Data Structures (15B11CI311)

Odd Semester 2020

3rd Semester , Computer Science and Engineering

Jaypee Institute Of Information Technology (JIIT), Noida

Lecture: 18-20

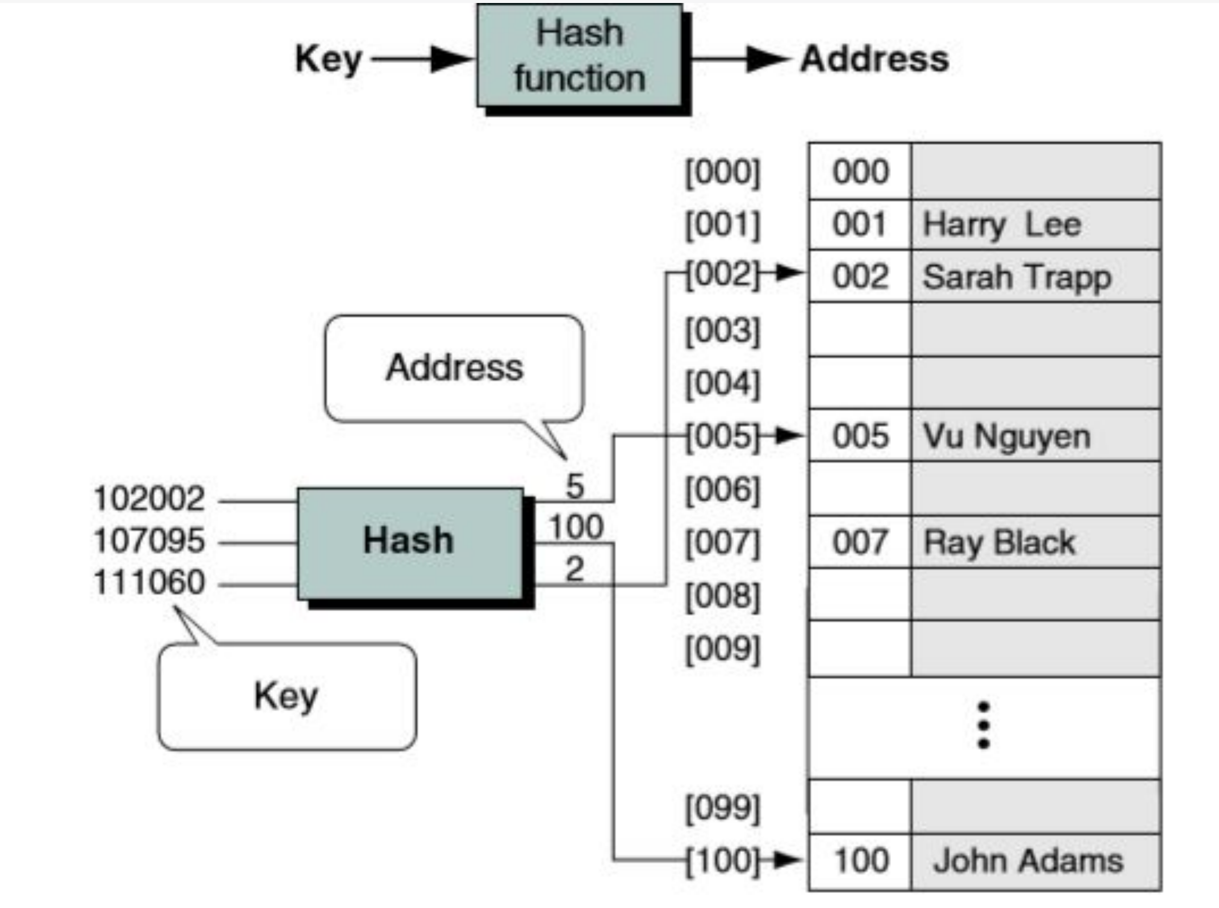
Topics to be covered:

- Hashing concept
- Hash Table
- Hash Function
- Collision Resolution Techniques
- Hashing Applications

Hashing

- In linear and binary search, we need to perform several operations to locate a particular target data. The operations are computation of search index, comparison of target with the data at that index and modifying the index in case target is not found.
- In ideal case, we expect to search the target in or fewer steps. One way to achieve this is we should obtain the address where the data is stored.
- **Hashing** is a technique to find the address where the data is located with the help of a key by using mathematical function called hash function.
- The process of mapping key to address is called as hashing.

Hashing Concept



Source: Gilberg, Richard F., and Behrouz A. Forouzan. *Data Structures: A pseudocode approach with C*. Nelson Education, 2004.

Components of Hashing

There are four key components of hashing:

1. Hash Table
2. Hash Functions
3. Collisions
4. Collision Resolution Techniques

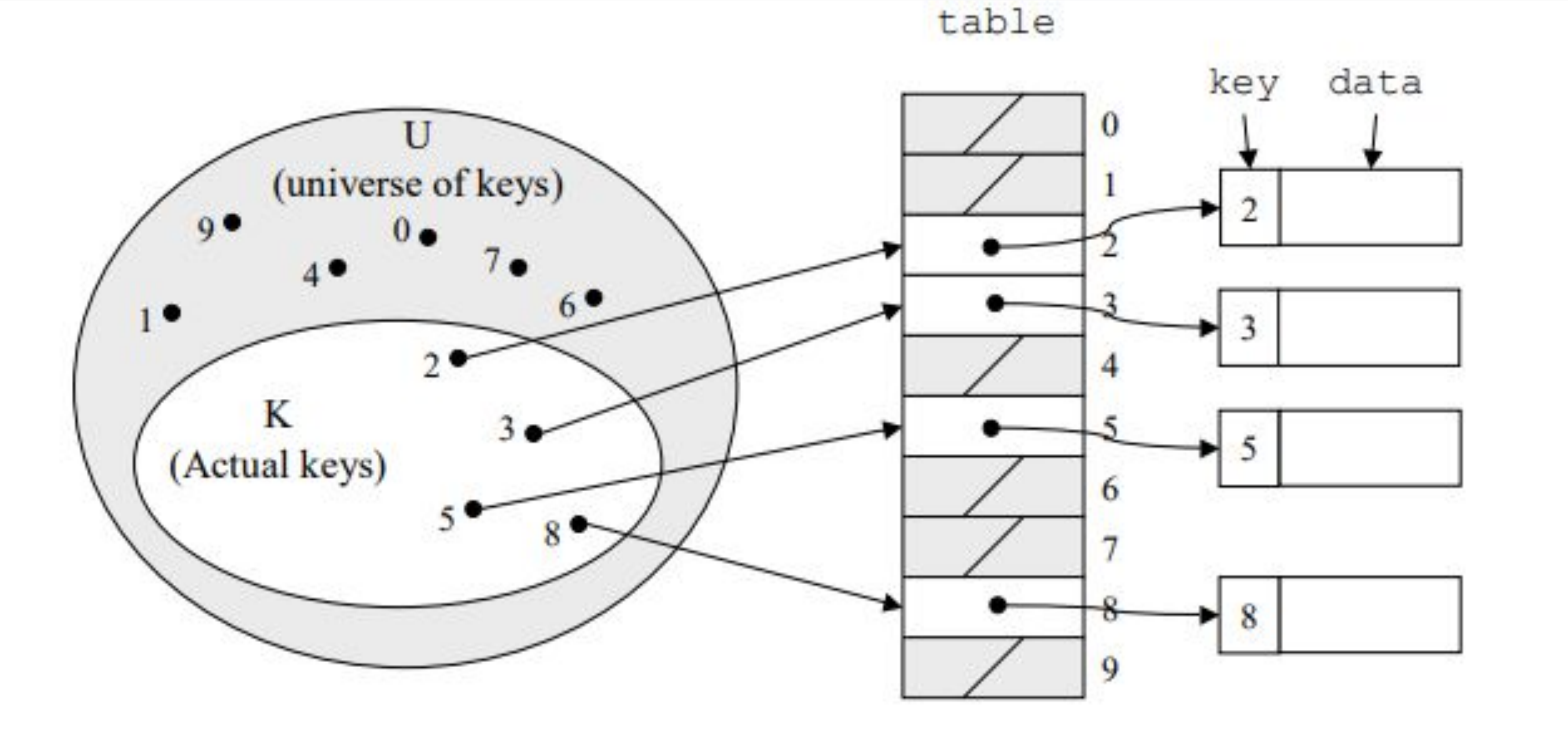
Hash Table

- Hash table is a data structure which stores keys and their associated values and it uses hash function for mapping keys with their associated values.
- Hash table is an array [0 to M-1] having size *M*.
- Insertion, deletion and search operations in hashing can be performed with constant time complexity (i.e. O(1)).

Direct Addressing

- Using an array, the element with key k is stored at a position k of the array. That means, given a key k, we find the element whose key is k by just looking in the kth position of the array. This is called **direct addressing**.
- Key values are assumed to be unique.
- Each key is taken from universe U = {0, 1, . . . , n - 1}
- Direct Address Table is represented as an array T[0 . . . n - 1]

Direct Addressing

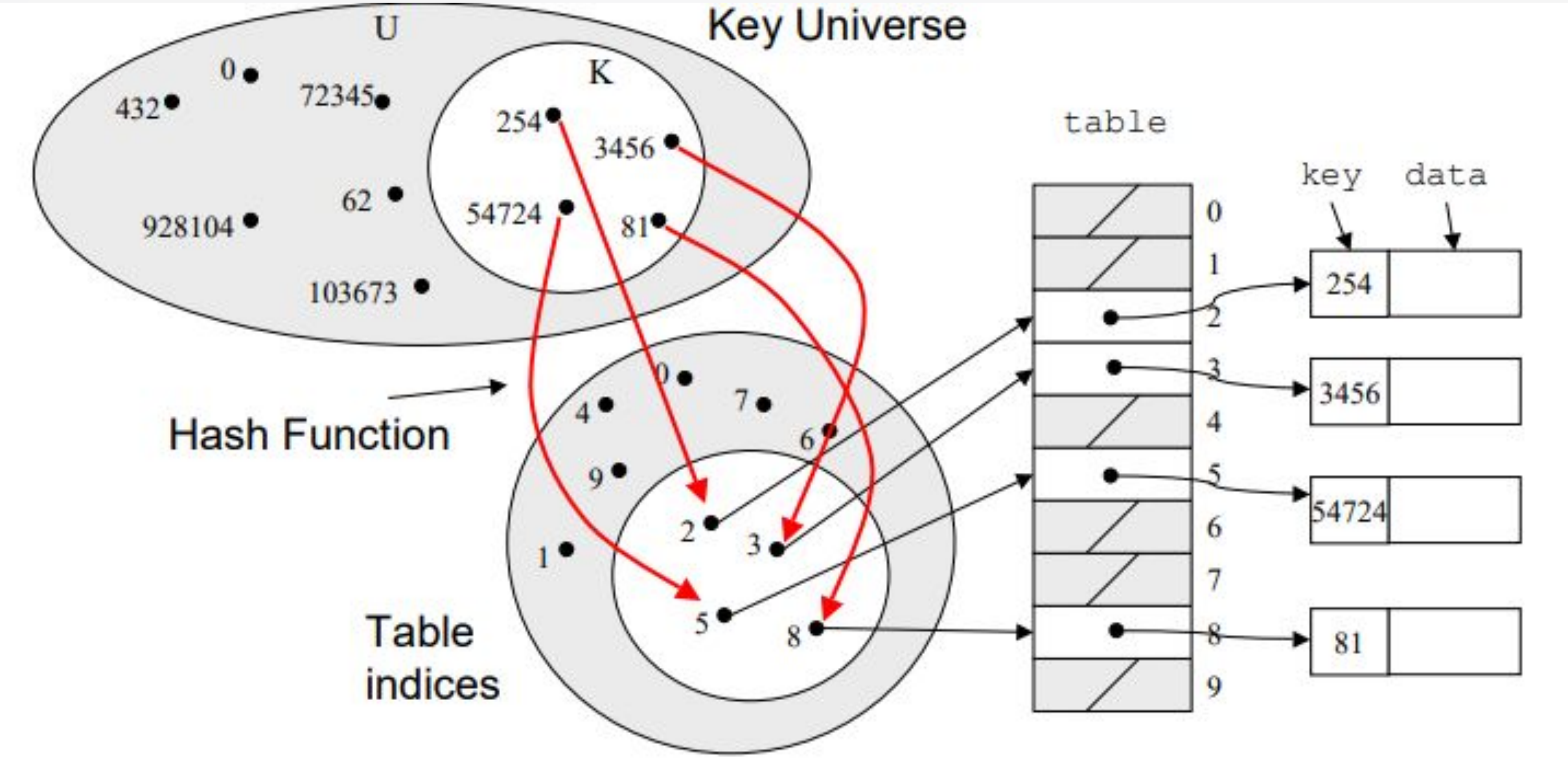


Source: Cormen, Thomas H., et al. Introduction to algorithms. MIT press, 2009.

Hash Table

- The problem with Direct addressing is that if the universe U is large, then to store a table T of size |U| is not possible, considering the limited memory available on a computer.
- In this case one approach is to use hash tables.
- With direct addressing, an element with key k is stored in slot k. With hashing, this element will be stored in the slot h(k) i.e. we use a **hash function** h to map keys to their associated values.

Mapping Keys



Source: Cormen, Thomas H., et al. Introduction to algorithms. MIT press, 2009.

Hash Function

- The hash function is used to map the key into the index of hash table.
- Given a set of elements, a hash function which maps each item into a unique slot is known as a ***perfect hash function***.
- Given a set of elements, unfortunately there is no systematic way to build a perfect hash function.
- Our goal is to create a hash function that minimizes the no. of collisions, which is easy to compute, and evenly distributes the elements in the hash table.

Features of a good Hash Function

- Addresses (or index values) generated from the key are distributed uniformly and randomly.
- If there is small variation in key values then it will cause large variation in indexes so that they are distributed evenly.
- The hash function should minimize the collisions.

Some Hash Function Methods

There are many methods to implement hash functions:

- **Division Method**

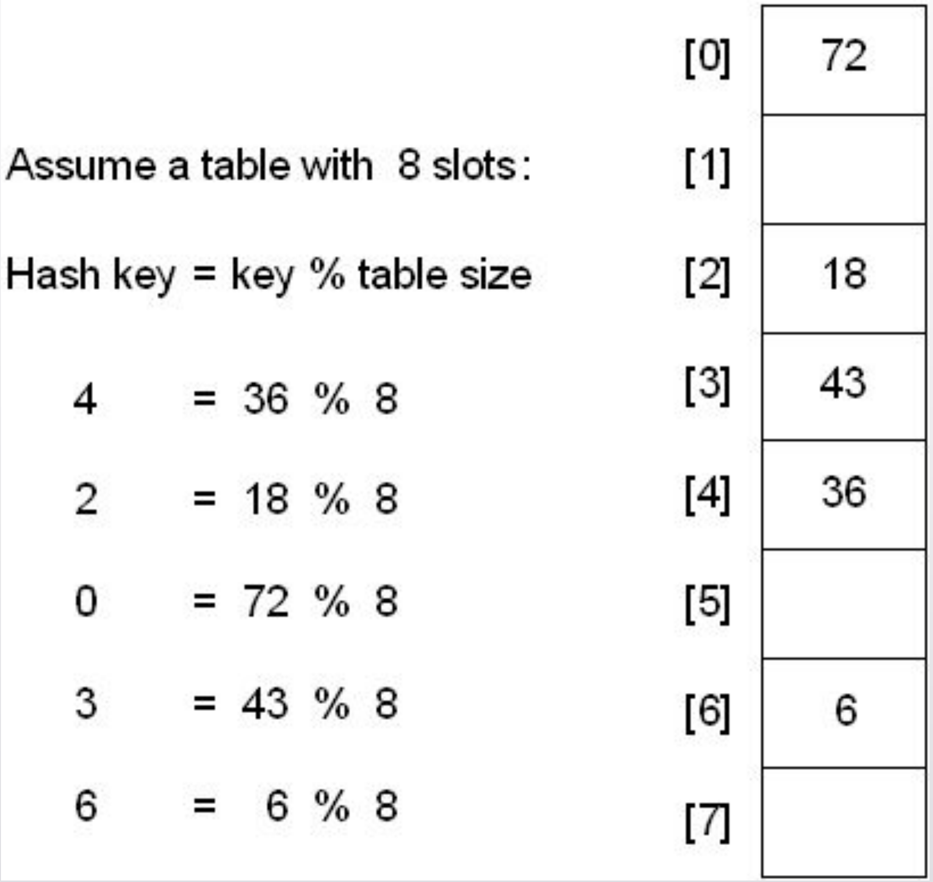
  Hash Function: Hash(Key)= Key % M

  where M is hash table size

Key is divided by M, and the remainder is used as the hash address.

The most appropriate hash table size is prime numbers.

Example



Assume a table with 8 slots:

Hash key = key % table size

```
4    =  36 % 8
2    =  18 % 8
0    =  72 % 8
3    =  43 % 8
6    =   6 % 8
```

[0]	72
[1]	
[2]	18
[3]	43
[4]	36
[5]	
[6]	6
[7]	

Source: http://faculty.cs.niu.edu/~freedman/340/340notes/340hash.htm

Some Hash Function Methods

- **Mid-square Hashing**

Take the square of key and take middle digits of squared key as address.

Eg. Key= 2341 Square= 5480281 Hashed Address= 802

The problem is when key is large. It will be difficult to store the squared value.

So, this method is used when key size is <= 4 digits.

Some Hash Function Methods

- **Folding Method**

In this method, key is divided into subparts of equal length (besides the last part) which are combined or folded to form a new address.

Fold Shift: Key value is divided into various parts of the size of address. Then add those parts.

Fold Boundary: Key value is divided into various parts of the size of address. Then parts are added on fixed boundary.

Eg: Key is 876743

For fold shift, the sum is 87+67+43= 197

Discard 1 , address is 97

For fold boundary, the sum is 78+76+34= 188

Discard 1 , address is 88

Some Terms

- **Load Factor:** It is the no. of elements in hash table divided by size of the table.
- **Probe:** Each action of address calculation and checking for success.
- **Collision:** The two or more keys hashing to the same address is known as collisions.

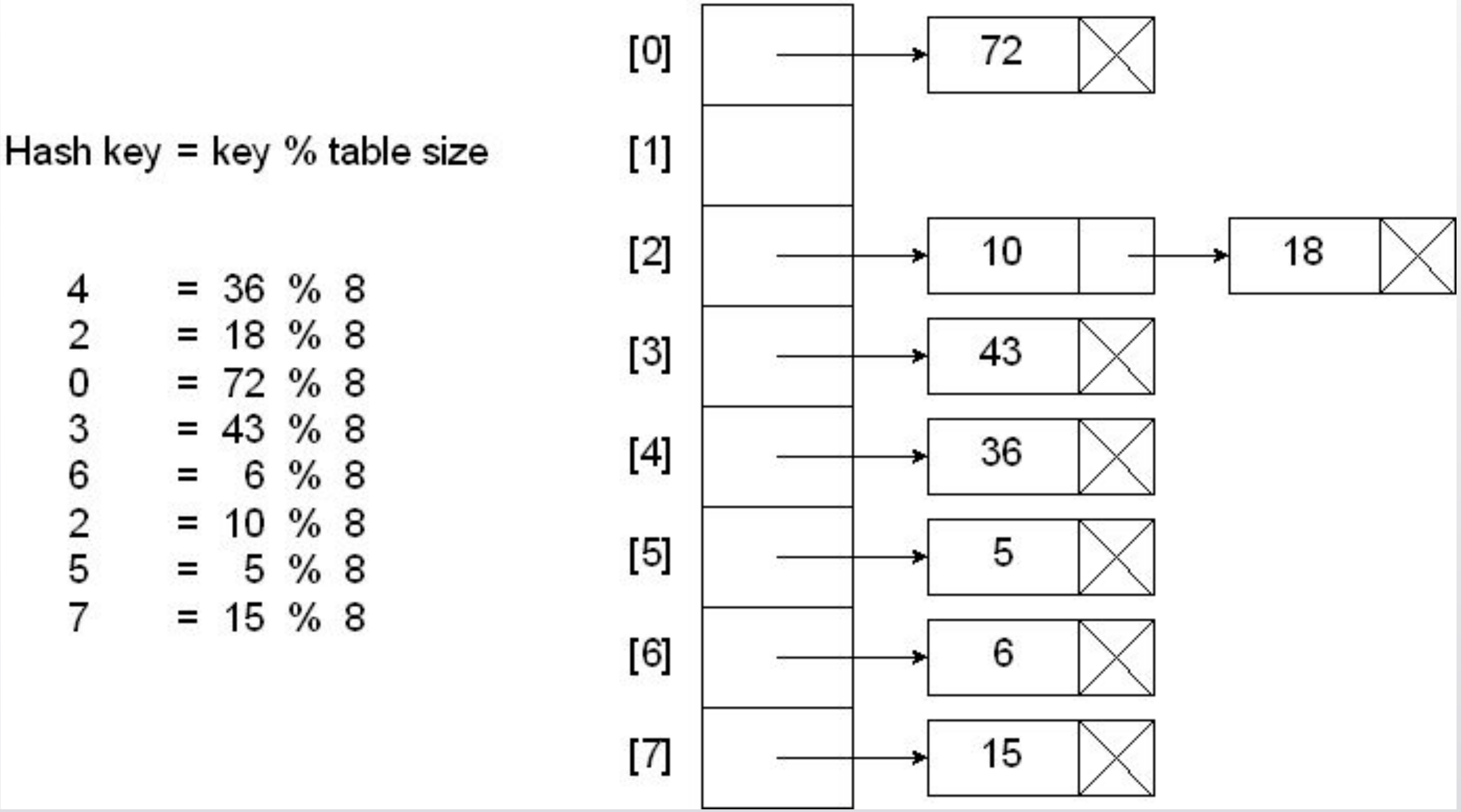
Collision Resolution Techniques

- The process of finding the alternate location is known as collision resolution.
- There are many collision resolution methods:
  - **Direct Chaining:** Linked list implementation
    - Separate chaining
  - **Open Addressing:** Array-based implementation
    - Linear probing (linear search)
    - Quadratic probing (nonlinear search)
    - Double hashing (use two hash functions)

Separate Chaining

- Collision resolution by separate chaining combines linked representation with hash table.
- When two or more records are hashed to the same location, these records are inserted into a singly-linked list called a chain.
- However, pointers are needed to handle to form a chain. The extra memory is required to store pointers.
- More items can be stored than the size of hash table.
- The hash table slot will not hold the key element. They will now store the address of a key element.

Separate Chaining



Hash key = key % table size

4 = 36 % 8
2 = 18 % 8
0 = 72 % 8
3 = 43 % 8
6 = 6 % 8
2 = 10 % 8
5 = 5 % 8
7 = 15 % 8

Source: http://faculty.cs.niu.edu/~freedman/340/340notes/340hash.htm

Separate Chaining

- When a new element is hashed to same address, it can be inserted either at the front or at the end of the list.
- New elements are sometimes inserted at the front of the list, since it may be convenient and it happens frequently that the recently inserted elements are more probably to be accessed in near future.
- Average length of linked list= X/Y, where X is number of elements, Y is no. of linked list.
- Analysis of Chaining:
  - If Y is too large, then there can be too many empty chains
  - If Y is too small, then chains can be too long.

Open addressing

- Separate chaining has the disadvantage of requiring pointers. This tends to slow down the algorithm a bit because time is required to allocate new cells, and also it requires the implementation of second data structure.
- Open addressing is also known as closed hashing.
- In open addressing method, all the keys are stored in the hash table itself.
- A collision is resolved by probing.
- If a collision occurs, then hash table is examined or probed until an available empty slot is found.

Open Addressing

- With open addressing, for every key k, the probe sequence is $h(k)+C(0), h(k)+C(1), \ldots\ldots.. h(k)+C(i)$

where h(k) is the hashing function, C(i) is the probing function with i^{th} probe.

- The probe sequence depend on the key to be inserted.
- The three common collision resolution techniques:
  - Linear Probing
  - Quadratic probing
  - Double hashing

Linear Probing

- In linear probing, the collision is resolved by storing the element in the next empty slot.
- It is implemented by using **linear search** for an empty slot.
- When we reach the end of table, the search is wrapped around to start.
- The function used for linear probing is as follows:

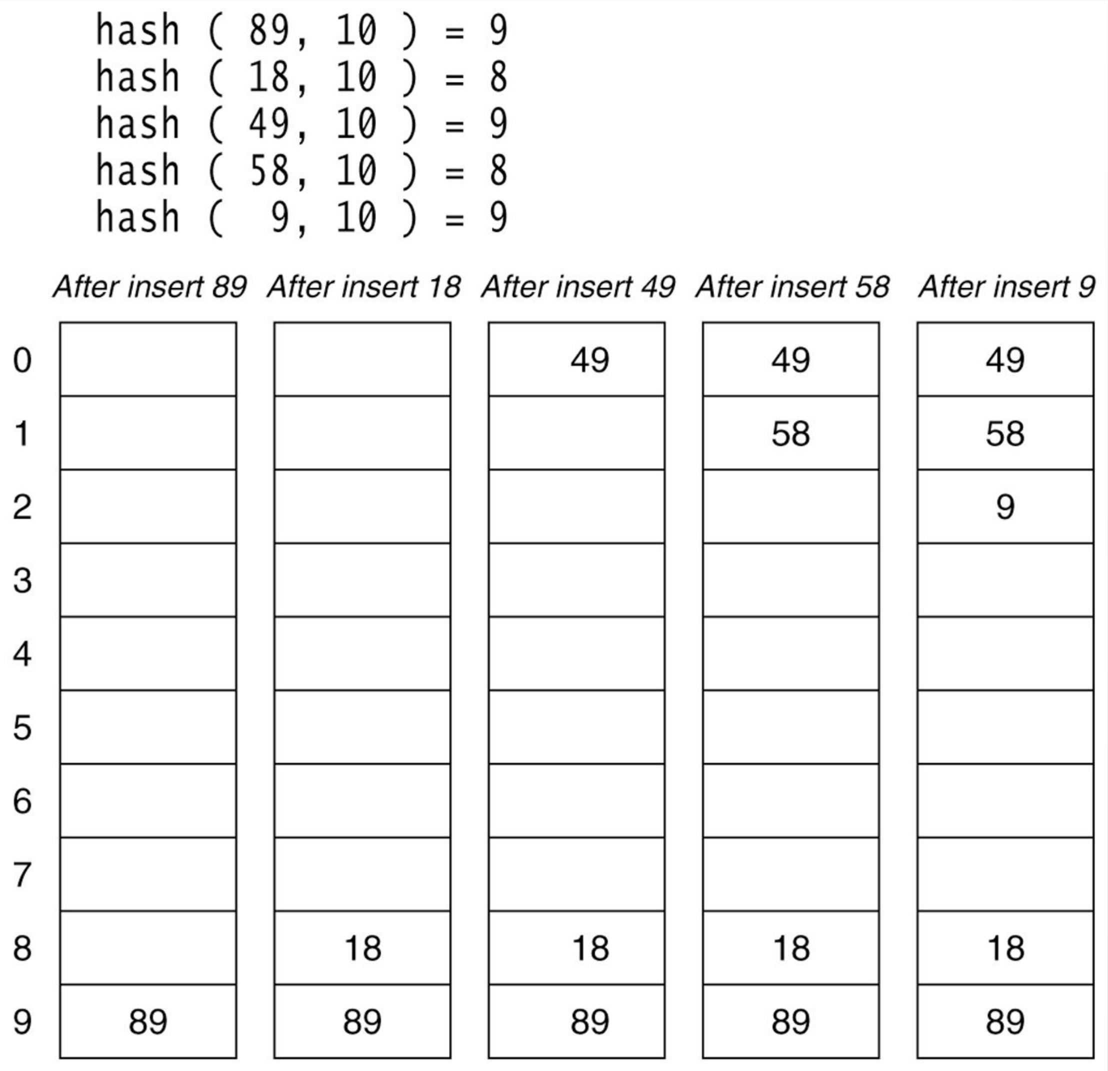
  (H(k)+p(i)) MOD M

  As p(i)=i for linear probing, the function becomes

  (H(k)+i) MOD M

  where H(k)=k MOD M, M is table size, i={0,1,2,… M-1}

Example



```
hash ( 89, 10 ) = 9
hash ( 18, 10 ) = 8
hash ( 49, 10 ) = 9
hash ( 58, 10 ) = 8
hash (  9, 10 ) = 9
```

	After insert 89	*After insert 18*	*After insert 49*	*After insert 58*	*After insert 9*
0			49	49	49
1				58	58
2					9
3					
4					
5					
6					
7					
8		18	18	18	18
9	89	89	89	89	89

Source: Weiss, Mark Allen. "Data Structures And Problem Solving Using C+."

Linear Probing

- The **search algorithm** follows the same probe sequence as **insert algorithm**.
- If an empty slot is reached then, the item is not found
- Otherwise, the match is found.
- Example,
- We have to search 58 then we will start at slot 8. The item at slot 8 is not 58, so we will try slot 9. Again, 58 is not at slot 9 , so we will try at slot 0 and then slot 1 until we the item is found. (4 probes)
- Now, we have to search 19, we will try slots 9, 0, 1, and 2 before the empty slot is reached at 3. Thus 19 is not found. (5 probes)

Linear Probing

- Standard deletion can’t be performed here because, an element in the hash table not only represents itself, but it also serves as a placeholder and connects other elements during collision resolution.
- Suppose, if we want to delete 89 from the table, then all the remaining search operations might fail.
- **Lazy deletion** is implemented. That means elements are marked as deleted rather than removing them physically from the hash table. This information can be stored as an extra data member. Each element can be either active or deleted.

Linear Probing

- One of the problems with linear probing is that elements tend to cluster together in the hash table. It means that the table contains blocks of consecutively occupied cells that is called **primary clustering**.
- Clusters can get close to each other, and can form a larger cluster. Therefore, one part in the table might be quite dense, and another part might be having relatively few items.
- Any key which is hashed into cluster will require excessive attempts for collision resolution causing long probe searches therefore results in poor overall efficiency.

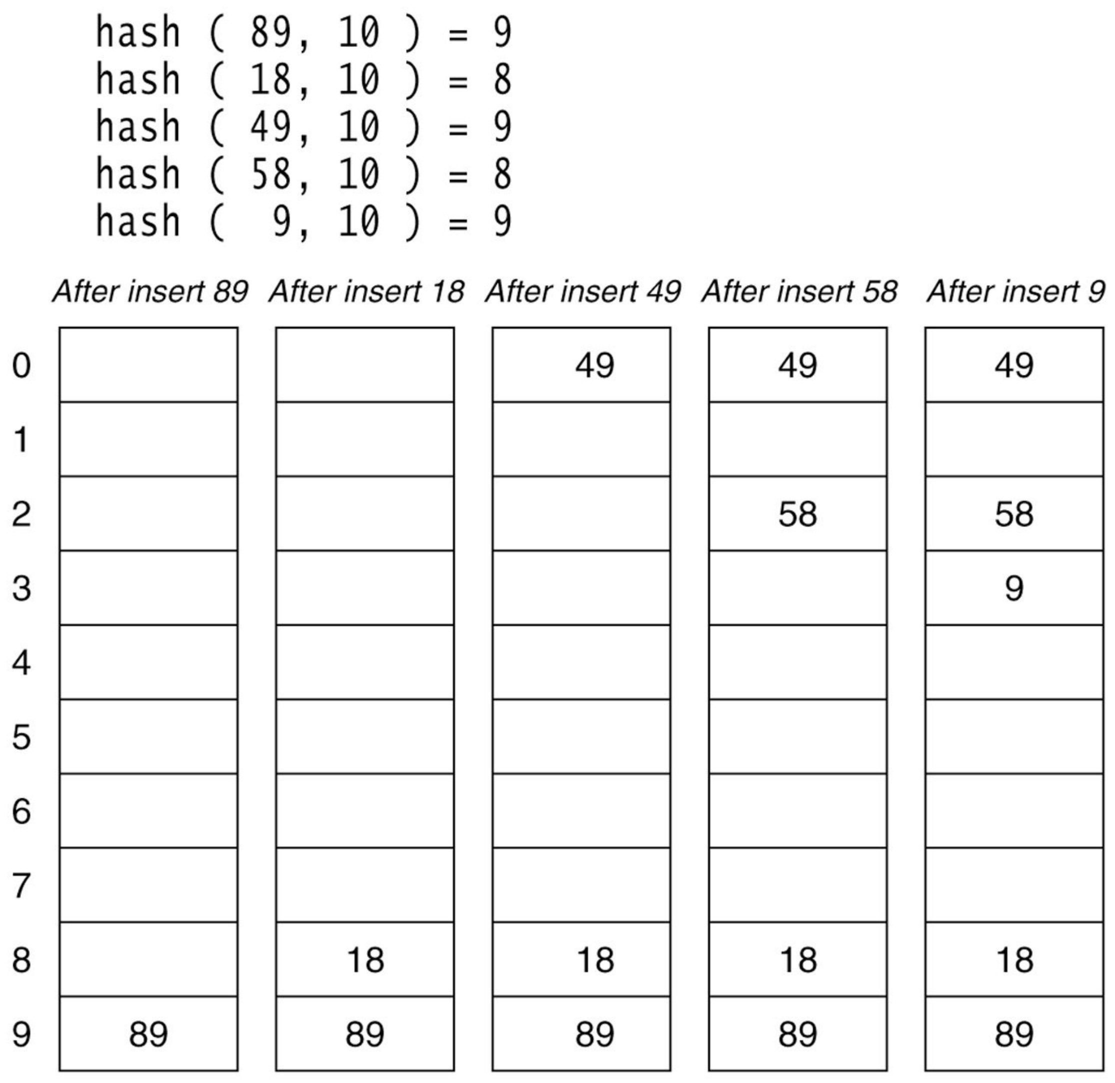
Quadratic Probing

- The primary clustering can be eliminated by using Quadratic Probing method by examining certain slots away from the original probed position.
- Here, Collision Function is $F(i)=i^2$ is quadratic.
- That means subsequent probes are quadratic no. of positions away from original probe point.
- If we are searching for key k, then we will check H(k), $H(k)+ 1^2$, $H(k)+2^2$…. $H(k)+i^2$ (wraparound) until k is found or an empty slot is found.
- The function used for quadratic probing is as follows:

$$\mathbf{(H(k)+i^2)\ MOD\ M}$$

where H(k)=k MOD M, M is table size

Example



```
hash ( 89, 10 ) = 9
hash ( 18, 10 ) = 8
hash ( 49, 10 ) = 9
hash ( 58, 10 ) = 8
hash (  9, 10 ) = 9
```

	After insert 89	*After insert 18*	*After insert 49*	*After insert 58*	*After insert 9*
0			49	49	49
1					
2				58	58
3					9
4					
5					
6					
7					
8		18	18	18	18
9	89	89	89	89	89

Source: Weiss, Mark Allen. "Data Structures And Problem Solving Using C+."

Quadratic Probing

- Does quadratic probing guarantee that if we are inserting K and the hash table is not full, will K be inserted?
- If the hash table size is prime no. and load factor doesn’t exceeds 0.5, then we have a guarantee that there is success in insertion of a new item K.
- But even if the table size is one more than half full then insertion could fail.

Analysis of Quadratic Probing

- In quadratic probing, items which hash to the same address will follow the same path or alternative cells, which is called as **secondary clustering**.
- Technique which can remove secondary clustering is **double hashing**.

Double Hashing

- When a collision occurs, we apply a second hash function in double hashing.
- The function used for double hashing is as follows:

  H(k,i)= (H1(k) + i H2(k)) mod M

  where , M is table size and i=0,1,...
- The size of hash table should be a prime number.

Example

Given, Table size=11, h1(k) = k Mod 11, h2(key) = 7- (k mod 7)

Insert following keys: 47, 14, 91, 25

$H(k,i) = (H_1(k) + i\ H_2(k)\ )\ \text{mod}\ 11$

H(47,0)= 47 mod 11 = 3

H(14,0)=14 mod 11 = 3

H(14,1)= (3 + 7- (14 mod 7)) mod 11 = 10

H(91,0)= 91 mod 11 = 3

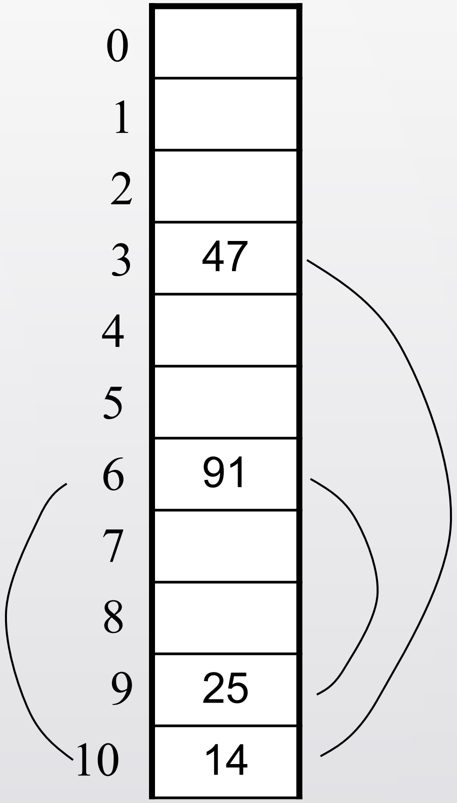
H(91,1)= (3 +7 –(91 mod 7)) mod 11 =10

H(91,2)= 3+ 2* 7 mod 11 = 6

H(25,0)= 25 mod 11 = 3

H(25,1)=(3 + 7- (25 mod 7)) mod 11=6

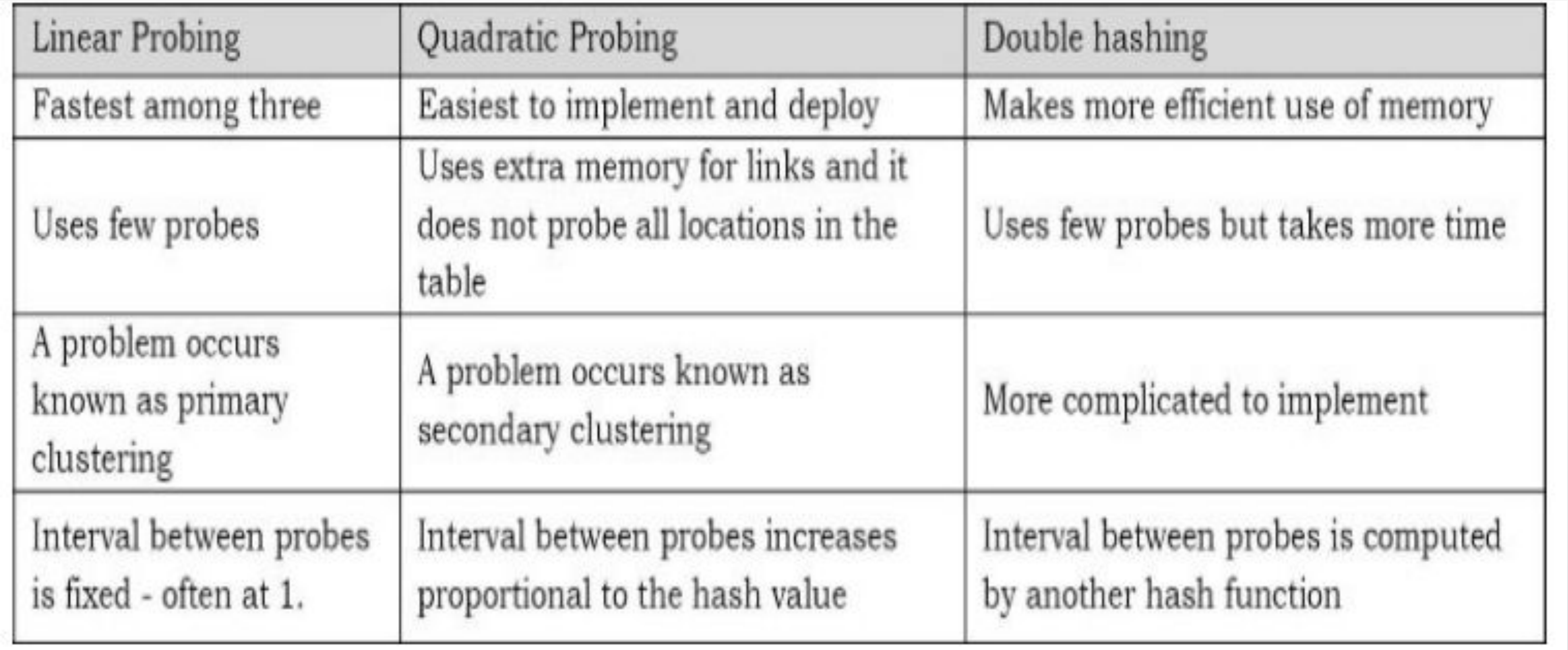
H(25,2)= (3 + 2 * 7- (25 mod 7)) mod 11= 9



Double Hashing

- When table size is prime or power of 2, it improves over linear or quadratic probing as max m^2 probe sequences are used in double hashing, rather than m, since each h1(k),h2(k) pair yields a different probe sequence.
- However, double hashing is somewhat more complicated to implement than quadratic probing.

Comparison of Open Addressing Methods



Linear Probing	Quadratic Probing	Double hashing
Fastest among three	Easiest to implement and deploy	Makes more efficient use of memory
Uses few probes	Uses extra memory for links and it does not probe all locations in the table	Uses few probes but takes more time
A problem occurs known as primary clustering	A problem occurs known as secondary clustering	More complicated to implement
Interval between probes is fixed - often at 1.	Interval between probes increases proportional to the hash value	Interval between probes is computed by another hash function

Source: Karumanchi, Narasimha. *Data Structures and Algorithms Made Easy*

Hashing Applications

- Hash tables are used by compilers to keep track of variables declared in source code. The data structure is known as symbol table.
- Another application is game programs. As the program look through different lines of play, it monitors positions that it has experienced by computing a hash function on the basis of position
- Hashing is also used in online spelling checkers.

Exercise

- Consider the keys {2451, 8253, 6173, 7389, 4344, 9699, 1889} and H(k)=k Mod 10. Show the Hash table for the following:

1. Open Addressing using linear probing.

2. Open Addressing using quadratic probing.

3. Open Addressing using double hashing, h2(k)=7- (k mod 7).

Solution

	Linear probing	Quadratic Probing	Double Hashing
0	9699	9699	1889
1	2451	2451	2451
2	1889		9699
3	8253	8253	8253
4	6173	6173	6173
5	4344	4344	
6			
7			4344
8		1889	
9	7389	7389	7389

References

- Cormen, Thomas H., et al. Introduction to algorithms. MIT press, 2009.
- Karumanchi, Narasimha. *Data Structures and Algorithms Made Easy*
- Weiss, Mark Allen. "Data Structures And Problem Solving Using C+."