פונקציות:

Insert:

Main function- insert:

הפונקציה תקבל key, value ותיצור IAVLnode.

תחילה הפונקצייה בודקת אם העץ ריק- אם כן מעדכנת את כל השדות (מינימום ומקסימום להיות הנקודה ומחזירה 0- לא היו סיבובים ועדכונים.

אחרת, נחפש את הצומת שאליה נכניס את הצומת החדש/ אם הצומת קיימת נחזיר אותה באמצעותplace to insert .

נחבר את הצומת החדש למקום המתאים ונבצע עדכוני מינימום ומקסימום.

נקרא לrebalance ולבסוף לupdate till root.

Place\_to\_insert:

טיול ימינה כל עוד המפתח גדול מהצומת החדש ושמאלה אם ההפך.

זאת נבצע כל עוד האיבר שאליו אנו רוצים ללכת אינו צומת פיקטיבית אם כן נחזיר את הצומת האחרונה שאינה פיקטיבית וגם אם הגענו למפתח של הצומת החדש (כבר קיים בעץ) נחזיר אותו.

Rebalance:

פונקצייה רקורסיבית שמאזנת את העץ לאחר הכנסה בהינתן צומת שממנו צריך להתחיל\ להימשך תהליך האיזון.

בהינתן קלט של צומת מסוים בעץ ומס' השינויים שנעשו בתהליך האיזון הנוכחי עד כה, הפונקציה בודקת את getrankright, getrankleft- הפרשי הדרגות (הגבהים) של הצומת הנתון ביחס לבניו ולפי זה מסווגת לאיזה מקרה איזון מחדש צריך לקרוא או האם הצומת מאוזן וישר לסיים את הריצה הרקורסיבית.

בחלק ממקרה הסיווג יש צורך הבחנה לתתי מקרים באופן הבא:

הפרש דרגות (0,1)/(1,0)- נבצע promotion ונקרא שוב לrebalace (בתנאי שזה לא הroot ואז אין צורך).

הפרש דרגות (0,2)+ הפרש דרגות של בן שמאלי (1,2) – סיבוב ימינה. נקרא לright rotation. ראינו כי פעולה זאת מביאה למצב תקין ומאוזן בעץ ולכן נסיים ונחזיר את מספר הפעולות שאספנו עד כה.

הפרש דרגות (0,2)+ הפרש דרגות של בן שמאלי (2,1) – סיבוב כפול – תחילה שמאלה של הבן השמאלי ואז ימינה של הצומת העליונה והבן השמאלי.

\*המקרים הסימטרים של (2,0) מטופלים באופן סימטרי בתת תנאי נוסף בפונקציה.

בסיום כל פעולה נקרא לupdateNode עבור הצמתים שיש לעדכן מהנמוך יותר בעץ אשר תלוי בצמתים שלא השתנו ולגבוה יותר- לכל היותר 3 עדכונים.

הפונקציה אוספת באופן רקורסיבי בעזרת משתנה מוחזר מטיפוס int אשר אוסף את מספר הפעולות -עבור כל rotation, promotion, demotion (promotion, demotion ייספרו באמצעות updateNode).

updateTillRoot:

הפונקציה מקבלת צומת שממנה צריך להתחיל לעדכן. בכל איטרציה הפונקציה בודקת האם יש לצומת אבת אם כן תעלה אליו ותעדכן אותו גםץ כלומר נבצע עדכונים מהצומת הנתון ועד לשורש העץ.

סיבוכיות כוללת של פעולת הinsert- בהתאם לכלל הפונקציות המשוייכות אליה:

נראה כי הסיבוכיות הינה logn:

Place\_to\_insert, במקרה הגרוע תרד משורש העץ אל עבר העלה הרחוק ביותר – כגודל גובה העץ-

Rebalance, כפי שהוכחנו בכיתה promotion עשוי לגרור את הבעיה מעלה ואילו סיבוב/ סיבוב כפול יובילו לreturn - כלומר ליציאה מהרקורסיה.

לאחר פעולת הpromote יש קריאה לrebalance עם parent- כלומר לכל היותר קריאות כגובה העץ.

בrotation\_right/left מבצעים ניתוקים וחיבורים קבועים- לכל היותר 6 ולכן סיבוכיות הזמן קבועה - .

בעדכון הצמתים נבצע טיפוס בעץ כגודל גובהו סיבוכיות O(logn).

סה"כ .

Join:

Join:

פונקציית מעטפת אשר מקבלת 2 עצים וקודקוד וממפה למצבים:

\*אם 2 העצים ריקים נקבע את x להיות השורש ונעדכן את שדות המינימום והמקסימום.

\*אם אחד העצים ריק ניצור קודקוד פיקטיבי ונשלח לפונקצייה הפנימית בהתאם לסדר האיברים (smaller, x, bigger).

נעדכן את שדות המינימום והמקסימום בהתאם.

\*2 העצים לא ריקים- נבדוק באיזה סדר לשלוח לפונקצייה הפנימית בהתאם ל(smaller, x, bigger).

נעדכן את שדות המינימום והמקסימום בהתאם.

בכל אחד מהשלבים נחזיר את הפרש הגבהים פלוס אחד כסיבוכיות זמן הריצה של join.

נשלח בכל פעם לjoin\_in את שלשת הקודקודים וjoin\_in מחזירה את הקודקוד המיועד להיות שורש העץ המעודכן. לכן נעדכן את שורש העץ להיות join\_in(..).

Join\_in:

הפונקציה מחברת בין קודקודים בהתאם לאלגוריתם join שלמדנו בכיתה, תוך שימוש בrebalance ובעדכון שדות הsize והhight של הקודקודים בקריאה לפונקצייה updateTillRoot . בנוסף, הפונקצייה תחזיר את הקודקוד המיועד להיות שורש העץ המעודכן.

נחלק למקרים:

אם העצים ריקים נחזיר את X.

אם גבהי העצים שווה, נחבר את איקס לשני הקודקודים בהתאם למיקום (קטן, גדול) ונחזיר את X.

אחרת נבדוק מי מבין הגבהים גדול יותר ונטייל עד לקודקודים שגובהו שווה לגובה העץ הקטן. נחבר את איקס אליו ונעדכן כלפי מעלה.

לבסוף, נקרא לפונקצייה find\_root אשר תעלה במעלה העץ עד לקודקוד שההורה שלו הוא null- זה יהיה שורש העץ החדש.

סיבוכיות כוללת של פעולת join- בהתאם לכלל הפונקציות המשוייכות אליה:

נראה כי הסיבוכיות הינה (|hight(t1)-hight(t2)|+1):

בjoin – פונקציית המעטפת ישנן פעולות אריתמטיות ועדכוני שדה ב עבודה.

נחקור את סיבוכיות פונקציית join\_in:

\*ירידה בעץ בעל הגובה הגדול יותר עד לגובה העץ הקטן- |hight(t1)-hight(t2)|.

*\*חיבורי X לקודקודים – פעולות קבועות בO(1).*

*\*הקריאה לrebalance מהאב של X – ראינו כי הפונקצייה הינה בסיבוכיות logn כאשר – כגובה העץ. כעת המעבר הינו על גובה של* |hight(t1)-hight(t2)|.

עדכוני שדות הקודקודים מX ועד לשורש העץ- מעבר על |hight(t1)-hight(t2)| קודקודים.

סה"כ נקבל סיבוכיות .(|hight(t1)-hight(t2)|+1