

Data Structures and Algorithms


Lecture 40: Collision Techniques

Collision in Hashing

In hashing technique, Collision is a situation when hash value of two key become similar. For example, let's take the Numbers to hash: 22, 9, 14, 17, 42.

- The first four values can be entered into the hash table without any issues.
- It is the last value, 42, however, that causes a problem. $42 \bmod 10 = 2$, but there is already a value in slot 2 of the hash table, namely 22. This is a **collision**.
- The value 42 must end up in one of the hash table's slots, but arbitrarily assigning it a slot at random would make accessing data in a hash table much more time consuming.

0	
1	
2	22
3	
4	14
5	
6	
7	17
8	
9	9



Collision Resolution Techniques

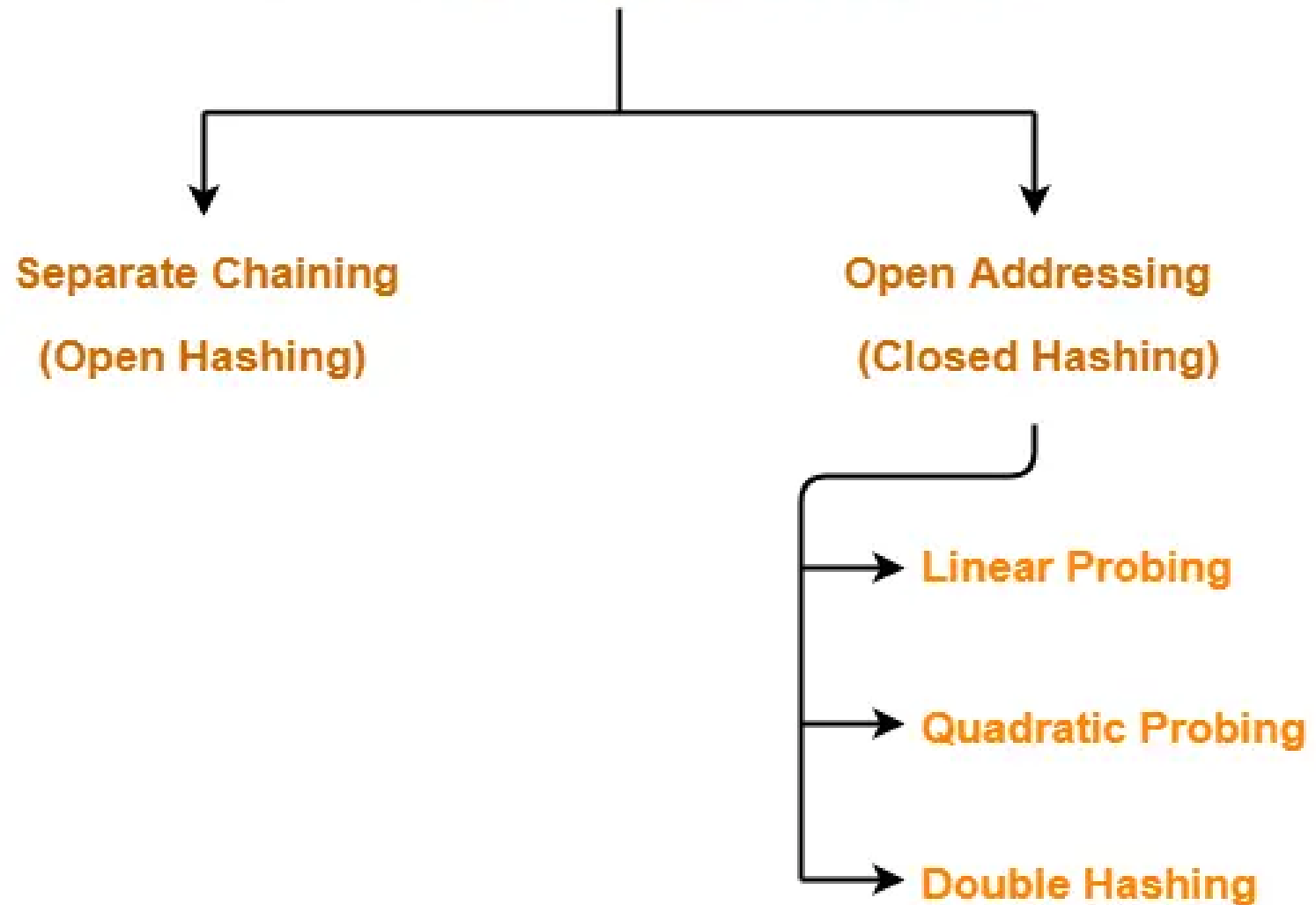
Separate Chaining
(Open Hashing)

Open Addressing
(Closed Hashing)

Linear Probing

Quadratic Probing

Double Hashing



Avoid collision using linear probing

Linear Probing

Calculate the hash key. $\text{key} = \text{data} \% \text{size}$;

If $\text{hashTable}[\text{key}]$ is empty, store the value directly. $\text{hashTable}[\text{key}] = \text{data}$.

If the hash index already has some value, check for next index.

$\text{key} = (\text{key} + 1) \% \text{size}$;

If the next index is available $\text{hashTable}[\text{key}]$, store the value. Otherwise try for next index.

Do the above process till we find the space.

Linear Probing Procedure

Insert table

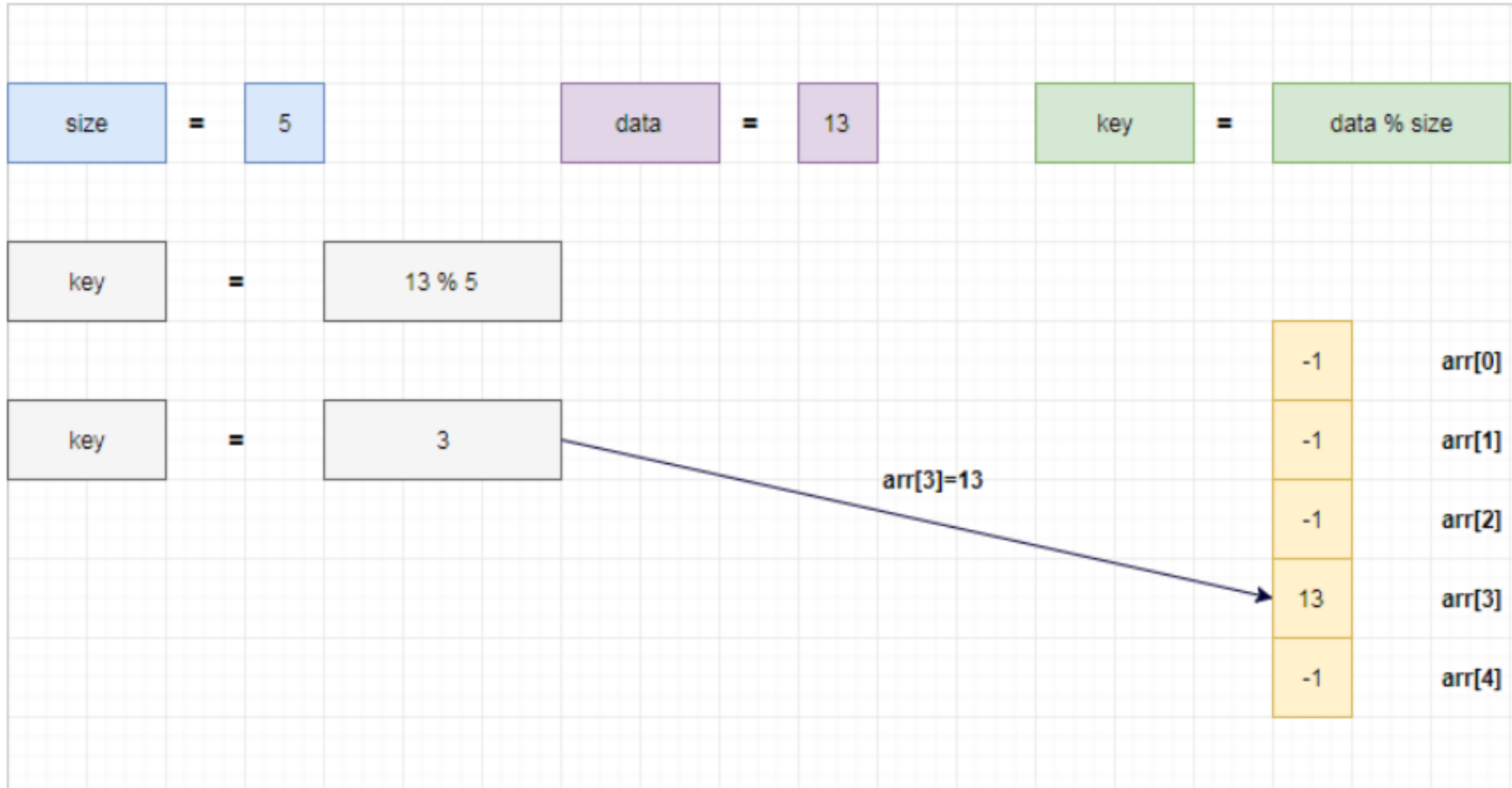
size = 5

-1	arr[0]
-1	arr[1]
-1	arr[2]
-1	arr[3]
-1	arr[4]

-1 indicates that the index is available to insert

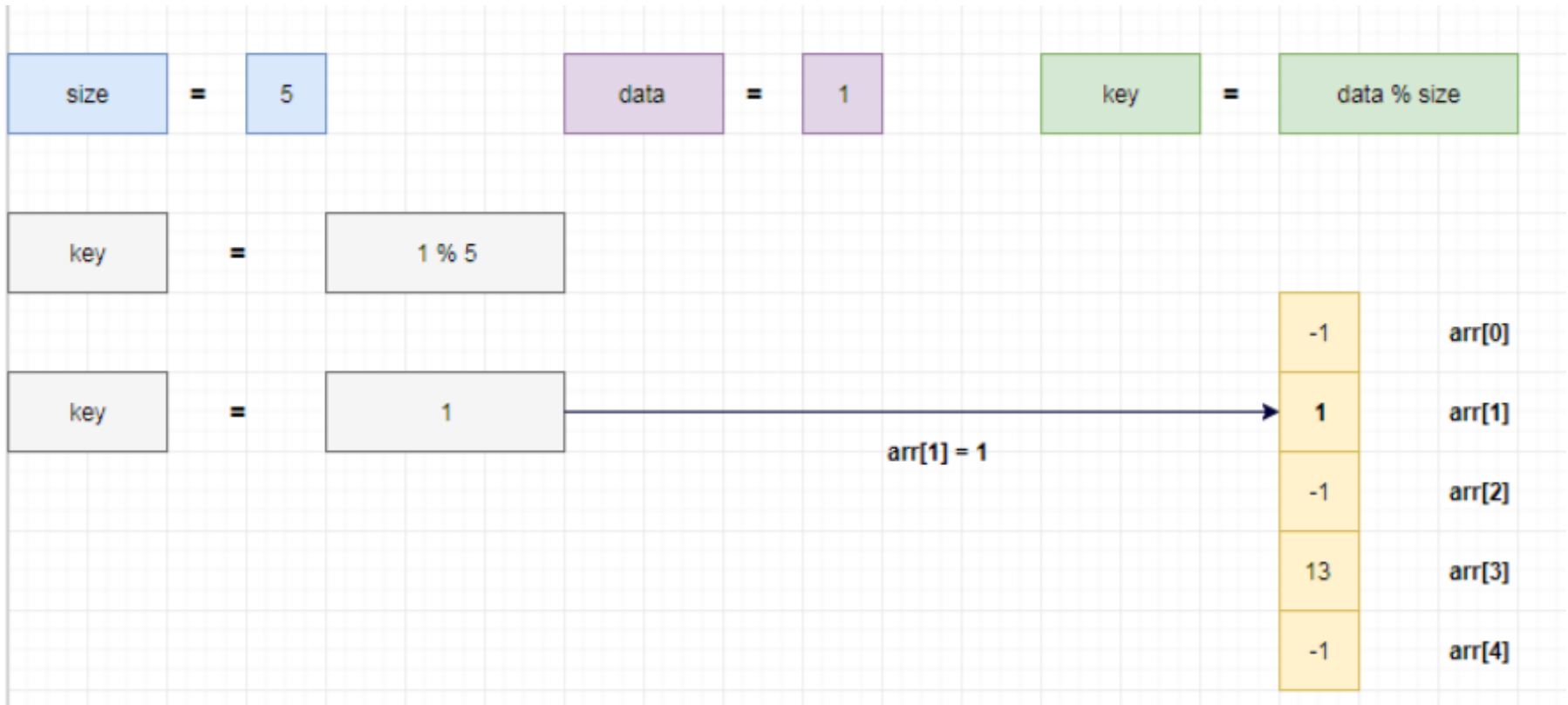
Linear Probing Procedure

Insert 13



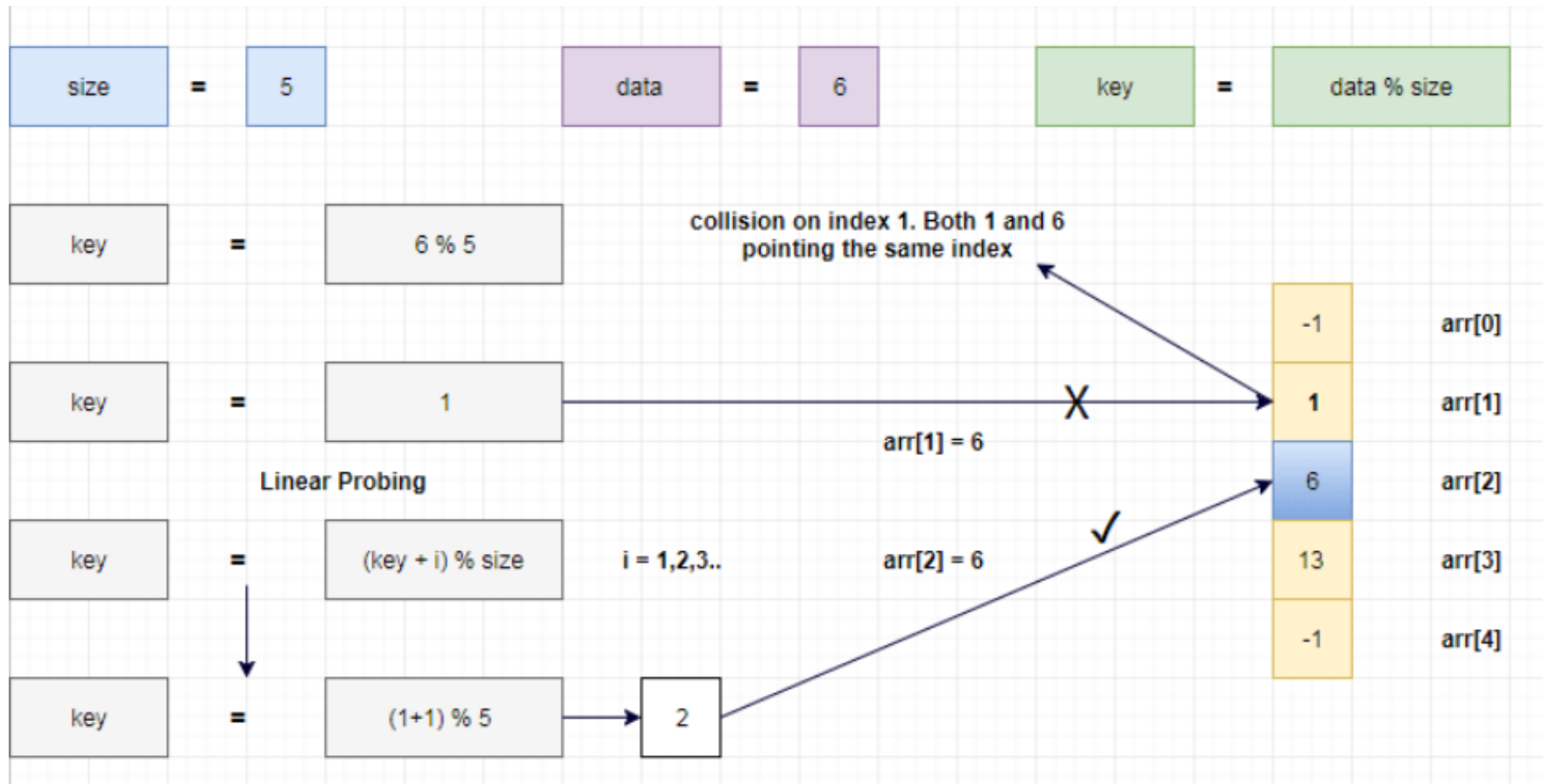
Linear Probing Procedure

Insert 1



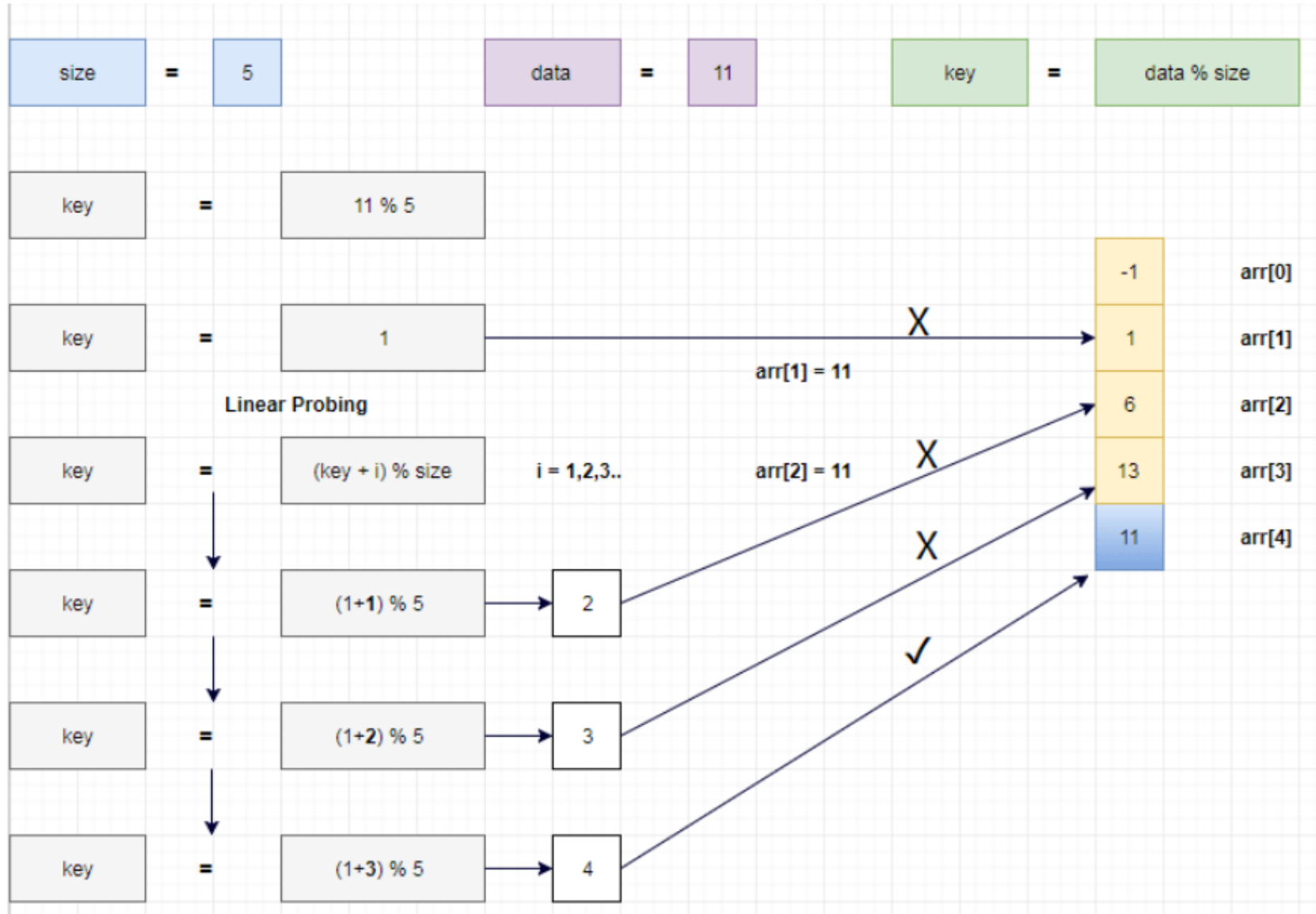
Linear Probing Procedure

Insert 6



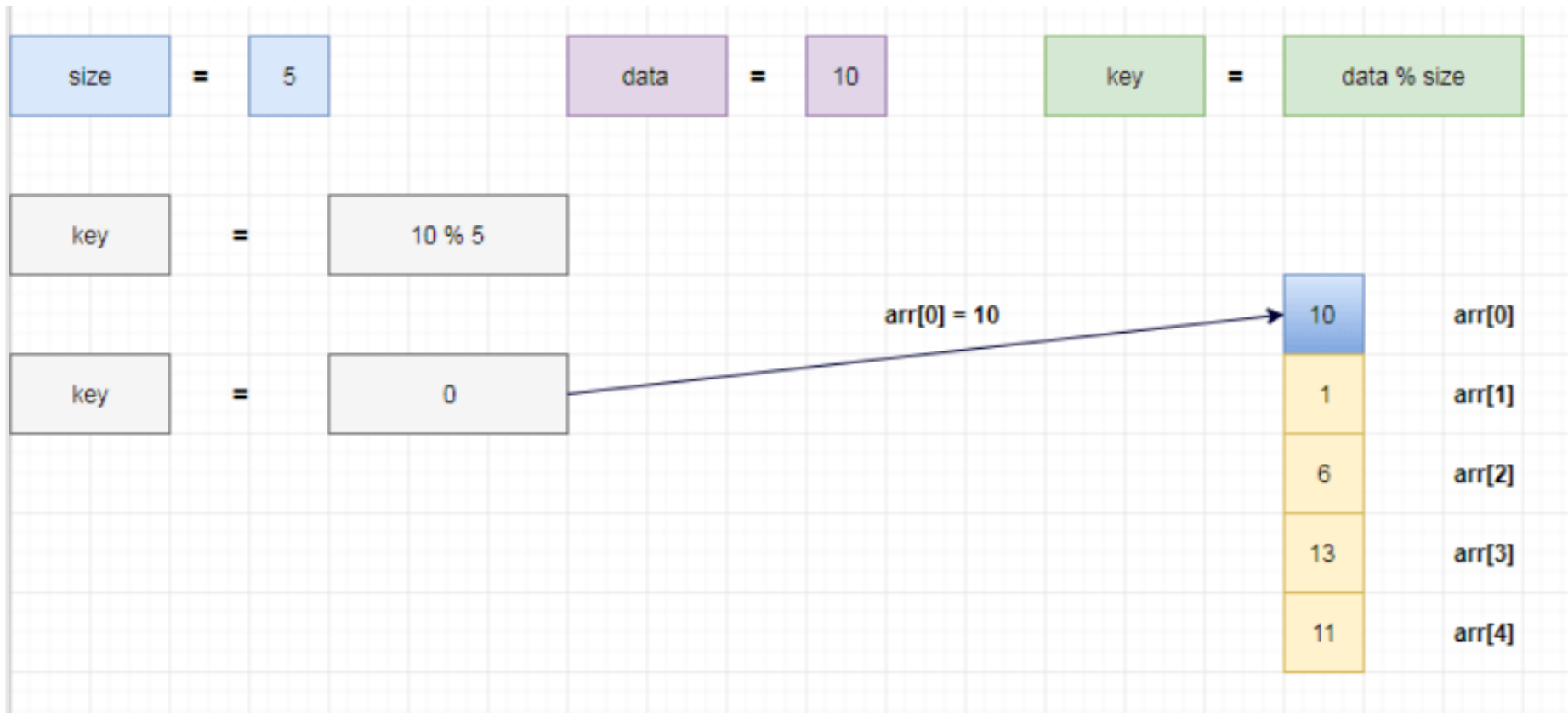
Linear Probing Procedure

Insert 11



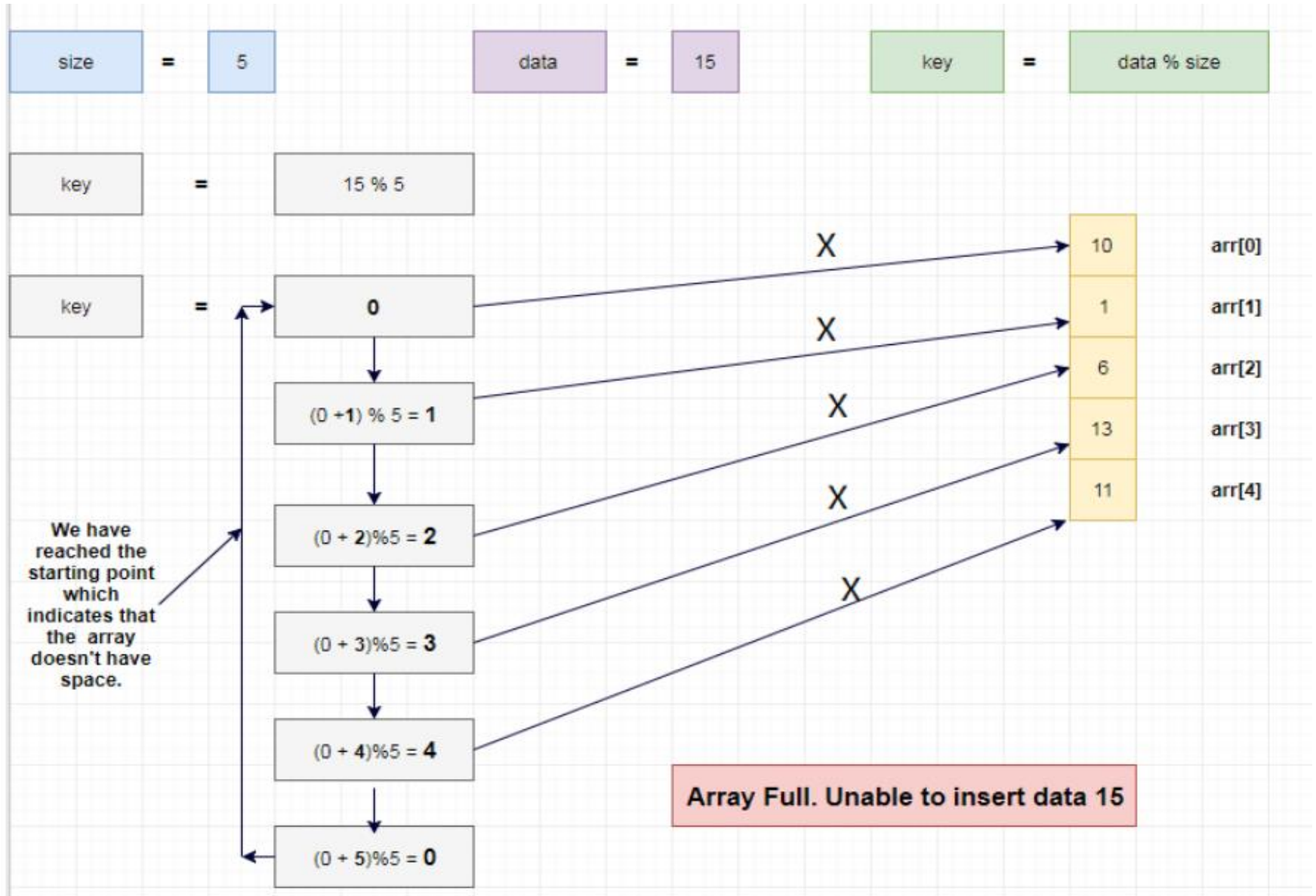
Linear Probing Procedure

Insert 10



Linear Probing Procedure

Insert 15



1. Linear Probing

In linear probing,

- When collision occurs, we linearly probe for the next bucket.
- We keep probing until an empty bucket is found.

Advantage-

- It is easy to compute.

Disadvantage-

- The main problem with linear probing is clustering.
- Many consecutive elements form groups.
- Then, it takes time to search an element or to find an empty bucket.

2. Quadratic Probing

In quadratic probing,

- When collision occurs, we probe for i^2 bucket in i^{th} iteration.
- We keep probing until an empty bucket is found.

2. Quadratic Probing

Suppose a record R with key k has the hash address $H(k) = h$ then instead of searching the location with addresses $h, h+1$, and $h+2 \dots$ We linearly search the locations with addresses

$$h, h+1, h+4, h+9 \dots h+i^2$$

Quadratic Probing uses a hash function of the form

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

Where (as in linear probing) h' is an auxiliary hash function c_1 and $c_2 \neq 0$ are auxiliary constants and $i=0, 1 \dots m-1$. The initial position is $T[h'(k)]$; later position probed is offset by the amount that depend in a quadratic manner on the probe number i .

2. Quadratic Probing

Example: Consider inserting the keys 74, 28, 36, 58, 21, 64 into a hash table of size $m = 11$ using quadratic probing with $c_1 = 1$ and $c_2 = 3$. Further consider that the primary hash function is $h'(k) = k \bmod m$.

Solution: For Quadratic Probing, we have

$$h(k, i) = [k \bmod m + c_1 i + c_2 i^2] \bmod m$$

0	1	2	3	4	5	6	7	8	9	10
/	/	/	/	/	/	/	/	/	/	/

This is the initial state of hash table

2. Quadratic Probing

Example: Consider inserting the keys 74, 28, 36, 58, 21, 64 into a hash table of size $m = 11$ using quadratic probing with $c_1=1$ and $c_2=3$. Further consider that the primary hash function is $h'(k) = k \bmod m$.

Solution: For Quadratic Probing, we have

$$h(k, i) = [k \bmod m + c_1 i + c_2 i^2] \bmod m$$

0 1 2 3 4 5 6 7 8 9 10

/	/	/	/	/	/	/	/	/	/	/
---	---	---	---	---	---	---	---	---	---	---

This is the initial state of hash table

Here $c_1 = 1$ and $c_2 = 3$

$$h(k, i) = [k \bmod m + i + 3i^2] \bmod m$$

2. Quadratic Probing

Insert 74.

$$\begin{aligned}h(74, 0) &= (74 \bmod 11 + 0 + 3 \times 0) \bmod 11 \\ &= (8 + 0 + 0) \bmod 11 = 8\end{aligned}$$

T [8] is free; insert the key 74 at this place.

Insert 28.

$$\begin{aligned}h(28, 0) &= (28 \bmod 11 + 0 + 3 \times 0) \bmod 11 \\ &= (6 + 0 + 0) \bmod 11 = 6.\end{aligned}$$

T [6] is free; insert key 28 at this place.

Insert 36.

$$\begin{aligned}h(36, 0) &= (36 \bmod 11 + 0 + 3 \times 0) \bmod 11 \\ &= (3 + 0 + 0) \bmod 11 = 3\end{aligned}$$

T [3] is free; insert key 36 at this place.

2. Quadratic Probing

Insert 58.

$$\begin{aligned}h(58, 0) &= (58 \bmod 11 + 0 + 3 \times 0) \bmod 11 \\&= (3 + 0 + 0) \bmod 11 = 3\end{aligned}$$

T [3] is not free, so next probe sequence is computed as

$$\begin{aligned}h(58, 1) &= (58 \bmod 11 + 1 + 3 \times 1^2) \bmod 11 \\&= (3 + 1 + 3) \bmod 11 \\&= 7 \bmod 11 = 7\end{aligned}$$

T [7] is free; insert key 58 at this place.

Insert 21.

$$\begin{aligned}h(21, 0) &= (21 \bmod 11 + 0 + 3 \times 0) \\&= (10 + 0) \bmod 11 = 10\end{aligned}$$

T [10] is free; insert key 21 at this place.

Insert 64.

$$\begin{aligned}h(64, 0) &= (64 \bmod 11 + 0 + 3 \times 0) \\&= (9 + 0 + 0) \bmod 11 = 9.\end{aligned}$$

T [9] is free; insert key 64 at this place.

2. Quadratic Probing

Thus, after inserting all keys, the hash table is

0	1	2	3	4	5	6	7	8	9	10
/	/	/	36	/	/	28	58	74	64	21

The advantages of quadratic probing is as follows –

- Quadratic probing is less likely to have the problem of primary clustering and is easier to implement than Double Hashing.

The disadvantages of quadratic probing are as follows –

- Quadratic probing has secondary clustering. This occurs when 2 keys hash to the same location, they have the same probe sequence. So, it may take many attempts before an insertion is being made.
- Also probe sequences do not probe all locations in the table.

3. Double Hashing

- Double hashing is a collision resolution technique used in conjunction with open-addressing in hash tables.
- In this technique, we use a two hash function to calculate empty slot to store value.
- In the case of collision we take the second hash function $h_2(k)$ and look for $i * h_2(k)$ free slot in an i^{th} iteration.
- Double hashing requires more computational time because two hash functions need to be computed.
- To start with, double hashing uses two hash function to calculate an empty location.

3. Double Hashing

In double hashing, we use two hash functions rather than a single function. The hash function in the case of double hashing can be given as:

$$h(k, i) = [h_1(k) + i h_2(k)] \bmod m$$

where m is the size of the hash table,

$h_1(k)$ and $h_2(k)$ are two hash functions

$$h_1(k) = k \bmod m,$$

$$h_2(k) = k \bmod m$$

i is the probe number that varies from 0 to $m-1$

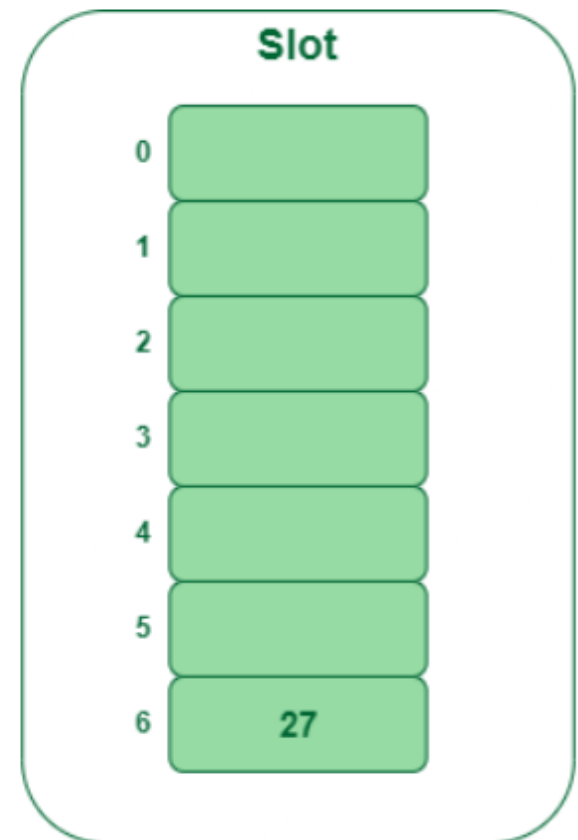
- When we have to insert a key k in the hash table, we first probe the location given by applying $[h_1(k) \bmod m]$ because during the first probe, $i = 0$. If the location is vacant, the key is inserted into it.
- And if the location is not vacant then increase value of i to calculate next location using: $h(k, 1) = [h_1(k) + 1 * h_2(k)] \bmod m$. Otherwise, $h(k, 0) = [h_1(k) + 0 * h_2(k)] \bmod m$ for next key

3. Double Hashing

Example: Insert the keys 27, 43, 692, 72 into the Hash Table of size 7. where first hash-function is $h1(k) = k \bmod 7$ and second hash-function is $h2(k) = 1 + (k \bmod 5)$

Step 1: Insert 27

$27 \% 7 = 6$, location 6 is empty so insert 27 into 6 slot.

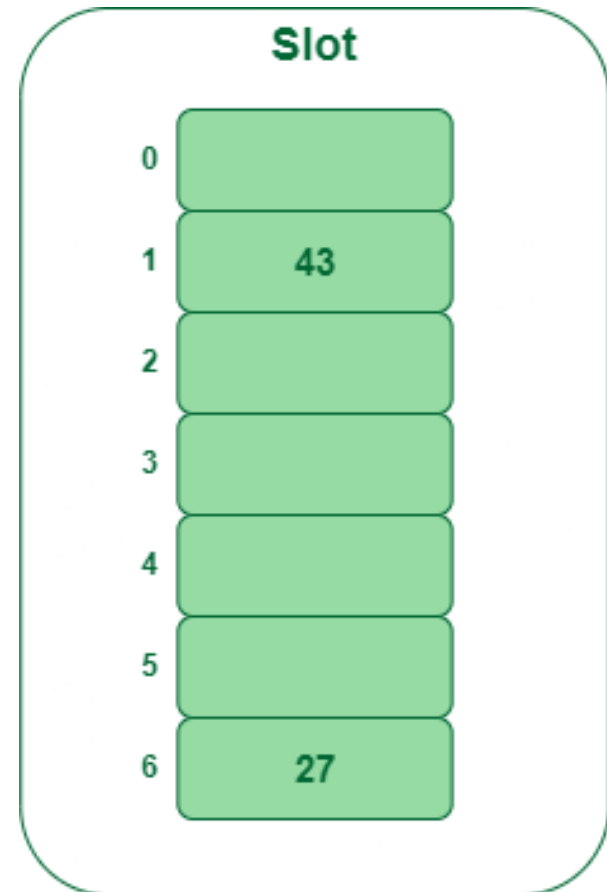


Insert key 27 in the hash table

3. Double Hashing

Step 2: Insert 43

$43 \% 7 = 1$, location 1 is empty so insert 43 into 1 slot.



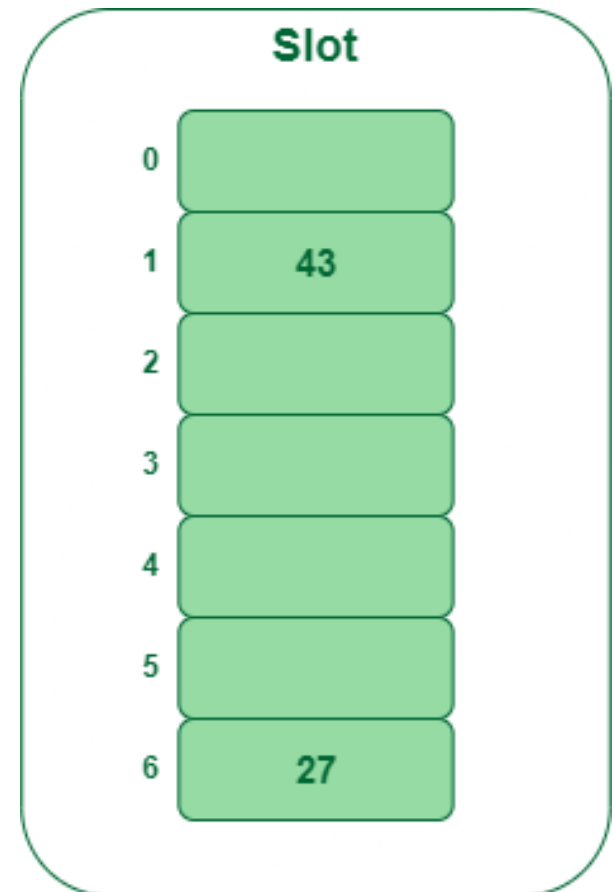
3. Double Hashing

Step 3: Insert 692

- $692 \% 7 = 6$, but location 6 is already being occupied and this is a collision
- So we need to resolve this collision using double hashing.

$$\begin{aligned} h_{new} &= [h1(692) + i * (h2(692))] \% 7 \\ &= [6 + 1 * (1 + 692 \% 5)] \% 7 \\ &= 9 \% 7 \\ &= 2 \end{aligned}$$

*Now, as 2 is an empty slot,
so we can insert 692 into 2nd slot.*



3. Double Hashing

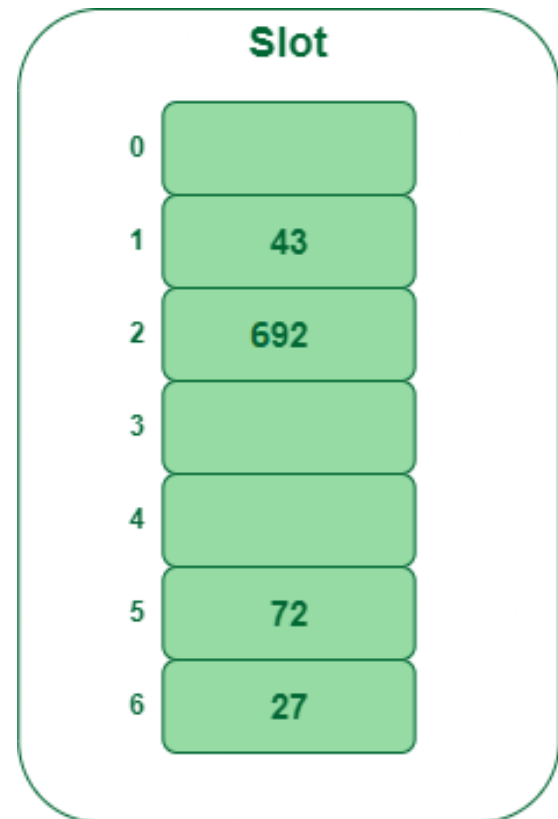
Step 4: Insert 72

$72 \% 7 = 2$, but location 2 is already being occupied and this is a collision.

So we need to resolve this collision using double hashing.

$$\begin{aligned} h_{new} &= [h1(72) + i * (h2(72))] \% 7 \\ &= [2 + 1 * (1 + 72 \% 5)] \% 7 \\ &= 5 \% 7 \\ &= 5, \end{aligned}$$

*Now, as 5 is an empty slot,
so we can insert 72 into 5th slot.*



3. Double Hashing

The advantage of double hashing is as follows –

- Double hashing finally overcomes the problems of the clustering issue.

The disadvantages of double hashing are as follows:

- Double hashing is more difficult to implement than any other.
- Double hashing can cause thrashing.

Comparison of Open Addressing Techniques-

	Linear Probing	Quadratic Probing	Double Hashing
Primary Clustering	Yes	No	No
Secondary Clustering	Yes	Yes	No
Number of Probe Sequence (m = size of table)	m	m	m^2
Cache performance	Best	Lies between the two	Poor

Open Hashing – Separate Chaining

Open hashing is a collision avoidance method which uses array of linked list to resolve the collision.

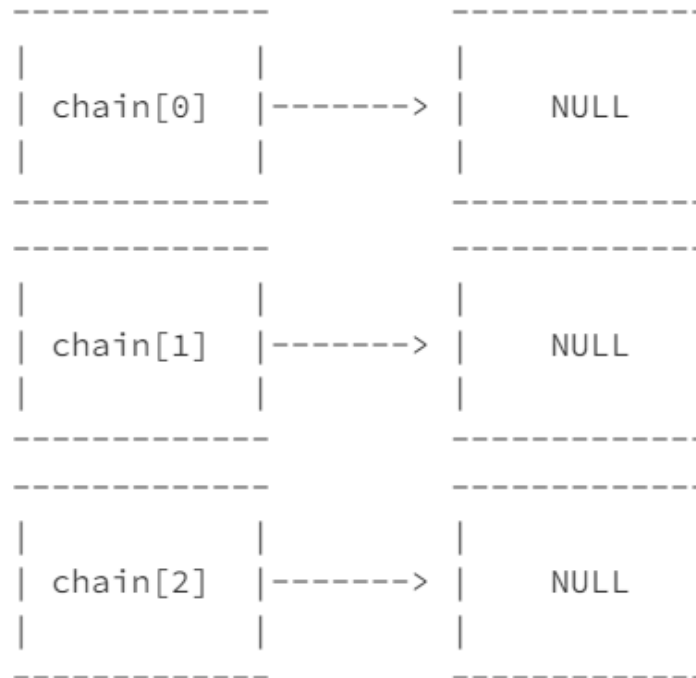
It is also known as the separate chaining method (each linked list is considered as a chain).

Open Hashing – Separate Chaining

Example:

Let's assume table size as 3.

Then the array of linked list will be,

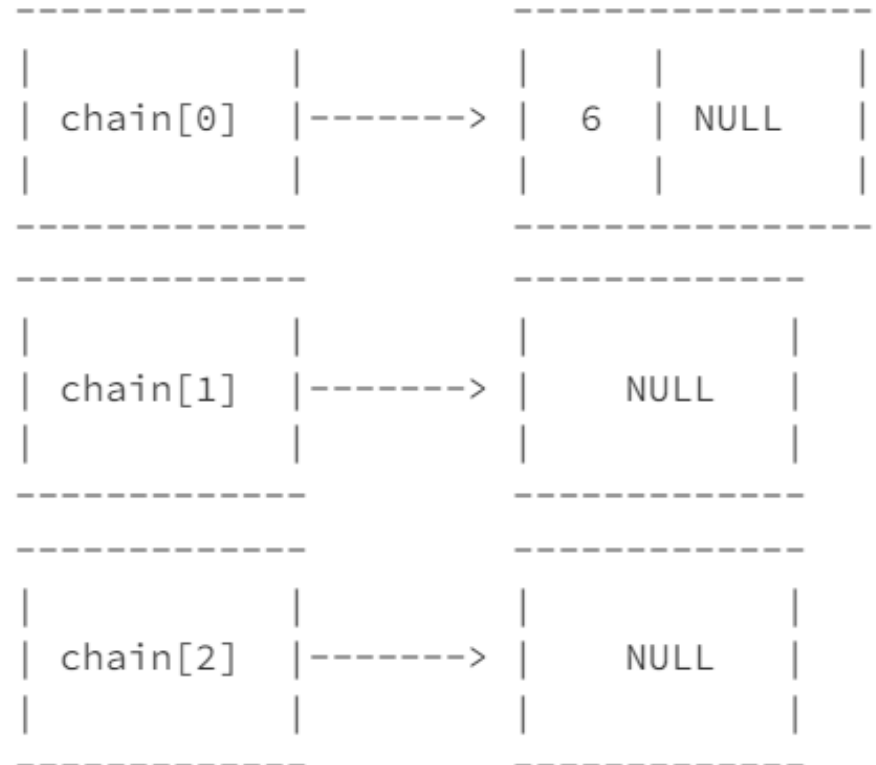


Initialize each list to NULL.

Open Hashing – Separate Chaining

i) Insert 6

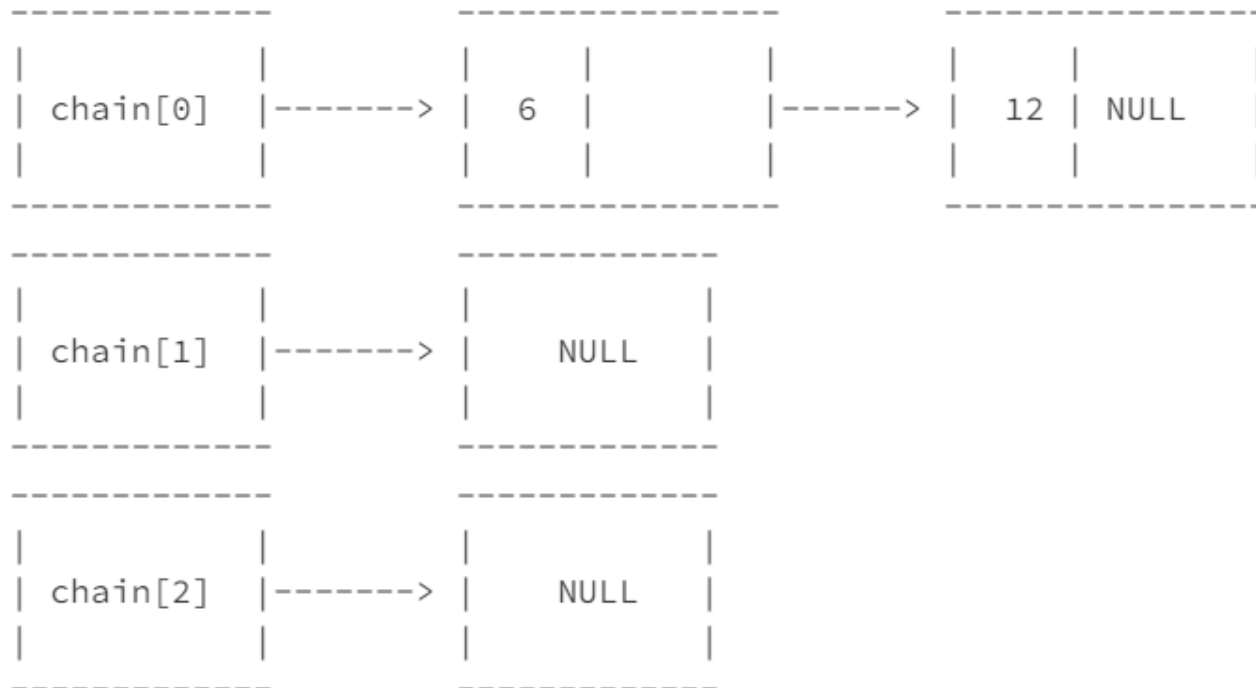
- Hash key = $6 \% 3 = 0$.
- Hence add the node with data 6 in the chain[0].



Open Hashing – Separate Chaining

ii) Insert 12

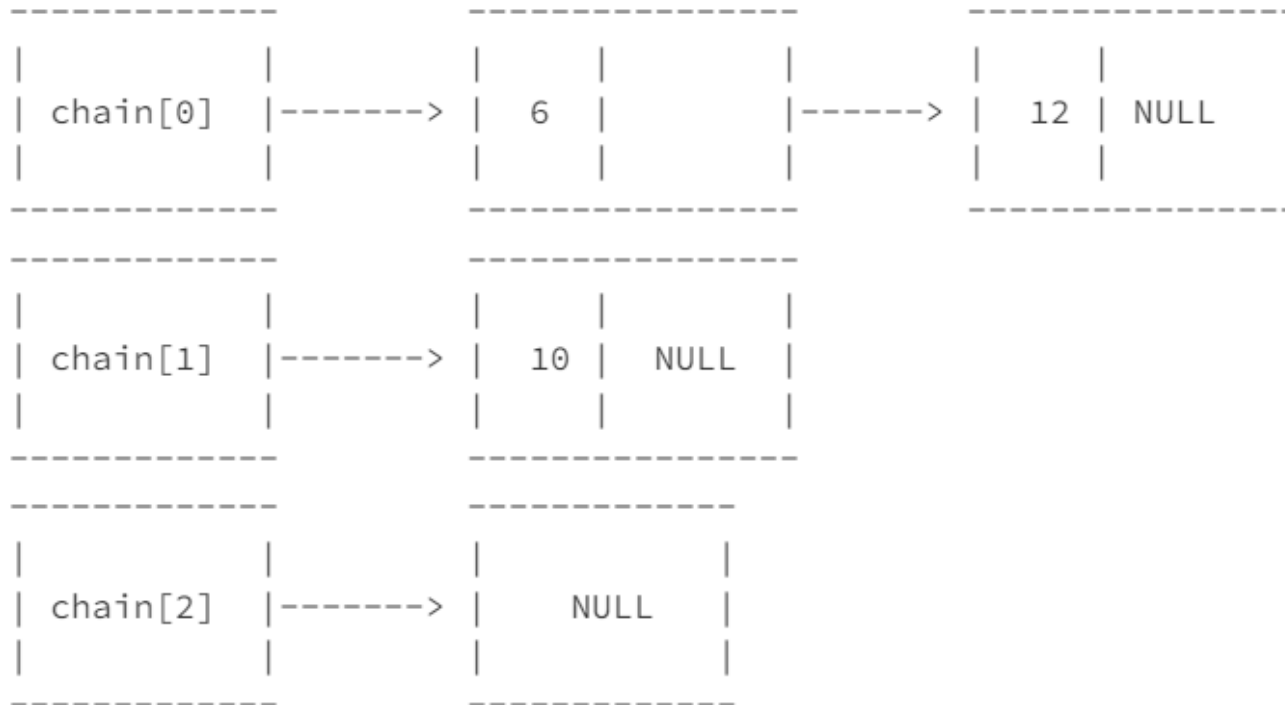
- Hash key = $12 \% 3 = 0$
- Collision! Both 6 and 12 points to the hash index 0.
- We can avoid the collision by adding data 12 at the end of the chain[0].
- This how we can avoid the collision in separate chaining method.



Open Hashing – Separate Chaining

iii) Insert 10

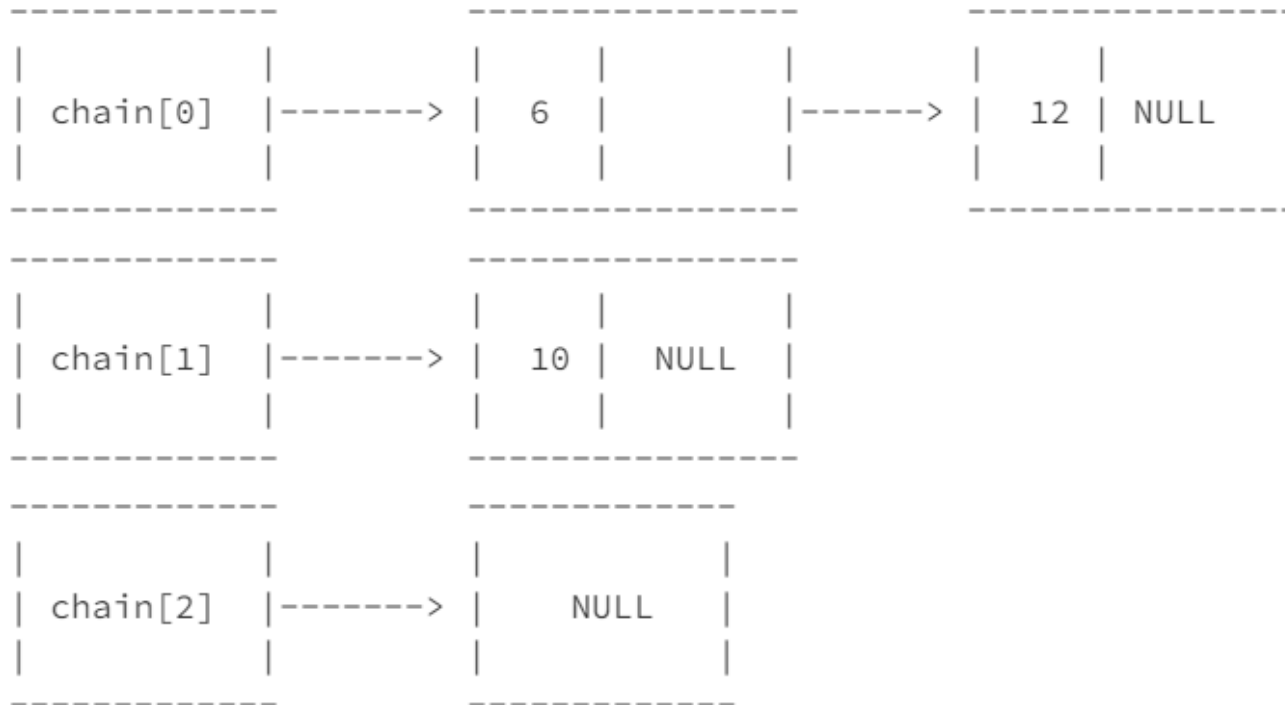
- Hash key = $10 \% 3 = 1$.
- Hence add node with data 10 in the chain[1].



Open Hashing – Separate Chaining

iii) Insert 10

- Hash key = $10 \% 3 = 1$.
- Hence add node with data 10 in the chain[1].



Open Hashing – Separate Chaining

The advantages of separate chaining hashing are as follows –

- Separate chaining technique is not sensitive to the size of the table.
- The idea and the implementation are simple.

The disadvantages of separate chaining hashing are as follows –

- Keys are not evenly distributed in separate chaining.
- Separate chaining can lead to empty spaces in the table.
- The list in the positions can be very long.

S.No.	Separate Chaining	Open Addressing
1.	Chaining is Simpler to implement.	Open Addressing requires more computation.
2.	In chaining, Hash table never fills up, we can always add more elements to chain.	In open addressing, table may become full.
3.	Chaining is Less sensitive to the hash function or load factors.	Open addressing requires extra care to avoid clustering and load factor.
4.	Chaining is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.	Open addressing is used when the frequency and number of keys is known.
5.	Cache performance of chaining is not good as keys are stored using linked list.	Open addressing provides better cache performance as everything is stored in the same table.
6.	Wastage of Space (Some Parts of hash table in chaining are never used).	In Open addressing, a slot can be used even if an input doesn't map to it.
7.	Chaining uses extra space for links.	No links in Open addressing