Hashing -- ערבול

© 1994 Itai June 4, 1996

בעית היחידות Element Uniqueness

- נתונים מספרים $0 \le x_0,\ldots, x_{n-1} < m$
- מצא אם קיים $i \neq j$ כך ש- $x_i = x_j$.

פתרון ראשון - מיון. זמן $O(n \log n)$.

פתרון שני - Bucket Sort:
עם טבלה בגודל m,

```
for(i=0; i<m; i++)
   T[i]=NULL;
for(i=0; i<n; i++){
   if( T[x_i] == NULL)
      T[x_i] = i ;
   else {output i, T[i];
         return;
   }
}
```

זמן $O(m + n)$.

מקום $O(m)$

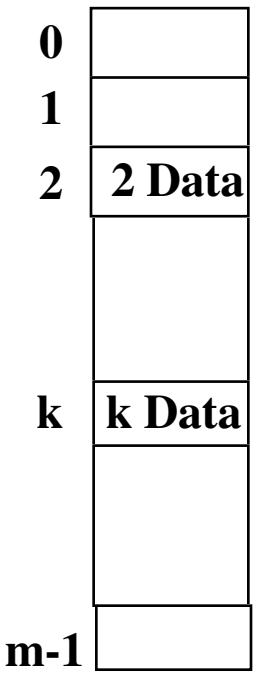
הערה: אפשר לחסוך מקום אם כל מה שרוצים הוא לדעת אם יש חזרה.

מה קורה כאשר m גדול או x_i-ים הם מספרים ממשיים?

מבנה נתונים: מילון - Dictionary:

הפעולות: insert, delete, search

גישה ישירה Direct Addressing (מערך)



זמן של כל פעולה O(1).

כאשר מרחב המפתחות U גדול:

פונקצית ערבול hash

$$h : U \longrightarrow \{0, \ldots, m-1\}$$

נשים את x במקום h(x) בטבלה.

<u>דוגמא</u>: $m = 10 \quad h(x) = x \bmod 10$

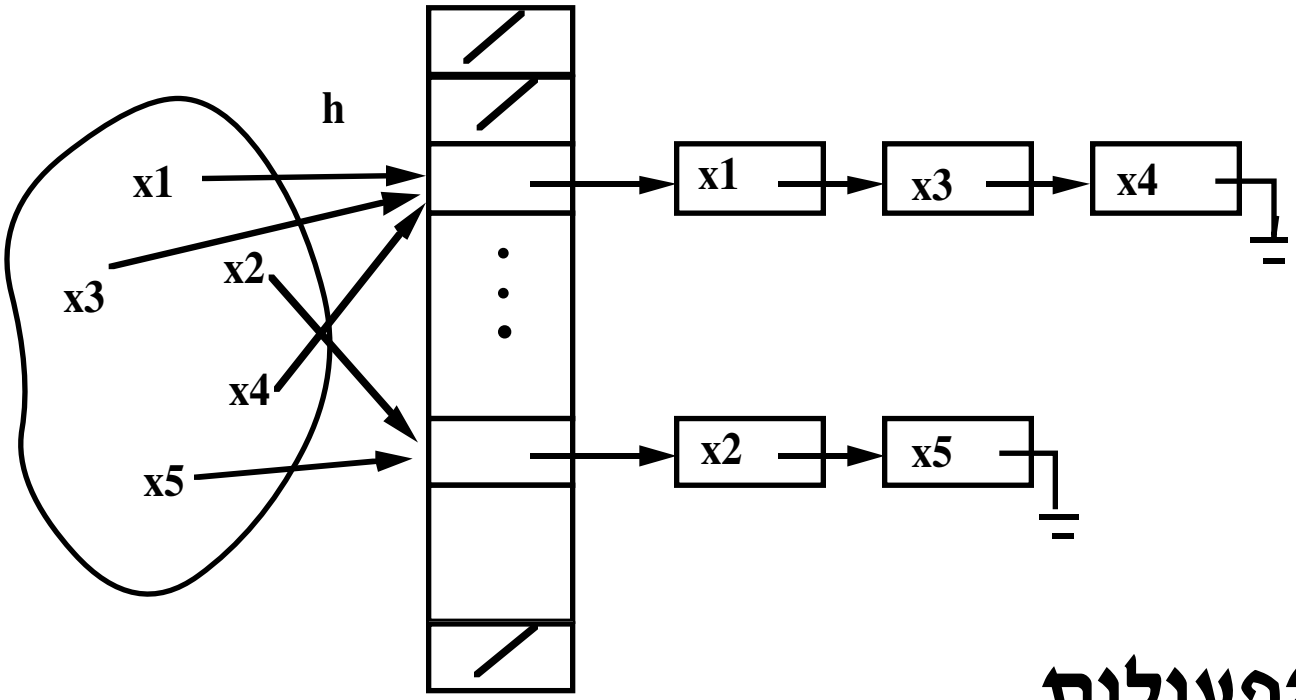
קלט 51, 17, 15, 92, 88, 29

0	
1	51
2	92
3	
4	
5	15
6	
7	17
8	88
9	29

בעיה <u>התנגשות</u>: $x \neq y$ **אך** $h(x) = h(y)$.

<u>דוגמא</u> $h(81) = 1 = h(51)$

פתרון 1: שרשראות



הפעולות

insert (T,x)

הכנס את x בראש הרשימה T[h (x. key)]

זמן O(1)

search (T, k)

חפש איבר עם מפתח k ברשימה T[h (k)] .

זמן Θ(אורך הרשימה)

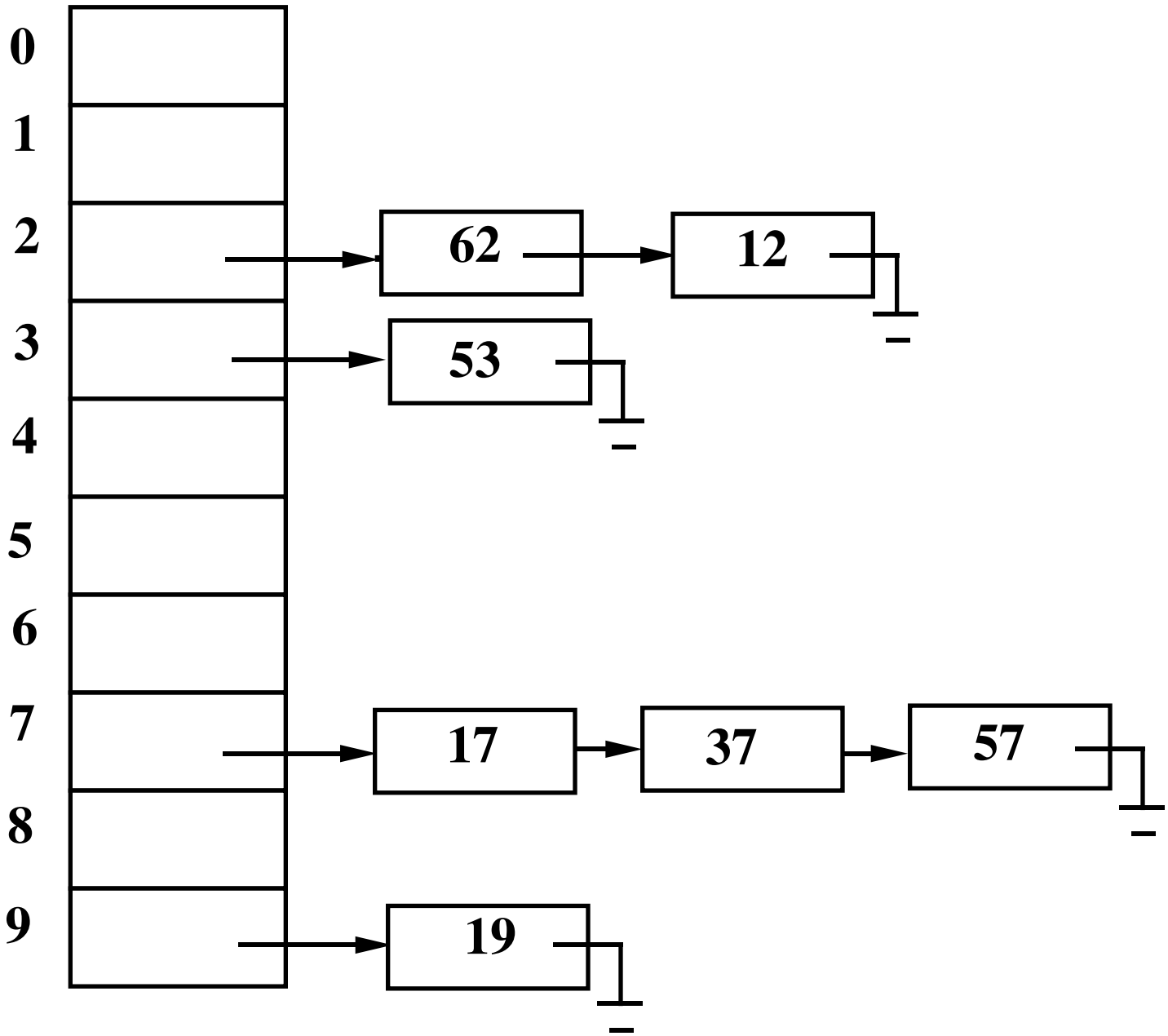
delete (T , x)

סלק את x מהרשימה T[h (x. key)] .

זמן Θ(אורך הרשימה)

דוגמא $h(k) = k \bmod 10$ $m = 10$

קלט 53, 62, 17, 19, 37, 12, 57



מהו אורך הרשימה?

מקרה גרוע = כל האיברים נכנסו לאותה הרשימה...

רצוי פיזור טוב!

הנחת הפיזור האחיד (simple uniform hash)

h מפזרת את המפתחות באופן אחיד.

$$P(\mathrm{h}(key) = i) = \frac{1}{\mathrm{m}}$$

הגדרה יהי n מספר המפתחות, m גודל הטבלה

פקטור העומס (load factor) מוגדר

$$\alpha = n / m$$

משפט האורך הממוצע של שרשרת תחת הנחת הפיזור האחיד הוא α.

מסקנה זמן חיפוש כושל הוא $\alpha + 1$.

זמן חיפוש מוצלח הוא $\frac{\alpha}{2} + 1$.

עבור $n = O(m)$ נקבל $\alpha = O(1)$ ולכן

כל הפעולות $O(1)$.

Open Addressing

שיטה נוספת לפתרון התנגשויות

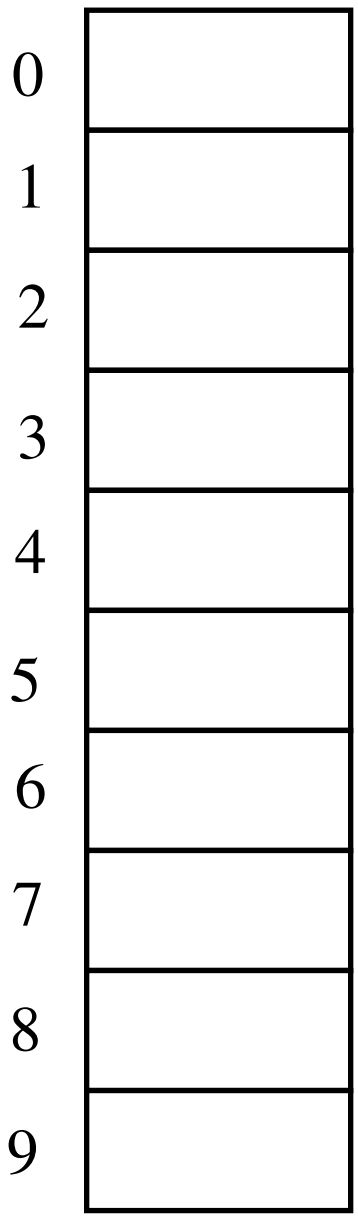
לא נשתמש בשרשראות, אלא כל האיברים יישבו בטבלה.

א. סריקה ליניארית - linear probing

אם אין מקום במקום h(k) שים במקום הבא (מודולו m).

דוגמא **m = 10** **h(k) = k mod 10**

53, 62, 17, 19, 37, 12, 57



חייבים להשאיר לפחות מקום אחד ריק.

משפט (Knuth, vol 3)

אורך חיפוש מוצלח הוא

$$C_N = \frac{1+ 1/(1-\alpha)}{2}$$

אורך חיפוש כושל הוא

$$C_N' = \frac{1+ 1/(1-\alpha)^2}{2}$$

טבלה: אורך החיפוש כפונקציה של α.

α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95
C_N	1.05	1.125	1.214	1.33	1.5	1.75	2.16	3	5.5	10.5
C_N'	1.117	1.218	1.520	1.888	2.5	3.625	6.055	13	50.5	200.5

הוצאת איבר

אי אפשר פשוט למחוק אותו

-- צריך לעבור על כל הקטע עד הרווח הבא ולתקן.

ב. rehashing

נניח שברשותנו סדרה אינסופית של פונקציות ערבול $h_1, h_2, h_3, \ldots$

ננסה לשמור את x במקום $h_1(x)$

אם תפוס, ננסה במקום $h_2(x), \ldots$

α - הסתברות שמקום תפוס

α^i - ההסתברות שב-i הנסיונות הראשונים המקום תפוס

$\alpha^{i-1}(1-\alpha)$: ההסתברות שאורך החיפוש בדיוק i

אורך החיפוש הממוצע

$$\ell = \sum_i i\alpha^{i-1}(1-\alpha) = \frac{1}{1-\alpha}$$

ג. ערבול כפול Double Hashing

נגיע לתוצאות דומות ל-rehashing ע"י שתי פונקציות בלבד h, d

$$h_i(x) = h(x) + i\, d(x) \mod m$$

- הפונקציות h, d בלתי תלויות.
- מה היחס בין d(x) לגודל הטבלה m?

איך נוציא איברים?

פונקציות ערבול

דרישות מפונקצית ערבול:

א. מפזרת היטב.

ב. קלה לחישוב.

פונקציות

א. שיטת החילוק (מודולו)

$$h(x) = x \bmod m$$

רצוי ש- m

- **לא יהיה חזקה של 2 או 10.**
- **יהיה ראשוני שאינו קרוב לחזקה של 2.**

ב. שיטת הכפל עבור קונסטנטה $0 < a < 1$

$$h(x) = \lfloor m\ (a \bullet x \bmod 1) \rfloor$$

דוגמא $a=0.81\ m=10\ \ x=7$

$ax = 5.67$

$ax \bmod 1 = 0.67$

$h(7) = \lfloor 10 * 0.67 \rfloor = 6$

הערך של m אינו קריטי.

ערך טוב לקבוע $a = \bar{\phi} = \frac{\sqrt{5} - 1}{2} \cong 0.6180$

טיפול במחרוזות

<u>הבעיה</u> איך לבנות פונקצית ערבול עבור מחרוזת ארוכה.

<u>פתרון ראשון</u>: בצע XOR בייט בייט.

<u>חסרון</u>: $h(\text{"aa"}) = h(\text{"bb"}) = 0$

$h(\text{"abccba"}) = 0$

<u>חסרון נוסף:</u> $h(x) \le 255$

<u>פתרון שני:</u>

בנה פרמוטציה אקראית על 0..255 ואחסן אותה במערך T

```
int hash(char *s)
{
   int h= 0;
   char *p;
   for(p=s; *p; p++)
        h = T[h]^ *p;          /* XOR */
   return h;
}
```

<u>דוגמא</u>

<u>נניח</u>

T[0] = 23 = 0001 0111

T[118] = 38 = 0010 0110

<u>קלט</u>

"a" = 97 = 0110 0001

hash("a") = T[0] ^ 97 = 0001 0111 ^ 0110 0001 = 0111 0110 = 118

hash("aa") = T[hash("a")] ^ "a" = T[118] ^ 97 =

0010 0110 ^ 0110 0001 = 0100 0111 = 71

<u>מסקנה</u> פתרנו את בעיית האותיות שמופיעות מספר זוגי של פעמים.

כדי להתגבר על מגבלת הטווח
אפשר להשמש בטבלה נוספת ולשרשר את התוצאות.

$$\mathbf{hash(x) = [hash_1(x), hash_2(x)] = hash_1(x)* 256 + hash_2(x)}$$

תוצאה דומה מתקבלת אם במקום $hash_2$ נוסיף 1 לאות הראשונה של המחרוזת ונשתמש ב- $hash_1$.

שימושים לערבול

- **טבלת שמות symbol table.**
- **זכרון אסוציאטיבי**
- **ארגון זכרון בשפה דמוית Lisp, APL**
- **פיזור עומס load balancing**

מגבלות

- **צריך לדעת מראש את n.**
- **יריב יכול למצוא קלט שיתנהג גרוע.**

ערבול אוניברסלי

רנדומיזציה באלגוריתם -- לא בקלט

<u>הגדרה</u> תהי H קבוצת פונקציות מתחום U לקבוצה $\{0,...,m-1\}$.

הקבוצה H היא <u>אוניברסלית</u> אם לכל זוג מפתחות שונים $x,y \in U$

מספר הפונקציות עבורן $h(x) = h(y)$ הוא $|H|/m$.

<u>משפט</u>

תהי H קבוצה אוניברסלית של פונקציות ערבול לתוך טבלה בגודל m.

אם h נבחרה באקראי מתוך H, ונשתמש בה לערבול $n \geq m$ מפתחות,

אזי לכל מפתח המספר הצפוי של התנגשויות ≥ 1.

שימוש בקבוצה אוניברסלית מאפשר לבנות אלגוריתמים רנדומאלים המשתמשים בערבול.

<u>קבוצה אוניברסלית</u>

יהי גודל הטבלה m מספר ראשוני. נבחר r כך שכל המפתחות $< 256^r$

עבור $a=[a_0, \ldots , a_r]$, $0 \leq a_i < m$

נגדיר $h_a(x)$

נשבור את x לבתים $x=[x_0, \ldots , x_r]$

$$h_a(x) = \sum a_i x_i \bmod m$$