טבלת גיבוב

Hash Table

1

- מה נלמד?
  - מבוא
  - פעולות בהן תומכת טבלת גיבוב
  - דוגמאות לשימוש בטבלת גיבוב
  - גיבוב עם שרשור
  - גיבוב פתוח
  - בחירת פונקציית גיבוב טובה

2

מבוא

- באפליקציות רבות עלינו לתחזק קבוצה *דינאמית* S לאחסון איברים, כך שנוכל לאתר אותם בעזרת מפתח *"יחודי"* שמצורף לכל איבר

x | key | Satellite data |

קבוצת עובדים במכללה
המפתח - ת"ז של עובד

קבוצת המשתנים בקוד
המפתח - שם המשתנה

```
public static void insertionSort(int[] arr) {

    for (int i = 1; i < arr.length; i++) {
        int key = arr[i];
        int j = i-1;

        while (j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}
```

קבוצת אתרים זדוניים
המפתחות - כתובת האתר

IP Addresses
103.236.162.56
162.252.172.41
180.181.68.221
203.134.40.41

3

מילון

Dictionary תומך בשלוש פעולות עיקריות

- $Search(k)$ – בהינתן מפתח k, הפעולה מחזירה איבר x המקיים $x.key = k$ או $NULL$ אם אין איבר כזה
- $Insert(x)$ – הכנסת לקבוצה את איבר x
- $Delete(x)$– בהינתן איבר x, יש למחוק אותו מהקבוצה

4

טבלת גיבוב

הבטחה על זמני ריצה

- $Search(k)$
- $Insert(x)$
- $Delete(x)$

$O(1)$ *

*
1. בממוצע
2. כאשר ממומש נכון

5

טבלת גיבוב

דוגמאות לשימוש

IP 103.212.162.56

בהינתן כתובת IP לבדוק האם האתר נמצא ברשימה שחורה

var

בהינתן שם המשתנה, לאתר את הערך והטיפוס שלו

בהינתן מספר הטלפון, לשלוף את הכתובת (הזמנות פיצה)

6

טבלת גיבוב
דוגמא לשימוש

נתון: שתי קבוצות של מספרים שלמים S_1 ו- S_2 בגודל n.
המטרה: לבדוק האם $S_1 = S_2$.

פתרונות

1. **נאיבי:** $O(n^2)$
   - עבור כל איבר מ-S_2 בדוק האם הוא נמצא ב-S_1.
2. **באמצעות מיון:** $O(n \log n)$
   - מיין את S_1.
   - עבור כל איבר מ-S_2 בדוק האם הוא נמצא ב-S_1 (חיפוש בינארי)
3. **שימוש בטבלת גיבוב:** $O(n)$
   - הכנס את איברי הקבוצה S_1 לטבלת גיבוב H.
   - עבור כל איבר מ-S_2 בדוק האם הוא נמצא ב-H

7

המטרה

- $Search(k)$
- $Insert(x)$
- $Delete(x)$

הנחה: המפתחות נלקחו מתוך קבוצה אוניברסאלית $U = \{0,1, \ldots, u\}$

8

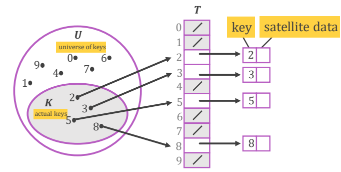
טבלת מיעון ישיר
Direct address table

- נתחיל מטכניקה פשוטה שתוביל אותנו לרעיון של טבלת גיבוב
- טבלת מיעון ישיר היא טכניקה מאוד פשוטה לייצוג קבוצה דינאמית של איברים

9

טבלת מיעון-ישיר

- לייצוג קבוצה S נשתמש במערך $T[0 \,..\, |U|-1]$
- לכל תא ב-T מתאים מפתח מ-U
  - $T[i] = x$ אם $x \in S$ ו- $x.key = i$
  - אחרת $T[i] = NULL$



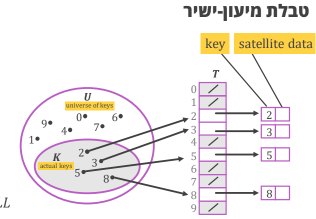
10

טבלת מיעון-ישיר

***Search*(k)**
return $T[k]$
$O(1)$

***Insert*(x)**
$T[x.key] \leftarrow x$
$O(1)$

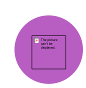
***Delete*(x)**
$T[x.key] \leftarrow NULL$
$O(1)$



? מה יכולה להיות הבעיה?

11

טבלת מיעון-ישיר
הבעיה

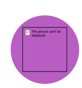


קבוצת עובדים באוניברסיטה

10^9 אפשרויות לתעודת זהות

קבוצת אתרים זדוניים

כתובת IP מורכבת מ-32 ביט ← 2^{32} אפשרויות



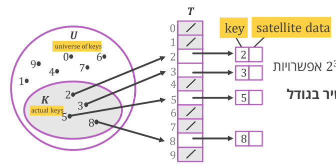
אם U גדולה, לאחסן טבלת T בגודל $|U|$ לא מעשי (או אפילו לא אפשרי)

12

טבלת מיעון-ישיר
הבעיה

מטרה: לתחזק קבוצה *דינאמית* S לאחסון איברים, כך שנוכל לאתר אותם בעזרת מפתח *ייחודי* שמצורף לכל איבר.

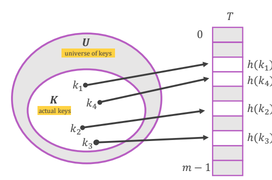
- קבוצת עובדים באוניברסיטה
  - 10^9 אפשרויות לתעודת זהות
- קבוצת אתרים זדוניים
  - כתובת IP מורכבת מ-32 ביט ← 2^{32} אפשרויות
- אם U גדולה, לאחסן טבלת מיעון-ישיר בגודל $|U|$ לא מעשי (או אפילו לא אפשרי)



13

טבלת גיבוב
הרעיון

- ליצוג קבוצה S נשתמש במערך $T[0..m-1]$, $m = \Theta(|S|)$
- לכל $x \in S$ נתאים מספר בין 0 ל- $m-1$
- מיפוי כזה נקרא **גיבוב (hashing)**
- פונקציית גיבוב (hash function) $h: U \to \{0,1,\ldots,m-1\}$
- איבר בעל מפתח k מגובב (hashes) לתא $h(k)$

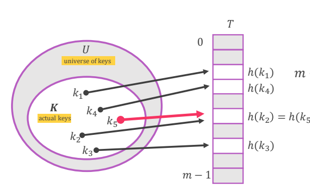


מה יכול להיות הבעיה?

14

טבלת מיעון-ישיר
הרעיון

- ליצוג קבוצה S נשתמש במערך $T[0..m-1]$, $m = \Theta(|S|)$
- לכל $x \in S$ נתאים מספר בין 0 ל- $m-1$
- מיפוי כזה נקרא **גיבוב (hashing)**
- פונקציית גיבוב (hash function) $h: U \to \{0,1,\ldots,m-1\}$
- איבר בעל מפתח k מגובב (hashes) לתא $h(k)$



מה יכול להיות הבעיה?

התנגשויות (collisions)

15

התנגשויות

התנגשות: עבור מפתחות <u>שונים</u> $x,y \in U$: $h(x) = h(y)$

16

נכון או לא נכון:

טענה: לא ניתן למנוע לחלוטין את ההתנגשויות.

1. הטענה נכונה
2. הטענה לא נכונה

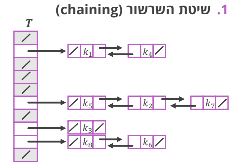
17

התנגשויות

התנגשות: עבור מפתחות <u>שונים</u> $x,y \in U$: $h(x) = h(y)$

פתרונות

1. שיטת השרשור (chaining)



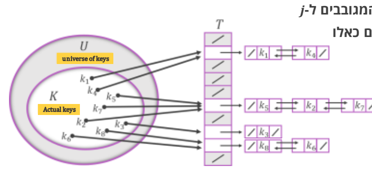
2. מיעון פתוח (open addressing)
   - רק איבר אחד בכל תא או $NULL$
   - בהכנסה בוחנים בזה אחר זה תאים עד שמוצאים תא ריק
   - פונקציית גיבוב מגדירה סדרה של m אינדקסים $h(k,0),h(k,1),\dots,h(k,m-1)$

18

שיטת השרשור

התנגשות: עבור מפתחות <u>שונים</u> $x,y \in U$: $h(x) = h(y)$

- **שיטת השרשור (chaining)**
- התא j מכיל מצביע לראש הרשימה של כל האיברים המאוחסנים המגובבים ל-j
- או $NULL$ אם אין איברים כאלו



19

מימוש הפעולות

Insert (T, x)
Insert x at the head of the list $T[\ h(x.key)]$

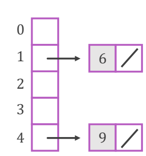
Search (T, k)
Search for an element with key k in list $T[h(k)]$

Delete (T, x)
Delete x from the list $T[h(x.key)]$

20

דוגמה

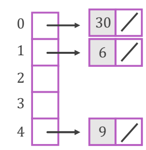
יש להכניס את המפתחות 6,9,19,26,30,106,309 **לטבלת גיבוב בגודל** $m = 5$
באמצעות שיטת השרשור עם פונקצית גיבוב $h(k) = k \bmod 5$



21

דוגמה

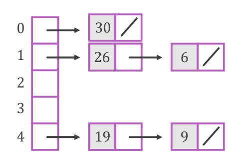
יש להכניס את המפתחות 6,9,19,26,30,106,309 לטבלת גיבוב בגודל $m = 5$
באמצעות שיטת השרשור עם פונקצית גיבוב $h(k) = k \bmod 5$



22

דוגמה

יש להכניס את המפתחות 6,9,19,26,30,106,309 לטבלת גיבוב בגודל $m = 5$
באמצעות שיטת השרשור עם פונקצית גיבוב $h(k) = k \bmod 5$



23

? **נתונה טבלת גיבוב T שבה התנגשויות נפתרות על ידי שרשור.**
נסמן ב- m את גודל הטבלה וב- n את מספר האיברים בטבלה.

השלימו את המשפט: זמן ריצה במקרה הגרוע של פעולת $Insert$ הוא:

1.	2.	3.	4.
$\Theta(n)$	$\Theta(m)$	$\Theta(1)$	נמצא ביחס ישיר לאורכה של הרשימה שבתא

24

! **נתונה טבלת גיבוב T שבה התנגשויות נפתרות על ידי שרשור.**
נסמן ב- m את גודל הטבלה וב- n את מספר האיברים בטבלה.

השלימו את המשפט: זמן ריצה במקרה הגרוע של פעולת $Insert$ הוא:

.1	.2	.3	.4
$\Theta(n)$	$\Theta(m)$	$\Theta(1)$	נמצא ביחס ישיר לאורכה של הרשימה שבתא

25

? **נתונה טבלת גיבוב T שבה התנגשויות נפתרות על ידי שרשור.**
נסמן ב- m את גודל הטבלה וב- n את מספר האיברים בטבלה.

השלימו את המשפט: זמן ריצה במקרה הגרוע של פעולת $Search$ הוא:

.1	.2	.3	.4
$\Theta(n)$	$\Theta(m)$	$\Theta(1)$	נמצא ביחס ישיר לאורכה של הרשימה שבתא

26

? **נתונה טבלת גיבוב T שבה התנגשויות נפתרות על ידי שרשור.**
נסמן ב- m את גודל הטבלה וב- n את מספר האיברים בטבלה.

השלימו את המשפט: זמן ריצה במקרה הגרוע של פעולת $Search$ הוא:

.1	.2	.3	.4
$\Theta(n)$	$\Theta(m)$	$\Theta(1)$	נמצא ביחס ישיר לאורכה של הרשימה שבתא

27

נתונה טבלת גיבוב T שבה התנגשויות נפתרות על ידי שרשור.
נסמן ב- m את גודל הטבלה וב- n את מספר האיברים בטבלה.

השלימו את המשפט: זמן ריצה במקרה הגרוע של פעולת $Delete$ הוא:

.1	.2	.3	.4
$\Theta(n)$	$\Theta(m)$	$\Theta(1)$	נמצא ביחס ישיר לאורכה של הרשימה שבתא

28

נתונה טבלת גיבוב T שבה התנגשויות נפתרות על ידי שרשור.
נסמן ב- m את גודל הטבלה וב- n את מספר האיברים בטבלה.

השלימו את המשפט: זמן ריצה במקרה הגרוע של פעולת $Delete$ הוא:

.1	.2	.3	.4
$\Theta(n)$	$\Theta(m)$	$\Theta(1)$	נמצא ביחס ישיר לאורכה של הרשימה שבתא

29

מימוש הפעולות

$\boldsymbol{Insert}\ (T, x)$
Insert x at the head of the list $T[\ h(x.key)]$ — $\Theta(1)$

$\boldsymbol{Search}\ (T, k)$
Search for an element with key k in list $T[h(k)]$ — $\Theta(list\ length)$

$\boldsymbol{Delete}\ (T, x)$
Delete x from the list $T[h(x.key)]$ — $\Theta(list\ length)$

אם m הוא גודל טבלת הגיבוב ו-n הוא מספר האיברים בטבלה,
אורך הרשימה יכול להיות כל מספר בין $\frac{n}{m}$ ל- n

30

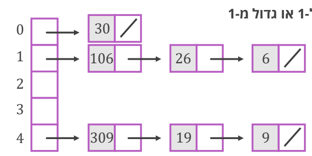
ניתוח של גיבוב עם שרשור

- הנחות
  1. פונקציית גיבוב צריכה לקיים את הנחת הגיבוב האחיד הפשוט:
  ההסתברות שמפתח כלשהו יגובב לתא מסוים שווה עבור כל m התאים, ואינה תלויה בערכי הגיבוב של האיברים האחרים.
  2. הזמן הדרוש לחישוב פונקציית גיבוב הוא $\Theta(1)$
     - פעולות חיפוש, הכנסה, מחיקה כוללות חישוב של פונקציית גיבוב

31

ניתוח של גיבוב עם שרשור

- נגדיר $\alpha = \frac{n}{m}$ מקדם העומס (load factor) – המספר הממוצע של איברים המאוחסנים ברשימה מקושרת אחת
  - α יכול להיות קטן מ-1, שווה ל-1 או גדול מ-1



$\alpha = \frac{7}{5} = 1.4$

32

ניתוח של גיבוב עם שרשור

- ננתח זמן ריצה של חיפוש כושל
  - זמן ריצה של חיפוש מוצלח ומחיקה חסום על ידי זמן ריצה של חיפוש כושל

33

ניתוח של גיבוב עם שרשור

משפט: בטבלת גיבוב שבה התנגשויות נפתרות על ידי שרשור, זמן ריצה הצפוי של חיפוש כושל הוא $\Theta(1+\alpha)$

- **נניח מחפשים איבר x שלא נמצא בטבלת הגיבוב**
- **חישוב פונקציית גיבוב $h(k)$** - $\Theta(1)$
- **מעבר על הרשימה שבתא $T[h(k)]$** - $\Theta(list\ length)$
- **תחת ההנחה של גיבוב אחיד ופשוט, תוחלת אורך הרשימה שבתא היא** α
- **מכאן, זמן ריצה של חיפוש כושל הוא $\Theta(1+\alpha)$**

34

ניתוח של גיבוב עם שרשור

משפט: בטבלת גיבוב שבה התנגשויות נפתרות על ידי שרשור, זמן ריצה הצפוי של חיפוש כושל הוא $\Theta(1+\alpha)$

- **אם $\alpha=\Theta(1)$ נקבל שזמן הריצה של חיפוש כושל הוא $\Theta(1)$**
- **תחת איזה תנאי $\alpha=\Theta(1)$?**
  - **$\alpha=\frac{n}{m}$, m הוא גודל טבלת הגיבוב ו-n הוא מספר האיברים בטבלה**
  - אם $n=\Theta(m)$ נקבל $\alpha=\Theta(1)$

35

מימוש הפעולות

$\boldsymbol{Insert}\ (\boldsymbol{T},\boldsymbol{x})$
Insert x at the head of the list $T[\ h(x.key)]$ — $\Theta(1)$

$\boldsymbol{Search}\ (\boldsymbol{T},\boldsymbol{k})$
Search for an element with key k in list $T[h(k)]$ — $\Theta(list\ length)$

$\boldsymbol{Delete}\ (\boldsymbol{T},\boldsymbol{x})$
Delete x from the list $T[h(x.key)]$ — $\Theta(list\ length)$

אם m הוא גודל טבלת הגיבוב ו-n הוא מספר האיברים בטבלה,
אורך הרשימה יכול להיות כל מספר בין $\frac{n}{m}$ ל- n

36

מהי פונקציית גיבוב טובה?

1. ביצועים טובים
   - פונקציית גיבוב צריכה לקיים את הנחת הגיבוב האחיד הפשוט: ההסתברות שמפתח כלשהו יגובב לתא מסוים שווה עבור כל m התאים.
2. חישוב מהיר $\Theta(1)$
   - פעולות חיפוש, הכנסה, מחיקה כוללות חישוב של פונקציית גיבוב

37

פונקציית גיבוב גרועות

דוגמא 1

- מפתחות: מספרי טלפון (הזמנת פיצה ביישוב קטן)
- מספר נייד מכיל 10 ספרות, $|U| = 10^{10}$
- $m = 1000$
- פונקציית גיבוב מאוד גרוע:

$$h(k) = first\ 3\ digits\ of\ k$$

38

פונקציית גיבוב גרועות

IP Addresses
103.236.162.56
162.252.172.41
180.181.68.221
203.134.40.41

דוגמא 2

- מפתחות: כתובות IP (רשימה שחורה של אתרים)
- כתובת IP מורכבת מ-32 ביטים, $|U| = 2^{32}$
- פונקציית גיבוב גרוע:

$$h(k) = last\ segment\ of\ the\ address\ (last\ 8\ bits)$$

$$h(103.256.162.56) = 56$$

- $m = 256$
- החלק האחרון של כתובת IP בדרך כלל מייצגת מספר קטן (חד או דו-ספרתי)

39

פונקציות גיבוב גרועות

דוגמא 3

- S היא קבוצת מספרים שלמים זוגיים
- $m = 1000$
- פונקציית גיבוב גרוע: $h(k) = |k| \; mod \; 1000$
- מובטח שתאים בעלי אינדקס אי-זוגי יהיו ריקים

22 4 2 46 30 94

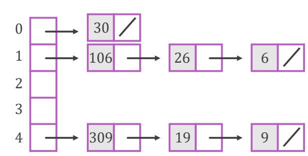
40

שיטת החילוק (Division method)

$h(k) = k \; mod \; m$
m – גודל הטבלה

דוגמה

- $S = \{6,9,19,26,30,106,309\}$
- $h(k) \; = \; k \; mod \; 5$



41

שיטת החילוק (Division method)

$h(k) = k \; mod \; m$
m – גודל הטבלה

- פשוט
- מהיר
- "להיזהר" מערכים מסוימים של m
  - לבחור את m להיות ראשוני

42

שיטת הכפל (Multiplication method)

$h(k) = \lfloor m(kA \bmod 1) \rfloor$
m – גודל הטבלה
$0 < A < 1$ קבוע

$13456.09 \bmod 1 = 0.09$
$57.891 \bmod 1 = 0.891$

43

שיטת הכפל (Multiplication method)

$h(k) = \lfloor m(kA \bmod 1) \rfloor$
m – גודל הטבלה
$0 < A < 1$ קבוע

דוגמה

$k = 123456$
$m = 2^{14} = 16384$
$A = (\sqrt{5}-1)/2 \approx 0.618$

$$\begin{aligned} h(123456) &= \lfloor 16384(123456 \cdot 0.618 \bmod 1) \rfloor = \\ &= \lfloor 16384(76295.808 \bmod 1) \rfloor = \\ &= \lfloor 16384 \cdot 0.808 \rfloor = \\ &= \lfloor 13238.272 \rfloor = 13238 \end{aligned}$$

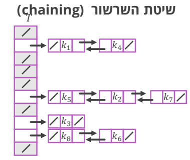
44

התנגשויות

התנגשות: עבור מפתחות <u>שונים</u> $h(x) = h(y) : x,y \in U$

פתרונות

1. שיטת השרשור (chaining)



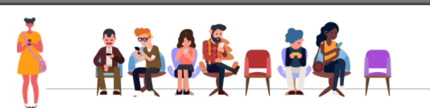
2. מיעון פתוח (open addressing)
- רק איבר אחד בכל תא או $NULL$
- פונקציית גיבוב מגדירה סדרה של m אינדקסים $h(k,0),h(k,1),\dots,h(k,m-1)$
- בהכנסה בוחנים בזה אחר זה תאים עד שמוצאים תא ריק
- מקדם העומס $\alpha \le 1$

45

46

הכנסה דוגמא

בזמן ההכנסה, בוחנים בזה אחר זה תאים עד שמוצאים תא ריק



47

הכנסה דוגמא

בזמן ההכנסה, בוחנים בזה אחר זה תאים עד שמוצאים תא ריק

48

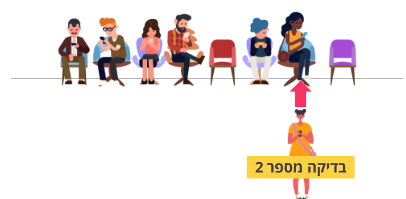
הכנסה דוגמא

בזמן ההכנסה, בוחנים בזה אחר זה תאים עד שמוצאים תא ריק

49

הכנסה דוגמא

בזמן ההכנסה, בוחנים בזה אחר זה תאים עד שמוצאים תא ריק



50

הכנסה דוגמא

בזמן ההכנסה, בוחנים בזה אחר זה תאים עד שמוצאים תא ריק

51

שיטת המיעון הפתוח

$$h: \underbrace{U}_{\text{Universe}} \times \underbrace{\{0,1,\dots,m-1\}}_{\text{מספר הבדיקה (probe number)}} \rightarrow \underbrace{\{0,1,\dots,m-1\}}_{\text{index}}$$

- $h(k,i)$ תא שנגיע אליו בבדיקה ה- i-ית (בהנחה שבבדיקות הקודמות הגענו לתאים תפוסים)
- אנו דורשים שסדרת התאים הנבדקים $h(k,0), h(k,1), \dots, h(k,m-1)$ תהיה תמורה של $\{0,1,\dots,m-1\}$, כך שבסופו של דבר נבחן את כל תאי הטבלה

52

הכנסה וחיפוש בשיטת המיעון הפתוח

```
Insert (T, x)
i ← 0
while (i < m)
    j ← h(x.key, i)
    if T[j] = NULL
        T[j] ← x
        return j
    else i ← i + 1
error "hash table overflow"
```

```
Search (T, k)
i ← 0
j ← h(k, i)
while (i < m AND T[j] ≠ NULL)
    if T[j].key = k
        return T[j]
    i ← i + 1
    j ← h(k, i)
return NULL
```

53

מחיקות בשיטת המיון הפתוח

- **לא ניתן פשוט לסמן את התא כתא ריק על ידי הצבת NULL**
  - אם נעשה זאת לא נוכל לשלוף מפתח אשר במהלך הכנסתו נבדק תא זה ונמצא תפוס
- **פתרון אפשרי: לסמן את התא על-ידי הצבת הערך המיוחד DELETED**
  - בהכנסה להתייחס לתא כזה כאל תא ריק
  - בחיפוש לעבור על התא מבלי לעצור
  - חסרון: זמני ריצה אינם תלויים עוד במקדם העומס α
- **כאשר יש למחוק איברים, בוחרים בדרך כלל בשיטת השרשור**

54

מיעון פתוח – איך מחפשים תא פנוי?

- בדיקה לינארית (linear probing)
- גיבוב כפול (double hashing)

55

שיטת המיעון הפתוח

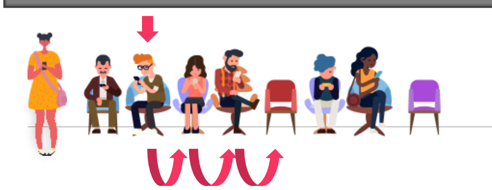
$$h: \underbrace{U}_{\text{Universe}} \times \underbrace{\{0,1,\dots,m-1\}}_{\text{מספר הבדיקה (probe number)}} \to \underbrace{\{0,1,\dots,m-1\}}_{\text{index}}$$

- $h(k,i)$ תא שנגיע אליו בבדיקה ה- i-ית (בהנחה שבבדיקות הקודמות הגענו לתאים תפוסים)
- אנו דורשים שסדרת התאים הנבדקים $h(k,0), h(k,1), \dots, h(k,m-1)$ תהיה תמורה של $\{0, 1, \dots, m-1\}$ כך שבסופו של דבר נבחן את כל תאי הטבלה
- מקדם העומס $\alpha \le 1$

56

בדיקה לינארית Linear Probing

בזמן ההכנסה, אם התא תפוס נבדוק האם התא הבא בטבלה ריק וכך נמשיך עד למציאת מקום ריק.



57

בדיקה לינארית Linear Probing

בזמן ההכנסה, אם התא תפוס נבדוק האם התא הבא בטבלה ריק וכך נמשיך עד למציאת מקום ריק.

58

בדיקה לינארית
Linear Probing

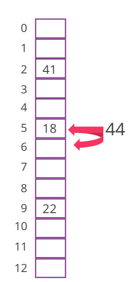
$h(k,i) = (h'(k) + i) \bmod m$

$h'(k)$ – פונקציית גיבוב רגילה

- $h'(k)$ אינדקס של התא הראשון שנבדק
- התאים נבדקים לפי סדר לינארי של אינדקסים (באופן מעגלי)
  $h'(k), h'(k)+1, h'(k)+2, \dots$

59

בדיקה לינארית
Linear Probing



דוגמה:

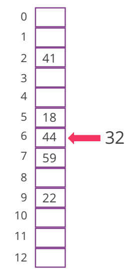
יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 לטבלת גיבוב בגודל $m = 13$ באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב $h'(k) = k \bmod 13$

60

בדיקה לינארית
Linear Probing

דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 לטבלת גיבוב בגודל $m = 13$ באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב $h'(k) = k \bmod 13$

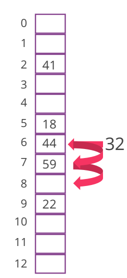


61

בדיקה לינארית
Linear Probing

דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 לטבלת גיבוב בגודל $m = 13$ באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב $h'(k) = k \bmod 13$

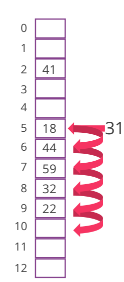


62

בדיקה לינארית
Linear Probing

דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 לטבלת גיבוב בגודל $m = 13$ באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב $h'(k) = k \bmod 13$

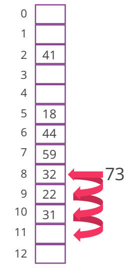


63

בדיקה לינארית
Linear Probing

דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב** $h'(k) = k \bmod 13$

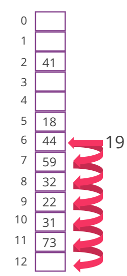


64

בדיקה לינארית
Linear Probing

דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב** $h'(k) = k \bmod 13$



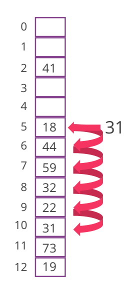
65

בדיקה לינארית
Linear Probing

דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הבדיקה הלינארית עם פונקציית גיבוב** $h'(k) = k \bmod 13$

$Search(31)$



66

בדיקה לינארית

- קל לממש
- סובלת מהצטברות ראשונית (primary clustering)
  - נוצרים רצפים ארוכים של תאים תפוסים, המאריכים את זמן החיפוש

67

גיבוב כפול
Double Hashing

$h(k,i) = (h_1(k) + ih_2(k)) \bmod m$

$h_1(k), h_2(k)$ – פונקצית עזר לגיבוב

- $h_1(k)$ אינדקס של התא הראשון שנבדק
- לאחר מכן נבדקים תאים שמיקומיהם רחוקים זה מזה במרחק של $h_2(k)$
- גם מיקום הבדיקה הראשונה וגם מרחק בין התאים תלויים במפתח k

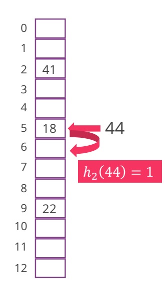
68

גיבוב כפול

דוגמה:

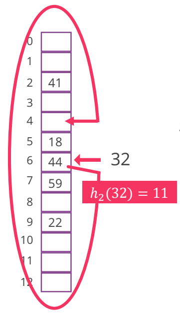
יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הגיבוב הכפול עם**

פונקציות גיבוב $h_1(k) = k \bmod 13$
$h_2(k) = 1 + k \bmod 11$



69

גיבוב כפול



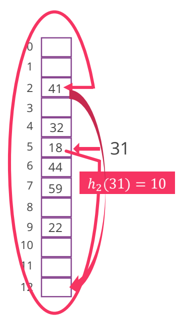
דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הגיבוב הכפול עם פונקציות גיבוב**

$h_1(k) = k \bmod 13$
$h_2(k) = 1 + k \bmod 11$

70

גיבוב כפול



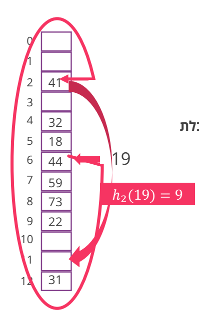
דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הגיבוב הכפול עם פונקציות גיבוב**

$h_1(k) = k \bmod 13$
$h_2(k) = 1 + k \bmod 11$

71

גיבוב כפול



דוגמה:

יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת גיבוב בגודל** $m = 13$ **באמצעות שיטת הגיבוב הכפול עם פונקציות גיבוב**

$h_1(k) = k \bmod 13$
$h_2(k) = 1 + k \bmod 11$

72

גיבוב כפול

דוגמה:

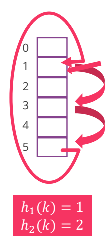
יש להכניס את המפתחות 18,41,22,44,59,32,31,73,19 **לטבלת**
גיבוב בגודל $m = 13$ **באמצעות שיטת הגיבוב הכפול עם**
פונקציות גיבוב $h_1(k) = k \bmod 13$
$h_2(k) = 1 + k \bmod 11$

0	
1	
2	41
3	
4	32
5	18
6	44
7	59
8	73
9	22
10	
11	19
12	31

73

גיבוב כפול

- עבור שני מפתחות שמתחילים בדיקה מאותו תא, הצעד יכול להיות שונה
- יש להבטיח שחיפוש יסרוק את טבלת הגיבוב כולה
  - הערך $h_2(k)$ חייב להיות זר לגודל הטבלה של m
    - $m = 2^p$ ולבנות את h_2 כך שתמיד תפיק מספר אי-זוגי
    - m ראשוני ולבנות את h_2 כך שתמיד תפיק מספר שלם חיובי קטן מ-m



74

ניתוח של שיטת הגיבוב הפתוח

- **הנחה: גיבוב אחיד (uniform hashing)**
  בהינתן מפתח כלשהו k**, ההסתברות של כל אחת מ-**$m!$ **התמורות של** $\{0,1,..,m-1\}$ **להיות סדרת הבדיקות ל-**k **היא** $\frac{1}{m!}$
- **בדיקה לינארית, בדיקה ריבועית -** $\Theta(m)$ **סדרות שונות**
- **גיבוב כפול -** $\Theta(m^2)$ **סדרות שונות**

75

ניתוח של שיטת הגיבוב הפתוח

משפט: תוחלת מספר הבדיקות הנערכות בעת הכנסת איבר היא $\frac{1}{1-\alpha}$, בהנחת הגיבוב האחיד.

- **אם** $\alpha = \frac{1}{2}$ **(טבלת הגיבוב חצי מלאה),** $\frac{1}{1-\alpha} = 2$ **(2 בדיקות)**
- **אם** $\alpha = 0.9$ **(טבלת הגיבוב** 90% **מלאה),** $\frac{1}{1-\alpha} = 10$ **(10 בדיקות)**
- **אם** α **מתקרב ל-1, מספר הבדיקות שואף לאינסוף**

76

ניתוח של שיטת הגיבוב הפתוח

משפט: תוחלת מספר הבדיקות הנערכות בעת הכנסת איבר היא $\frac{1}{1-\alpha}$, בהנחת הגיבוב האחיד.

- **הסבר אינטואיטיבי:**
  - **מספר הבדיקות עד לתא הפנוי הראשון ≈ התפלגות גיאומטרית עם הסתברות להצלחה** $1-\alpha$
  - **תוחלת מספר הבדיקות היא** $\frac{1}{1-\alpha}$

77

ניתוח זמני ריצה של טבלת גיבוב

- $\alpha = \frac{n}{m}$ **מקדם העומס (load factor)**
- $\alpha = O(1)$ **הוא תנאי הכרחי לביצוע פעולות בזמן קבוע**
  - $n = \Theta(m)$
- **פיזור אחיד**
  - **לבחור פונקציית גיבוב כך שעבור כל קבוצה של נתונים הפיזור יהיה אחיד**
    - **בעייה: לא קיימת פונקציית גיבוב שמקיימת תנאי זה**
    - **לפי עקרון שובך היונים קיים אינדקס i כך שלפחות $\frac{|U|}{m}$ מפתחות מתוך U יגובבו לתא i לכל פונקציית גיבוב h.**

78

בעיה

- לפי עקרון שובך היונים קיים אינדקס i כך שלפחות $\frac{|U|}{m}$ מפתחות מתוך U יגובבו לתא i לכל פונקציית גיבוב h.
- אם קבוצת המפתחות נבחרה מתוך $\frac{|U|}{m}$ מפתחות אלו, הם כולם ייכנסו לאותו תא
- זמני ריצה $\Theta(n)$
- מסקנה: יריב זדוני יכול לחבל בביצועים של המערכת

79

גיבוב אוניברסלי
רעיון

- בחירת פונקציית גיבוב באופן אקראי, בדרך שאינה תלויה במפתחות שיאוחסנו בטבלת הגיבוב
- הרעיון: לבחור פונקצית גיבוב באופן אקראי מתוך מחלקה $\mathcal{H}$ של פונקציות שתוכננה מראש.
- הבחירה האקראית מבטיחה שלא קיים קלט יחיד שעבורו התנהגות האלגוריתם היא תמיד הגרועה ביותר

80

גיבוב אוניברסלי
הגדרה

- תהי $\mathcal{H}$ קבוצה סופית של פונקציות גיבוב מ-U אל התחום $\{0,1,\ldots,m-1\}$

$\mathcal{H}$ תיקרא קבוצה אוניברסלית אם ורק אם עבור כל זוג מפתחות שונים $x,y\epsilon U$

$$\Pr\big(h(x)=h(y)\big)\leq\frac{1}{m}$$

- מכאן, ניתן לראות את בעייה הבחירת של פונקציה גיבוב טובה כבחירת קבוצה אוניברסלית $\mathcal{H}$ של פונקציות גיבוב.

81

דוגמא לקבוצה אוניברסלית של פונקצית גיבוב
גיבוב כתובת IP

- **כתובת IP מורכבת מ-**32 **ביט**
- **ניתן לראות כתובת IP כרביעיה** (x_1,x_2,x_3,x_4), x_i **מקבל ערכים בין** 0 **ל-**255
- **נבחר** m **להיות ראשוני**
- תהי $a = (a_1,a_2,a_3,a_4)$ **סדרה של מקדמים שנבחרו באופן אקראי מתוך** $\{0,1,\ldots,m-1\}$
- **נגדיר פונקציית גיבוב** $h_a = (a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4)$
- $\mathcal{H}=\{h_a | a_1,a_2,a_3,a_4 \in \{0,1,\ldots,m-1\}\}$ **היא קבוצה אוניברסלית**
  - m^4 פונקציות שונות
  - סיבוכיות חישוב של פונקציית גיבוב - O(1)
  - סיבוכיות מקום - O(1)

82

סיכום
טבלת גיבוב
Hash Table

83