Data Structures – CST 201
Module ~ 5

Syllabus

- **Sorting Techniques**
  - Bubble Sort
  - Selection Sort
  - Insertion Sort
  - Merge Sort
  - Quick Sort
  - Heap Sort
- **Hashing**
  - Hashing Techniques
  - Hashing functions: Mid square, Division, Folding, Digit Analysis
  - Collision Resolution
  - Overflow handling

Hash Table

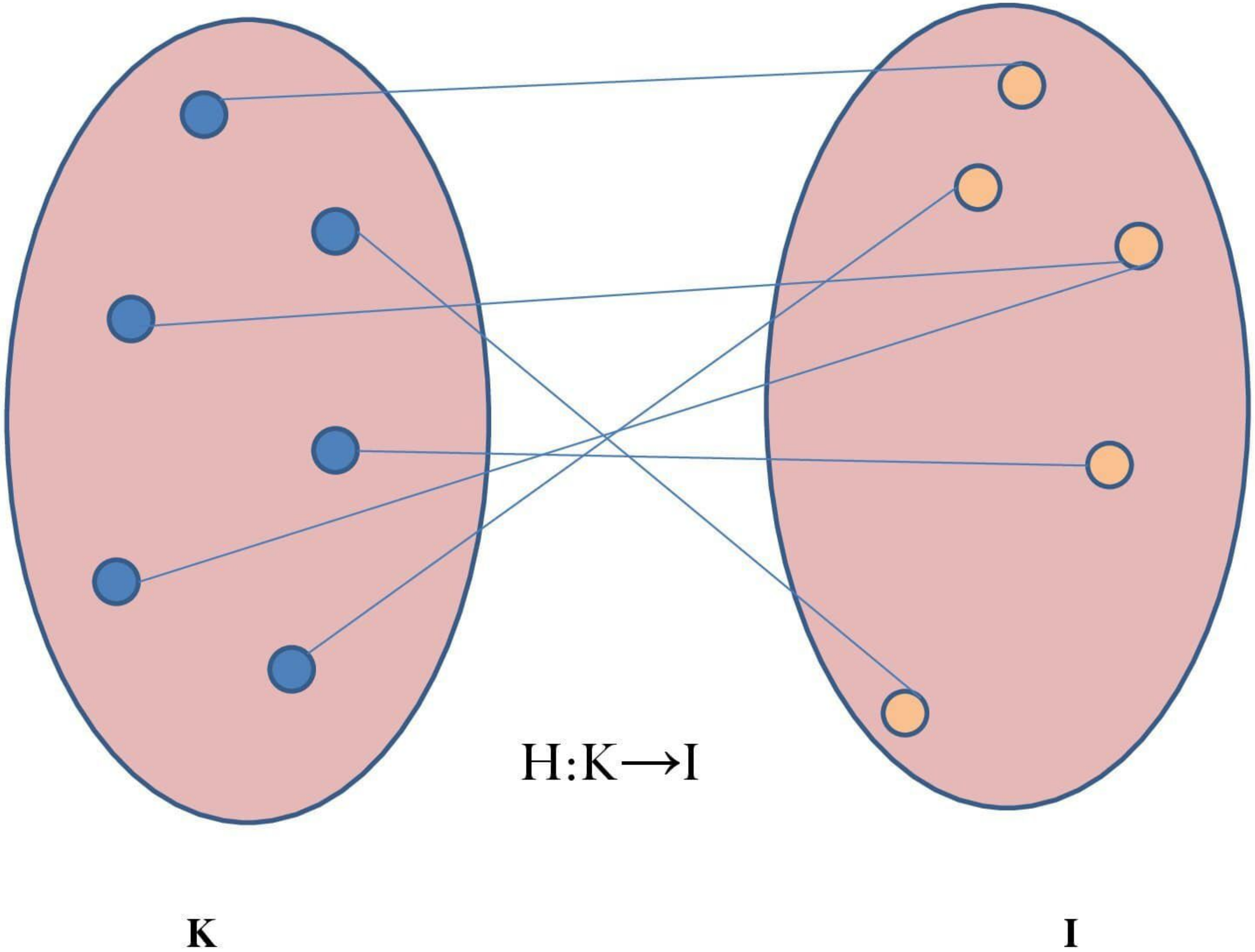
- Hash Table is a data structure which help us to retrieve information very effectively.
- In hash table, data is stored in array format where each data values has its own unique index value.
- Access of data becomes very fast if we know the index of desired data.
- Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of size of data.
- Hash Table uses array as a storage medium and uses hash technique to generate index where an element is to be inserted or to be located from

Hashing

- Hashing is a technique of mapping keys into the hash table by using a hash function

- H: K→I where H is the mapping hash function
  K is the set of key value
  I is the range of indices
- The main idea behind any hashing technique is to find a one-to-one correspondence between a key value and index in the hash table where the key value can be placed.

Hashing



Hashing

- Few criteria while deciding hash function:
  - The function H should be very easy and quick to compute
  - The function H should as far as possible give two different indices for two different key values. There should be minimum no. of collision

Hash Functions

- Some popular hash functions are:
  1. Division method
  2. Mid square method
  3. Folding method
  4. Digit Analysis method

Hash Functions

1. **Division method**
   - The hash function H is
     - $H(k) = k \% m$

k is the key value

m is the size of the hash table

m is usually a prime number or a number without small divisors. This will minimize the number of collisions

- Example: k=102 and m=23

$$H(k) = k \% m = 102 \% 23 = 10$$

Hash Functions

1. <u>Division method</u>
   - **Advantages:** Fastest and easiest hash function
   - **Disadvantage:** Have to avoid certain value of m
   - **Good choice for m** :
     - Prime numbers
     - Not too close to power of 2 or 10

- Example: Assume that the hash table size is 8. Consider the key values 10,21,4,41,24,14,17,15,31. Apply division method and find the index of each key value.

- Ans: Here the hash function is H(k) = k%8

Key	Hash Function	Index
10	10%8	2
21	21%8	5
4	4%8	4
41	41%8	1
24	24%8	0
14	14%8	6
17	17%8	1
15	15%8	7
31	31%8	7

Hash Table

Index	Key	
0	24	
1	41,17	Collision
2	10	
3		
4	4	
5	21	
6	14	
7	15,31	Collision

Hash Functions

2. <u>**Mid square method**</u>

- The hash function H is
  - H(k) = x
- x is obtained by extracting some digits from the middle of k^2

k	k^2	H(k)	
65	4225	22	Extract 2 digits
90	8100	10	
150	22500	50	

Hash Functions

3. **Folding method**

- Here the key k is partitioned in to number of parts, $k_1, k_2 \ldots k_r$
- Each part except possibly the last, has the same no of digits as the required address.
- These parts are added together ignoring the last carry.
- The hash function $H(k) = k_1 + k_2 + \ldots K_r$

k	H(k)	Index
3407	34+07 = 41	41
9020	90+20=110	10
15887	15+88+9=112	12

Hash Functions

3. **Folding method**
   - Three variation of folding
     - Pure folding
     - Fold shifting
     - Fold boundary

Hash Functions

4. **Digit Analysis method**

- Here, the hash address is formed by extracting the digits and rearranging them.
- Example: key-6732541 can be transformed to hash function 427 by extracting digits in even positions and then reversing the combination.
- The decision for extraction and rearrangement is based on some analysis.
- Here, an analysis is performed to determine which key position should be used in forming hash address.
- For each criterion, hash addresses are calculated and then a graph is plotted, then that criterion is selected which produces the most uniform distribution

Collision Resolution Techniques

- Complete removal of collision is almost impossible
- Two collision resolution techniques are:
  - **Closed hashing \ Open Addressing**
  - **Open hashing \ Closed Addressing \ Chaining**

Closed Hashing(Open Addressing)

- The simplest method to resolve a collision.
- Here all elements are stored in the hash table itself. That's why it is called **Open addressing**.
- The hash table is considered circular. That's why the technique is termed **closed hashing**.
- The size of the hash table is greater than or equal to the total number of keys

Closed Hashing(Open Addressing)

- **Insert(k)**: Keep probing(searching) until an empty slot is found. Once an empty slot is found, insert k.
- **Search(k)**: Keep probing until slot's key become equal to k or an empty slot is reached or the search has reach the location where it is started.
- **Delete(k)**:
  - Delete the key
  - The slots of deleted keys are marked as "deleted".

- Note: The insert can insert an item in a deleted slot, but the search doesn't stop at a deleted slot.

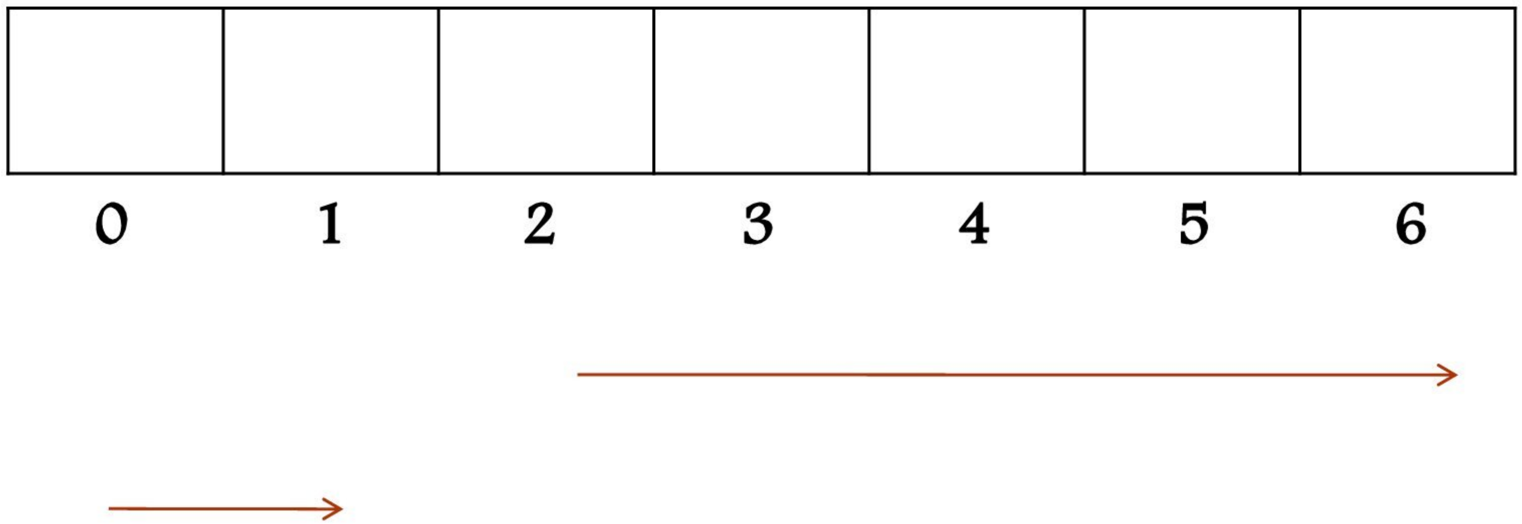
Closed Hashing(Open Addressing)

- Closed hashing is done in the following ways:
  - **Linear Probing**
  - **Quadratic Probing**
  - **Double Hashing**

Linear Probing

Insert the key at the first free location from (u+i)%m, where i=0 to m-1

Here u is the hash function and m is usually the hash table size.



Linear Probing

- **Insertion:**
  - Suppose there is a hash table of size ‘m’ and a key value is to be mapped to an index ‘i’.
  - If the index i is free, then insert the key to index i.
  - Otherwise check the subsequent locations one by one and insert the key at the first free location.(Note that the table is circular)

Linear Probing

- **Searching:**
  - Suppose there is a hash table of size 'm' and a key value is to be mapped to an index 'i'.
  - If the key is present in the index i, then the search is success.
  - Otherwise check the subsequent locations one by one until
    - Key value is found
    - An unoccupied (empty)location is encountered.
    - The search has reach the location where it is started.

Linear Probing

- **Deletion:**
  - Suppose there is a hash table of size ‘m’ and a key value is to be mapped to an index ‘i’.
  - If the key is not present in the index i, then check the subsequent locations one by one.
  - If the search key is found, then delete that key and mark this slot as ‘deleted’.

Linear Probing

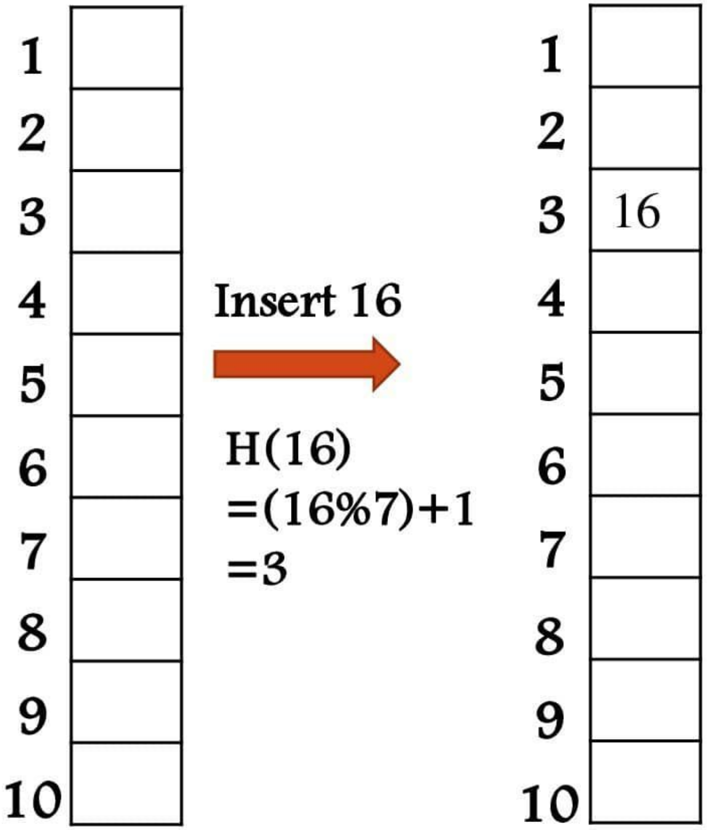
- Example: Consider a hash table of size 10 and a hash function $H(k)=k \bmod 7 + 1$.

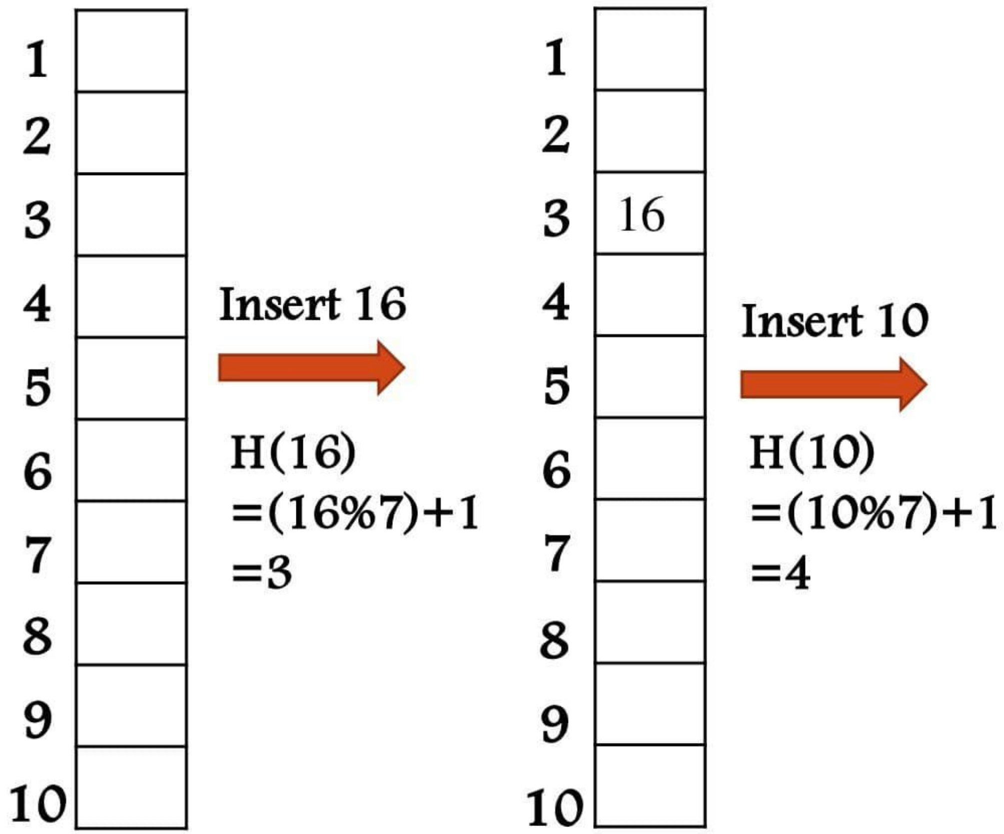
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

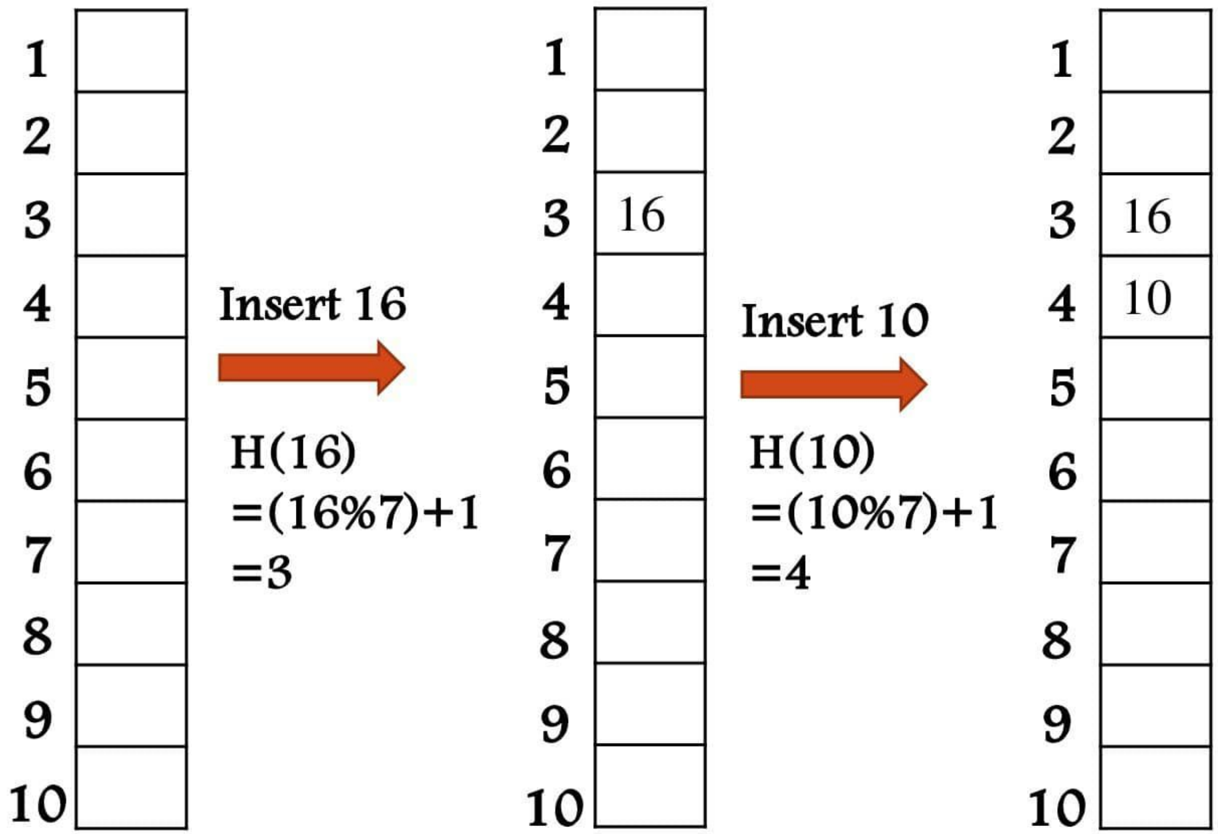
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

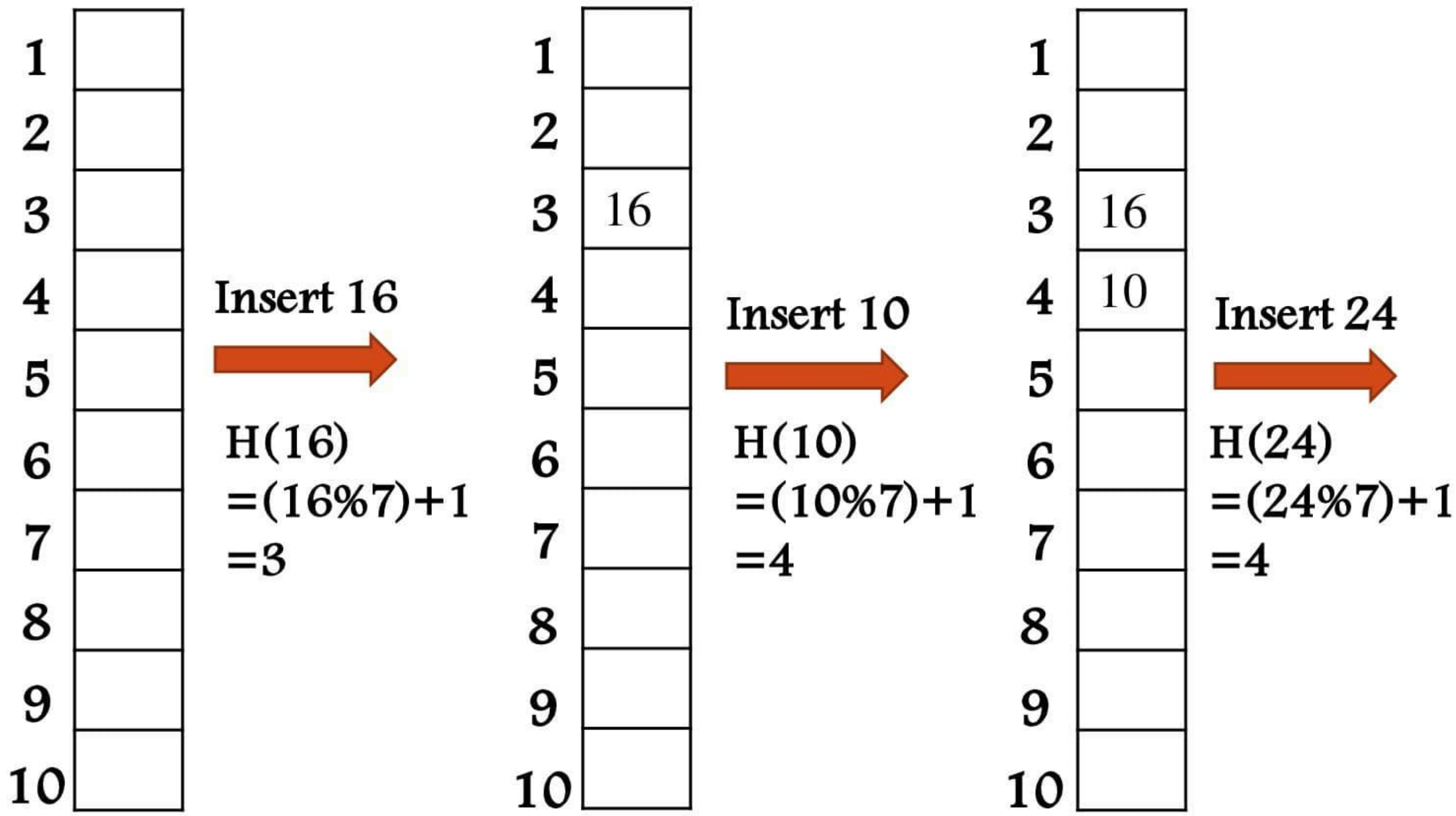
Insert 16

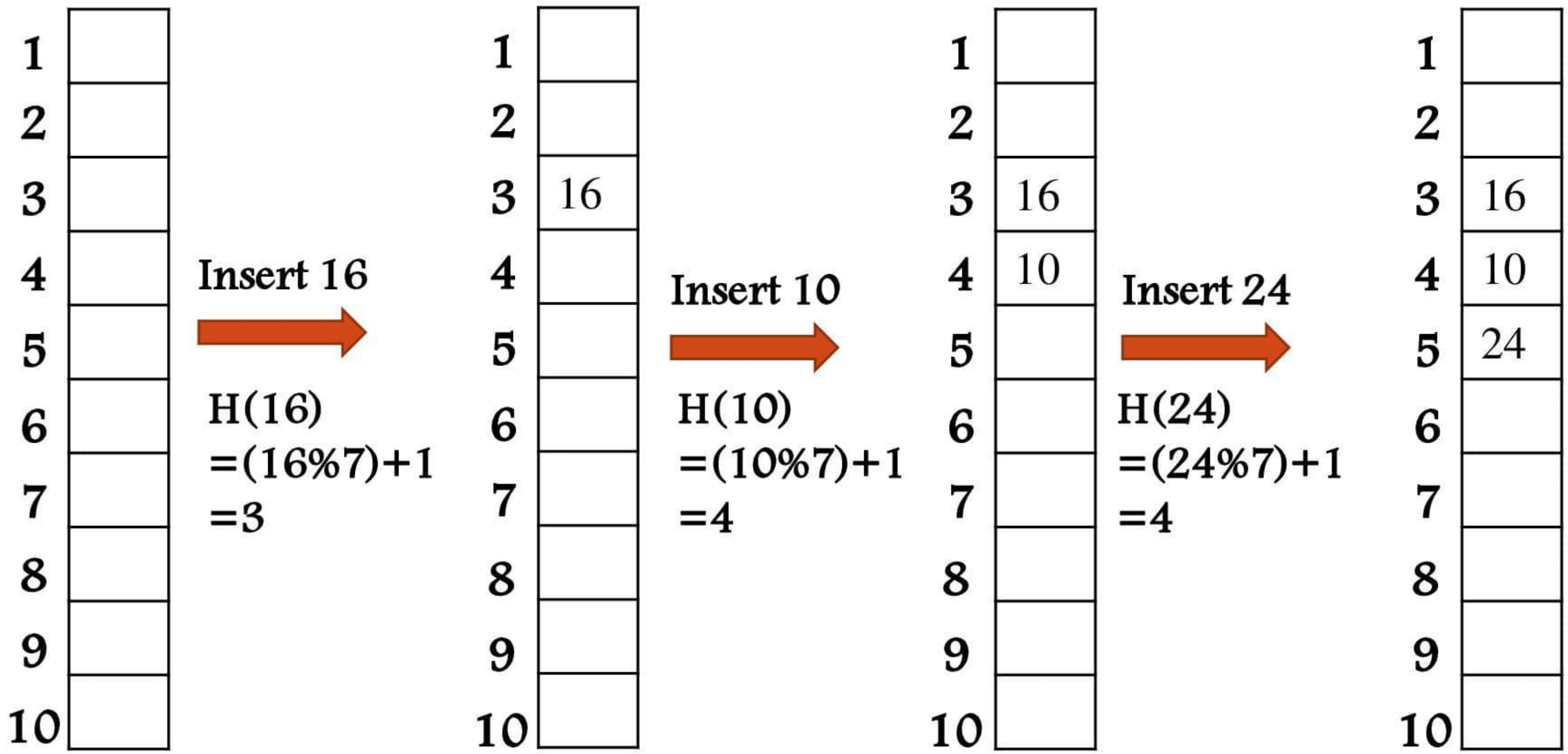
H(16)
=(16%7)+1
=3


1
2
3
4
5
6
7
8
9
10
Insert 16
H(16)
=(16%7)+1
=3
1
2
3 16
4
5
6
7
8
9
10


1
2
3
4
5
6
7
8
9
10
Insert 16
H(16)
=(16%7)+1
=3
1
2
3 16
4
5
6
7
8
9
10
Insert 10
H(10)
=(10%7)+1
=4


1
2
3
4
5
6
7
8
9
10
Insert 16
H(16)
=(16%7)+1
=3
1
2
3 16
4
5
6
7
8
9
10
Insert 10
H(10)
=(10%7)+1
=4
1
2
3 16
4 10
5
6
7
8
9
10


1
2
3
4
5
6
7
8
9
10
Insert 16
H(16)
=(16%7)+1
=3
1
2
3 16
4
5
6
7
8
9
10
Insert 10
H(10)
=(10%7)+1
=4
1
2
3 16
4 10
5
6
7
8
9
10
Insert 24
H(24)
=(24%7)+1
=4


1
2
3
4
5
6
7
8
9
10
Insert 16
H(16)
=(16%7)+1
=3
16
Insert 10
H(10)
=(10%7)+1
=4
10
Insert 24
H(24)
=(24%7)+1
=4
24

Insert 15

H(15)
=(15%7)+1
=2

Insert 15

H(15)
=(15%7)+1
=2

1	
2	15
3	16
4	10
5	24
6	
7	
8	
9	
10	

Insert 15

H(15)
=(15%7)+1
=2

1	
2	15
3	16
4	10
5	24
6	
7	
8	
9	
10	

Insert 6

H(6)
=(6%7)+1
=7

Insert 15

H(15)
=(15%7)+1
=2

1	
2	15
3	16
4	10
5	24
6	
7	
8	
9	
10	

Insert 6

H(6)
=(6%7)+1
=7

1	
2	15
3	16
4	10
5	24
6	
7	6
8	
9	
10	

Insert 15

H(15)
=(15%7)+1
=2

1	
2	15
3	16
4	10
5	24
6	
7	
8	
9	
10	

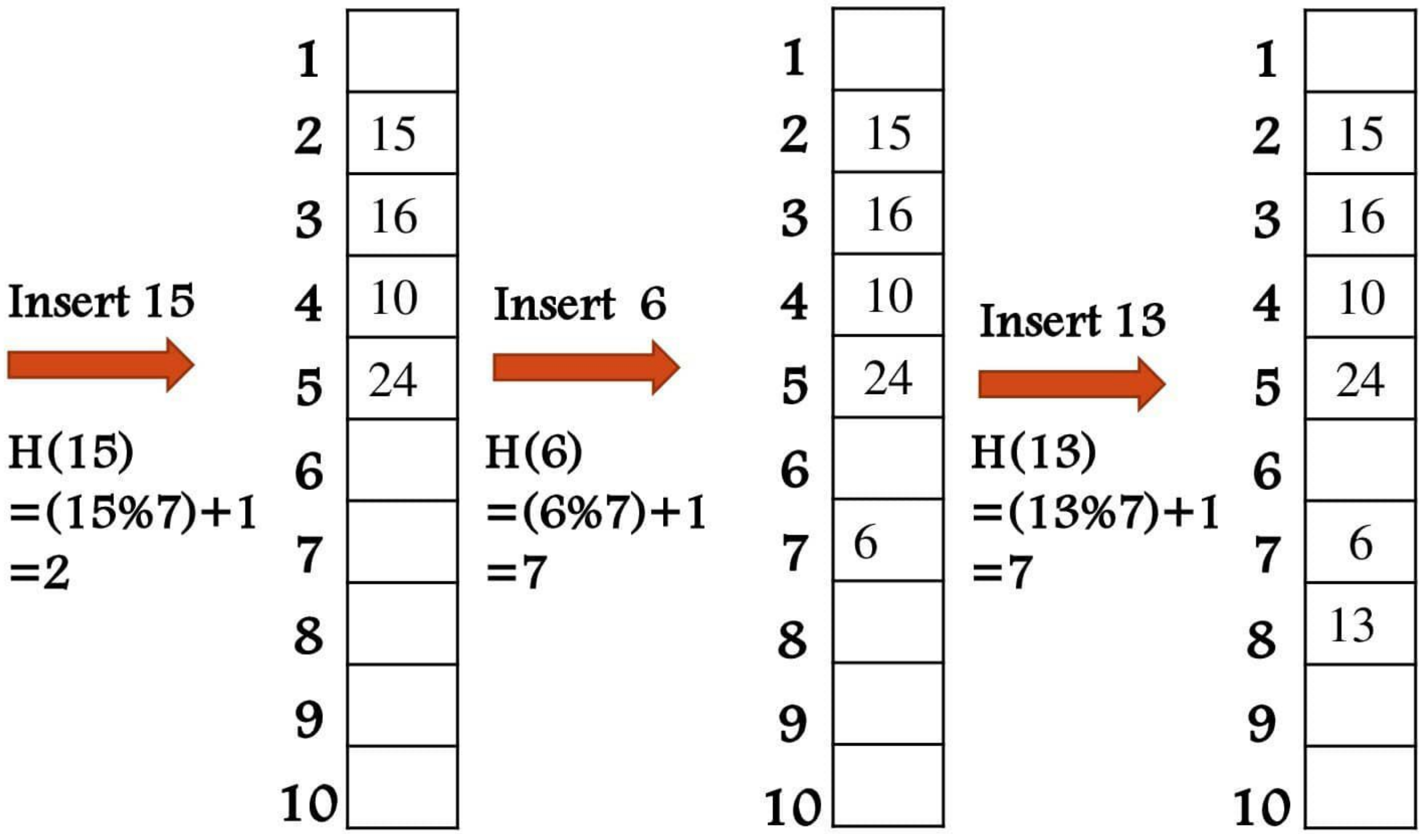
Insert 6

H(6)
=(6%7)+1
=7

1	
2	15
3	16
4	10
5	24
6	
7	6
8	
9	
10	

Insert 13

H(13)
=(13%7)+1
=7


Insert 15
H(15)
=(15%7)+1
=2
1
2 15
3 16
4 10
5 24
6
7
8
9
10
Insert 6
H(6)
=(6%7)+1
=7
1
2 15
3 16
4 10
5 24
6
7 6
8
9
10
Insert 13
H(13)
=(13%7)+1
=7
1
2 15
3 16
4 10
5 24
6
7 6
8 13
9
10

Insert 15

H(15)
=(15%7)+1
=2

1	
2	15
3	16
4	10
5	24
6	
7	
8	
9	
10	

Insert 6

H(6)
=(6%7)+1
=7

1	
2	15
3	16
4	10
5	24
6	
7	6
8	
9	
10	

Insert 13

H(13)
=(13%7)+1
=7

1	
2	15
3	16
4	10
5	24
6	
7	6
8	13
9	
10	

Insert 3

H(3)
=(3%7)+1
=4

Insert 15

H(15)
=(15%7)+1
=2

1	
2	15
3	16
4	10
5	24
6	
7	
8	
9	
10	

Insert 6

H(6)
=(6%7)+1
=7

1	
2	15
3	16
4	10
5	24
6	
7	6
8	
9	
10	

Insert 13

H(13)
=(13%7)+1
=7

1	
2	15
3	16
4	10
5	24
6	
7	6
8	13
9	
10	

Insert 3

H(3)
=(3%7)+1
=4

1	
2	15
3	16
4	10
5	24
6	3
7	6
8	13
9	
10	

Linear Probing

- Example: Consider a hash table of size 10 and a hash function $H(k)=2k + 1$. Use division method and open hashing/linear probing to store the following key values in the hash table.

  key values: 3, 2, 9, 6, 11, 13, 7, 12

Linear Probing

- Example: Consider a hash table of size 10 and a hash function H(k)=2k + 1. Use division method and open hashing/linear probing to store the following key values in the hash table.

  key values: 3, 2, 9, 6, 11, 13, 7, 12

- Ans:
  - Here m=10 and u=2k+1
  - Insert the key at the first free location from (u+i)%m, where i=0 to m-1.

Key	Index	Actual Index
3	$((2x3+1) + 0)\%10 = 7$	7

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3	

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4
13	((2x13+1) + 0)%10 = 7 ((2x13+1) + 1)%10 = 8	8

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1) + 0)\%10 = 7$	7
2	$((2x2+1)+ 0)\%10 = 5$	5
9	$((2x9+1)+ 0)\%10 = 9$	9
6	$((2x6+1) + 0)\%10 = 3$	3
11	$((2x11+1) + 0)\%10 = 3$ $((2x11+1) + 1)\%10 = 4$	4
13	$((2x13+1) + 0)\%10 = 7$ $((2x13+1) + 1)\%10 = 8$	8

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	13
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4
13	((2x13+1) + 0)%10 = 7 ((2x13+1) + 1)%10 = 8	8
7	((2x7+1) + 0)%10 = 5 ((2x7+1) + 1)%10 = 6	6

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	13
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4
13	((2x13+1) + 0)%10 = 7 ((2x13+1) + 1)%10 = 8	8
7	((2x7+1) + 0)%10 = 5 ((2x7+1) + 1)%10 = 6	6

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4
13	((2x13+1) + 0)%10 = 7 ((2x13+1) + 1)%10 = 8	8
7	((2x7+1) + 0)%10 = 5 ((2x7+1) + 1)%10 = 6	6
12	((2x12+1) + 0)%10 = 5 ((2x12+1) + 1)%10 = 6 ((2x12+1) + 2)%10 = 7 ((2x12+1) + 3)%10 = 8 ((2x12+1) + 4)%10 = 9 ((2x12+1) + 5)%10 = 0	0

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4
13	((2x13+1) + 0)%10 = 7 ((2x13+1) + 1)%10 = 8	8
7	((2x7+1) + 0)%10 = 5 ((2x7+1) + 1)%10 = 6	6
12	((2x12+1) + 0)%10 = 5 ((2x12+1) + 1)%10 = 6 ((2x12+1) + 2)%10 = 7 ((2x12+1) + 3)%10 = 8 ((2x12+1) + 4)%10 = 9 ((2x12+1) + 5)%10 = 0	0

	Hash Table
0	12
1	
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Key	Index	Actual Index
3	((2x3+1) + 0)%10 = 7	7
2	((2x2+1)+ 0)%10 = 5	5
9	((2x9+1)+ 0)%10 = 9	9
6	((2x6+1) + 0)%10 = 3	3
11	((2x11+1) + 0)%10 = 3 ((2x11+1) + 1)%10 = 4	4
13	((2x13+1) + 0)%10 = 7 ((2x13+1) + 1)%10 = 8	8
7	((2x7+1) + 0)%10 = 5 ((2x7+1) + 1)%10 = 6	6
12	((2x12+1) + 0)%10 = 5 ((2x12+1) + 1)%10 = 6 ((2x12+1) + 2)%10 = 7 ((2x12+1) + 3)%10 = 8 ((2x12+1) + 4)%10 = 9 ((2x12+1) + 5)%10 = 0	0

	Hash Table
0	12
1	
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Order of elements in the hash table➔ 12, _, _, 6, 11, 2, 7, 3, 13, 9

Linear Probing

- Drawback:
  - Primary Clustering
    - As half of the hash table is filled, there is a tendency towards clustering. That is key values are clustered in large groups, thus sequential search becomes slower and slower.
    - This kind of clustering is called primary clustering.
  - Secondary Clustering
    - If the same sequence of locations is generated for two different keys, then secondary clustering takes place

Quadratic Probing

- Insert the key at the first free location from $(u+i^2)\%m$, where i=0 to m-1
- Here u is the hash function and m is usually the hash table size.

- If there is a collision at location i, then the next locations will be $i+1^2$, $i+2^2$, $i+3^2$ etc etc.

Quadratic Probing

- Example: Consider a hash table of size 10 and a hash function $H(k)=2k + 1$. Use division method and quadratic probing to store the following key values in the hash table.

  key values: 3, 2, 9, 6, 11, 13, 7, 12

Quadratic Probing

- Example: Consider a hash table of size 10 and a hash function $H(k)=2k + 1$. Use division method and quadratic probing to store the following key values in the hash table.

  key values: 3, 2, 9, 6, 11, 13, 7, 12

- Ans: Here m=10

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	3
8	
9	

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	3
8	
9	

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9
6	$((2x6+1) +0^2)\%10 = 3$	3

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$	

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9
6	$((2x6+1) +0^2)\%10 = 3$	3
11	$((2x11+1) +0^2)\%10 = 3$ $((2x11+1) +1^2)\%10 = 4$	4

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$ $((2x11+1)+1^2)\%10 = 4$	4

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$ $((2x11+1)+1^2)\%10 = 4$	4
13	$((2x13+1)+0^2)\%10 = 7$	

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9
6	$((2x6+1) +0^2)\%10 = 3$	3
11	$((2x11+1) +0^2)\%10 = 3$ $((2x11+1) +1^2)\%10 = 4$	4
13	$((2x13+1) +0^2)\%10 = 7$ $((2x13+1) +1^2)\%10 = 8$	8

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9
6	$((2x6+1) +0^2)\%10 = 3$	3
11	$((2x11+1) +0^2)\%10 = 3$ $((2x11+1) +1^2)\%10 = 4$	4
13	$((2x13+1) +0^2)\%10 = 7$ $((2x13+1) +1^2)\%10 = 8$	8

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	13
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1) +0^2)\%10 = 5$	5
9	$((2x9+1) +0^2)\%10 = 9$	9
6	$((2x6+1) +0^2)\%10 = 3$	3
11	$((2x11+1) +0^2)\%10 = 3$ $((2x11+1) +1^2)\%10 = 4$	4
13	$((2x13+1) +0^2)\%10 = 7$ $((2x13+1) +1^2)\%10 = 8$	8
7	$((2x7+1) +0^2)\%10 = 5$	

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	13
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$ $((2x11+1)+1^2)\%10 = 4$	4
13	$((2x13+1)+0^2)\%10 = 7$ $((2x13+1)+1^2)\%10 = 8$	8
7	$((2x7+1)+0^2)\%10 = 5$ $((2x7+1)+1^2)\%10 = 6$	6

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	
7	3
8	13
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$ $((2x11+1)+1^2)\%10 = 4$	4
13	$((2x13+1)+0^2)\%10 = 7$ $((2x13+1)+1^2)\%10 = 8$	8
7	$((2x7+1)+0^2)\%10 = 5$ $((2x7+1)+1^2)\%10 = 6$	6

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$ $((2x11+1)+1^2)\%10 = 4$	4
13	$((2x13+1)+0^2)\%10 = 7$ $((2x13+1)+1^2)\%10 = 8$	8
7	$((2x7+1)+0^2)\%10 = 5$ $((2x7+1)+1^2)\%10 = 6$	6
12	$((2x12+1)+0^2)\%10 = 5$ $((2x12+1)+1^2)\%10 = 6$ $((2x12+1)+2^2)\%10 = 9$ $((2x12+1)+3^2)\%10 = 4$ $((2x12+1)+4^2)\%10 = 1$	1

	Hash Table
0	
1	
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Key	Index	Actual Index
3	$((2x3+1)+0^2)\%10 = 7$	7
2	$((2x2+1)+0^2)\%10 = 5$	5
9	$((2x9+1)+0^2)\%10 = 9$	9
6	$((2x6+1)+0^2)\%10 = 3$	3
11	$((2x11+1)+0^2)\%10 = 3$ $((2x11+1)+1^2)\%10 = 4$	4
13	$((2x13+1)+0^2)\%10 = 7$ $((2x13+1)+1^2)\%10 = 8$	8
7	$((2x7+1)+0^2)\%10 = 5$ $((2x7+1)+1^2)\%10 = 6$	6
12	$((2x12+1)+0^2)\%10 = 5$ $((2x12+1)+1^2)\%10 = 6$ $((2x12+1)+2^2)\%10 = 9$ $((2x12+1)+3^2)\%10 = 4$ $((2x12+1)+4^2)\%10 = 1$	1

	Hash Table
0	
1	12
2	
3	6
4	11
5	2
6	7
7	3
8	13
9	9

Double Hashing

- An alternative method to avoid secondary clustering problem is to use a second hash function in addition to the first one.
- Assume that $H_1(k)$ and $H_2(k)$the two hash functions.

- Insert the key at the first free place from (u+ v x i)%m, where i=0,1,2, ...(m-1)
- Here u is the index generated by $H_1(k)$ and v is the index generated by $H_2(k)$.
- m is usually the size of the hash table.

Double Hashing

- Example: Consider a hash table of size 10 and the two hash functions

  $H_1(k)=2k+1$

  $H_2(k)=3k+2$

  Use division method and double hashing to store the following key values in the hash table.

  key values: 3, 2, 9, 6, 11, 13, 7, 12

- Ans: Here m=10

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5

	Hash Table
0	
1	
2	
3	
4	
5	
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3

	Hash Table
0	
1	
2	
3	
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3	

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	11
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8
13	((2x13+1) + (3x13+2) x 0)%10 = 7 ((2x13+1) + (3x13+2) x 1)%10 = 8 ((2x13+1) + (3x13+2) x 2)%10 = 9 ((2x13+1) + (3x13+2) x 3)%10 = 0	0

	Hash Table
0	
1	
2	
3	6
4	
5	2
6	
7	3
8	11
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8
13	((2x13+1) + (3x13+2) x 0)%10 = 7 ((2x13+1) + (3x13+2) x 1)%10 = 8 ((2x13+1) + (3x13+2) x 2)%10 = 9 ((2x13+1) + (3x13+2) x 3)%10 = 0	0

	Hash Table
0	13
1	
2	
3	6
4	
5	2
6	
7	3
8	11
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8
13	((2x13+1) + (3x13+2) x 0)%10 = 7 ((2x13+1) + (3x13+2) x 1)%10 = 8 ((2x13+1) + (3x13+2) x 2)%10 = 9 ((2x13+1) + (3x13+2) x 3)%10 = 0	0
7	((2x7+1) + (3x7+2) x 0)%10 = 5 ((2x7+1) + (3x7+2) x 1)%10 = 8 ((2x7+1) + (3x7+2) x 2)%10 = 1	1

	Hash Table
0	13
1	
2	
3	6
4	
5	2
6	
7	3
8	11
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8
13	((2x13+1) + (3x13+2) x 0)%10 = 7 ((2x13+1) + (3x13+2) x 1)%10 = 8 ((2x13+1) + (3x13+2) x 2)%10 = 9 ((2x13+1) + (3x13+2) x 3)%10 = 0	0
7	((2x7+1) + (3x7+2) x 0)%10 = 5 ((2x7+1) + (3x7+2) x 1)%10 = 8 ((2x7+1) + (3x7+2) x 2)%10 = 1	1

	Hash Table
0	13
1	7
2	
3	6
4	
5	2
6	
7	3
8	11
9	9

Key	Index	Actual Index
3	((2x3+1) + (3x3+2) x 0)%10 = 7	7
2	((2x2+1) + (3x2+2) x 0)%10 = 5	5
9	((2x9+1) + (3x9+2) x 0)%10 = 9	9
6	((2x6+1) + (3x6+2) x 0)%10 = 3	3
11	((2x11+1) + (3x11+2) x 0)%10 = 3 ((2x11+1) + (3x11+2) x 1)%10 = 8	8
13	((2x13+1) + (3x13+2) x 0)%10 = 7 ((2x13+1) + (3x13+2) x 1)%10 = 8 ((2x13+1) + (3x13+2) x 2)%10 = 9 ((2x13+1) + (3x13+2) x 3)%10 = 0	0
7	((2x7+1) + (3x7+2) x 0)%10 = 5 ((2x7+1) + (3x7+2) x 1)%10 = 8 ((2x7+1) + (3x7+2) x 2)%10 = 1	1
12	((2x12+1) + (3x12+2) x 0)%10 = 5 ((2x12+1) + (3x12+2) x 1)%10 = 3 ((2x12+1) + (3x12+2) x 2)%10 = 1 ((2x12+1) + (3x12+2) x 3)%10 = 9 ((2x12+1) + (3x12+2) x 4)%10 = 7 ((2x12+1) + (3x12+2) x 5)%10 = 5	Cannot insert 12

	Hash Table
0	13
1	7
2	
3	3
4	
5	2
6	
7	3
8	11
9	9

Open Hashing (Closed Addressing\Chaining)

- The closed hashing technique deals with arrays as hash tables.
- Two main difficulties of this method are:
  - Very difficult to handle the situation of table overflow.
  - Key values are intermixed. Thus majority of the key values are far from their hash location, thus increasing the number of probes which degrades the overall performances.
- To resolve these problem, open hashing or chaining method is used.

Open Hashing

- Here hash table consist of array of pointers.
- Each pointer points a linked list.

- **Advantages:**
  - An overflow situation never arises. The hash table maintains lists which can contain any number of key values.
  - Collision resolution can be achieved quickly.
  - Insertion and deletion is fast
  - This method is best suitable where number of key values varies drastically.

- Example: Assume that the hash table size is 8. Consider the key values 10,21,4,41,24,14,17,15,31. Apply division method and open hashing to find the index of each key value.

- Ans: Here the hash function is H(k) = k%8

Key	Hash Function	Index
10	10%8	2
21	21%8	5
4	4%8	4
41	41%8	1
24	24%8	0
14	14%8	6
17	17%8	1
15	15%8	7
31	31%8	7