דניאל פלדמן ודור מנדיל
danielf1, 302575436
dormendil, 200968873

תיעוד תרגיל מעשי 1 במבני נתונים

עץ אדום שחור

עץ אדום שחור הוא עץ חיפוש בינארי מאוזן בקירוב.

מחלקת RBTree

פונקציות פומביות

O(logn) - Insert

הפונקציה מקבלת מפתח וערך ומכניסה אותם לעץ, ושומרת על מבנה "האדום שחור" של העץ. מחזירה את מספר שינויי הצבע כפי שהוגדר ובמידה והמפתח קיים כבר בעץ הפונקציה מחזירה 1-.

בעקבות החיפוש של המפתח בעץ ושימוש ב- insertBalancer הסיבוכיות היא O(logn)

פונקציות עזר:

שם הפונקציה	תיאור	סיבוכיות
insertBalancer	פונקצית רקורסיבית שלאחר הכנסת האיבר דואגת לאיזון העץ ע"י שימוש בפונקציות הבאות:	O(logn)
insertCase1	מתפעלת את ה case 1	O(1)
insertCase2	מתפעלת את ה case 2	O(1)
insertCase3	מתפעלת את ה case 3	O(1)
updateMinMaxAfterInsertion	מעדכן את שדות המינימום והמקסימום בהתאם.	O(1)

O(logn) - Delete

הפונקציה מקבלת מפתח, מחפשת את האיבר המתאים בעץ ומוחקת אותו אם הוא קיים תוך כדי שהיא שומרת על איזור העץ "אדום שחור". במידה ומחקנו את האיבר המקסימלי או המינימלי, בעזרת פונקציות עזר (getSuccessor ו- getPredecessor) יש עדכון של השדות המתאימים.

לבסוף היא מחזירה את מספר שינויי הצבעים, במידה ולא התבצעה מחיקה היא מחזירה 1-.

בעקבות החיפוש של המפתח בעץ ושימוש של deleteNode, הסיבוכיות היא O(logn)

פונקציות עזר:

שם הפונקציה	תיאור	סיבוביות
deleteNode	מקבלת מצביע לצומת שאותה רוצים למחוק, מוחקת אותו ולאחר מכן מבצעת תיקון איזון ע"י הפונקציה הבאה.	בעקבות שימוש בפונקציה הבאה: O(logn)
deleteBalancer	מקבלת איבר שהוא double-black ומבצעת תיקון איזון עץ.	O(logn)

O(1) –getRoot

מחזיר את השורש של העץ במידה וקיים, null אם לא.

הוא מוחזק בשדה פנימי rootNode .

דניאל פלדמן ודור מנדיל
danielf1, 302575436
dormendil, 200968873

O(1) –empty

מחזירה true אם העץ ריק, false אם לא. משתמש בשדה nodesCount

Size

מחזירה את מספר האיברים בעץ ע"י שימוש בשדה nodesCount

O(1) – max -ו min

מחזירים את הערך של האיבר המקסימלי והמינימלי בהתאם. Null אם הם לא קיימים.

O(n) – keysToArray

מחזירה מערך ממוין המכיל את המפתחות בעץ. במידה והעץ ריק המערך יהיה ריק. הפונקציה לוקחת את האיבר המינימאלי השמור בשדה המתאים ומשתמשת בפונקצית עזר getSuccessor כדי לעבור על כל האיברים לפי הסדר.

סיבוכיות הפונקציה מושפעת מ- getSuccessor אשר סיבוכיות ה-amortize שלה הוא O(1) בקריאות עוקבות והיא מופעלת n פעמים ולכן עבור n קריאות עוקבות זה O(n)

O(n) -valuesToArray

מחזירה מערך ממוין לפי המפתחות של הערכים בעץ. במידה והעץ ריק המערך יהיה ריק. הפונקציה לוקחת את האיבר המינימאלי השמור בשדה המתאים ומשתמשת בפונקציית עזר getSuccessor כדי לעבור על כל האיברים לפי הסדר.

סיבוכיות הפונקציה מושפעת מ- getSuccessor אשר סיבוכיות ה-amortize שלה הוא O(1) בקריאות עוקבות והיא מופעלת n פעמים ולכן עבור n קריאות עוקבות זה O(n)

O(logn) - Search

מקבלת מפתח ומחזירה את הערך של האיבר המבוקש. Null אם הוא לא קיים.

משתמשת בפונקציה הפנימית searchKeyInSubTree אשר מחזירה אובייקט בסוג SearchKeyInSubTreeResult והיא תפורט בהמשך.

פונקציות עזר

O(logn) - searchKeyInSubTree

הפונקציה מקבלת תת-עץ המיוצג ע"י איבר RBNode ומפתח לחיפוש. היא מחפשת את המפתח בתת העץ ומחזירה אובייקט מסוג SearchKeyInSubTreeResult.

O(1) –rotateEx

מבצעת Rotate המתאים לכלל המצבים שאנו צריכים (הוספה ומחיקה). מקבלת איבר ואת האבא שלו ולפי היחסים של האיבר לאביו היא יודעת לבצע את ה- Rotate בכיוון הנכון.

O(1) –isParentLeftChild

מחזירה True במידה ואבא של האיבר הוא ילד שמאלי.

O(1) –isRedNode

מקבלת איבר (יכול לקבל גם null) ובמידה והאיבר הוא אדום, מחזירה True. במידה והיא מחזירה False, האיבר הוא שחור או עלה.

O(logn) – getPredecessor -ו getSuccessor

הפונקציות מחזירות את האיבר העוקב והקודם בהתאמה.

דניאל פלדמן ודור מנדיל
danielf1, 302575436
dormendil, 200968873

O(1) – resetColorSwitchCounter -ו setColorAndUpdateCounter

כדי לספור את מספר שינויי הצבע בצורה מדויקת בזמן תיקון העץ לאחר פעולות הוספה ומחיקה של איברים, יצרמו פונקציה אשר מעדכנת מונה (השמור כשדה במחלקה currentOperationSwitchColorCoutner) בכל פעולה של צביעת איבר. בזמן צביעת האיבר, הפונקציה בודקת האם היא משנה את צבעו, ובמידה וכן, היא מעלה את המונה. פעולת האיפוס מתבצעת בתחילת כל אחד מהפעולות הוספה ומחיקה.

O(1) – replaceNode

פונקציה אשר מחליפה איבר בעץ עם איבר אחר ע"י שינוי שדה ה- Child המתאים ושדה ה- Parent המתאים. האיבר המוחלף אינו מתעדכן אלא רק איבר האבא והאיבר החדש.

שדות של האובייקט

מוסברים כחלק מהתיעוד לפונקציות הרלונטיות.

rootNode – מצביע לשורש

minNode – מבציע למינמום

maxNode – מצביע למקסימום

nodesCount – גודל העץ

currentOperationSwitchColorCounter – מונה שסופר את מספר החלפות הצבע, מתאפס בתחילת כל פעולה delete או insert.

דניאל פלדמן ודור מנדיל
danielf1, 302575436
dormendil, 200968873

מחלקת RBNode

כחלק מהתרגיל המחלקה הזאת היתה קיימת והרחבנו אותה.

שדות פנימיים של RBNode

כשמם כן הם (לכל אחד קיים setter ו- getter):

Key
Value
leftNode
rightNode
parentNode
isRed

פונקציות עזר של RBNode

getBrother ו- getUncle – O(1)

פונקציות אלו יחזירו את האיברי האח והדוד בהתאם. הם יודעים למצוא את האיבר המתאים ע״י בדיקה של הצד של האיבר הרלוונטי. הכוונה היא האם האיבר הוא ילד שמאלי או ימני של אביו.

hasChildren – O(1)

האם לאיבר קיימים ילדים (לפחות אחד) או שלא.

isUncleRed – O(1)

האם האח של האבא הינו אדום.

isLeftChild – O(1)

האם האיבר הוא ילד שמאלי של אביו.

isRightChild – O(1)

האם האיבר הוא ילד ימיני של אביו.

SearchKeyInSubTreeResult

על-מנת שנוכל להשתמש בלוגיקת חיפוש אחת גם להוספה של איבר וגם למחיקה, יצרנו אובייקט אשר הפונקציה searchKeyInSubTree מחזירה והוא מכיל גם את האיבר שחיפשנו, אם הוא קיים, וגם את האיבר Parent שלו. כאשר אנחנו רוצים להוסיף איבר, נחפש אותו בעץ ובעזרת האובייקט המוחזר נוכל לדעת אם הוא קיים, ובמידה ולא, מתחת איזה איבר עלינו להוסיף את האיבר החדש.

פונקציות עזר לבדיקות

getMaxKey

getMinKey

getNodeByKey

דניאל פלדמן ודור מנדיל
danielf1, 302575436
dormendil, 200968873

בדיקות החלפות

#	מספר פעולות	ממוצע ל- Insert	ממוצע ל- Delete
1	10000	2.315	2.466
2	20000	2.328	2.472
3	30000	2.311	2.464
4	40000	2.311	2.460
5	50000	2.318	2.469
6	60000	2.316	2.461
7	70000	2.319	2.469
8	80000	2.315	2.463
9	90000	2.313	2.460
10	100000	2.325	2.471

ציפיות : אנחנו נצפה שממוצע שינויי הצבעים בכל פעולה לא ישתנה כתלות במספר האיברים (גובה העץ) מכיוון שלמדנו שפעולת איזון העץ בהכנסה ומחיקה היא ב- Amortize O(1). זה אומר שמספר שינוי שינויי הצבע, בין אם בעקבות rotate ובין אם בעקבות color flip יהיה קבוע ללא תלות בגודל הקלט.

אנו רואים לפי התוצאות שצדקנו.