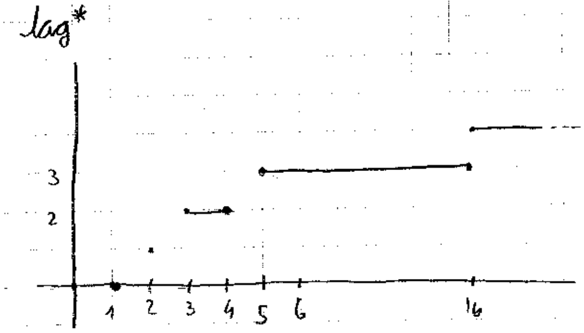
20.7.08 1

מבני נתונים - שיעור השלמה (ראשון)

ניתוח amortized של המימוש ה-II (בעזרת עצים) של Union-Find



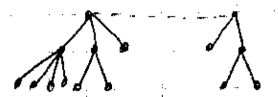
רצף של l פעולות (על n איברים):

$O(l \log\log n)$ 1972

$O(l \cdot \log^* n)$ ← נוכיח היום 1973

$O(l\,\alpha(n))$ ← יש בספר במהדורה השנייה (הוכחה מורכבת) 1975

תזכורת למימוש באמצעות עצים



make-set: יצירת עץ עם קודקוד יחיד עם rank 0. $\Theta(1)$.

Union (by rank): תולים שורש של אחד העצים על השורש של השני. (תולים את זה עם ה rank הקטן. במקרה של שוויון מגדילים ב-1 את ה rank של השורש החדש).

find(x) (עם כיווץ מסלולים): עולים במסלול מהקודקוד x עד לשורש העץ והופכים כל הקודקודים לאורך המסלול לבנים ישירים של השורש (ללא שינוי ה- rank שלהם).

<u>תזכורת:</u> יש בדיוק n קודקודים שיהיו אי פעם עם rank 0, יהיו לכל היותר $\frac{n}{2}$ קודקודים שיהיו אי פעם עם rank 1. יהיו לכל היותר $\frac{n}{4}$ קודקודים עם rank 2.

באופן כללי, יהיו לכל היותר $\frac{n}{2^k}$ קודקודים שיהיו אי פעם עם rank k.

הגדרת מדיניות חיוב

make-set, union מחויבים בעלות האמיתית $\Theta(1)$.

ב find יחויב בטיפול רק בקודקודים לאורך המסלול אותן מכוונים שמקיימים את התנאי $\log^* \text{rank}\, v < \log^* \text{rank parent}(v)$. את שאר העלות של ה find, כלומר את הטיפול בשאר הקודקודים לאורך המסלול נשלם על הקודקודים עצמם.

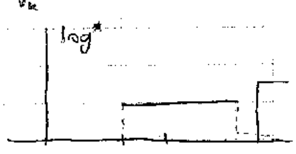
כלומר, כל קודקוד יחויב בטיפול בכל ה-find-ים בהם הוא מעורב ובהם מתקיים התנאי $\log^* \text{rank}(v) = \log^* \text{rank parent}(v)$

<u>כמה מחויב כל find?</u> לכל היותר $\log^* n$.

(למעשה אפילו לכל היותר $\log^*(\log n) = \log^* n - 1$).

<u>כמה מחויב קודקוד x בסה"כ?</u>

נשים לב כי rank x נשאר קבוע בכל ה-find-ים בהם x מעורב ובעזרת ה-find-ים האלו, ה-rank-ים של האבות של x הם סדרה עולה ממש.



מספר ה-find-ים שבהם הקודקוד x יחויב הוא לכל היותר אורך הבלוק של $\log^*$ שבו rank(x) נמצא.

20.7.08 (1)

מבני נתונים - שעור השלמה (ראשון)

מה החיוב הכולל של כל הקודקודים?

נחשב ראשית מה החיוב הכולל של הקודקודים v שבהם $\log^* \mathrm{rank}\, v = t$. (ואחר כך נסכם על כל ה-$t$-ים).

$$\sum_{\text{rank } r = 2^{\cdot^{\cdot^{2}}}\}t-1\,+1}^{2^{\cdot^{\cdot^{2}}}\}t} \underbrace{\frac{n}{2^r}}_{\text{כמות הקודקודים עם rank } r} \cdot \underbrace{2^{\cdot^{\cdot^{2}}}\}t - \dots}_{\text{חיוב מקסימלי של כל קודקוד ("אורך הקטע")}} \le 2^{\cdot^{\cdot^{2}}}\}t \cdot n \sum_{r=2^{\cdot^{\cdot^{2}}}\}(t-1)\,+1}^{2^{\cdot^{\cdot^{2}}}\}t} \frac{1}{2^r} = n \cdot 2^{\cdot^{\cdot^{2}}}\}t \cdot \Theta\left(\frac{1}{2^{2^{\cdot^{\cdot^{2}}}\}(t-1)}}\right)$$

$$= \Theta(n)$$

כעת נסכם על t ונקבל שהחיוב הכולל של כל הקודקודים

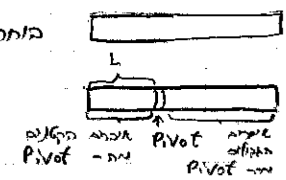
$$\sum_{t=0}^{\log^*\log n = \log^* n - 1} \Theta(n) = \Theta(n\cdot\log^* n)$$

לסיכום, רצף של l פעולות יתבצעו בזמן:

$$O(\underbrace{n\cdot\log^* n}_{\text{חיוב הקודקודים}} + \underbrace{(l-n)\log^* n}_{\text{חיוב ה find-ים}} + \underbrace{l}_{\text{חיוב שאר הפעולות}}) = O(l\cdot\log^* n)$$

ניתוח הסתברותי של QuickSort

בוחרים Pivot מבצעים Partition.



מפעילים את קריאה רקורסיבית של QuickSort בשני החלקים.

נניח שה pivot מוגרל כל פעם באופן אקראי מבין איברי המערך בהסתברות אחידה.
נתבונן בשני משתנים מקריים:

T זמן הריצה של האלגוריתם על קלט נתון בין n איברים.

L מספר האיברים שמשמאל ל-Pivot לאחר ה Partition הראשון.

טענה: $E(T)=O(n\log n)$

הוכחה: ידוע לנו שה Partition מתבצע ב $O(n)$. כלומר קיים קבוע c כך שה Partition של מערך בין n איברים מתבצע תמיד בזמן $\le c\cdot n$.

נוכיח באינדוקציה על n ש: $E(T)\le 4\cdot c\cdot n\log n$.

בסיס האינדוקציה: "$n=1$": אפשר להכיל ע"י הגדלת c (אם יש צורך).

מעבר האינדוקציה:

$$E(T) = \sum_{l=0}^{n-1} E(T \mid L=l)\cdot \underbrace{P(L=l)}_{\frac{1}{n}}$$

לפי נוסחת התוחלת השלמה.

$$= \frac{1}{n}\cdot\sum_{l=0}^{n-1} E\left(\text{זמן של Partition} + \text{זמן של QuickSort בצד שמאל} + \text{זמן של QuickSort בצד ימין} \mid L=l\right)$$

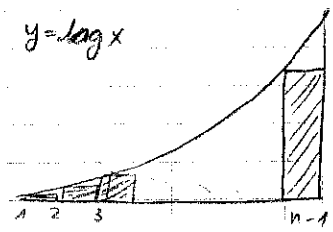
$$= \frac{1}{n}\sum_{l=1}^{n-1} E\Big(\underbrace{\text{זמן Partition}}_{\le cn} \Big| L=l\Big) + \underbrace{E\Big(\text{QuickSort על צד זה} \Big| L=l\Big)}_{\le 4cl\log l \text{ עפ"י הנחת האינדוקציה}} + E\Big(\text{QuickSort על הצד הימין} \Big| L=l\Big)$$

לינאריות התוחלת המותנה.

$$\le \frac{1}{n}\sum_{l=1}^{n-1} c\cdot n + 4c\,l\cdot\log l + 4c\cdot(n-1-l)\log(n-1-l) = cn + 2\cdot\frac{4c}{n}\sum_{l=1}^{n-1} l\log l$$

$$\le c\cdot n + 2\,\frac{4c}{n}\cdot\int_1^n x\log x\,dx$$

$$= c\cdot n + 2\cdot\frac{4c}{n}\cdot\Big[\frac{x^2}{2}\log x\Big|_1^n - \int_1^n \overbrace{\frac{x^2}{2}\cdot\frac{1}{x}}^{\frac{x}{2}}\,dx\Big]$$



$$= cn + 2\,\frac{4c}{n}\Big[\frac{n^2}{2}\log n - \frac{n^2}{4} + \frac{1}{4}\Big] = c\cdot n + 4cn\cdot\log n - 2cn + \frac{2c}{n}$$

$$\le 4c\cdot n\log n$$

כי $c\cdot n - 2cn + \frac{2c}{n} = \frac{2c}{n} - cn = c\left(\frac{2}{n} - n\right) \underset{(n>1)}{<} 0$

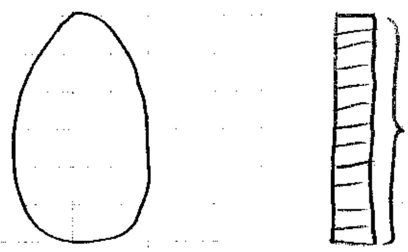
מבוא ל- hashing

ראינו כבר מימוש של ADT מילון שתומך בפעולות insert, delete, search (חיפוש לפי ערך מפתח)

ע"י עצים אדומים - שחורים. במימוש זה, כל הפעולות התבצעו ב-$O(\log n)$. (מספר האיברים במילון)

מימוש יעיל יותר היה ע"י שימוש במערך כאשר כל נתון מאוחסן בתא שהאינדקס שלו הוא המפתח של הנתון. כל הפעולות יבוצעו ב-$\Theta(1)$.

בעיה: לפעמים המפתחות אינם מספרים טבעיים, וגם כאשר הם טבעיים ייתכן שהם נלקחים מקבוצה עצומה בגודלה ולכן נזדקק למערך בגודל לא סביר.



פתרון: פונקציית hash היא פונקציה קלה לחישוב מקבוצת כל המפתחות האפשריים למערך בגודל סביר שמהווה שיכון מוצלח (ללא התנגשויות) כאשר מספר המפתחות בפועל הוא קטן מספיק.

בהמשך הלימוד:

I. איך מתמודדים עם התנגשויות (כאשר הן כן בכל זאת קורות).

II. איך מונעים למנוע מצב של התנגשויות. אפשר לבנות משפחה של פונקציות hash כך ש:

לכל 2 מפתחות אפשריים x ו- y, $P(h(x)=h(y)) \le \frac{1}{m}$ (עבור בחירת h במשפחה.)

מבני נתונים - שיעור השלמה (ראשון)

אם משתמשים במשפחה כזו של פונקציות hash כדי לאחסן n מפתחות במערך בגודל m, כמה התנגשויות צפויות?

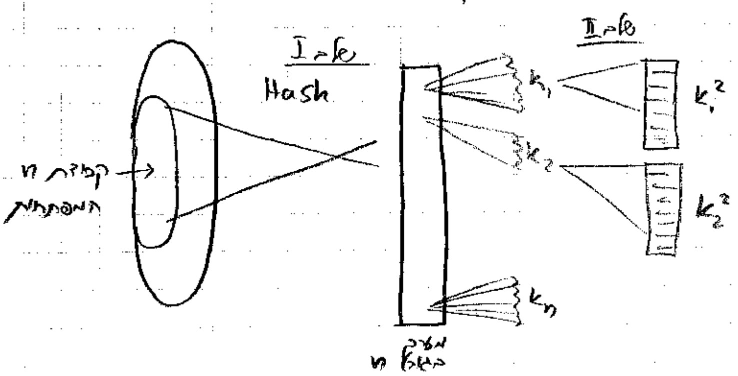
Y - מספר ההתנגשויות (זוגות המפתחות שנכנסו לאותו תא)

$$E(Y) = E\Big(\sum_{\substack{x,y \\ \text{מפתחות שונים} \\ \text{מבין } n \text{ המפתחות}}} Y_{\{x,y\}}\Big) = \sum_{x,y} E(Y_{\{x,y\}}) = \sum P(Y_{\{x,y\}}=1) = \sum_{\substack{x,y \\ \text{מפתחות} \\ \text{שונים}}} \underbrace{P(h(x)=h(y))}_{\le \frac{1}{m}} \le \frac{\binom{n}{2}}{m} \le \frac{n^2}{2m}$$

אי שוויון מרקוב: $P(Y > c \cdot E(Y)) < \frac{1}{c}$

לכן, אם בוחרים $m=n^2$ אז $E(Y) \le \frac{1}{2}$ ואז למשל $P(Y>10) = P_r(Y>20 \cdot E(Y)) < \frac{1}{20}$

שיפור שניתן להשיג כאשר ידועה קבוצת n המפתחות שיש לאחסן



סיבוכיות זמן
כפול מ- hash רגיל אך עדיין $\Theta(1)$ לפעולה.

סיבוכיות מקום
$\sum_{i=1}^{n} k_i^2$

תרגיל

$$E\Big(\sum_{i=1}^{n} k_i^2\Big) = O(n)$$