מבנ"ת 6

שחר פרץ

28 באפריל 2025

מרצה: עמית ווינשטיין

הבעיה בתחזוק עומק: כאשר עושים רוטציה יכול להיות שיש כמות לינארית של צמתים בעץ שעבורם צריך לעדכן את המידע הזה. אפשר אולי לעשות עדכונים בצורה lazy ואז זה יעבוד, אבל זה לא הדיון שלנו לבינתיים. (הערה: בעיה בעדוכנים שהם lazy היא שלא תמיד מתחילים מהשורש ואז אי אפשר להניח איפה תעבור).

תנאי מספיק אך לא הכרחי הוא שאם התכונה תלויה אך ורק בבנים הישירים תהיה ניתנת לתחזוק (אפשר לעדכן כל מה שצריך ב־$\log n$). ההגיון: כאשר מזיזים תת עץ הוא לא השתנה, ורק יש לטפס ממנו והלאה. עתה נדבר על מחיקה מ־AVLs.

נעשה את אותם המשחקים של successor, ונעשה רוטציה בסוף. נקודה לשים לב אליה: אחרי הרוטציה יכול להיות שהגובה של הקודקוד כן השתנה מהמצב המקורי, כלומר, אי אפשר לעצור אחרי רוטציה ועדיין יהיה צורך לבדוק את ה־BF של הצומת מעליך. כאן לא נראה שהתיקון הוא $O(1)$ ב־amortized אבל לא נוכיח (ואולי זה לא נכון). זה לא כזה משנה מבחינת סיבוכיות כי בכל מקרה גובה העץ $\log n$.

הערה: יש משהו בשם WAVL שעושה דברים lazy כך שגם שהעלות של האיזון של סדרה של גם מחיקות וגם הוספות (לא כל אחד בנפרד) הוא amort ב־$O(1)$.

"לא הוכחתי, נופפתי בידיים" $\sim$ עמית 2025. עכשיו זה הקטע שרתם טוען לקיום משהו שגורר קיום אלגוריתם מסוג בעיית העצירה. שבוע הבא נלמד עצי B, שברמה התיאורטית הם לא טובים יותר, אך הם יותר אופטימיים לזכרון מחשב מודרני. "אין מבנה נתונים שפותר את כל הבעיות האלו כזה כללי. חוץ מ־dictionary של פייתון. הוא מושלם". עכשיו עמית מצייר על הלוח דוגמה של מחיקה שאין לי כוח להעתיק כי זה tikz אז תפתחו את המצגת. סיוכיות amortized של האיזון מחדש:

- עולים מהעלה לשורש העץ עד ש־$|BF = 2$
- מחפשים קודקודיים שהגובה שלהם השתנה, וכך $|BF| \leq 1$ (אך $BF = \pm 1$ ויוקד ל־0, איננו בין הגבהיים והקובה שלל הקודקוד נשאר זהה)
- נגדיר פונ' פוטנציאל: מספר הצמתים בעץ כך ש־$BF = 0$. בזמן הריצה, פונקציית הפוטנציאל ב־1 כל עוד לא עוצרים (מממן את הטיפוס). מצד שני, מגדילים את פונקציית הפוטנציאל לכל היותר בקבוע (בקודקוד שהשפענו שעשינו עליו רוטציות).

מסקנה. אם נכניס איברים ממויינים לעץ AVL ותמיד נתחיל מהקודקוד האחרון שהכנסנו, הסיבוכיות הכוללת תהיה $O(n)$. הסיבה – תמיד תיקח בן ימני, והתיקונים בזמן קבוע.

הערה: מיון מסוג זה מנצל היטב את העובדה שהרשימה ממויינת, ולכן הוא יכול להתאים טוב למיון רשימות שהן יחסית ממויינות, (כלומר את כמות ה־$i < j$ כך ש־$a_i > a_j$) וכך או אחרת הוא ב־worst case ב־$n \log n$.

לשם כך, נצטרך להחזיק תמיד את המינימום, ובאופן דומה נעשה למקסימום. נקרא לזה finger tree – עץ שמתחזק מצביעים לכל מני מקומות (לעיתים לאמצע, לעיתים מקסימום/מינימום, וכו'). אם כל מני שימושים פר מימוש. finger tree ששומר מינימום/מקסימום מאפשר למצוא את האיבר ה־i ב־$O(\log i)$, כי אפשר ללכת לראש העץ שמכיל אותו ואת המינימום ולחפש בו. (הערה: אם שומרים מצביע גם למינ' וגם למקס' אז $(O(\min(\log i,\ \log(n-i)))$).

גם rank tree וגם זה מאפשר לנו להשתמש במבנה הזה קצת כמו רשימה. נתעלם מה־keys, נשמור רק rank, וכך אפשר לממש tree list: עץ שמייצג רשימה.

. . . . . .