Data Structures ~ Home Work #4 ~ Semester B 2025

Shahar Perets

9 ביוני 2025

מידע כללי

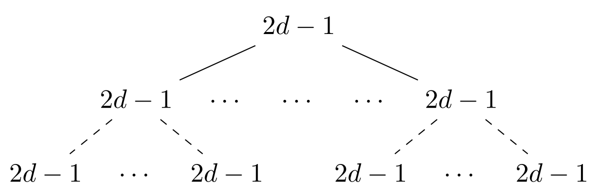
שחר פרץ ~ ת.ז. 334558962 ~ במודל shaharperets

(1)

יהי מבנה B-tree עם פרמטר d המתאר את גודל הצמתים בעץ.

א. בהינתן $7 \cdot 10^7$ איברים, נרצה להכניס אותם למבנה B-tree כך שגובה העץ לא יהיה יותר מ־7 (האיברים ידועים מראש). נחשב את ה־d המינימלי.

משום שהאיברים ידועים מראש, נוכל לבנות את סדר ההכנסה כך שכל צומת ב־B-tree יהיו $2d-1$ צמתים (כאשר החסם התחתון של העץ הוא $d-1$) עד לכדי הצומת האחרון. נתבונן ב־B-tree כלשהו מגודל מגודל $(d, 2d)$ כדי לנסות להבין כמה צמתים יהיו בו:



כאשר לכל צומת שאינו עלה $2d$ בנים. חסם עליון לכמות הצמתים הנכנסים בעץ $(d, 2d)$. נמצא את כמות הצמתים בו:

$$\text{count} = 1 + 2d + (2d)^2 + \cdots + (2d)^h = \sum_{i=1}^{h} (2d)^i = \frac{(2d)^{h+1} - 1}{2d - 1}$$

נצטרך לכפול כמות זו ב־$(2d-1)$, כמות האיברים בכל צומת, כדי לקבל את כמות האיברים שאפשר להכניס ל־B-tree מקסימלי:

$$\max \#\text{nodes} = \text{count} \cdot (2d - 1) = \frac{(2d)^{h+1} - 1}{\cancel{2d - 1}} \cdot \cancel{2d - 1} = (2d)^{h+1} - 1$$

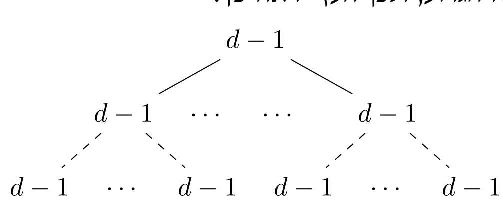
עתה, נציב את נתוני השאלה. אנחנו רוצים לאכסן $7 \cdot 10^7$ איברים, בעץ מגובה $h = 7$, נמצא את d:

$$\begin{aligned}
& 7 \cdot 10^7 \le \max \#\text{nodes}(2d)^{7+1} - 1 \\
\iff \quad & 7 \cdot 10^7 + 1 \le (2d)^8 \\
\iff \quad & \log_8(7 \cdot 10^7 - 1) \le 2d \\
\iff \quad & \mathbb{N} \ni d \ge \frac{\log_8(7 \cdot 10^7 - 1)}{2}
\end{aligned}$$

ולכן ה־d המינימלי הוא:

$$d = \left\lceil \frac{\log_8(7 \cdot 10^7 - 1)}{2} \right\rceil = \lceil 4.641531315 \rceil = \mathbf{5}$$

ב. נענה על אותה השאלה, אך עתה לא ידועים לנו האיברים מראש, כלומר, לא ידוע סדר הכנסת האיברים. אז במקרה הזה, נוכל להבטיח רק קיום של $d-1$ איברים בצומת במקרה הגרוע, ולכן העץ יראה כך:



ולכן כמות הצמתים:

$$\text{count} = 1 + d + d^2 + \cdots + d^h = \sum_{i=1}^{h} d^i = \frac{d^{h+1} - 1}{d - 1}$$

גם כאן, יהיו $d-1$ איברים בכל צומת ולכן:

$$\max \#\text{nodes} = \text{count} \cdot (d-1) = \frac{d^{h+1} - 1}{\cancel{d-1}} \cdot \cancel{d-1} = d^{h+1} - 1$$

גם כאן $h = 7$ ולכן חסם תחתון:

$$\begin{aligned} & 7 \cdot 10^7 \leq d^{7+1} - 1 \\ \iff & \quad 7 \cdot 10^7 + 1 \leq d^8 \\ \iff & \log_8(7 \cdot 10^7 + 1) \leq d \in \mathbb{N} \end{aligned}$$

סה"כ:

$$d = \lceil \log_8(7 \cdot 10^7 + 1) \rceil = \lceil 8.686950536 \rceil = 9 \quad \top$$

המשך בעמוד הבא

נוכיח חסם אמורטייז $O(1)$ על מספר הפעולות split/fuse במהלך הכנסות ומחיקות מלמטה ומלמעלה בעץ $(d, 2d)$.

הוכחה. נסמן ב־$\min T$ את הצמתים המינימליים בעץ T, וכן ב־$\max T$ את הצמתים המקסימלים בעץ T נתבונן בפונקציית הפוטנציאל הבאה:

$$\Phi(i) = \#\min T_i + \#\max T_i$$

כאשר T_i המצב של ה־B-tree אחרי הפעולה ה־i, בעבור רצף כלשהו של פעולות מחיקה והוספה. עבור הפעולה ה־i, נפרק למקרים:

- אם הפעולה הזו היא פעולת הוספה: נניח שבמסלול בין הצומת שהסרנו לבין השורש יש a צמתים רצופים מקסימליים (הנחה זו אפשרית שכן יתכן מקרה מנוון). אזי יהיו a פעולות splits, ובהוספה אין פעולות fuse לכן 0 פעולות fuse, וסה"כ $\text{cost}\, op_i = a$. כלומר $\min T_i = \min T_{i-1}$ וכן $\#\max T_i + a = \#\max T_{i-1}$. במקרה ה"רע" הפוטנציאל יגדל באחד שכן יהיה צומת מקסימלי חדש/יהיה צומת מינימלי חדש (אחד משני מקרים אלו בלבד). כך נוכל לקבוע חסם עליון.

$$\begin{aligned}
\text{amort cost} &= \Phi(i) - \Phi(i-1) + \text{cost}\, op_i \\
&= \#\max T_i + \#\min T_i - \#\max T_{i-1} - \#\min T_{i-1} + a \\
&\geq \cancel{\#\max T_i - \#\max T_{i-1}} - a + \cancel{\#\min T_i - \#\min T_{i-1}} + a + 1 \\
&= a - a + 1 = O(1)
\end{aligned}$$

- אם זוהי פעולת מחיקה: תהי "משפחה" קבוצת הבנים הישירים של קודקוד יחיד (כלומר, קבוצת כל בניו של קודקוד נתון). נניח שלאורך b משפחות עבור הקודקודים במסלול בין הצומת ממנו מחקנו לבין השורש, כל הצמתים מלאים (בפרט יהיו db צמתים מלאים לפחות בעץ). אזי נצטרך לעשות b פעולות borrow שלא יספרו, וכן b פעולות fuse.

  לבסוף, במקרה הריק בו $b = 0$ או לאחר סיום רצף הפעולות, נצטרך לבצע פעולת borrow אחת שלא תספר וסיימנו. כל פעולת fuse משמעותה צומת מינימלי אחד פחות, כלומר $\#\min T_i + b = \#\min T_{i-1}$. במקרה הטוב (שכן זה יגרום להקטנת הפרש הפוטנציאל), ייתכן שיתבצע borrow אחד מצומת מקסימלי (לא ייתכן יותר מאחר כי במקרה כזה המחיקה תיעצר). נוסף על כך במקרה ה"רע" יכול להיווצר צומת מחינימלי חדש. סה"כ:

$$\begin{aligned}
\text{amort cost} &= \Phi(i) - \Phi(i-1) + \text{cost}\, op_i \\
&= \#\max T_i + \#\min T_i - \#\max T_{i-1} - \#\min T_{i-1} + b \\
&\leq (1 + \cancel{\#\max T_i - \max T_i}) + (\cancel{\#\min T_i - \min T_i} - b) + b \\
&= 1 + \cancel{b - b} = 1 = O(1)
\end{aligned}$$

  כדרוש.

סה"כ, בין אם מחסירים ובין אם מוסיפים איבר לעץ, הפרש הפוטנציאלים יאזן את עלות הפעולה, ונקבל $O(1)$ למספר הפעולות ה־fuse/split בעץ.

■

המשך בעמוד הבא

תיאור המבנה: נתחזק 100 ערימות פיבונאצי, כאשר המינימום המספר ה־i נמצא כמינימום של הערמה ה־i.

תיאור התחזוקה: בכל אחת מהפעולות, נעבור על 100 הערימות שלנו לפי הסדר – מזו ששומרת את המינימום המוחלט עד לזו ששומרת את המינימום המאה.

Insert: נוסיף את האיבר ערימה ערימה, עד שנגיע לערימה בה המינימום גדול מהאיבר שמוסיפים. מכאן ואילך, נבצע תהליך שנקרא לו תהליך ההעברה – נשמור את המינימום של הערימה הקודמת (טרם הוספה), ונוסיף אותו לערימה הבאה בתור, וכן הלאה.

סה"כ, משום שבכל מקרה לא נוסיף יותר מאיבר אחד פר־ערימה, והוספת איבר לערימת פיבונאצי מתבצע ב־$O(1)$, סה"כ סיבוכיות $O(100) = O(1)$.

DelMin: בדומה ל־Insert, גם כאן נחסר את האיבר מכל ערימה בה הוא נמצא. בעבור הערימה הראשונה בה הוא לא נמצא, נבצע תהליך העברה הפוך – נשמור בכל איטרציה את המינימום מהערימה הקודמת (לאחר ההחסרה), ונמחק אותו מהערימה ההבאה בתור.

סה"כ ביצענו לכל היותר 100 פעולות של Delete-Min בערימת פיבונאצי, כלומר סה"כ $100 \cdot O(\log n) = O(\log n)$ כדרוש.

DecKey: נבצע Decrease-Key רגיל של ערימת פיבונאצי לכל אחת מהערימות בה נמצא האיבר, עד שנגיע לערימה בה לא נמצא האיבר. אם האיבר לא קיים בערימה וערכו לאחר ההחסרה (נוכל לחשב את זה עוד בערימה הראשונה) גדול מהמינימום בערימה הנוכחית, נוסיף אותו.

אם בדרך האיבר שעשינו לו Decrease-Key הגיע לראש הערימה הנוכחית עליה עוברים, אז מכאן ואילך נבצע את תהליך העברה המתואר ב־Insert, עד שנגיע לערימה האחרונה, נוסף על הוספת האיבר לערימה במידת הצורך כמתואר לעיל (לא ייתכן שמצב כזה בו האיבר מגיע לראש הערימה יחזור פעמים בקריאה אחת ל־Decrease-Key, ומכאן נכונות).

הערה: נכונות ה־Decrease-Key כאן נכונה בין היתר כי $\Delta > 0$, כנתון בשאלה.

משום שביצענו פעם אחת בלבד פעולות Increase ו־Decrease-Key רגילות על ערימת פיבונאצי, שתיהן אורכות $O(1)$, סה"כ חסם עליון של $100 \cdot O(1) + 100 \cdot O(1) = O(1)$.

הערה: השמות תהליך העברה ותהליך הערה הפוך אלו שמות שאני המצאתי כדי לא לתאר את אותו התהליך מספר פעמים במהלך שאלה זו.

סה"כ את Print100 נוכל לממש בפשטות ע"י מעבר על 100 ערימות הפיבונאצי ולקיחת המינימום מכל אחת מהן, דהיינו, 100 המינימומים הראשונים במבנה הנתונים. סיבוכיות הזמן בעבור Print100 תהיה $100 \cdot O(1) = O(1)$ כי אורך $O(1)$ למצוא מינימום בערימת פיבונאצי

המשך בעמוד הבא

ראשית כל, ניקח את המערך בעל n האיברים, ונבנה ממנו עץ בינארי מאוזן ב־$O(n)$ (בלי שום סידור פנימי). סידור זה שקול למעבר מייצוג של ערימה, ולכן נוכל לבנות תוך כדי בניית העץ מערך בזכרון (בה"כ שמו BottomUp) שיעבור על צמתי העץ מלמטה למעלה (הוא למעשה הייצוג כ־heap של העץ, בעוד העץ עצמו יהיה שמור כעץ הפניות רגיל). לאחר מכן, נפעיל עליו את הלאוריתם הבא כדי להפוך את העץ לנחמד:

```
input  : T tree
output: Nice binary tree

for node ∈ T.BottomUp do
    FixNode(node)
end
return T
```

```
function FixNode(node) is
    if node.right = null then
        if node.left ≠ null and node.key < node.left.key then
            replace node.left.key and node.key
        end
        return
    else if node.left < node.right < node.key then
        replace node.right.key and node.key
        FixNode(node.right)
    else if node.key < node.left < node.right then
        replace node.left.key and node.key
        FixNode(node.left)
    else if node.right < node.key < node.left then
        replace the pointers node.right and node.left
        // since the pointers themselves got replaced, there is no need to preform a recursive
           call
    end
end
```

ניתוח סיבוכיות: בדומה לאלגוריתם שראינו בכיתה, סיבוכיות הקריאה עצמה ל־FixNode היא $O(1)$, ובהתחשב בקריאות רקורסיביות, $O(h)$ עבור גובה h (חסום ב־$h \leq \log n$) שכן הקריאות הרקורסיביות כאשר נגיע לעלה. סה"כ, משום שב־BottomUp יש $\frac{n}{2^i}$ צמתים מגובה i (שכן קיבלנו עץ נחמד והוצאנו עץ נחמד, שכן מתכתחילה בנינו heap של עץ נחמד וממנו יצרנו BST ב־$O(n)$) הסיבוכיות תהיה:

$$\text{cost} = \overbrace{O(n)}^{\text{Creating the initial tree}} + \sum_{i=1}^{H} \underbrace{O(h)}_{\text{FixNode}} \frac{n}{2^h} \leq O(n) + n \cdot \underbrace{O\left(\sum_{h=1}^{H} h \frac{1}{2^h}\right)}_{H\to\infty:\, O(1)} = O(n) + O(n) + O(n) \quad \top$$

המשך בעמוד הבא

נתבונן בכל הערימות שנלמדו, ונממש את הפעולה WhereToInsert(x), המקבלת מפתח x ומחזירה באיזה מקום בערימה האיבר היה מוכנס.

א. **בערימה בינארית:** נבצע חיפוש בינארי על $\log n$ האיברים בדרך לצומת האחרונה, שיתבצע תוך מעבר בתוך ה־heap בלי להעתיק את $\log n$ האיברים למערך נפרד באמצעות המעברים שראינו בכיתה – $2i+1, 2i$ בשביל ילד ו־$\lfloor \frac{i}{2} \rfloor$ בשביל לגשת לאבא. כלומר:

```
min ← 1
max ← ⌈log Q.size⌉
while Q[current] ≠ last do
    last ← Q[current]
    current ← ⌊i/⌋
    if Q[current] > x then
        max = ⌊current/2⌋
    else
        min = 2 · current + 1
    end
end
return current
```

חיפוש בינארי על p איברים אורך $O(\log p)$ זמן, וכאן $p = \log n$ הוא גובה העץ המקסימלי, וסה"כ סיבוכיות $O(\log \log n)$.

הערה לבודק: ההסבר המילולי מספיק, או שזה טוב שהוספתי פסאדו קוד?

ב. **בערימה בינומית:** נסתמך על חיבור של מספרים בינאריים. בסיבוכיות $O(\log n)$ נוכל למצוא את גודל כל הערימות שנמצאות במבנה, ולהמרי אותן לייצוג בינארי – כלומר, בעבור העץ הבינומי ה־i מגודל 2^i היא B_i, הספרה ה־i בייצוג הבינארי תהיה 1 אם היא נוכחת ו־0 אם היא אינה. עתה, נבצע פעולה increment למספר הבינארי שקיבלנו, ונחזיר את הספרה בה הוא יעצור.

משום שהכנסה לערימה בינומית שקולה לחיבור מספרים בינארים, הספרה בה ה־increment יעצור (בניסוח שקול, הביט האחרון שנהפוך) יהיה הערימה שאליה המספר שלנו יתווסף. סה"כ סיבוכיות $O(\log n) + O(\log n)$ בעבור ה־increment ובעבור היצירה של המספר הבינארי מתוך הערימה, וסה"כ $O(\log n)$.

הצעה חילופית: בכלל שהן פעולות הוספת איבר והן פעולת מחיקתו מתצבעת ב־$O(\log n)$, אפשר להוסיף את האיבר, לקבל מההוספה את המקום שאליו הוא נכנס, ולמחוק אותו. בסיום האלגוריתם לא שינינו את המבנה ביחס לתחילתו, אך אני לא בטוח אם זה עונה על דרישות השאלה שכן במהלך האלגוריתם כן שנינו את המבנה.

ג. בערימת פיבונאצי, נשתמש באלגוריתם המתקדם הבא כדי למצוא את דרגת העץ אליו היה מוכנס x בסיום הפעולה: להחזיר 0.

משום שחישוב המספר 1 לוקח זמן קבוע, סיבוכיות הזמן של האלגוריתם היא $O(1)$. נכונות נובעת מכך שערימות עצלות ובפרט ערימת פיבונאצי מוסיפות כל איבר חדש לעץ B_0 עם צומת יחידה, בלי לבצע פעולות נוספות.

המשך בעמוד הבא

א. נכניס לערימה הבינומית העצלה המתוארת בסעיף, את האיברים $1, 2, 3 \dots n$. נבצע לאחד מכן שתי פעולות delete-min, בזו אחר זו. נמצא את סיבוכיות זמן הריצה של אותן הפעולות.

במימוש הנוכחי

עבור הפעולה הראשונה, יהיו n עצים מגודל 1 במבנה. עבור הצומת i, נבחין שניקח אותו ואת הצומת ה־$i+1$, שניהם מגודל 1 ולכן נחבר אותם לעץ מגודל 2 ונעביר אותם לעץ הסופי. נעשה זאת עבור כל זוג עוקב של צמתים. משום שאלו צמתים עוקבים ועץ מגודל קבוע עלות בניית כל אחד מ־$\lceil \frac{n}{2} \rceil$ העצים הללו (יכול להיות שישאר שארית 1 בסוף, מכאן העיגול מעלה) תהיה קבועה. סה"כ עלות יצירת העצים הפעולה הראשונה היא $O\left(\lceil \frac{n}{2} \rceil\right) = O(n)$. המעבר על הצמתים יארך $O(n)$ וסה"כ $O(n) + O(n) = \boldsymbol{O(n)}$

עלות הפעולה השנייה היא באופן דומה $O\left(\lceil \frac{n}{4} \rceil\right)$ ליצירת עצים, וכן $O(\lceil \frac{n}{2} \rceil) = O(n)$ למעבר על הצמתים - כלומר $\boldsymbol{O(n)}$ גם כאן.

במימוש המקורי

במימוש המקורי, הריצה הראשונה של פעולת successive-linking. חסם תחתון לפעולה הראשונה: נצטרך ליצור את המערך שעליו מבצעים successive-linking, דבר שיארך $O(n)$ (יש n עצים במבנה בהתחלה). חסם עליון: בהרצאה ראינו חסם עליון ל־successive-linking של $O(n)$. סה"כ $\Theta(n)$.

חסם לפעולה השנייה: העץ מאוזן לאחר ה־successive-linking הראשון, והמחיקה של איבר אחד מהעץ תוביל במקרה הרע לפיצול אחד מהצמתים לכדי 2 עצים חדשים. ביצוע successive-linking לעץ יחיד בנפרד, יארך $O(\log n)$ כי יש $O(\log n)$ עצים במבנה. יהיו $\log n$ עצים חדשים, ויוכלו להיווצר לכל היותר $\log n$ עצים חדשים שנצטרך לטפל בהם בלפחות $O(1)$, ובתום פעולת successive-linking, אז סה"כ $O(\log n + \log n) = \boldsymbol{O(\log n)}$.

סיכום

בדרך החדשה	בדרך השינה
פעולה ראשונה $O(n)$	פעולה ראשונה $O(n)$
פעולה שנייה $O(n)$	פעולה שנייה $O(\log n)$

ב. נוכל לחסום מלמעלה את זמן הריצה של פעולת Delete-Min באמצעות נוסחה זהה לזו שראינו בכיתה:

$$\text{cost}(op_i) \leq T_i + \log n + L \leq 2(T_i + \log n)$$

כאשר, בדומה לסימונים בכיתה, $\log n$ כמות העצים החדשים שנוצרים וצריך לטפל בהם, T_i כמות העצים לפני המחיקה, ו־L כמות פעולות ה־Link שביצענו בין שני עצים. גם כאן נוכל לחסום את הביטוי $T_i + \log n + L - 1$ ע"י $2(T_i + \log n)$ כי $L \leq T_i + \log n$ – נעשה linking אך ורק בין עצים שהיו קודם, או בין אחד מלכל היותר $\log n$ הצמתים החדשים שנוספו ב־Delete-Min. מה שישתנה ביחס למימוש שראינו בהרצאה, הוא השינוי ב־T_i ביחס לפעולות שישפיע על הפרש הפוטנציאלים, ולכן נצטרך לבחור פונקציית פוטנציאל מעט שונה.

$$\Phi(op_i) := 4 \cdot \#(\text{Binomial trees in the heap } \textbf{after the operation}) = 4T_{i+1}$$

נראה שהיא עובדת. לשם כך, נוכיח את השמורה הבאה: עבור $\forall i \in \mathbb{N}\colon T_{i+1} \leq \frac{T_i}{2} + \log n$. הוכחה: על כל זוג עצים עודף, הם יתחברו בזוגות ומספרם יקטן פי 2. בהינתן T_i עצים, אם $T_i \leq \log n$ לא נוכל להבטיח שמוזגו עצים ולכן $T_{i+1} \leq T_i \leq \log n \leq \frac{T_i}{2} + \log n$. אחרת $T_i > \log n$ ומוזגו $T_i - \log n$ צמתים ועתה ישנם לכל היותר $T_{i+1} \leq T_1 - \frac{T_i - \log n}{2} = \frac{T_i + \log n}{2} \leq \frac{T_i}{2} + \log n$ כדרוש.

תהי פעולה op_i כלשהי. נראה שהפרש הפוטנציאלים מאזן את הפרש העלויות:

$$\begin{aligned}
\text{amort}(\text{cost}) &= \text{cost}(op_i) + \Delta(\Phi_{i \to i-1}) \\
&\leq 2(T_i + \log n) + 4T_{i+1} - 4T_i \\
&= O(\log n) + 2(2T_{i+1} - T_i) \\
&\overset{(*)}{\leq} O(\log n) + 2\left(\cancel{2} \cdot \frac{T_i}{\cancel{2}} + 2\log n - T_i\right) \\
&= O(\log n) + 2\log n \\
&= O(\log n) + O(\log n) = \boldsymbol{O(\log n)} \quad \top
\end{aligned}$$

כאשר $(*)$ נכון מלמה שהוכחנו. סה"כ הוכחנו חסם אמורטייז $O(\log n)$ למספר התיקונים, כדרוש.

שחר פרץ, 2025

קומפל ב־LaTeX ונוצר באמצעות תוכנה חופשית בלבד