סיבוכיות

|  |  |  |
| --- | --- | --- |
| פונקציה | סיבוכיות | הסבר |
| Empty | O(1) | בדיקה של קיום שורש – באם לא קיים (השורש מוגדר כשדה של מחלקת הAVLTree) |
| Search  recursiveSearch | O(logn) | חיפוש בינארי סטנדרטי – בכל צומת לבחור האם לרדת ימינה או שמאלה |
| Insert | O(logn) | התהליך מורכב משלושה שלבים:  1)חיפוש סטנדרטי  2) הכנסת הצומת פיזית O(1)  3) לולאה שעולה במעלה הדרך עד לשורש. אורך הדרך לכל היותר O(logn) כגובה העץ.  4) בכל צומת בחינה האם יש צורך בסיבובים. |
| Delete | O(logn) |  |
| Min | O(logn) | הפונקציה יורדת בלולאה עד לעלה השמאלי ביותר. מאחר שהעץ מאוזן, במקרה הגרוע ביותר הלולאה תיאלץ לפעול לאורך כמות צמתים השווה לגובה העץ – logn |
| Max | O(logn) | הפונקציה יורדת בלולאה עד לעלה הימני ביותר. מאחר שהעץ מאוזן, במקרה הגרוע ביותר הלולאה תיאלץ לפעול לאורך כמות צמתים השווה לגובה העץ – logn |
| keysToArray | O(n) | מעבר inorder על כל צמתי העץ. מבקרים בכל אחת מהצמתים פעם אחת בלבד. |
| infoToArray | O(n) | בדומה לפונקציה הקודמת, גם כאן מתבצע מעבר inorder על כל צמתי העץ ולאחר מכן העתקה שלו למערך המכיל את הערכים. מבקרים בכל אחת מהצמתים פעם אחת בלבד. |
| Size | O(1) | שדה של העץ שמתעדכן בכל מחיקה והכנסה של צומת חדש. |
| getRoot | O(1) | פעולה אחת של שליפת המידע מתא בזיכרון. |
| successor | O(logn) | הפונקציה בודקת קודם כל האם לצומת קיים בן ימני. במידה וכן ניקח את האיבר המינימלי מתת עץ שהבן המיני הוא השורש שלו – פעולה שלכל היותר חסומה על ידי logn שהוא גובה העץ הראשי. במידה ואין לו בן ימני, הפונקציה תחפש במעלה הדרך את הצומת הראשון שאין לו בן ימני (ולכל היותר תעלה logn רמות) כך שבכל רמה תתבצע פעולת השוואה אחת. |
| perfixXor | O(logn) |  |
| succPerfixXor | O(nlogn) | בתחילת הפונקציה יוצרים את מערך המפחות – פעולה שלוקחת O(n). לאחר מכן מבצעים חיפוש בינארי על כל אחד מאיברים הקטנים מk שהוזן. במקרה הגרוע זה יהיה nlogn פעולות. |

מדידות

* 1. עבור הפונקציה (היעילה) prefixXor (התוצאות מוצגות בננו-שניות):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מספר סידורי | עלות prefixXor ממוצעת )כל הקריאות( | עלות succPrefixXor ממוצעת )כל הקריאות( | עלות prefixXor ממוצעת )100 קריאות ראשונות( | עלות succPrefixXor ממוצעת )100 קריאות ראשונות( |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

* 1. עבור הפונקציה (הלא יעילה) succPrefixXor (התוצאות מוצגות בננו-שניות):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מספר סידורי | עלות prefixXor ממוצעת )כל הקריאות( | עלות succPrefixXor ממוצעת )כל הקריאות( | עלות prefixXor ממוצעת)100 קריאות ראשונות( | עלות succPrefixXor ממוצעת ) 100קריאות ראשונות( |
| 1 | 11609 | 15924 | 12050 |  |
| 2 | 8578 | 11930 | 7643 |  |
| 3 | 13193 | 16816 | 8479 |  |
| 4 | 19332 | 24762 | 11123 |  |
| 5 | 22337 | 31409 | 13851 |  |

1. השוואת עץ AVL עם עץ בינארי רגיל:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| מספר סידורי i | עץ AVL סדרה חשבונית | עץ ללא מנגנון איזון סדרה חשבונית | עץ AVL סדרה מאוזנת | עץ ללא מנגנון איזון סדרה מאוזנת | עץ AVL סדרה אקראית | עץ ללא מנגנון איזון סדרה אקראית |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |