

Hashing Functions

Ms. Rajanpreet Kaur Chahal

Department of Computer Science & Engineering
Thapar Institute of Engineering & Technology, Patiala

Hashing Function

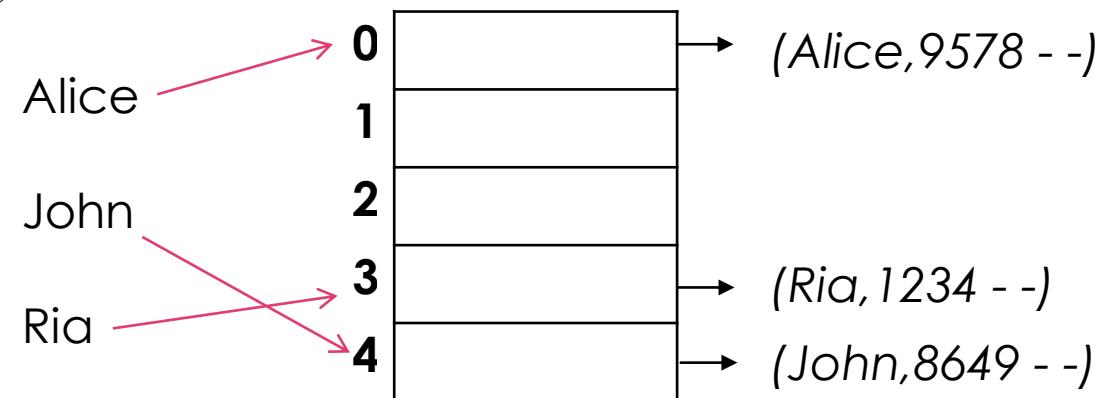
- ▶ Hashing function is a function which is applied on a key by which it produces an integer, which can be used as an address in hash table.
- ▶ A simple hashing function: $h(k) = k \bmod m$

Properties of Hashing Functions

- ▶ Easy to compute
- ▶ Uniform distribution
- ▶ Less collisions
- ▶ Onto

Hash Table: Example

- ▶ **Example:** phone book with table size $N = 5$
- ▶ **hash function** $h(w) = (\text{length of the word } w) \bmod 5$
- ▶ **Problem:** collisions
- ▶ Where to store Joe (collides with Ria)



Collisions

- ▶ Collisions occur when different elements are mapped to the same cell.
- ▶ Keys k_1, k_2 with $h(k_1) = h(k_2)$ are said to collide

What should we do now?

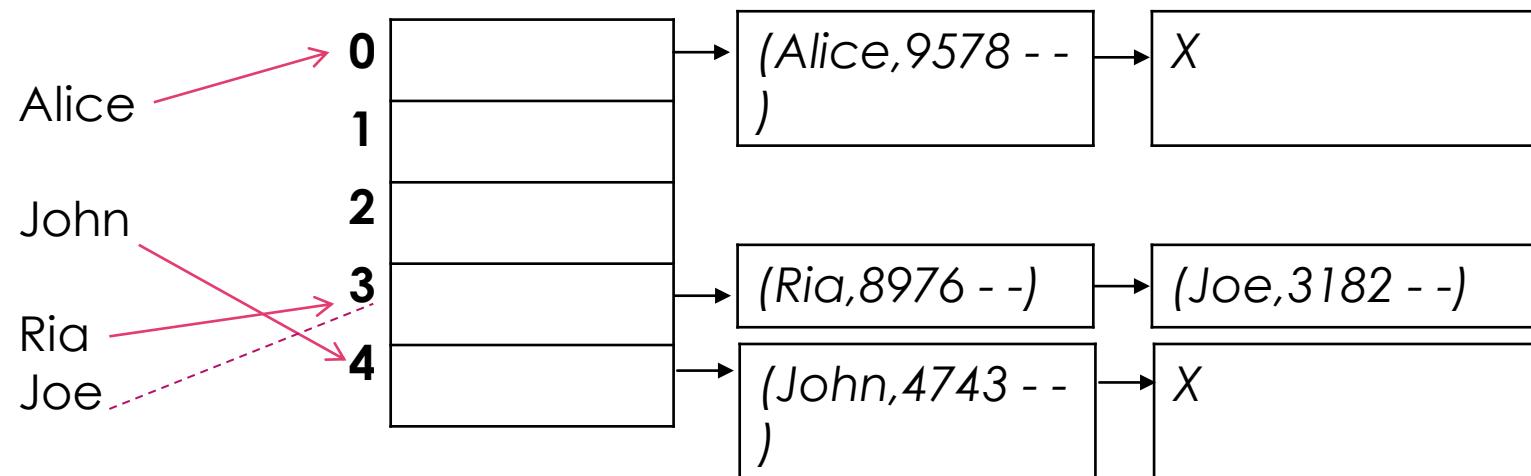
- ▶ Find a better hashing algorithm
- ▶ Use a bigger table
- ▶ Need a system to deal with collisions

Resolving Collisions

- ▶ Two different methods for collision resolution:
- ▶ **Separate Chaining:** Use a dictionary data structure (such as a linked list) to store multiple items that hash to the same slot.
- ▶ **Closed Hashing (or *Open Addressing*):** search for empty slots using a second function and store item in first empty slot that is found

Separate Chaining

- ▶ Each cell of the hash table points to a linked list of elements that are mapped to this cell.
- ▶ Simple, but requires additional memory outside of the table.



Closed Hashing or Open Addressing

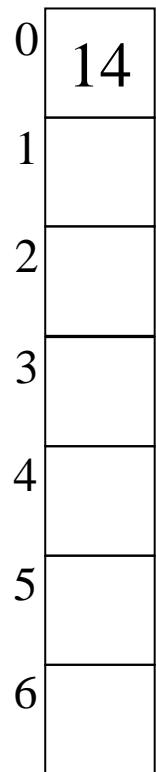
- ▶ Open addressing does not introduce a new structure.
- ▶ If a collision occurs then we look for availability in the next spot generated by an algorithm
- ▶ There are many implementations of open addressing, using different strategies for where to probe next:
 1. Linear Probing
 2. Quadratic Probing
 3. Double Hashing

Contd..

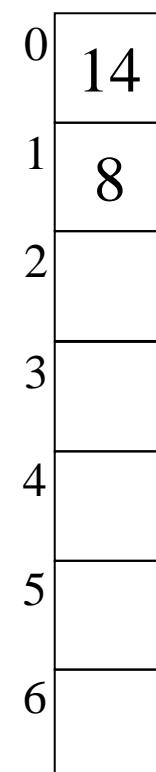
- ▶ Given an item X , try cells $h_0(X), h_1(X), h_2(X), \dots, h_i(X)$
 - ▶ $h_i(X) = (\text{Hash}(X) + F(i)) \bmod \text{TableSize}$
 - ▶ $F(0) = 0$
- ▶ F is the *collision resolution* function. Some possibilities:
 - ▶ **Linear:** $F(i) = i$
 - ▶ **Quadratic:** $F(i) = i^2$
 - ▶ **Double Hashing:** $F(i) = i * \text{Hash}_2(X)$

Linear Probing Example

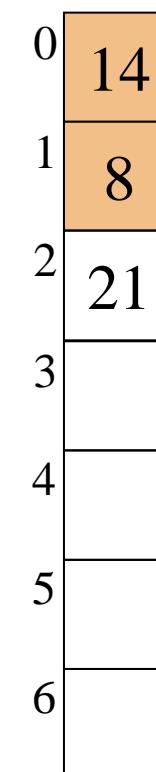
insert(14)
 $14 \% 7 = 0$



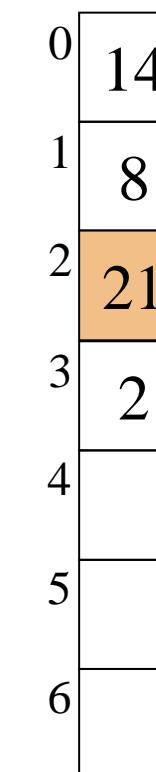
insert(8)
 $8 \% 7 = 1$



insert(21)
 $21 \% 7 = 0$

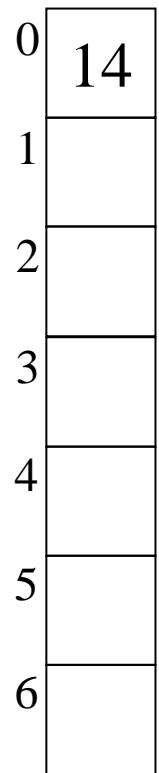


insert(2)
 $2 \% 7 = 2$

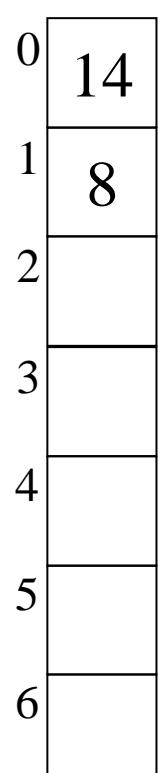


Quadratic Probing Example

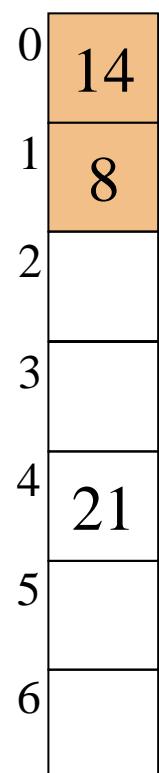
insert(14)
 $14 \% 7 = 0$



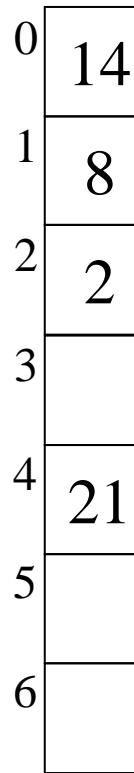
insert(8)
 $8 \% 7 = 1$



insert(21)
 $21 \% 7 = 0$



insert(2)
 $2 \% 7 = 2$



Double Hashing

- ▶ Double hashing can be done using :
$$(\text{hash1}(\text{key}) + i * \text{hash2}(\text{key})) \% \text{TABLE_SIZE}$$
- ▶ First hash function is typically
$$\text{hash1}(\text{key}) = \text{key \% TABLE_SIZE}$$
- ▶ A popular second hash function is :
$$\text{hash2}(\text{key}) = \text{PRIME} - (\text{key \% PRIME})$$

where PRIME is a prime smaller than the TABLE_SIZE.

Double Hashing Example

insert(19)
 $19 \% 13 = 6$

insert(27)
 $27 \% 13 = 1$

insert(36)
 $36 \% 13 = 10$

insert(10)
 $10 \% 13 = 10$

0	
1	
2	
3	
4	
5	
6	19
7	
8	
9	
10	
11	
12	

0	
1	27
2	
3	
4	
5	
6	19
7	
8	
9	
10	
11	
12	

0	
1	27
2	
3	
4	
5	
6	19
7	
8	
9	
10	36
11	
12	

0	
1	27
2	
3	
4	
5	10
6	19
7	
8	
9	
10	36
11	
12	

Collision 2

Let $\text{Hash2}(\text{key})=7-(\text{key} \% 7)$

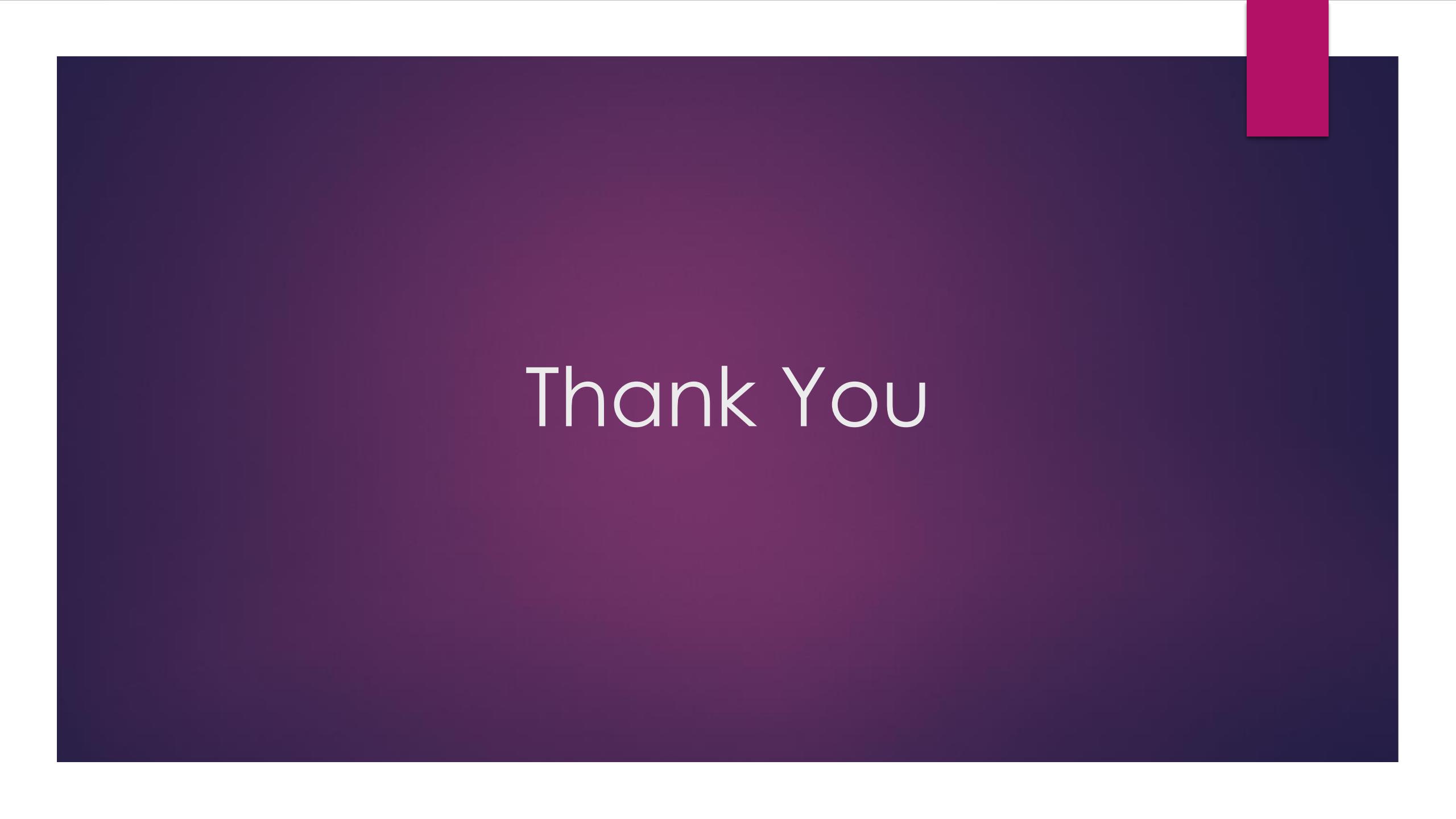
$\text{Hash1}(10)=10 \% 13=10$ (Collision 1)

$\text{Hash 2}(10)=7-(10 \% 7)=4$

$(\text{Hash1}(10)+1*\text{Hash2}(10)) \% 13=1$ (Collision 2)

$(\text{Hash1}(10)+2*\text{Hash2}(10)) \% 13=5$

Collision 1



Thank You