טבלאות ערבול – Hash Tables

Lecture 7 of Geiger & Itai's slide brochure
www.cs.technion.ac.il/~dang/courseDS

חומר קריאה לשיעור זה

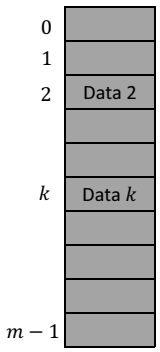
Chapter 12- Hash tables (pages 219—243)

מילון עם מפתחות שלמים (המשך)

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

תשובה:

- לעתים גודל הטווח של ערכי המפתחות גדול בהרבה *ממספר* המפתחות בהם משתמשים, מה שמניב פתרון בזבזני בזיכרון.



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.

מילון עם מפתחות שלמים

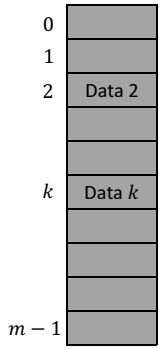
הפעולות הבסיסיות של מילון הן כזכור חיפוש, הכנסה, והוצאה. אם ידוע כי המפתחות מספרים שלמים בתחום $[0,\dots,m-1]$, ניתן לממש את שלושת הפעולות בזמן $O(1)$ במקרה הגרוע, באמצעות שימוש במערך בגודל m.

מימוש הפעולות:

Insert(k) כתיבת Data k בתא ה-k במערך. זמן: $O(1)$.

Delete(k) מחיקת המידע בתא ה-k במערך. זמן: $O(1)$.

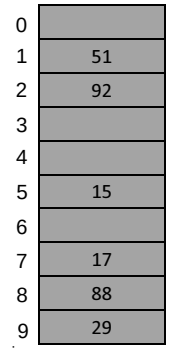
Find(k) החזר "קיים" אמ"מ התא ה-k לא ריק. זמן: $O(1)$.



ערבול (Hashing)

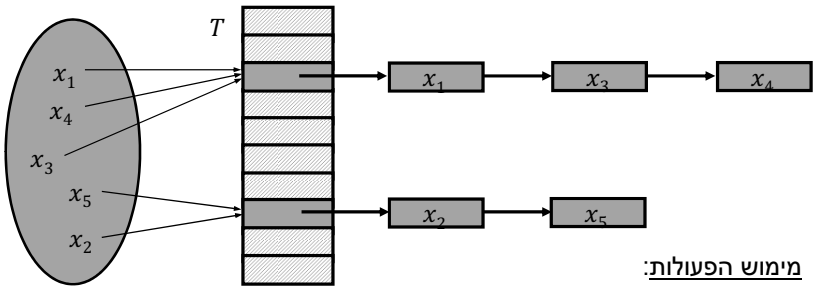
מימוש מילון באמצעות מערך נקרא גישה ישירה (Direct Addressing): המפתח עצמו משמש כאינדקס במערך.
כאשר מרחב המפתחות גדול נחשב את האינדקס מתוך המפתח באמצעות פונקצית ערבול. המטרה לממש את פעולות החיפוש, הכנסה, והוצאה בזמן ממוצע של $O(1)$.
נגדיר פונקצית ערבול (hash) $h: U \to \{0,\dots,m-1\}$, אשר בהינתן מפתח בתחום U מחשבת אינדקס בטווח המתאים. האינדקס של מפתח k יהיה $h(k)$.

דוגמא: $m = 10 \quad h(k) = k \bmod 10$
קלט להכנסה: $51, 17, 15, 92, 88, 29$



בשיטת הערבול נוצרות התנגשויות כאשר $x \neq y$ אבל $h(x) = h(y)$. לדוגמא, עבור הפונקציה h הנ"ל: $h(81) = 1 = h(51)$.

פתרון להתנגשויות באמצעות רשימות מקושרות



<u>מימוש הפעולות</u>:

Insert(T, x) הכנס את x בראש הרשימה $T[h(x.key)]$.
<u>זמן</u>: $O(1)$ במקרה הגרוע*.

Search(T, k) חפש איבר עם מפתח k ברשימה $T[h(k)]$.
<u>זמן</u>: (אורך הרשימה)Θ במקרה הגרוע.

Delete(T, x) סלק את x מהרשימה $T[h(x.key)]$.
<u>זמן</u>: (אורך הרשימה)Θ במקרה הגרוע.

* <u>הערה</u>: הכנסה ב-$O(1)$ אפשרית אם נרשה מופעים חוזרים של אותו מפתח. אחרת, בהכנסה נבצע קודם חיפוש, ולכן גם פעולה זו תיקח (אורך הרשימה)Θ זמן במקרה הגרוע.

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. פורמלית:

<u>הגדרה</u>: נאמר כי פונקציית ערבול h מקיימת את <u>הנחת הפיזור האחיד הפשוט (simple uniform hash)</u> אם h ממפה איבר אקראי בהסתברות אחידה $(1/m)$ לכל אחד מהתאים, ובאופן בלתי תלוי בשאר האיברים.

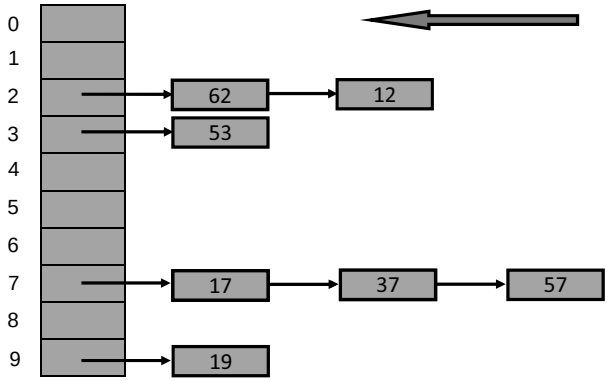
ננתח כעת את זמני החיפוש תחת הנחת הפיזור האחיד הפשוט.

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

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

דוגמא

<u>נניח</u>: $m = 10 \quad h(k) = k \bmod m$
<u>קלט</u>: $53, 62, 17, 19, 37, 12, 57$



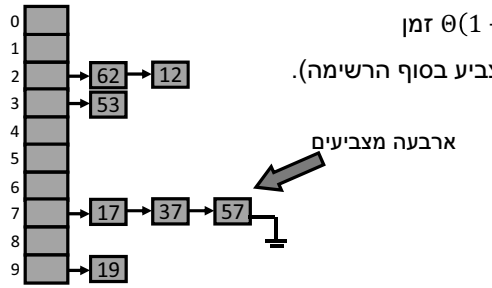
<u>הערה</u>: $m = 10$ נבחר לצורכי נוחיות ההסבר. נראה בהמשך שבחירה טובה יותר היא 11.

ניתוח זמנים (המשך)

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

<u>הוכחה</u>:

- 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 $\Theta(1 + \alpha)$ זמן (הכולל את זמן בדיקת המצביע בסוף הרשימה).



ניתוח זמנים (המשך)

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

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

<u>מסקנה:</u> 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 בשיטת השרשראות, תחת הנחת הפיזור האחיד הפשוט, כל הפעולות דורשות זמן $O(1)$ בממוצע.

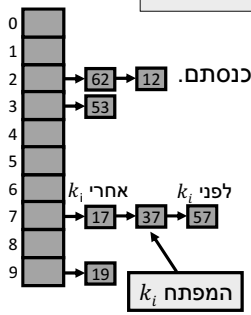
<u>לדוגמא:</u> עבור 2100 מפתחות מטווח כלשהו U של מספרים שלמים, נאמר עד 10^6, נוכל להחזיק מערך ובו 700 מקומות ובממוצע אורך כל שרשרת יהיה 3 וזמני החיפוש יהיו בהתאם.

ניתוח זמנים (המשך)

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

<u>הוכחה:</u>

- יהיו $k_1,\dots,k_n$ המפתחות בטבלה בזמן החיפוש, לפי סדר הכנסתם.
- 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 $\frac{n-i}{m}$.
- מכאן שזמן החיפוש הממוצע של המפתח k_i הוא $\frac{n-i}{m}+1$.



- <u>זמן החיפוש הממוצע t למפתח כלשהו יהיה לפיכך:</u>

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

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

פתרון ללא שרשראות להתנגשויות
Open Addressing

בשיטת Open addressing לא נשתמש בשרשראות. במקום זאת, כל האיברים יוכנסו לטבלה. ברור כי בשיטה זו פקטור העומס קטן\שווה אחד ($n\le m$).

כיוון שלא נפתור את בעיית ההתנגשויות באמצעות רשימות, נצטרך להציע פתרון אחר להתנגשויות.

<u>נבחן שלושה פתרונות להתנגשויות:</u>

1. סריקה ליניארית.
2. ערבול נשנה.
3. ערבול כפול.

סריקה לינארית – Linear Probing

בסריקה ליניארית, בעת הכנסה, אם המקום המיועד $h(k)$ תפוס, נסה במקום הבא מודולו m (וכך הלאה).

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

<u>קלט:</u> $53, 62, 17, 19, 37, 12, 57$

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

חיפוש מתבצע באותה שיטה.

הוצאה בשיטת Open Addressing

כיצד נוציא איבר?
לא ניתן פשוט למחוק איבר שכן שרשרת החיפוש תינתק עבור איברים אחרים.

- <u>פתרון:</u> נסמן את מקום האיבר שהוצא בסימן delete.
- <u>בזמן חיפוש x:</u> במידה וניתקל בסימן delete, נמשיך את סריקת הרשימה עד למציאת x או עד להגעה למקום ריק (המסומן ב- Null).
- <u>בזמן הכנסת x:</u> במידה וניתקל בסימן delete, נשתמש במקום זה לשמירת x.

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

<u>קלט:</u> $53, 62, 17, 19, 37, 12, 57$

הוצא 37, חפש 17, הכנס 27

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

סריקה ליניארית - 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.

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.

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.
<u>לדוגמא:</u> 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).
- לכן ההסתברות שאורך החיפוש בדיוק i היא $\alpha^{i-1} \cdot (1-\alpha)$.
- מכאן שאורך החיפוש הממוצע, תחת הנחת הפיזור האחיד הוא:

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

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

ערבול כפול - Double Hashing

נגיע לתוצאות דומות לערבול נשנה ע"י שתי פונקציות בלבד d, h.
<u>כאשר:</u> $h_i(x) = h(x) + i \cdot d(x)$
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 $h_0(x), \ldots, h_{m-1}(x)$ יכסו את כל האינדקסים האפשריים בתחום $\{0, \ldots, m-1\}$. לפיכך נוח לבחור את m להיות מספר ראשוני.

<u>הערה:</u> בשלושת הדוגמאות שראינו לערבול בשיטת Open Addressing (סריקה ליניארית, ערבול נשנה וערבול כפול), ההוצאות נעשות ע"י שימוש בסימון delete.

פונקציות ערבול

דרישות מפונקציות ערבול: מפזרת היטב וקלה לחישוב.

בשקפים הבאים נציג מספר פונקציות ערבול קלות לחישוב, יחד עם דרישות מהן, כדי שידמו היטב את הנחת הפיזור האחיד הפשוט.

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

פונקציות ערבול (המשך)

שיטת החילוק מודולו m:

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

עבור פונקצית ערבול זו, 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 $2^8 = 256$.

פונקציות ערבול (המשך)

שיטת הכפל עבור קבוע $0 < a < 1$

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

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

הערך של m אינו קריטי.
ערך של a הגורם לפיזור טוב הוא :

$$\bar{\phi} = (\sqrt{5} - 1)/2 = 0.61803 \ldots$$

דוגמא: $m = 10000,\ k = 123456\ (a = \bar{\phi})$.

$$\begin{aligned} h(k) &= \lfloor 10000 \cdot (123456 \cdot 0.61803 \ mod\ 1) \rfloor \\ &= \lfloor 10000 \cdot (76300.0041151 \ mod\ 1) \rfloor \\ &= \lfloor 10000 \cdot \ 0.0041151 \rfloor \\ &= \lfloor 41.151 \ldots \rfloor \ = \ 41 \end{aligned}$$

פונקציות ערבול למחרוזות ארוכות

כדי לדון במחרוזת כמספרים, 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.
למשל: $h("abccba") = 0$

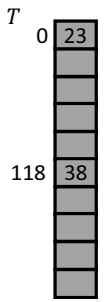
חסרון שני: טווח הערכים מוגבל: $h(x) \leq 255$.

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.

פונקצית הערבול: עבור המפתח $key = s = s_1 s_2 \dots s_n$.

1. $a_1 \leftarrow s_1 \; xor \; T[0]$
2. בשלב ה- i :
   $a_i \leftarrow s_i \; xor \; T[a_{i-1}]$
3. תוצאת פונקצית הערבול:
   $hash(key) = s_n \; xor \; a_{n-1}$



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.

פונקציות ערבול למחרוזות ארוכות (המשך)

מימוש של פונקצית הערבול מהשקף הקודם בשפת C/C++:

```
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)

פתרון לבעית הטווח של הפונקציה מהעמוד הקודם:

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 $256^2 = 2^{16}$. (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 שתיווצר 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.

בכדי שגישה זו תועיל לנו, נרצה שקבוצת הפונקציות תהיה כזו, שעבור כל סדרת מפתחות, בחירה אקראית של אחת הפונקציות תיצור פיזור טוב (הגדרה בעמוד הבא).

ערבול אוניברסלי (המשך)

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

הבחנה: ההסתברות p שבבחירה אקראית של פונקצית ערבול מתוך H, מפתח x יתנגש עם מפתח אחר y היא

$$p=\left(\frac{|H|}{m}\right)/|H|=1/m$$

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

בניית קבוצה אוניברסלית

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

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

קבוצת הפונקציות H היא $\bigcup_a\{h_a\}$ ומספר הפונקציות בה 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: $m=503$, טווח המפתחות $0,1,\dots,(2^{24}-1)$. נשבור כל מפתח לשלושה חלקים באורך 8 ביטים. נניח שהוגרלו המספרים $a=[248,223,101]$. בהינתן המפתח $x=1025=[0,4,1]$ נחשב את מקומו בטבלת הערבול ע"י הנוסחה:
$(248\cdot 0+223\cdot 4+101\cdot 1)\ mod\ 503=490$

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- $\frac{n-1}{m}$.

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 (תוך שימוש בעובדה שממוצע של סכום משתנים אקראיים שווה לסכום הממוצעים):

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

מסקנה: מספר ההתנגשויות הצפוי לכל מפתח קטן מגורם העומס.

בניית קבוצה אוניברסלית (המשך)

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

הוכחה: יהיו $x=[x_0,\dots,x_r]$ ו- $y=[y_0,\dots,y_r]$ מפתחות שונים. בלי הגבלת הכלליות נניח כי $x_0\neq y_0$. בפרט, $x_0\not\equiv y_0\ \ (mod\ m)$.
טענה: לכל ערך קבוע של $a_1,\dots,a_r$ קיים a_0 יחיד כך שמתקיים $h_a(x)=h_a(y)$.
הערך a_0 מתקבל מהפתרון היחיד למשוואה:

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

הניתנת לשכתוב כדלקמן:

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

נוכיח את הטענה בשקף הבא. בהנחה שהטענה נכונה, נובע שכל זוג מפתחות x,y מתנגשים עבור m^r ערכים של a וזאת כיוון שלכל ערך של $(a_1,\dots,a_r)$ קיים ערך אחד a_0 עבורו x,y מתנגשים.
נזכר שמספר הפונקציות ב- H הוא כמספר ערכי a, כלומר $|H|=m^{r+1}$.
לכן מתקיים כי מספר הפונקציות עבורן $x\neq y$ מתנגשות הוא $m^r=|H|/m$, כנדרש מקבוצה אוניברסלית.

בניית קבוצה אוניברסלית (המשך)

טענה: למשוואה הבאה כפונקציה של a_0 יש פתרון והפתרון יחיד.

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

הוכחת הטענה:

- נזכר שכאשר m ראשוני מתקיים: לכל מספר $z \not\equiv 0 \pmod m$ קיים מספר w יחיד כך ש- $z \cdot w \equiv 1 \pmod m$. המספר w נקרא ההוכפי של z.
  למשל: $1 \cdot 1 \equiv 1 \pmod 3$ $2 \cdot 2 \equiv 1 \pmod 3$ $2 \cdot 3 \equiv 1 \pmod 5$
- במשוואה הנתונה מתקיים, $z = x_0 - y_0 \not\equiv 0$
- נכפיל את המשוואה בהופכי של z ונקבל את הפתרון היחיד ל- a_0:

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

- יחידות הפתרון: יהיו a ו-b פתרונות למשוואה זו. אזי $a(x_0 - y_0) \equiv b(x_0 - y_0)$.
  נכפול בהופכי של $z = (x_0 - y_0)$ ונקבל כי $a \equiv b \pmod m$.

שימוש: בעיית היחידות
Element Uniqueness

קלט: n מספרים שלמים $0 \le x_1 \le \cdots \le x_n < T$.

הבעיה: מצא האם קיימים $i \neq j$ עבורם $x_i = x_j$.

פתרון ראשון – מיון:

אחרי המיון נבדוק האם קיימים איברים סמוכים שווים.

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: $O(n)$ בממוצע.

מגבלות לערבול

בעיה: בכדי להבטיח כי $n = O(m)$ ולכן פקטור העומס הוא $\alpha = \frac{n}{m} = O(1)$, צריך לדעת מראש סדר גודל למספר האיברים שמתעתדים להכניס למבנה (n).

פתרון חלקי: 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.

עוד על פתרון זה בתרגולים.

ניתוח זמנים עבור open addressing

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

משפט: בהנחת הפיזור האחיד, בשיטת ערבול open addressing מתקיים:

- זמן ממוצע של חיפוש כושל קטן מ- $1/(1-\alpha)$.
- זמן ממוצע של חיפוש מוצלח קטן מ- $\frac{1}{\alpha}\ln\frac{1}{\alpha} + \frac{1}{\alpha}$.

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

הערה: המשפט תאורטי שכן קשה לקיים את הנחת הפיזור האחיד.

ניתוח זמנים עבור סריקה לינארית

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

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

הוכחה בספר: Knuth, "The Art of Computer Programming", Vol 3, 1973

הערה: המשפט מראה שסריקה ליניארית אפשרית לשימוש. את הנחת הפיזור האחיד הפשוט קל יותר לקרב מאשר את הנחת הפיזור האחיד.

סיכום השיעור

בשיעור זה ראינו:

1. טבלת ערבול – מימוש נוסף של מילון, בעל זמן ריצה $O(1)$ בממוצע ל-3 פעולות המילון.
2. מימושים שונים של טבלת ערבול – כולל אופן הטיפול בהתנגשויות של פונקצית הערבול:
   1) פתרון באמצעות רשימות מקושרות
   2) מספר שיטות Open Addressing:
      - סריקה לינארית
      - ערבול נשנה
      - ערבול כפול.
3. פונקציות ערבול שונות – ביניהן דוגמה לקבוצה אוניברסלית.