תיעוד מחלקת WAVL Tree

שם: אופק דוד

שם משתמש:ofekdavid

ת.ז.: 205517865

שם: טל דרורי

שם משתמש:Drori1

ת.ז.: 312535917

|  |  |  |
| --- | --- | --- |
| **שם** | **הסבר** | **סיבוכיות** |
| WAVL Tree | | |
| Public WAVLTree() | בנאי המייצר עץ ריק ומאתחל את EXT\_NODE להיות WAVLNode ריק (ראה מתודת בנאי WAVLNode) ואת השדה root להיות EXT\_NODE | O(1) |
| Public Boolean empty() | בודקת אם השדה שורש הוא EXT\_NODE, כלומר אם העץ ריק מחזירה true, אחרת false | O(1) |
| Public String search(int k) | בודקת האם קיים איבר בעץ עם מפתח k. אם כן מחזירה את השדה info של אותו איבר, אחרת מחזירה null. המתודה משתמשת בתכונות העץ הבינארי ובכך עוברת על המסלול היחיד האפשרי למציאת איבר עם מפתח k. | O(logn) כאשר n הוא מספר האיברים בעץ |
| Public String min() | הפונקציה מחזירה את הערך של האיבר בעל מפתח מינמלי. אם העץ ריק מחזירה null. הפונקציה משתמשת בפונקציית min שפועלת על WAVLNode. | O(logn) |
| Private WAVLNode min(WAVLNode x) | הפונקציה מחזירה את האיבר בעל המפתח המינמלי, בעזרת ירידה שמאלה בעץ עד הגעה לעלה. | O(logn) |
| Public String max() | הפונקציה מחזירה את הערך של האיבר בעל מפתח מקסימלי, על ידי ירידה ימינה בעץ עד הגעה לעלה. אם העץ ריק מחזירה null | O(logn) |
| Public int[] keysToArray() | הפונקציה מחזירה מערך של ממוין של כל מפתחות העץ. הפונקציה קוראת לפונקציה הרקורסיבית keysToArray2 עם הפרמטרים root,arr,0, כאשר arr בגודל size של העץ. אם העץ ריק מחזיקה מערך ריק. | O(n) |
| Private int[] keysToArray(WAVLNode x, int[] arr, int index) | הפונקציה מבצעת רישום של מפתחות האיברים בעזרת מיון inorder רקורסיבי. הפונקציה מחזירה את האינדקס לרישום מפתח האיבר במערך arr. | O(n) |
| Public String[] infoToArray() | הפונקציה מחזירה מערך של מחרוזות שהן כל הערכים שבעץ, ממוינים על פי סדר המפתחות. הפונקציה קוראת לפונקציה הרקורסיבית infoToArray2, עם הפרמטרים root,arr,0, כאשר arr בגודל size של העץ. אם העץ ריק מחזיקה מערך ריק. | O(n) |
| Private String[] infoToArray2(WAVLNode x, String[] arr, int index) | הפונקציה מבצעת רישום של ערכי האיברים בעזרת מיון inorder רקורסיבי על ערכי המפתחות. הפונקציה מחזירה את האינדקס לרישום ערך האיבר במערך arr. | O(n) |
| Public int size() | מחזירה את מספר האיברים בעץ על ידי קריאה לשדה sizen של שורש העץ. | O(1) |
| Public String select (int i) | מחזירה את ערך האיבר הi בגודלו בעץ(הספירה מתחילה מ-1 לפי ההוראות שניתנו בפורום). משתמשת בselect\_rec (ראה מתודה) | O(logn) |
| Public String(WAVLNode root , int i) | המתודה מקבל צומת בעץ(תת עץ) וערך i ומחזירה את האיבר הi בגודלו בתת העץ שניתן. הפונקציה בודקת לי גודל הסייז של הבן השמאלי האם לחזיר את השורש, להמשיך עם תת העץ השמאלי עם אותו הערך i, או להמשיך עם תת העץ הימני עבור ערך i קטן יותר כדי להתאים זאת לכל הצמתים שהיו והפסקנו להסתכל עליהם (השורש ותת העץ השמאלי). | O(logn) |
| Public WAVLNode getRoot() | מחזירה את שורש העץ על ידי קריאה לשדה root של המופע. | O(1) |
| insert | | |
| Public int insert (int k, String i) | המתודה מכניסה איבר בעל מפתח k וערך i לעץ. ראשית המתודה מייצרת WAVLNode עם המפתח והערך הרלוונטיים, לאחר מכן מחפשת האם קיים Node עם מפתח כזה בעץ (ראה treepos) אם קיים המתודה מחזירה -1 וסיימנו. אחרת ממתודת treepos קיבלנו את המקום הנכון להכנסה. מכניסים את האיבר למקום המתאים ומבצעים rebalance בהתאם בלולאת While, כאשר תנאי הלולאה הוא השוואת הrank של האיבר עליו אנו מצביעים ואבא שלו (כל עוד אבא שלו לא null). פעולת הrebalance תתבצע בהתאם להשוואת הrank הרלוונטים (ראה מתודות איזון בהמשך ומתודת מציאת צד הבעיה). המתודה מחזירה את מספר פעולות האיזון שנדרשו על מנת להכניס את האיבר. אם האיבר היה קיים בעץ, היא מחזירה -1. | O(logn) כאשר n הוא מספר התאים בעץ. |
| Private WAVLNode treePos(WAVLTree tree, int key) | המתודה מחפשת את המיקום המתאים להכנסת איבר עם מפתח k לעץ tree. המתודה מחפשת את המיקום כשהיא משתמשת בתכונות של עץ חיפוש בינארי ותוך כדי מעלה את שדה sizen של כל WAVLNode שהיא עוברת דרכו. אם לא מצאה איבר עם מפתח k, מחזירה את המקום הרלוונטי אליו צריך להכניס את האיבר. אם מצאה איבר עם מפתח k, משמע קיים איבר בעץ עם המפתח של האיבר שרצינו להכניס, לכן המתודה קוראת לrefix (ראה מתודה, על מנת לתקן את הsizen) ומחזירה את האיבר עצמו. | O(logn) |
| Private void refix(WAVLNode x) | המתודה מתקנת את הsizen שהעלנו לכל WAVLNode במהלך treePos במידה והאיבר שניסינו להכניס בפעולת insert היה קיים בעץ. המתודה עוברת על אותו מסלול דרך קריאה להורה עד שמגיעה לשורש. | O(logn) |
| Private String parentside(WAVLNode d,WAVLNode y) | מקבלת 2 צמתים d ו y, ומחזירה "right" אם y הוא הבן הימני של d, אחרת "left". | O(1) |
| פעולות איזון insert | | |
| Private void promote(WAVLNode x) | פעולת זו נקראת במידה ובלולאת הwhile בדקנו את התנאי המתאים לכך בהתחשב בהפרשי הrank המתאימים. המתודה מעלה את הrank של הparent ב-1. (ע"פ המתואר בשקף 23) | O(1) |
| Private void rightRotate (WAVLNode x) | מסובבת ימינה את תת העץ של הצומת הנתון, ע"פ האלגוריתם הנתון בשקף 26. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי) | O(1) |
| Private void leftRotate (WAVLNode x) | מסובבת שמאלה את תת העץ של הצומת הנתון, ע"פ האלגוריתם הנתון בשקף 26 [הבעיה הסימטרית]. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי) | O(1) |
| Private void doubleRotateright(WAVLNode x) | מבצעת סיבוב כפול ימינה לתת העץ של הצומת הנתון, ע"פ האלגוריתם הנתון בשקפים 25,27. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי) | O(1) |
| Private void doubleRotateleft(WAVLNode x) | מבצעת סיבוב כפול שמאלה לתת העץ של הצומת הנתון, ע"פ האלגוריתם הנתון בשקפים 25,27 [הבעיה הסימטרית]. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי) | O(1) |
| delete | | |
| Public int delete(int k) | הפונקציה קוראת לפונקצייה treePosForDel, אשר מחזירה את האיבר עם מפתח k , אם קיים, אותו אנחנו רוצים למחוק ומסדרת את שדה הsizen בהתאם. אם לא מצאה, האיבר שלנו לא קיים בעץ ונחזיר -1 מבלי לשנות את העץ.  הפונקציה מוחקת את האיבר המתאים אם נמצא בהתחשב בכמה תנאים:   1. האיבר הוא שורש שהוא עלה (מטופל במקרי קצה בתחילת הפונקציה). 2. האיבר הוא שורש שהוא אונרי (מטופל במקרי קצה בתחילת הפונקציה). 3. האיבר הוא איבר פנימי (מטופל בעזרת פונקציית successor וswap, ומעבר לאחד מתנאים 4 או 5. 4. האיבר הוא עלה. 5. האיבר הוא אונרי.   מתבצעת מחיקה של האיבר ופעולות איזון בהתאם בלולאת While לפי התנאים המתאימים עד קבלת עץ מאוזן (ראה פונקציות איזון בהמשך). | O(logn) |
| Private WAVLNode treePosForDel(WAVLTree, int k) | המתודה מחפשת את האיבר בעל מפתח k בעץ tree. תוך כדי החיפוש מתקנת את שדה הsizen של כל WAVLNode שעוברת דרכו. במידה ולא מצאה איבר עם מפתח k, קוראת לreFixForDel (ראה מתודה) ומחזירה את האיבר האחרון בשרשרת החיפוש בעץ הבינארי. | O(logn) |
| Private void reFixForDel (WAVLNode y) | המתודה מופעלת אם לא נמצא האיבר למחיקה ומטרתה לתקן את הsizen ששינינו בtreePosForDel. המתודה מטפסת מעלה בעץ בעזרת קריאה לparent עד הגעה לשורש. | O(logn) |
| Private boolean isInnerNode (WAVLNode x) | הפונקציה בודקת אם האיבר הוא איבר פנימי בעזרת בדיקת הבנים שלו. מחזירה true אם כן, אחרת מחזירה false. | O(1) |
| Private WAVLNode successorForDel (WAVLNode x) | הפונקציה מופעלת אם x איבר פנימי לשם פעולת המחיקה המתוארת באלוגריתם במצגת. הפונקציה מחזירה את הsuccessor של האיבר. | O(logn) |
| Private WAVLNode swap (WANLNode x, WAVLNode z) | הפונקציה מופעלת על x שהוא צומת פנימי וz שהוא הsuccessor שלו. הפונקציה שמה את z במקום של x (ובכך מוחקת את x), משכפלת את z ושמה את השכפול במקום של z) ומחזירה את השכפול לשם המשך מחיקה בפונקציית delete. הפונקציה קוראת לparentside לשם ביצוע התיקון המתאים בהצבעות. | O(1) |
| Private Boolean onaryLeft (WAVLNode z) | מקבלת צומת z ובודקת האם הצומת היא צומת אונארי שמאלי, אם כן מחזריה true אחרת false (מקרה 5 ב delete) | O(1) |
| פעולות איזון delete | | |
| Private void demote(WAVLNode z) | פעולת זו נקראת במידה ובלולאת הwhile בדקנו את התנאי המתאים לכך בהתחשב בהפרשי הrank המתאימים. המתודה מורידה את הrank של z ב-1. (ע"פ המתואר בשקף 55) | O(1) |
| Private void doubleDemoteLeft(WAVLNode z) | פעולת זו נקראת במידה ובלולאת הwhile בדקנו את התנאי המתאים לכך בהתחשב בהפרשי הrank המתאימים. המתודה מורידה את הrank של z ב-1 ושל הבן הימני של z. (ע"פ המתואר בשקף 56) | O(1) |
| Private void doubleDemoteRight(WAVLNode z) | פעולת זו נקראת במידה ובלולאת הwhile בדקנו את התנאי המתאים לכך בהתחשב בהפרשי הrank המתאימים. המתודה מורידה את הrank של z ב-1 ושל הבן השמאלי של z. (המקרה הסימטרי למתואר בשקף 56) | O(1) |
| Private void rotateLefttDel (WAVLNode Z) | מסובבת שמאלה את תת העץ של הצומת הנתון, ע"פ האלגוריתם הנתון בשקף 57. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי). | O(1) |
| Private void rotateRightDel (WAVLNode Z) | מסובבת ימינה את תת העץ של הצומת הנתון, ע"פ האלגוריתם הסימטרי לנתון בשקף 57. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי). | O(1) |
| Private void doubleRotateleftDel (WAVLNode Z) | מבצעת סיבוב כפול שמאלה לתת העץ של הצומת הנתון, ע"פ האלגוריתם הנתון בשקף 58. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי). | O(1) |
| Private void doubleRotateRightDel (WAVLNode Z) | מבצעת סיבוב כפול ימינה לתת העץ של הצומת הנתון, ע"פ האלגוריתם הסימטרי לנתון בשקף 58. בנוסף מבצעת קריאה למתודה parentSide (ראה מתודה, על מנת לבדוק האם x בן ימני או שמאלי). | O(1) |
| WAVL Node | | |
| Public WAVLNode(int key1, String info1) | בנאי. מאתחל את השדה key לkey1 ואת info לinfo1, את sizen ל1, rank ל 0 ואת כל שאר השדות לnull. | O(1) |
| Public WAVLNode() | בנאי. מאתחל את sizen ל0, rank ל-1, ואת כל השאר לnull | O(1) |
| Public int getKey() | מחזירה את שדה key של המופע | O(1) |
| Public String getValue() | מחזירה את שדה value של המופע | O(1) |
| Public WAVLNode getLeft() | מחזירה את הבן השמאלי של המופע | O(1) |
| Public WAVLNode getRight() | מחזירה את הבן הימני של המופע | O(1) |
| Public Boolean isInnernode() | מחזירה true אם הצומת הוא צומת פנימי בעץ | O(1) |
| Public int getSubtreeSize() | מחזירה את השדה sizen של המופע. | O(1) |

**מדידות**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר סידורי | מספר פעולות | מספר פעולות האיזון הממוצע לפעולת insert | מספר פעולות האיזון הממוצע לפעולת delete | מספר פעולות האיזון המקסימאלי לפעולת insert | מספר פעולות האיזון המקסימאלי לפעולת delete |
| 1 | 10,000 | 3.366 | 2.6678 | 15 | 11 |
| 2 | 20,000 | 3.3721 | 2.6398 | 16 | 10 |
| 3 | 30,000 | 3.353 | 2.6301 | 19 | 11 |
| 4 | 40,000 | 3.3583 | 2.6078 | 18 | 13 |
| 5 | 50,000 | 3.3311 | 2.60548 | 17 | 11 |
| 6 | 60,000 | 3.30275 | 2.58664 | 18 | 12 |
| 7 | 70,000 | 3.2937 | 2.5720 | 20 | 31 |
| 8 | ­80,000 | 3.29513 | 2.5064 | 19 | 13 |
| 9 | 90,000 | 3.2667 | 2.537 | 20 | 12 |
| 10 | 100,000 | 3.23632 | 2.52938 | 20 | 13 |

**הסבר תוצאות המדידות:**

תוצאות המדידות תואמות לציפיות. למדנו בכיתה כי הזמן הממוצע (amortized) של פעולות הrebalance לאחר insert\delete הן O(1). ניתן לראות כי זה מסתדר עם תוצאות מדידת מספר הפעולות הממוצע (עמודות 3,4) שכן הזמן נשאר קבוע ללא תלות במספר האיברים למעט שינויים זעירים.

כמו כן, למדנו כי זמן המקסימאלי של פעולת איזון חסום על ידי . ניתן לראות כי דבר זה מתקיים עבור כל אחת מהמדידות.