

## ערבול (Hashing)

חומר קריאה לשיעור זה:

Chapter 12- Hash tables (pages 219–243)

## ערבול (Hashing)

הפעולות הבסיסיות של מילון הן כזכור חיפוש, הכנסה, והוצאה. שלוש הפעולות מתבצעות בזמן  $O(1)$  כאשר משתמשים במערך.

שלוש הפעולות מתבצעות בזמן  $O(\log n)$  כאשר משתמשים בעץ חיפוש מאוזן או ברשימות דילוגים. מדוע לפיכך משתמשים במבנים אלה?

0	
1	
2	Data 2
k	Data k
m-1	

**תשובה:** לעיתים גודל הטווח של ערכי המפתחות גדול בהרבה ממספר המפתחות בהם משתמשים.

**דוגמא 1:** מספרי תעודת זהות מורכבים מתשע ספרות עשרוניות כלומר קיימים  $10^9$  מפתחות אך בישראל יש פחות מ  $10^7$  אנשים. לפיכך שימוש במערך ינצל פחות מ 1% בודד של הזיכרון המוקצה למערך.

**דוגמא 2:** מספר המחרוזות של אותיות עבריות באורך 30 (באמצעותן ניתן לתאר שם פרטי, שם האב, ושם משפחה של תושבי ישראל) הוא  $22^{30}$  בעוד מספר האנשים קטן מ  $10^7$ .

**אבחנה:** אם נשתמש במערך, זמן כל פעולה יהיה אמנם  $O(1)$ , אך דרישות המקום הן  $O(r)$ , כאשר  $r$  הוא גודל הטווח. ייתכן  $n = o(r)$ .

## ערבול (Hashing)

מימוש מילון באמצעות מערך נקרא גישה ישירה (Direct Addressing): המפתח עצמו משמש כאינדקס במערך. כאשר מרחב המפתחות גדול נחשב את האינדקס מתוך המפתח באמצעות פונקציית ערבול.

המטרה לממש את פעולות החיפוש, הכנסה והוצאה בזמן ממוצע של  $O(1)$ .

נגדיר פונקציית ערבול (hash):  $h: U \rightarrow \{0, \dots, m-1\}$  אשר בהינתן מפתח בתחום  $U$  מחשבת אינדקס בטווח המתאים.

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

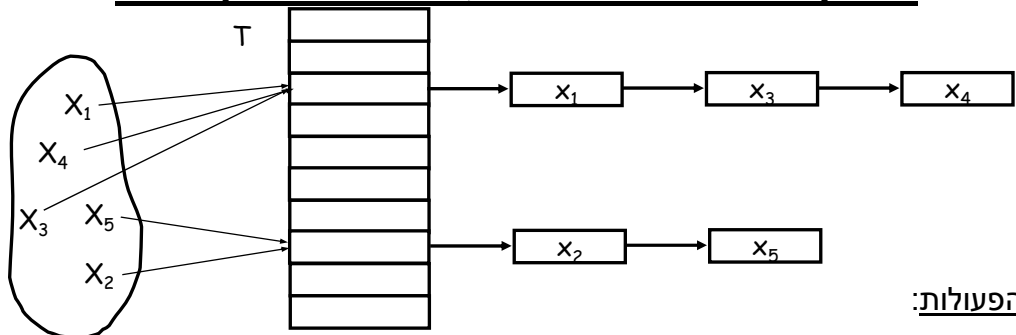
האינדקס של מפתח  $k$  הוא  $h(k)$ .

דוגמא:  $m = 10 \quad h(k) = k \bmod 10$

51, 17, 15, 92, 88, 29

בשיטת הערבול נוצרות התנגשות כאשר  $x \neq y$  אבל  $h(x) = h(y)$ .  
לדוגמא:  $h(81) = 1 = h(51)$ .

## פתרון להתנגשויות באמצעות רשימות מקושרות



הפעולות:

$Insert(T, x)$  הכנס את  $x$  בראש הרשימה  $T[h(x.key)]$ .

זמן במקרה הגרוע ביותר  $O(1)$ .

$Search(T, k)$  חפש איבר עם מפתח  $k$  ברשימה  $T[h(k)]$

זמן במקרה הגרוע ביותר (אורך הרשימה)  $\Theta$ .

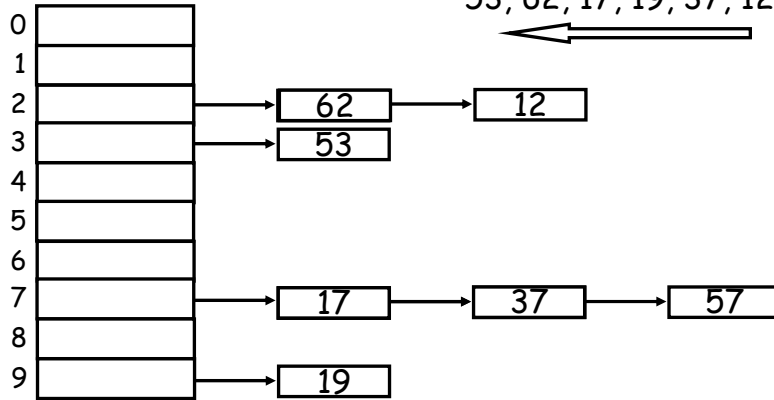
$Delete(T, x)$  סלק את  $x$  מהרשימה  $T[h(x.key)]$ .

זמן במקרה הגרוע ביותר (אורך הרשימה)  $\Theta$ .

## דוגמא

נניח:  $m = 10$   $h(k) = k \bmod m$

קלט: 53, 62, 17, 19, 37, 12, 57



הערה:  $m = 10$  נבחר לצורכי נוחיות ההסבר. נראה בהמשך שבחירה טובה יותר היא 11.

## ניתוח זמנים

במקרה הגרוע ביותר כל האיברים נכנסו לאותה הרשימה ואז זמן חיפוש/הוצאה הוא  $\Theta(n)$ . ברור לפיכך שאין משתמשים בערבול בגלל הזמן המקסימלי לפעולה אלא בגלל הזמן הממוצע לפעולה. נרצה לבחור פונקצית ערבול שמפזרת היטב את המפתחות לרשימות השונות.

נניח לרגע שהצלחנו, כלומר  $h$  מפזרת את המפתחות באופן אחיד.

הנחה זו נקראת הנחת הפיזור האחיד הפשוט (simple uniform hash).

ננתח כעת את זמני החיפוש תחת הנחת הפיזור האחיד הפשוט.

הגדרה: יהי  $n$  מספר המפתחות בשימוש ויהי  $m$  גודל הטבלה.

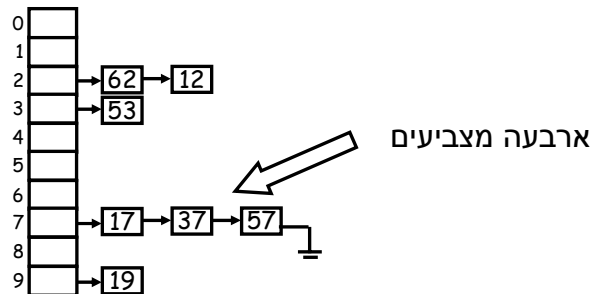
פקטור העומס מוגדר ע"י  $\alpha = n / m$ .

תחת הנחת הפיזור האחיד הפשוט ממוצע אורך שרשרת הוא  $\alpha$ , מכיוון שהאיברים מתחלקים שווה בשווה בין השרשראות השונות.

## ניתוח זמנים (המשר)

**משפט:** בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט הזמן הממוצע לחיפוש כושל הוא  $\Theta(1+\alpha)$ .

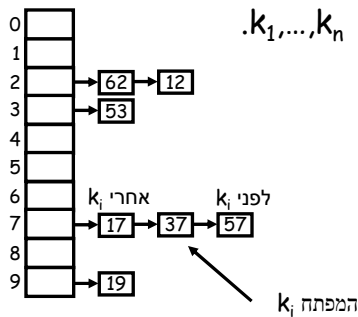
- הוכחה:** בהנחת הפיזור האחיד הפשוט כל מפתח מגיע באקראי לאחת מ- $m$  הרשימות. יהזמן לחיפוש כושל הוא לפיכך הזמן הממוצע לחפש באחת הרשימות עד סופה. אורכה הממוצע של רשימה בהנחת הפיזור האחיד הוא  $\alpha = n / m$ . לפיכך בממוצע יידרש זמן  $\Theta(1 + \alpha)$  (הכולל את זמן בדיקת המצביע בסוף הרשימה).



©cs,Technion

## ניתוח זמנים (המשר)

**משפט:** בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט הזמן הממוצע לחיפוש מוצלח הוא  $\Theta(1+\alpha/2)$ .



**הוכחה:** נאמר שבזמן החיפוש ישנם  $n$  מפתחות שהוכנסו בסדר  $k_1, \dots, k_n$ .

מהו זמן חיפוש הממוצע של המפתח  $k_i$  ?

אחרי מפתח זה נוספו  $n-i$  מפתחות נוספים.

לכן בממוצע גודל הרשימה משמאל למפתח  $k_i$  הוא  $(n-i)/m$ .

מכאן שזמן החיפוש הממוצע של המפתח  $k_i$  הוא  $1 + (n-i)/m$ .

זמן החיפוש הממוצע  $t$  למפתח כלשהו יהיה לפיכך:

$$\begin{aligned}
 t &= \frac{1}{n} \sum_{i=1}^n \left( 1 + \frac{n-i}{m} \right) = 1 + \frac{1}{n \cdot m} \sum_{i=1}^n (n-i) = 1 + \frac{1}{n \cdot m} \sum_{i=0}^{n-1} i = 1 + \frac{1}{n \cdot m} \frac{(n-1)n}{2} \\
 &= 1 + \frac{n-1}{2m} = 1 + \frac{\alpha}{2} - \frac{1}{2m}
 \end{aligned}$$

## ניתוח זמנים (המשך)

לפיכך, כאשר סדר הגודל של מספר המפתחות  $n$  בהם משתמשים הוא כגודל המערך  $m$ , כלומר עבור  $n = O(m)$ , נקבל שגורם העומס קבוע כלומר  $\alpha = O(1)$  ולכן כל הפעולות דורשות זמן ממוצע  $O(1)$ .

לדוגמא עבור 2100 מפתחות מטווח כלשהו  $U$  של מספרים שלמים, נאמר עד  $10^6$ , נוכל להחזיק מערך ובו 700 מקומות ובממוצע אורך כל שרשרת יהיה 3 זמני החיפוש יהיו בהתאם.

## פתרון ללא שרשראות להתנגשויות

לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שיטה כזו נקראת Open addressing. נבחן שלושה פתרונות להתנגשויות: סריקה ליניארית, ערבול נשנה, וערבול כפול.

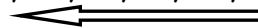
סריקה ליניארית -- linear probing

אם המקום המיועד  $h(k)$  תפוס, שים במקום הבא מודולו  $m$ .

0	17
1	
2	12
3	62
4	53
5	
6	
7	57
8	37
9	19

דוגמא:  $m = 10$   $h(k) = k \bmod m$

קלט: 53, 62, 17, 19, 37, 12, 57



## מספר האיברים המכסימלי בטבלה

ברור שבשיטות open addressing פקטור העומס קטן  $\geq 1$  ( $n \leq m$ ).  
אלגוריתם החיפוש

```
for(i = h(x); T[i] is not empty; i = i+1 % m)
;
T[i] = x;
```

מה קורה כאשר הטבלה מלאה לגמרי?  
מכאן חייבים להשאיר מקום ריק.  
כלומר  $n > m$  ופקטור העומס  $\alpha < 1$

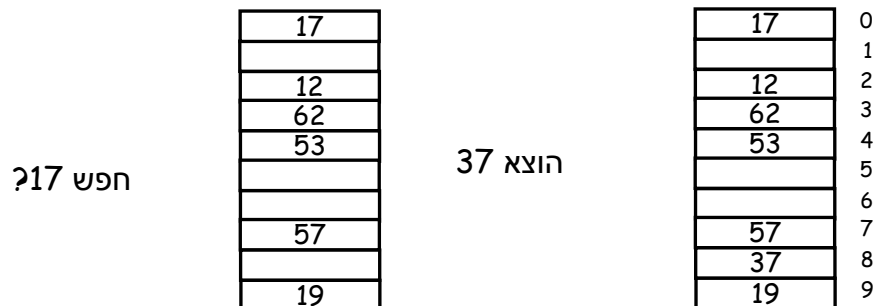
## הוצאה בשיטת linear probing

כיצד נוציא איברים?

•לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.

דוגמא:  $m = 10$   $h(k) = k \bmod m$

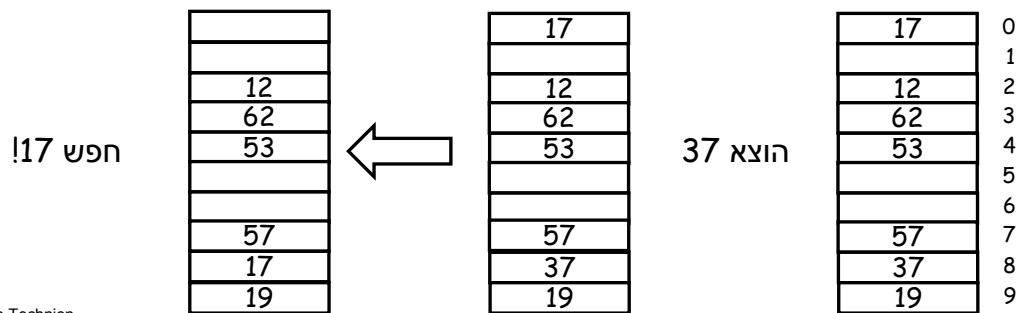
קלט: 53, 62, 17, 19, 37, 12



## linear probing הוצאה בשיטת (המשך)

פתרון:

בזמן ההוצאה הוצא את כל האיברים עד לרווח הבא, והכנס את כולם חזרה פרט לאיבר שרוצים להוציא.



©cs,Technion

## הוצאה בשיטת המציבה



• כדי לא לנתק את שרשרת החיפוש תינתק נסמן את מקום האיבר שהוצא בסימן `deleted`. (שיטת המציבה)

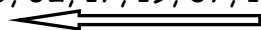
• בזמן חיפוש `x`, במידה וניתקל בסימן `delete`, נמשיך את סריקת הרשימה עד למציאת `x` או עד שנגיע למקום ריק (המסומן ב- `Null`).

• בזמן הכנסת `x`, במידה וניתקל בסימן `delete`, נשתמש במקום זה לשמירת `x`, אחרת נשמור את `x` במקום הריק בסוף הרשימה (המסומן ב- `Null`).

0	17
1	
2	12
3	62
4	53
5	
6	
7	57
8	27
9	19

דוגמא:  $m = 10 \quad h(k) = k \bmod m$

קלט: 53, 62, 17, 19, 37, 12, 57



הוצא 37, חפש 17, הכנס 27

## יתרונות וחסרונות

- היתרון העיקרי של שיטת המציבה הוא פשטות. אבל ...
- כאשר השימוש דורש הוצאות, אורך החיפוש תלוי גם באיברים שכבר הוצאו ולא רק באיברים שכרגע במבנה.

### דוגמאות לשימוש במילון ללא הוצאות:

- טבלה של שמות משתנים בהרצת תוכנית (Symbol Table).
- מספרי תעודות זהות אינם ממוחזרים.

נתאר כעת שיטות נוספות ל-**open addressing**. שיטות אלו שימושיות במיוחד במימושי מילון ללא הוצאות. כאשר יש צורך בהוצאות, עדיפה שיטת הרשימות המקושרות.

## ערבול נשנה -- Rehashing

נניח שברשותנו סדרה אינסופית של פונקציות ערבול:  $h_0, h_1, h_2, \dots$ .  
 ננסה לשמור את  $x$  במקום  $h_0(x)$ .  
 אם תפוס, ננסה במקום  $h_1(x)$ . נמשיך עד שנצליח.  
 לדוגמא, בסריקה ליניארית מתקיים  $h_i(x) = h(x) + i$ .



## ערבול כפול -- Double Hashing

נגיע לתוצאות דומות לערבול נשנה ע"י שתי פונקציות בלבד  $h, d$ .

$$h_i(x) = h(x) + i d(x) \quad \text{כאשר:}$$

הפונקציות  $h, d$  נבחרות באופן בלתי תלוי.

מהו היחס בין  $d(x)$  לגודל הטבלה  $m$  ?

גודל הטבלה  $m$  ו-  $d(x)$  צריכים להיות מספרים זרים כך ש  $h_0(x), \dots, h_{m-1}(x)$  תכסה את כל האינדקסים האפשריים בתחום  $\{0, \dots, m-1\}$ . לפיכך נוח לבחור את  $m$  להיות מספר ראשוני.

הוצאות נעשות ע"י שימוש בסימון `delete`.

## ניתוח זמנים עבור Rehashing

הנחת הפיזור האחיד: הסדרה  $(h_0(x), h_1(x), h_2(x), \dots, h_{m-1}(x))$  היא פרמוטציה אקראית של  $\{0, \dots, m-1\}$ .

משפט: בהנחת הפיזור האחיד, בשיטת ערבול rehashing מתקיים:

זמן ממוצע של חיפוש כושל קטן מ-  $1/(1 - \alpha)$

זמן ממוצע של חיפוש מוצלח קטן מ-  $\frac{1}{\alpha} \ln \frac{1}{1-\alpha} + \frac{1}{\alpha}$

$\alpha$	חיפוש כושל	חיפוש מוצלח
0.5	2	<del>3.386</del> 1.836
0.8	5	3.261
0.9	10	3.669

$$\text{ע"פ הביטוי המקורי} \\ H_m - H_{m/2} \rightarrow \ln 2$$

הוכחה בספר הלימוד: Introduction to algorithms, Cormen et al., pp 238-239

## ניתוח זמנים עבור סריקה ליניארית

משפט: בהנחת הפיזור האחיד הפשוט, בשיטת ערבול open addressing בסריקה ליניארית מתקיים:

$$\frac{1}{2} \left( 1 + \frac{1}{(1-\alpha)^2} \right) \text{ זמן ממוצע של חיפוש כושל קטן מ-}$$

$$\frac{1}{2} \left( 1 + \frac{1}{(1-\alpha)} \right) \text{ זמן ממוצע של חיפוש מוצלח קטן מ-}$$

$\alpha$	ערבול נשנה חיפוש כושל	ערבול נשנה חיפוש מוצלח	סריקה ליניארית חיפוש כושל	סריקה ליניארית חיפוש מוצלח
0.5	2	1.836	2.5	1.5
0.8	5	3.261	13.0	3.0
0.9	10	3.669	50.5	5.5

הוכחה בספר : Knuth, The art of computer programming, Vol 3, 1973

הערה: המשפט מראה שאפשר להשתמש בסריקה ליניארית, כל עוד הטבלה לא מלאה מדי (80%).

## פונקציות ערבול

דרישות מפונקציות ערבול: מפזרת היטב וקלה לחישוב.

$$\text{שיטת החילוק מודולו } m: h(x) = x \bmod m$$

רצוי ש- $m$ :

לא יהיה חזקה של 2 או 10. בחזקות של 2 פונקצית הערבול מסתמכת רק על  $\log_2(m)$  הביטים הראשונים (LSB). בחזקות של עשר, פונקצית הערבול מסתמכת רק על  $\log_{10}(m)$  הספרות הראשונות. רצוי שפונקציות הערבול ישתמשו בכל האינפורמציה הנמצאת במפתח כדי לקרב עד כמה שניתן את הנחת הפיזור האחיד.

יהיה ראשוני שאינו קרוב לחזקה של 2. חזקות קרובות של 2 גורמות לפיזור לא אחיד כאשר המפתחות כתובים בבסיס שהוא חזקה של 2, למשל מחרוזות תווים נכתבות בבסיס  $2^8 = 256$ .

הערה: רצוי לבדוק את פונקצית הערבול על תת קבוצה של מפתחות "אמיתיים" וכך לוודא שהנחת הפיזור האחיד מתקיימת בקרוב.

## פונקציות ערבול (המשך)

שיטת הכפל עבור קבוע  $0 < a < 1$

• הכפל את המפתח  $k$  בקבוע  $a$ .

• מצא את החלק השבור של התוצאה.

• הכפל את החלק השבור ב-  $m$  ועגל כלפי מטה:  
 $h(k) = \lfloor m \cdot (a \cdot k \bmod 1) \rfloor$

הערך של  $m$  אינו קריטי.

ערך של  $a$  הגורם לפיזור טוב הוא:  $a = (\sqrt{5} - 1) / 2 = 0.61803 \dots$

דוגמא:  $m = 10000$   $k = 123456$

$$h(k) = \lfloor 10000 \cdot (123456 \cdot 0.61803 \bmod 1) \rfloor$$

$$= \lfloor 10000 \cdot (76300.0041151 \bmod 1) \rfloor$$

$$= \lfloor 10000 \cdot 0.0041151 \rfloor$$

©cs,Technion

$$= \lfloor 41.151 \dots \rfloor = 41$$

## פונקציות ערבול למחרוזות ארוכות

נשתמש בקוד ascii: "a" = 97 = 0110 0001

"b" = 98 = 0110 0010

וכך הלאה ...

פתרון נאיבי: בצע xor בייט בייט.

לדוגמא:  $h("ab") = h( (0110\ 0001) \text{ xor } (0110\ 0010) ) = (0000\ 0011) = 3$

חסרון ראשון:  $h("aa") = h( (0110\ 0001) \text{ xor } (0110\ 0001) ) = (0000\ 0000) = 0$

$h\{ "bb" \} = h( (0110\ 0010) \text{ xor } (0110\ 0010) ) = (0000\ 0000) = 0$

התוצאה אפס מתקבלת כאשר כל אות מופיעה מספר זוגי של פעמים:  $h\{ "abccba" \} = 0$

חסרון שני: טווח הערכים מוגבל.  $h(x) \leq 255$

## פונקציות ערבול למחרוזות ארוכות (המשך)

פתרון עדיף: (על עקרון השפעת הפרפר מגינאה על מזג האוויר - שינוי קטן מביא שינוי גדול)  
 בחר פרמוטציה אקראית  $(\pi_0, \dots, \pi_{255})$  של  $0 \dots 255$  ואחסן אותה במערך T.  
 בצע xor בין האות הראשונה של המפתח  $s_0$  והערך  $T[0]$ .  
 התוצאה  $a_1$  נמצאת בתחום  $0 \dots 255$ .



בשלב ה- $i$  בצע xor בין האות  $s_i$  של המפתח והערך  $T[a_i]$  כאשר  $a_i$  היא תוצאת ה-xor בשלב הקודם.

תוצאת פונקציית הערבול היא תוצאת ה-xor עם האות האחרונה של המפתח כלומר

$$\text{hash}(\text{key}) = s_n \text{ xor } a_n$$

דוגמא:  $\text{hash}(\text{aa})$

$$\text{hash}(a) = T[0] \text{ xor } 97 = 0001\ 01111 \text{ xor } 0110\ 0001 = 0111\ 0110 = 118$$

$$\text{hash}(aa) = T[\text{hash}(a)] \text{ xor } a = T[118] \text{ xor } 97 = 0010\ 0110 \text{ xor } 0110\ 0001 = 71$$

## פונקציות ערבול למחרוזות ארוכות (המשך)

מימוש של פונקציית הערבול:

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

## פונקציות ערבול למחרוזות ארוכות (המשך)

כדי להתגבר על בעיית הטווח ניתן להשתמש בשתי פרמוטציות  $T_1, T_2$  ולשרשר את התוצאות.

$$\text{hash}(k) = [\text{hash}_1(k), \text{hash}_2(k)] = \text{hash}_1(k) * 256 + \text{hash}_2(k)$$

גודל הטווח החדש, כלומר גודל טבלת הערבול, הוא  $2^{16} = 256^2$ . (בעוד גודל כל  $T_i$  הוא 256).

ניתן להגיע לטווח הרצוי ע"י שימוש במספר קטן של פרמוטציות נוספות.

תוצאה דומה מתקבלת אם במקום  $\text{hash}_2$  נוסף 1 לאות הראשונה של המחרוזת ונשתמש שוב בפונקצית הערבול  $\text{hash}_1$ .

$$\text{לדוגמא: } \text{hash}(acb) = \text{hash}_1(acb) * 256 + \text{hash}_1(abc)$$

## ערבול אוניברסלי

לכל בחירה של פונקצית ערבול קיימת סדרה גרועה של מפתחות כך שתוצר רשימה באורך מקסימלי. תכונה זו יכולה ליצור בעיה.

לדוגמא, יתכן מתכנת המשתמש באופן עקבי בשמות מסוימים למשתני התוכנית שהוא כותב ולצערו פונקצית הערבול הבונה את ה-**symbol table** ממפה את כל השמות הנ"ל לאותו המקום בטבלת הערבול. לפיכך **כל תוכנית מחשב** של משתמש זה אינה יעילה כפי שיכולה הייתה להיות!

**הפתרון:** לבחור באקראי, בזמן יצירת טבלת ערבול, פונקצית ערבול מתוך קבוצת פונקציות שהוגדרה מראש. נרצה שקבוצת הפונקציות תהיה כזו, שעבור כל סדרת מפתחות, בחירה אקראית של אחת הפונקציות תיצור פיזור טוב.

**הגדרה:** תהי  $H$  קבוצת פונקציות ערבול מתחום  $U$  לקבוצה  $\{0, \dots, m-1\}$ . הקבוצה  $H$  נקראת **אוניברסלית** אם לכל זוג מפתחות שונים  $x, y \in U$  מספר הפונקציות עבורן  $h(x) = h(y)$  הוא  $|H|/m$ .

**הערה:** ההסתברות  $p$  שבבחירה אקראית של פונקצית ערבול מתוך  $H$ , מפתח  $x$  יתנגש עם מפתח  $y$  היא  $p = (|H|/m)/|H| = 1/m$ .

נראה כעת ששימוש בקבוצה אוניברסלית גורם לפיזור טוב. אח"כ נראה כיצד לבנות קבוצה כזו.

## ערבול אוניברסלי (המשך)

**משפט:** תהי  $H$  קבוצה אוניברסלית של פונקציות ערבול לתוך טבלה  $T$  בגודל  $m$ . אם  $h$  נבחרה באקראי מתוך  $H$ , ונשתמש בה לערבול  $n$  מפתחות כלשהם, אזי לכל מפתח, המספר הצפוי של התנגשויות בשיטת הרשימות המקושרות שווה ל-  $(n-1)/m$ .

**הוכחה:** ראינו שהסתברות להתנגשות של מפתח מסוים  $x$  עם מפתח מסוים  $y$  היא  $p = 1/m$ . המספר הצפוי של התנגשויות של מפתח מסוים  $x$  עם מפתח כלשהו נתון לפיכך ע"י:

$$\begin{aligned}\bar{L}_x &= E_H \left( \sum_{\{y \in T \mid y \neq x\}} \delta(h(x), h(y)) \right) = \sum_{\{y \in T \mid y \neq x\}} E_H (\delta(h(x), h(y))) \\ &= \sum_{\{y \in T \mid y \neq x\}} P(h(x) = h(y)) = \sum_{\{y \in T \mid y \neq x\}} \frac{1}{m} = \frac{n-1}{m} < \alpha\end{aligned}$$

**מסקנה:** מספר ההתנגשויות הצפוי לכל מפתח קטן מגורם העומס.

## בניית קבוצה אוניברסלית

נבחר את גודל הטבלה להיות מספר ראשוני  $m$ .  
נשבור כל מפתח  $x$  ל-  $r+1$  חלקים באורך קבוע  $x = [x_0, \dots, x_r]$ . (למשל באורך בייט = 8 ביטים). מספר הביטים יבחר כך שהערך של  $x_i$  יהיה לכל היותר  $m$ .  
לכל סדרה  $a = [a_0, \dots, a_r]$  מהתחום  $\{0, \dots, m-1\}^{r+1}$  נגדיר פונקצית ערבול  $h_a(x)$  בצורה הבאה:

$$h_a(x) = \left( \sum_{i=0}^r a_i x_i \right) \bmod m$$

קבוצת הפונקציות  $H$  מוגדרת להיות  $\cup_a \{h_a\}$  ומספר הפונקציות בקבוצה זו הוא:  $m^{r+1}$ .  
**דרך השימוש בשיטה זו:** כאשר משתמש מגדיר טבלת ערבול  $T$  בגודל  $m$ , התוכנית מגרילה מספר  $a$  ומשתמשת בפונקצית הערבול  $h_a$  לכל הפעולות הנעשות בטבלת ערבול זו.

**דוגמא:**  $m=257$ , טווח המפתחות  $0-2^{24}$ . נשבור כל מפתח לשלושה חלקים באורך 8 ביטים. נניח שהוגרלו המספרים  $a=[248, 223, 101]$ . בהינתן המפתח  $x = 1025 = 4 \cdot 2^8 + 1 \cdot 2^0 = [0, 4, 1]$  הנוסחה:  
 $(248 \cdot 0 + 223 \cdot 4 + 101 \cdot 1) \bmod 257 = 993 \bmod 257 = 222$

## בניית קבוצה אוניברסלית (המשך)

משפט: קבוצת הפונקציות  $H = \{h_a\}$  שהוגדרה בשקף הקודם היא קבוצה אוניברסלית.

הוכחה: יהיו  $x = [x_0, \dots, x_r]$  ו-  $y = [y_0, \dots, y_r]$  מפתחות שונים. ללא הגבלת הכלליות נניח  $x_0 \neq y_0$ .

טענה לכל ערך קבוע של  $a_1, \dots, a_r$  קיים ערך יחיד ל- $a_0$  כך שמתקיים  $h_a(x) = h_a(y)$ . הערך  $a$  מתקבל מהפתרון היחיד למשוואה:

$$h_a(x) - h_a(y) = \sum_{i=0}^r a_i(x_i - y_i) \equiv 0 \pmod{m}$$

$$a_0(x_0 - y_0) \equiv -\sum_{i=1}^r a_i(x_i - y_i) \pmod{m} \quad \text{הניתנת לשכתוב כדלקמן:}$$

בהנחה שהטענה נכונה, נובע שכל זוג מפתחות  $x, y$  מתנגשים עבור  $m^r$  ערכים של  $a$  שכן לכל ערך של  $(a_1, \dots, a_r)$  קיים ערך אחד  $a_0$  עבורו  $x, y$  מתנגשים.

נזכור שמספר הפונקציות ב-  $H$  הוא  $m^{r+1}$ .

לפיכך, ההסתברות ש-  $x$  ו-  $y$  יתנגשו היא  $1/m = m^r/m^{r+1}$  כנדרש מקבוצה אוניברסלית.

## בניית קבוצה אוניברסלית (המשך)

טענה: למשוואה הבאה יש פתרון והפתרון יחיד.

$$a_0(x_0 - y_0) \equiv -\sum_{i=1}^r a_i(x_i - y_i) \pmod{m}$$

נזכר שכאשר  $m$  ראשוני מתקיים:  $Z_m$  הוא שדה ולכן עבור כל מספר  $z \neq 0$  קיים מספר  $w$  יחיד

$$z \cdot w = 1 \pmod{m}$$

$$\text{למשל } 2 \cdot 2 = 1 \pmod{3} \quad 1 \cdot 1 = 1 \pmod{3}$$

$$z = x_0 - y_0 \neq 0$$

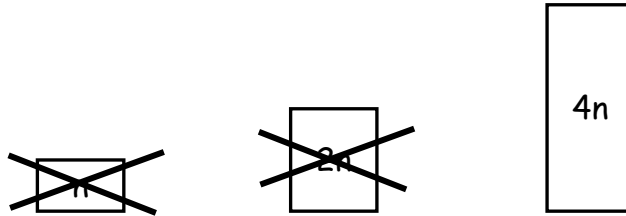
במשוואה הנתונה הנחנו  $z = x_0 - y_0 \neq 0$  ונקבל את הפתרון היחיד ל- $a_0$ .

$$a_0 \equiv \left[ -\sum_{i=1}^r a_i(x_i - y_i) \right] (x_0 - y_0)^{-1} \pmod{m}$$

## מגבלות לערבול

צריך לדעת מראש סדר גודל למספר האיברים שמתעתדים להכניס למבנה (n).

פתרון חלקי: כאשר טבלת ערבול מתמלאת ניתן להקצות טבלה חדשה בגודל כפול, להכניס את כל האיברים לטבלה החדשה, ולהיפטר מהטבלה הישנה.



הזמן המשוערך הממוצע יהיה  $O(1)$  למרות שמדי פעם תתבצע פעולה יקרה.

## שימוש: בעיית היחידות Element Uniqueness

נתונים מספרים  $0 \leq x_0, \dots, x_{n-1} < T$

מצא אם קיים  $i \neq j$  עבורו  $x_i = x_j$ .

פתרון ראשון – מיון: זמן  $O(n \log n)$

פתרון שני – ערבול:

הכנס את המספרים לטבלת ערבול בגודל  $O(n)$   
(שיתכן וקטנה בהרבה מ- $T$ ).

בזמן התנגשות, בדוק שוויון.

זמן ממוצע  $O(n)$ .