מבני נתונים 2

16 מרץ, 2000

ערימה בינומיאלית וערימת פיבונצ'י

ערימות (heaps)

סימונים: x=מצביע לאיבר, Δ=מספר ממשי, k מפתח ו- h=שם של ערימה.

ערימה היא מבנה נתונים התומך בפעולות:

- make_heap(k) יצירת ערימה חדשה עם איבר בודד שמפתחו k
- insert(k,h)
- find_min(h)
- del_min(h)
- meld(h1, h2) יצירת ערימה חדשה שאיבריה הם איחוד איבריו - h1, h2
- delete(h, x)
- decrease(Δ, x)

בעץ T, שבכל אחד מצמתיו יש מפתח, מתקיים סדר ערימה אסם לכל צומת v מתקיים $key(parent(v)) \le key(v)$. אם בעץ מתקיים סדר ערימה, המינימום נמצא בשורש ולכן Time(find_min)=O(1).

ניתן לממש ערימה באמצעות עץ בינארי מאוזן (למשל, עץ AVL). בפרק זה נראה מימוש אחר, הנקרא ערימה בינומיאלית. לאחר מכן נרחיב מבנה זה לערימת פיבונצ'י. הטבלא הבאה מראה את זמן הביצוע הגרוע ביותר של כל אחת מהפעולות על כל אחד מסוגי הערימות. כפי שניתן לראות, ערימות בינאריות הן טובות מספיק כל עוד אין צורך במיזוג ערימות. אולם, אם פעולה זו דרושה, לערימות המורכבות יותר יש יתרון ניכר. לכן ערימות אלה נקראות גם mergable heaps.

	ערימה בינארית (במערך)	ערימה בינומיאלית	ערימה בינומיאלית עם lazy meld (זמן משוערך)	ערימת פיבונצ'י (זמן משוערך)
make_heap	O(1)	O(1)	O(1)	O(1)
Insert	O(log n)	O(log n)	O(1)	O(1)
find_min	O(1)	O(log n)	O(1)	O(1)
del_min	O(log n)	O(log n)	O(log n)	O(log n)
meld	O(n)	O(log n)	O(1)	O(1)
decrease	O(log n)	O(log n)	O(log n)	O(1)
delete	O(log n)	O(log n)	O(log n)	O(log n)

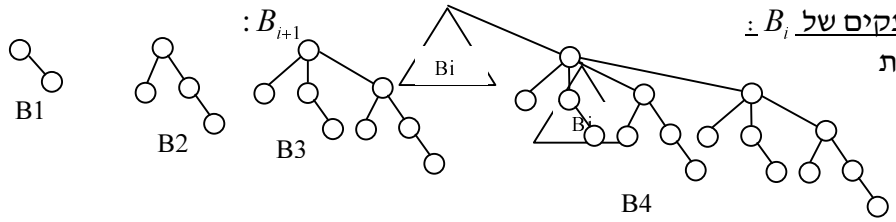
ערימה בינומיאלית

נגדיר עץ בינומיאלי באופן רקורסיבי:

עלה: • B_0

שני עותקים של B_i: B_{i+1}:

דוגמאות



<u>תכונות של עצים בינומיאליים</u>: (הוכחה באינדוקציה)

1. ל B_n יש 2^n צמתים.
2. לשורש B_n יש n בנים. הבן ה i הוא שורש של B_i.
3. ל B_n יש $\binom{n}{k}$ צמתים בעלי עומק k.
4. הגובה של B_n הוא n.

מסקנה: הדרגה המקסימלית של צומת בעץ בינומיאלי בן n צמתים היא $\log n$.

נממש ערימה ע"י יער של עצים בינומיאלים כך שמתקיים:

1. כל עץ ביער מקיים סדר ערימה.
2. יש ביער לכל היותר עץ אחד מכל דרגה.
3. נחזיק מצביע לאיבר המינימלי כדי לאפשר find_min בזמן O(1).

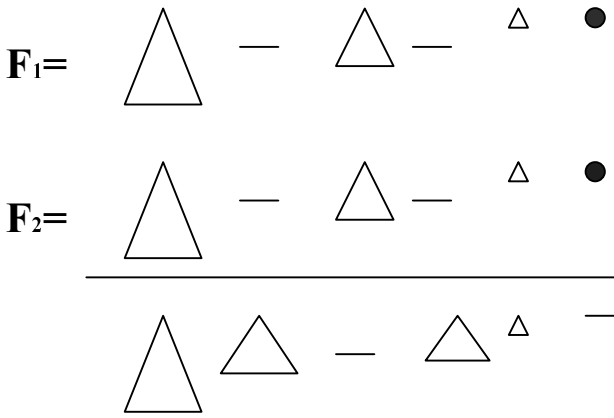
יהיה N מספר המפתחות בערימה. כדי לקבוע את דרגות העצים הקיימים ביער, נתבונן בייצוג הבינארי של N: $\langle b_0, b_1, ..., b_{\lfloor \log N - 1 \rfloor}, b_{\lfloor \log N \rfloor} \rangle$. מתקיים $N = \sum_{i=0}^{\lfloor \log N \rfloor} b_i 2^i$ ולכן העץ B_i יופיע ביער אסם $b_i = 1$.

פעולת האיחוד (meld):

נרצה לחבר שני עצים בינומיאליים מדרגה שווה, B_i לעץ בינומיאלי אחד מדרגה B_{i+1}. כדי לשמור על סדר הערימה, נשווה בין ערכי המפתחות של שורשי שני העצים. את השורש בעל המפתח הגדול יותר נחבר כבן הימני ביותר של השורש שמפתחו הוא הקטן.

איחוד שתי ערימות בינומיאליות ייעשה באופן הבא: אם בשתי הערימות ישנם עצים מדרגה 0, נאחד אותם לעץ מדרגה 1. אחרת, יש לכל היותר רק עץ אחד מדרגה 0 והוא יהיה חלק הערימת האיחוד.
לאחר שלב זה, נסתכל על הדרגה הבאה. אם יש סה"כ (משתי הערימות ומאיחוד בשלב הקודם) יותר מעץ אחד, נאחד שני עצים ונשאר עם עץ אחד מדרגה זו לכל היותר. (שימו לב לדמיון בין תהליך זה לחיבור אלגברי בין שני שלמים).

דוגמא: איחוד שתי ערימות בינומיאליות, $F_1 = \{B_0, B_1, B_3, B_5\}$ ו- $F_2 = \{B_0, B_1, B_3, B_5\}$.
(במקרה שתי הערימות הן בעלות מבנה זהה, אך אין לכך משמעות).



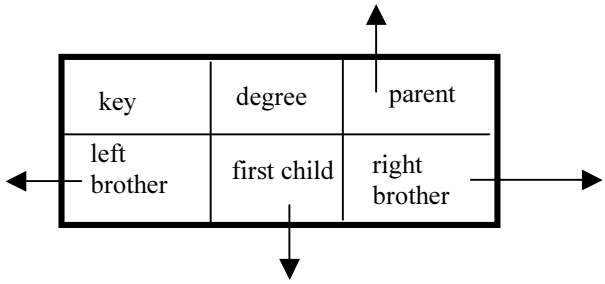
חיבור שני עותקים של B_i ייתן עותק של B_{i+1}, כאשר השורש של הוא השורש של העץ שערך המפתח שלו יותר קטן.

קל לראות שהזמן הדרוש לאיחוד שתי ערימות המכילות n_1 ו- n_2 איברים הוא $O(\log \max(n_1,n_2))$.

פעולת insert(k,h) תתבצע על ידי make_heap(k) ואיחוד הערימה שנוצרה עם הערימה h. באופן דומה, שיטת איחוד זאת גורמת לכך שכל הפעולות יתבצעו בזמן של $\log n$.

מימוש יער בינומיאלי

נבנה יער כרשימהמעגלית דו-כיוונית של עצים.
מבנה צומת בעץ:



הבנים של כל צמת יסודרו ברשימה מקושרת דו-כיוונית) ע"פ דרגתם (מספר הבנים = דרגת העץ הבינומיאלי התלוי בצומת).
כדי לאפשר להוסיף בן אחרון, נשמור את הבנים כרשימה מעגלית דו-כיוונית. את השורש של העץ בעל המפתח הקטן נוסיף בין הבן האחרון לבן הראשון של העץ עם המפתח הגדול. כיון שהרשימה מעגלית אזי `root->first_child->left_bother` מצביע לבן האחרון.
(במחוון ל- parent נשתמש רק בהמשך).

Lazy Meld

נראה עכשו כיצד ניתן להקטין את הזמן המשוערך של find_min ו- meld (ולכן של insert) ל- $O(1)$.

find_min: נחזיק מחוון לשורש בעל המפתח המינימלי, ונעדכן אותו בזמן insert, meld ו- del_min. תחזוקת המחוון תוסיף קונסטנטה לפעולות האחרות, אך תאשפר לבצע find_min בזמן (גרוע) $O(1)$.

Lazy Meld

נבטל את הדרישה שאיחוד של שתי ערימות יניב ערימה בה יש עץ בינומיאלי אחד לכל היותר מכל דרגה. יער של עצים יהיה עדיין רשימה מקושרת של כל העצים, ואיחוד של שתי ערימות יתבצע על ידי שרשור שתי הרשימות. גם כאן נחזיק מצביע לעץ שבשורש שלו מוחזק האיבר המינימלי.

זמן ה-del_min יהיה $O(f)$ כאשר f הוא מספר העצים. אך עכשו כבר לא מתקיים $f = O(\log N)$ (N מספר המפתחות בערימות). כיון שדרגת העצים $\geq \log N$, הזמן הממשי יהיה $t = f + \log N$. (ביצוע Bucket-Sort על f מספרים חסומים ע"י $\log N$ לוקח זמן $t = f + \log N$).

כדי לחשב זמן משוערך נגדיר פונקצית פוטנציאל:

$$\varphi(F) = f = \text{מספר העצים בכל הערימות}$$

יהי f מספר העצים לפני ביצוע ה-del_min על כן $\varphi(before) = f$.

אחרי ביצוע ה-del_min מספר העצים הוא לכל היותר $\log N$. לכן $\varphi(after) \leq \log N$

$$\Delta\varphi = \varphi(after) - \varphi(before) \leq \log N - f$$

$$a = t + \Delta\varphi \leq (\log N + f) + (\log N - f) = 2\log N$$

<u>הערת מימוש</u>

אם נשמור את השורשים כרשימה מקושרת מעגלית דו-כיוונית נוכל לשרשר שתי ערות בזמן $O(1)$. שימו לב שבמקרה זה, השרשים של היער יופיעו בסדר שרירותי.

(עדיין אין צורך בשדה parent.)

<u>ערימות פיבונצ'י</u>

כעת נראה מימוש לערימה כך שכל הפעולות למעט del_min יתבצעו בזמן משוערך O(1).

(כלומר נקטין את זמן decrease).

<u>מימוש הפעולות:</u>

מימוש הפעולות כמו ביער בינומיאלי עם lazy meld:

make_heap(k) - יצירת עץ יחיד עם צומת אחד שמפתחו k . האיבר המינימלי הוא שורש עץ זה.

insert(h,k) - הוספת עץ עם צומת יחיד לרשימה המקושרת. עדכון המצביע לבן המינימלי.

find_min(h) - החזרת המפתח המוצבע ע"י המצביע לשורש בעל המפתח המינימלי.

meld(h_1,h_2) - שרשור שתי הרשימות המקושרות של שתי הערימות. עדכון המצביע למינימום מבין המפתחות המינימלים של שני היערות.

del_min(h) - ראשית ניקח את הצומת שמכיל את המפתח המינימלי ונמחוק אותו. העץ שמפתח זה היווה את שורשו, מתפרק עכשיו למספר עצים. נצרף עצים אלה לרשימה המקושרת הקיימת. עכשיו יש צורך למצוא את השורש בעל המפתח המינימלי מבין כל העצים שביער. מכיוון שזה מצריך מעבר על כל העצים בערימה, ננצל מעבר זה לביצוע איחוד של עצים.

ראשית נמיין את כל העצים לפי דרגתם (מספר הבנים של השורש), בעזרת *bucket sort*. עכשיו נעבור על כל העצים בעלי אותה דרגה (באותו *bucket*), החל מהדרגה הנמוכה ביותר, נאחד כל שניים תוך שמירה על סדר ערימה ונקבל עץ חדש שדרגתו גדולה באחד. נמשיך עד שנסיים לעבור על כל התאים (יתכן ויווצרו תאים חדשים). לבסוף נקבל לכל היותר עץ אחד מכל דרגה. תוך כדי המעבר נמצא גם את השורש בעל המפתח המינימלי ונעדכן את המצביע אליו.

הפעולות הנוספות:

decrease (x, Δ) - נתק את תת העץ של x מאביו, והוסף אותו לרשימה המקושרת של הערימה.

בעץ החדש הורד את ערך המפתח של x ב Δ . בנוסף, כל צומת שהתנתקו ממנו שני בנים, מתנתק מאביו.

delete(x) - נממש פעולה זו ע"י **decrease(x,∞)** ואח"כ **del_min**.

נשים לב שכאשר אין מבצעים decrease ו-delete כל העצים נשארים בינומיאלים. אך אחרי ביצוע decrease ו delete, נוצר מצב שבו העצים בערימה אינם בהכרח בינומיאליים. בהמשך נראה שסכמת ההתנתקויות מבטיחה שהדרגה המינימלית של צומת היא גדולה מספיק כדי להבטיח שהסיבוכיות עדיין טובה.

לצורך האנליזה נצטרך לדעת מספר עובדות על סדרת פיבונצ'י

הגדרה:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{n+1} = F_n + F_{n-1} \end{cases}$$

דוגמאות: $F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8, F_7 = 13$

תכונות:

א. $F_{n+2} = 2 + \sum_{i=2}^{n} F_i$

ב. נוסחה סגורה לסדרת פיבונצ'י:

$$F_n = \frac{1}{\sqrt{5}}\left(\Phi^n - \overline{\Phi}^n\right)$$

כאשר:

$$\overline{\Phi} = \frac{1-\sqrt{5}}{2} = -0.688 \qquad \Phi = \frac{1+\sqrt{5}}{2} = 1.608$$

מכוון ש $|\Phi| > 1$, $|\overline{\Phi}| < 1$ מתקיים: $F_n \approx \frac{1}{\sqrt{5}}\Phi^n$

נרצה לחסום את מספר העצים בערימה שלאחר ביצוע מחיקה, וכן את מספר העצים שנוצרים כאשר מוחקים שורש של עץ ב $O(\log n)$. כדי שנוכל לעשות זאת, נראה שמספר הצמתים בתת-עץ כלשהו בערימת פיבונצ'י הוא אקספוננציאלי בדרגת העץ.

למה 1: בערימת פיבונצ'י לבן ה - ***i*** של צומת ***x*** יש דרגה לפחות ***i - 2***.

הוכחה:

יהי ***y*** הבן ה ***i*** של ***x***. נניח שבזמן הצטרפותו של ***y*** לעץ הוא הבן ה $d+1$ של ***x***. אז הדרגה של ***y*** ברגע ההצטרפות היא ***d***, כי מאחדים עצים עם דרגה שווה. לא יתכן שמאז התנתקו מ- ***y*** יותר מבן אחד, כי אחרת ***y*** היה מתנתק מאביו, לכן כרגע יש ל ***y*** לפחות $d-1$ בנים. מצד שני, כל בן חדש שנוסף ל ***x***, נוסף אחרי ***y***, ולכן:

$$i \le d+1$$
$$\deg(y) \ge d-1 \ge i-2$$

למה 2: בערימת פיבונצ'י לצומת מדרגה ***k*** יש לפחות F_{k+2} צאצאים.

הוכחה:

נסמן ב S_k את מס' הצמתים בתת-העץ המינימלי (מבחינת מס' הצמתים) שדרגת שורשו ***k***, שיכול להיווצר בערימת פיבונצ'י. קל לראות שסדרה זו מונוטונית עולה ב ***k*** (לכל עץ שדרגתו $k+1$, קיים עץ שדרגתו ***k***, ומכיל פחות צמתים - זה המתקבל ע"י ביצוע ***decrease()***, לאחד מבניו של שורש תת-העץ). מספיק להראות ש $S_k \ge F_{k+2}$.

באינדוקציה:

S_0: בעץ שדרגתו 0 יש צומת אחד בדיוק, ולכן: $S_0 = 1 = F_2$.

S_1: בעץ שדרגתו 1 יש לפחות שני צמתים (השורש של תת העץ, ובנו האחד), ולכן $S_1 \ge 2 = F_3$.

צעד: נתבונן בעץ המינימלי ***T*** האפשרי שדרגתו ***k*** בערימות פיבונצ'י. נסמן ב ***size(v)*** את מספר הצמתים בתת העץ ששורשו ***v***. מתקיים:

$$|T| = 1 + \sum_{i=1}^{k} size(son_i) \quad \geq \quad 2 + \sum_{i=2}^{k} size(son_i)$$

מלמה 1 מתקיים:

$$\deg(son_i) \geq i - 2$$

ולכן, בגלל מונוטוניות:

$$size(son_i) \geq S_{i-2}$$

מהנחת האינדוקציה מתקיים:

$$S_{i-2} \geq F_i$$

ולכן נקבל:

$$|T| \geq 2 + \sum_{i=2}^{k} F_i \quad = \quad F_{k+2}$$

<u>מסקנה:</u>
מס' הצמתים בתת-עץ ששורשו בדרגה k הוא $\Omega(\Phi^k)$ (נובע מכך שהמספר הפיבונצ'י ה k מקיים: $F_k = \Theta(\Phi^k)$).

לכן, הדרגה המקסימלית במבנה שמכיל ערימות פיבונצ'י עם סה"כ n צמתים היא: $O(\log_\Phi n) = O(\log n)$.

נוסיף סימון לצמתים: בתחילה צומת איננו מסומן. כאשר מתנתק ממנו בן, הוא מסומן. כאשר צומת מתנתק מאביו, הוא הופך ללא מסומן.

ניקח פונקציית פוטנציאל שעבור מבנה שמכיל מספר ערימות פיבונצ'י ערכה הוא מספר העצים + פעמיים מס' הצמתים המסומנים. באופן דומה לניתוח עבור ***lazy meld*** לארבע הפעולות הראשונות יש זמן משוערך $O(1)$.
עבור פעולת ***del_min*** הניתוח הוא בדיוק כמו קודם, כי מספר הצמתים המסומנים לא עולה, כלומר $O(\log n)$ זמן משוערך.

בפעולת ***decrease*** יתכן ונוצרה סדרה של ניתוקים. נניח שקרו l התנתקויות. אז פרט לצומת הראשון, ***l-1*** צמתים מסומנים התנתקו מאבותיהם, ולכן הם כבר אינם מסומנים. הצומת האחרון שהתנתק הפך את אביו למסומן, ולכן צומת אחד הפך מלא-מסומן למסומן. סה"כ השינוי במס' המסומנים הוא: $-(l-1)+1 = -(l-2)$. מס' העצים החדשים שנוצרו הוא l. והזמן האמיתי שערכה הפעולה הוא $O(l)$. סה"כ מקבלים:

$$a = f + \Delta\varphi = O(l) + l - 2(l-2) = O(l) - l + 4 = O(1)$$

פעולת ***delete(x)*** מורכבת מפעולת ***decrease*** ופעולת ***del_min*** ולכן הזמן המשוערך שלה הוא $O(\log n)$, כשל del_min.

<u>סיכום</u>
מימשנו את הפעולות ***make_heap*** , ***insert*** , ***find_min*** , ***meld*** , ***decrease*** ב זמן משוערך $O(1)$ ואת הפעולות ***del_min*** ו***delete*** ב $O(\log n)$, כאשר n הוא מספר המפתחות במבנה ברגע נתון.