Lecture7 of Geiger & Itai's slide brochure
www.cs.technion.ac.il/~dang/courseDS

ערבול (Hashing)

חומר קריאה לשיעור זה:

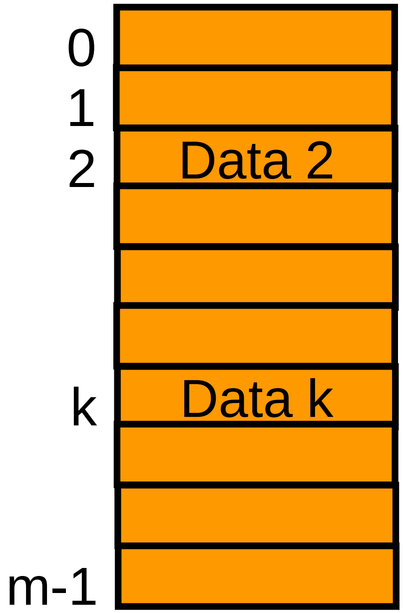
Chapter 12- Hash tables (pages 219—243)

©Geiger & Itai, 2001

ערבול (Hashing)

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

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

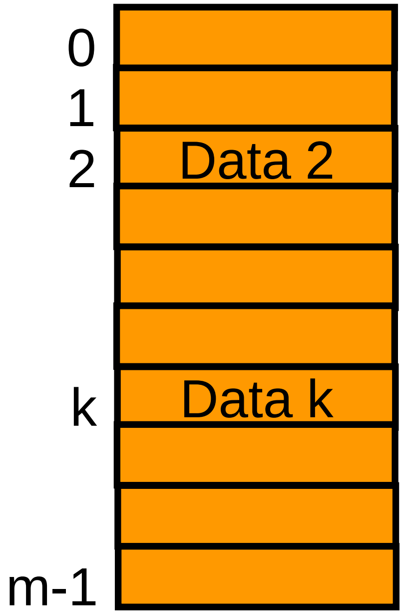


©cs,Technion

ערבול (Hashing)

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

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



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.

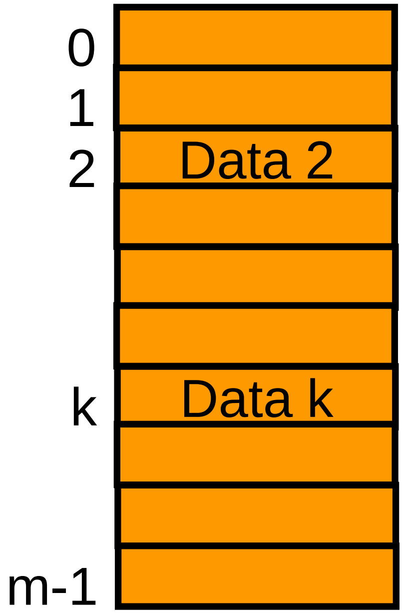
דוגמא 2: מספר המחרוזות של אותיות עבריות באורך 30 (באמצעותן ניתן לתאר שם פרטי, שם אמצעי, ושם משפחה של תושבי ישראל) הוא 22^{30} בעוד מספר האנשים קטן מ 10^7.

©cs,Technion

ערבול (Hashing)

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

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



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 אמצעי, ושם משפחה של תושבי ישראל) הוא 22^{30} בעוד מספר האנשים קטן מ 10^7.

אבחנה: אם נשתמש במערך, זמן של כל פעולה יהיה אמנם O(1) אך הקבוע המתחבא ב-O יהיה גדול.לעומת זאת בעץ חיפוש, הקבוע יהיה קטן וכן $\log_2 n \le 40$ כאשר n הוא מספר תושבי ישראל (או סין).

©cs,Technion

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

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

האינדקס של מפתח k הוא h(k).

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

51, 17, 15, 92, 88, 29

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

©cs,Technion

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

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

האינדקס של מפתח k הוא h(k).

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

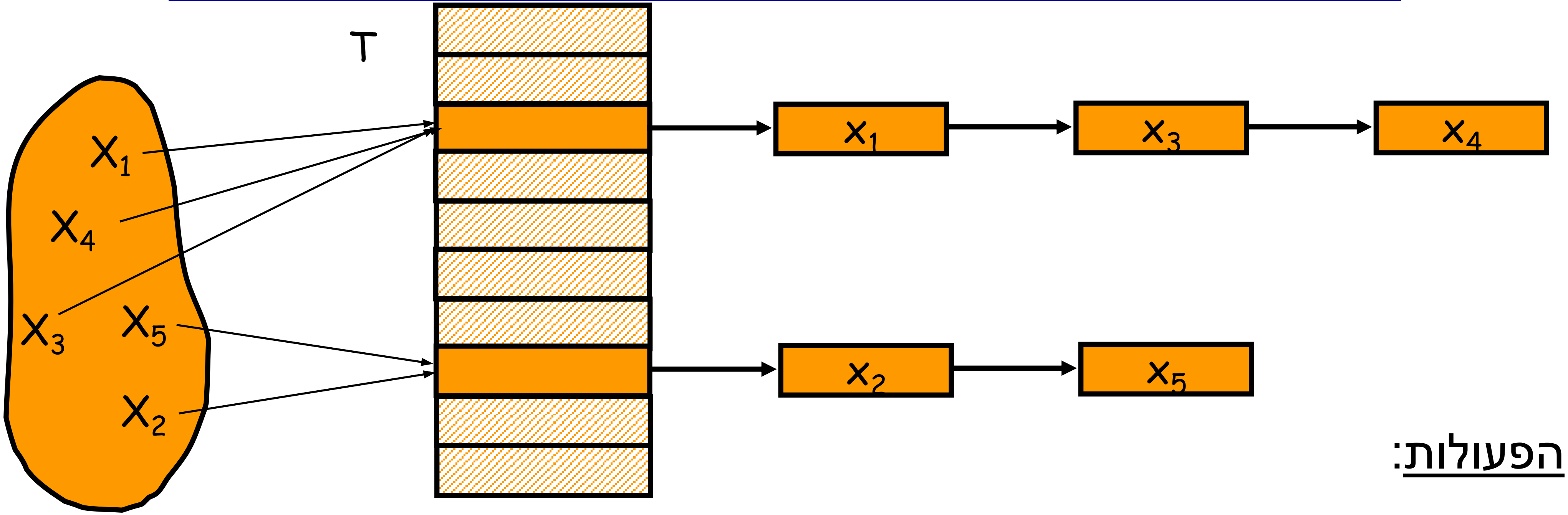
51, 17, 15, 92, 88, 29

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

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

©cs,Technion

פתרון להתנגשויות באמצעות רשימות מקושרות



<u>הפעולות</u>:

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)Θ.

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

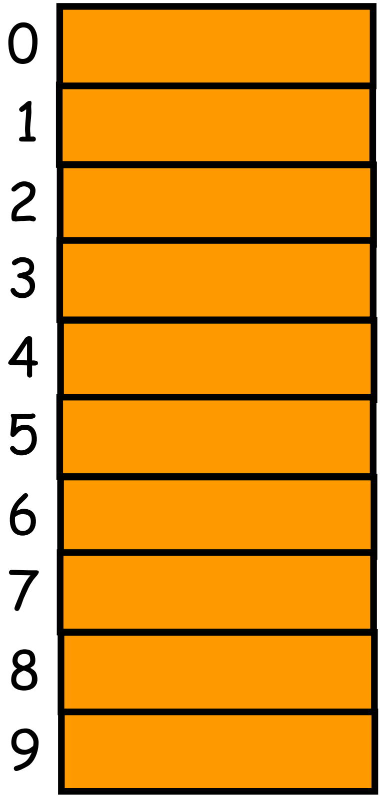
זמן במקרה הגרוע ביותר (אורך הרשימה) Θ.

©cs,Technion

דוגמא

נניח: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

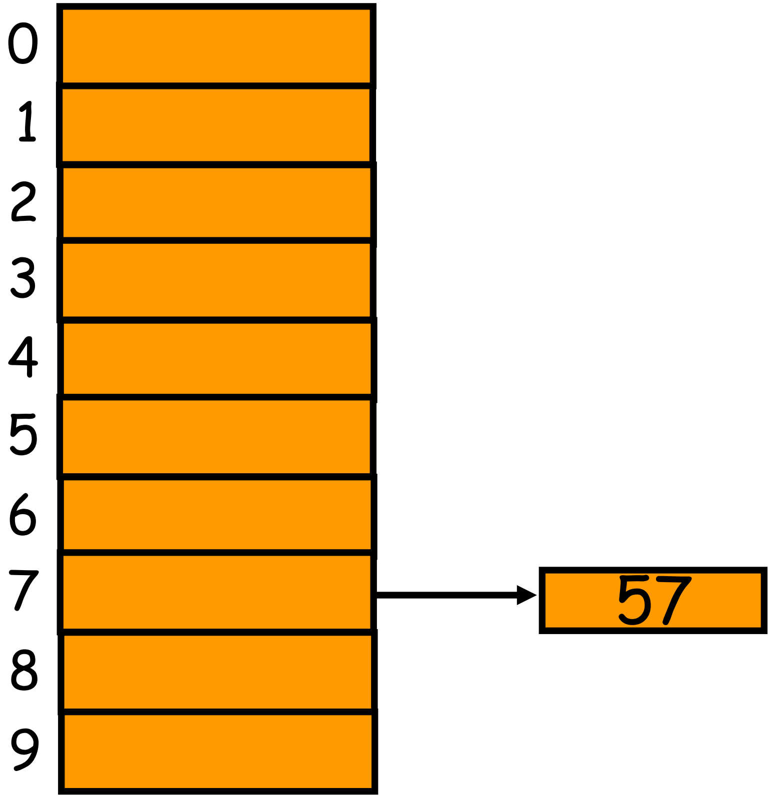


©cs,Technion

דוגמא

נניח: h(k) = k mod m m = 10

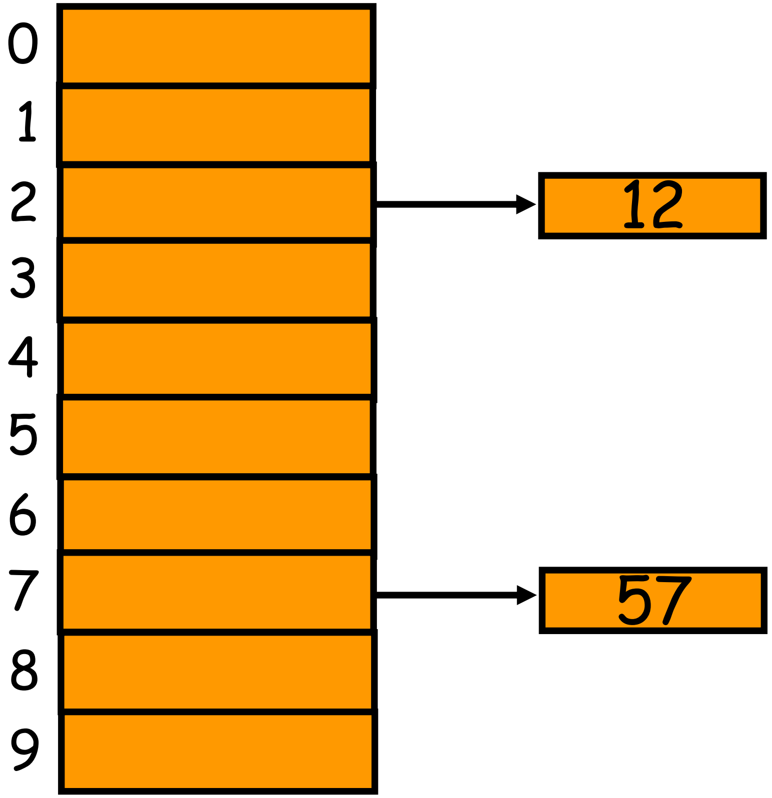
קלט: 53, 62, 17, 19, 37, 12, 57



דוגמא

הניח: $h(k) = k \bmod m \quad m = 10$

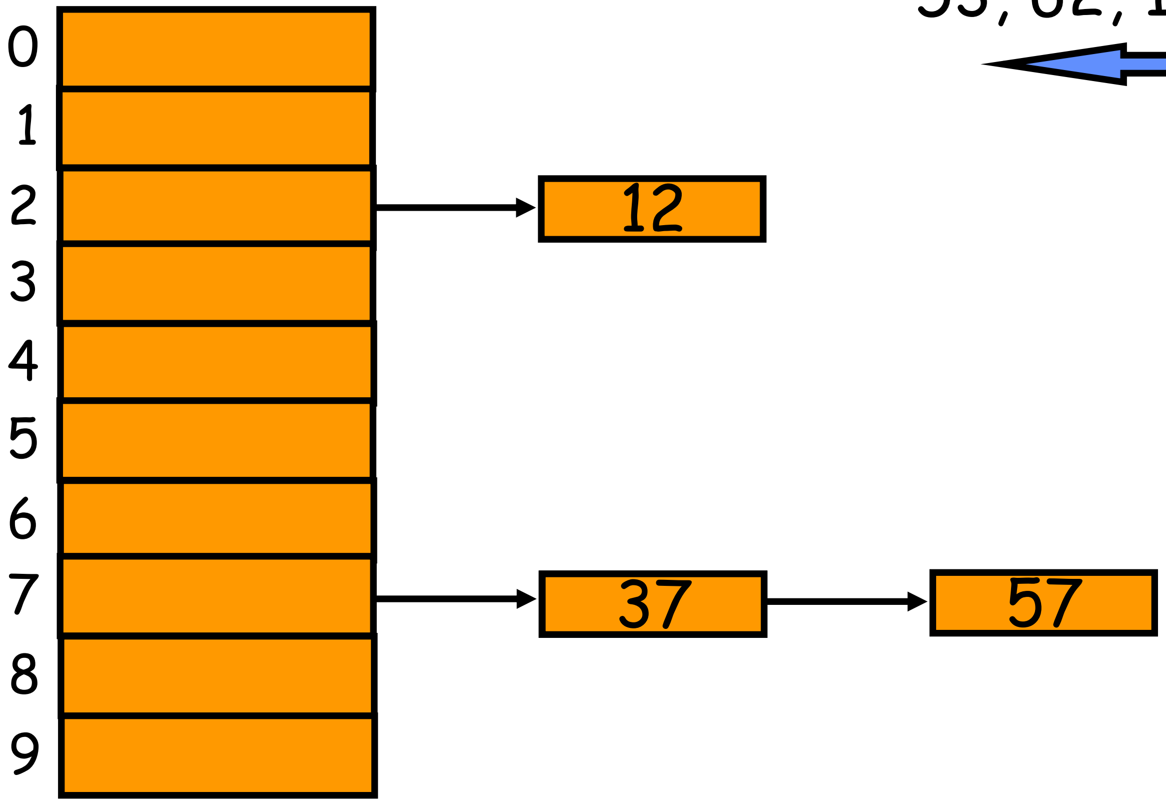
קלט: 53, 62, 17, 19, 37, 12, 57



דוגמא

נניח: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

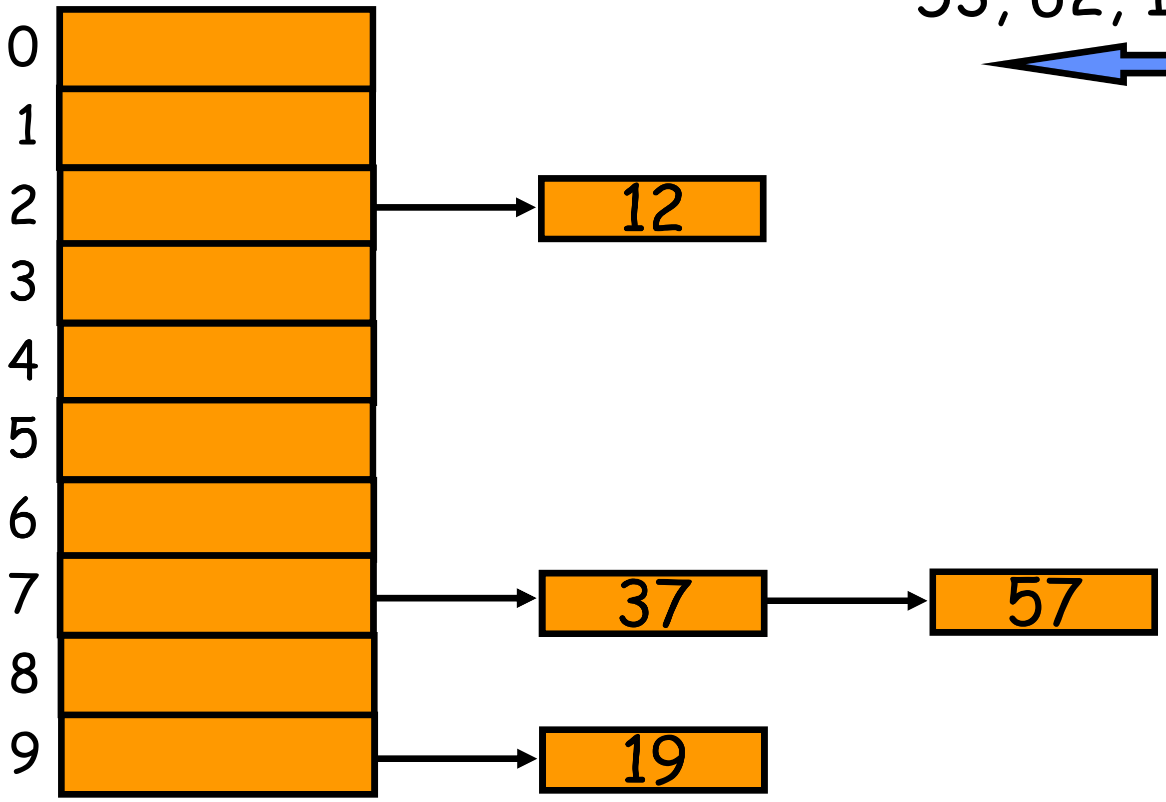


©cs,Technion

דוגמא

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

קלט: 53, 62, 17, 19, 37, 12, 57

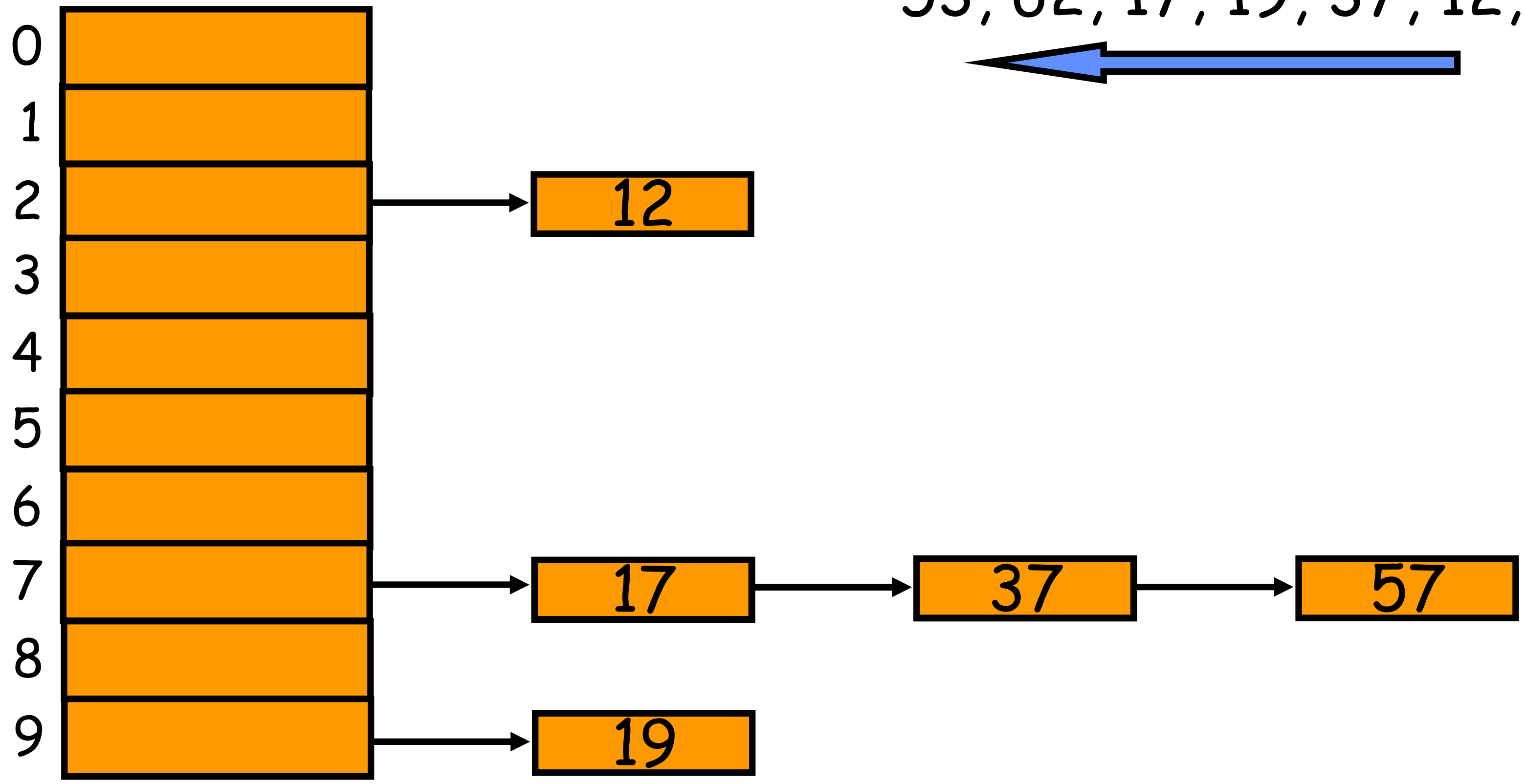


©cs,Technion

דוגמא

נניח: $h(k) = k \bmod m \quad m = 10$

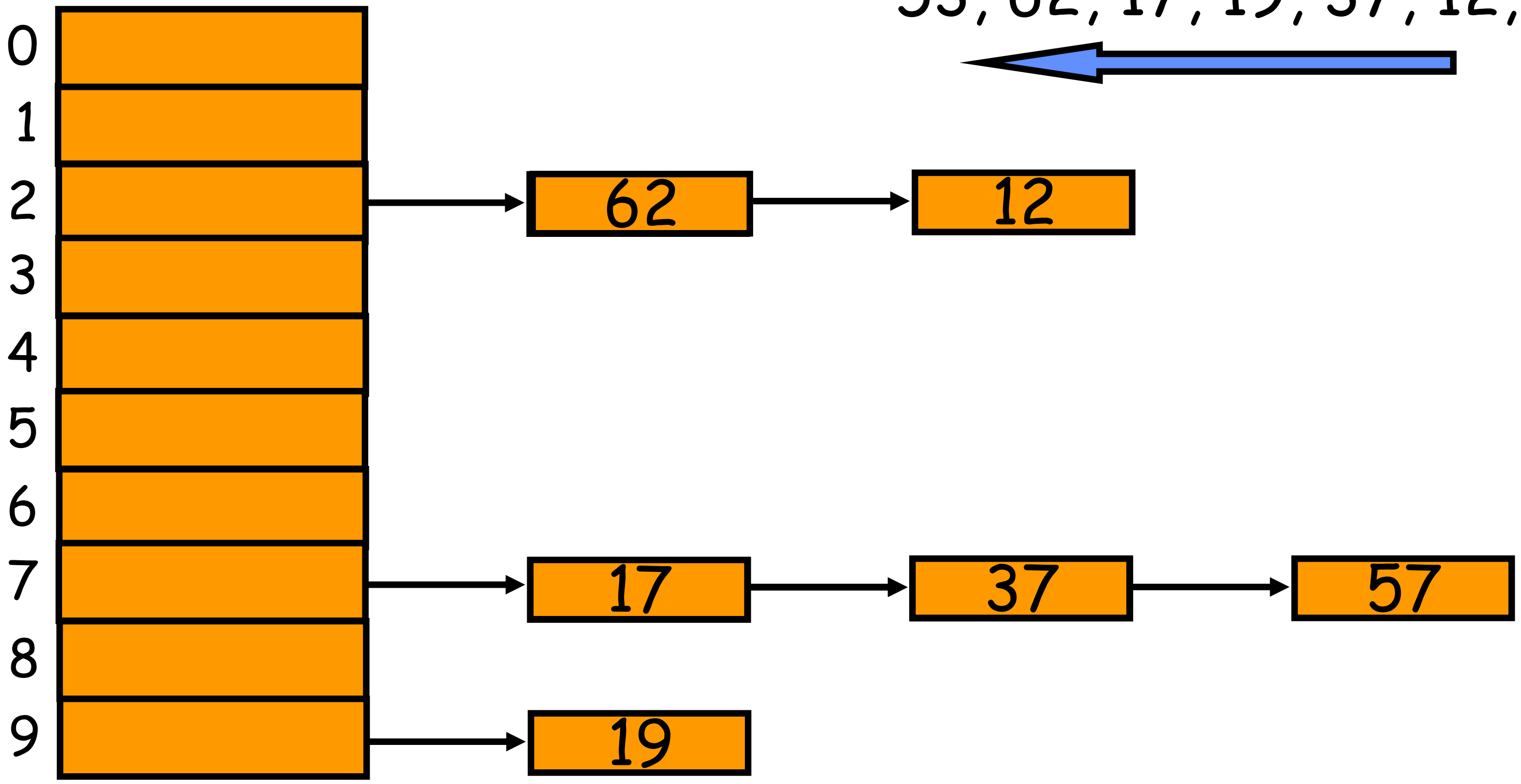
קלט: 53, 62, 17, 19, 37, 12, 57



דוגמא

נניח: h(k) = k mod m   m = 10

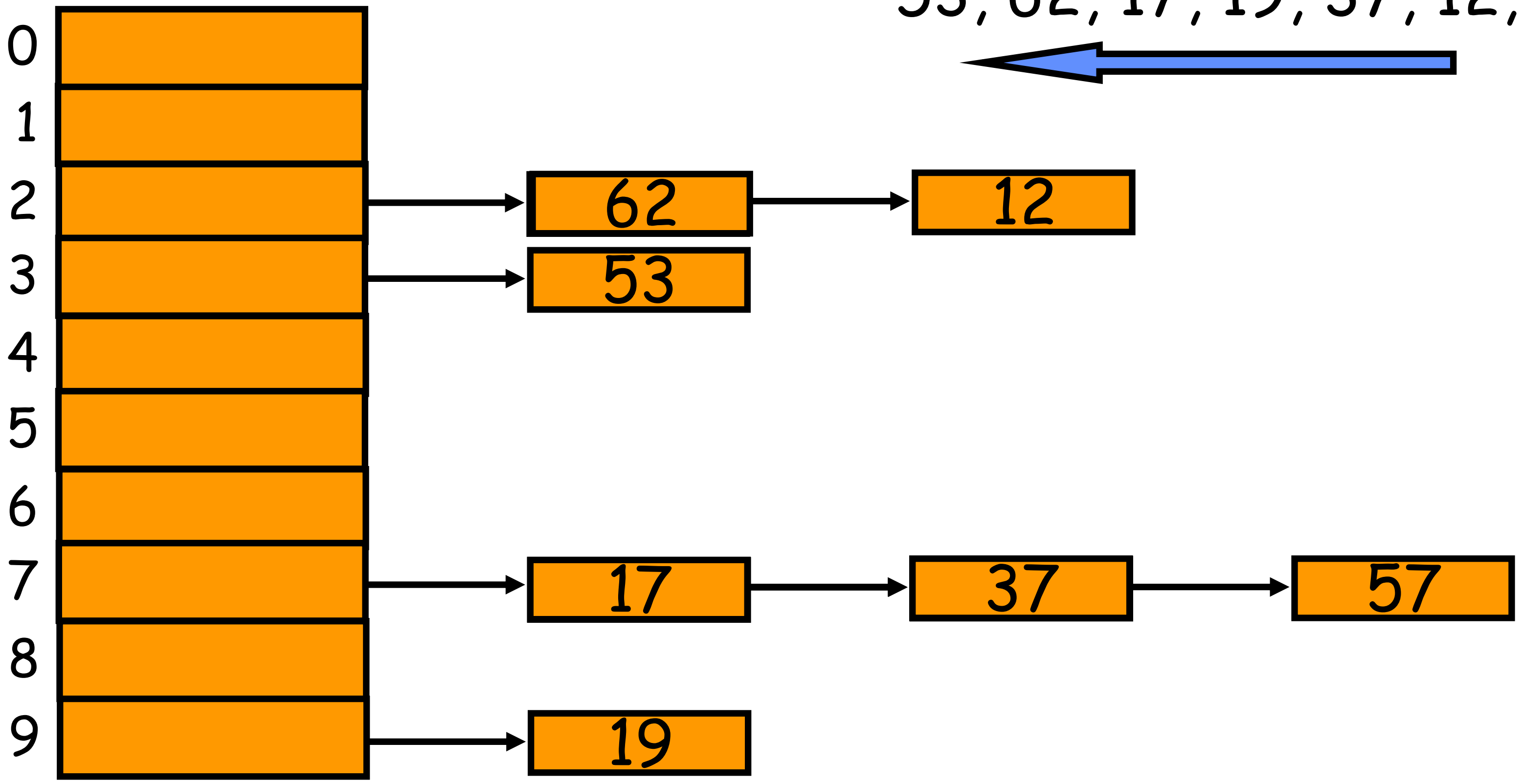
קלט: 53, 62, 17, 19, 37, 12, 57



דוגמא

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

קלט: 53, 62, 17, 19, 37, 12, 57

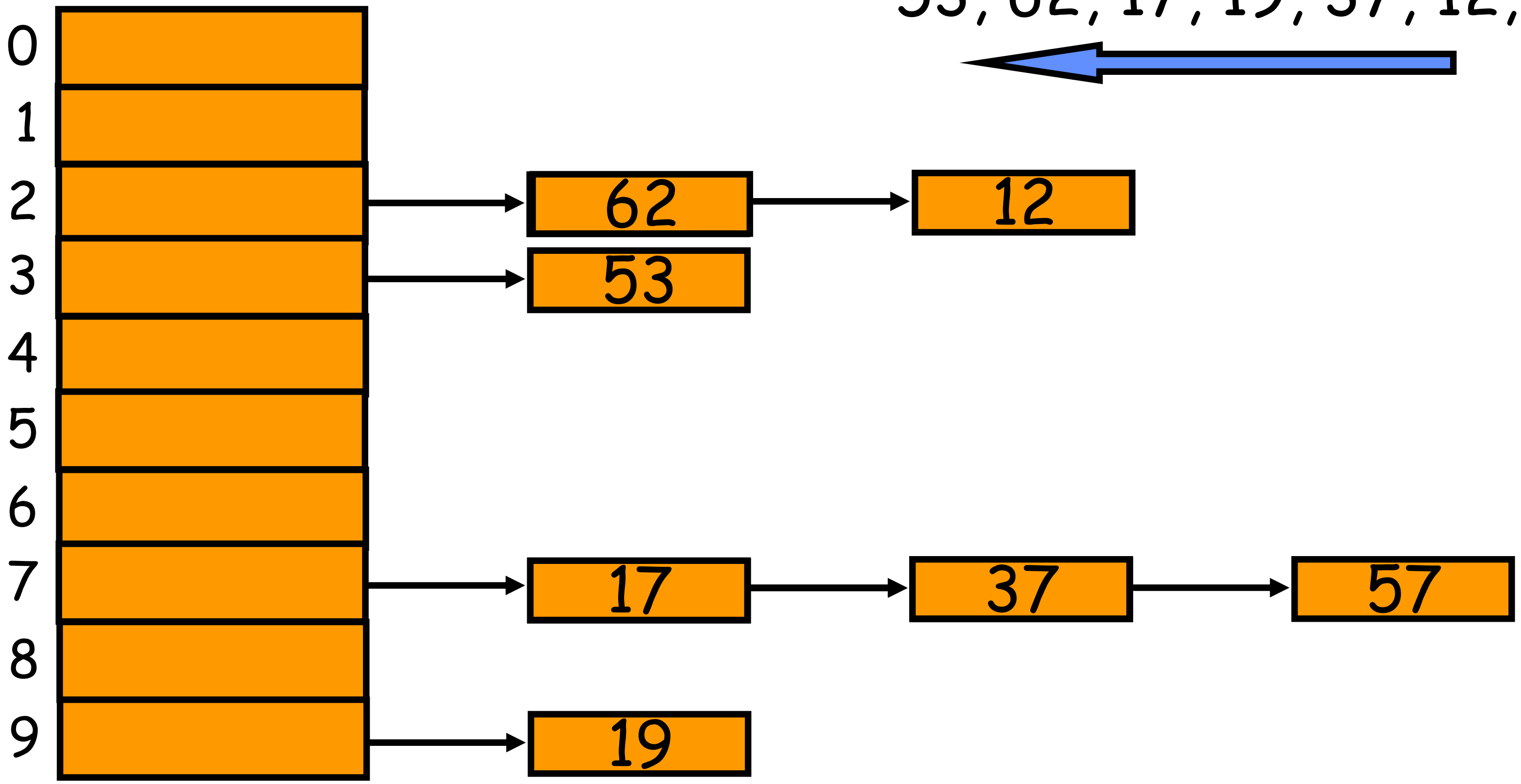


©cs,Technion

דוגמא

נניח: h(k) = k mod m   m = 10

קלט: 53, 62, 17, 19, 37, 12, 57



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

©cs,Technion

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.

©cs,Technion

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.

©cs,Technion

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 <u>הנחת הפיזור האחיד הפשוט (simple uniform hash).</u>

ננתח כעת את זמני החיפוש תחת הנחת הפיזור האחיד הפשוט.

<u>הגדרה</u>: יהי n מספר המפתחות בשימוש ויהי m גודל הטבלה. פקטור העומס מוגדר ע"י $\alpha = n / m$.

תחת הנחת הפיזור האחיד הפשוט האורך הממוצע של שרשרת הוא α מכיוון שהאיברים מתחלקים בצורה שווה בין השרשראות השונות.

©cs,Technion

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש כושל ממוצע הוא $\Theta(1+\alpha)$.

©cs,Technion

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש כושל ממוצע הוא $\Theta(1+\alpha)$.

הוכחה: בהנחת הפיזור האחיד הפשוט כל מפתח מגיע באקראי לאחת מ-m הרשימות.

- הזמן לחיפוש כושל הוא לפיכך הזמן הממוצע לחפש באחת הרשימות עד סופה.
- אורכה הממוצע של רשימה בהנחת הפיזור האחיד הוא $\alpha = n / m$.

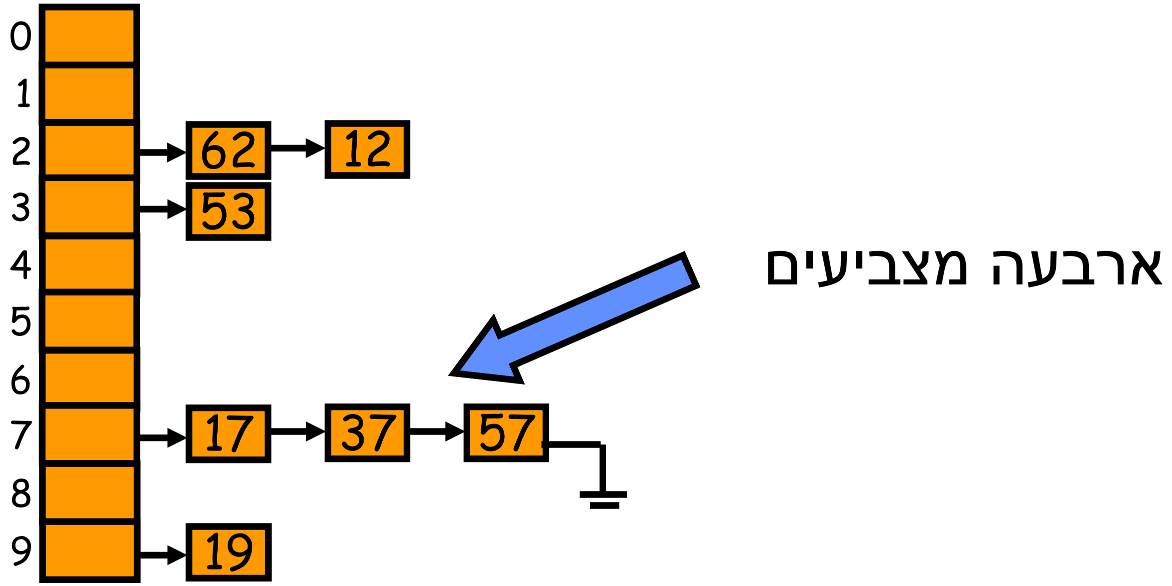
©cs,Technion

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש כושל ממוצע הוא $\Theta(1+\alpha)$.

הוכחה: בהנחת הפיזור האחיד הפשוט כל מפתח מגיע באקראי לאחת מ-m הרשימות.

- הזמן 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 (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.

ניתוח זמנים (המשך)

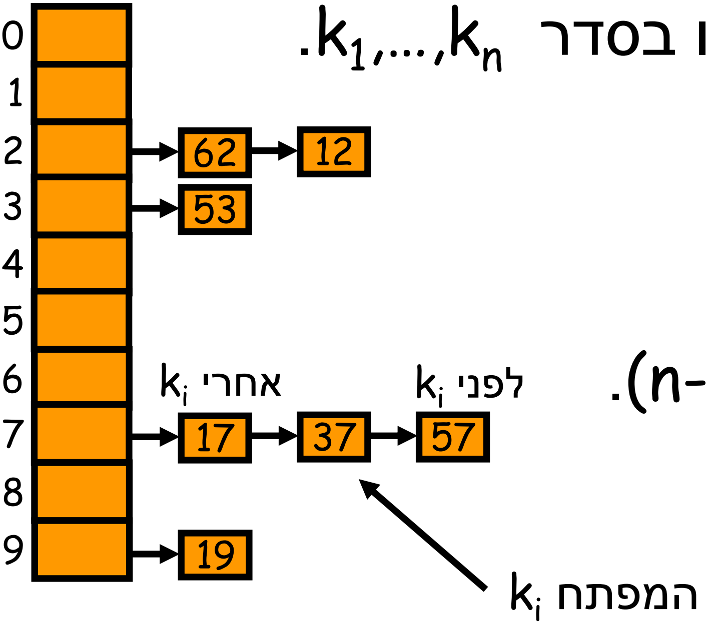
משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.

- הוכחה: נאמר שבזמן החיפוש ישנם n מפתחות שהוכנסו בסדר $k_1,\ldots,k_n$.
- 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 $(n-i)/m$.

ניתוח זמנים (המשך)

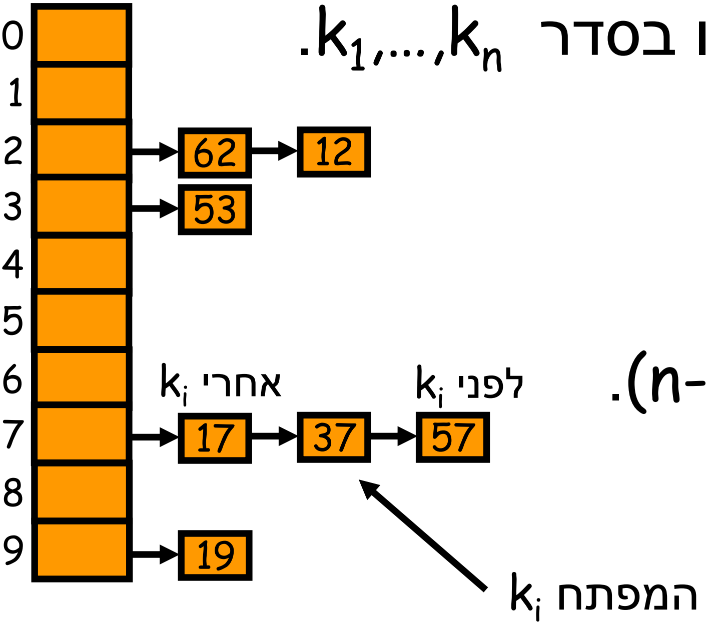
משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.

- הוכחה: נאמר שבזמן החיפוש ישנם n מפתחות שהוכנסו בסדר $k_1,\ldots,k_n$.
- 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 $(n-i)/m$.



ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.



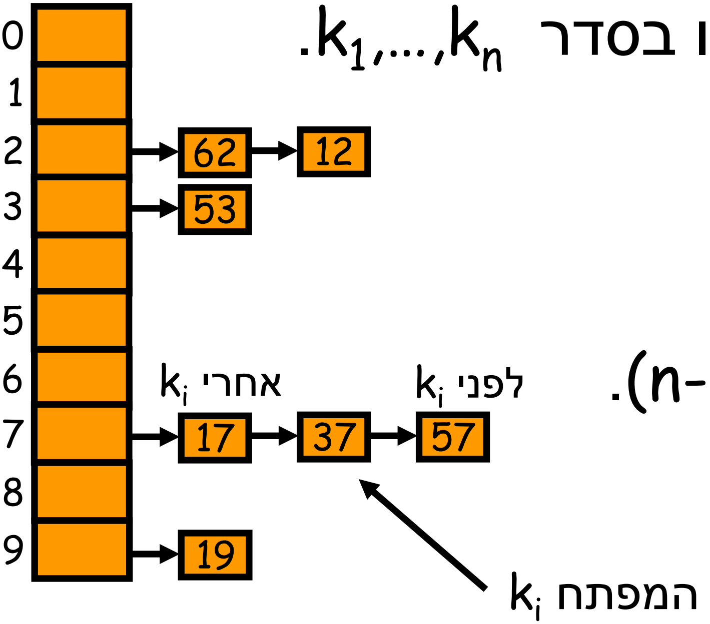
- הוכחה: נאמר שבזמן החיפוש ישנם n מפתחות שהוכנסו בסדר $k_1,\dots,k_n$.
- מהו זמן חיפוש הממוצע של המפתח k_i ?
- אחרי מפתח זה נוספו $n-i$ מפתחות נוספים.
- לכן בממוצע גודל הרשימה משמאל למפתח k_i הוא $(n-i)/m$.
- מכאן שזמן החיפוש הממוצע של המפתח k_i הוא

$$1+ (n-i)/m$$

- זמן החיפוש הממוצע t למפתח כלשהו יהיה לפיכך:

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע 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 למפתח כלשהו t יהיה לפיכך:

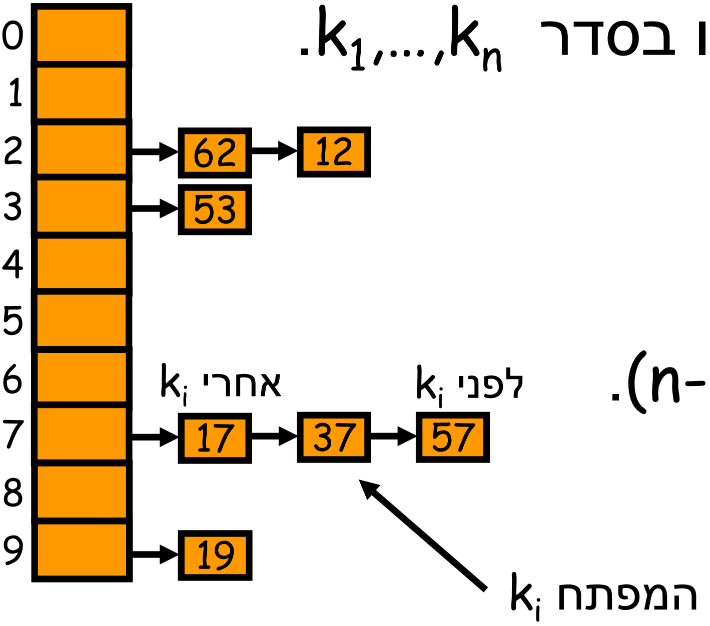
$$t = \frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{n-i}{m}\right)$$

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.

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

$$1+(n-i)/m$$

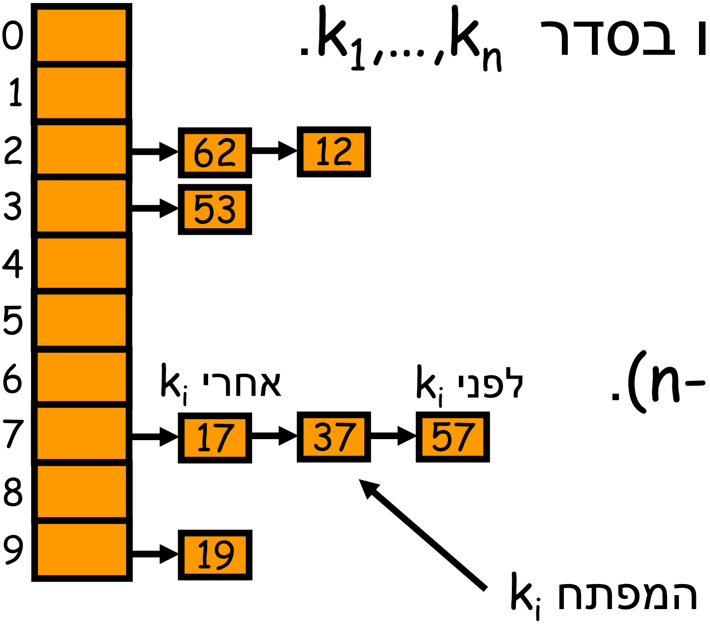


- זמן החיפוש הממוצע למפתח כלשהו t יהיה לפיכך:

$$t=\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{n-i}{m}\right) \quad =1+\frac{1}{n\cdot m}\sum_{i=1}^{n}(n-i)$$

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור הפשוט האחיד זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.



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

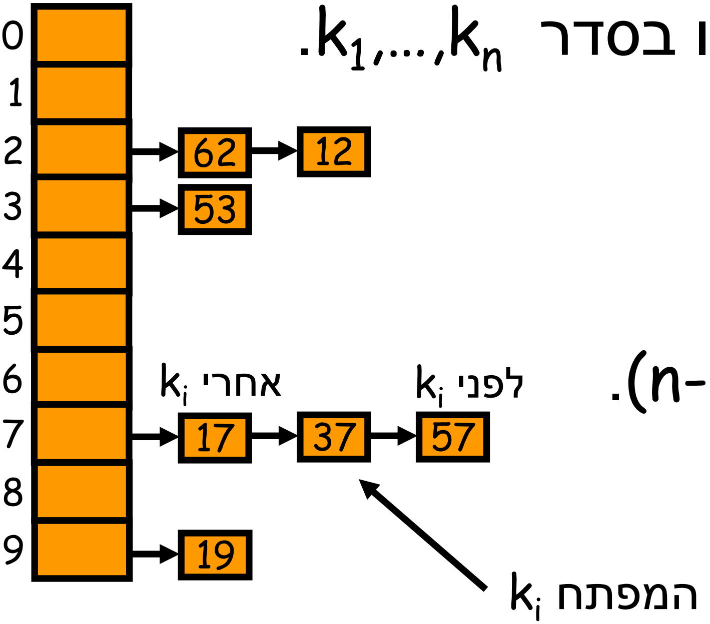
$$1+(n-i)/m$$

- זמן החיפוש הממוצע t למפתח כלשהו יהיה לפיכך:

$$t=\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{n-i}{m}\right) \quad =1+\frac{1}{n\cdot m}\sum_{i=1}^{n}(n-i) \quad =1+\frac{1}{n\cdot m}\sum_{i=0}^{n-1}i$$

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.



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

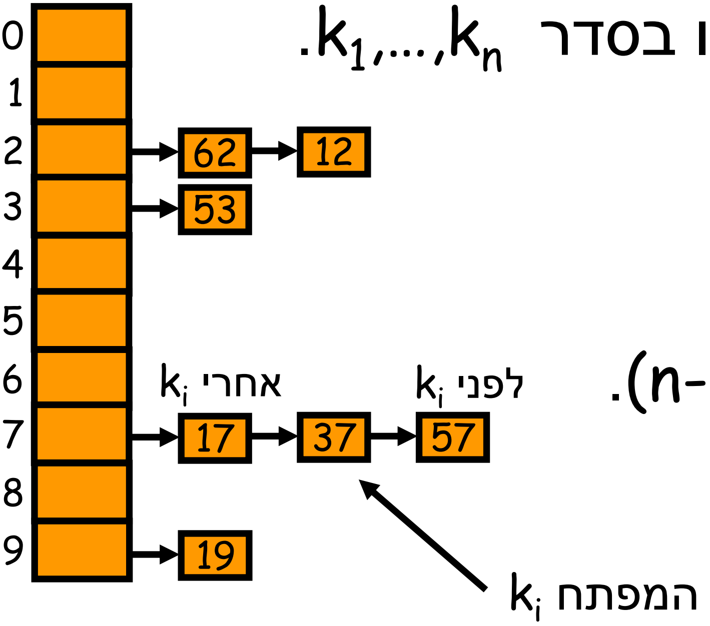
$$1+(n-i)/m$$

- זמן החיפוש הממוצע t למפתח כלשהו יהיה לפיכך:

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

ניתוח זמנים (המשך)

משפט: בשיטת השרשראות ותחת הנחת הפיזור האחיד הפשוט זמן חיפוש מוצלח ממוצע הוא $\Theta(1+\alpha/2)$.



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

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

$$=1+\frac{n-1}{2m}=1+\frac{\alpha}{2}-\frac{1}{2m}$$

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 $O(1)$ בממוצע.

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 וזמני החיפוש יהיו בהתאם.

פתרון ללא שרשראות להתנגשויות

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

פתרון ללא שרשראות להתנגשויות

לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה 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.

פתרון ללא שרשראות להתנגשויות

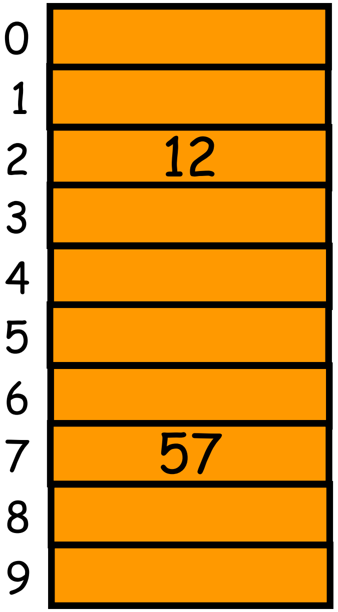
לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה כזו נקראת Open addressing. נבחן שלושה פתרונות להתנגשויות : סריקה ליניארית, ערבול נשנה, וערבול כפול.

סריקה ליניארית -- linear probing

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

דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

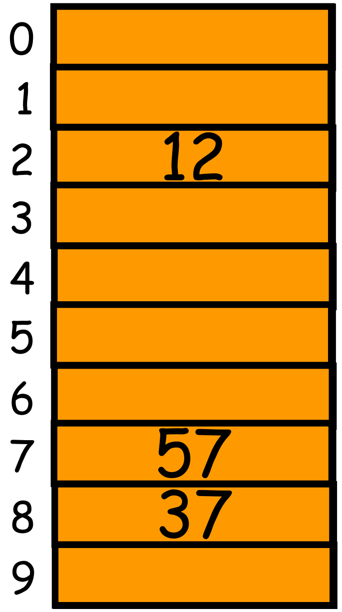


פתרון ללא שרשראות להתנגשויות

לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה 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.



דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

פתרון ללא שרשראות להתנגשויות

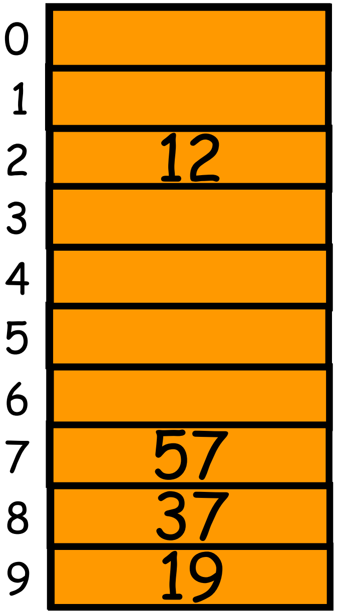
לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה כזו נקראת Open addressing. נבחן שלושה פתרונות להתנגשויות : סריקה ליניארית, ערבול נשנה, וערבול כפול.

סריקה ליניארית -- linear probing

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

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

קלט: 53, 62, 17, 19, 37, 12, 57

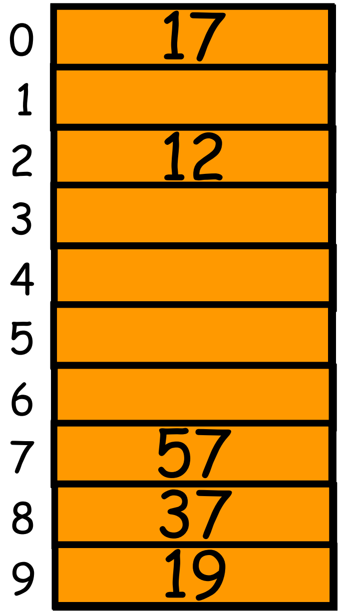


פתרון ללא שרשראות להתנגשויות

לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה 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.



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

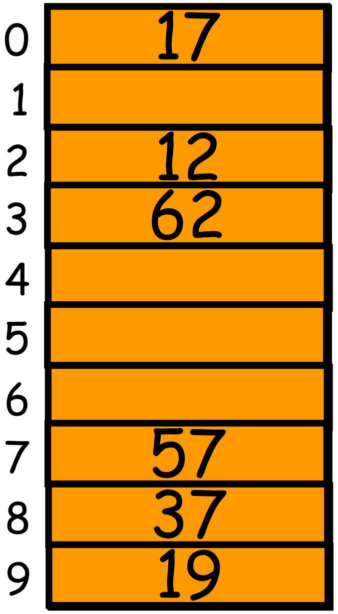
קלט: 53, 62, 17, 19, 37, 12, 57

פתרון ללא שרשראות להתנגשויות

לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה 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.



דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

פתרון ללא שרשראות להתנגשויות

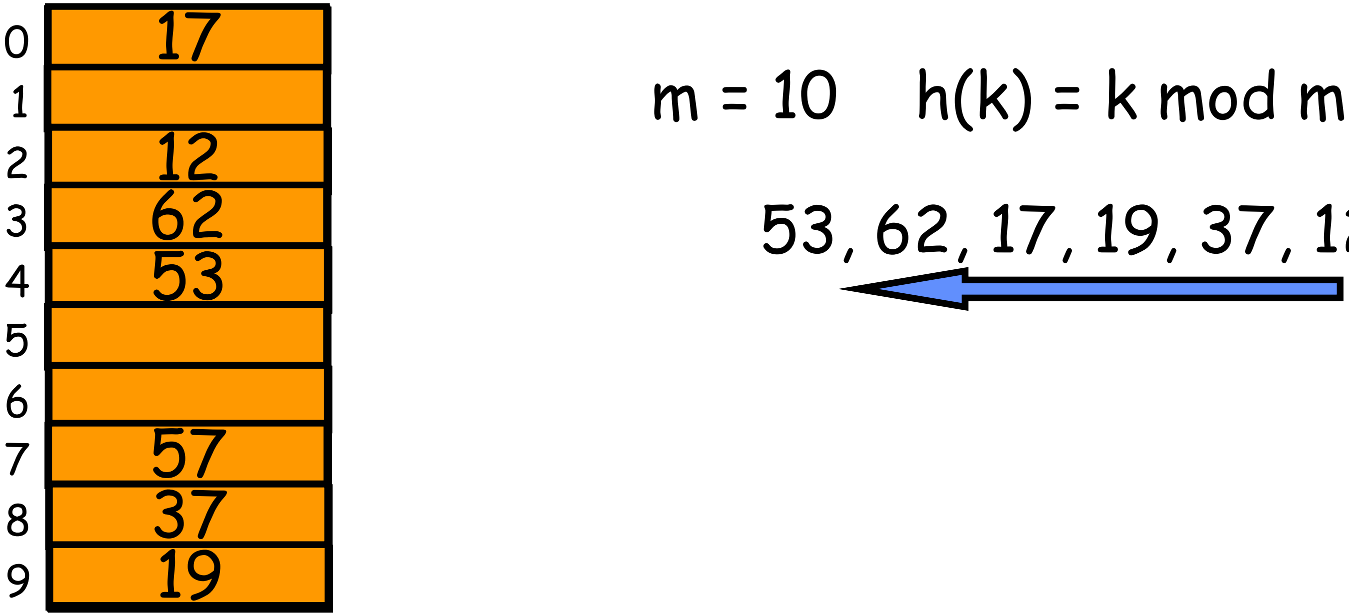
לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה כזו נקראת Open addressing. נבחן שלושה פתרונות להתנגשויות : סריקה ליניארית, ערבול נשנה, וערבול כפול.

סריקה ליניארית -- linear probing

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

דוגמא: h(k) = k mod m   m = 10

קלט: 57 ,12 ,37 ,19 ,17 ,62 ,53

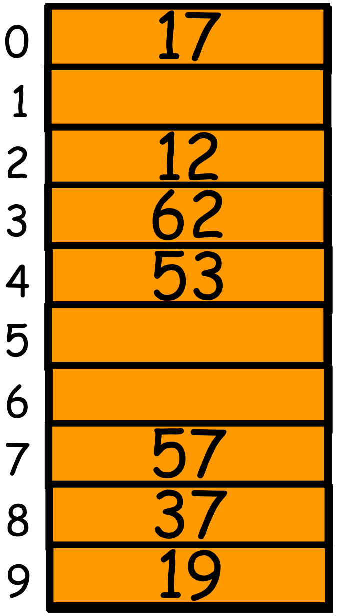


פתרון ללא שרשראות להתנגשויות

לא נשתמש בשרשראות, אלא כל האיברים יוכנסו לטבלה. שטה 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.



דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 57 ,12 ,37 ,19 ,17 ,62 ,53

ברור שבשיטות open addressing פקטור העומס קטן שווה אחד $(n \le m)$.

הוצאה בשיטת open addressing

כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

הוצאה בשיטת open addressing

כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

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

קלט: 57 ,12 ,37 ,19 ,17 ,62 ,53

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

הוצאה בשיטת open addressing

כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

הוצא 37,

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

הוצאה בשיטת open addressing

כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 57 ,12 ,37 ,19 ,17 ,62 ,53

הוצא 37,

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

הוצאה בשיטת open addressing

כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 53, 62, 17, 19, 37, 12, 57

הוצא 37, חפש 17,

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

הוצאה בשיטת open addressing

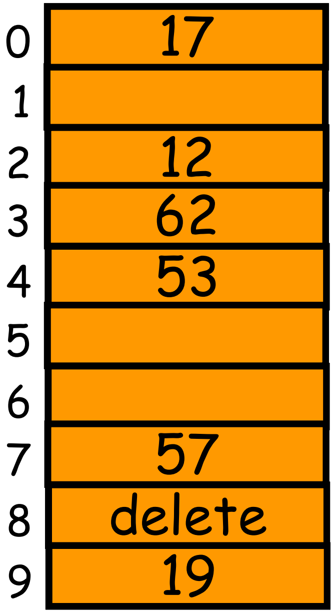
כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

דוגמא: h(k) = k mod m   m = 10

קלט: 53, 62, 17, 19, 37, 12, 57

הוצא 37, חפש 17, הכנס 27



הוצאה בשיטת open addressing

כיצד נוציא איברים ?

- לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק.
- נסמן את מקום האיבר שהוצא בסימן delete.
- בזמן חיפוש x, במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה 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).

דוגמא: $h(k) = k \bmod m$ $m = 10$

קלט: 57 ,12 ,37 ,19 ,17 ,62 ,53

הוצא 37, חפש 17, הכנס 27

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

יתרונות וחסרונות

- היתרון העיקרי של סריקה ליניארית הוא פשטות. אבל ...
- סריקה ליניארית נוטה למלא בלוקים מכיוון שכאשר קיים בלוק, האיברים הבאים מצטרפים אליו בסבירות גבוהה יותר מאשר הסבירות למלא תא חדש בודד. לפיכך, סריקה ליניארית אינה מהווה קרוב טוב להנחת הפיזור האחיד.
- כאשר השימוש דורש הוצאות, אורך החיפוש תלוי גם באיברים שכבר הוצאו ולא רק באיברים שכרגע במבנה.

©cs,Technion

יתרונות וחסרונות

- היתרון העיקרי של סריקה ליניארית הוא פשטות. אבל ...
- סריקה ליניארית נוטה למלא בלוקים מכיוון שכאשר קיים בלוק, האיברים הבאים מצטרפים אליו בסבירות גבוהה יותר מאשר הסבירות למלא תא חדש בודד. לפיכך, סריקה ליניארית אינה מהווה קרוב טוב להנחת הפיזור האחיד.
- כאשר השימוש דורש הוצאות, אורך החיפוש תלוי גם באיברים שכבר הוצאו ולא רק באיברים שכרגע במבנה.

<u>דוגמאות לשימוש במילון ללא הוצאות</u>:

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

©cs,Technion

יתרונות וחסרונות

• היתרון העיקרי של סריקה ליניארית הוא פשטות. אבל ...

• סריקה ליניארית נוטה למלא בלוקים מכיוון שכאשר קיים בלוק, האיברים הבאים מצטרפים אליו בסבירות גבוהה יותר מאשר הסבירות למלא תא חדש בודד. לפיכך, סריקה ליניארית אינה מהווה קרוב טוב להנחת הפיזור האחיד.

• 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 שמדמות טוב יותר את הנחת הפיזור האחיד הפשוט. שיטות אלו שימושיות במיוחד במימושי מילון ללא הוצאות. כאשר יש צורך בהוצאות, עדיפה שיטת הרשימות המקושרות.

©cs,Technion

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0, h_1, h_2, \ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

©cs,Technion

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0, h_1, h_2, \ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

©cs,Technion

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0,h_1,h_2,\ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0,h_1,h_2,\ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

ההסתברות שב- i הניסיונות הראשונים המקום תפוס היא α^i (בהנחה שפונקציות הערבול בלתי תלויות).

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0,h_1,h_2,\ldots$

נכסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

ההסתברות שב- i הניסיונות הראשונים המקום תפוס היא α^i (בהנחה שפונקציות הערבול בלתי תלויות).

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

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0,h_1,h_2,\ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

ההסתברות שב- i הניסיונות הראשונים המקום תפוס היא α^i (בהנחה שפונקציות הערבול בלתי תלויות).

ההסתברות שאורך החיפוש בדיוק i היא $(1- \alpha)\, \alpha^{i-1}$.

אורך החיפוש הממוצע בהנחת הפיזור האחיד הוא :

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0, h_1, h_2, \ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

ההסתברות שב- i הניסיונות הראשונים המקום תפוס היא α^i (בהנחה שפונקציות הערבול בלתי תלויות).

ההסתברות שאורך החיפוש בדיוק i היא $(1-\alpha)\,\alpha^{i-1}$.

אורך החיפוש הממוצע בהנחת הפיזור האחיד הוא :

$$l = \sum_{i=1}^{\infty} i\alpha(1-\alpha)^{i-1} = \frac{1}{1-\alpha}$$

Rehashing -- ערבול נשנה

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0, h_1, h_2, \ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

ההסתברות שב- i הניסיונות הראשונים המקום תפוס היא α^i (בהנחה שפונקציות הערבול בלתי תלויות).

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

אורך החיפוש הממוצע בהנחת הפיזור האחיד הוא :

$$l = \sum_{i=1}^{\infty} i\alpha(1-\alpha)^{i-1} = \frac{1}{1-\alpha}$$

לדוגמא עבור $\alpha = 2/3$ אורך החיפוש הממוצע הוא 3.

©cs,Technion

ערבול נשנה -- Rehashing

נניח שברשותנו סדרה אינסופית של פונקציות ערבול: $h_0, h_1, h_2, \ldots$

ננסה לשמור את x במקום $h_0(x)$.

אם תפוס, ננסה במקום $h_1(x)$. נמשיך עד שנצליח.

לדוגמא, בסריקה ליניארית מתקיים $h_i(x) = h(x) + i$.

<u>מהו זמן ההכנסה הממוצע</u> ?

תחת הנחת הפיזור האחיד הפשוט ההסתברות שמקום תפוס היא α.

ההסתברות שב- i הניסיונות הראשונים המקום תפוס היא α^i (בהנחה שפונקציות הערבול בלתי תלויות).

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

אורך החיפוש הממוצע בהנחת הפיזור האחיד הוא :

$$l = \sum_{i=1}^{\infty} i\alpha(1-\alpha)^{i-1} = \frac{1}{1-\alpha}$$

לדוגמא עבור $\alpha = 2/3$ אורך החיפוש הממוצע הוא 3.

הוצאות נעשות ע"י שימוש בסימון delete.

Double Hashing -- ערבול כפול

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

כאשר: $h_i(x) = h(x) + i\ d(x)$

הפונקציות d, h נבחרות באופן בלתי תלוי.

©cs,Technion

ערבול כפול -- Double Hashing

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

כאשר: $h_i(x) = h(x) + i\ d(x)$

הפונקציות d, h נבחרות באופן בלתי תלוי.

מהו היחס בין d(x) לגודל הטבלה m ?

©cs,Technion

Double Hashing -- ערבול כפול

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

כאשר: $h_i(x) = h(x) + i\ d(x)$

הפונקציות d, h נבחרות באופן בלתי תלוי.

מהו היחס בין d(x) לגודל הטבלה m ?

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

©cs,Technion

Double Hashing -- ערבול כפול

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

כאשר: $h_i(x) = h(x) + i\ d(x)$

הפונקציות d, h נבחרות באופן בלתי תלוי.

מהו היחס בין d(x) לגודל הטבלה m ?

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

הוצאות נעשות ע"י שימוש בסימון delete.

פונקציות ערבול

דרישות מפונקציות ערבול: מפזרת היטב וקלה לחישוב.

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.

©cs,Technion

פונקציות ערבול (המשך)

<u>שיטת הכפל עבור קבוע $0 < a < 1$</u>

- הכפל את המפתח k בקבוע a.
- מצא את החלק השבור של התוצאה.
- הכפל את החלק השבור ב- m ועגל כלפי מטה:

$$h(k) = \lfloor m \cdot (a \cdot k \bmod 1) \rfloor$$

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 :

$$a = \left(\sqrt{5} - 1\right)/2 = 0.61803...$$

פונקציות ערבול (המשך)

<u>שיטת הכפל עבור קבוע 0 < a < 1</u>

- 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 :

$$a = \left(\sqrt{5} - 1\right) / 2 = 0.61803\ldots$$

<u>דוגמא:</u> k = 123456 m = 10000

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

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

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

$$= \lfloor 41.151 \ldots \rfloor = 41$$

©cs,Technion

פונקציות ערבול למחרוזות ארוכות

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

"b" = 98 = 0110 0010

וכך הלאה ...

פתרון נאיבי: בצע xor ביט ביט.

לדוגמא: h("ab") = h((0110 0001) xor (0110 0010)) = (0000 0011) = 3

חסרון ראשון: h("aa") = h((0110 0001) xor (0110 0001)) = (0000 0000) = 0

h{"bb") = h((0110 0010) 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. $h(x) \leq 255$

©cs,Technion

פונקציות ערבול למחרוזות ארוכות (המשך)

פתרון עדיף: (על עקרון השפעת הפרפר מגינאה על מזג האוויר – שינוי קטן מביא שינוי גדול)

בחר פרמוטציה אקראית $(\pi_0,\ldots,\pi_{255})$ של 0...255 ואחסן אותה במערך T.

בצע xor בין האות הראשונה של המפתח s_0 והערך T[0].

התוצאה a_1 נמצאת בתחום 0...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 פונקצית הערבול היא תוצאת ה-xor עם האות האחרונה של המפתח כלומר

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

פונקציות ערבול למחרוזות ארוכות (המשך)

<u>פתרון עדיף</u>: (על עקרון השפעת הפרפר מגינאה על מזג האוויר – שינוי קטן מביא שינוי גדול)

בחר פרמוטציה אקראית $(\pi_0,\ldots,\pi_{255})$ של 0...255 ואחסן אותה במערך T.

בצע xor בין האות הראשונה של המפתח s_0 והערך T[0].

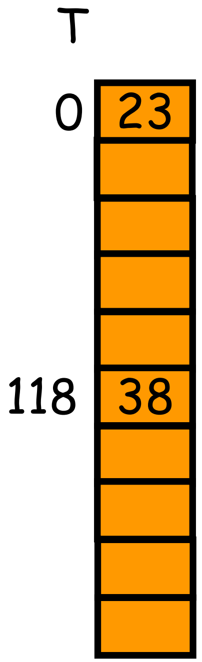
התוצאה a_1 נמצאת בתחום 0...255.

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

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

דוגמא: hash(aa)



hash(a) = T[0] xor 97 =

©cs,Technion

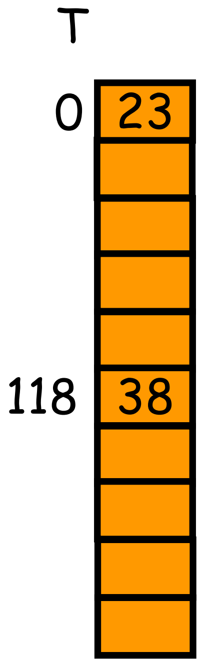
פונקציות ערבול למחרוזות ארוכות (המשך)

<u>פתרון עדיף</u>: (על עקרון השפעת הפרפר מגינאה על מזג האוויר - שינוי קטן מביא שינוי גדול)

בחר פרמוטציה אקראית $(\pi_0,\ldots,\pi_{255})$ של 0...255 ואחסן אותה במערך T.

בצע xor בין האות הראשונה של המפתח s_0 והערך T[0].

התוצאה a_1 נמצאת בתחום 0...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 פונקצית הערבול היא תוצאת ה-xor עם האות האחרונה של המפתח כלומר

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

דוגמא: hash(aa)

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

©cs,Technion

פונקציות ערבול למחרוזות ארוכות (המשך)

<u>פתרון עדיף</u>: (על עקרון השפעת הפרפר מגינאה על מזג האוויר – שינוי קטן מביא שינוי גדול)

בחר פרמוטציה אקראית $(\pi_0,\dots,\pi_{255})$ של 0...255 ואחסן אותה במערך T.

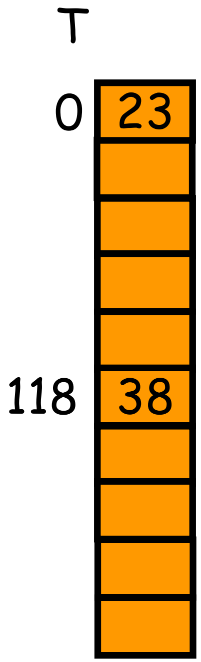
בצע xor בין האות הראשונה של המפתח s_0 והערך T[0].

התוצאה a_1 נמצאת בתחום 0...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 פונקצית הערבול היא תוצאת ה-xor עם האות האחרונה של המפתח כלומר $hash(key) = s_n \text{ xor } a_n$.

דוגמא: hash(aa)



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

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

©cs,Technion

פונקציות ערבול למחרוזות ארוכות (המשך)

פתרון עדיף: (על עקרון השפעת הפרפר מגינאה על מזג האוויר – שינוי קטן מביא שינוי גדול)

בחר פרמוטציה אקראית $(\pi_0,\ldots,\pi_{255})$ של 0…255 ואחסן אותה במערך T.

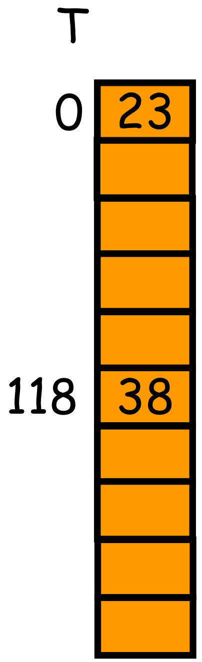
בצע xor בין האות הראשונה של המפתח s_0 והערך T[0].

התוצאה a_1 נמצאת בתחום 0…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 פונקצית הערבול היא תוצאת ה-xor עם האות האחרונה של המפתח כלומר

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



דוגמא: hash(aa)

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

hash(aa) = T[hash(a)] xor a = T[118] xor 97 = 0010 0110 xor 0110 0001 = 71

פונקציות ערבול למחרוזות ארוכות (המשך)

<u>פתרון עדיף</u>: (על עקרון השפעת הפרפר מגינאה על מזג האוויר – שינוי קטן מביא שינוי גדול)

בחר פרמוטציה אקראית $(\pi_0,\ldots,\pi_{255})$ של 0...255 ואחסן אותה במערך T.

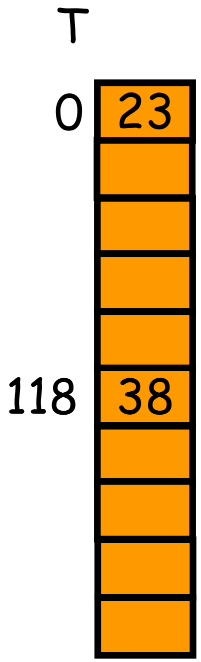
בצע xor בין האות הראשונה של המפתח s_0 והערך T[0].

התוצאה a_1 נמצאת בתחום 0...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 פונקצית הערבול היא תוצאת ה-xor עם האות האחרונה של המפתח כלומר

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



דוגמא: hash(aa)

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

hash(aa) = T[hash(a)] xor a = T[118] xor 97 = 0010 0110 xor 0110 0001 = 71

הערה: בשיטה זו נפתרה בעיית האותיות המופיעות מספר זוגי של פעמים.

©cs,Technion

פונקציות ערבול למחרוזות ארוכות (המשך)

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

$$hash(k) = [hash_1(k), hash_2(k)] = hash_1(k) * 256 + hash_2(k)$$

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 $hash_2$ נוסיף 1 לאות הראשונה של המחרוזת ונשתמש שוב בפונקצית הערבול $hash_1$.

לדוגמא: $hash(acb) = hash_1(acb) * 256 + hash_1(bcb)$

ערבול אוניברסלי

לכל בחירה של פונקצית ערבול קיימת סדרה גרועה של מפתחות כך שתווצר רשימה באורך מקסימלי.

תכונה זו יכולה ליצור בעיה.

לדוגמא, יתכן מתכנת המשתמש באופן עקבי בשמות מסוימים למשתני התוכניות 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,…,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 $1/m = (|H|/m)/|H| = p$.

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

ערבול אוניברסלי (המשך)

<u>משפט:</u> 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- $(n-1)/m$.

©cs,Technion

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 שההסתברות להתנגשות של מפתח מסוים x עם מפתח מסוים y היא $p = 1/m$.

©cs,Technion

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 שההסתברות 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 (בשימוש בעובדה שממוצע של סכום משתנים אקראיים שווה לסכום הממוצעים):

©cs,Technion

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 שההסתברות 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 (בשימוש בעובדה שממוצע של סכום משתנים אקראיים שווה לסכום הממוצעים):

$$\overline{L} = \sum_{\{y \in T | y \neq x\}} \frac{1}{m}$$

©cs,Technion

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 שההסתברות 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 (בשימוש בעובדה שממוצע של סכום משתנים אקראיים שווה לסכום הממוצעים):

$$\bar{L} = \sum_{\{y \in T | y \neq x\}} \frac{1}{m} = \frac{n-1}{m} = \alpha - \frac{1}{m}$$

©cs,Technion

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 שההסתברות 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 (בשימוש בעובדה שממוצע של סכום משתנים אקראיים שווה לסכום הממוצעים):

$$\overline{L} = \sum_{\{y \in T | y \neq x\}} \frac{1}{m} = \frac{n-1}{m} = \alpha - \frac{1}{m}$$

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 $x = [x_0,\ldots,x_r]$. (למשל באורך בייט = 8 ביטים).

מספר הביטים של x_i יהיה לכל היותר כמספר הביטים לייצוג גודל טבלה m.

עבור סדרה $a = [a_0,\ldots,a_r]$ מהתחום $\{0,..,m-1\}^{r+1}$ נגדיר פונקצית ערבול $h_a(x)$ בצורה הבאה:

©cs,Technion

בניית קבוצה אוניברסלית

נבחר את גודל הטבלה להיות מספר ראשוני m.

נשבור כל מפתח x ל- $r + 1$ חלקים באורך קבוע $x = [x_0,\ldots,x_r]$. (למשל באורך בייט = 8 ביטים).

מספר הביטים של x_i יהיה לכל היותר כמספר הביטים לייצוג גודל טבלה m.

עבור סדרה $a = [a_0,\ldots,a_r]$ מהתחום $\{0,..,m-1\}^{r+1}$ 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}.

<u>בניית קבוצה אוניברסלית</u>

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 של x_i יהיה לכל היותר כמספר הביטים לייצוג גודל טבלה m.

עבור 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}.

<u>דרך השימוש בשיטה זו</u>: כאשר משתמש מגדיר טבלת ערבול T בגודל m, התוכנית מגרילה מספר a ומשתמשת בפונקצית הערבול h_a לכל הפעולות הנעשות בטבלת ערבול זו.

©cs,Technion

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 של x_i יהיה לכל היותר כמספר הביטים לייצוג גודל טבלה m.

עבור סדרה $a = [a_0,\dots,a_r]$ מהתחום $\{0,..,m-1\}^{r+1}$ 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}.

<u>דרך השימוש בשיטה זו</u>: כאשר משתמש מגדיר טבלת ערבול T בגודל m, התוכנית מגרילה מספר a

ומשתמשת בפונקצית הערבול h_a לכל הפעולות הנעשות בטבלת ערבול זו.

<u>דוגמא</u>: $m=253$, טווח המפתחות $0\text{-}2^{24}$. נשבור כל מפתח לשלושה חלקים באורך 8 ביטים. נניח שהוגרלו המספרים $a=[248,223, 101]$. בהינתן המפתח $x=1025=[0,2,1]$ נחשב את מקומו בטבלת הערבול ע"י הנוסחה: $(248\cdot 0 + 223\cdot 2 + 101\cdot 1) \bmod 253 = 41$

©cs,Technion

בניית קבוצה אוניברסלית (המשך)

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

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

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

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

©cs,Technion

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 $x_0 \neq y_0$.

אנו טוענים שלכל 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:

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

הניתנת לשכתוב כדלקמן:

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

©cs,Technion

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 $x_0 \neq y_0$.

אנו טוענים שלכל 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:

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

הניתנת לשכתוב כדלקמן:

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

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.

בניית קבוצה אוניברסלית (המשך)

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

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

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

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

הניתנת לשכתוב כדלקמן:

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

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

נזכר שמספר הפונקציות ב- H הוא כמספר ערכי a כלומר m^{r+1} ולכן מתקיים $m^r = |H|/m$, כנדרש מקבוצה אוניברסלית. לפיכך ההסתברות ש- x ו- y יתנגשו היא $m^r/m^{r+1} = 1/m$.

©cs,Technion

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: עבור כל מספר z חיובי קיים מספר w יחיד

כך ש $z \cdot w = 1 \pmod{m}$

למשל $1 \cdot 1 = 1 \pmod{3}$ $2 \cdot 2 = 1 \pmod{3}$ $2 \cdot 3 = 1 \pmod{5}$

©cs,Technion

בניית קבוצה אוניברסלית (המשך)

<u>טענה</u>: 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: עבור כל מספר z חיובי קיים מספר w יחיד

כך ש $z \cdot w = 1 \pmod{m}$

למשל $1 \cdot 1 = 1 \pmod{3}$ $2 \cdot 2 = 1 \pmod{3}$ $2 \cdot 3 = 1 \pmod{5}$

במשוואה הנתונה $z = x_0 - y_0 \neq 0$

נכפיל את המשוואה בהופכי של z 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).

<u>פתרון חלקי</u>: כאשר טבלת ערבול מתמלא 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 $0\leq x_0,\ldots,x_n-1 < T$

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

©cs,Technion

שימוש: בעיית היחידות Element Uniqueness

נתונים מספרים $0 \le x_0, \ldots, x_n-1 < T$

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 $O(n)$.

©cs,Technion

Hash