



מבני נתונים ואלגוריתמים

טבלאות ערבול

הקדמה – תרגיל

נתון מערך **ממזין** של מספרים $V = x_1, x_2 \dots x_n$.

כתוב אלוגריתם המקבל כקלט מספר y ומחזיר האם קיים זוג איברים בקבוצה
כך $x_i, x_j \in V$ ש $x_i + x_j = y$.

הקדמה – תרגיל

נתון מערך **ממוין** של מספרים $V = [x_1, x_2 \dots x_n]$ מאורך n .

כתוב אלגוריתם המקבל כקלט מספר y ומחזיר האם קיים זוג איברים בקבוצה $x_i, x_j \in V$ כך ש $x_i + x_j = y$, במידה וכן האלגוריתם מחזיר את האינדקסים.

פתרון:

נניח $k < t$ - האינדקסים אותם נרצה להחזיר. נסתכל תחילה על הסכום $V[0] + V[n-1]$:

אם $V[0] + V[n-1] < y$ אזי נרצה להגדיל את הסכום: אזי $k > 0$

אם $V[0] + V[n-1] > y$ אזי נרצה להגדיל את הסכום: אזי $t < n-1$.

אחרת: מצאנו.

הקדמה – תרגיל

פסאודו קוד:

```
Find_sum(V,y):  
k=0,t=n-1  
While k<t:  
    if V[k]+V[t]== y:  
        Return k,t  
    Else if V[k]+V[t] >y:  
        t=t-1  
    Else:  
        k=k+1  
Return false
```

הקדמה – תרגיל

פסאודו קוד:

```
Find_sum(V,y):  
k=0,t=n-1  
While k<t:  
    if V[k]+V[t]== y:  
        Return k,t  
    Else if V[k]+V[t] >y:  
        t=t-1  
    Else:  
        k=k+1  
Return false
```

ומה עם המערך לא ממוין?

מוטיבציה

אתם פונים למדור מיטב ומזדהים לפי מספר תעודת זהות. במערכת של צה"ל רשומים מיליוני אנשים, אבל אתם לחוצים לדעת את תוצאות המיונים ברגע זה.

נרצה להשתמש במבנה נתונים התומך בפעולות `Remove`, `Insert`, `Search`.

מוטיבציה

אתם פונים למדור מיטב ומזדהים לפי מספר תעודת זהות. במערכת של צה"ל רשומים מיליוני אנשים, אבל אתם לחוצים לדעת את תוצאות המיונים ברגע זה.

נרצה להשתמש במבנה נתונים התומך בפעולות `Remove, Insert, Search`.

חיפוש במערך או ברשימה יקח $O(n)$.

ניתן לחפש בעזרת עץ חיפוש בינארי (מאוזן) אך עדיין לא מספיק טוב לנו...

נרצה יעילות של $O(1)$!

מילון

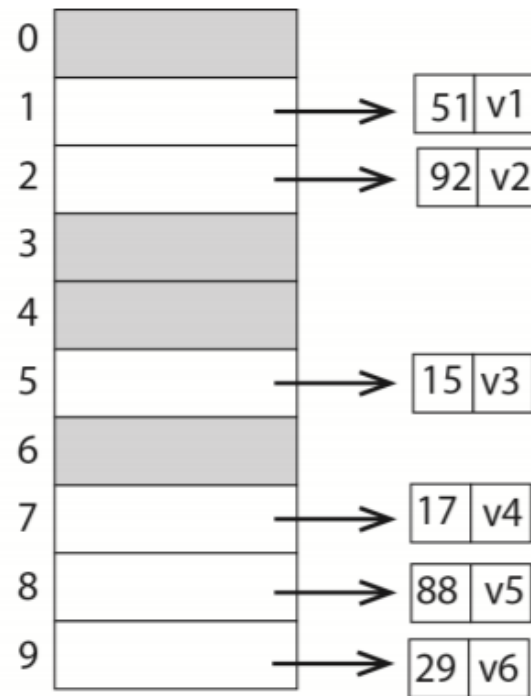
הפעולות הבסיסיות של מילון הן כזכור חיפוש, הכנסה, והוצאה. אם ידוע כי המפתחות מספרים שלמים בתחום $[0, \dots, m - 1]$, ניתן לממש את שלושת הפעולות בזמן $O(1)$ במקרה הגרוע, באמצעות שימוש במערך בגודל m .

בעיה: כאשר טווח המפתחות האפשריים גדול בהרבה ממספר המפתחות בהם משתמשים בפועל ואז יש בזבוז של מקום. לדוגמא – יש 10^9 מספרי זהות אפשריים, ופחות מ- 10^7 תושבים בישראל.

פיתרון אפשרי – אפשר לממש מילון באמצעות עץ AVL, ונקבל שהפעולות לוקחות זמן של $O(\log n)$ כאשר n הוא מספר המפתחות בעץ.

נגדיר טבלה בגודל m וניצור פונקציית ערבול שמחשבת אינדקס בטבלה מתוך המפתח עצמו :

$$h: U \rightarrow \{0, \dots, m - 1\}$$



כאשר U הוא תחום המפתחות האפשריים.

← נרצה שבממוצע הפעולות יתבצעו ב- $O(1)$

דוגמא: $m=10, h(k) = k \bmod 10$

קלטים:

$$51 \rightarrow 51 \bmod 10 = 1$$

$$17 \rightarrow 7$$

$$15 \rightarrow 5$$

$$92 \rightarrow 2$$

$$88 \rightarrow 8$$

$$29 \rightarrow 9$$

איזה בעיה יכולה להיווצר?

איזה בעיה יכולה להיווצר?

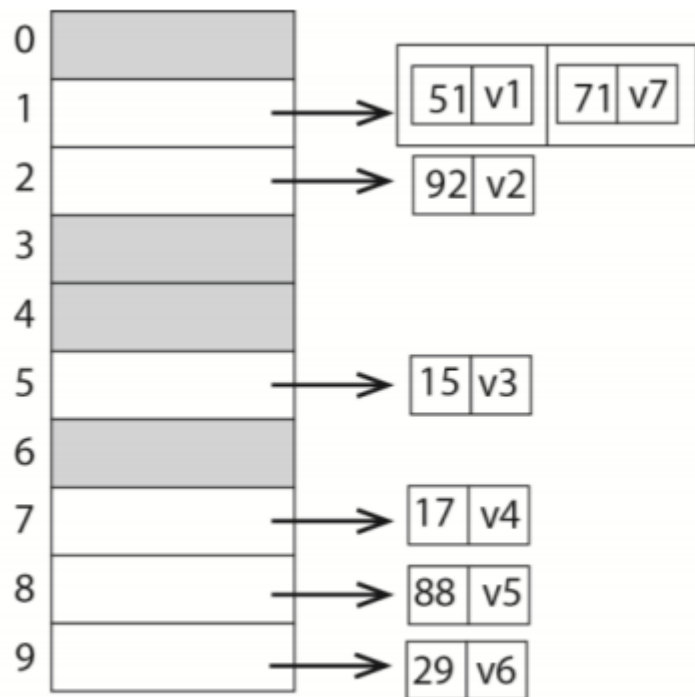
התנגשות- כאשר $x_1 \neq x_2$ אך $h(x_1) = h(x_2)$.

איזה בעיה יכולה להיווצר?

התנגשות- כאשר $x_1 \neq x_2$ אך $h(x_1) = h(x_2)$.

הצעת פתרון: שרשור

יוצרים רשימה מקושרת מכל האיברים המעורבלים לאותו תא.
הצעת שיפור: ניתן להשתמש בעץ מאוזן במקום ברשימה מקושרת.



מימוש פעולות:

Insert- נכניס את האיבר החדש לראש הרשימה המקושרת

Search- מחפשים את האיבר ברשימה המקושרת

Delete- מחפשים את האיבר ומוחקים אותו

סיבוכיות הפעולות: מניחים שפונקציית הערבול מפזרת את

המפתחות בצורה אחידה. כלומר הסיכוי שאיבר מסוים יגובב לתא מסוים שווה לכל m התאים.

תזכורת: מקדם עומס


בהינתן טבלה בגודל m וקבוצת מפתחות S בגודל n נסמן $\alpha = \frac{n}{m}$ **מקדם העומס** של הטבלה – מספר האיברים הממוצע בתא.

← זמן חיפוש כושל בממוצע: $O(1+\alpha)$.

הוכחה: בהנחה שהערבול אחיד ופשוט, ההסתברות שמפתח k יגובב לתא מסוים שווה לכל m התאים. לכן הזמן הממוצע לחיפוש כושל הוא סריקה של הרשימה המקושרת המתאימה עד סופה. האורך הממוצע של רשימה הוא מקדם העומס $\alpha = \frac{n}{m}$. לכן תוחלת מספר האיברים הנבדקים היא α . לכן הזמן הנדרש לחיפוש כושל (כולל חישוב $h(k)$) הוא $O(1+\alpha)$.

← זמן חיפוש מצליח בממוצע: $O(1+\alpha)$.

המשמעות: אם מספר התאים הוא לפחות ביחס ישר למספר האיברים בטבלה אזי $n=O(m)$ ומכאן נקבל $\alpha = n/m = O(m)/m = O(1)$. לכן חיפוש אורך בזמן קבוע.



בעיות בשיטה

- גישה לא רצופה בזיכרון – בגלל השימוש ברשימות מקושרות.
- עדיין ישנם תאים בהם זמן החיפוש יהיה גדול יותר.

פתרון שני-REHASH

הרעיון – כל האיברים נמצאים בטבלה עצמה. כאשר מכניסים איבר חדש, בודקים את התאים בזה אחר זה עד שמוצאים מקום ריק. באותו אופן על מנת לחפש איבר, בוחנים בשיטתיות תאים בטבלה עד שמוצאים את האיבר המבוקש או שמגיעים למסקנה שהוא לא נמצא.

במקום לחפש איבר בטבלה לפי הסדר הקבוע $0,1,2,\dots,m$ (שייקח $O(n)$), ניצור i פונקציות גיבוב חוזר $(Rehash)$ $h_i(x)$, כך שאם $h(x_1) = h(x_2)$ ו- x_1 כבר נמצא בטבלה, נלך ל- $h_1(x_2)$, ואז ל- $h_2(x_2)$ וכו', עד שנמצא מקום פנוי.

דוגמאות לפונקציות *Rehash* בהניתן פונקציית ערבול $h(x)$:

Double hashing: $h_i(x) = (h(x) + i \cdot r(x)) \bmod m$ ($r(x)$ היא פונקציית צעד)

Linear probing: $h_i(x) = (h(x) + i) \bmod m$

Quadratic probing: $h_i(x) = (h(x) + i^2) \bmod m$

← זמן חיפוש מצליח/כושל בממוצע: $O(\frac{1}{1-\alpha})$ (ללא הוכחה).

ערבול קוקיה (Cuckoo): יש 2 פוקנציות ערבול h_1, h_2 שמקיימות $h_1(x) \neq h_2(x)$.

אם $h_1(x) = h_1(y)$ ו- $h_2(x) = h_2(y)$ אז $x = y$.

הרעיון: בודקים אם $h_1(x)$ פנוי. אם תפוס נבדוק את $h_2(x)$. אם גם תפוס, נוציא את האיבר שנמצא

ב- $h_2(x)$ (נקרא לו y) ובמקומו נשים את x . באותה שיטה נחפש מקום ל- y .

עלולים להיכנס ללולאה, ולכן נשמור בנפרד מערך של ערכים שנפלטו מלולאות. הוא יהיה קטן ולכן החיפוש בו זל.

ערבול אוניברסלי

לכל בחירה של פונקציית ערבול קיימת סדרה גרועה של מפתחות כך שתיווצר רשימה באורך מקסימלי. הפיתרון: בזמן יצירת טבלת ערבול, לבחור באקראי פונקציית ערבול מתוך קבוצה 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 $H: U \rightarrow \{0, \dots, m-1\}$. הקבוצה H נקראת אוניברסלית אם

לכל זוג מפתחות שונים $x, y \in U$, מספר הפונקציות עבורן $h(x) = h(y)$ הוא $\frac{|H|}{m}$.

← ההסתברות שבבחירה אקראית של h , x יתנגש עם y היא $p = \frac{1}{|H|} \left(\frac{|H|}{m} \right) = \frac{1}{m}$.

תרגיל:

נתונה טבלת $hash$ בגודל $m=11$ ו-2 פונקציות $hash$:

$$h1(x) = (\text{value of the last letter of } x) \bmod m$$

$$h2(x) = (\text{sum of the values of the first and last letters of } x) \bmod (m-1) + 1$$

כאשר הערך של אות הוא מיקומה באלפבית ($a=1, b=2, \dots$)

	bread	milk	apple	pasta	water	salad	steak	rice	chips	chocolate
h1	4	0	5	1	7	4	0	5	8	5
h2	7	5	7	8	2	4	1	4	3	9

א) ציירו את טבלת ה $hash$ – לאחר הכנסת כל אחת מהמילים בכל אחת מהשיטות הבאות:

a. *Chaining with $h1$ as your hash function*

b. *Linear probing with $h1$ as your hash function*

c. *Rehashing with $h1$ as your first hash function, and $h2$ as a step function*

ב) למה אי אפשר להשתמש ב- $h1$ כפונקצית צעד?

ג) למה אי אפשר להשתמש ב- $h2$ כפונקציה ראשונה?

	bread	milk	apple	pasta	water	salad	steak	rice	chips	chocolate
h1	4	0	5	1	7	4	0	5	8	5
h2	7	5	7	8	2	4	1	4	3	9

(x

	a	b	c
0	milk, steak	Milk	Milk
1	pasta	Pasta	Pasta
2		Steak	Steak
3			chips
4	bread, salad	Bread	Bread
5	apple, rice, chocolate	Apple	Apple
6		Salad	
7	water	Water	Water
8	chips	Rice	Salad
9		Chips	Rice
10		chocolate	chocolate



ב) כי $h1$ יכולה להחזיר 0, ואז לא נבדוק מקום אחר כאשר נעשה *rehash*.

ג) כי $h2$ יכולה "לפספס" ערכים מסוים בטבלה, למשל 0.

שאלה

נתון מערך **לא ממוין** של מספרים $V = x_1, x_2 \dots x_n$.

כתוב אלוגריתם המקבל כקלט מספר y ומחזיר האם קיים זוג איברים בקבוצה $x_i, x_j \in V$ כך ש $x_i + x_j = y$.

פתרון:

א. נכניס את האיברים לטבלת ערבול.

ב. נעבור על האיברים במערך ונבדוק, האם $y - x_i$ נמצא בטבלת ערבול? אם כן, נחזיר את הפתרון, אם לא, נמשיך הלאה עד שנמצא או עד שנגיע לסוף המערך.

הזמן ריצה: $O(n)$.