Hashing Functions

Ms. Rajanpreet Kaur Chahal

Department of Computer Science & Engineering

Thapar Institute of Engineering & Technology, Patiala

Hashing Function
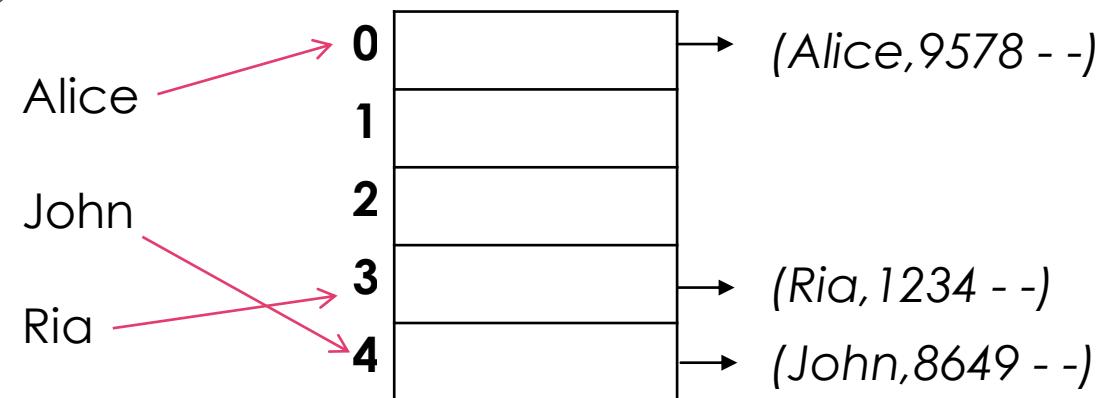
▶ Hashing function is a function which is applied on a key by which it produces an integer, which can be used as an address in hash table.

▶ A simple hashing function: h(k) = k mod 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) = (length of the word w) mod 5

▶ **Problem:** collisions

▶ Where to store Joe (collides with Ria)

0	→ (Alice,9578 - -)
1	
2	
3	→ (Ria,1234 - -)
4	→ (John,8649 - -)

Alice → 0
John → 4
Ria → 3

Collisions

▶ Collisions occur when different elements are mapped to the same cell.

▶ Keys *k1, k2* with *h(k1) = h(k2)* are said to collide

What should we do now?
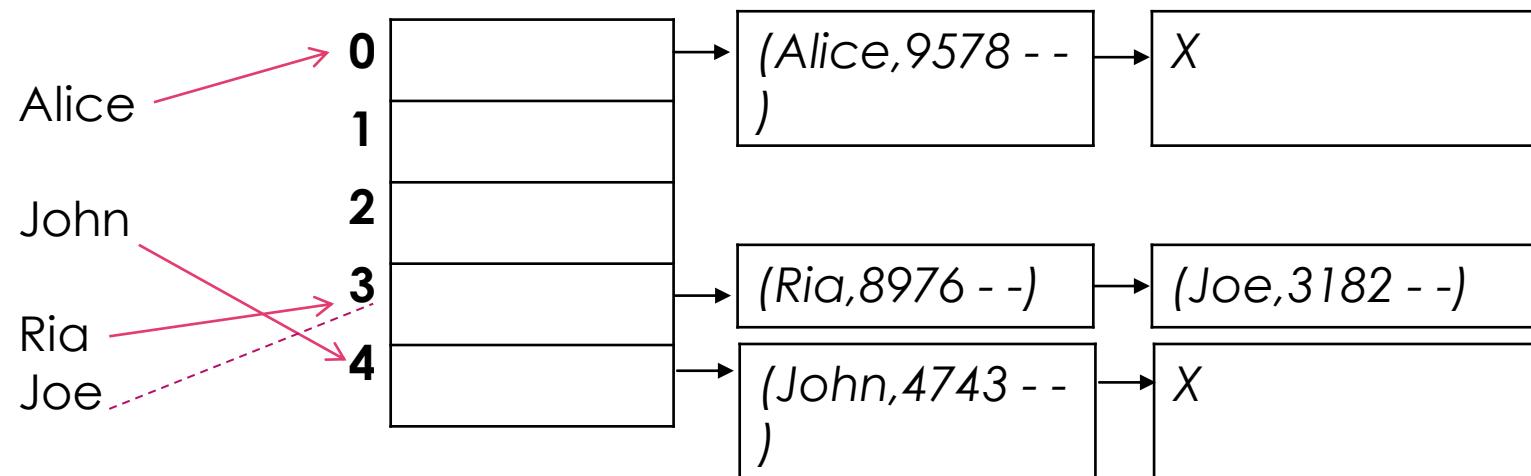
▶ Find a better hashing algorithm

▶ 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..
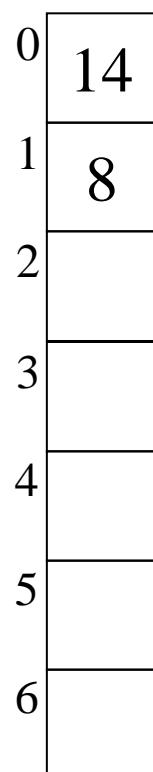
▶ Given an item X, try cells $h_0(X)$, $h_1(X)$, $h_2(X)$, …, $h_i(X)$

  ▶ $h_i(X) = (Hash(X) + F(i)) \bmod TableSize$

  ▶ $F(0) = 0$

▶ F is the *collision resolution* function. Some possibilities:

  ▶ Linear: $F(i) = i$

  ▶ Quadratic: $F(i) = i^2$

  ▶ Double Hashing: $F(i) = i*Hash_2(X)$

Linear Probing Example

insert(14)

$14\%7 = 0$

insert(8)

$8\%7 = 1$

insert(21)

$21\%7 = 0$

insert(2)

$2\%7 = 2$

	Table 1		Table 2		Table 3		Table 4
0	14	0	14	0	14	0	14
1		1	8	1	8	1	8
2		2		2	21	2	21
3		3		3		3	2
4		4		4		4	
5		5		5		5	
6		6		6		6	
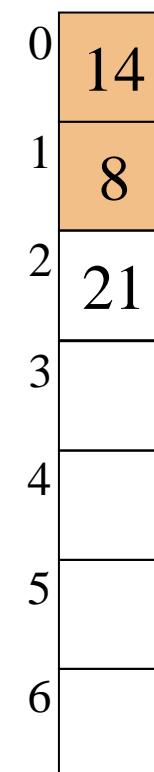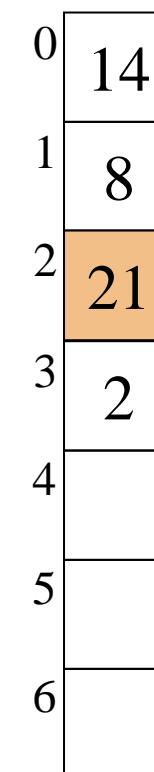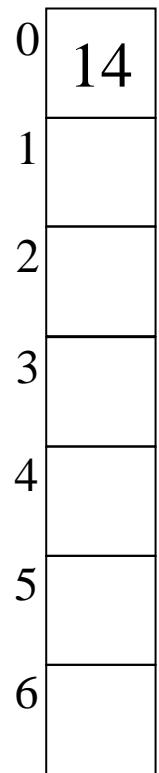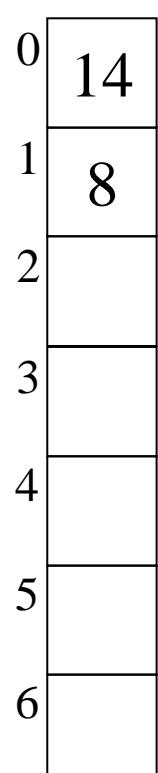
Quadratic Probing Example

insert(14)
14%7 = 0

insert(8)
8%7 = 1

insert(21)
21%7 = 0

insert(2)
2%7 = 2

insert(14)	insert(8)	insert(21)	insert(2)
0: 14	0: 14	0: 14	0: 14
1:	1: 8	1: 8	1: 8
2:	2:	2:	2: 2
3:	3:	3:	3:
4:	4:	4: 21	4: 21
5:	5:	5:	5:
6:	6:	6:	6:
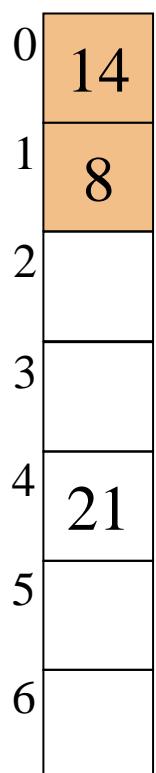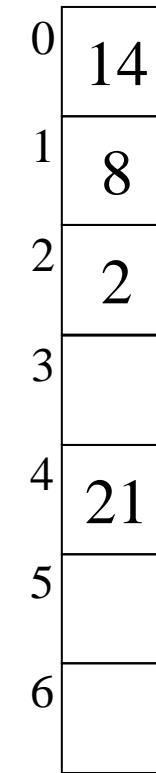
Double Hashing

▶ *Double hashing can be done using :*
*(hash1(key) + i * hash2(key)) % TABLE_SIZE*

▶ First hash function is typically

hash1(key) = key % TABLE_SIZE

▶ A popular second hash function is :

 hash2(key) = PRIME – (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)
$10 \% 13 = 10$

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

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

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

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

Let Hash2(key)=7-(key % 7)

Hash1(10)=10%13=10 (Collision1)
Hash 2(10)=7-(10%7)=4

(Hash1(10)+1*Hash2(10))%13=1(Collision 2)
(Hash1(10)+2*Hash2(10))%13=5

Thank You