עידו פביאן 208604660

ניב שגיא טננבאום 206170219

**תרגיל מעשי 1**

תיעוד הקוד

Retrieve(i) – Olog(n)

הפונקציה מבצעת בדיקות תקינות של הקלט ולאחר מכן משתמשת בפונקציה רקורסיבית retrieve\_node\_rec. פונקציה זו מקבלת צומת ואינדקס ולאחר מכן יש שלוש אפשרויות:  
במידה וגודל העץ של בנו השמאלי שווה לאינדקס היא מחזירה את הצומת.  
במידה וגודל תת העץ השמאלי גדול יותר מהאינדקס היא מבצעת קריאה רקורסיבית לבן השמאלי של הצומת שבה היינו עם אותו האינדקס.  
לבסוף במידה ואף אחת מהאופציות הנ"ל לא התקיימו הפונקציה תקרא רקורסיבית לבן הימני תוך החסרה מהאינדקס של גודל תת העץ השמאלי ועוד אחד.  
בכל קריאה לretrieve\_node\_rec אנחנו יורדים לאחד הבנים או מחזירים צומת ולכן עומק העץ הינו לכל היותר כגובה העץ משמע Olog(n). בכל פעם מתבצעת O1 של פעולות ולכן הפונקציה retrieve\_node\_rec מתקיימת בסיבוכיות של log(n) ומכאן גם Retrieve בסיבוכיות זהה.

Delete(i) – Olog(n)

הפונקציה תחילה מבצעת בדיקה של חוקיות הקלט ולאחר מכן במידה ומדובר בעץ בגודל קטן או שווה לשתיים מבצעת פעולות שעלותן קבועה (עדכוני מצביעים עבור הצומת שנותרה או לחילופין "איפוס" של העץ). במידה וגודל העץ גדול מ2 ניגש לאיבר במיקום ה-i בעלות Olog(n) ונשמור את כל המצביעים שלו.  
אם מדובר בעלה נבצע מחיקה שלו במספר קבוע של פעולות ולאחר מכן נעלה במעלה העץ בפונקציית fixtree ועלותה Olog (n) אשר תפורט בהמשך. במידה ומדובר באיבר שלו בן יחיד נבצע "מעקף" בעלות קבועה של פעולות ולאחר מכן נבצע fixtree במעלה העץ.  
במידה ומדובר בצומת שלה שני בנים נמצא את הsuccessor שלה, נחליף את ערכי הצומת בערכי צומת ה הsuccessor ונמחק רקורסיבית את צומת ה הsuccessor. כמו כן נעדכן מצביעים של first ו-last במידת הצורך.

Last() – O(1)

ניגש למצביע ששמרנו בעץ של maxnode ונחזיר אותו

Length() – O(1)

ניגש למצביע ששמרנו בעץ של size ונחזיר אותו

Sort() – O(nlog(n))

תחילה נמיר את העץ הנוכחי לרשימה בעלות של n ונייצר עץ AVL שהשורש שלו הוא האיבר הראשון ברשימה. כעת, נכניס את כל איברי הרשימה לפי הסדר לעץ החדש באמצעות הפונקציה Insert\_sort.  
פונקציה זו מקבלת צומת הכנסה וצומת בדיקה, במידה וערך צומת הבדיקה גדול לקסיקוגרפית מצומת ההכנסה היא תבדוק האם לצומת הבדיקה בן שמאלי. אם לצומת הבדיקה בן שמאלי היא תקרא לעצמה רקורסיבית עם הבן השמאלי. אחרת, הפונקציה תכניס את צומת ההכנסה להיות הבן השמאלי של פונקציית הבדיקה ותבצע fixtree. במקרה וצומת הבדיקה קטנה מצומת ההכנסה נקבל מקרה "מראה" הסימטרי למקרה שצויין.  
הפונקציה Insert\_sort יורדת במורד העץ כל פעם ולכן עומק הרקורסיה של הוא Olog(n) כמו כן fixtree גם היא בעלות Olog(n). נבחין כי אנו מבצעים את הקריאה לפונקציה insert\_sort עבור כל איברי הפונקציה ומכאן נקבל O(nlog(n)).

Concat(lst) – O(log(n)-log(m))

הפונקציה מבצעת מספר בדיקות בעלות קבועה ולאחר מכן בודקת איזה מהרשימות ארוכה יותר, self או lst. הפונקציה שולחת לפונקציה concat\_fixed את הרשימה הארוכה, הקצרה ומציינת האם הארוכה היא self או lst. הפונקציה concat\_fixed שולחת את הרשימה הארוכה לפונקציה find\_node\_in\_correct\_height יחד עם גובה העץ של הרשימה הקצרה וערך המציינת האם הגבוהה היא self או לא.   
פונקציה זו רצה על הרשימה הגבוהה כאשר בכל פעם היא פונה רק ימינה או רק שמאלה בהתאם למצביע המציין האם self היא הגבוהה ומחזירה את הצומת הראשונה שגובהה קטן או שווה לגובה העץ הקטן יחד עם האינדקס שלו. עלות פונקציה זו היא לכל היותר כגובה העץ הגדול פחות גובה העץ הקטן משמע O(log(n)-log(m)) . נקרא לצומת המוחזרת b.   
לאחר מכן הפונקציה concat\_fixed מייצרת צומת שלה נקרה x ושומרת את ההורה של הצומת שאליה הגענו בפונקציה הקודמת. כעת במידה והעץ הגבוה היה self נבצע מספר פעולות שבסופן ההורה של x יהיה האב של הצומת שחיפשנו מקודם, הבן השמאלי יהיה הצומת שחיפשנו מקודם והן הימני שורש העץ הנמוך. במידה וself הוא הנמוך נקבל את המקרה הסימטרי וההפוך. כל הפעולות הללו לוקחות זמן קבוע. כעת, נחזיר את הצומת b.  
לבסוף פונקציית הconcat מחזיקה כעת בצומת b ובעץ שאינו בהכרח נראה כעץ AVL תקין שבו צומת לא רצויה אך ידוע לנו שערכי self קטנים מערכי lst ו-xבינהם. נבצע את פונקציית fixtree בעלות O(log(n)-log(m)) כי אנחנו מתחילים מצומת x ולאחר מכן נמחק את הצומת x בעלות של O(log(n)-log(m)). כך נקבל כי עלות הפונקציה כולה היא O(log(n)-log(m))

Search(Val) – O(n)

הפונקציה מייצרת רשימה באמצעות listToArray בעלות של O(n) ולאחר מכן עוברת על כל איברי הרשימה ומשווה את הערך שהוכנס לפונקציה עבור כל אחד מהערכים עד שהיא מוצאת את הערך המבוקש.  
ניתן היה לשמור עץ נוסף המחזיק את הערכים לפי סדרם הלקסיקוגרפי ולהחזיר ערך זה בעלות של O(log(n)) אך מבחינת זיכרון היינו מקבלים מבנה מורכב מאוד בעיקר בתחומי הבדיקה שאנו מבצעים בחלק התיאורטי וזמני הריצה היו מוכפלים.

חלק תיאורטי

|  |  |  |  |
| --- | --- | --- | --- |
| **ניסוי 3 – הכנסות ומחיקות לסירוגין** | **ניסוי 2 - מחיקות** | **ניסוי 1 - הכנסות** | **מספר סידורי i** |
| 1427 | 1151 | 2046 | **1** |
|  |  |  | **2** |
| 5629 | 4429 | 8398 | **3** |
| 11590 | 8738 | 16741 | **4** |
| 23143 | 17935 | 33836 | **5** |
| 46194 | 35615 | 66808 | **6** |
|  |  |  | **7** |
|  |  |  | **8** |
|  |  |  | **9** |
|  |  |  | **10** |