Algorithms and Data Structures 2

Recap Lectures 17-18

Dr Michele Sevegnani

School of Computing Science
University of Glasgow

michele.sevegnani@glasgow.ac.uk

Topics we covered so far

- **Map ADT**
- **Direct-address tables**
- **Hash table data structure**
- **Hash functions**
- **Collision resolution by chaining**
- **Collision resolution by open addressing**
  - Linear probing
  - Quadratic probing
  - Double hashing
- **Perfect hashing**

Question 1

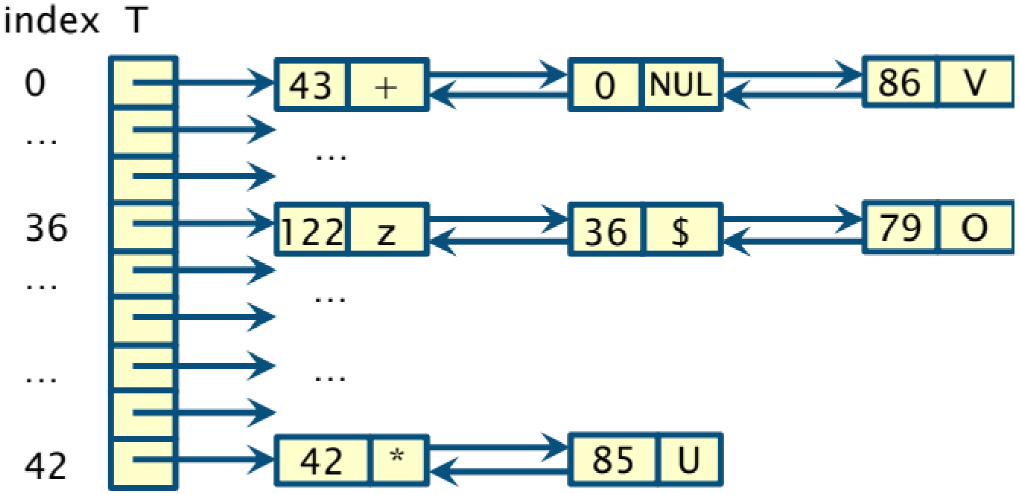
- **Briefly describe collision resolution by chaining and open addressing in the hash table data structure**

Question 1: solution

- **In collision resolution by chaining, all elements that hash to the same slot are stored in a linked list (called chain)**
  - Doubly linked list to support fast deletion
  - Usually, no check is performed to prevent the insertion of elements with duplicate keys
  - Some implementations store the first element of each list directly in T
  - Operations are based on the standard operations for doubly linked lists

- **Example**

```
CHAINED-HASH-INSERT(T,x)
  insert x at the head of list T[h(x.key)]
```



Question 1: solution (cont.)

- **In collision resolution by open addressing, alternative cells are probed until an empty cell is found**

- **To determine which slots to probe, the hash function is extended to include the probe number (starting from 0) as a second input: $h: U \times \{0,1,\ldots,m-1\} \rightarrow \{0,1,\ldots,m-1\}$**
  - Linear probing
  - Quadratic probing
  - Double hashing

- **With open addressing, we require that every hash-table position is eventually considered as a slot for a new key as the table fills up**

Question 1: solution (cont.)

- **Linear probing, uses a hash function of the form $h(k,i) = (h'(k) + i) \bmod m$ where h' is an ordinary hash function**

- **Linear probing is easy to implement, but it suffers from primary clustering**
  - Long runs of occupied slots build up, increasing the average search time

Question 1: solution (cont.)

- **Quadratic probing uses a hash function of the form $h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$ where h' is an ordinary hash function and c_1, c_2 constants**

- **If two keys have the same initial probe position, then their probe sequences are the same**
  - This leads to a milder form of clustering, called secondary clustering
  - In general, better than linear probing

- **There is no guarantee of finding an empty cell once the table gets more than half full**

Question 1: solution (cont.)

- **Double hashing uses a hash function of the form $h(k,i) = (h_1(k) + ih_2(k)) \bmod m$ where h_1 and h_2 are ordinary hash functions**

- **The value $h_2(k)$ must be relatively prime to the hash-table size m for the entire hash table to be searched**

1. Let m be a power of 2 and design h_2 to always produce an odd number
2. Let m be prime and design h_2 to always return a positive integer less than m

- **One of the best methods available for open addressing**

Question 2

- **Draw a representation of a 7-celled hash table T and its contents after the hash function h(x) = (3x+1) mod 7 has been used to insert the elements in the sequence 100,19,3,5,4,7 into T, assuming collisions are handled using**

a) the chaining method

b) linear probing

Question 2: solution

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using chaining**

Hashes

h(100) = 0

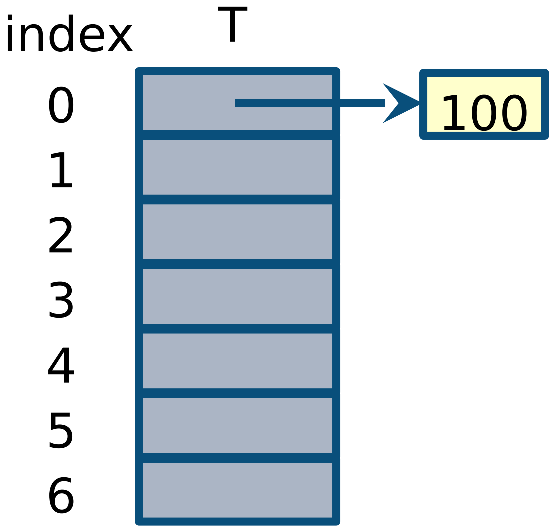
h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1



Question 2: solution

- h(x) = (3x+1) mod 7
- insert the elements in the sequence 100,19,3,5,4,7 into H
- assuming collisions are handled using chaining

Hashes

h(100) = 0

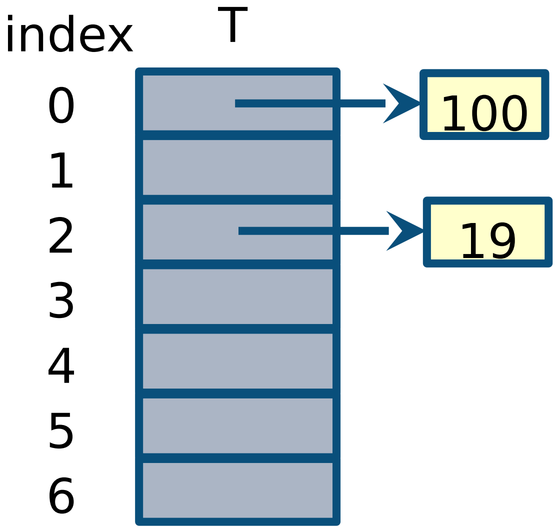
h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1



Question 2: solution

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using chaining**

Hashes

h(100) = 0

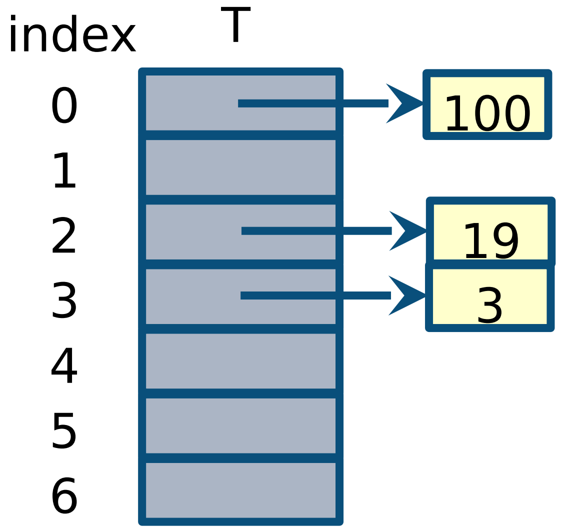
h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1



Question 2: solution

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using chaining**

Hashes

h(100) = 0

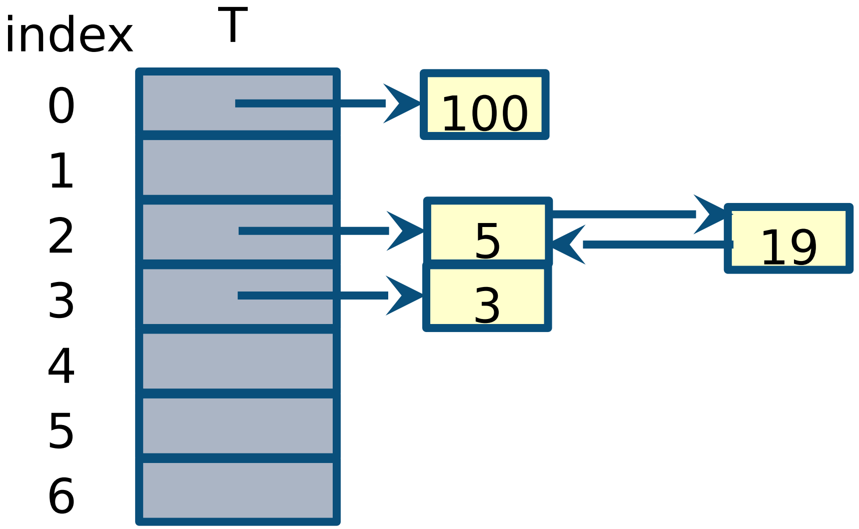
h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1



Question 2: solution

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using chaining**

Hashes

h(100) = 0

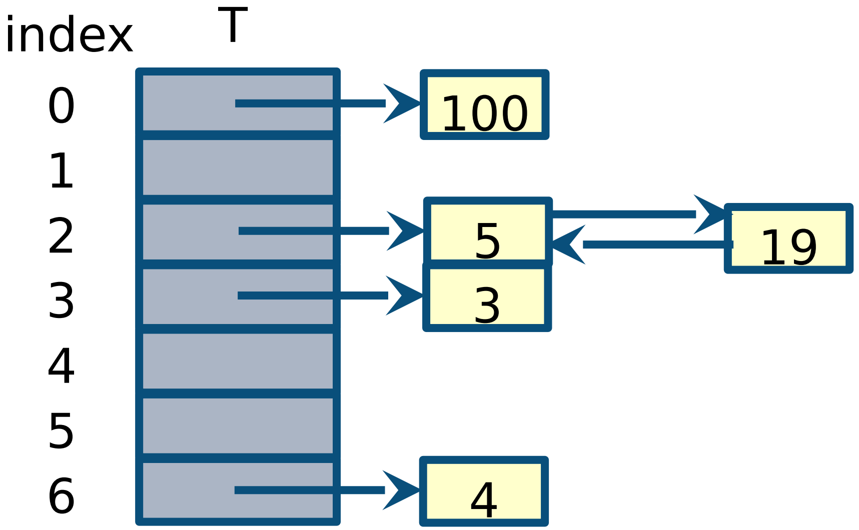
h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1



Question 2: solution

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using chaining**

Hashes

h(100) = 0

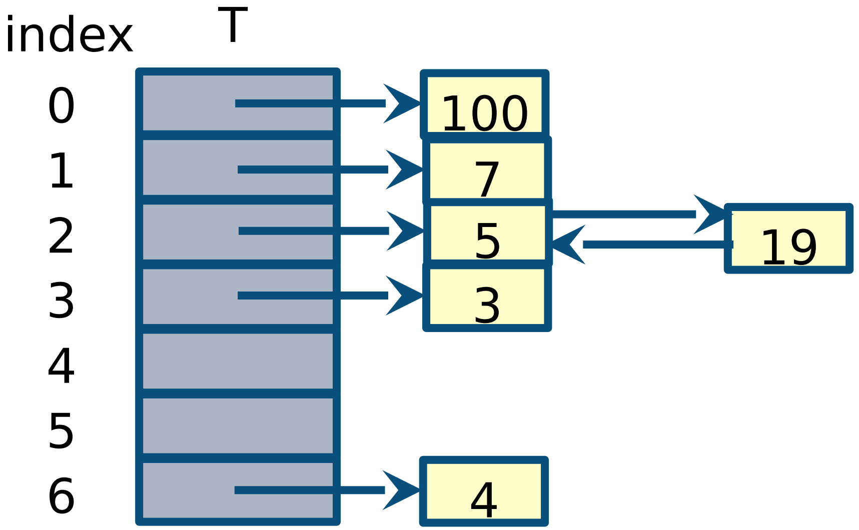
h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1



Question 2: solution (cont.)

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using linear probing**

Hashes

h(100) = 0

h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1

index	T
0	100
1	
2	
3	
4	
5	
6	

Question 2: solution (cont.)

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using linear probing**

Hashes

h(100) = 0

h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1

index	T
0	100
1	
2	19
3	
4	
5	
6	

Question 2: solution (cont.)

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using linear probing**

Hashes

h(100) = 0

h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1

index	T
0	100
1	
2	19
3	3
4	
5	
6	

Question 2: solution (cont.)

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using linear probing**

Hashes

h(100) = 0

h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1

index	T
0	100
1	
2	19
3	3
4	5
5	
6	

Question 2: solution (cont.)

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using linear probing**

Hashes

h(100) = 0

h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1

index	T
0	100
1	
2	19
3	3
4	5
5	
6	4

Question 2: solution (cont.)

- **h(x) = (3x+1) mod 7**
- **insert the elements in the sequence 100,19,3,5,4,7 into H**
- **assuming collisions are handled using linear probing**

Hashes

h(100) = 0

h(19) = 2

h(3) = 3

h(5) = 2

h(4) = 6

h(7) = 1

index	T
0	100
1	7
2	19
3	3
4	5
5	
6	4