# חידה לחימום

דנה ותמר משחקות במשחק הבא:

על לוח המשחק מונחים בהתחלה 25 אסימונים. השחקניות משחקות לסירוגין. דנה משחקת ראשונה. בכל תור, השחקנית שזהו תורה מסירה מספר אסימונים מהלוח. היא חייבת להסיר לפחות אסימון אחד, אבל לא יותר מ-$\left\lceil \frac{X}{4} \right\rceil$ אסימונים, כאשר $X$ הוא מספר האסימונים שנמצאים באותו רגע על הלוח. המנצחת במשחק היא השחקנית שלוקחת את האסימון האחרון.

**לדוגמא:**

אם על הלוח יש 12 אסימונים, מותר להסיר בין 1 ל-3 אסימונים.

אם על הלוח 13 אסימונים, מותר להסיר בין 1 ל-4 אסימונים.

במשחק שמתחיל מ-3 אסימונים, דנה יכולה (ולמעשה, חייבת) להסיר אסימון אחד בתור הראשון (ואז נותרים שני אסימונים), ואז תמר חייבת להסיר אסימון אחד, ולבסוף – דנה מסירה את האסימון האחרון, ומנצחת.

א. דנה יכולה לשחק באופן שיבטיח לה את הניצחון. כמה אסימונים היא צריכה להסיר בתור הראשון?

ב. דנה ותמר משחקות שוב, אבל הפעם מונחים על הלוח בתחילה 30 אסימונים. כמה אסימונים צריכה דנה להסיר בתור הראשון הפעם?

# מבני נתונים ויעילות אלגוריתמים

**(9.2.2015)**

## מימוש טנ"מ 'מילון' באמצעות טבלות גיבוב

כזכור, אחד מטיפוסי הנתונים המופשטים (טנ"מ) הבסיסיים, הוא טנ"מ בשם 'מילון' (Dictionary), התומך בשלוש פעולות ממשק: הכנסת איבר (Insert), חיפוש איבר (Search), מחיקת איבר (Delete). נניח כי כל איבר מאופיין על-ידי שדה מיוחד הנקרא מפתח (key).

הפעולה של הכנסת איבר – מקבלת איבר מסוים x ומכניסה אותו למקום מסוים בתוך המילון;

הפעולה של חיפוש איבר – מקבלת מפתח מסוים k, ומחזירה את האיבר במילון שזהו המפתח שלו (או – אם אין איבר כזה, מחזירה ערך המציין שגיאה).

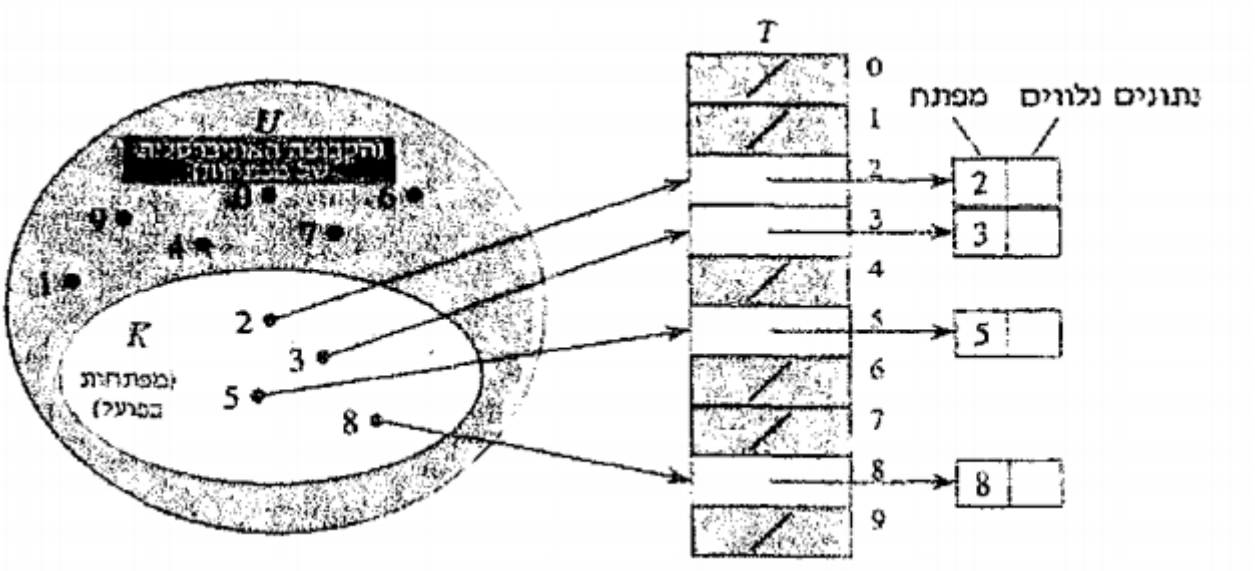
הפעולה של מחיקת איבר – מקבלת איבר מסוים הנמצא במילון (למשל – ייחוס או מצביע אליו), ומוחקת אותו מהמילון.

ישנן דרכים רבות לממש טנ"מ מילון, ואחת היעילות שבהן – היא באמצעות מבנה נתונים בשם 'טבלת גיבוב' (Hash Table), הנקרא גם לעיתים 'טבלת ערבול'. טבלת גיבוב ניתן לממש בצורות שונות.

## א. טבלאות מיעון ישיר (Direct Access Tables)

נניח שכל איבר מאופיין על-ידי מפתח ייחודי, ונניח שכל המפתחות לקוחים מתוך קבוצה אוניברסאלית של מפתחות, שנסמנה $U = \{0,1,2,3,\ldots,k-1\}$. נניח כי גודלה של הקבוצה, $|U| = k$, אינו מספר גדול.

נוכל לייצג את האיברים באמצעות מערך T שגודלו k, שכל תא בו מכיל מצביע. כאשר תא הוא ריק – הוא יכיל את הערך NULL, וכאשר תא i אינו ריק ($0 \le i \le k-1$) – הוא יצביע לאיבר שהמפתח שלו הוא i.



- מדוע דרשנו שכל איבר יהיה מאופיין על-ידי מפתח **ייחודי** (ערך שאינו חוזר על עצמו פעם נוספת), ולא הרשנו שיהיו כפילויות במפתח?
- מדוע הנחנו כי הקבוצה האוניברסאלית U, ממנה לקוחים ערכי המפתחות, אינה קבוצה גדולה?

ברור שבשיטת ייצוג זו, שלוש פעולות המילון (הכנסה, חיפוש ומחיקה), אורכות זמן קבוע $\Theta(1)$ כל אחת.

**<u>שאלה</u>**

ממשו, בעזרת אלגוריתם מתאים, כל אחת משלוש פעולות המילון, ובדקו שאכן ביצוען אורך זמן קבוע.

**<u>שאלה</u>**

נתונה טבלת מיעון ישיר בגודל k תאים, המכילה n איברים. מעוניינים למצוא את האיבר בעל המפתח המקסימלי מבין האיברים המאוחסנים בטבלה. מהי סיבוכיות זמן הריצה של הפעולה?

## ב. טבלאות גיבוב (Hash Tables)

הקושי בשימוש במיעון ישיר הוא ברור: אם הקבוצה האוניברסאלית U היא גדולה, אחסונה בזיכרון המחשב של טבלת מיעון ישיר בגודל |U| עשוי להיות בלתי מעשי, ואולי אפילו בלתי אפשרי. בנוסף, ייתכן שקבוצת האיברים המאוחסנים בפועל היא מאוד קטנה ביחס ל-U, כך שרוב המקום שהקצנו בזיכרון יתבזבז (במקרה כזה אומרים שנצילות המקום בזיכרון היא נמוכה).

לדוגמא, אם לכל סטודנט מחמש מאות הסטודנטים הלומדים במכללה ניתן מספר זיהוי בין 0 ל-499, אז אפשר לאחסן את כל פרטיהם בטבלת מיעון ישיר בת 500 תאים, באופן שהוא יעיל הן מבחינת סיבוכיות זמן ריצה (איתור סטודנט אורך זמן קבוע) והן מבחינת סיבוכיות מקום בזיכרון (נצילות המקום בזיכרון היא 100%).

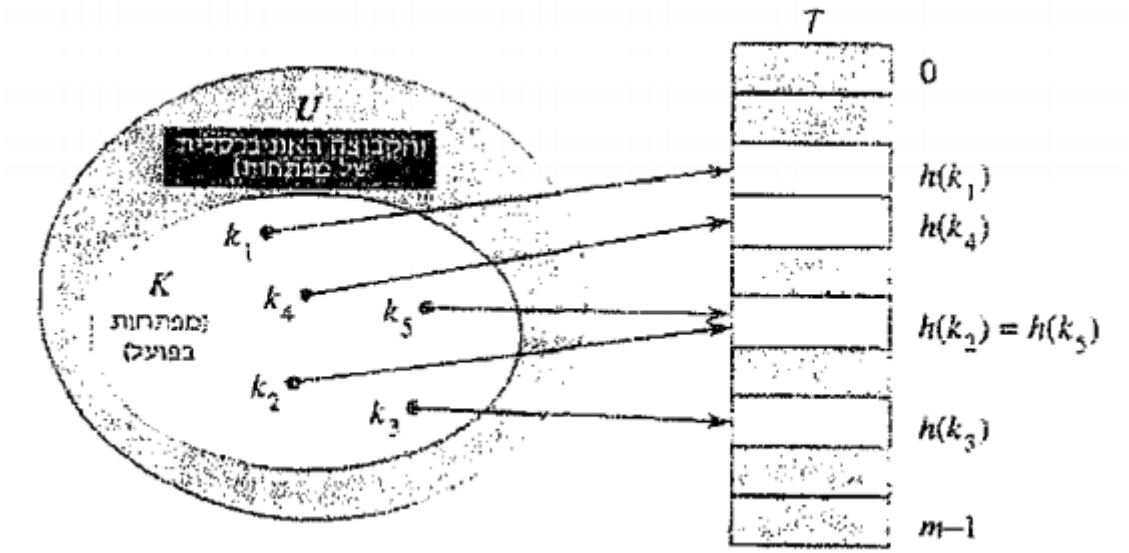
אם, לעומת זאת, מספר הזיהוי של כל סטודנט יהיה מספר תעודת הזהות שלו (רצף של תשע ספרות עשרוניות), אז גודלה של טבלת המיעון הישיר יהיה מיליארד תאים ($10^9 = 1{,}000{,}000{,}000$), אך רק 500 מתוכם יהיו מנוצלים. זו נצילות של 0.0005%, כלומר – של פחות מאלפית האחוז.

אם ידוע לנו שקבוצת המפתחות **המאוחסנת בפועל** במילון קטנה בהרבה מהקבוצה האוניברסאלית U של כל המפתחות האפשריים, אז נשתמש בטבלת גיבוב (Hash Table), במקום בטבלת מיעון ישיר.

מה הרעיון מאחורי טבלת גיבוב? בעוד במיעון ישיר, האיבר שהמפתח שלו הוא k מאוחסן בתא T[k], הרי שכעת נגדיר פונקצית גיבוב, המכונה לעיתים גם – פונקצית ערבול (hash function), שתסומן בד"כ ב-h, ונאחסן את האיבר שהמפתח שלו הוא k, בתא שהאינדקס שלו הוא h(k). כלומר, בתא T[h(k)].

אם גודלו של המערך T הוא m תאים (הממוספרים מ-0 עד m-1), אז תפקידה של פונקצית הגיבוב h הוא למפות את הקבוצה האוניברסאלית U אל התאים שבטבלה T: $\{0,1,\ldots,m-1\}$. $h: U \rightarrow \{0,1,\ldots,m-1\}$

נדרוש שפונקצית הגיבוב תהיה כזו שקל לחשב אותה. כלומר, שחישוב h(k) מתבצע בזמן קבוע.

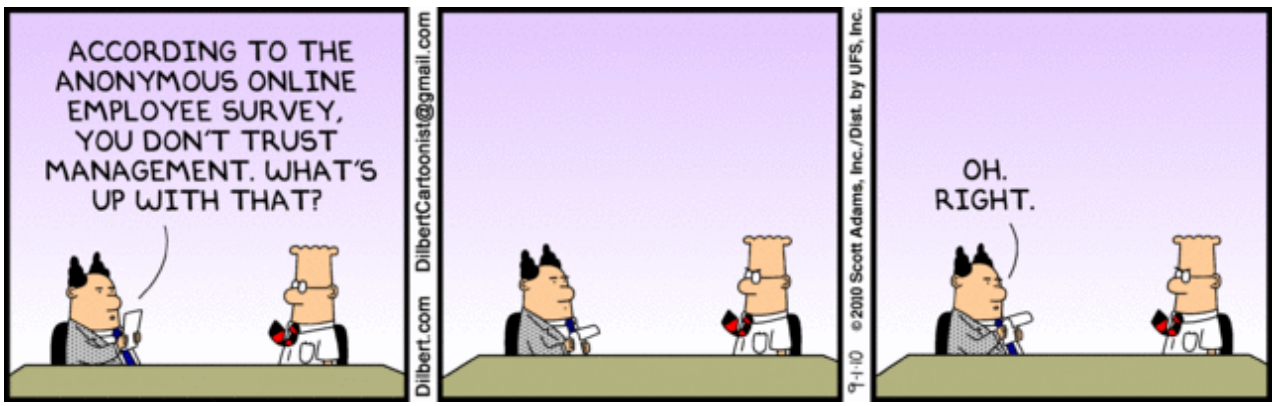



**<u>שאלה</u>**

נתונה קבוצת המפתחות הבאה: 102, 515, 673, 679, 252, 604, 701, 883. מעוניינים לאחסן אותם בטבלת גיבוב T שגודלה 10 תאים (הממוספרים מ-0 עד 9). פונקציית הגיבוב היא: h(key) = key % 10.
שרטטו את טבלת הגיבוב שתיווצר כתוצאה מהכנסת המפתחות. איזו בעיה התעוררה?

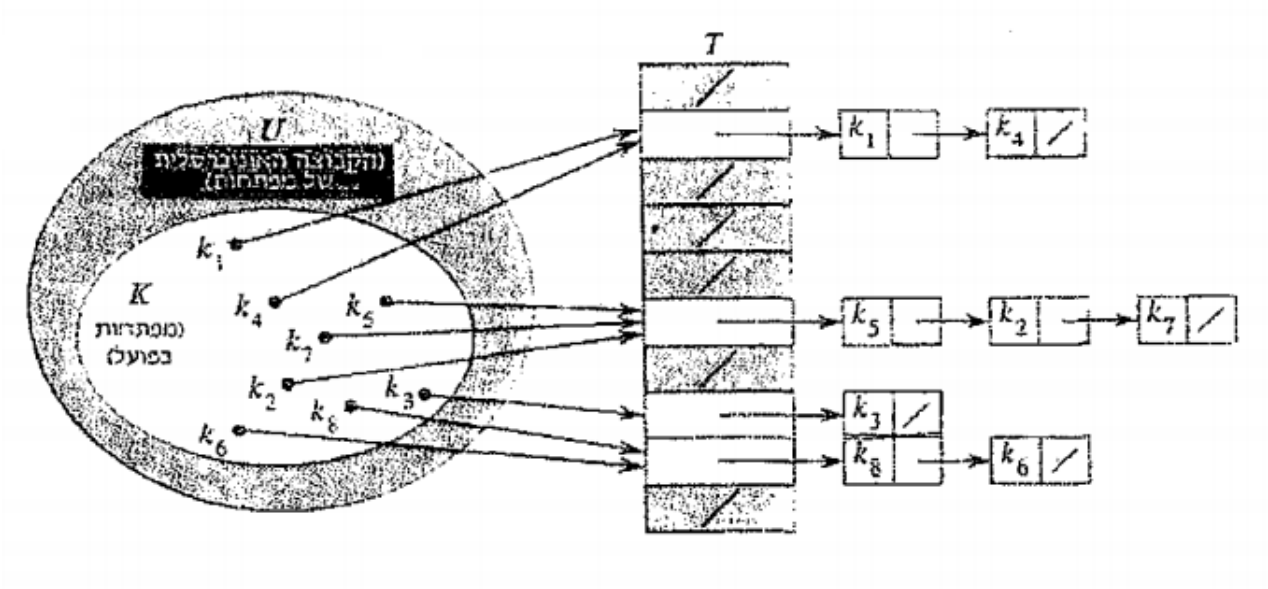
הבעיה בשיטה זו היא שייתכנו מקרים שבהם שני מפתחות שונים יגובבו לאותו תא. מקרה כזה מכונה 'התנגשות' (collision), ויש למצוא דרכים לטפל בהתנגשויות.

כשבוחרים פונקציית גיבוב, היינו רוצים לבחור אחת כזו המצמצמת את כמות ההתנגשויות. אולם לא ניתן למנוע לגמרי התנגשויות, שכן אי אפשר למפות |U| מפתחות ל-m תאים, כאשר |U| גדול מ-m, מבלי שיהיה לפחות זוג אחד של מפתחות המתנגשים האחד עם השני. לכן, נצטרך לפתח שיטות לטיפול בהתנגשויות.



## ג. פתרון התנגשויות על-ידי שרשור

בשיטת השרשור (chaining), אנו יוצרים רשימה מקושרת מכל האיברים המגובבים לאותו תא. התא j יכיל מצביע לראש הרשימה של כל האיברים שפונקציית הגיבוב מיפתה את מפתחותיהם לתא j. אם אין איברים כאלה, התא מכיל את הערך NULL.



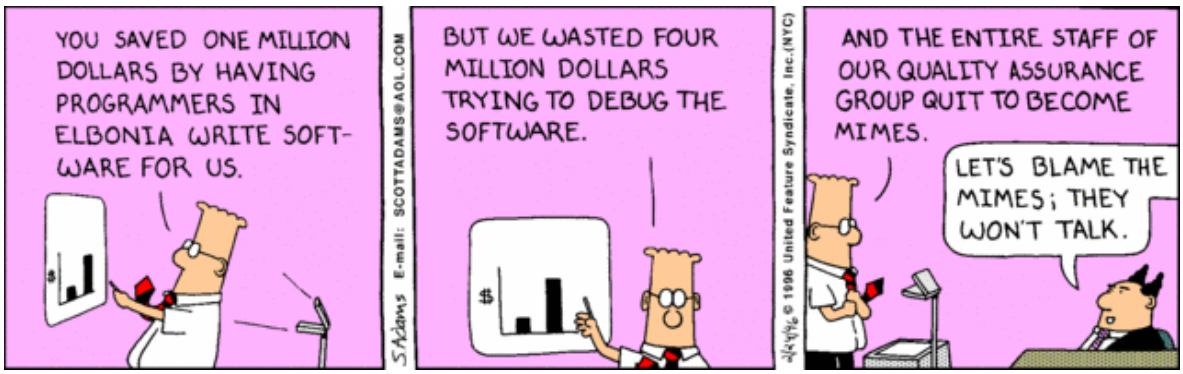

בשיטה זו, מימוש פעולת הכנסה (Insert) של איבר x, שהמפתח שלו הוא x.key, יעשה על-ידי הוספת האיבר לראש הרשימה T[h(x.key)].

פעולת חיפוש (Search) של איבר שהמפתח שלו k, תיעשה על-ידי סריקת הרשימה המקושרת T[h(k)].

מחיקת איבר (Delete), תיעשה על-ידי מחיקת האיבר המבוקש, הנמצא ברשימה T[h(x.key)]. כדי שהמחיקה תיעשה ביעילות – כדאי שהרשימה המקושרת תהיה רשימה דו-כיוונית (אחרת, נצטרך לסרוק את הרשימה מתחילתה כדי למצוא את החוליה הקודמת לאיבר x).

מהי הסיבוכיות של כל אחת מהפעולות האלה?

- ברור שהכנסה תתבצע בזמן קבוע $\Theta(1)$.
- חיפוש של איבר דורש מאיתנו לסרוק את הרשימה T[h(x.key)]. במקרה הגרוע ביותר, נצטרך לסרוק את הרשימה הזו כולה. נדון בסיבוכיות של חישוב זה בפירוט בהמשך.
- מחיקה של איבר ניתן לבצע בזמן קבוע, בתנאי שהרשימות הן דו-כיווניות. במידה ולא, עלינו לסרוק את הרשימה T[h(x.key)] מתחילתה, כדי שנוכל למצוא את האיבר הקודם ל-x, ולעדכן את ה-next שלו לעוקב של x. גם כאן, כמו במקרה של חיפוש, צריך, במקרה הגרוע, לסרוק את הרשימה כולה.



## ד. ניתוח סיבוכיות של גיבוב עם שרשור

עד כמה טובים הביצועים של גיבוב עם שרשור? כמה זמן אורך חיפוש של איבר בעל מפתח נתון?

בהינתן טבלת גיבוב T בעלת m תאים המכילה n איברים, נגדיר את **מקדם העומס** (load factor) של T, אשר יסומן באות $\alpha$, על-ידי $\frac{n}{m}$. כלומר, מדובר במספר **הממוצע** של איברים המאוחסנים ברשימה מקושרת אחת. נשים לב ש- $\alpha$ יכול להיות שווה ל-1, גדול מ-1, או קטן מ-1.

מהו המקרה הגרוע ביותר (W.C)? במקרה הזה, כל n האיברים מגובבים לאותו תא במערך T, ויוצרים רשימה מקושרת בת n איברים. במקרה הזה, חיפוש של איבר זו פעולה המתבצעת בזמן לינארי $\Theta(n)$ .

במקרה הממוצע (A.C) ביצועי הגיבוב תלויים במידת הפיזור שמפזרת פונקצית הגיבוב h, בממוצע, את קבוצת המפתחות, בין m תאי המערך. ננית כי הגיבוב הוא אחיד (uniform). במקרה כזה, חיפוש אורך זמן ממוצע של $\Theta(\alpha)$, כלומר - $\Theta(\frac{n}{m})$.

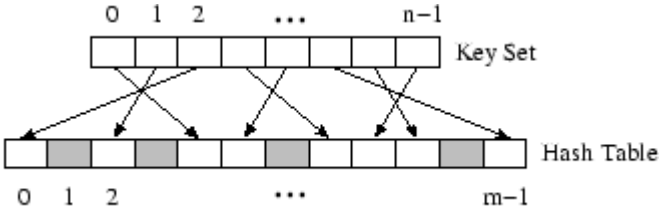
כעת, אם נבחר את מספר התאים m בטבלת הגיבוב T לעמוד ביחס ישר לכמות האיברים n שיש לאחסן בטבלה, כלומר – אם $m = \Theta(n)$, אז $\alpha = \frac{n}{m} = \frac{n}{\Theta(n)} = \Theta(1)$, דהיינו – חיפוש יתבצע **בממוצע** בזמן קבוע.

מכיוון שהכנסה מתבצעת בזמן קבוע, ומחיקה מתבצעת אף היא בזמן קבוע (אם הרשימות הן דו-מקושרות), נקבל שבמימוש טנ"מ מילון באמצעות טבלת גיבוב – כל פעולות המילון מתבצעות בזמן **ממוצע** קבוע $\Theta(1)$ .

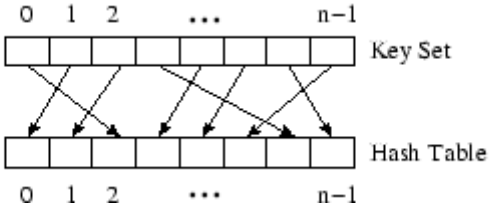
**שאלה**

הכניסו את המפתחות הבאים (משמאל לימין) 10, 17, 12, 33, 20, 15, 19, 28, 5 לטבלת גיבוב שבה התנגשויות נפתרות על-ידי שרשור. הטבלה מכילה 9 תאים, ופונקצית הגיבוב היא h(k) = k % 9 .

פונקצית גיבוב תקרא **מושלמת** (perfect) אם היא ממפה מפתחות שונים לתאים שונים, ללא התנגשות.




פונקצית גיבוב תקרא **מינימאלית מושלמת** (minimal perfect) אם היא פונקצית גיבוב מושלמת, אשר ממפה קבוצה אוניברסאלית U של מפתחות לתוך טבלה בגודל |U|.



## ה. שיטות לבחירת פונקצית גיבוב

אחת מהדרכים לבניית פונקצית גיבוב היא שיטת החלוקה (division method), שבה ממפים מפתח k אל אחד מ-m תאי הטבלה, על-ידי חישוב שארית החלוקה של k ב-m, כלומר, באופן הבא: $h(k) = k \% m$.

כאשר משתמשים בשיטת החלוקה, נמנעים בדרך כלל מבחירת ערכים מסוימים של m. לדוגמא, רצוי ש-m לא יהיה חזקה של 2, שכן אם $m = 2^p$, אז $h(k)$ הוא פשוט p הסיביות הפחות משמעותיות של k. כדי להשיג התפלגות אחידה, מוטב שפונקצית הגיבוב תהיה תלויה בכל הסיביות של המספר, ולא רק ב-p הימניות ביותר. ערכים טובים של m הם מספרים ראשוניים, שאינם קרובים מדי לחזקות של 2.

לדוגמא, נניח שאנחנו מעוניינים להקצות טבלת גיבוב, עם פתרון התנגשויות על-ידי שרשור, לאחסון $n = 2000$ מפתחות. נניח גם כי אנו מוכנים לבחון, במהלך חיפוש כושל, 3 איברים בממוצע. בתנאים אלה, נבחר טבלת גיבוב בת $m = 701$ תאים. המספר 701 נבחר מכיוון שהוא ראשוני הקרוב ל-2000/3, אולם רחוק מכל חזקה של 2. פונקצית הגיבוב שלנו תהיה: $h(k) = k \% 701$.

שיטה אחרת לבניית פונקצית גיבוב נקראת שיטת הכפל (multiplication method), והיא עובדת בשני שלבים: תחילה, כופלים את המפתח k בקבוע A בתחום $0 < A < 1$, ולוקחים את החלק השברי של התוצאה kA. לאחר מכן, כופלים ערך זה ב-m, ולוקחים את הערך השלם של התוצאה. בקיצור, פונקצית הגיבוב היא:

$$h(k) = \lfloor m \cdot (kA - \lfloor kA \rfloor) \rfloor$$

אף ששיטה זו פועלת עם כל ערך של הקבוע A, היא פועלת עם ערכים מסוימים טוב יותר מאשר עם אחרים. הבחירה האופטימלית תלויה במאפייני הנתונים שיש לגבב. מדען המחשב Donald Knuth טען כי הערך $A = \frac{\sqrt{5}-1}{2} = 0.6180339887...$ הוא מוצלח, מבחינה מעשית.

דוגמא: נניח כי A הוא הקבוע הנ"ל, ונגבב את המפתח $k = 123456$ לתוך טבלה בת $m = 10000$ תאים:

$$\begin{aligned} h(123456) &= \lfloor 10000 \cdot (123456 \cdot 0.61803 - \lfloor 123456 \cdot 0.61803 \rfloor) \rfloor \\ &= \lfloor 10000 \cdot (76299.51168 - \lfloor 76299.51168 \rfloor) \rfloor \\ &= \lfloor 10000 \cdot 0.51168 \rfloor \\ &= \lfloor 5116.8 \rfloor \\ &= 5116 \end{aligned}$$

כלומר, האיבר שהמפתח שלו הוא 123456 ימופה לתא 5116 בטבלה.

## ו. פתרון התנגשויות בשיטת המיעון הפתוח

בשיטת המיעון הפתוח (open addressing) כל האיברים מאוחסנים בטבלת הגיבוב עצמה. כל תא בטבלה מכיל איבר או NULL. כאשר מחפשים איבר, בוחנים בשיטתיות תאים בטבלה, עד שמוצאים את האיבר המבוקש או מגיעים למסקנה שהוא לא נמצא בטבלה.

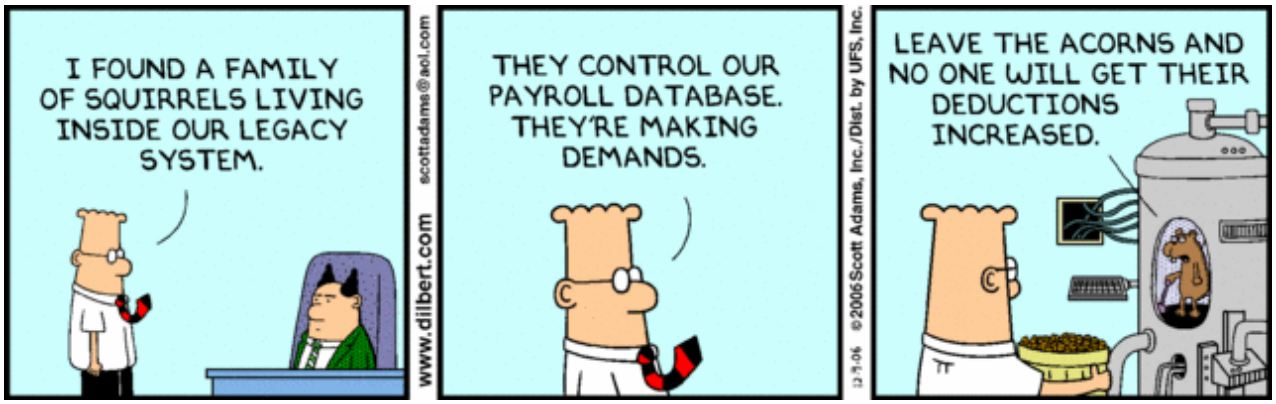
בניגוד לשיטת השרשור, בשיטה זו לא משתמשים ברשימות מקושרות, ולא מאחסנים איברים מחוץ לטבלה. לפיכך, במיעון פתוח קיימת אפשרות שטבלת הגיבוב "תתמלא", ולא נוכל להכניס אליה איברים נוספים. מקדם העומס $\alpha$ אינו יכול לעלות על 1.

כדי להכניס איבר לטבלת גיבוב עם מיעון פתוח, אנו **בודקים** (probe) בזה אחר זה תאים בטבלה, עד שמוצאים תא ריק, שניתן להציב בתוכו את המפתח. סדרת התאים הנבדקת, תלויה במפתח שאותו מכניסים.

אחת השיטות לביצוע מיעון פתוח, היא שיטה הנקראת בדיקה לינארית (linear probing). בשיטה זו, בהינתן מפתח k, בודקים את התא T[h(k)]. התא הבא שנבדק הוא T[h(k)+1], ואז T[h(k)+2], וכך עד ל-T[m-1] . אז עוברים בצורה מעגלית לתאים T[0], T[1], … , עד שלבסוף נבדק התא T[h(k)-1]. זאת נבצע בעת הכנסה ובעת חיפוש.

בעת מחיקת איבר, נסמן את התא בו הוא מאוחסן כ-'משומש', כדי שנדע להבדיל בינו לבין תא פנוי.

בדיקה לינארית היא קלה למימוש, אולם היא סובלת מבעיה של הצטברות (clustering): נוצרים רצפים ארוכים של תאים תפוסים, המאריכים את זמן החיפוש הממוצע.

### שאלה

יש להכניס את המפתחות 59, 88, 17, 28, 15, 4, 31, 22, 10 לטבלת גיבוב בגודל m = 11 באמצעות מיעון פתוח עם בדיקה לינארית, כאשר פונקצית הגיבוב היא h(x) = x % 11 .