תרגיל 7- מבני נתונים 67109:

שם: דן קורנפלד

חלק 1:

שאלה 1:

1. סעיף א:

נכתב ידנית מכיוון שיש צורך באיור

תרגיל 7- מבני נתונים 67109:

שם: דן קורנפלד

חלק 2:

שאלה 2:

1. ניצור מבנה נתונים נתונים חדש מאפשר את ה-API הנתון. נקרא למבנה הנתונים $hash-tree$ מכיוון שהוא מכיל גם AVL וגם $hash-table$ מכיוון שהוא משתמש ביכולות של 2 תתי מבני הנתונים, בנוסף נשמור פוינטר לערך המינימלי במבנה הנתונים, ופוינטר לערך המקסימלי במבנה הנתונים- 2 הפוינטרים יאותחלו להיות Null בהתחלה, ובהכנסת האיבר הראשון נשנה את הערכים שלהם להיות האיבר הראשון. כמו כן, לכל איבר ב AVL נשמור 2 פוינטרים עם הערך ל- $Successor(m), predecessor(m)$ של כל עץ.
★ - מכיוון שאנחנו משתמשים ב-AVL הכנסת איבר והוצאת איבר יהיו בזמן ממוצע של $O(log\ n)$.
2. $insert(m)$: עבור הכנסת איבר כלשהו למבנה הנתונים החדש בזמן ממוצע של $O(log\ n)$: למדנו שהכנסת איבר לAVL הוא זמן ממוצע של $O(log\ n)$, בנוסף מכיוון שהכנסת איבר לטבלת גיבוב לוקחת בזמן ממוצע $O(1)$ נכניס גם את האיבר לטבלת הגיבוב, כמו כן במידה והפוינטרים Min, Max הם $null$, נאתחל אותם להיות הערך של m, במידה והם לא $null$ נבדוק אם m גדול מהערך המקסימלי או קטן מהערך המינימלי ונעדכן בהתאם. בנוסף בכל הכנסה נעדכן את ה $Successor(m), predecessor(m)$ בזמן הכנסת האיבר החדש. פעולות ה- $Successor(m), predecessor(m)$ בסיבוכיות זמן של $O(log\ n)$ לכן אנחנו עומדים בדרישות השאלה.
3. $delete(m)$: נוציא תחילה את האיבר הרצוי מטבלת הגיבוב בסיבוכיות זמן ממוצע של O(1). במידה והאיבר לא נמצא, נוכל להחזיר כבר שגיאה, שכן עבור מקרה זה, טבלת הגיבוב יעילה יותר. במידה והערך נמצא, נוציא את האיבר מAVL בסיבוכיות זמן ממוצעת של $O(log\ n)$. במידה והמספר שמוציאים מצביע על המקסימום או על המינימום, נעדכן בהתאם את הערך החדש של המקסימום והמינימום, חיפוש המינימום והמקסימום ב-AVL הוא $O(log\ n)$ לכן עונה על התנאים של מבנה הנתונים הרצוי. כמו כן, בכל הוצאת איבר נעדכן את ה $Successor(m), predecessor(m)$ בזמן הוצאת האיבר, בהתאם לצורך. פעולות ה-$Successor(m), predecessor(m)$ בסיבוכיות זמן של $O(log\ n)$ לכן אנחנו עומדים בדרישות השאלה.
4. $find(m)$: לצורך פעולות ה$find$ נצטרך מהתחלה את טבלת הגיבוב כדי למצוא אם מפתח כלשהו נמצא במבנה הנתונים בסיבוכיות זמן של $O(1)$ בתוחלת, לכן עבור בדיקה אם איבר נמצא במבנה הנתונים נשתמש בטבלת הגיבוב ונחזיר אמת אם האיבר נמצא, ונחזיר שקר אם לא נמצא.
5. $getMin(), getMax()$ מציאת ערכי המינימום והמקסימום במקרה הגרוע ביותר ב$O(1)$: מכיוון שיש לנו 2 פוינטרים שאחראי להחזיר את הערכי המינימום והמקסימום, גישה לזיכרון תהיה לכל היותר $O(1)$, לכן הפונקציות בסיבוכיות זמן לכל היותר $O(1)$.
6. $Successor(m), predecessor(m)$- מכיוון שיש לכל איבר בעץ פוינטר לאיבר הקודם, ולאיבר הבא בעץ, נוכל לגשת ולתא הרלוונטי ולהחזיר אותו ב $O(1)$.
7. $byOrder()$ החזרת מערך ממוין במקרה הגרוע של $O(n)$ למדנו שניתן לסרוק את המערך "$Inorder\_tree\_walk(x)$" בסיבוכיות זמן של $O(n)$ כך שיוחזר המערך המבוקש ב$O(n)$ כמבוקש

לכן, מבנה הנתונים שהצגתי עומד בכל תנאי השאלה

תרגיל 7- מבני נתונים 67109:

שם: דן קורנפלד

חלק 3:

שאלה 3:

- נרצה לבדוק עבור כל איבר בעץ, אם $|hd(v)| \leqslant 1$. עבור כל קודקוד, נבדוק תחילה אם הקודקוד הוא Null, אם כן, נחזיר 0.
- נזמן את הפונקציה באופן רקורסיבי עבור הבן השמאלי, ונשמור את הערך בתור sub_tree_height_left , נבדוק אם הערך הוא (1-)- אותו נגדיר אם העץ הוא לא מאוזן.
- באותו אופן נבצע זאת עבור תת העץ הימני
- נבדוק אם הפרש הגבהים בערך מוחלט גדול מ-1, אם כן- נחזיר (1-) ,
- אחרת, נחזיר את הערך הגדול יותר מגובה העץ הימני או השמאלי ונוסיף לערך 1.

ניתוח זמן הריצה:

עבור כל קודקוד בעץ, אנחנו עוברים עליו פעם אחת, ובכל בדיקה על הקודקוד נבצע $O(1)$ עבודה, לכן מעבר על העץ בדומה ל- $Inorder\_tree\_walk(x)$ בסיבוכיות זמן של $O(n)$" ולכן סיבוכיות הזמן בצורה המירבית היא $O(n)$.

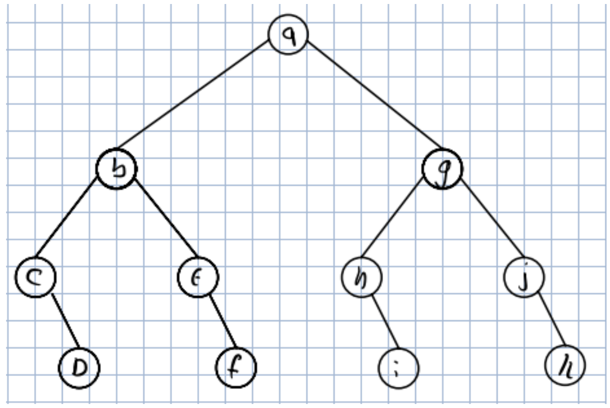
שאלה 4:

- נשתמש בטבלת גיבוב $(Hash - Table)$, נפתור את ההתנגשויות בשיטת השרשראות (chaining), אך במקום רשימה מקושרת, נשתמש ב-AVL במקום לכן בכל איבר בטבלה יהיה פוינטר לעץ AVL שיפתור את ההתנגשויות במקרה ונוצרות בפעולת הגיבוב.
- נתון שהמפתח ללא חזרות, לכן נכניס, נחפש ונמחק מטבלת הגיבוב ומה-AVL לפי הצורך לפי המפתח, ובכל קודקוד נאחסן גם את המפתח, וגם את הערך.
- הכנסה: עבור טבלת גיבוב למדנו שבאופן ממוצע הכנסת איבר חדש לוקחת O(1), ובמקרה הגרוע מכיוון שאנחנו משתמשים בAVL, נבצע הוספה של הקודקוד החדש לAVL המתאים, בסיבוכיות זמן לכל היותר $O(log\ n)$.
- מחיקה: באותו אופן כמו הכנסה: נבדוק תחילה אם הערך של המפתח נמצא, עבור טבלת גיבוב למדנו שבאופן ממוצע חיפוש איבר לוקח O(1), ובמקרה הגרוע מכיוון שאנחנו משתמשים בAVL חיפוש איבר יהיה ב$O(log\ n)$. כמו כן, למדנו שמחיקת איבר מהAVL ותיקון השגיאות במידה והעץ לא מאוזן בסיבוכיות זמן של $O(log\ n)$ ולכן המחיקה כולה תיקח $O(log\ n)$ במקרה הגרוע.
- חיפוש: נבדוק תחילה אם הערך של המפתח נמצא, עבור טבלת גיבוב למדנו שבאופן ממוצע חיפוש איבר לוקח O(1), ובמקרה הגרוע מכיוון שאנחנו משתמשים בAVL חיפוש איבר יהיה ב$O(log\ n)$.

לכן מבנה הנתונים שהצעתי עומד בדרישות הטענה.

שאלה 5:

העץ יהיה מהצורה:



אראה שהעץ הנוכחי מקיים את תנאי הטענה

עבור מחיקת 2 קודקודים:

עבור כל מחיקת 2 קודקודים סדר המחיקה לא שינה את גובה העץ, אפרוט את כל הדוגמאות למחיקת 2 קודקודים שיכולים ליצור בעיקר בעיות בגובה העץ:

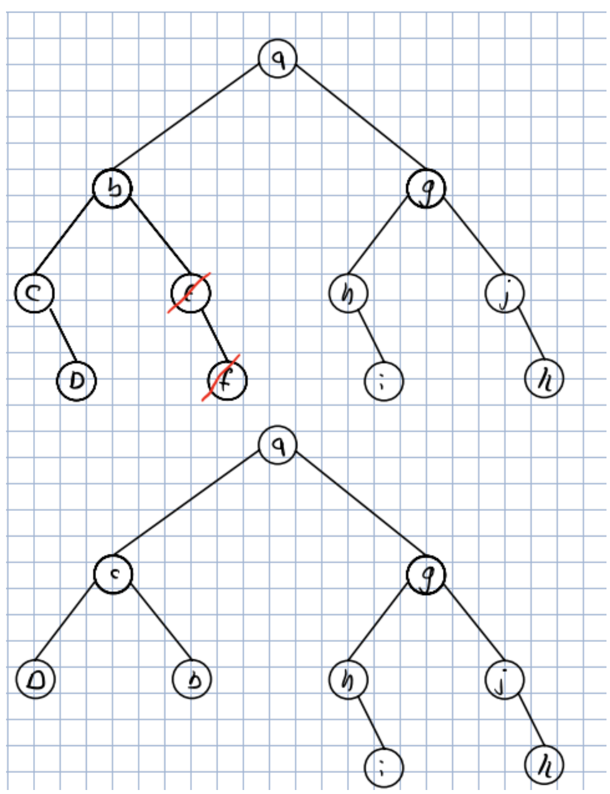
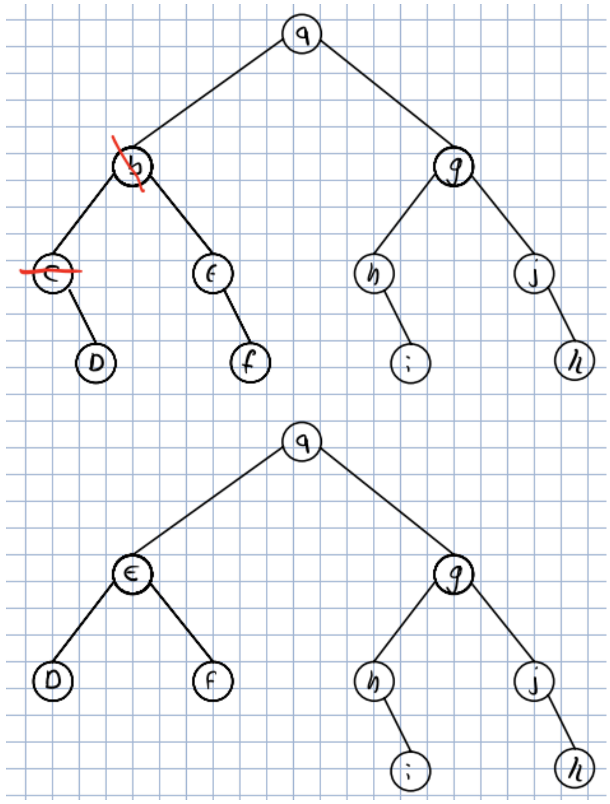
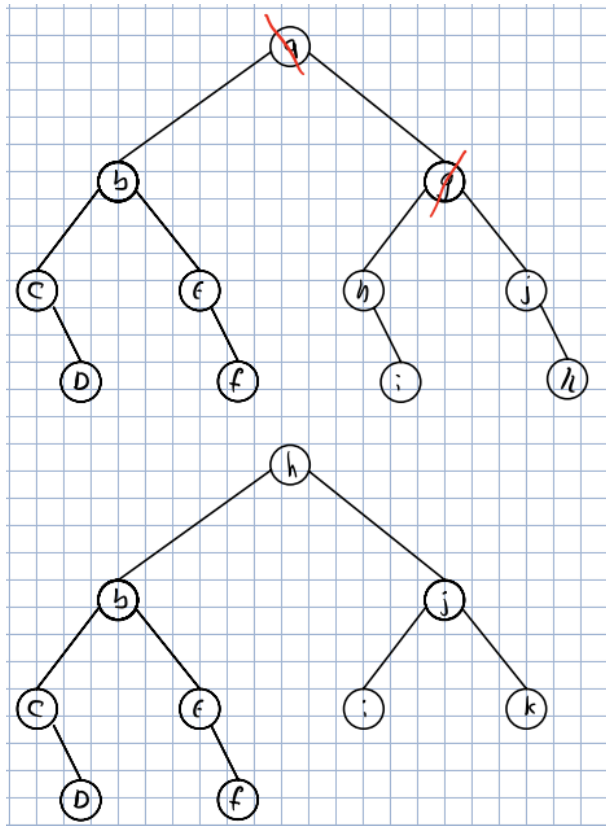


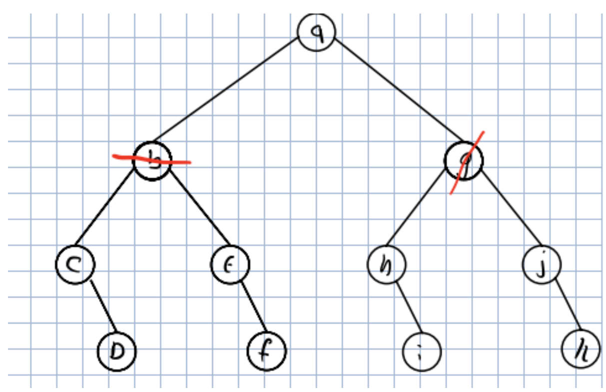
Figure 1: מחיקת 2 קודקודים אחד מהם עלה, השני הורה של אותו עלה, אין זה משנה באיזה עלה נבחר, הגובה של העץ לא ישתנה



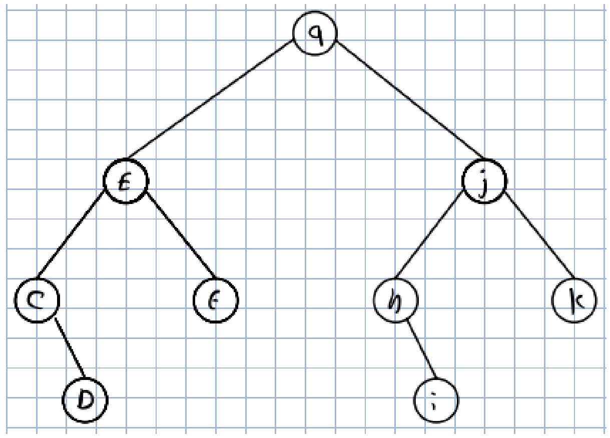
מחיקת הורה של עץ והורה של הורה, אין זה משנה אם נבחר בצד השני של ההורה, ניתן לראות שהגובה של העץ כולו לא משתנה, והעץ עדיין AVL תקין.



במידה ונרצה למחוק את השורש ואחד מקודקודיו, ניתן לראות לפי האיור הנל שהגובה עדיין נשאר אותו גובה בחיסור 2 קודקודים

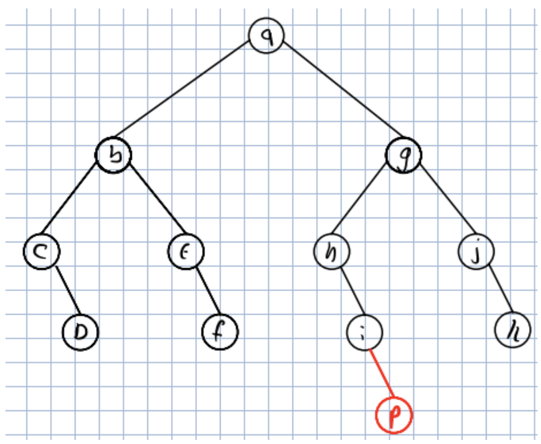


עבור המצב של הסרת 2 בנים מאותו הגובה:

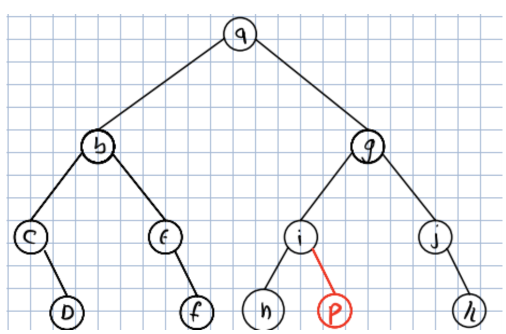


לאחר הסרת הקודקודים עדיין העץ יהיה בגובה המקורי.

עבור הכנסת איבר: במידה ונכניס איבר בגובה העלים, גובה העץ לא ישתנה
במידה ונכניס איבר נוסף שיהיה בן של אחד העלים כמו בדוגמה:



תחילה העץ לא יהיה מאוזן, לפי הדוגמה תהיה הפרה מסוג RR לכן לאחר טיפול בהפרה נראה שהגובה נשאר בגובה העץ המקורי



ולכן העץ עונה לדרישות השאלה כמבוקש