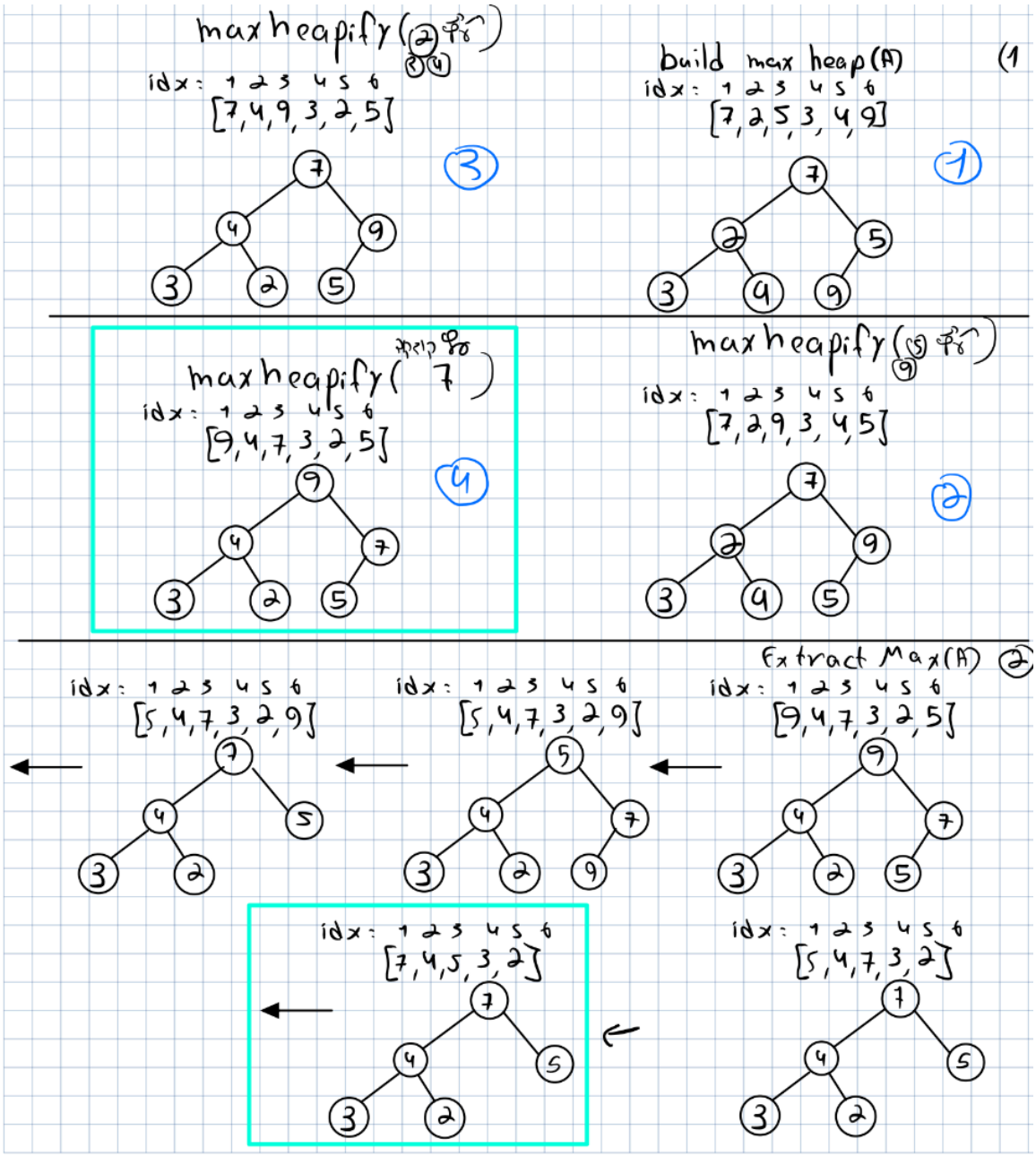
סיכום ביניים: בכל תת עץ עד גובה $h-1$ יש $\left\lfloor 2^{h-1} \right\rfloor = \left\lfloor \frac{2^h-1}{2} \right\rfloor$ קודקודים, ובגובה h באגף השמאלי יש 2^{h-1} קודקודים. כלומר סה"כ הקודקודים $n = 3 \cdot \left\lfloor 2^{h-1} \right\rfloor$ ובתת עץ השמאלי כולו יש $2 \cdot \left\lfloor 2^{h-1} \right\rfloor$ כלומר גודל התת עץ הוא לכל היותר $\left\lfloor \frac{2n}{3} \right\rfloor$ מסה"כ הקודקודים בעץ. ■

הערה: עבור כל תת עץ אחר, מספר הקודקודים יהיה קטן יותר מהתת-עץ השמאלי במצב המתואר לעיל, לכן הטענה מתקיימת.

שאלה 2:

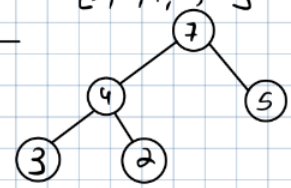
- $Max - heapify(A, i)$
- bool is_changed = true
-
- while (is_changed)
- do:
    - largest = -1 /identify largest
    - l←left(i)
    - r←right(i)
    - if l⩽heapsize(A) and a[l] >A[i]
        - then largest ←l
        - else largest ←i
    - if r⩽heapsize(A) and a[r] >A[largest]
        - then largest←r
    - if largest ≠ i
        - then Exchange (A[i],A[largest])
        - i=largest
    - else
        - break while;
- נוכיח את נכונות פתרון הקוד בצורה איטרטיבית עבור האלגוריתם באמצעות שמורת לולאה:
- שמורת הלולאה: בתחילת האיטרציה ה-j, בערימה המושרשת ב-i_1 הזוגות היחידים של אב קדמון וצאצא שיכולים להפר את תכונת הערימה הם, i_j וצאצאיו, כאשר i_j הוא התא עליו מבצעים את ההשוואות באיטרציה ה-j.
- אתחול: לפני תחילת האיטרציה הראשונה, הזוגות היחידים שיכולים להפר את הערימה הם i_1 וצאצאיו מהנחות הטענה.
- שימור: נניח שבתחילת האיטרציה ה-j החלפנו $j-1$ זוגות שהפרו את האיזון בערימה, נבחן אם הזוג הi_j האם ניתן לבצע בניהם החלפה, במידה וכן נבצע ונעבור לזוג הבא, במידה ולא הערימה מאוזנת כרצוי ונסיים את הריצה. בתהליך הבדיקה של האיטרציה מול הזוג ה i_j אנו בודקים מי הבן הגדול מבין הבנים ובמידה והבן הגדול יותר, גדול מההורה שלו, נבצע את ההחלפה. במצב זה, התת עץ שהשורש שלו הוא הבן עם הערך הקטן מבין 2 הבנים לא משתנה כלל, לכן לא יווצרו בעיות באיזון העץ בעקבות הבדיקה והשינוי.
- סיום: בסוף האיטרציה האחרונה, או שהגענו למצב בו לא התבצע החלפה בין זוג איברים, הגענו לערימה מאוזנת כרצוי, כאשר largest=i

שאלה 3:

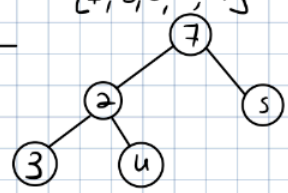

build max heap(A)
(1
idx: 1 2 3 4 5 6
[7,2,5,3,4,9]
1
max heapify(2 פיר)
idx: 1 2 3 4 5 6
[7,4,9,3,2,5]
3
max heapify(9 פיר)
idx: 1 2 3 4 5 6
[7,2,9,3,4,5]
2
max heapify(7 קודקוד)
idx: 1 2 3 4 5 6
[9,4,7,3,2,5]
4
Extract Max(A) (2
idx: 1 2 3 4 5 6
[9,4,7,3,2,5]
idx: 1 2 3 4 5 6
[5,4,7,3,2,9]
idx: 1 2 3 4 5 6
[5,4,7,3,2,9]
idx: 1 2 3 4 5 6
[5,4,7,3,2]
idx: 1 2 3 4 5 6
[7,4,5,3,2]

Delete(A, P(4))

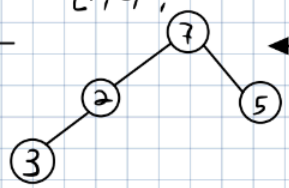
idx: 1 2 3 4 5 6
[5, 4, 7, 3, 2]



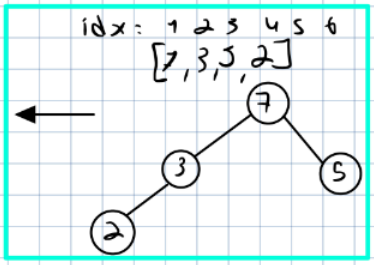
idx: 1 2 3 4 5 6
[7, 2, 5, 3, 4]



idx: 1 2 3 4 5 6
[7, 2, 5, 3]



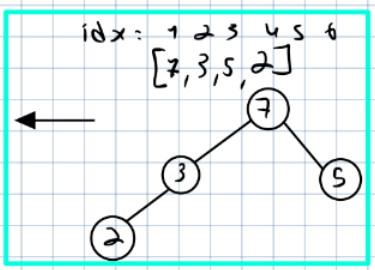
idx: 1 2 3 4 5 6
[7, 3, 5, 2]



return

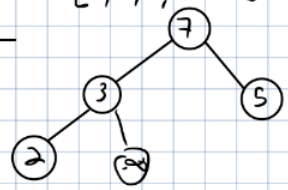
← max(A) = 7

idx: 1 2 3 4 5 6
[7, 3, 5, 2]

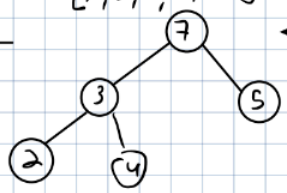


insert(A, 4)

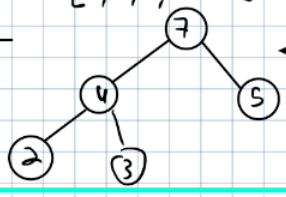
idx: 1 2 3 4 5 6
[7, 3, 5, 2, -∞]



idx: 1 2 3 4 5 6
[7, 3, 5, 2, 4]



idx: 1 2 3 4 5
[7, 4, 5, 2, 3]



תרגיל 6- מבני נתונים 67109:

שם: דן קורנפלד

חלק 2:

שאלה 4:

❖ סעיף א:

ראשית מכיוון שאנחנו מסתכלים על ערימה מקסימלית, אם נסתכל על עץ בגובה שגדול מ-10 נוכל לראות שבהכרח הקודקוד ה-10 בגודלו נמצא ב10 הקומות הראשונות לכל היותר (כלומא אם נסתכל על הקומה ה-11, בהכרח יש 10 קודקודים מעליו [ההורה, וההורה של ההורה] שגדולים מאותו קודקוד). מכיוון שאנחנו יודעים מהו המספר המקסימלי של קודקודים בתחום של ה-10 קומות הראשונות (הוכחנו שהוא לכל היותר $2^{11}-1$ קודקודים) וזהו חסם קבוע, נוכל לעבור על כל הקודקודים, להעתיק אותם למערך, למיין את המערך ולהחזיר את המספר באינדקס העשירי במערך הממוין. כל פעולה תרוץ בזמן קבוע ולכן האלגוריתם יהיה בסיבוכיות זמן של $O(1)$.

❖ סעיף ב:

נכתוב אלגוריתם המדפיס את k המספרים הגדולים ביותר בערימה. על זמן הריצה של האלגוריתם להיות $O(klog(k))$. לאורך כל האלגוריתם, במידה והערימה גדולה מגובה k, נסתכל על האיברים עד הקומה ה-k, במידה והיא קטנה יותר, נסתכל על הערימה בצורתה הרגילה (דבר שיאפשר לא לבצע פעולות מיותרות למטרת האלגוריתם).

(a) נשתמש בערימה נוספת T, ונכניס את השורש של הערימה הנתונה (H) ל-T.

i. מבנה הנתונים יכיל : $\{value, ptr\_left\_children, ptr\_right\_children\}$ כלומר את הערך של הקודקוד שהכנסנו, ופוינטרים ל-2 ילדיו, במידה ואין לו ילד באותו צד, נשים null כערך דיפולטיבי

(b) כל איבר שנכנס ל T, ישמור בנוסף לערך הצומת, גם 2 פוינטרים לילדים שלו ב H

(c) נוציא ונדפיס את השורש של T, ונוסיף ל- T את 2 הבנים שהשורש הצביע עליהם, נבצע את הפעולות הללו k פעמים.

אכן ניתן לגשת לצאצאים של כל קודקוד ב $O(1)$, וכל פעולת הוצאה מ-T אכן מדפיסה את האיבר ה-k הכי גדול, כלומר צריך לבצע k פעמים את b,c.

נתחו את זמן הריצה:

ביצוע: הכנסת קודקוד לערימה, שמירת 2 בניו כפוינטרים, הוצאת השורש של T והדפסתו, הוספת 2 בניו לערימה בסיבוכיות זמן ריצה של $O(log(k))$, ניתן להסתכל על סיבוכיות זמן זו, מכיוון שאנחנו לא מסתכלים על כל הערימה, אלא על תת ערימה עד גובה ה-k, כדי לצמצם ביצוע פעולות מיותרות. נבצע את הפעולה הזו k פעמים, $k \cdot O(log(k)) = O(k \cdot log(k))$

נתחו את המקום שהאלגוריתם דורש:

אנחנו מקצים באופן זמני ערימה חדשה, שעבור כל איבר שאנחנו מוציאים, אנחנו מכניסים 2, אנחנו מכניסים k איברים, לכן נכניס לאורך האלגוריתם ערימה בגודל של כ-k איברים (מכיוון שאנחנו בכל פעם מוציאים אחד ומכניסים 2, במצב המירבי יהיה כ-k איברים בערימה), כלומר $O(n)$ (פתרון זה חסכוני ביחס לפתרון השני אם היינו מקצים תת ערימה בגובה k שיכילו את כל התחלת המערך, היינו צריכים להקצאת כמערך בסדר גודל של כ-$2^{k+1}-1$ תאים.)

הוכיחו את נכונותו:

תחילה מכיוון שאנחנו מסתכלים על ערימה מקסימלית, האיבר המקסימלי נמצא בשורש. עבור כל תת עץ בערימה, האיבר המקסימלי נמצא בשורש העץ, לכן לכל תת עץ שנתייחס אליו, נוכל להתייחס ישר שהשורש של אותו תת עץ הוא המקסימום של אותו תת עץ.

לכן, לאחר הכנסת האיבר המקסימלי הראשון (השורש של העץ) לערימה השניה, הוצאתו והדפסתו, האיבר השני בגודלו, יהיה אחד מבניו, לכן לאחר שנדפיס את האיבר המקסימלי הראשון, נכניס את 2 בניו לערימה שאנחנו מקצים, ונסדר אותה לפי ערימת מקסימום תקינה, כך נקבל שהשורש הערימה החדשה מכיל את האיבר השני בגודלו. לאחר הוצאתו והדפסתו יש 2 אופציות: האיבר ה-3 הגדול ביותר הוא האיבר שכבר נמצא בערימה, או אחד מהבנים של האיבר שהודפס בהדפסה השניה, לכן נוסיף את 2 הבנים שלו לערימה, נבצע את הסידור וכך חוזר שוב ושוב עד האיבר ה-k. מסיבה זו כל פעם מסתכלים על האיברים האפשריים הגדולים ביותר, מוסיפים אותם לערימה החדשה, מסדרים את הערימה לערימת מקסימום תקינה כך ששורש הערימה החדשה יכיל את האיבר הבא הגדול ביותר שלא הודפס.

כתבו פסאודו קוד עבור האלגוריתם

```
❖ print_k_max(A,k)
    ➢ count =1 //the counter of the k max
    ➢ T // T is a new heap
    ➢ T.insert(root(A))
    ➢ while(k⩾count)
    ➢ do
        ■ cur = T.extract_max()
        ■ if (cur !=null)
            ● print(cur)
            ● T.insert(cur.left())
            ● T.insert(cur.right())
            ● count←count+1
```

תרגיל 6- מבני נתונים 67109:

שם: דן קורנפלד תז: 314915703

חלק 3:

שאלה 5:

סעיף א:

1. נשאיר את האיברים במערך ((O(1, ועבור כל זימון של הפונקציה נעבור לאינדקס i, ונסרוק את התת מערך עד האינדקס j, נאתחל משתנה מינימום כלשהו לערך של האינדקס הראשון ובכל פעם שנמצא איבר קטן יותר מהמינימום הנוכחי, נשנה את המינימום הנוכחי להיות האיבר, כך בסוף האיטרציה ה-j, נמצא את האיבר הקטן בתת מערך באופן לינארי.

סעיף ב בעמוד הבא

סעיף ב:

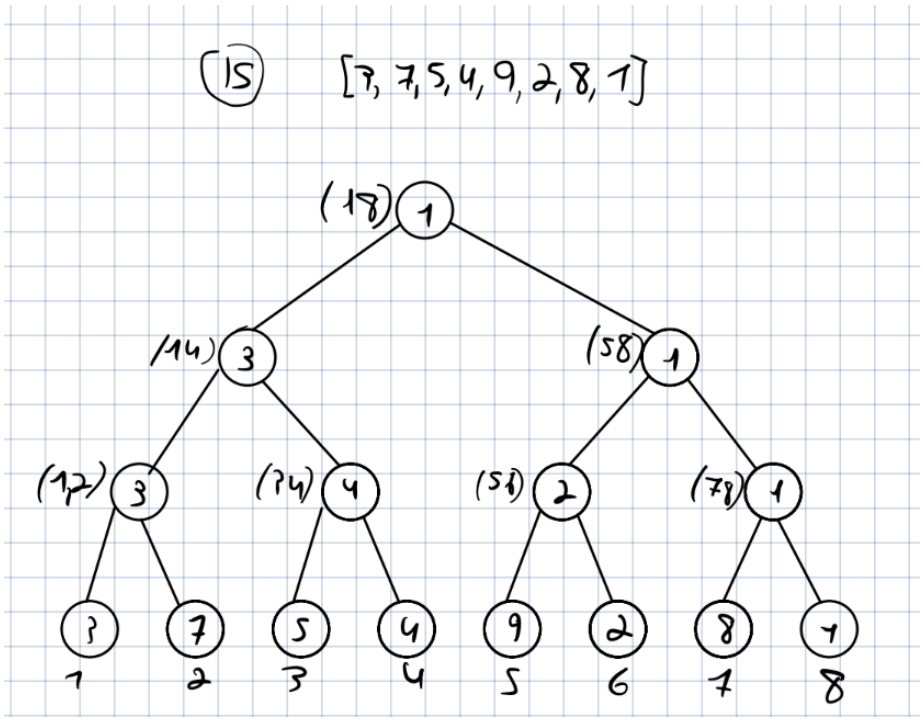


Figure 1: כדי להמחיש את ההסבר, הוספתי דוגמה

1. תיאור ראשוני: נבנה ערימת מינימום כך שהעלים הם האיברים הנמצאים במערך הנתון, וכל צומת מהווה המינימום בין הצאצים שלה.
2. בניית מבנה הנתונים: ניקח את המערך הנתון, ונבנה heap, כך שכל העלים הם האיברים (★ -הערה בהמשך) הנמצאים במערך הנתון, לצורך כך נתון שגודל המערך הנתון הוא חזקה של 2, לכן העץ יהיה עץ בינארי מלא, ונקצה מערך בגודל $2n-1$ לטובת הערימה. נעבור בצורה דומה על כל תא בערימה החדשה לאחר השמת העלים (בדומה לאופן בניית הערימה שנלמדה בהרצאה), רק שכאן מכיוון שאין נתונים, כל תא יושם להיות המינימום בין 2 הבנים שלו. מכיוון שבניית המערך משתמשת באותו אלגוריתם הנלמד, בניית המערך בסיבוכיות זמן ריצה של $O(n)$ כנלמד.

★ -בנוסף לערכים עצמם, נשמור 2 משתנים נוספים לכל איבר במערך, את הטווח בו האיבר הוא אכן האיבר המינימלי, כלומר: עבור עלה כלשהו, הוא יהיה בטווח שלו בלבד אז הערכים הנוספים יהיו $low = i,\ high = i$ כאשר מדברים על האינדקס i כלשהו מבין העלים. עבור צומת אחרת במערך, הערכים יכילו את הטווח בו אותו איבר הוא המינימום, לדוגמה: אם נסתכל על הדוגמה לעיל הבן השמאלי של השורש יכיל את האיבר 3, ואת הערכים 1 ו4 כלומר: 3 הוא הערך המינימלי עבור תת המערך 1-4.

המשך בעמוד הבא

3. אלגוריתם למציאת המינימום: עבור ערך תחתון וערך עליון נתונים (הטווח) נתחיל משורש הערימה המינימלית ונבדוק: האם הטווח שהשורש מוגדר מוכל בטווח הנתון? נסתכל על 3 אופציות לאורך האלגוריתם:
   (a) אם הטווח מוכל לחלוטין בטווח נתון: החזר את ערך הצומת
   (b) אם הטווח הנוכחי כלל לא נמצא בטווח הנתון: החזר את הערך הדיפולטיבי "אינסוף"
   (c) אם הטווח הנוכחי מוכל חלקית בטווח הנתון: זמן מחדש את הפונקציה עם 2 הבנים של הצומת הנוכחית.

   בכל שלב בו נקרא הפונקציה מחדש (רקורסיה) הצומת הנוכחית תחכה לערך שיוחזר מהבנים שלה, ותחזיר כלפי מעלה את הערך המינימלי מבניהם.
   האלגוריתם למעשה מחלק את הטווח הנתון לתתי-טווחים, כך בכל פעם לוקח את הערך המינימלי בכל תת טווח כך שלבסוף כל תתי הטווחים נבחנים.
4. בנוסף להסבר, אסביר כדוגמה מוחשית עבור הטווח 4,5 בדוגמה הנ"ל.
   תחילה נבדוק אם הטווח 1,8 מוכל בטווח 4,5 והתשובה היא חלחקית, לכן נבצע את (c) ונבצע זימון רקורסיבי ל-2 הבנים: 3 (הבן השמאלי), 1 (הבן הימני).
   עבור 1, הטווח 5,8 מוכל ב4,5? חלקית לכן נבצע זימון רקורסיבי ל-2 הבנים שלו 1 (הבן הימני) שלאחר מכן לא בטווח ויחזיר אינסוף, ו5,6 שבטווח חלקית לכן יזומן שוב ל-2 הבנים 2 ו9, 2 מחוץ לטווח לכן גם יחזיר אינסוף ו9 נמצא כולו בטווח לכן יחזיר את הערך 2 (9 נמצא באינדקס 5,5). ערכי ההחזרה יגיעו לצומת אב שאחראי על הטווח 5,6 והמינימום יהיה 9, לאחר מכן יגיע לצומת 5,8 ויחזיר את המינימום בין 9 לאינסוף (9) יעלה לשורש העץ ואותו תהליך נבצע עבור התת עץ השמאלי, כך לבסוף יוחזר בתת עץ 4, והשורש יבחר את המינימום מבין 9 ל-4, כלומר 4 ויחזיר מהפעולה את הערך 4 (אכן הוא המינימום בתת המערך של האינדקסים 4,5).
5. ניתוח זמן ריצה: עבור בניית הערימה:מכיוון שבניית המערך משתמשת באותו אלגוריתם הנלמד, בניית המערך בסיבוכיות זמן ריצה של $O(n)$ כנלמד.
   עבור הפעולה למציאת מינימום לתת מערך: נוכיח למטה טענת עזר שבכל קומה יש לכל היותר 4 צמתים בהם האלגוריתם עובר עליהם, מכיוון שסריקה מהשורש לעלה הוא $O(log\ n)$, הכפלת הסיבוכיות זמן ריצה ב-4 לא תשפיע על החסם האסימפטוטי שהוא כאמור $log\ n$, ולכן פעולת החיפוש תהיה $O(log\ n)$
   טענת עזר: נוכיח באינדוקציה ש בכל קומה יש לכל היותר 4 צמתים.
   בסיס: עבור הקומה הראשונה (גובה 0) אכן יש צומת אחת, שקטן מ4 צמתים לכן הבסיס מתקיים.
   שלב: נניח שהטענה נכונה עבור הקומה ה-$i-1$ ונוכיח את הטענה עבור i.
   צעד: עבור הצעד ה-i: במידה ויש בקומה ה-$i-1$ צומת אחת או 2, וכל צומת תקרא באופן רקורסיבי ל-2 הבנים, בקומה ה-i יהיה 4 צמתים, לכן הטענה מתקיימת במצב זה.
   עבור מצב בו עברו על 3\4 צמתים בקומה ה$i-1$: עבור הקומה ה-i, בכל קומה בעץ חיבור כל הטווחים מכילים את כל הטווח כולו, ובפרט את הטווח שאנחנו מחפשים. הקטע שאנחנו מחפשים הוא תת קטע רציף, ואם יש לנו 3\4 צמתים יהיו לנו לכל היותר 2 צמתים חיצוניים לחלוטין שכל אחד מהם יחזיר ∞ מכיוון שאם היה בטווח בקומה ה-$i-1$ לא צורך לפיצול ל-2 קריאות, וה2 קריאות הנוספות עבור 2 טווחים פנימיים שמוכלים לחלוטין בטווח שאנחנו מחפשים.
   לכן: אם נסתכל בכל רמה על לכל היותר 4 צמתים, ומעבר על כל העץ הוא $O(log\ n)$ אזי גם $O(4log\ n) = O(log\ n)$ ולכן הפונקציה עונה לתנאי סיבוכיות זמן הריצה כמבוקש.