מבני נתונים 3

שחר פרץ

31 במרץ 2025

LINKED LIST & AMORTIZED COMPLEXITY .... (1)

דיברנו על רשימות מקושרות בתרגול הקודם. איך נוכל, בלי להוסיף מצביעים, להוריד איברים ב־$O(1)$ בסיבוכיות amortized?

התשובה: "נסמן" את האיבר שרוצים למחוק (נשלם "token" אחד) ובמעבר על הרשימה, כאשר נבחין לראשונה שוב באיבר נעביר פוינטר שיעדוף את האיבר הזה (כלומר אם n נמחק, המצביע של $n-1$ שבעבר הצביע ל־n יעבור להצביע ל־$n+1$). במקרה הזה נטרח לטפל המחיקת האיבר רק כאשר נעשה get, נוסף על זה שזה יקח לנו $O(1)$. הבעיה שזה פותר: הצורך לעבור על הרשימה המקושרת ולהגיע כל הדרך לאיבר שרוצים למחוק.

בכל מחיקה, השקענו שני "tokens" – אחד בעבור הסימון של האיבר כמחוק, והשני בעבור מתי שלא בעתיד נעביר את הפוינטר.

נקרא לזה מחיקה lazy חד כיוונית ברשימה מקושרת חד־כיוונית – תקרא lazy שכן לא נטרח לבצע את הזזת הפוינטר עד שבאמת נצטרך להזיז אותו. נסכם: נוכל רק לסמן איברים במחוקים בלי ממש למחוק, זה יעלה $O(1)$, ובזמן get נמצק איברים שסומנו מחורים מהרשימה. נשלם פעולה נוספת על כל מחיקה בדיוק פעם אחת, זמן get עתידי. בשיטת הבנק כל פעולה תעלה 2 מטבעות. בשיטת פונ' פוטנציאל $\phi(L) =$ מס' האיברים שמסומנים כמחוקים. בזמן get יקר, פונ' הפוטנציאל בדיוק בעלות הנוספת ה־get.

BINARY SEARCH TREE .... (2)

2.1 מילונים באופן כללי

2.1.1 ADT

נבנה ADT. בדוגמה למטה x כולל מאפיינים של x.key ו־x.value.

- Insert(D, x)
- Delete(D, x)
- Search(D, x)

And assuming order of keys:

- Min(D)
- Max(D)
- Seccessor(D, x)
- Predesecesssor(D, x)

2.1.2 מבני נתונים למילונים

להלן הסיבוכיות worst case בעבורי סוגי המילונים, כאשר בהקשר המתאים d עומק העץ:

op	dual-linked list	ordered dual-linked list	ordered circular array	binary tree
Insert	$O(n)$	$O(n)$	$O(n)$	$O(d)$
Delete	$O(n)$	$O(n)$	$O(n)$	$O(d)$
Search	$O(n)$	$O(n) = O(index)$	$O(\log n)$	$O(d)$
Min	$O(n)$	$O(1)$	$O(1)$	$O(d)$
Max	$O(n)$	$O(1)$	$O(1)$	$O(d)$
Successor	$O(n)$	$O(1)$	$O(1)$	$O(d)$
Predecessor	$O(n)$	$O(1)$	$O(1)$	$O(d)$

כאשר על מערך מעגלי ממוין דיברנו בתרגול והשאר ממבוא מורחב.

2.2 עצי חיפוש בינאריים

2.2.1 טרמינולוגיה

- עומק של קודקוד: אורך מסלול ממנו לשורש (במסלול שרק עולה).
- עומק של העץ: עומק מקסימלי של קודקוד.

- גובה של קודקוד: אורך מסלול הכי ארוך ממני ומטה (סמלול שרק יורד) בניסוח חילופי עומק תת־העץ שמתחיל ממנו.

ברמה הטכנית של המימוש, בדרך־כלל נוח שבעבור עלים המצביעים שלהם ימינה ושמאלה יצביעו לאותו משום חסר משמעות בזכרון.

2.2.2 פעולות

- חיפוש, החלפה והוספה: בעזרת חיפוש בינארי על העץ.
- מה חגבי מחיקה? נחלק למקרים.
  - עלה, אפשר למחוק כמו שהוא.
  - אם יש לו בן אחד, אז נעביר את תת העץ של הבן להיות במקומו.
  - אחרת, שני פתרונות עלו:
    * פתרון שלא שובר את העץ: להחליף את האיבר שמוחקים ב־predecessor שלו. לא תהיה בעיה למחוק את ה־predecessor מהמיקום שלו כי יש לו רק בן אחד. העומק במקרה הגרוע נשאר אותו הדבר.
    * פתרון כאוטי שהורג את האיזון: לקחת את העץ השמאלי ולשים אותו ב־predecessor של השורש של תת העץ הזה, שנמצא איפשהו בתת העץ הימני. עומק במקרה הגרוע הכפיל עצמו.

המטרה: לשמור על עומק $O(\log n)$. למה? כי העומק המינימלי של עץ בינארי הוא $\log n$