מבנה נתונים
מטלה 3

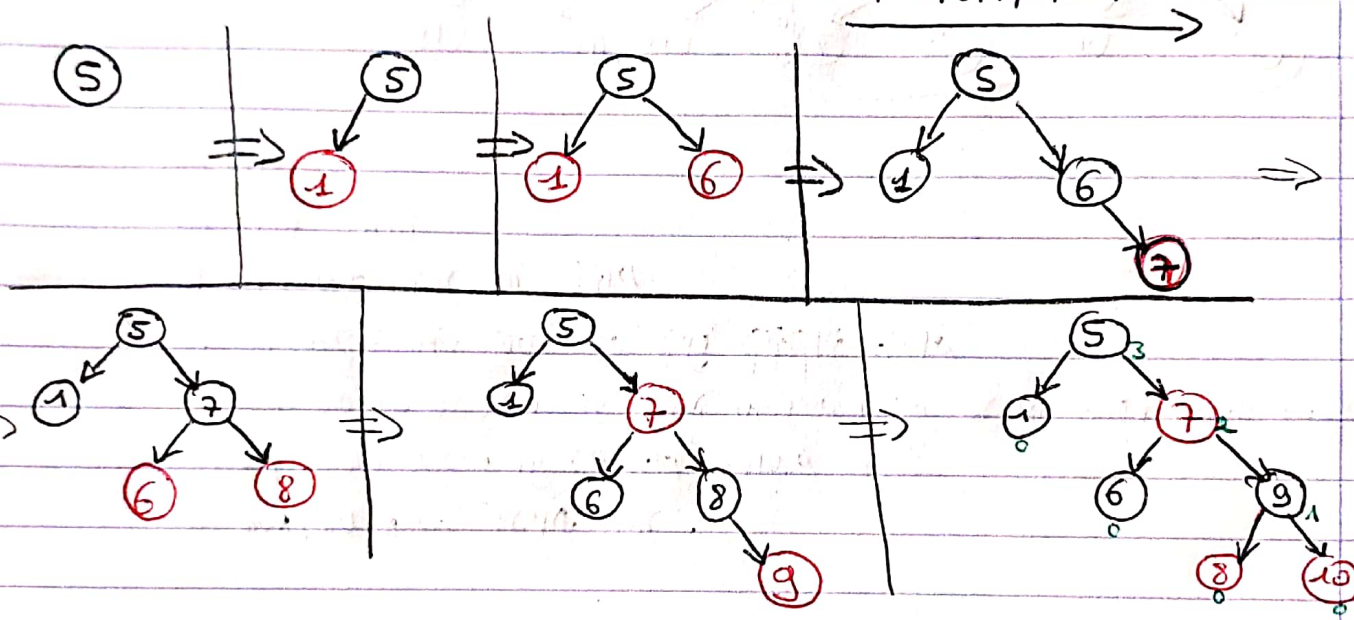
סופר
מ"ע
342615648

שאלה 1:

א. מה יהיה הגובה של העץ?

ציירו את העץ המתקבל לאחר הכנסת המספרים לפי הסדר הבא:

5, 1, 9, 7, 8, 6, 10, 0



• ← שורש

ולאחר מכן מחקו את השורש של העץ שקיבלתם. הסבירו מדוע בחרתם באיבר זה, ושקיבלתם עץ חיפוש בינארי כתוצאה (5) ולאחר מכן מחיקת השורש 7:

$$2 = 0 - 2$$

הסבירו כיצד ניתן למצוא עץ חיפוש בינארי שגובהו 2 עבור קבוצת המספרים הנתונה.

Scanné avec CamScanner

כל איבר (מלבד השורש) בתת עץ ימני גדול מהאב.

מלמדת כלומר שכל תת-עץ השמאלי של אבא:

1) מכיל/לא מכיל איברי שורש.
2) כולו מכיל איברים קטנים מהאבא בהכרח
3) כולו מכיל איברים שהם גדולים מהאבא
4) כל הקבקוד שם שהם גדולים מהאבא
5) כל בניו של אבא הם קבקוד גדולים
6) Root שורש

מסקנות:

כאן ניתן לשים לב שהעץ שורשו הינו שורש 10 כל הקטנים:

א) נמצאים בתת עץ שמאלי, כלומר הם קטנים מהשורש (10)

שאלה 2:

ד) הטענה אינה נכונה: על עץ מסלולי שקיים בו מסלול אחד שאורכו $\log n$ לפחות, ייתכן שכל שאר המסלולים הפשוטים בעץ הם באורך 1 בלבד. לדוגמה, עץ בינארי שבו לכל צומת פנימי יש בן ימני יחיד ובן שמאלי עלה.

ג) הטענה אינה נכונה, ניתן לבנות עץ חיפוש בינארי שבו לכל צומת פנימי יש בן ימני יחיד ובן שמאלי עלה. בעץ כזה מספר העלים הוא $\lceil \frac{n}{2} \rceil$ ולכן מספר העלים שווה למספר הצמתים הפנימיים. בנוסף, לכל צומת פנימי יש $\frac{n}{2}$ צאצאים, ולכן לפחות $\frac{n}{2}$ צמתים הם צאצאים של הבן הימני של השורש.

ב) הטענה אינה נכונה: על מנת לבנות עץ AVL שמקיים את התנאים הנתונים, נתחיל מעץ ריק ונוסיף לו צמתים בזה אחר זה. לאחר הוספת כל צומת נבדוק האם העץ הפך לא מאוזן, ואם כן נבצע את הגלגול המתאים. מכיוון שבכל שלב מספר הצמתים בעץ גדל ב-1, לא ייתכן שבשלב כלשהו נוסיף צומת שיגרום לגובה העץ לגדול ביותר מ-1.

א) הטענה אינה נכונה: נתבונן בסדרת המפתחות הבאה: $4, 11, 10, 6, 18, 7, 9, 5$. לאחר הכנסת המפתחות לעץ בסדר זה נקבל את העץ הבא:

לאחר הכנסת 9 שגובהו שני כלי 1-ל עולה לפחות $\lceil \frac{h}{2} \rceil$ צמתים מאוזנים כי $1 = \lceil \frac{1}{2} \rceil$ ו-9 הוא 0 מאוזן מכיוון שגובה תת-העץ של 9 הוא 1.

Scanné avec CamScanner

שאלה 3:

א) צריך לעבור שוב בסדר in-order על עץ חיפוש בינארי ולבדוק אות אחרי
מעל מספר איין.
ניתן להשתמש בעץ בינארי למשהו כי שמרנו בקשתות in-order יקבלו אותו סדר
מספר על איין.

מכיוון שהעץ שלנו סופי חייבת להיות מלכתחילה כזה - שהוא מקבל עץ
בגובה T.
כמות T שמובילים לעלות יותר (למשל מן שלה) יכלו ל-T חייב לעלות בזווית למשהו
$R \leftarrow \overset{x}{\nearrow} T$ כשאר כל הזה צריך קודקוד הראשון מין כמה שכל ואחרי
ל-R כזה תכונה. מכיוון ש-T הזה מינימלי ואחרון, יש לפחות ל-R
צדיק שה מספרים שהם לפני מ"ב לב פחות מספרים.
ואז לפה קודקודים מ-T יכולים לעלות קטן מ-x ולפי קודקודים מ-R
ובעלם לעלות גדול מ-x.
אבל ניתן ל-T כזה $T = L + x + R$
לפני שהעלה את T מינימלי שהעלה לפני T
כמה אז חייב לפחות איין בהתאם את הזהות שלהן
$T = L + x + R$

ב) ניתן עץ מינארי T ונתון שהפעלת in-order על T מחזירה מספר איין
צריך לעבור ב-T כזה עץ חיפוש מינארי
שם אינטואיציה אז יצה איין מה יותר ב קודקודים מעל:
אלגוריתם ב in-order ~~Left → Root → Right~~ כי Left → Root → Right
Right ≥ Root ≥ Left
כלל אלגוריתם שמשנה אות מה לפחות מחזירים את T ואלגוריתם כולנו שלפני
הוא Left קטן או שווה לפחות את ה-Root ואחר Right גדול או שווה
לעץ ה Left.

Scanné avec CamScanner

3) **הוכחה על נכונה**: באינדוקציה על k:

עלינו להוכיח שבכל עומק k בעץ מלא יש 2^k צמתים ועלים 2^{k-1} בעומק k.

בסיס: עומק $k=0$ – העץ מורכב מצומת יחיד (השורש), שהוא גם עלה.
$2^0 = 1$ צומת, $2^{0-1} = 2^{-1} = \frac{1}{2}$ — נבדוק $k=1$:
שורש ושני בנים, $2^1 = 2$ צמתים, $2^{1-1} = 1$ עלים.

הנחת האינדוקציה: נניח שלכל k טבעי, בעץ מלא בעומק k יש 2^k צמתים ו-2^{k-1} עלים.

צעד: נוכיח עבור $k+1$. עץ מלא בעומק $k+1$ נבנה מעץ מלא בעומק k על ידי הוספת שני בנים לכל עלה. לפי הנחת האינדוקציה, בעומק k יש 2^{k-1} עלים, ולכן בעומק $k+1$ יש $2 \cdot 2^{k-1} = 2^k$ צמתים. כל העלים בעומק k הפכו לצמתים פנימיים, ובעומק $k+1$ נוספו 2^k עלים חדשים.

מסקנה: $N = 2^k$, ולכן $k = \log_2(N)$.

מספר הצמתים הכולל בעץ הוא $2^{k+1} - 1 = 2N - 1$, ולכן סיבוכיות הזמן היא $O(N)$.

4) **הוכחה כי נכונה**, שלפי הלולאה מתקיים אלגוריתם $O(\log_2 n)$ מחזיר את הערך $\log_2 n$.

אלגוריתם יחפש את הלוג של n ע"י מספר הפעמים שהלולאה $O(\log_2 n)$ תתבצע עד שהתנאי $i \geq n$ יתקיים. כל איטרציה מכפילה את i ב-2, ולכן i מקבל את הערכים $1, 2, 4, 8, \ldots, 2^k$ עד שהתנאי $2^k \geq n$ יתקיים. כלומר, k הוא המספר הקטן ביותר שעבורו $2^k \geq n$, ולכן $k = \lceil \log_2 n \rceil$.

$\log_2 n$ הוא k כי k הוא מספר האיטרציות שהלולאה תתבצע עד שהתנאי $i \geq n$ יתקיים.

5) **הוכחה על נכונה**: באינדוקציה

עלינו להוכיח שבכל עומק k בעץ מלא יש 2^k צמתים ועלים 2^{k-1}.

עץ בינרי שלם מלא:

- **בסיס**: עומק $k=0$ – העץ מורכב מצומת יחיד (השורש), שהוא גם עלה.
$2^0 = 1$ צומת, $2^{0-1} = 2^{-1} = \frac{1}{2}$ — נבדוק $k=1$:
שורש ושני בנים, $2^1 = 2$ צמתים, $2^{1-1} = 1$ עלים.

- **הנחת האינדוקציה**: נניח שלכל k טבעי, בעץ מלא בעומק k יש 2^k צמתים ו-2^{k-1} עלים.

- **צעד**: נוכיח עבור $k+1$. עץ מלא בעומק $k+1$ נבנה מעץ מלא בעומק k על ידי הוספת שני בנים לכל עלה. לפי הנחת האינדוקציה, בעומק k יש 2^{k-1} עלים, ולכן בעומק $k+1$ יש $2 \cdot 2^{k-1} = 2^k$ צמתים. כל העלים בעומק k הפכו לצמתים פנימיים, ובעומק $k+1$ נוספו 2^k עלים חדשים.

- **מסקנה**: $N = 2^k$, ולכן $k = \log_2(N)$.

מספר הצמתים הכולל בעץ הוא $2^{k+1} - 1 = 2N - 1$, ולכן סיבוכיות הזמן היא $O(N)$.

Scanné avec CamScanner

PrintLevelOrder(node):

1. Create empty queue Q
2. Q.enqueue(node)
3. While Q is not empty:
    3.1. v = Q.dequeue()
    3.2. print v
    3.3. if there is left node
        3.3.1. Q.enqueue(v.left)
    3.4. if there is right node
        3.4.1. Q.enqueue(v.right)

PrintReverseLevelOrder(node):

1. Create empty stack S
2. Create empty queue Q
3. Q.enqueue(node)
4. While Q is not empty:
    4.1. v = Q.dequeue()
    4.2. S.push(v)
    4.3. If there is right node
        4.3.1. Q.enqueue(v. right)
    4.4. If there is left node
        4.4.1. Q.enqueue(v. left)
5. While S is not empty:
    5.1. v = S.pop()
    5.2. print v

Scanné avec CamScanner