מבני נתונים ואלגוריתמים - הרצאה 18

1 בינואר 2012

טבלאות גיבוב

פונק' גיבוב $f(x) \to i$ כאשר $x \in V$, V מרחב רב מימדי ו $i \in \{0, ..., N\}$.
רוצים שההתפלגות של הפונק' תהיה אחידה.
אז, בהינתן x בודקים את המקום $f(x)$, ויש שתי אפשרויות:

1. כל מקום מכיל מצביע לרשימה וניתן להכניס מספר ערכים לאותו מקום בטבלה i. (טבלת גיבוב פתוחה, Open Addressing).

2. כל מקום מכיל את הנתונים עצמם, ואם המקום תפוס עושים Rehash (טבלת גיבוב סגורה, Closed Addressing).

מגדירים α להיות היחס בין כמות הנתונים לבין גודל הטבלה.
הזמן הטיפוסי לפעולה בטבלה פתוחה הוא $1+\alpha$ ובטבלה סגורה $\frac{1}{1-\alpha}$.

עלות אתחול

צריך לאתחל את המערך שלנו בהתחלה.
נניח שאתחלנו אותו בגודל k. אנו רוצים שיתקיים $\alpha < 0.5$, אך כאשר $\alpha > 0.5$ נצטרך להגדיל את המערך כדי להחזיר את הα לערך קטן.
נגיד שנכפיל את המערך לגודל $2k$, אך צריך להעביר את הערכים כי פונק' הגיבוב משתנה.
דרך אחת לעשות זאת היא להחזיק את שתי הפונק', החדשה והישנה, ואז כשמחפשים ערך מחפשים אותו בשני המקומות, אחד של הפונק' החדשה ואחד של הישנה.
אפשרות שניה, היא לעבור על כל המערך ולהעביר כל מקום למקום החדש. זה אמנם לוקח יותר זמן, אך זה בדר"כ מה שעושים כי אחרת נצטרך להחזיק הרבה פונק'.
בדר"כ בטבלאות גיבוב עושים גם את ההפך - מחלקים את המערך פי 2 אם α קטנה מספיק.
כלומר אם $\alpha > \rho_0$ מגדילים את המערך פי 2 ואם $\alpha < \rho_1$ מקטינים אותו פי 2. נשים לב שמתקיים $\rho_1 \ll \frac{\rho_0}{2}$, כדי ששינויים קטנים בכמות הנתונים לא יגרמו לנו להכפיל ולחלק הרבה.

Cuckoo Hashing - שיטה לRehash

מחזיקים 2 פונק' גיבוב, f_1 וf_2.
כשרוצים להכניס את x, בודקים את f_1 וf_2. אם אחד מהם לא תפוס, מכניסים שם את x. אם שניהם תפוסים, מכניסים את x באחד מהם ואת מה שהוצאנו מנסים להכניס למקום אחר, לפי הפונק' השניה, וכך הלאה.

השוואת רצפים

נניח שיש לנו טקסט T באורך L ויש לנו מילה M באורך K. אנו רוצים לבדוק האם ואיפה נמצא המילה בטקסט.
האלגוריתם הנאיבי זה לעבור אות אות בטקסט ולבדוק האם היא מתאימה למילה, אך אז העלות היא $O(L \cdot K)$.

שיטת Rabin Karp

1. נגדיר פונק' גיבוב.

2. נחשב $f(M)$.

3. נעבור בכל מקום בטקסט ונחשב $z_i = f(T(i : i+k-1))$.

4. אם $z_i = f(M)$ אז השווה את המילים אות אות. אחרת, $i++$.

גם הפונק' הזו לוקחת $O(L \cdot K)$ בינתיים.

Rolling Hash

פונק' גיבוב היא מתגלגלת אם ניתן לחשב אותה באמצעות הפונק' במילה מוסטת אחת לאחור.
למשל, $f(x) = \sum \text{Ascii}(x_j)$ היא פונק' גיבוב מתגלגלת (אך היא לא טובה כי היא לא מתפלגת טוב).
פונק' אחרת, יותר טובה, היא $f(x) = \sum \text{Ascii}(x_j) \cdot A^j$ כאשר A מס' ראשוני גדול.
פה, כדי "לגלגל", צריך להוריד את האות הראשונה של המילה הקודמת, לחלק הכל בA ולהוסיף את האות האחרונה כפול A^{k-1}.

$$f(x_{i+1}) = \frac{f(x_i) - \text{Ascii}(0)}{A} + \text{Ascii}(i+k) \cdot A^{k-1}$$

עם פונק' גיבוב כזו, אלגוריתם Rabin Karp פועל בעלות של $O(L)$.

אלגוריתם Knuth-Moris-Pratt

אלגוריתם 1 אלגוריתם Knuth-Moris-Pratt

```
j=0
i = 0
while (i < L-k)
 while j<k && i<L-k:
  if T(i + j)=M(j):
   j++
   if (j==k):
    output("victory!")
   endif
  else
   i += j-C(j)
   j = max(C(j), 0)
  endif
 endwhile
 j=C(k)
 i=i+k-C(k)
endwhile
```

כאשר נגדיר את $C(j)$ כאורך הסיפא הגדולה ביותר שהיא גם רישא של המילה $M(0 : j-1)$.
נגדיר $C(0) = -1$ בצורה מלאכותית כדי שיסתדר עם האלגוריתם.
$C(1) = 0$.
אם $M(j-1) = M(C(j-1))$ אז $C(j) = C(j-1) + 1$.
אחרת, $C(j) = 0$ (זה לא בדיוק נכון, יש דוגמאות שזה לא נכון, נראה בשיעור הבא).

אלגוריתם Boyer-Moore

זה אלגוריתם שעובד ברוב המקרים ב$O\left(\frac{L}{k}\right)$.
נכתוב אותו מסודר בשבוע הבא.