ת.ז. 325149383

ת.ז.

קובץ תיעוד מחלקת AVLTree

תיאור המחלקה: עץ חיפוש בינארי AVL.

מתודות המחלקה

search: המתודה מקבלת ערך key, ומחזירה את המצביע ל-node עם הערך המתאים. אם לא נמצא ערך כזה המתודה תחזיר None.

סיבוכיות: המתודה מחפשת באופן איטרטיבי מהשורש עד לעלים, כבחיפוש בעץ בינארי רגיל. לפיכך נסיק כי סיבוכיות זמן הריצה לינארית בגובה של העץ - $O(h)$. מכיוון שזהו עץ חיפוש מאוזן, גובה העצה לוגריתמי במספר הצמתים ולכן נסיק $O(\log n)$.

insert: המתודה מקבלת ערכי key,value יוצרת צומת חדשה ומכניסה אותה לעץ. בסיום, המתודה "מתקנת" את העץ, ומחזירה את מספר פעולות הגלגול שביצעה.

סיבוכיות: נחלק לשלבים.

- בשלב הראשון, נבצע הכנסה רגילה לעץ. ההכנסה תהיה כבן לעלה מתאים, ולכן נצטרך לרדת מהשורש לעלה מתאים - $O(h)$, והיות והעץ מאוזן - $O(\log n)$.
- בשלב השני נקרא למתודה fix_subtree מהצומת שהכנסנו. fix_subtree רצה ב-$O(d)$, d עומק הצומת שקיבלה. העומק שהכנסנו הוא לכל היותר גובה העץ, והיות ומדובר בעץ מאוזן נסיק $O(\log n)$.

נסכום את הסיבוכיות בכל אחד מהשלבים ונקבל - $O(\log n)$

delete: המתודה מקבלת מצביע לצומת, ומוחקת אתו מהעץ. בסיום, המתודה "מתקנת" את העץ, ומחזירה את מספר הגלגולים שביצעה.

סיבוכיות: נחלק למקרים:

- אם לצומת אין בן ימני או שמאלי, נחבר את הבן הנותר לאב הצומת המקורית במקומה, ונתקן משם במעלה העץ באמצעות fix_subtree שרצה בזמן $O(\log n)$
- אם לצומת שני בנים, נמצא את העוקב באמצעות find_successor שרצה בזמן $O(\log n)$. נחליף במיקומים בין העוקב לבין הצומת המקורית, ונקרא שוב למתודה מהמיקום החדש.

נבחין כי לאחר המקרה השני, אנחנו נהיה במקרה הראשון בקריאה השנייה, שכן אם היה לו בן ימני אז הוא היה העוקב. לכן, במקרה הגרוע נכנס במקרה השני, ולאחר מכן למקרה הראשון. לפיכך נוכל לסכום ולקבל - $O(\log n)$.

avl_to_array: מתודה שמחזירה מערך של איברי העץ כ-tuple של ה-(key,value).

סיבוכיות: המתודה קוראת למתודת עזר רקורסיבית recAvlToArr שרצה בזמן $O(\log n)$

recAvlToArr: מתודת עזר שנקראית ע"י avl_to_array, מקבלת צומת ומערך, ומחזירה מערך ממוין של תת העץ.

סיבוכיות: נאמר שגודל תת העץ של הצומת הוא n. נבקר בכל צומת פעם אחת - $O(n)$, ובכל צומת נוסיף לסוף הרשימה את הערך והמפתח של הצומת. הוספה לסוף הרשימה היא ב-$O(1)$ ולכן נכפול ונסיק - $O(n)$ סיבוכיות זמן הריצה של המתודה.

size: המתודה מחזירה את גודל העץ. היא ניגשת למצביע לשורש ומחזירה את שדה הגודל של הצומת. אם אין שורש, המתודה תחזיר אפס.

rank: המתודה מקבלת מצביע לצומת ותחזיר את הדרגה של הצומת בעץ (מקומה במערך ממוין). המתודה יורדת מהשורש עד לצומת, ובכל פעם שתרד ימינה, תוסיף את גודל תת העץ השמאלי (שכל איבריו קטנים מהצומת).

סיבוכיות: המתודה מתחילה בשורש העץ ובכל איטרציה יורדת רמה. לכן, נסיק כי זמן הריצה לינארי בגובה העץ, והיות והעץ מאוזן לוגריתמית במספר הצמתים - $O(\log n)$

select: המתודה מקבלת מספר i, ומחזירה מצביע לצומת בעל הדרגה i. נתחיל את החיפוש מהשורש, ובכל פעם נבדוק אם גודל תת העץ השמאלי קטן שווה לדרגה. אם לא נחסר מ-i את הגודל ונרד ימינה, אחרת נרד שמאלה.

סיבוכיות: נבחין כי המתודה מתחילה בשורש ויורדת בכל פעם רמה אחת בעץ באופן איטרטיבי. לכן זמן הריצה לינארי בגובה העץ, ובדומה לקודם לוגריתמי במספר הצמתים - $O(\log n)$

max_range: המתודה מקבלת שני מספרים a,b , ותחזיר מצביע לאיבר עם ערך ה-value הכי גבוה בתחום המפתחות בין a ל-b.

סיבוכיות: נחלק לשלבים

- תחילה נשיג מערך ממוין של איברי העץ באמצעות avl_to_array - $O(n)$
- נבדוק אם המפתח בתחום ואם ערך ה-value מקסימלי - $O(n)$
- נחזיר מצביע לצומת המקסימלית שמצאנו - $O(\log n)$

נסכום ונקבל - $O(n)$.

get_root: מחזירה מצביע לשורש העץ

find_successor: המתודה מקבלת מצביע לצומת בעץ ומחזירה מצביע לעוקב. אם אין, המתודה תחזיר None.

סיבוכיות: נחלק למקרים

- אם קיים בן ימני, רד עליו, ואז לך ימינה עד הסוף.
- אם לא קיים בן ימני, עלה במעלה העץ עד שקיים אב גדול יותר.\

נבחין כי בכל מקרה אנחנו או (אקסקלוסיבי) עולים במעלה העץ, או יורדים בעץ, ולכן זמן הריצה במקרה הגרוע לינארי בגובה העץ, ולפיכך $O(\log n)$ במקרה הגרוע.

fix_subtee: המתודה מקבל מצביע ולצומת, ו"מתקנת" את העץ במעלה העץ מהצומת עד לשורש. בסיום ריצתה המתודה מחזירה את מספר הגלגולים שביצעה.

סיבוכיות:

- מספר האיטרציות: נבחין כי בכל איטרציה נעלה לאב מהצומת המקורית, עד שנגיע לשורש. לכן, מספר האיטרציות הוא $O(d)$ כאשר d עומק הצומת ההתחלתית. היות והעץ מאוזן נסיק $O(\log n)$ איטרציות לכל היותר.
- בכל איטרציה נקרא ל-calc_balance_factor שרצה בזמן קבוע, ונבצע גלגול אחד או שניים, כל אחד מהם גם רץ בזמן קבוע. לכן כל איטרציה רצה ב-$O(1)$.

נכפול ונקבל כי זמן הריצה הסופי - $O(\log n)$.

update_height: המתודה מקבלת מצביעה לצומת ומעדכנת את הגובה של הצומת.

calc_balance_factor: המתודה מקבלת מצביע לצומת ומחזירה את ה-balance factor של הצומת.

rightRotation: המתודה מקבלת מצביע לצומת ומבצעת "גלגול לימין" של הצומת כמו שמתואר בהרצאות.

leftRotation: המתודה מקבלת מצביע לצומת ומבצעת "גלגול לשמאל" של הצומת כמו שמתואר בהרצאות.