**קובץ תיעוד**

|  |  |  |
| --- | --- | --- |
| שם מתודה | סיבוכיות | הסבר |
| Empty() | O(1) | לא מכיל לולאה/רקורסיה לכן קבוע |
| Retrieve() | logn | קריאה לפונקציית TreeSelectRec() והסיבוכיות נובעת משם. |
| TreeSelectRec()  Tree select –  הפונקציית מעטפת | logn | מקבלת אינדקס ומחזירה את האיבר באינדקס המתאים.  בכל קריאה רקורסיבית יש סיבוכיות זמן קבועה כי רק עושים מספר סופי של פעולות מתמטיות.  לכן הסיבוכיות תלויה בכמות הקריאות הרקורסיביות והן תלויות בגובה העץ, כיוון שעבור כל קלט הולכים לבן הימני/השמאלי.  במקרה הגרוע נגיע לעלה, כלומר יהיו כמות קריאות בגודל גובה העץ. כיוון שמדובר בעצי AVL גובה העץ יהיה logn. ולכן במקרה הגרוע יהיו logn קריאות רקורסיביות – כלומר הסיבוכיות הכוללת תהיה logn. |
| First() | (1) | לא מכיל לולאה/רקורסיה לכן קבוע |
| Last() | (1) | לא מכיל לולאה/רקורסיה לכן קבוע |
| getRoot() | O(1) |  |
| listToArray | O(n) | עברנו על כל צומת פעם אחת בלבד בעזרת פונקציה רקורסיבית דומה ל-tree\_walk ממבוא מורחב. בכל קריה רקורסיבית מתבצע פעולה של append, וזה מתקיים בזמן קבוע. אז סהכ יש n קריאות עם סיבוכיות O(1)- סהכ O(n) |
| Length() | O(1) | לא מכיל לולאה/רקורסיה לכן קבוע |
| Tree\_Rank() | O(log(n)) | מקבלת node ומחזירה את האידנקס של ה-node. הnode בהכרח נמצא בעץ.  בכל רמה בעץ אנחנו מבזבזים זמן קבוע לכן הסיבוכיות תלויה בגובה העץ- log(n) |
| Search() | O(n) | עוברים על כל הצמתים בעץ בפונקציה רקורסיבית פנימית ובכל קריאה רקורסיבית יש פעולות שמתקיימות בזמן קבוע. לכן יש n קריאות של O(1). לכן סיבוכיות הרקורסיה הינה O(n).  בסוף קוראים לtree\_rank אם נמצא הערך המבוקש. סיבוכיות הקריאה הינה O(log(n)). אז מטענה הסיבוכיות הכוללת זה O(n). |
| MinNode | O(log(n)) | אותו קוד כמו first, רק מחזיר AVLnode וניתן להתחיל עם כל node שנרצה (שלא וירטואלי) |
| getSuccessor | O(log(n)) | מימשנו כמו בשיעור\*\*\*\* |
| getPredecesor | O(log(n)) | מימשנו כמו בשיעור\*\*\*\* |
| getBF | O(1)) | לא מכיל לולאה/רקורסיה לכן קבוע |
| MaxNode | O(log(n) | אותו קוד כמו last, רק מחזיר AVLnode וניתן להתחיל עם כל node שנרצה (שלא וירטואלי) |
| updatePathMeasurements | O(log(n)) | עובר על כל צומת מהnode לroot, ומעדכן שדות גובה וגודל. |
| ImplementRotation | O(1) | לא מכיל לולאה/רקורסיה. יש מספר קבוע של קריאות לפונקציות שמתבצעות גם בזמן קבוע ( InsertRotation, LeftRotation) לכן בסך הכל זמן הריצה הוא קבוע |
| LeftRotation | O(1) | לא מכיל לולאה/רקורסיה לכן קבוע . (וכל הפעולות תבצעות בזמן קבוע בפונקציה). |
| RightRotation | O(1) | לא מכיל לולאה/רקורסיה לכן קבוע . (וכל הפעולות תבצעות בזמן קבוע בפונקציה). |
| Insert | O(log(n)) | לא מכיל לולאה/רקורסיה לכן הסיבוכיות היא כסיבוכיות הפעולות/קריאות לפונקציות שאנו משתמשים בהם בפונקציה-  MaxNode()  TreeSelectRec()  getPredecesor()  updatePathMeasurements()  getBF()  ImplementRotation ()  כל הפעולות בפונקציה הן בסיבוכיות לכל היותר ,O(log(n)) ואנו משתמשים בהן מספר קבוע של פעמים לכן בסך הכל נקבל O(log(n)). |
| Is\_left\_child | O(1) | מקבלים צומת עם הורה.  הפונקציה בודקת האם צומת הוא בן שמאלי או ימני- אם שמאלי מחזיר אמת. אם ימני מחזיר שקר.  לא מכיל לולאה/רקורסיה והפעולות שיש הן בזמן קבוע לכן הסיבוכיות היא O(1). |
| delete | O(log(n)) | מחיקת האיבר במקום ה-i ברשימה, אם הוא קיים. הפונקציה מחזירה את מספר (i(delete פעולות האיזון שנדרשו בסך הכל בשלב תיקון העץ על מנת לשמר את תכונת האיזון. אם לא קיימים מספיק איברים ברשימה הפונקציה מחזירה 1.  בתחילת הפונקציה אין לולאה/רקורסיה לכן הסיבוכיות היא כסיבוכיות הפעולות/קריאות לפונקציות שאנו משתמשים בהם בפונקציה-  TreeSelectRec()  getSuccessor()  updatePathMeasurements()  Delete\_style\_balancing  כל הפעולות אלה הן בסיבוכיות לכל היותר .,O(log(n)  לכן בסך הכל נקבל O(log(n)). |
| Delete\_style\_balancing |  | מבצעים את איזון לעץ החל מצומת מסויימת. הולכים מהצומת עד השורש ומאזנים.  יש לולאה אשר עושה log n איטרציות (לכל היותר) בגלל שאנחנו עולים בכל איטרציה להורה מהצומת המסויימת, בכל איטרציה יש פעולות בזמן קבוע/קריאה לפונקציות שהן בזמן קבוע –  ImplementRotation (),getBF()  לכן בסך הכל נקבל O(log(n)). |
| AVLDelete | O(1) | אין רקורסיה/ לולאה. פעולות בזמן קבוע |
| AVLjoin | O(log(n)) | הפונקציה מאחדת שני עצי AVL בהינתן צומת מקשר X.ומחזירה עץ מאוחד.  עושים מספר פעולות בזמן קבוע ואז קוראים לפונקציה פנימית join-  בפונקציה זו יש לולאה שמחפשת את הצומת בה הגובה בין שתי העצים בערך שווים – logn  לאחר מכן עושים מהצומת המקשרת עד לשורש איזונים – logn.  טיפול מקרה קצה: אם רשימה אחת ריקה, אז במקום להפעיל פעולות join רגילות, הפונקציה מבצעת insert באינדקס 0 או list.length. insert לוקת גם log(n) זמן, אז סיבוכיות לא מושפע.  לכן בסה"כ נקבל סיבוכיות של logn |
| Concat | O(||) | הפונקציה מקבלת רשימה. על הפונקציה לשרשר אותה אל סוף הרשימה הנוכחית.  הפונקציה מחזירה את הערך המוחלט של הפרש הגבהים של עצי הAVL שמוזגו.  בתחילת הפונקציה אין לולאה/רקורסיה לכן הסיבוכיות היא כסיבוכיות הפעולות/קריאות לפונקציות שאנו משתמשים בהם בפונקציה-  AVLjoin  לכן בסך הכל נקבל O(log(n)). |
| Split |  |  |
|  |  |  |

updateMeasurements

**לעדכן שזה סיבוכיות 0(1)**

**Fix size rec – לוג n – לעדכן שמתמשות בה גם בדליט וג באינסרט**

**מתודות של העצי -AVL ניתוח סיבוכיות**