הוכן ע"י אמיר רובינשטיין

מבני נתונים ומבוא לאלגוריתמים

נושא 11

טבלאות גיבוב
Hash tables

בתוכנית

פרק 11 בספר הלימוד

- נכיר עוד מימוש למילון - טבלאות גיבוב* (hash tables).
- זהו מימוש יעיל בממוצע - כל הפעולות ירוצו בזמן קבוע

* נקראות גם לעיתים **טבלאות ערבול**

מוטיבציה

ראינו מימוש למילון באמצעות עץ חיפוש מאוזן, שבו כל הפעולות רצות בזמן $O(\log n)$, כאשר n הוא מספר האיברים במבנה.

האם אפשר לממש מילון בסיבוכיות זמן טובה יותר?

התשובה היא כן!

ניתן לממש מילון במערך (זאת בהנחה שניתן למפות את תחום המפתחות לאינדקסים של המערך).

כל הפעולות ירוצו בזמן קבוע.

מדוע אם כן בכלל משתמשים בעצי חיפוש?

כי גודל התחום עלול להיות גדול מאוד.

דוגמה 1: מילון בו המפתח הוא מספר ת"ז בן 9 ספרות עשרוניות.

מערך יכיל 10^9 תאים, בעוד שבישראל פחות מ- 10^7 תושבים (ניצול של פחות מ- 1% של המערך).

דוגמה 2: מילון אנגלי. כמות הערכים בו קטנה מאוד יחסית לכמות התמורות של a-z באורך עד 35.

מוטיבציה

נגדיר כעת טבלאות גיבוב (hash tables)

ופונקציות גיבוב (hash functions)

זוהי מעין הכללה של הרעיון של מערך.

במקום לשמור תא לכל איבר פוטנציאלי, נמפה את עולם האיברים לטבלה קטנה יחסית, ונחשב לכל מפתח את האינדקס שלו בטבלה.

כפי שנראה, ניתן לדאוג שכל הפעולות ירוצו בזמן $\Theta(1)$ בממוצע.

טבלאות גיבוב - הגדרה

נתון עולם של איברים U.

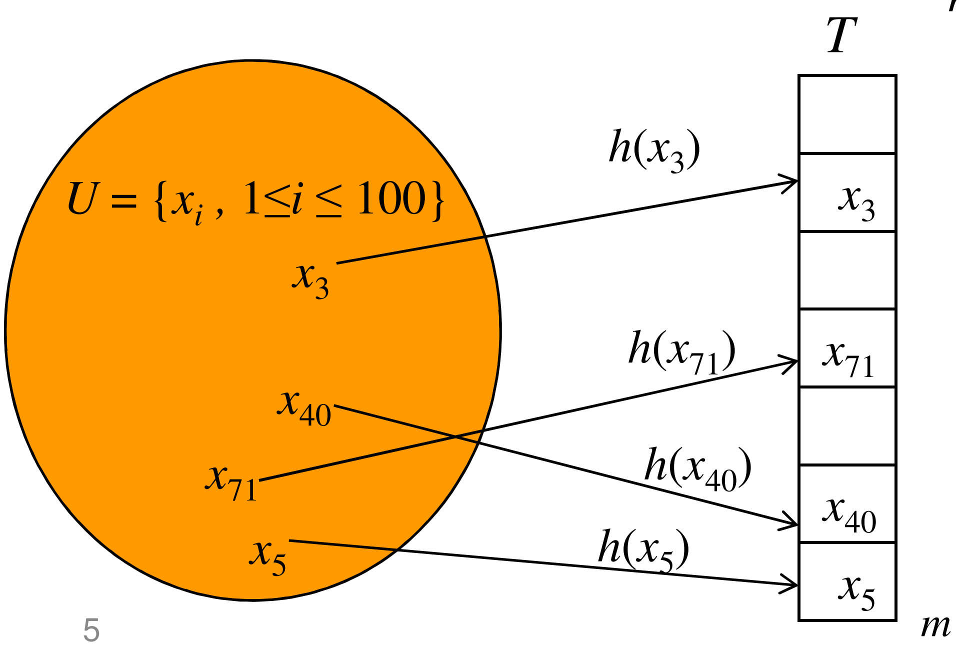
דרוש מימוש למילון, שבו יהיו בכל רגע נתון $O(n)$ איברים, כאשר $n = o(|U|)$.

נקצה מערך T שייקרא **טבלת גיבוב**.

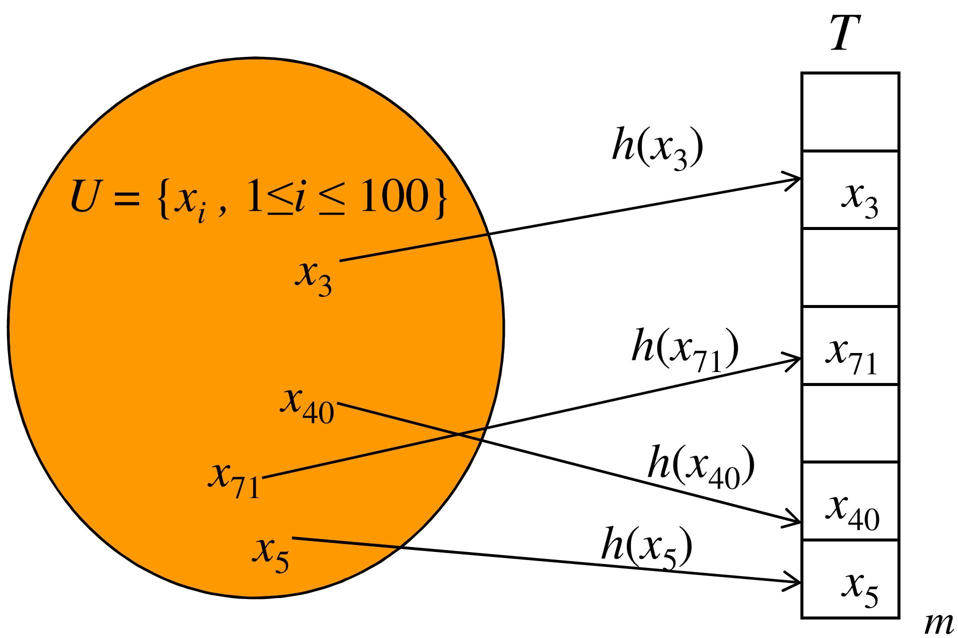
גודלו יסומן ב- m. הרעיון הוא ש- m יכול להיות קרוב ל- n, ולא ל- $|U|$.

נגדיר **פונקצית גיבוב**: $h: U \rightarrow \{0, 1, \ldots, m\text{-}1\}$

איבר $x \in U$ ימופה ל- $T[h(x)]$



בעיות



<u>בעיות</u>

1) מהי פונקציית גיבוב טובה?

2) מה עושים במקרה של התנגשות?

התנגשות: עבור $x \neq y$ מתקיים $h(x) = h(y)$.

מה בהמשך?

1. <u>מהי פונקצית גיבוב טובה?</u>

נגדיר זאת, ונראה כמה שיטות לבחירת פונקצית גיבוב טובה.

2. <u>מה עושים במקרה של התנגשות?</u>

נכיר שתי גישות:

1. שיטת השרשור
2. שיטת המיעון הפתוח, עם 3 אפשרויות:
   - א. בדיקה ליניארית
   - ב. בדיקה ריבועית
   - ג. גיבוב כפול

בחירת פונקצית גיבוב טובה

אילו תכונות נרצה שתקיים פונקצית הגיבוב?

1) זמן חישוב הפונקציה $O(1)$.
2) הפונקציה "מפזרת היטב" את המפתחות בטבלה.

באופן פורמאלי: הפונקציה מקיימת את **הנחת הגיבוב האחיד הפשוט**:

- ההסתברות שמפתח יגובב לתא מסוים זהה עבור כל התאים
- אין תלות בין ערכי הגיבוב של מפתחות שונים

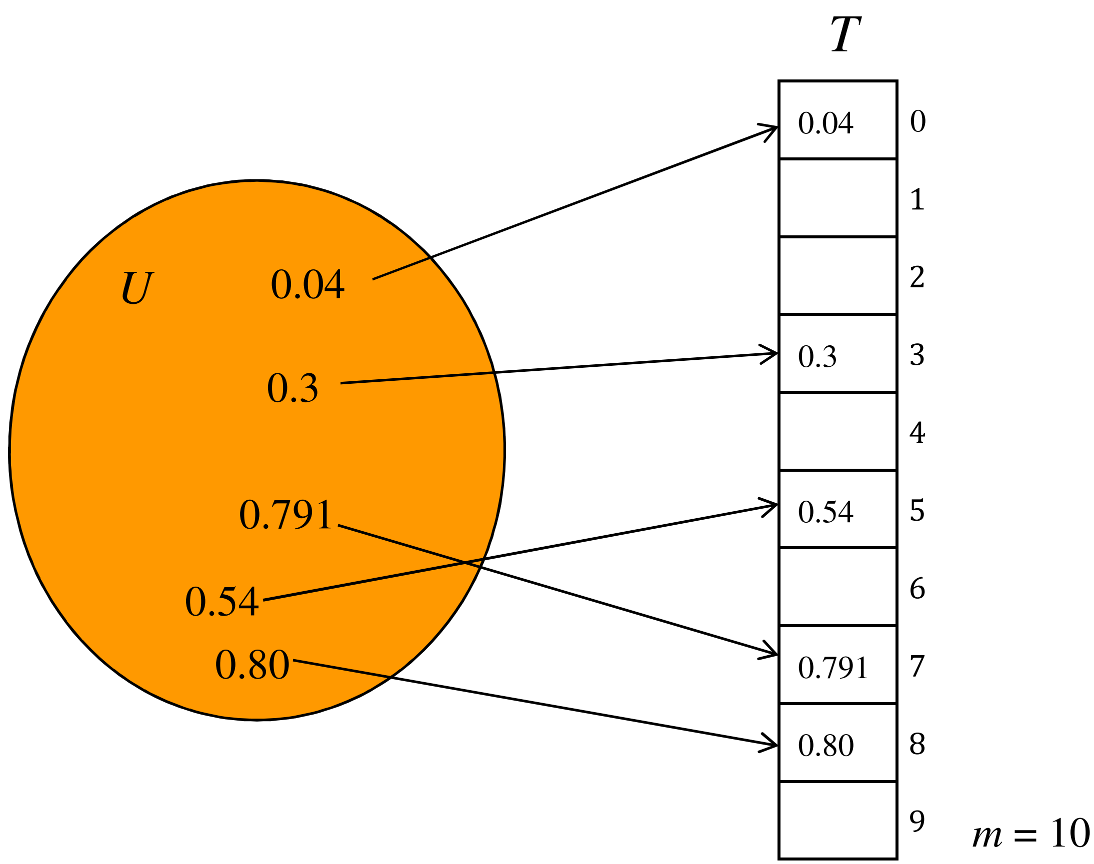
האם קיום תכונה 2 תמנע התנגשויות?

בחירת פונקצית גיבוב טובה

דוגמה לפונקצית גיבוב טובה, כאשר המפתחות הם ממשיים המתפלגים באופן אחיד ובלתי תלוי בקטע $[0,1)$.

$$h(k) = \lfloor km \rfloor$$

לדוגמה:



בחירת פונקצית גיבוב טובה

מידע על התפלגות המפתחות עוזר בתכנון פונקצית גיבוב טובה.

אך לא תמיד אנו יודעים מהי התפלגות המפתחות.

עדיין, ישנן שיטות לבחירת פונקצית גיבוב שנותנות בד"כ פיזור "טוב".

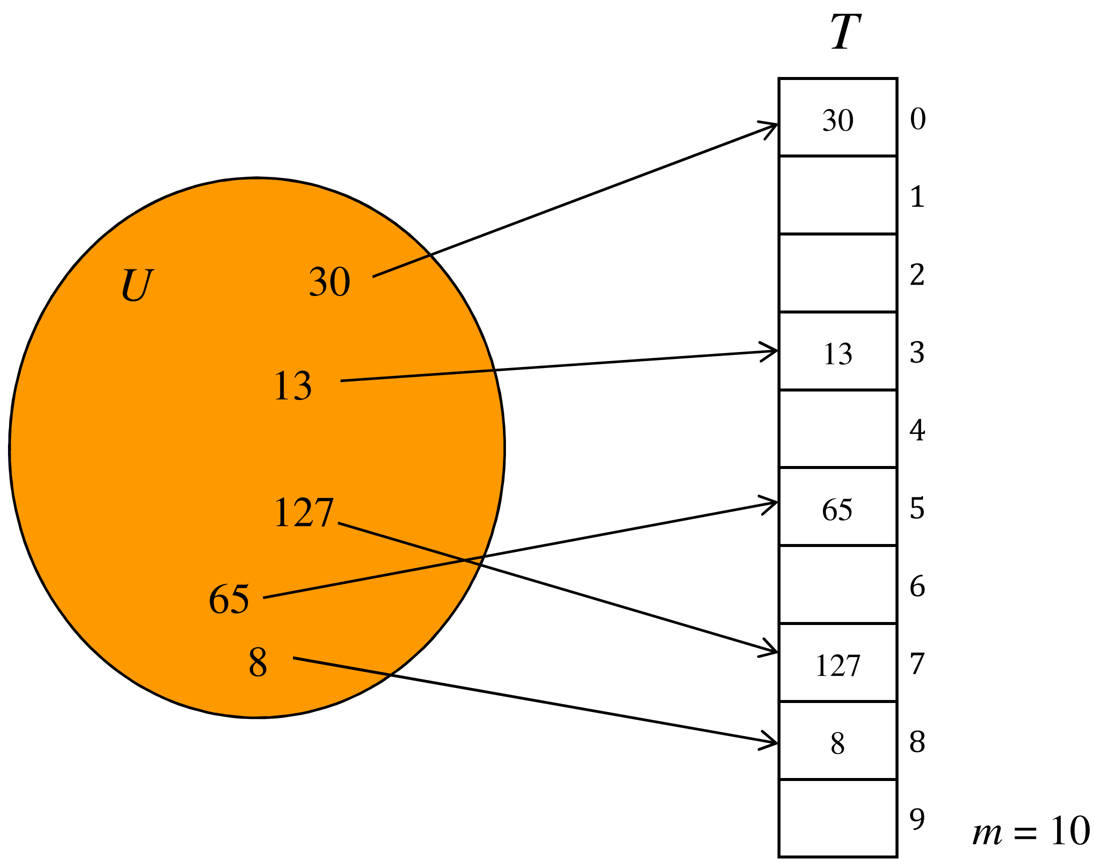
נכיר כעת 2 שיטות כאלו:

1. שיטת החילוק
2. שיטת הכפל

הרעיון הוא לבצע חישוב כזה, שצפוי שלא יהיה תלוי בתבניות הקיימות בהתפלגות המפתחות.

שיטת החילוק

$h(k) = k \bmod m$



שיטת החילוק

$h(k) = k \bmod m$

איזה חסרון קיים בבחירת m שהוא חזקה שלמה של 2?

רמז: חישבו על הייצוג הבינארי של המפתחות.

אם $m=2^p$, רק p הסיביות הפחות משמעותיות נלקחות בחשבון בחישוב ערך הגיבוב.

מסקנה: בשיטת החילוק, עדיף לא לבחור $m=2^p$, אלא אם ידוע מראש שיש הסתברות שווה להופעת כל אחת מהתבניות האפשריות של p הסיביות הפחות משמעותיות.

שיטת החילוק

המסקנה נכונה לא רק עבור בסיס 2:

- אם המפתחות הם מספרים בבסיס b כלשהו, בחירת $m=b^p$ יוצרת גיבוב שמתחשב רק ב- $p=\log_b m$ הספרות הפחות משמעותיות של המפתח.

- עדיף שערך הגיבוב יהיה תלוי בכמה שיותר מידע מהמפתח.

שיטת הכפל

$$h(k) = \lfloor m\,(k\,A - \lfloor k\,A \rfloor) \rfloor$$

כאשר A קבוע בתחום $0 < A < 1$.

בחירת הערך הספציפי של A תלויה במאפייני המפתחות.

הרעיון הוא שהערכים של $k\,A - \lfloor k\,A \rfloor$ יתפלגו באופן אחיד בקטע $[0, 1)$.

בשיטת הכפל ערכו של m אינו סובל בד"כ מההגבלות שראינו קודם בשיטת החילוק.

<u>תרגיל</u>

נתונה טבלת גיבוב בגודל m=10, ופונקצית גיבוב בשיטת הכפל, עם $A = (\sqrt{5}-1)/2$.

חשבו את המיקומים שאליהם ממופים המפתחות: 61, 62, 63, 64, 65.

מפתחות שאינם מספרים

3 השיטות שראינו לבחירת פונקציות גיבוב הניחו שהמפתחות הינם מספרים.

מה קורה כאשר אין הדבר כך?

למשל, כאשר המפתחות הם מחרוזות?

אז "נתרגם" את המפתחות למספרים.

למשל: נשתמש בקוד ה- ASCII של האותיות, וכך כל אות תתורגם למספר בבסיס 128:

a = 97

b = 98

…

“ab” = 97*128 + 98 = 12514

ישנן שיטות רבות אחרות, מתוחכמות הרבה יותר.

בחירת השיטה המתאימה תלויה במידה רבה במידע על מאפייני המפתחות.

פתרונות להתנגשויות

נציג כעת 2 שיטות לפתרון בעיית ההתנגשויות:

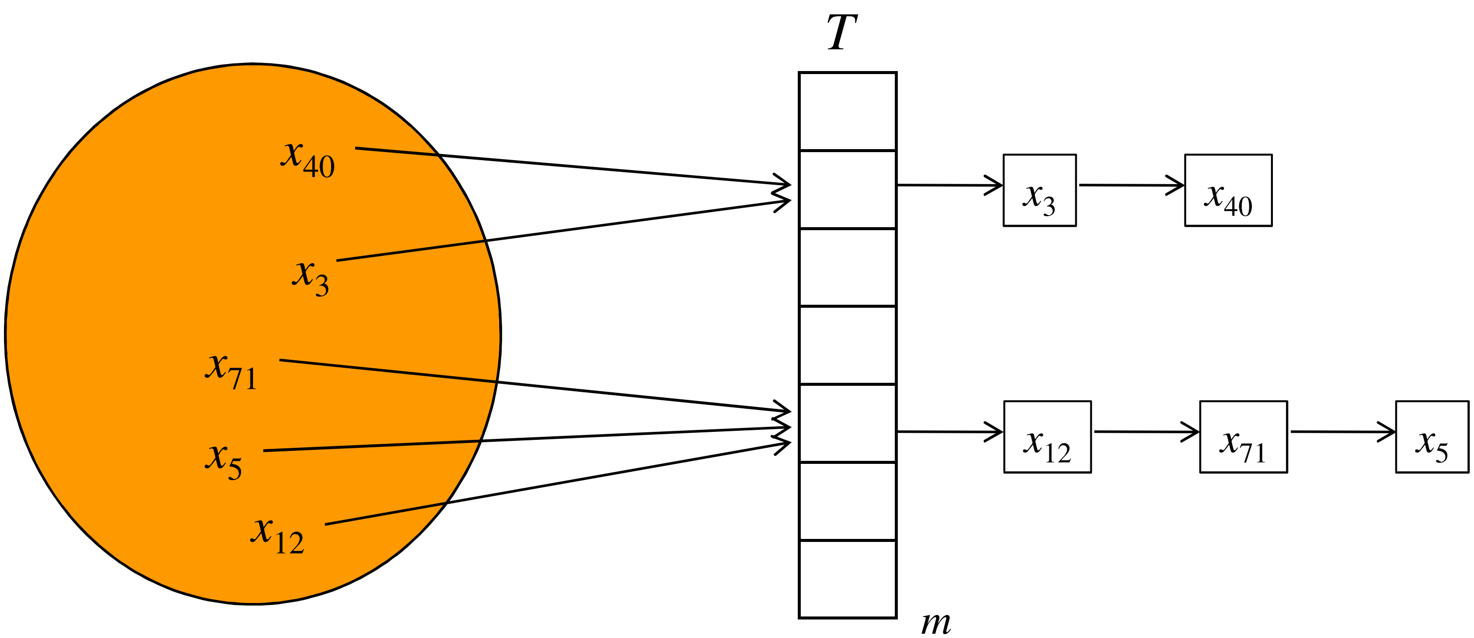
1) שיטת השרשור (chain hashing)

2) שיטת המיעון הפתוח (open addressing)

פתרונות להתנגשויות - שיטת השרשור

כל תא ב- T מצביע על רשימה מקושרת.

מפתח x יימצא ברשימה המקושרת בתא $T[h(x)]$.



Hash-Insert(x) – הכנס את x לראש הרשימה בתא $T[h(x)]$

Hash-Search(k) – חפש את המפתח k ברשימה בתא $T[h(k)]$

Hash-Delete(x) – מחק את x מהרשימה המקושרת בתא $T[h(x)]$

שיטת השרשור - סיבוכיות

ניתוח סיבוכיות

המקרה הגרוע

כל האיברים הוכנסו לאותה רשימה.

$\Theta(1)$ – Hash-Insert(x) *בהנחה שאין צורך לבדוק תחילה אם האיבר קיים כבר

$\Theta(n)$ – Hash-Search(k)

$\Theta(1)$ – Hash-Delete(x) *בהנחה שהרשימות דו-כיווניות, ונתון מצביע לאיבר למחיקה.

אחרת - $\Theta(n)$.

תרגיל

הציעו שינוי בשיטת השרשור, שיאפשר הכנסה, חיפוש ומחיקה ב- $\Theta(\log n)$ במקרה הגרוע.

שיטת השרשור - סיבוכיות

<u>ניתוח סיבוכיות ממוצעת, תחת הנחת הפיזור האחיד הפשוט</u>

נגדיר את פקטור העומס (load factor): $\alpha = n/m$

זהו למעשה האורך הממוצע של רשימה.

נוכיח כעת שכאשר התנגשויות נפתרות בשיטת השרשור, פעולת החיפוש רצה בזמן $\Theta(1+\alpha)$ בממוצע.

לפיכך, אם נבחר את גודל הטבלה m כך שיתקיים $n = O(m)$

אז $\alpha = O(m)/m = O(1)$ וכל הפעולות ירוצו בזמן $\Theta(1)$ בממוצע.

שיטת השרשור - סיבוכיות

משפט

בשיטת השרשור, ותחת הנחת הפיזור האחיד הפשוט,

חיפוש אחר מפתח ירוץ בזמן $\Theta(1+\alpha)$ בממוצע.

הוכחה

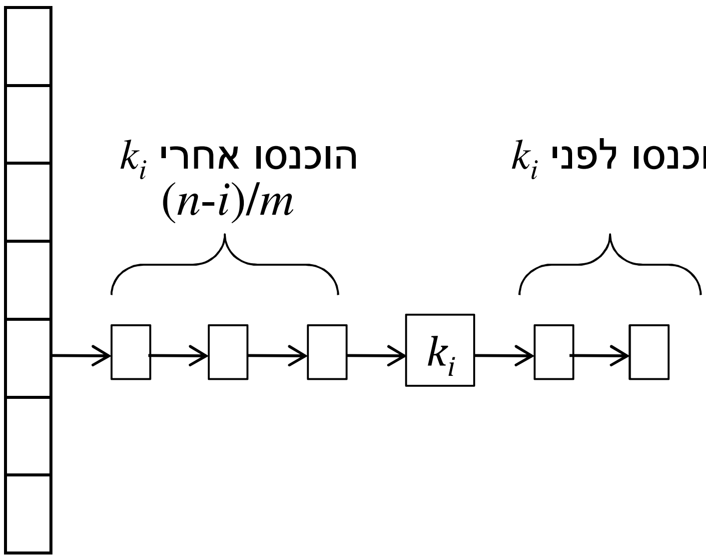
נתייחס תחילה לחיפוש כושל (המפתח לא נמצא).

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

- חיפוש אחר מפתח k כלשהו דורש מעבר על הרשימה $T[h(k)]$ עד סופה.
- אורכה הממוצע של רשימה זו, בהנחת הפיזור האחיד הפשוט הוא $\alpha = n/m$.
- לפיכך יש לעבור על $\alpha+1$ מצביעים בממוצע (כולל זה שבסוף הרשימה).
- יתר הפעולות (חישוב פונקצית הגיבוב וגישה לתא המתאים) יתבצעו ב- $\Theta(1)$ במקרה הגרוע.
- סה"כ $\Theta(1+\alpha)$ בממוצע.

שיטת השרשור - סיבוכיות

<u>הוכחה (המשך)</u>



נתייחס כעת למקרה של <u>חיפוש מוצלח</u> של מפתח.

נניח שסדר הכנסת האיברים היה $k_1,\ldots,k_n$.

- אחרי מפתח k_i כלשהו, נוספו למבנה $n\text{-}i$ מפתחות.
- לכן בממוצע, אורך הרשימה עד ל- k_i הוא $(n\text{-}i)/m$.
- מכאן שזמן החיפוש הממוצע של המפתח k_i הוא $1+\ (n\text{-}i)/m$
- נחשב את הממוצע על פני כל המפתחות:

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

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

שיטת השרשור - סיכום

פעולה	מקרה גרוע	ממוצע
חיפוש	$\Theta(n)$	$\Theta(1+\alpha)$
הכנסה	$\Theta(1)$*	$\Theta(1)$*
מחיקה	$\Theta(1)$**	$\Theta(1)$**

אם $n = O(m)$ אז כל הפעולות ירוצו ב- $\Theta(1)$ בממוצע.

* בהנחה שלא צריך לבדוק אם האיבר כבר נמצא. אחרת הזמן הוא כזמן החיפוש.

** בהנחה שהרשימות דו-כיווניות ונתון מצביע לאיבר למחיקה.

פתרונות להתנגשויות – מיעון פתוח

- בשיטת המיעון הפתוח (open addressing) אם תא מסוים תפוס, מחפשים תא פנוי אחר.

  כלומר כל תא מחזיק מידע על מפתח אחד לכל היותר.

  הדבר מחייב ש- $n \le m$, כלומר $\alpha \le 1$

- פונקצית הגיבוב מקבלת כפרמטר גם את מספר הניסיונות הקודמים, ומחזירה את המיקום הבא לבדיקה:

  $h : U \times \{0,1, \ldots, m\text{-}1\} \rightarrow \{0,1,\ldots, m\text{-}1\}$

  כלומר סדרת הבדיקות לתא פנוי היא $h(k, 0), \quad h(k, 1), \quad \ldots \quad , \quad h(k, m-1)$

  ונרצה שסדרה זו תהיה תמורה של $\{0,1,\ldots, m\text{-}1\}$

- נראה כעת 3 סוגי בדיקות שעובדים בשיטת המיעון הפתוח:

  1. בדיקה ליניארית, 2. בדיקה ריבועית, 3. גיבוב כפול

בדיקה ליניארית

פונקציית הגיבוב:

$$h(k, i) = (h'(k) + i) \bmod m$$

כאשר h' היא פונקציית גיבוב "רגילה"
(שיטת החילוק, הכפל, שיטה אחרת...)

בשיטה זו, אם תא מסוים תפוס, ננסה את התא הבא אחריו.

הדגמה:

שיטת החילוק: $h'(k) = k \bmod 17$

בדיקה ליניארית: $h(k, i) = (h'(k) + i) \bmod 17$

הקלט: *Input: 3, 5, 8, 13, 21, 55, 76, 131, 207, 338*

0		
1		
2		
3	**3**	3
4	**21**	4
5	**5**	5
6	**55**	4
7	**207**	3
8	**8**	8
9	**76**	8
10		
11		
12	**131**	12
13	**13**	13
14		
15	**338**	15
16		

יתרון: קל למימוש, נוצרת תמורה של $\{0,1,\ldots, m\text{-}1\}$

חיסרון: בעיית **ההצטברות הראשונית**: אם לתא פנוי קודמים i תאים תפוסים, הסיכוי שהוא יהיה התא הבא שיתמלא הוא $(i+1)/m$ ולא $1/m$. נוצרים "רצפים" ארוכים של תאים תפוסים.

מיעון פתוח – הכנסה וחיפוש

<u>הכנסת איבר בעל מפתח k</u>

1. $i \leftarrow 0$
2. $j \leftarrow h(k, i)$
3. **אם** התא $T[j]$ פנוי
4. מכניסים ל- $T[j]$ את האיבר החדש, ו**מחזירים** j
5. **אחרת:** $i \leftarrow i+1$
6. **אם** $i<m$ חוזרים ל- 2
7. **אחרת מחזירים** "*טבלה מלאה*"

<u>חיפוש איבר בעל מפתח k</u>

1. $i \leftarrow 0$
2. $j \leftarrow h(k, i)$
3. **אם** האיבר בתא $T[j]$ בעל מפתח k – **מחזירים** j
4. **אחרת אם** $T[j]$ פנוי – מחזירים "*האיבר לא נמצא*"
5. **אחרת:** $i \leftarrow i+1$
6. **אם** $i<m$ חוזרים ל- 2
7. **אחרת מחזירים** "*האיבר לא נמצא*"

מיעון פתוח - מחיקה

איזו בעיה מתעוררת במחיקה?

- אם פשוט נמחק איבר מהטבלה ע"י הצבת nil במקומו, אנו עלולים "לנתק שרשרת חיפוש".

0		
1		
2		
3	**3**	3
4	~~**21**~~	4
5	**5**	5
6	**55**	4
7	**207**	3
8	**8**	8
9	**76**	8
10		
11		
12	**131**	12
13	**13**	13
14		
15	**338**	15
16		

למשל:
Delete(21), ואח"כ Search(207).
החיפוש יגיע לתא ריק וייכשל.

<u>פתרון לבעיה</u>:
במקום להציב nil,
נציב בתא ערך מיוחד ***deleted***,
שמשמעותו: "התא פנוי לצורך הכנסה ותפוס לצורך חיפוש".

<u>החיסרון</u>:
אחרי מחיקות רבות, הטבלה עלולה להיות עמוסה בערכי ***deleted***, וזמני החיפוש יהיו ארוכים יחסית לכמות האיברים שנמצאים במבנה בפועל.
← לכן אם ישנן הרבה מחיקות, פתרון להתנגשויות בשיטת השרשור עדיף בד"כ.

מיעון פתוח - המשך

נראה כעת שתי שיטות בדיקה אחרות, שמנסות לשפר את שיטת הבדיקה הליניארית:

- בדיקה ריבועית
- גיבוב כפול

האלגוריתמים להכנסה, חיפוש ומחיקה לא ישתנו
(הם נכונים עבור הפרדיגמה הכללית של מיעון פתוח).

בדיקה ריבועית

פונקצית הגיבוב: $h(k, i) = (h'(k) + ai+bi^2) \bmod m$

כאשר a ו- $b\neq 0$ קבועים

0		
1		
2		
3	**3**	3
4	**21**	4
5	**5**	5
6	**55**	4
7		
8	**8**	8
9	**207**	3
10	**76**	8
11		
12	**131**	12
13	**13**	13
14		
15	**338**	15
16		

הדגמה:

שיטת החילוק: $h'(k) = k \bmod 17$

בדיקה ריבועית: $h(k, i) = (h'(k) + i + i^2) \bmod 17$

הקלט: *Input: 3, 5, 8, 13, 21, 55, 76, 131, 207, 338*

יתרון: לא נוצרים רצפים

חיסרון: בעיית **ההצטברות המשנית**: עדיין, אם לשני מפתחות $k_1 \neq k_2$ מתקיים ש- $h'(k_1) = h'(k_2)$
אז לשני המפתחות תהיה בדיוק אותה סדרת בדיקה.
זוהי צורה מתונה יותר של הצטברות.

גיבוב כפול

פונקצית הגיבוב:

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m$$

h_1 נקראת "פונקצית הבסיס"
h_2 נקראת "פונקצית הצעד"

הדגמה:

שיטת החילוק: $h_1(k) = k \bmod 17$

גיבוב כפול: $h_2(k) = k \bmod 16 + 1$

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod 17$$

הקלט: *Input: 3, 5, 8, 13, 21, 55, 76, 131, 207, 338*

0	**76**	8,13
1		
2	**207**	3,16
3	**3**	3,4
4	**21**	4,6
5	**5**	5,6
6		
7		
8	**8**	8,9
9		
10		
11		
12	**55**	4,8
13	**13**	13,14
14		
15	**338**	15,3
16	**131**	12,4

<u>יתרון</u>: סדרת הבדיקה תלויה לא רק במיקום הראשוני $h(k,0) = h_1(k)$, אלא גם במפתח.

גיבוב כפול – בחירת פונקצית הצעד

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m$$

בדוגמה האחרונה פונקצית הבסיס היתה $h_1(k) = k \bmod 17$
ופונקצית הצעד היתה $h_2(k) = k \bmod 16 + 1$

מדוע בחרנו כך את פונקצית הצעד?

<u>תרגיל</u>

א. מהי הבעיה בפונקצית הצעד הבאה? $h_2(k) = (k \bmod 17)$

ב. מהי הבעיה בפונקצית הצעד הבאה? $h_2(k) = (k \bmod 17) + 1$

גיבוב כפול – בחירת גודל הטבלה

$h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m$

איך כדאי לבחור את גודל הטבלה m?

נרצה שסדרת הבדיקה תבצע מעבר על כל התאים במערך (h תהיה "על" לכל מפתח).

כלומר:

- לכל k $h_2(k) \neq 0$
- לכל k ל- $h_2(k)$ ול- m אין מחלקים משותפים > 1

למשל אם m=20, אסור ש- h_2 תקבל (עבור אף מפתח k) את אחד מן הערכים:
2, 4, 5, 6, 8, 10, 12, 14, 15, 16, 18, 20,…

כדי לעמוד באילוצים האלו, אחת האפשרויות היא לבחור m ראשוני ולהגדיר:

$h_1(k) = k \bmod m$

$h_2(k) = (k \bmod (m-c)) + 1$

עבור c>0 קטן (למשל c=1 או c=2).

השוואה בין שיטות הבדיקה

ננסה להבין מדוע לגיבוב כפול ביצועים טובים יותר מאשר לשתי שיטות הבדיקה האחרות.

תרגיל

א – הסבירו מדוע במיעון פתוח אידיאלי ישנן $m!$ סדרות בדיקה שונות אפשריות.

כמה סדרות בדיקה שונות אפשריות:

ב - בבדיקה ליניארית? ג - בבדיקה ריבועית? ד - בגיבוב כפול?

פתרון

א - כי כל תמורה של $(0,1,\ldots,m\text{-}1)$ מהווה סדרת בדיקה.

ב+ג - המיקום ההתחלתי קובע את סדרת הבדיקה. לכן אפשריות רק m סדרות בדיקה שונות.

ד - סדרת הבדיקה נקבעת הן לפי המיקום ההתחלתי והן לפי גודל הצעד.
למיקום ההתחלתי יש m אפשרויות
לגודל הצעד $m\text{-}1$ אפשרויות (כאשר m ראשוני, כל צעד בגודל 1 עד $m\text{-}1$ יתאים)
לכן יש $\Theta(m^2)$ סדרות בדיקה שונות אפשריות.

לכן גיבוב כפול מהווה קירוב טוב יותר למיעון פתוח אידיאלי מאשר בדיקה ליניארית/ריבועית.

מיעון פתוח עם גיבוב כפול - תרגיל

תרגיל

נתונה טבלת גיבוב בגודל 7, עם פתרון להתנגשויות בשיטת מיעון פתוח עם גיבוב כפול:

פונקצית הבסיס: $h_1(k) = k \bmod 7$

פונקצית הצעד: $h_2(k) = (k \bmod 5) + 1$

הראו את תוצאת ביצוע הפעולות הבאות (משמאל לימין):

Insert(20) Insert(13) Insert(17) Delete(13) Search(17)

מיעון פתוח – סיבוכיות זמן

משפט (הוכחה בספר הלימוד, עמ' 203-205)

בהינתן טבלת גיבוב עם מיעון פתוח, שמקדם העומס שלה $\alpha = n/m < 1$,

ותחת הנחה (אידיאלית*) שלכל אחת מ- $m!$ סדרות בדיקה יש אותה הסתברות:

תוחלת מספר הבדיקות בעת חיפוש כושל חסומה ע"י $\dfrac{1}{1-\alpha}$

ואילו בעת חיפוש מוצלח היא חסומה ע"י $\dfrac{1}{\alpha}\ln\dfrac{1}{1-\alpha}$.

מסקנה

בתנאי המשפט, ועבור α קבוע, חיפוש (כושל או מוצלח) רץ בממוצע ב- $\Theta(1)$.

* כאמור, שיטות הבדיקה שראינו אינן מקיימות הנחה זו, אבל גיבוב כפול מקרב אותה טוב יחסית

מיעון פתוח - סיכום

פעולה		מקרה גרוע	ממוצע
חיפוש	כושל	$\Theta(n)$	$\Theta\left(\frac{1}{1-\alpha}\right)$
	מוצלח		$\Theta\left(\frac{1}{\alpha}\ln\frac{1}{1-\alpha}\right)$
הכנסה		$\Theta(n)$	כמו חיפוש כושל
מחיקה		$\Theta(1)$*	$\Theta(1)$*

זמני הריצה הממוצעים לחיפוש הם בהנחה (אידיאלית) שכל סדרות הבדיקה שוות הסתברות.

אם α קבוע, כל הפעולות ירוצו בזמן $\Theta(1)$ בממוצע.

* בהנחה שנתון מצביע או אינדקס למקום המחיקה

שאלות חזרה

1. השוו בין שתי השיטות לפתרון בעיית ההתנגשויות (שיטת השרשור, ומיעון פתוח) מן הבחינות הבאות:
   - הקשר הרצוי בין m ו- n
   - סיבוכיות זמן להכנסת איבר במקרה הגרוע
   - אופן מחיקת איברים

2. חיפוש איבר בטבלת גיבוב שבה התנגשויות נפתרות עם מיעון פתוח רץ במקרה הגרוע בזמן $\Theta(n)$. הסבירו מדוע.
   האם נכון יהיה לכתוב $\Theta(m)$ במקום $\Theta(n)$?
   האם הטענה הנ"ל נכונה גם לאחר מחיקות רבות מן הטבלה?

3. מדוע במיעון פתוח עם גיבוב כפול כדאי לבחור את גודל הטבלה m כמספר ראשוני?
   נתונה טבלת גיבוב בגודל m=8, כאשר התנגשויות נפתרות בשיטת המיעון הפתוח עם גיבוב כפול. פונקצית הבסיס היא $h_1(k) = k \bmod 8$ ופונקצית הצעד היא $h_2(k) = k \bmod 7+1$.
   מה יקרה אם ננסה להכניס איבר בעל מפתח 2, כאשר כל האינדקסים הזוגיים תפוסים כבר?
   האם יימצא תא פנוי להכנסה, אם הטבלה אינה מלאה?

   מה היה משתנה אילו גודל הטבלה היה 7 במקום 8?

תשובות לשאלות חזרה

1.

	Chain hashing	Open addressing
הקשר בין m ו- n	$n = O(m)$ (לצורך יעילות החיפוש)	$n \le m$
סיבוכיות זמן להכנסת איבר במקרה הגרוע	$O(1)$ בהנחה שלא צריך לבדוק אם האיבר כבר נמצא	$\Theta(n)$
מחיקת איברים	הוצאה רגילה מרשימה מקושרת	סימון המקום כ"פנוי להכנסה ותפוס לחיפוש"

2. במקרה הגרוע, סדרת החיפוש תיתקל שוב ושוב בתאים תפוסים. אם כן אורך סדרת חיפוש במקרה הגרוע הוא כמספר האיברים שמאוחסנים בטבלה ועוד אחד.

לא יהיה נכון לכתוב $\Theta(m)$, כי אורך סדרת החיפוש אינו בהכרח מסדר גודל של m. (ייתכן ש- $n=o(m)$).

לאחר מחיקות רבות, תאים רבים בטבלה יכילו את הערך *deleted*, וזמני הפעולות יהיו תלויים גם בכמות התאים הללו, ולא רק בכמות האיברים בטבלה.

3. למפתח 2 מתאימה סדרת הבדיקות הבאה: 2, 4, 6, 0, 2,... לכן לא יימצא תא פנוי, גם אם יש כזה (באינדקס אי-זוגי).
אם m=7 אז סדרת הבדיקות תהיה: 2, 4, 6, 1, 3, 5, 0. אם קיים תא פנוי – הוא יימצא.

תרגילים

<u>תרגילים נוספים</u>

1. אחת המגבלות לשימוש במיעון פתוח היא הצורך לדעת מראש את כמות האיברים שייכנסו לטבלה, כדי לדעת איזה גודל טבלה יש להקצות.

טבלת גיבוב דינאמית מציעה את הפתרון הבא:

בכל פעם שרוצים להכניס איבר והטבלה מלאה, ניצור טבלה חדשה, גדולה יותר (נשנה בהתאם את פונקציית הגיבוב), ונכניס אליה את כל האיברים ה"ישנים" אחד-אחד, וכן את האיבר החדש.

ציינו מהו זמן הריצה <u>הכולל</u> הממוצע, עבור הכנסת $2n$ איברים לטבלה בגודל n, ריקה בהתחלה:
א. כאשר גודל הטבלה החדשה גדל ב- 1 בכל פעם.
ב. כאשר גודל הטבלה החדשה גדל פי 2 בכל פעם.

הערה: הניחו כי שיטת הבדיקה בה משתמשים היא אידיאלית.

2. נתון מערך A של n מספרים כלשהם ומספר z.
תארו אלגוריתם שרץ בתוחלת זמן ליניארית, המחזיר זוג מספרים ב- A שסכומם z.

אם אין זוג מספרים כזה, האלגוריתם יחזיר nil.

3. בעיית היחידות (Element uniqueness problem):

נתונים n מספרים $x_1,\ldots,x_n$. הציעו אלגוריתם שבודק אם קיימים $i \neq j$ עבורם $x_i = x_j$.

פתרון 1

במיעון פתוח, אם שיטת הבדיקה בה משתמשים היא אידיאלית (כלומר לכל סדרות הבדיקה יש אותה הסתברות), אז הכנסה לטבלה רצה בממוצע ב- $\Theta(1)$.

בשני הסעיפים, n ההכנסות הראשונות לא דורשות הגדלה של הטבלה, ולכן רצות בזמן כולל של $n \cdot \Theta(1) = \Theta(n)$.

סעיף א'

כל אחת מההכנסות הבאות דורשת הגדלה של הטבלה, והעתקת כל האיברים. בפעם הראשונה יתבצעו $n+1$ הכנסות לטבלה החדשה, בפעם השנייה $n+2$, וכך הלאה (כל הכנסה בזמן קבוע בממוצע).

סה"כ מספר ההכנסות הוא $\sum_{i=n+1}^{2n} i = \Theta(n^2)$, ולכן זמן הריצה הכולל הממוצע הוא $\Theta(n^2)$.

סעיף ב'

תתבצע הגדלה אחת בלבד, בעת הכנסת האיבר ה- $n+1$, לטבלה בגודל $2n$. הגדלה זו (כולל הכנסת האיברים) תרוץ בזמן ממוצע $\Theta(n)$. כל n-1 ההכנסות הבאות ירוצו בזמן קבוע בממוצע.

זמן הריצה הכולל הממוצע הוא אם כן $\Theta(n)$.

פתרון 2

נכניס את כל המספרים לטבלת גיבוב, ולאחר מכן נעבור על המספרים בזה אחר זה, ולכל מספר k נחפש בטבלה את z-k. אם מצאנו, נחזיר את זוג המספרים שמצאנו. אחרת נחזיר בסוף nil.

נבחר את הפרמטרים השונים של טבלת הגיבוב:

- פונקצית הגיבוב תהיה בשיטת הכפל (שיטת החילוק מתאימה רק לשלמים).
- התנגשויות ייפתרו, למשל, בשיטת השרשור (אין מניעה גם לשימוש במיעון פתוח)
- בתנאים אלו האילוץ היחיד על גודל הטבלה הוא n=O(m), לכן נבחר למשל m=0.5n.

סיבוכיות: יש n הכנסות ו(לכל היותר) n חיפושים.

כל הכנסה תרוץ בזמן קבוע במקרה הגרוע, ואילו חיפוש בזמן קבוע בממוצע.

לכן בסה"כ האלגוריתם ירוץ בזמן ליניארי בממוצע:

$$\underset{\text{worst-case}}{\Theta(n)} + \underset{\text{average}}{\Theta(n)} = \underset{\text{average}}{\Theta(n)}$$

פתרון 3

פתרון ראשון

מיון, ואח"כ מעבר נוסף ובדיקה האם יש שני איברים סמוכים זהים.

סיבוכיות זמן: $\Theta(n \log n)$ (למשל מיון מיזוג).

פתרון שני

נכניס את המספרים לטבלת גיבוב, בשיטת השרשור (אפשר היה לפתור גם עם מיעון פתוח).

פונקצית הגיבוב וגודל הטבלה m ייבחרו באחת הדרכים שלמדנו.

אם יש התנגשות, נבדוק האם האיבר שמכניסים שווה לאחד האיברים ברשימה.

סיבוכיות זמן: נשים לב שהכנסה כאן שונה מעט מהכנסה רגילה, והיא למעשה שקולה מבחינת סיבוכיות זמן לחיפוש.

נסמן ב- α_i את פקטור העומס בעת הכנסת האיבר ה- i. הכנסה זו רצה בזמן ממוצע של $\Theta(1+\alpha_i) = \Theta(1+ i/m)$. ובסה"כ, אם אין אף שני איברים זהים:

$$\sum_{i=1}^{n} \Theta(1+i/m) = \Theta(n) + \Theta(n^2/m) = \Theta(n+\alpha n)$$

אם נבחר את m כך ש- $n=O(m)$ אז α קבוע והסיבוכיות הממוצעת תהיה ליניארית ב- n.