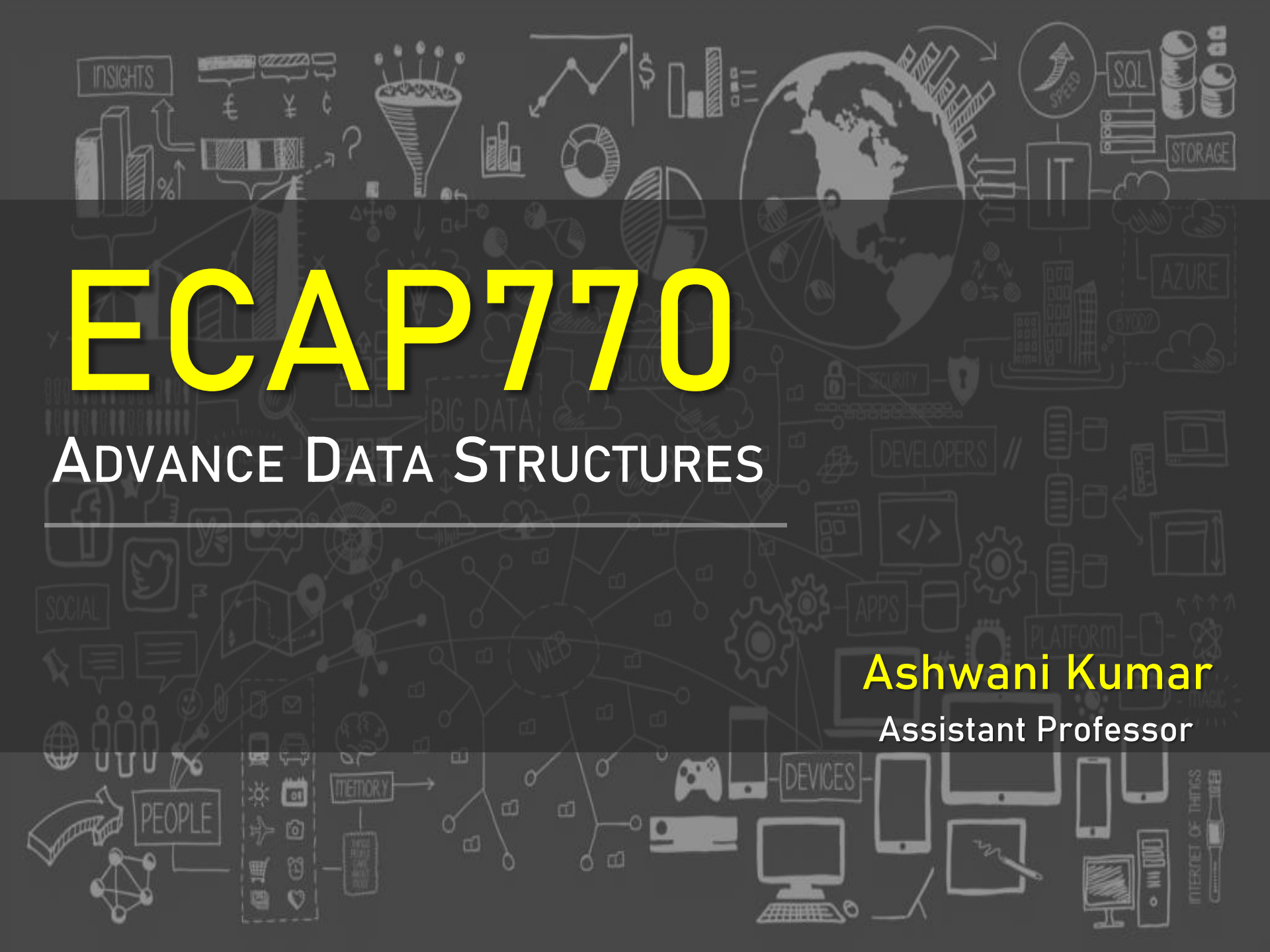


ECAP770

Advance Data Structures

Ashwani Kumar

Assistant Professor

Learning Outcomes

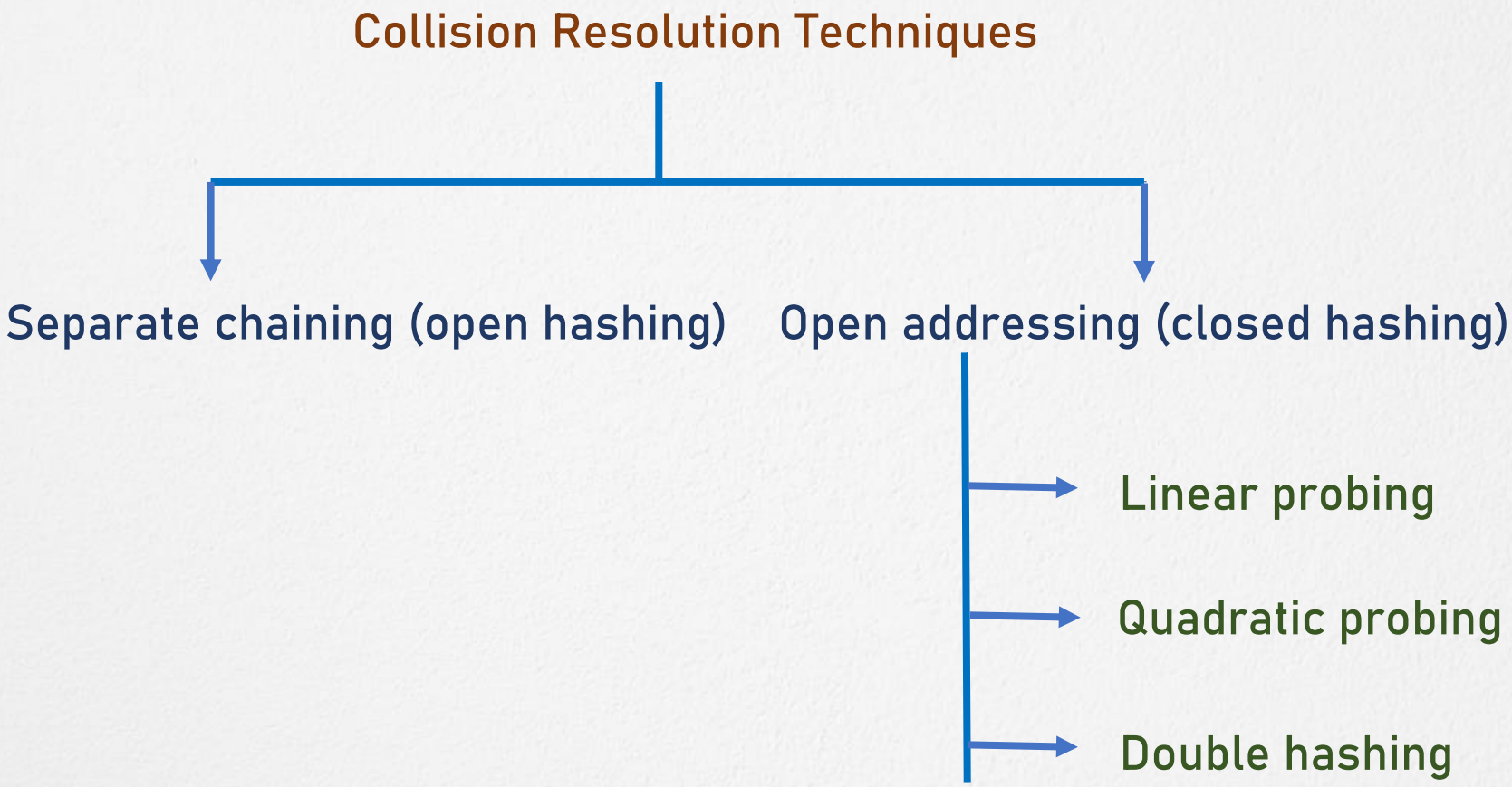
After this lecture, you will be able to

- Understand Collision Resolution Techniques
  - Open addressing (closed hashing)
    - Linear probing
    - Quadratic probing

Collision Resolution

- When two keys or hash values compete with a single hash table slot, then Collision occur.
- To resolve collision we use collision resolution techniques.
- Collisions can be reduced with a selection of a good hash function.

Collision Resolution Techniques



Open Addressing

- The open addressing technique requires a hash table with fixed and known size.
- All elements are stored in the hash table itself.

Open Addressing

- The size of the table must be greater than or equal to the total number of keys.
- During insertion, if a collision is encountered, alternative cells are tried until an empty bucket is found.

Open Addressing

- In case of collision:
  - Probing is performed until an empty bucket is found.
  - Once an empty bucket is found, the key is inserted.
  - Probing is performed in accordance with the technique used for open addressing.

Operations in Open Addressing

- Insert(k): Keep probing until an empty slot is found. Once an empty slot is found, insert k.
- Search(k): Keep probing until slot's key doesn't become equal to k or an empty slot is reached.
- Delete: The key is first searched and then deleted. After deleting the key, that particular bucket is marked as "deleted".

Open Addressing

- Closed Hashing methods:
  - Linear Probing
  - Quadratic probing
  - Double hashing

Linear Probing

- In linear probing fixed sized hash table is used and when hash collision situation occur then, we linearly traverse the table in a cyclic manner to find the next empty slot.
- In this approach searches are performed sequentially so it's known as linear probing.

Linear Probing

- Let hash(x) be the slot index computed using a hash function and n be the table size
- If slot hash(x) % n is full, then we try (hash(x) + 1) % n
- If (hash(x) + 1) % n is also full, then we try (hash(x) + 2) % n
- If (hash(x) + 2) % n is also full, then we try (hash(x) + 3) % n so on…

Example: Linear Probing

$h(k) = k \bmod 10$

$h(k, i) = (h(k)+i) \bmod 10$

25%10 = 5

10%10 = 0

15%10 = 5

18%10 = 8

13%10 = 3

23%10 = 3

Hash table

Example: Linear Probing

$h(k) = k \bmod 10$

$h(k, i) = (h(k)+i) \bmod 10$

25%10 = 5

10%10 = 0

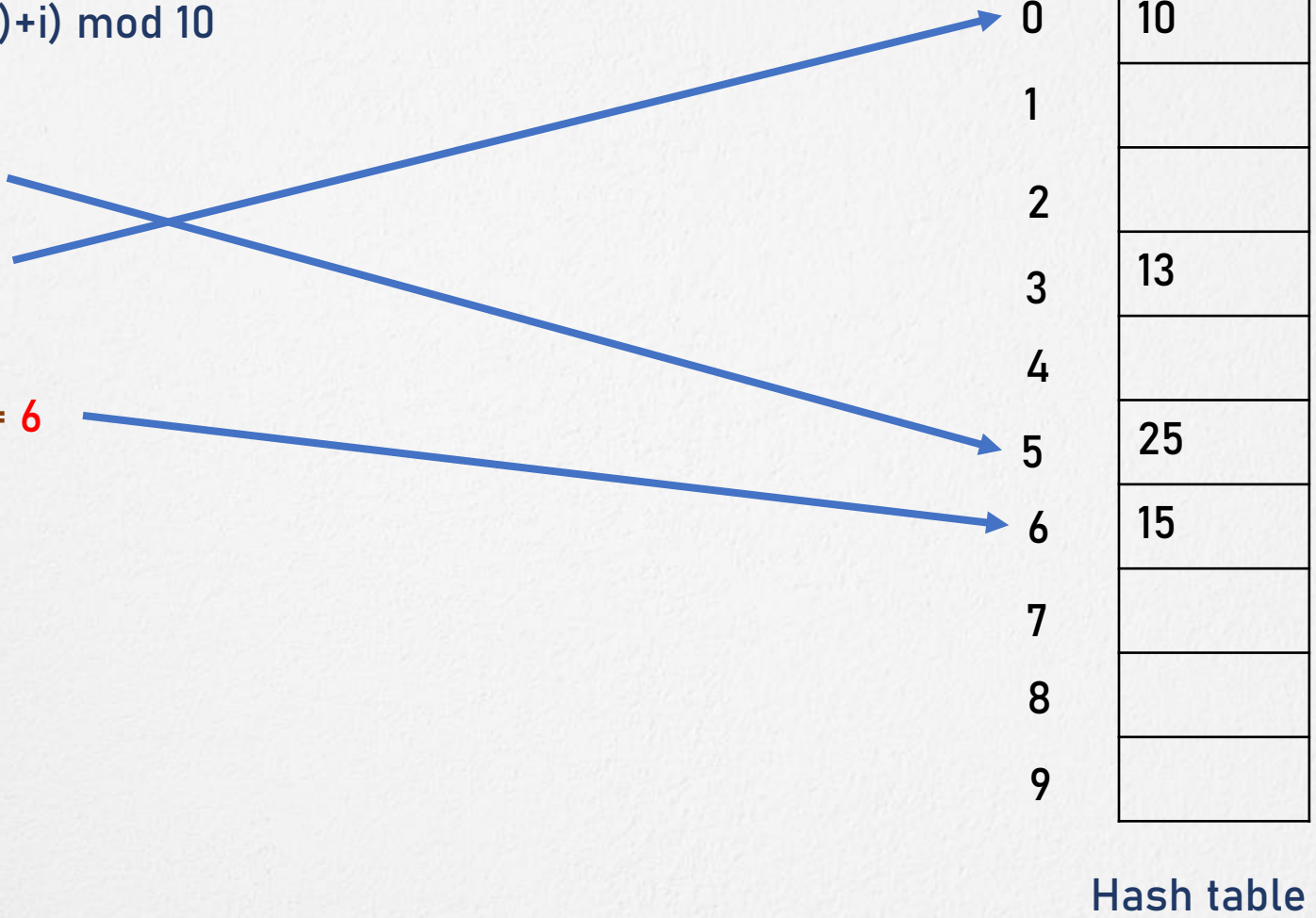
15%10 = 5

5+1 mod 10= 6

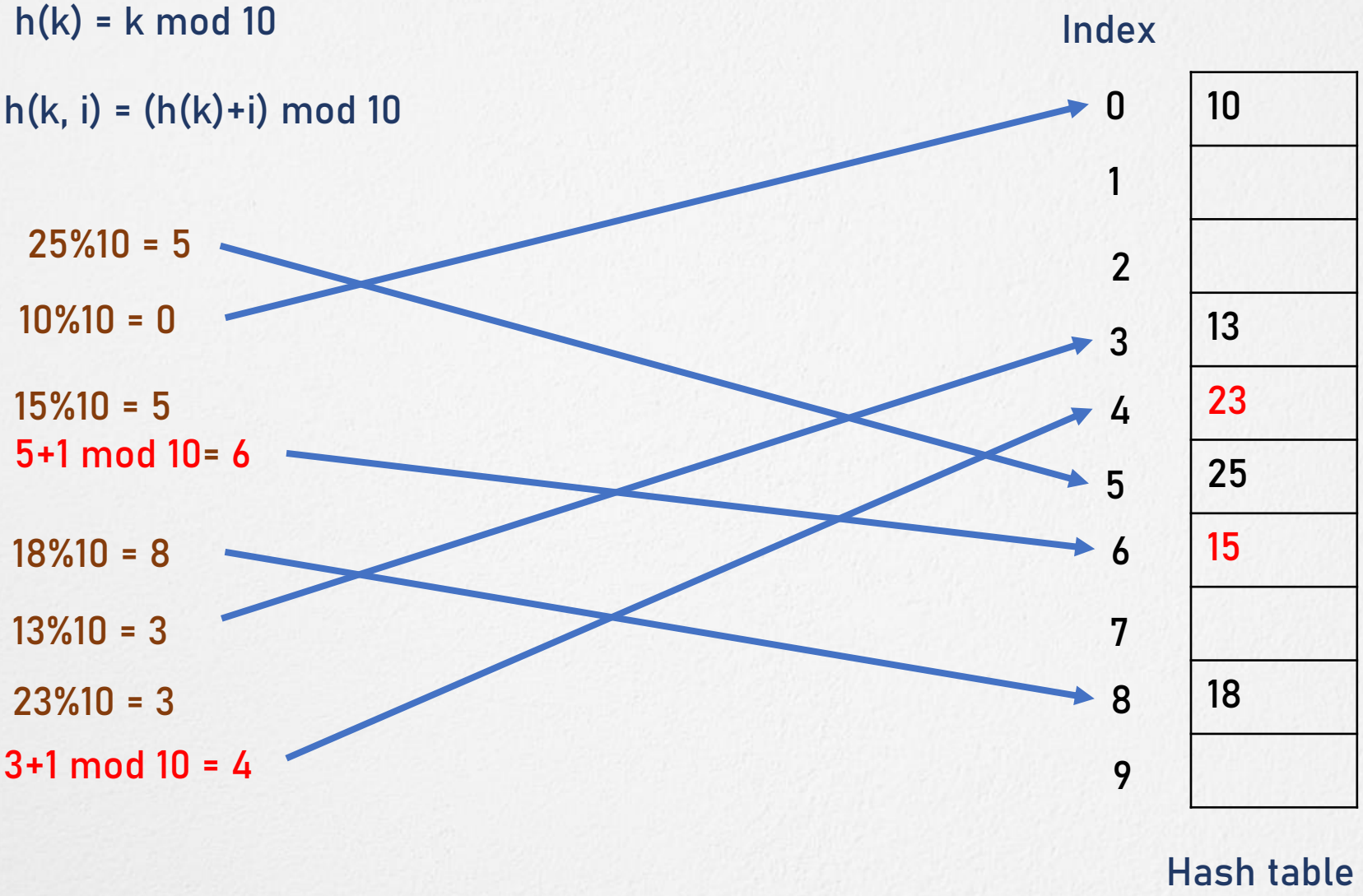
18%10 = 8

13%10 = 3

23%10 = 3



Example: Linear Probing



Linear probing

Advantage

- It is easy to compute.

Disadvantage

- The clustering is major problem with linear probing
- Many consecutive elements form groups.
- It takes too much time to find an empty slot.

Time complexity

- Worst time to search for an element is O (table size).

Quadratic Probing

- It is an open-addressing scheme.
- Here we look for i^2 slot in i^{th} iteration if the given hash value x collides in the hash table.
- It is used to eliminate the primary clustering problem of linear probing.

Quadratic Probing

- In quadratic probing the sequence is that $H+1^2$, $H+2^2$, $H+3^2$,....$H+K^2$
- The hash function for quadratic probing is

$h_i(X)$ = (Hash(X) + $F(i)^2$) % Table Size for i = 0, 1, 2, 3,...etc.

Quadratic Probing

- If the slot hash(x) % S is full, then we try

  (hash(x) + 1*1) % S.

- If (hash(x) + 1*1) % S is also full, then we try (hash(x) + 2*2) % S.

- If (hash(x) + 2*2) % S is also full, then we try (hash(x) + 3*3) % S.

- This process continue until an empty slot is found.

Example: Quadratic Probing

$h(k) = k \bmod 10$

$h(k, i) = (h(k)+i^2) \bmod 10$

Values: 25,12,15, 22,14, 35

25%10 = 5

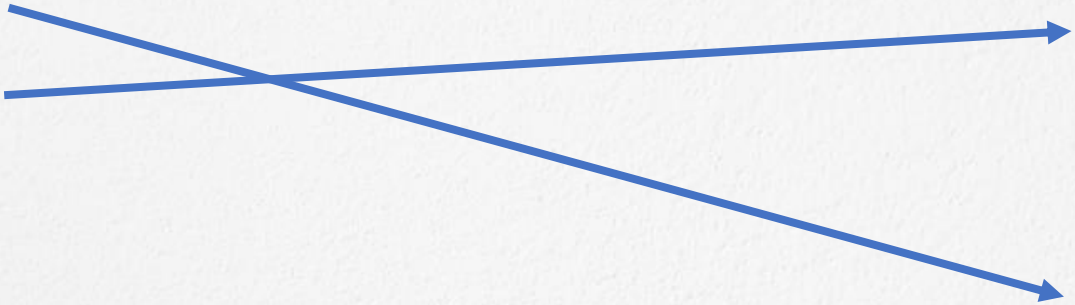
12%10 = 2

15%10 = 5

22%10 = 2

14%10 = 4

35%10 = 5



Index	
0	10
1	
2	12
3	
4	
5	25
6	
7	
8	
9	

Hash table

Example: Quadratic Probing

$h(k) = k \bmod 10$

$h(k, i) = (h(k)+i^2) \bmod 10$

25%10 = 5

12%10 = 2

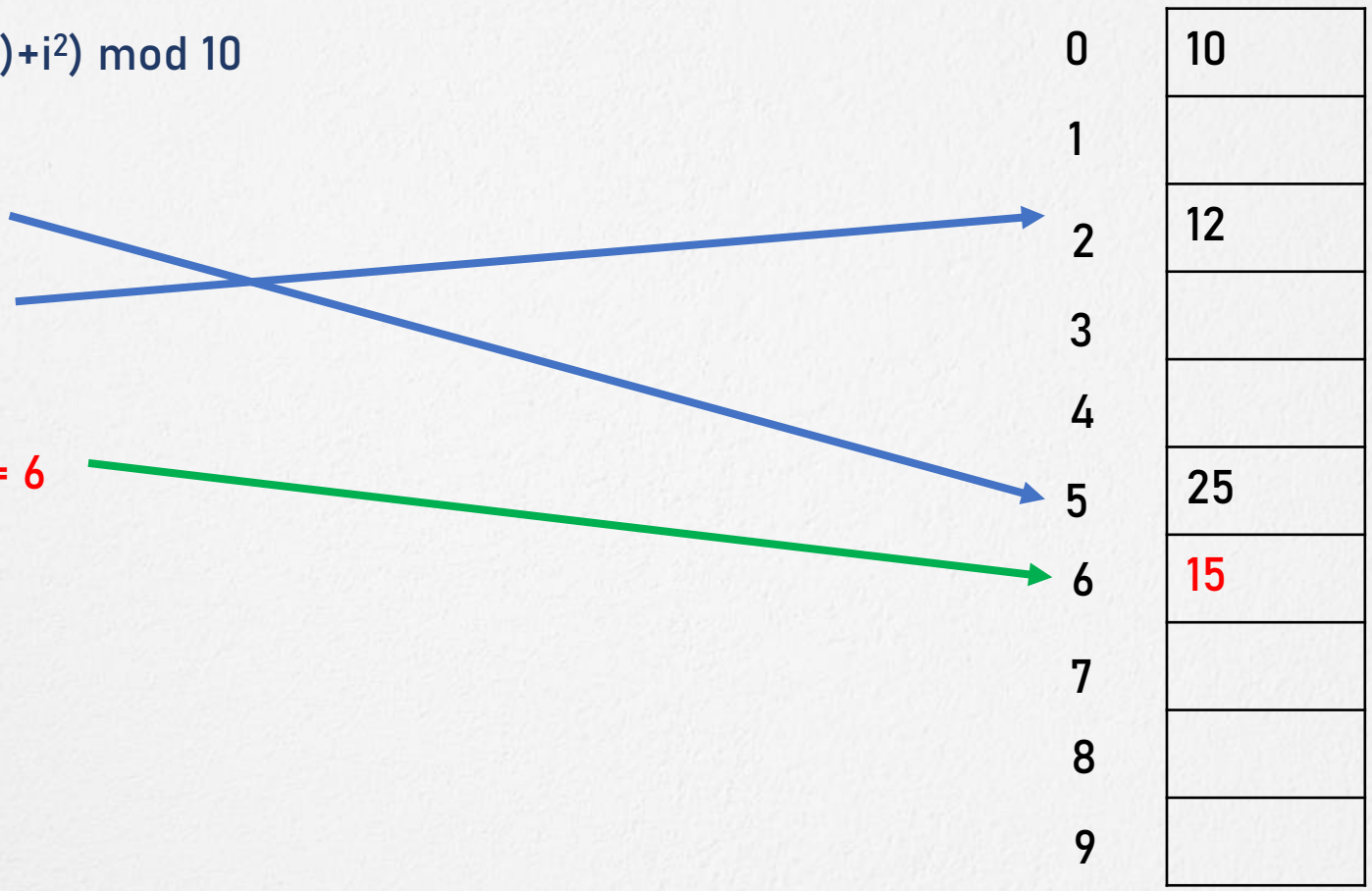
15%10 = 5

$5+1^2 \bmod 10 = 6$

22%10 = 2

14%10 = 4

35%10 = 5



Example: Quadratic Probing

$h(k) = k \bmod 10$

$h(k, i) = (h(k)+i^2) \bmod 10$

25%10 = 5

12%10 = 2

15%10 = 5

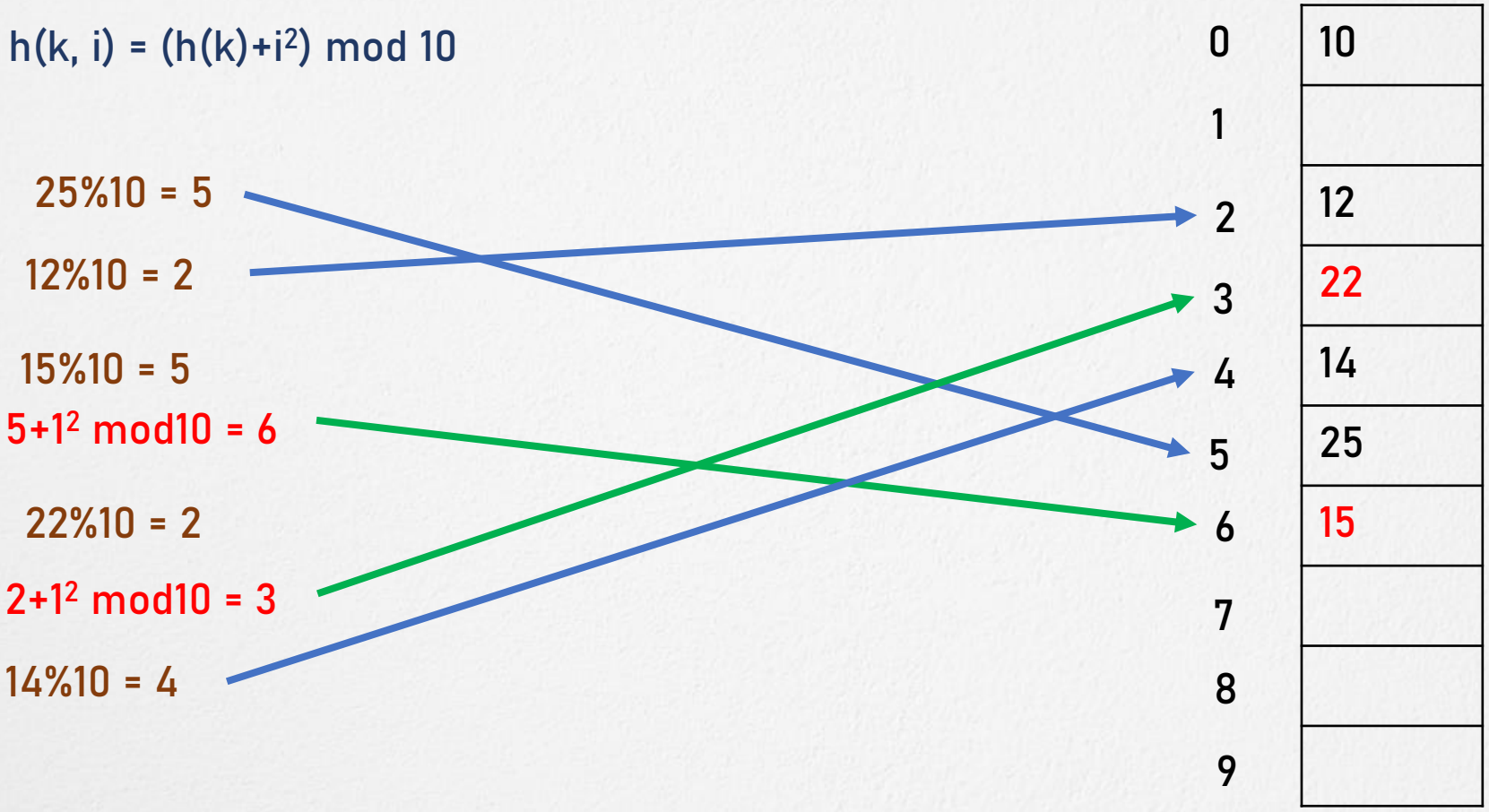
$5+1^2 \bmod 10 = 6$

22%10 = 2

$2+1^2 \bmod 10 = 3$

14%10 = 4

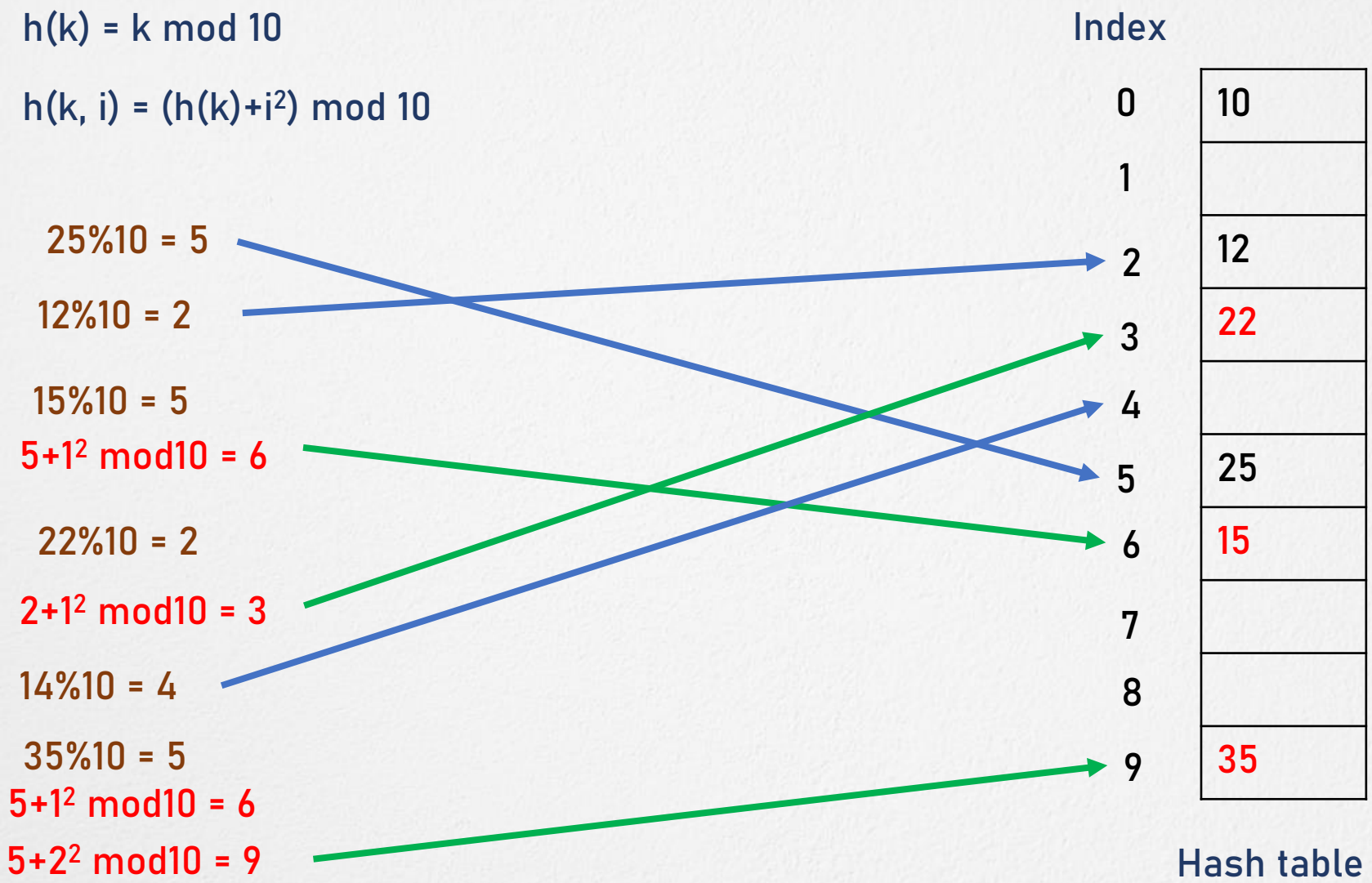
35%10 = 5



Example: Quadratic Probing

$h(k) = k \bmod 10$

$h(k, i) = (h(k)+i^2) \bmod 10$



Quadratic Probing

- In Quadratic Probing to get slot your table size must meet these requirements:

  -Be a prime number

  -never be more than half full (even by one element)

Quadratic Probing

Advantage:

- Primary clustering problem resolved

Disadvantage:

- Secondary clustering
- No guarantee for finding slots

Separate Chaining Vs. Open Addressing

Separate Chaining	Open Addressing
Keys are stored inside the hash table as well as outside the hash table.	All the keys are stored only inside the hash table. No key is present outside the hash table.
The number of keys to be stored in the hash table can even exceed the size of the hash table.	The number of keys to be stored in the hash table can never exceed the size of the hash table.
Deletion is easier.	Deletion is difficult.
Extra space is required for the pointers to store the keys outside the hash table.	No extra space is required.
Cache performance is poor. This is because of linked lists which store the keys outside the hash table.	Cache performance is better. This is because here no linked lists are used.
Some buckets of the hash table are never used which leads to wastage of space.	Buckets may be used even if no key maps to those particular buckets.

That's all for now...