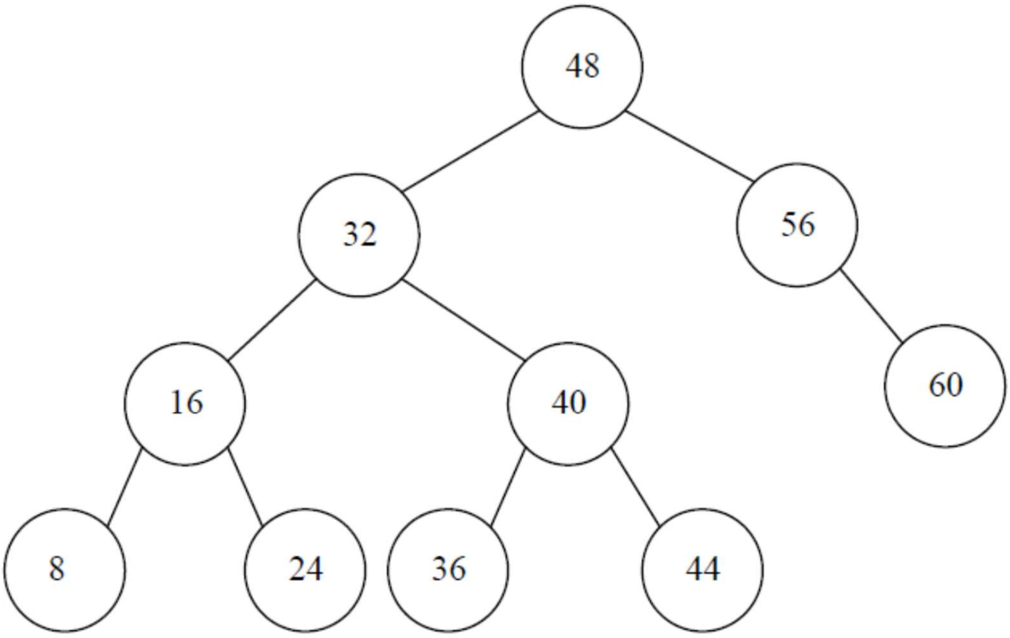
תרגילים נוספים במבני נתונים

<u>סעיף א (10 נקודות)</u>: נתון **<u>עץ ה – AVL</u>** הבא:



1. (5 נקודות) האם קיים מפתח **<u>שהכנסתו</u>** לעץ (וביצוע רוטציות אם נדרשות) תגרום לשינוי בגובה העץ? **כן/לא (סמן את התשובה הנכונה)**. אם כן, איזה? __________ (אם קיימת יותר מאפשרות אחת, ציין רק אחת מהן).

2. (5 נקודות) האם קיים מפתח **<u>שמחיקתו</u>** מהעץ (וביצוע רוטציות אם נדרשות) תגרום לשינוי בגובה העץ? **כן/לא (סמן את התשובה הנכונה)**. אם כן, איזה? __________ (אם קיימת יותר מאפשרות אחת, ציין רק אחת מהן).

בסעיף א: תשובה א תשובה ג ותשובה ח לא בחומר

ד הכוונה לטבלת גיבוב רגילה כנ"ל לגבי הסעיף השני

שאלה 1 (16 נקודות)

סעיף א' (8 נקודות)

עבור אילו ממבני הנתונים הבאים הטענה הבאה מתקיימת **תמיד** (**הקיפו** את כל התשובות הנכונות, אין צורך בהסבר):
מבנה הנתונים המתקבל מהכנסת איבר ומיד מחיקתו מהמבנה, זהה לחלוטין למבנה הנתונים לפני ביצוע שתי הפעולות הנ"ל.

א. Skip List
ב. AVL
ג. Hash – Double Hashing
ד. Hash – Chaining
ה. BST
ו. B-Tree
ז. Linked list
ח. BB-[α]-tree
ט. אף תשובה אינה נכונה

סעיף ב' (8 נקודות)

עבור אלו ממבניי הנתונים הבאים הטענה הבאה מתקיימת **תמיד** (**הקיפו** את כל התשובות הנכונות, אין צורך בהסבר):
מבנה הנתונים המתקבל ממחיקת איבר (קיים) ומיד הכנסתו בחזרה למבנה, זהה לחלוטין למבנה הנתונים לפני ביצוע שתי הפעולות הנ"ל.

א. Skip List
ב. AVL
ג. Hash – Double Hashing
ד. Hash – Chaining
ה. BST
ו. B-Tree
ז. Linked list
ח. BB-[α]-tree
ט. אף תשובה אינה נכונה

שאלה 1 (25 נקודות)

בשאלה זו יש לתאר מבנה נתונים ששומר קבוצות של מספרים שלמים. **כל קבוצה יכולה להכיל עד 10 מספרים**. הקבוצות לא זרות (כלומר מספר מסוים יכול להופיע במספר קבוצות). השם של קבוצה הוא המספר הסידורי בו הוכנסה הקבוצה למבנה.

מבנה הנתונים נדרש לתמוך בפעולות הבאות (n הוא מספר הקבוצות שנמצאות כרגע במבנה):

שם פעולה	תאור פעולה	זמן ריצה במקרה הגרוע
Init()	אתחול מבנה נתונים ריק.	O(1)
Add(S)	הוסף את הקבוצה S למבנה. הקבוצה S נתונה ע"י רשימה מקושרת של המספרים בקבוצה.	O(log n)
Delete(i)	מחק את הקבוצה עם שם i. הניחו כי קבוצה עם שם i נמצאת במבנה.	O(log n)
First(x,k)	החזר i מינימלי כך שקבוצה i מכילה את האיבר x, ומתקיים ש-i גדול ממש מ-k. הניחו כי איבר x נמצא במבנה. אם לא קיים i כנדרש יש להחזיר 1-.	O(log n)

<u>מבנה נתונים יהיה מורכב מ-</u>

1. **עץ AVL, T1, המסודר לפי המספר הסידורי של הקבוצה. כל צומת בעץ מכיל כמפתח את מספר הקבוצה, i. בנוסף כל צומת שומר רשימה מקושרת של המספרים ששייכים לקבוצה.**
2. **עץ AVL, T2, ששומר את קבוצת כל האיברים (מספרים) שמופיעים בכל הקבוצות (איבר שמופיע במספר קבוצות נשמר בעץ רק פעם אחת). מפתח של הצומת הוא מספר x. בנוסף, לכל צומת בעץ קיים מצביע לעץ AVL שמכיל את כל מספרי הקבוצות המכילות את x. כיוון שבכל קבוצה יש לכל היותר 10 איברים, אז העץ T2 מכיל לכל היותר 10n צמתים.**
3. **כמו כן, נחזיק מונה בשם counter של מספר הקבוצות שהוכנסו למבנה.**

<u>**פעולות:**</u>

Init()
T1←NULL
T2←NULL
counter ←0

זמן ריצה O(1).

Add(S)

counter ←counter + 1

הכנס לעץ T1 צומת בעל מפתח counter, שזה מספר הסידורי של הקבוצה החדשה S.

עבור כל $x\in S$, חפש את x בעץ T2.

- אם x נמצא בעץ T2, אז הוסף את מספר הקבוצה S לעץ ה-AVL שבצומת של x.
- אחרת, הוסף את x לעץ T2, וצור בצומת של x עץ AVL שמכיל איבר בודד, שהוא מספר הקבוצה S.

כיוון שב-S יש לכל היותר 10 איברים הזמן הוא $O(\log n)$.

Delete(i)

חפש את קבוצה עם מספר סידורי i בעץ הקבוצות, נקרא לה S. זמן ריצה $O(\log n)$:

עבור כל $x\in S$ (לפי הרשימה ששמורה בצומת), חפש צומת בעץ T2 שמכיל מפתח x.

מחק את i מעץ ה- AVL שבצומת של x. אם לאחר מחיקה זו העץ הפך להיות ריק, אז מחק את x מעץ T2.

כיוון שכל קבוצה מכילה לכל היותר 10 מספרים, זמן ריצה הכולל הוא $O(\log n)$.

First(x,k)

חפש את x בעץ T2. בעץ AVL ששמור בצומת x בצע חיפוש של i בדרך הבאה: אתחל משתנה בשם min לערך אינסוף. התחל בשורש. כאשר נמצאים בצומת v עם מפתח i, נשווה בין i ו-k.

1. אם $i \le k$ נרד לבן הימני של v, אם הוא קיים.
2. אם $i > k$ בצע:
   a. אם $i < min$ הצב $min \leftarrow i$.
   b. רד לבן השמאלי של v, אם הוא קיים.

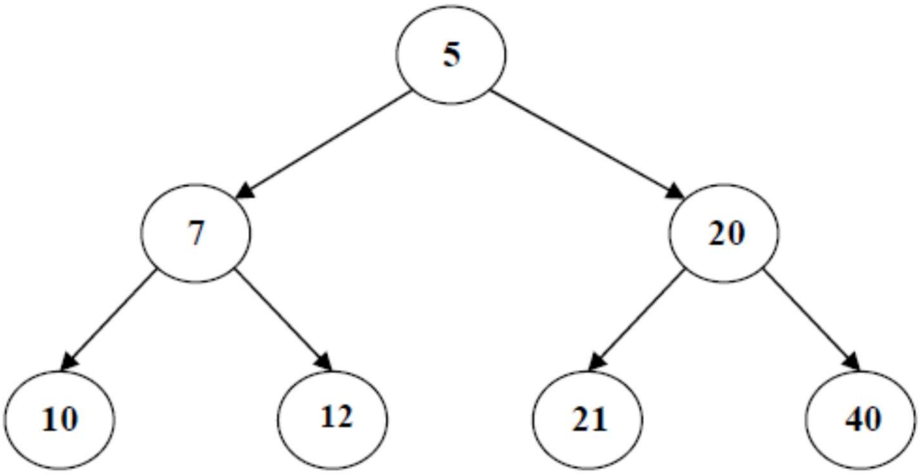
כאשר לא ניתן להמשיך לרדת בעץ, אם הערך של min הוא לא אינסוף, החזר min. אחרת החזר -1.

שאלה 2

סיור preorder על עץ חיפוש בינארי הינו אלגוריתם רקורסיבי לביקור המפתחות בעץ. סיור זה מבקר בשורש העץ ולאחר מכן מבקר באופן רקורסיבי בתת העץ השמאלי שלו ובתת העץ הימני שלו.

Preorder Tree (בקיצור PT) הוא עץ בינארי שבקודקודיו מאוחסנים מפתחות (מספרים) שונים זה מזה. המפתחות מסודרים לפי סדר ה- preorder, כלומר הדפסתם תוך סיור preorder היא סדרה ממוינת עולה.

דוגמא ל-PT:



א. הציעו אלגוריתמים למציאת האיבר המינימאלי והאיבר השני המינימאלי ב-PT בזמן $O(1)$.
ב. האם לכל n קיים PT בן n איברים שונים שהוא גם עץ חיפוש בינארי? אם כן, תארו אותו; אם לא נמקו מדוע.
ג. הציעו אלגוריתם לחיפוש איבר ב-PT, בזמן $O(h)$ כאשר h הוא גובה העץ. הסבירו בקצרה את נכונותו ואת נכונות זמן הריצה שלו.
ד. הציעו אלגוריתם המוחק איבר ב-PT, כך שהעץ המתקבל לאחר המחיקה גם הוא PT. על האלגוריתם לפעול בזמן של $O(h)$ כאשר h הוא גובה העץ. הסבירו בקצרה את נכונות האלגוריתם ונכונות זמן הריצה שלו.

תשובה

שאלה 2

א. המפתח המינימאלי הוא שורש העץ.

מציאת המפתח השני המינימאלי: אם קיים לשורש בן שמאלי נחזיר אותו. אחרת נחזיר את הבן הימני.

ב. כן. עץ כזה יהיה שרשרת מוטה ימינה מהשורש בגובה $O(n)$ (כלומר כל הבנים ימניים).

ג. עבור כל צומת x בעץ מתקיים:

x.key < x.left.key < x.right.key

```
Find(k,T)
x ← T.root
While ( x !=null && k>x.key )


    if (x.key == k)
            return x
    if (k < x.right.key)
            x ← x.left
    else
            x ← x.right
return "not found"
```

הפונקציה עובדת ב O(h) כיוון שבכל קריאה רקורסיבית אנו יורדים רמה בעץ, כלומר במקרה הגרוע נטייל במסלול הארוך ביותר מהשורש לעלה, שאורכו h.

ד. מחיקת מפתח k: נחפש את הצומת x בו נמצא המפתח k בעץ. אם x עלה מחק אותו וסיים. אחרת, מצא את y העוקב של x, החלף את x ב-y ומחק את y רקורסיבית.
(מציאת עוקב: בן שמאלי אם קיים כזה, אחרת בן ימני).

שאלה 4

במכולת של משה הוחלט להטמיע מערכת הזמנות ממוחשבת. המערכת מנוהלת על ידי תור (FIFO). כאשר לקוח מבצע הזמנה, פרטי ההזמנה (מוצר, כמות ושם המזמין) נכנסים לסוף התור. במכולת יושב משה, שולף הזמנות מראש התור ומבצע אותן אחת אחת.
על מנת לשפר את ניהול המלאי שלו, משה רוצה שתהיה לו היכולת לדעת מתי מלאי של מוצר מסוים עומד להיגמר. לצורך כך משה דורש לדעת בכל פעם שהוא מוציא הזמנה מראש התור כמה הזמנות נוספות קיימות בתור לאותו מוצר. כך יוכל לדאוג מראש שתהיה לו כמות מספקת במלאי.

כדי לעזור למשה, הציעו מבנה נתונים התומך בפעולות הבאות:

- enqueue(r) – הכנסת הזמנה r לסוף התור (r מכילה שלושה שדות: שם המוצר, כמות ושם המזמין). זמן ריצה: O(1) בממוצע.
- dequeue() – הוצאת ההזמנה שבראש התור. זמן ריצה: O(1) בממוצע.
- Query(p) – החזרת הכמות המוזמנת בתור של מוצרים מסוג p. זמן ריצה: O(1) בממוצע.

ידוע שבמכולת קיימים n מוצרים. על התור להכיל m הזמנות בו זמנית לכל היותר. ידוע ש-m<n.
מחוסר תקציב על מבנה הנתונים להשתמש רק ב- O(m) זיכרון.

תשובה

נשתמש בתור ובטבלת hash בגודל O(m). הטבלה תנוהל בשיטת השרשור (chaining). איבר בטבלה (כלומר חוליה) מכיל שדה מפתח – שם מוצר, ושדה נוסף – כמות.

Enqueue(r)

הכנס את ההזמנה r לסוף התור.

חפש את r.p בטבלה. אם נמצא – הוסף לשדה הכמות את r.quantity

ואם לא נמצא – הכנס את r.p לטבלה עם כמות r.quantuty.

Dequeue()

הוצא את ההזמנה r שבראש התור.

חפש את r.p בטבלה. הפחת משדה הכמות את r.quantity.

אם הכמות היא עתה 0 – הוצא את r.p מהטבלה.

Query(p)

חפש את p בטבלה.

אם נמצא – החזר את הכמות שלו, אחרת – החזר 0.

נשים לב כי:

$$\alpha = \frac{\text{מס' המוצרים השונים בטבלה}}{\text{גודל הטבלה}} \le \frac{m}{O(m)} = O(1)$$

ולכן זמן הריצה של כל אחת משלושת הפעולות הוא $O(1)$ בממוצע.

סעיף א (5 נק')

אם לקודקוד בעץ חיפוש בינארי יש שני ילדים אז לקודם (predecessor) שלו אין בן ימני **(הקף את התשובה הנכונה):**

1. נכון
2. לא נכון
3. **לא ניתן לדעת**

סעיף ב (7 נק')

מחפשים את 124 בעץ חיפוש בינארי. איזה מבין הסדרות הבאות עשויה להיות סדרת המספרים בה ניתקל במהלך החיפוש **(הקף את <u>כל</u> התשובות הנכונות):**

1. 1,370,391,120,123,124
2. 1,924,911,63,898,101,113,124
3. 363,12,258,93,99,244,125,124
4. 800,750,379,512,401,430,124

סעיף ג (8 נק')

אם n,m צמתים שונים בעץ חיפוש בינארי אז ידוע כי בדיוק אחת מהאפשרויות הבאות נכונה:

a. n הוא משמאל ל- m (כלומר יש אב קדמון משותף, x , של n ו- m, כך ש- n בתת העץ השמאלי של x ו-m בתת העץ הימני של x).
b. n הוא מימין ל- m.
c. n הוא אב קדמון של m.
d. n צאצא של m.

נניח עתה כי inorder(n) < inorder(m) אזי האפשרויות היחידות הן (**הקף את התשובה הנכונה**):

1. a, c
2. a, c ,d
3. b, c
4. b, c, d
5. a, d

שאלה 4 (30 נקודות)

אנו מעוניינים לתחזק מבנה נתונים בו מאוחסנים n איברים המכילים מפתחות טבעיים כלשהם ומתקיימים:

(1) כל מפתח מופיע לכל היותר פעם אחת במבנה הנתונים.
(2) קיים סדר ליניארי בין האיברים המאוחסנים במבנה הנתונים. הסדר נקבע עפ"י ביצוע פעולות ה-Insert וה- Delete המוגדרות להלן.

נתאר עתה פעולות על מבנה הנתונים, ולאחר מכן את זמן הריצה הדרוש לכל פעולה.

הפעולות המוגדרות הן:

1. Init(k) – אתחל מבנה עם איבר בודד. זהו האיבר הראשון בסדר הליניארי.
2. Find(k) – החזר true אם האיבר שמפתחו k נמצא במבנה, ו-false אחרת.
3. Insert(k_1 , k) – הכנס איבר חדש עם מפתח k כך שמיקומו בסדר הליניארי הוא מיד אחרי האיבר בעל המפתח k_1. ניתן להניח שלפני ביצוע הפעולה, איבר עם מפתח k_1 קיים במבנה הנתונים ואיבר עם מפתח k לא קיים במבנה.
4. Delete(k) – הוצא מן המבנה את האיבר שהמפתח שלו הוא k, אם נמצא במבנה.
5. Order(k_1, k_2) – החזר true אם האיבר שהמפתח שלו הוא k_1 נמצא בסדר הליניארי לפני האיבר שהמפתח שלו הוא k_2, אחרת החזר false (האיברים k_1, k_2 הם **לא בהכרח עוקבים** בסדר וניתן להניח כי נמצאים במבנה)

דוגמת ריצה:

```
Init (6)
Insert (6, 1)
Insert (1, 3)
Insert (6, 7)
Insert (1, 4)
Find (1)        //true יחזיר
Order (7, 4)    //true יחזיר
Order (3, 4)    //false יחזיר
Delete (6)
Find (6)        //false יחזיר
Order (7, 3)    //true יחזיר
```

עבור כל אחד מהסעיפים הבאים הצע מבנה נתונים בגודל O(n) *המבוסס בין השאר* ***על טבלת hash בגודל m***, העומד בדרישות סיבוכיות זמן הריצה ***במקרה הממוצע*** הנתון לכל פעולה. (המספר n מציין את מספר האיברים המאוחסנים במבנה בזמן ביצוע הפעולה):

סעיף א (15 נק') Init ב-O(m), Find, Insert ו-Delete ב-O(1), Order ב-O(n).

סעיף ב (15 נק') Init ב-O(m), Find, Delete ו-Order ב-O(1), Insert ב-O(n).

יש לתאר בקצרה, אך במדויק את הפתרונות. אפשר (וכדאי) להיעזר בציורים של מבנה הנתונים.

אין צורך לתת פסאודו-קוד של השיטות, רק לתאר בעברית בקצרה מה הן עושות.

רמז: כל איבר בטבלה יכול להכיל מספר קבוע של שדות נוספים.

רמז לסעיף ב': ניתן לחשוב על תוספת קטנה למבנה הנתונים שהצעתם בסעיף א