אוניברסיטת
אריאל
בשומרון

מבני נתונים

תרגול 6 – Hash Table

צבי מינץ – zvimints@gmail.com

מוטבציה

נרצה לאחסן אובייקט במבנה נתונים אשר תומך בפעולות הבאות:

	במקרה הממוצע	במקרה הרע
זיכרון	$O(n)$	$O(n)$
הכנסה	$O(1)$	$O(n)$
חיפוש	$O(1)$	$O(n)$
מחיקה	$O(1)$	$O(n)$

בלתי תלות בכמות המידע שהמבנה מכיל

הצגת הבעיה

נניח כי יש לנו אוסף של n איברים

key	data

כאשר ערכי המפתח (key) שלהם הם **יחודיים**
בתחום $U := \{1,2,\dots,|U|\}$ כאשר $|U| > n$

אין שני איברים עם מפתח זהה.

U מרחב כל המפתחות האפשריים

Record Example

אוניברסיטת אריאל בשומרון

נניח כי נרצה לאחסן 1000 אנשים לפי תעודת זהות

ID	NAME	D.O.B
111111111	Zvi	25/10/1995
222222222	Dan	26/10/1995
...		
999999999	Ilan	27/12/1995

Input: ID

Output:

```
class Person {
    private long _id;
    private String name;
    private String _uni;
    private Date dob;
}
```

$$|U| = |\{0,999,999,999\}| = 10^9$$

U מרחב כל המפתחות האפשריים

חשבו על מבנה נתונים דינמי אשר תומך
בפעולות הכנסה, מחיקה וחיפוש ב-$O(1)$

Record Example

נניח כי נרצה לאחסן 1000 סטודנטים לפי תעודת זהות

ID	NAME	D.O.B
111111111	Zvi	25/10/1995
222222222	Dan	26/10/1995

...

999999999	Ilan	27/12/1995

	Add	Find	Delete	Space
Array	*O(1)*	*O(n)*	*O(n)*	*O(n)*

zvimints@gmail.com – צבי מינץ

Record Example

אוניברסיטת אריאל בשומרון

נניח כי נרצה לאחסן 1000 סטודנטים לפי תעודת זהות

ID	NAME	D.O.B
111111111	Zvi	25/10/1995
222222222	Dan	26/10/1995

...

999999999	Ilan	27/12/1995

	Add	Find	Delete	Space
Array	*O(1)*	*O(n)*	*O(n)*	*O(n)*
Sorted Array by Key	*O(n)*	*O(n)*	*O(logn)*	*O(n)*

צבי מינץ – zvimints@gmail.com

Record Example

נניח כי נרצה לאחסן 1000 סטודנטים לפי תעודת זהות

ID	NAME	D.O.B
111111111	Zvi	25/10/1995
222222222	Dan	26/10/1995

...

999999999	Ilan	27/12/1995

	Add	Find	Delete	Space
Array	*O(1)*	*O(n)*	*O(n)*	*O(n)*
Sorted Array by Key	*O(n)*	*O(n)*	*O(logn)*	*O(n)*
BBST	*O(logn)*	*O(logn)*	*O(logn)*	*O(n)*

Record Example

נניח כי נרצה לאחסן 1000 סטודנטים לפי תעודת זהות

ID	NAME	D.O.B
111111111	Zvi	25/10/1995
222222222	Dan	26/10/1995

...

999999999	Ilan	27/12/1995

	Add	Find	Delete	Space
Array	*O(1)*	*O(n)*	*O(n)*	*O(n)*
Sorted Array by Key	*O(n)*	*O(n)*	*O(logn)*	*O(n)*
BBST	*O(logn)*	*O(logn)*	*O(logn)*	*O(n)*
Direct Access Table		...		

zvimints@gmail.com – צבי מינץ

Direct Access Table

גישה אחת היא לאחסן את הערכים במערך
ענק בגודל $|U| = 10^9$

כאשר **האינדקס** במערך הוא **ת.ז של הסטודנט**

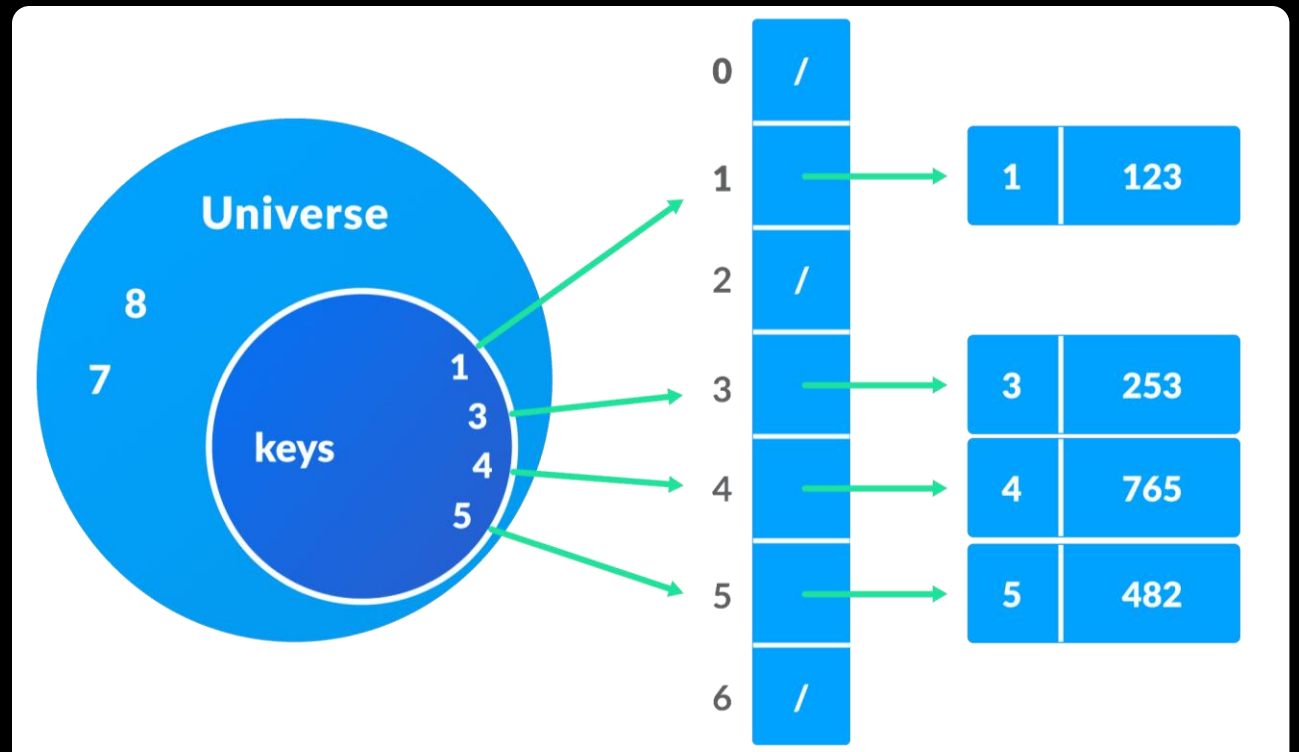
לדוגמא על מנת לאחסן סטודנט עם ת.ז 111
אז נכניס את הערך שלו למערך בתא 111 $(A[111])$

תרשים ←

Direct Access Table

גישה אחת היא לאחסן את הערכים במערך
ענק בגודל $|U| = 10^9$

כאשר **האינדקס** במערך הוא **ת.ז של הסטודנט**



https://www.programiz.com/dsa/hash-table

Direct Access Table

גישה אחת היא לאחסן את הערכים במערך
ענק בגודל $|U| = 10^9$

כאשר **האינדקס** במערך הוא **ת.ז של הסטודנט**

```
directAddressSearch(A, k)
    return A[k]

directAddressInsert(A, x)
    A[x.key] = x

directAddressDelete(A, x)
    A[x.key] = NIL
```

Direct Access Table

גישה אחת היא לאחסן את הערכים במערך
ענק בגודל $|U| = 10^9$

סיבוכיות

הוספה	O(1)
מחיקה	**O(1)**
חיפוש	**O(1)**
מקום	O(|U|)

Direct Access Table

אוניברסיטת
אריאל
בשומרון

הגבלות:

1. חוסר מידע קודם על ערך המפתח המקסמלי
2. תחום המפתחות יכול להיות ענק
3. שימושי רק אם כמות המפתחות המקסמלית קטנה
4. גורם לבזבוז זכרון אם קיים הבדל גדול בין כמות הרשומות לערך המקסמלי

אוניברסיטת
אריאל
בשומרון

Hash Table

טבלת גיבוב (Hash Table) הוא מבנה נתונים אשר מאפשר מיפוי של מפתחות (keys) לערכים (values)
טבלת גיבוב היא טבלאות אינדקסים אשר מקשרת בין איבר למקומו במערך, מבנה נתונים המאפשר הכנסה,מחיקה וחיפוש מהיר.

Key Value

$$111111111 \rightarrow Person(Zvi, 11111111, 25/10/1995)$$

$$222222222 \rightarrow Person(Dan, 222222222, 26/10/1995)$$

Hash Function

פונקצייה $h(x)$ אשר ממפה מפתח למספר בטווח קבוע

$$h : KEYS \rightarrow \{0 \dots |A| - 1\}$$

לדוגמה 1 :

0	
1	51
2	92
3	
4	
5	15
6	
7	17
8	88
9	29

האינדקס של מפתח k הוא h(k).

דוגמא: $m = 10 \quad h(k) = k \bmod 10$

51, 17, 15, 92, 88, 29

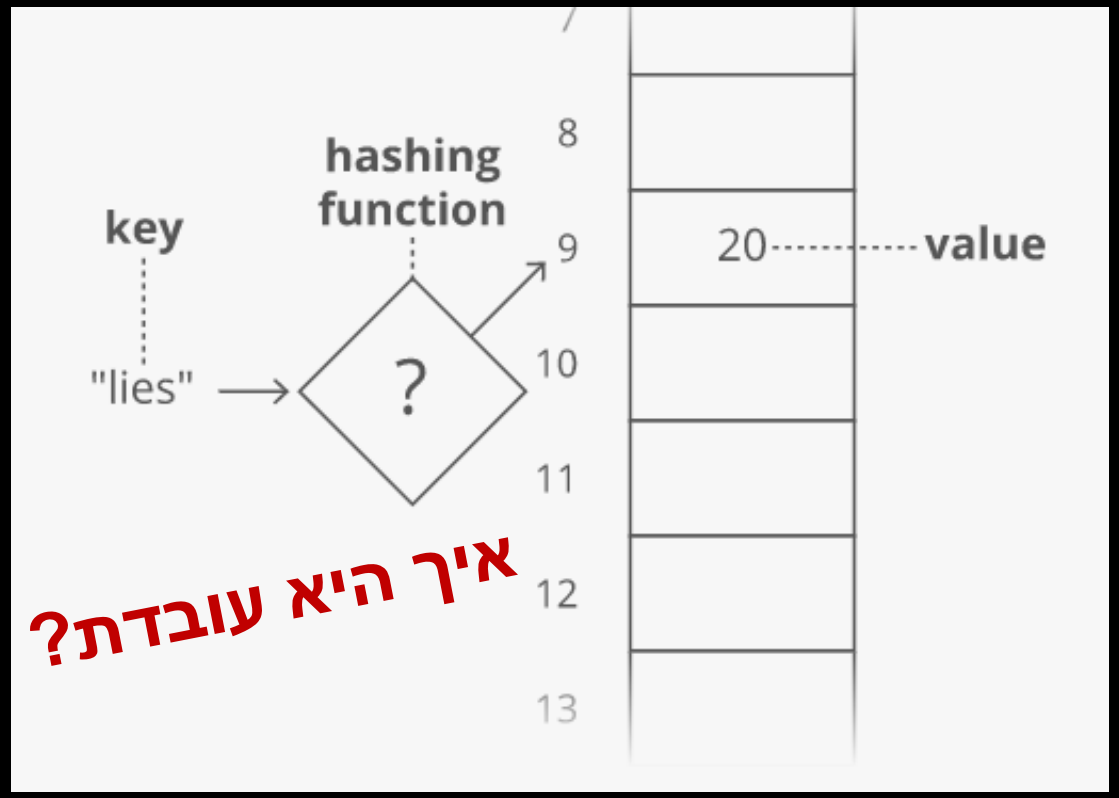
בשיטת הערבול נוצרות התנגשויות כאשר $x \neq y$ אבל $h(x) = h(y)$. לדוגמא: $h(51) = 1 = h(81)$.

Hash Function

פונקצייה $h(x)$ אשר ממפה מפתח למספר בטווח קבוע

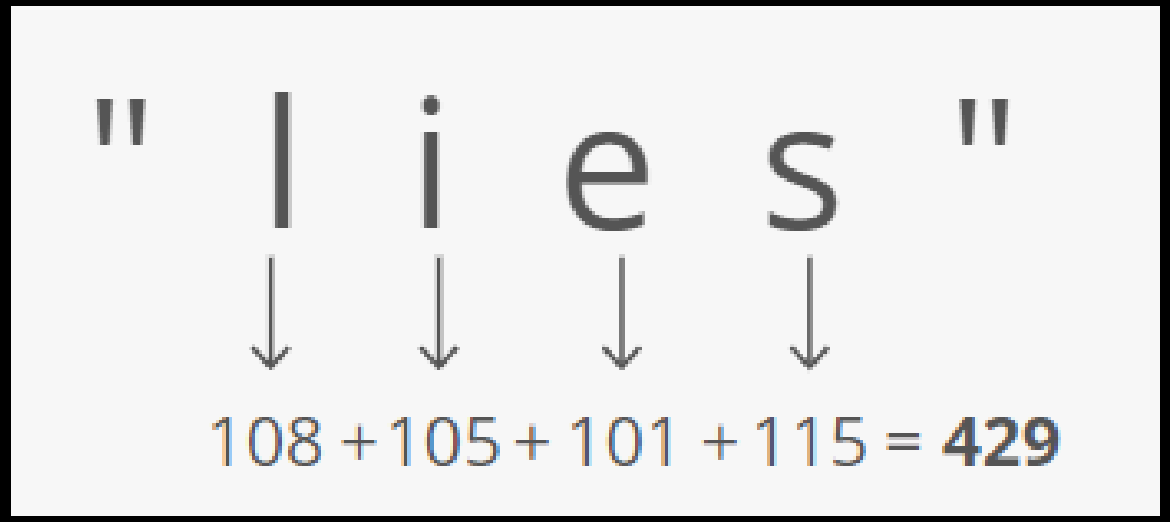
$$h : KEYS \rightarrow \{0 \dots |A| - 1\}$$

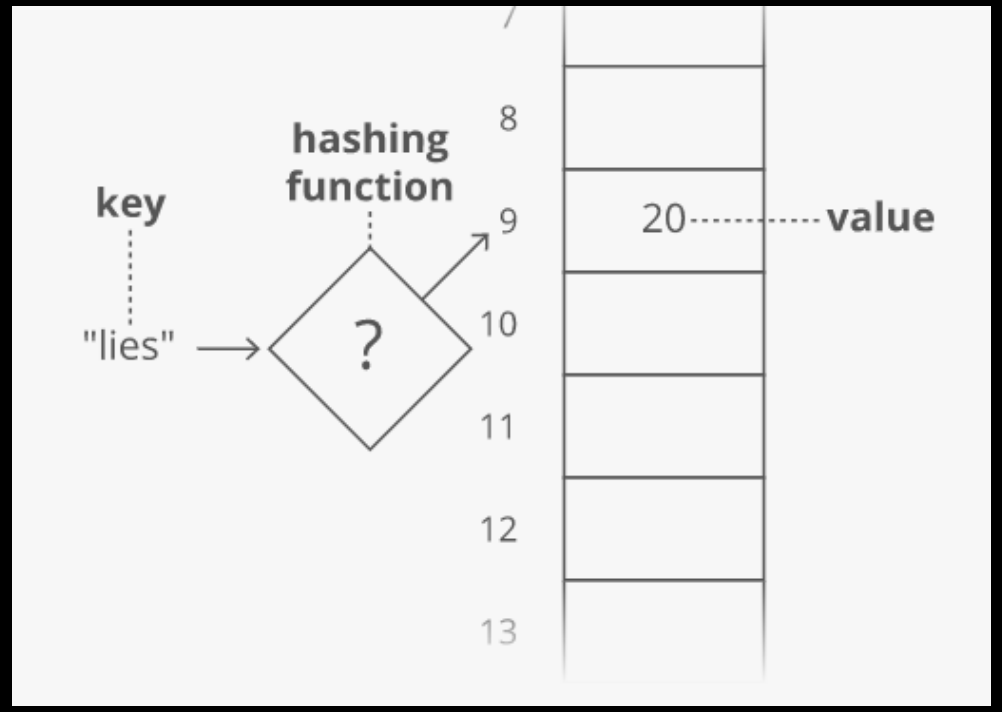
לדוגמה 2 :



Hash Function

נניח שגודל המערך
הוא 30 מקומות





sum ASCII Values % 30

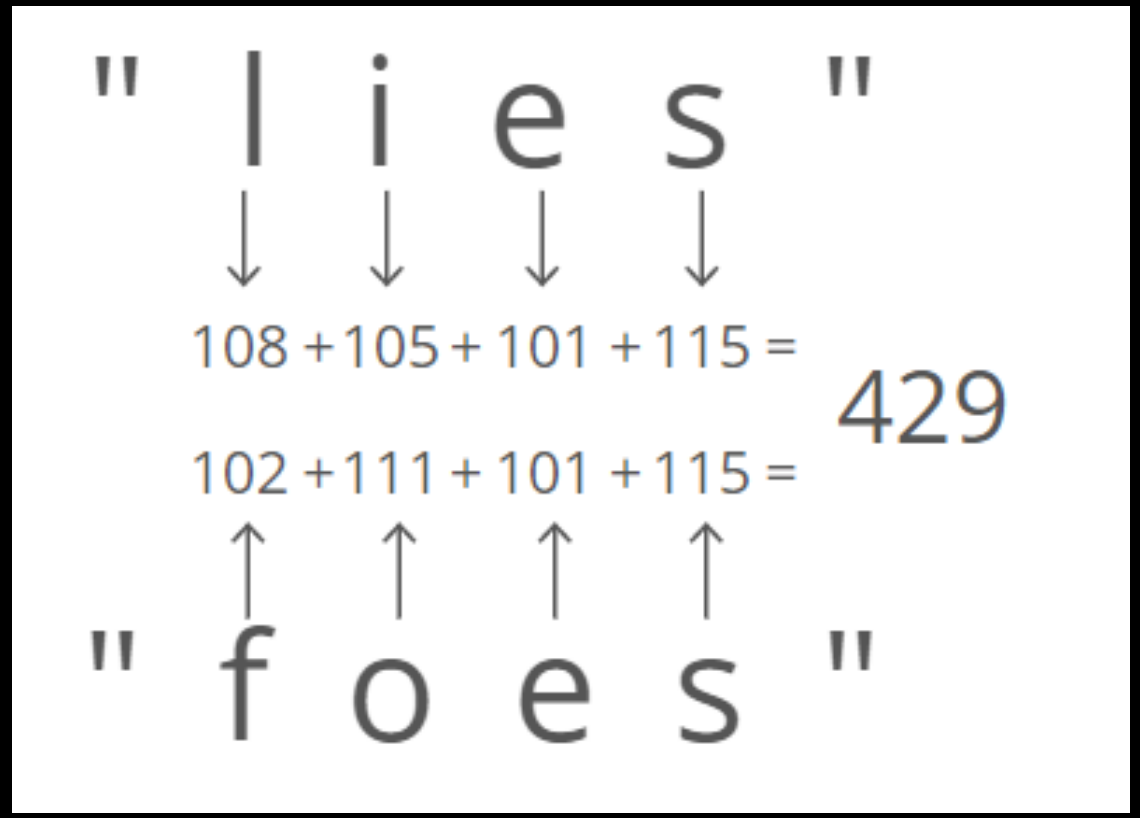
לא יכולים להיות שני קלטים זהים (מפתח הוא יחודי)

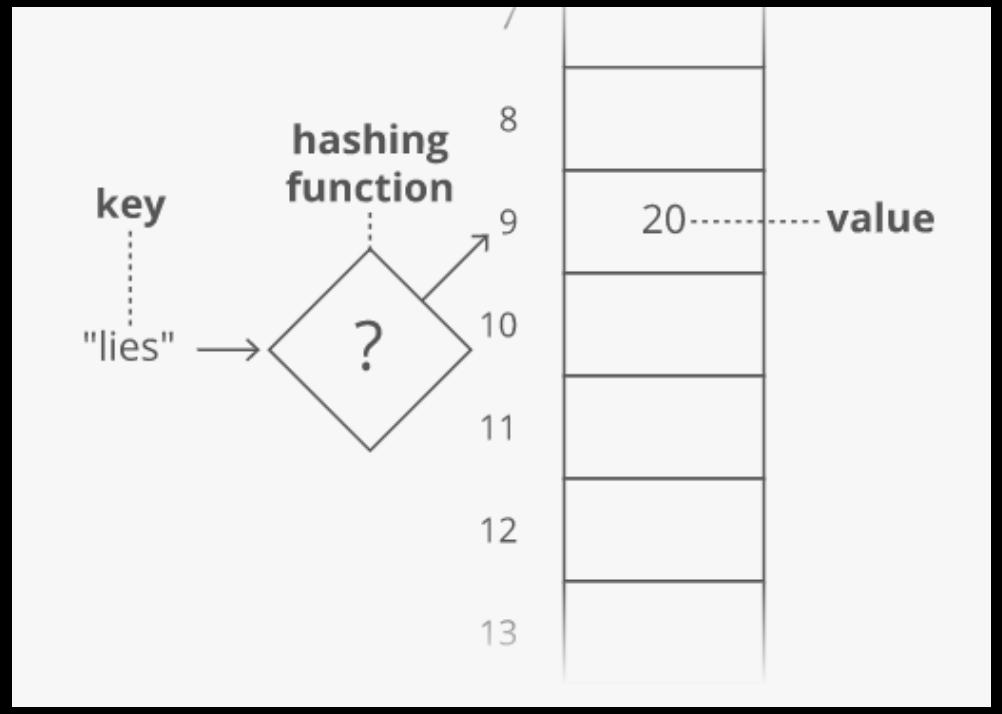
יכול להיות 2 קלטים עם פלט זהה למה?

collisions

$h(x) = h(y)$ but $x \neq y$

נניח שגודל המערך
הוא 30 מקומות





Hash Function

בדוגמה שלנו:

ID	NAME	D.O.B
111111111	Zvi	25/10/1995
222222222	Dan	26/10/1995
...		
999999999	Ilan	27/12/1995

$$|U| \left\{ \begin{array}{c} h(111111111) = 2 \\ h(222222222) = 3 \\ h(999999999) = 997 \\ \dots \end{array} \right\} [0 \dots |A| - 1]$$

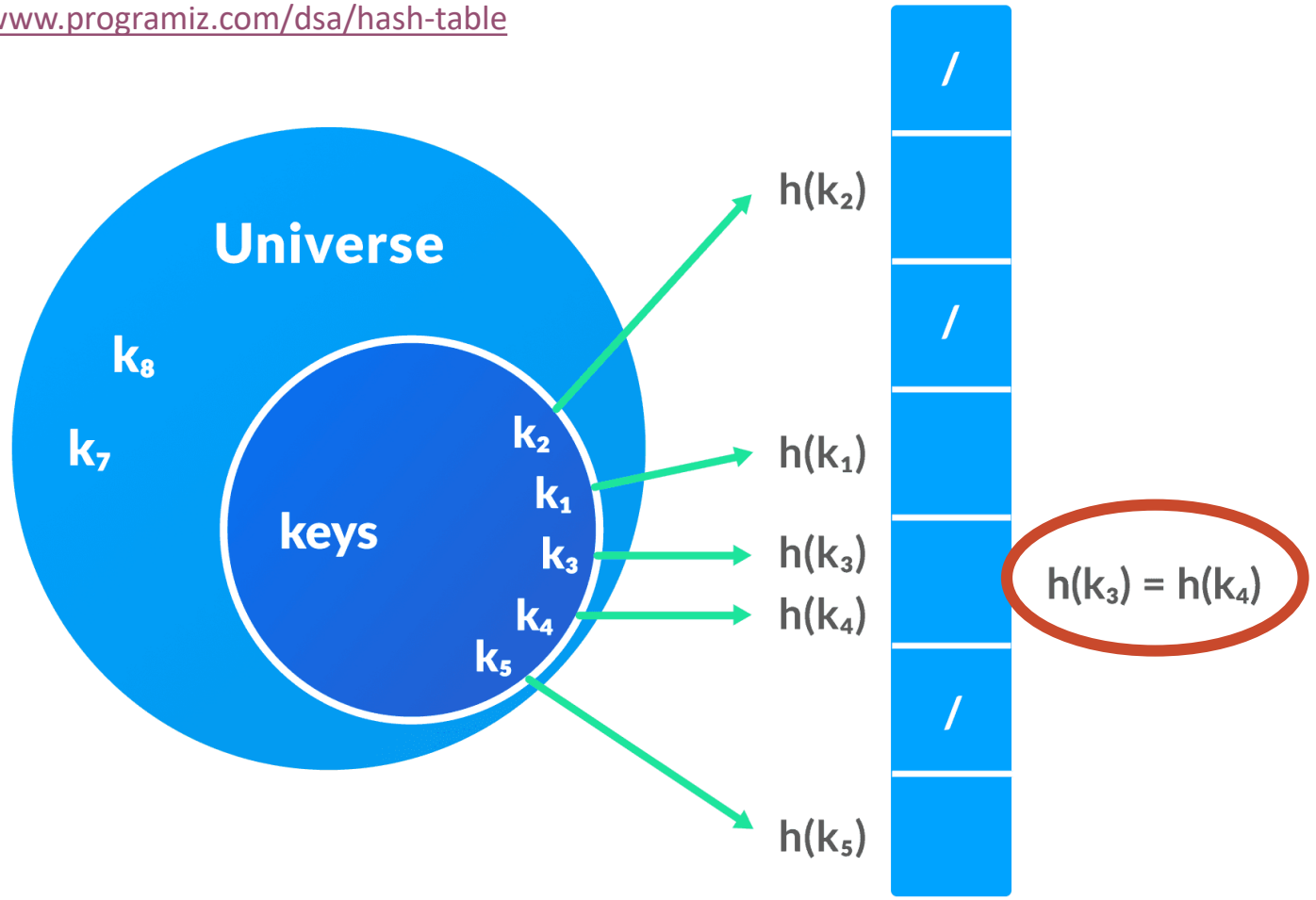
U מרחב כל המפתחות האפשריים

zvimints@gmail.com – צבי מינץ

Hash Table

אוניברסיטת
אריאל
בשומרון

https://www.programiz.com/dsa/hash-table



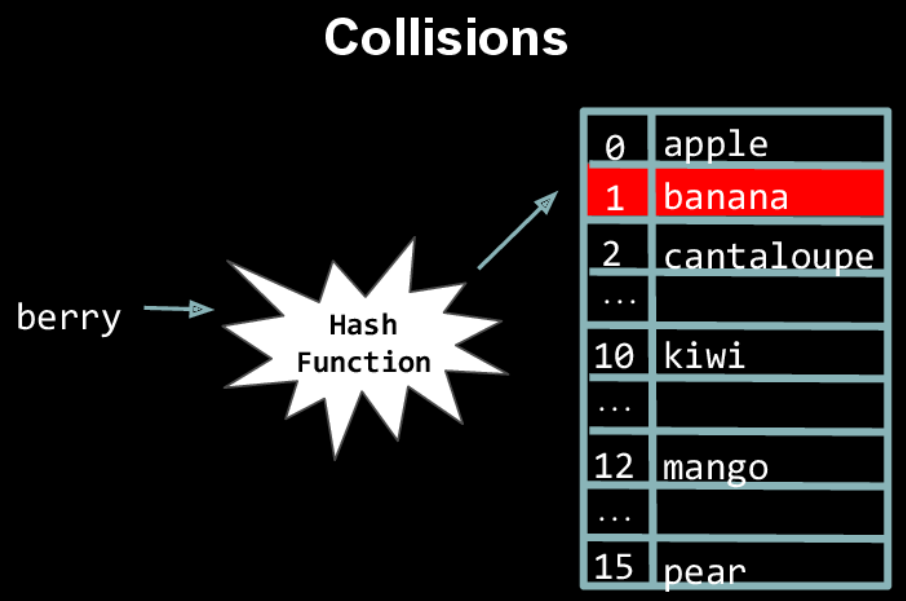

```
HashSearch(A, k)
  return T[h(k)]

HashInsert(A, x)
  T[h(x.key)] = x

HashDelete(A, x)
  T[h(x.key)] = NIL
```

הגבלות של Hash Table

אם אינדקס מסויים מופיע עבור **כמה** פלט של הפונקייה אז נגיע להתנגשות collision



על מנת **למנוע** זאת, יש צורך בפונקציית hash חח"ע, אבל, זה בלתי אפשריות כי $|U| > |A|$
ולכן **פונקציית hash** טובה **לא יכולה** למנוע אבל יכולה **להקטין** משמעותית את כמות הקונפליקים

אוניברסיטת
אריאל
בשומרון

Good Hash Functions

1. כמות הפלטים (אינדקסים) **קטנה** מכמות הקלטים האפשריים
2. כמות ההתנגשיות צריכה להיות **קטנה ככל האפשר**
3. **קלה** לחישוב
4. **דטרמינסטית**

   הפונקציה תמיד תחזיר את אותו הפלט עבור אותו הקלט

5. תשתמש **בכל** המידע של המפתח
6. מפזרת היטב (**אחיד**)
7. חד כיוונית

Hash of my email address:
8d935def1f9e0353b0f19f3c765bdeec15
1862a199084ae4f4b417ca42608914

Good Hash Functions

אתגר:

נניח כי יש לנו את המסד הבא של אנשים עם השדות
Name, Age and Sex

Name	Age	Sex	Hash
William	21	M	?
Kate	19	F	?
Bob	33	M	?
Rose	26	F	?

חשבו על פונקציית Hash אשר ממפה אובייקט למספרים $\{0,1,2,3,4,5\}$

אוניברסיטת
אריאל
בשומרון

Good Hash Functions

Name	Age	Sex	Hash
William	21	M	?
Kate	19	F	?
Bob	33	M	?
Rose	26	F	?

פתרון אפשרי:

```
function H(person):
    hash := person.age
    hash = hash + length(person.name)
    if person.sex == "M":
        hash = hash + 1
    return hash mod 6
```

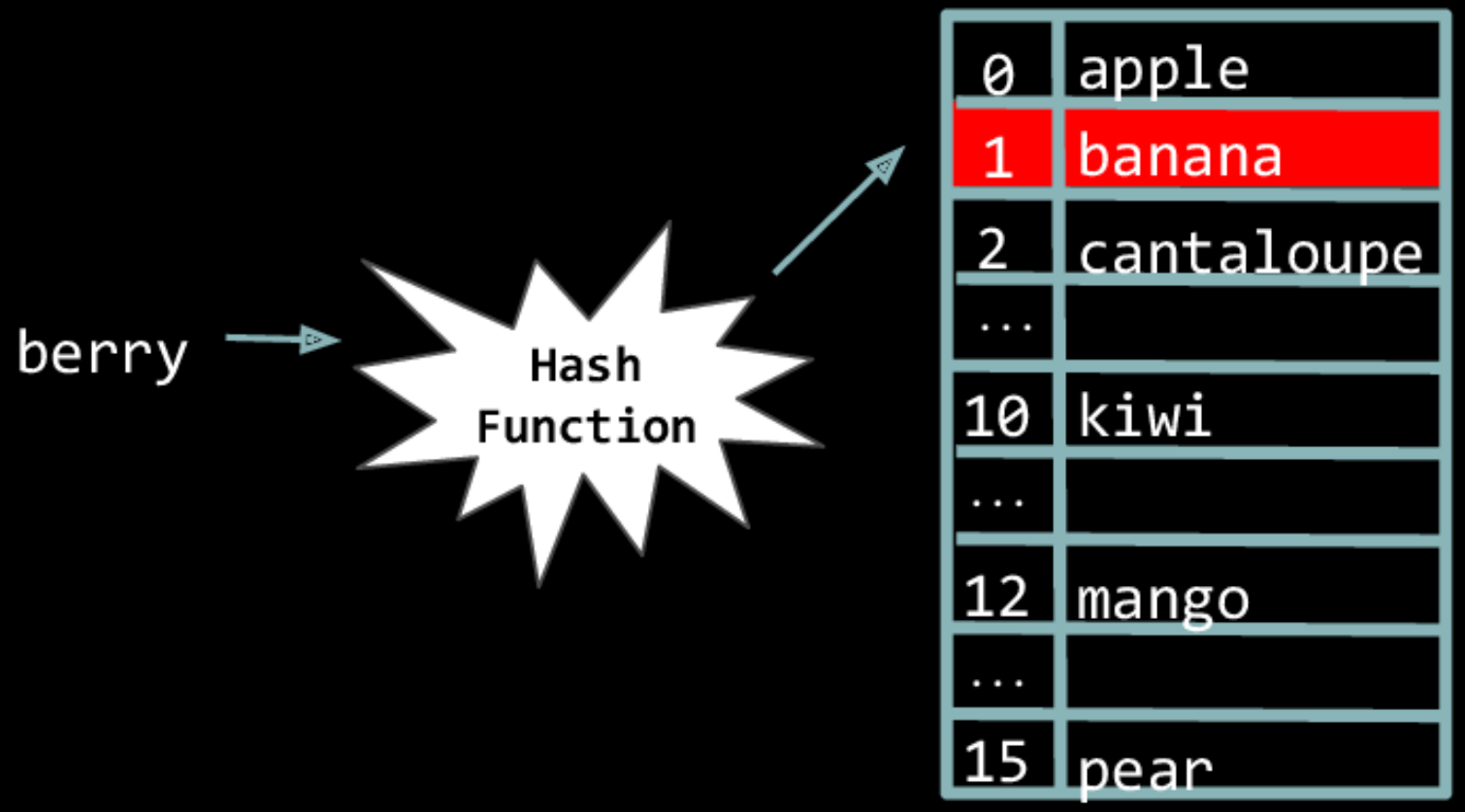

https://www.youtube.com/watch?v=2E54GqF0H4s&list=PLDV1Zeh2NRsB6SWUrDFW2RmDotAfPbeHu&index=29

Hash Table

נניח כי $h(x)$ לוקחת זמן קבוע
ונניח כי $h(x)$ מצליחה לפתור התנגשיות

	במקרה הממוצע	במקרה הרע
זיכרון	$O(n)$	$O(n)$
הכנסה	$O(1)$	$O(n)$
חיפוש	$O(1)$	$O(n)$
מחיקה	$O(1)$	$O(n)$

Collisions



איך פותרים התנגשות?

איך פותרים התנגשות?

Separate Chaining

מערך של רשימות מקושרות

Open addressing

מציאת מקום אחר לאובייקט

Linear Probing

Double hashing

Separate chaining

במערך רגיל **אי אפשר** להכניס שני ערכים לאותו מקום, על כן ניצור מערך של רשימות מקושרות, שבו אפשר להכניס באינדקס אחד מספר ערכים המוחזקים ברשימה. באופן זה, ערך נכנס **תמיד** ישירות לאינדקס שפונקצית Hashהחזירה, ואם הגיע ערך לאותו אינדקס, הוא ישורשר אחרי זה שכבר קיים ברשימה מקושרת. בשיטה זו רוב הערכים יהיו בפיזור טוב.

insert(T, x) - הכנס את x בראש הרשימה T[h(x.key)]. סיבוכיות זמן: O(1).

Search(T, k) - חפש איבר עם מפתח k ברשימה T[h(k)]. סיבוכיות זמן: Θ(List length)

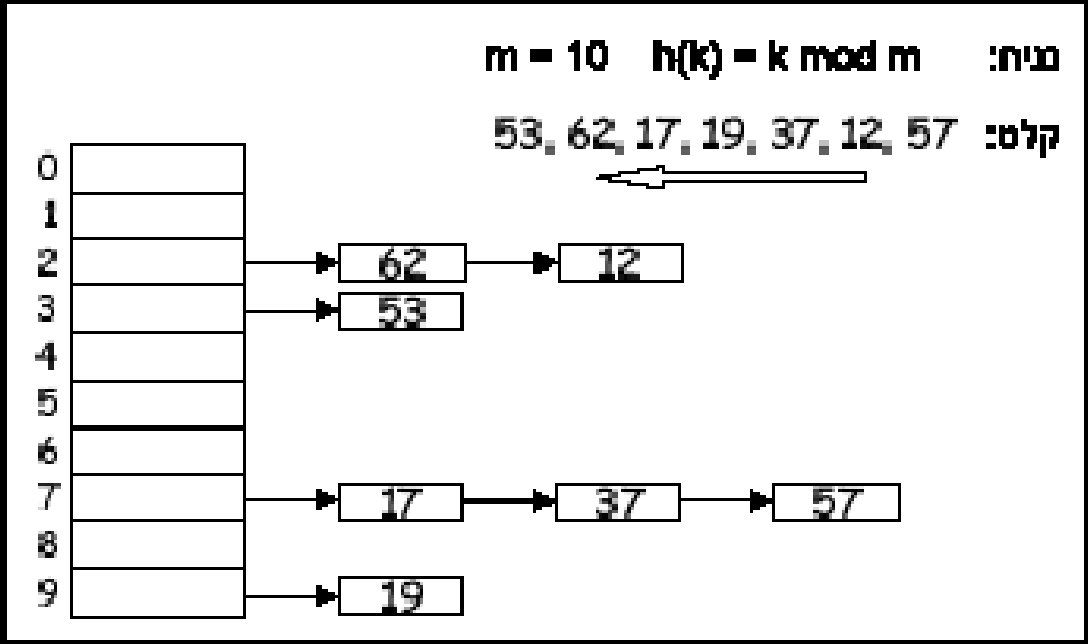
Delete(T, x) - סלק את x מהרשימה T[h(x.key)]. סיבוכיות זמן: Θ(List length)

במקרה הגרוע ביותר כל האיברים נכנסו לאותה הרשימה ואז זמן החיפוש/הוצאה הוא $\Theta(n)$. ברור לפיכך שאין משתמשים בערבול בגלל הזמן המקסימאלי לפעולה אלא בגלל הזמן הממוצע לפעולה. נרצה לבחור פונקצית ערבול שמפזרת היטב את המפתחות לרשימות השונות.

Separate chaining

במערך רגיל **אי אפשר** להכניס שני ערכים לאותו מקום, על כן ניצור מערך של רשימות מקושרות, שבו אפשר להכניס באינדקס אחד מספר ערכים המוחזקים ברשימה. באופן זה, ערך נכנס **תמיד** ישירות לאינדקס שפונקצית Hashהחזירה, ואם הגיע ערך לאותו אינדקס, הוא ישורשר אחרי זה שכבר קיים ברשימה מקושרת. בשיטה זו רוב הערכים יהיו בפיזור טוב.

דוגמה:



zvimints@gmail.com – צבי מינץ

Separate chaining

אוניברסיטת
אריאל
בשומרון

סיבוכיות הכנסת איבר וחיפוש איבר ב- Hash Table

פונקצית הגיבוב מחשבת את האינדקס בזמן O(1).

זמן הכנסה והחיפוש יהיה מהיר, שכן באינדקס משורשר יש מספר קטן מאוד של איברים.

מספר האיברים שמשורשרים בכל כניסה **ממוצע** הוא: מספר האיברים הכללי מחולק למספר הכניסות.

נסמן את מספר האיברים בפועל ב-N, ומספר הכניסות במערך ב-M.

אזי סה"כ מספר האיברים **ממוצע** עבור כל כניסה הוא $\alpha = \frac{M}{N}$, משתנה זה נקרא Load Factor.

מציאת איבר ברשימה מקושרת מחייבת מעבר על כל איברי הרשימה באופן ליניארי $O(\alpha)$.

לכן הסיבוכיות ברשימה של α איברים היא $O(\alpha)$.

הסיבוכיות של מציאת הכניסה היא O(1).

וסה"כ הסיבוכיות של חיפוש ב- Hash Table: $O(\alpha + 1) = O(\alpha)$.

java.util.Hashtable

```java
public static void main(String[] args) {
Hashtable<Integer,Person> map = new Hashtable<>();
Person p1 = new Person(111111111,95);
Person p2 = new Person(222222222,85);
Person p3 = new Person(333333333,75);

map.put(1,p1);
map.put(2,p2);
map.put(3,p3);

System.out.println(map.isEmpty()); // false
System.out.println(map.isEmpty()); // false
System.out.println(map.remove(1)); // (111111111,95)
map.put(2, p3); // Do bothing ( key UNIQUE )
System.out.println(map.containsKey(2)); // true
System.out.println(map.containsKey(p2)); // false
System.out.println(map.containsValue(2)); // false
System.out.println(map.containsValue(p3)); // true

System.out.println(map); // {3=(333333333,75), 2=(222222222,85)}

}
```

```java
static class Person {
    private long _id;
    private int _score;
    public Person(long _id, int _score) {
        this._id = _id;
        this._score = _score;
    }
    public String toString() {
        return "("+this._id + ","+this._score+")"
    }
}
```

שאלה 1

כתוב מחלקה המקראת `MyHashTable` וממשת **Hash Table** שמבוססת על רשימה מקושרת של Java.

טבלת גיבוב זו מכילה רשימת סטודנטים של אוניברסיטת אריאל.

המפתח (key) צריך להיות מטיפוס מספר שלם (Integer), שהוא מספר זהות של סטודנט.

הערך צריך להיות אובייקט מטיפוס Student שמכיל שם וגיל של סטודנט.

המחלקה מכילה שיטות הבאות:

- בנאי `public MyHashTable(int size)`
- בנאי מעתיק `public MyHashTable(MyHashTable ht)`
- הוספת סטודנט חדש. כאשר רוצים להוסיף סטודנט שמספר זהות שלו כבר מופיע ברשימה צריך להחליף את שמו בשם חדש. `public Integer insert(Integer key, Student data)`
- מחיקת סטודנט מהרשימה: `public Student remove(String key)`
  הפונקציה מחזירה את נתוני הסטודנט שנמחק.
- חיפוש הסטודנט ע"פ מספר זהות שלו. `public Student get(String key)`
  הפונקציה מחזירה את נתוני הסטודנט.
- `public String toString()` – המחזירה את כל נתוני הטבלה כמחרוזת.

כמובן צריך להגדיר מחלקת **Node** שמייצגת איבר של הטבלה.

עבודה עצמית (20 דקות)

אוניברסיטת
אריאל
בשומרון

סעיף ג' (8%): נתונה טבלת hash בגודל m=9 (כלומר T[0..8]). הטבלה מנוהלת בשיטת linear probing עם הפונקציה h(k) = k mod 9. המפתחות הבאים הוכנסו לטבלה (מימין לשמאל): 8,2,9,17 באיזה תא ימצא המפתח 8? _____

0	
1	
2	
3	
4	
5	
6	
7	
8	17

0	9
1	
2	
3	
4	
5	
6	
7	
8	17

0	9
1	
2	2
3	
4	
5	
6	
7	
8	17

0	9
1	8
2	2
3	
4	
5	
6	
7	
8	17

סעיף ב' (10 נקודות)

נתונה טבלת גיבוב (hash) בגודל 10 המנוהלת בשיטת double hashing.

פונקצית הגיבוב: h1(x)=x mod 10

פונקציה הצעד: h2(x)= $\lfloor x/10 \rfloor$

$$(h_1(x) + i \cdot h_2(x)) \; mod \; 10$$

מה הייתה סדרת ההכנסות לטבלה הבאה?

index	0	1	2	3	4	5	6	7	8	9
element	46	10	39			35	42			45

התשובה (משמאל לימין): 35, 45 39, 42, 46, 10

שאלה 3

נתון מערך A בגודל n של מספרים ממשיים ומספר ממשי נוסף z
הציעו אלגוריתם שמכריע האם קיימים שני אינדקסים שונים i,j כך ש-
$A[i] + A[j] = z$ $\quad O(n)$

```
HashMap<Integer,Integer> map = new HashMap<>();
containsKey(key)
Get(key)
```

https://leetcode.com/problems/two-sum/

אוניברסיטת
אריאל
בשומרון

```
public class TwoSums {
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int[] ans = new int[2];
        HashMap<Integer,Integer> map = new HashMap<>();
        // Key -> A[i];
        // Value -> i
        for(int i=0; i<nums.length; i++) {
        if(map.containsKey(target - nums[i])) {
            ans[1] = i;
            ans[0] = map.get(target - nums[i]);
         }
         else
            map.put(nums[i],i);
         }
        return ans;
         }
    }
}
```