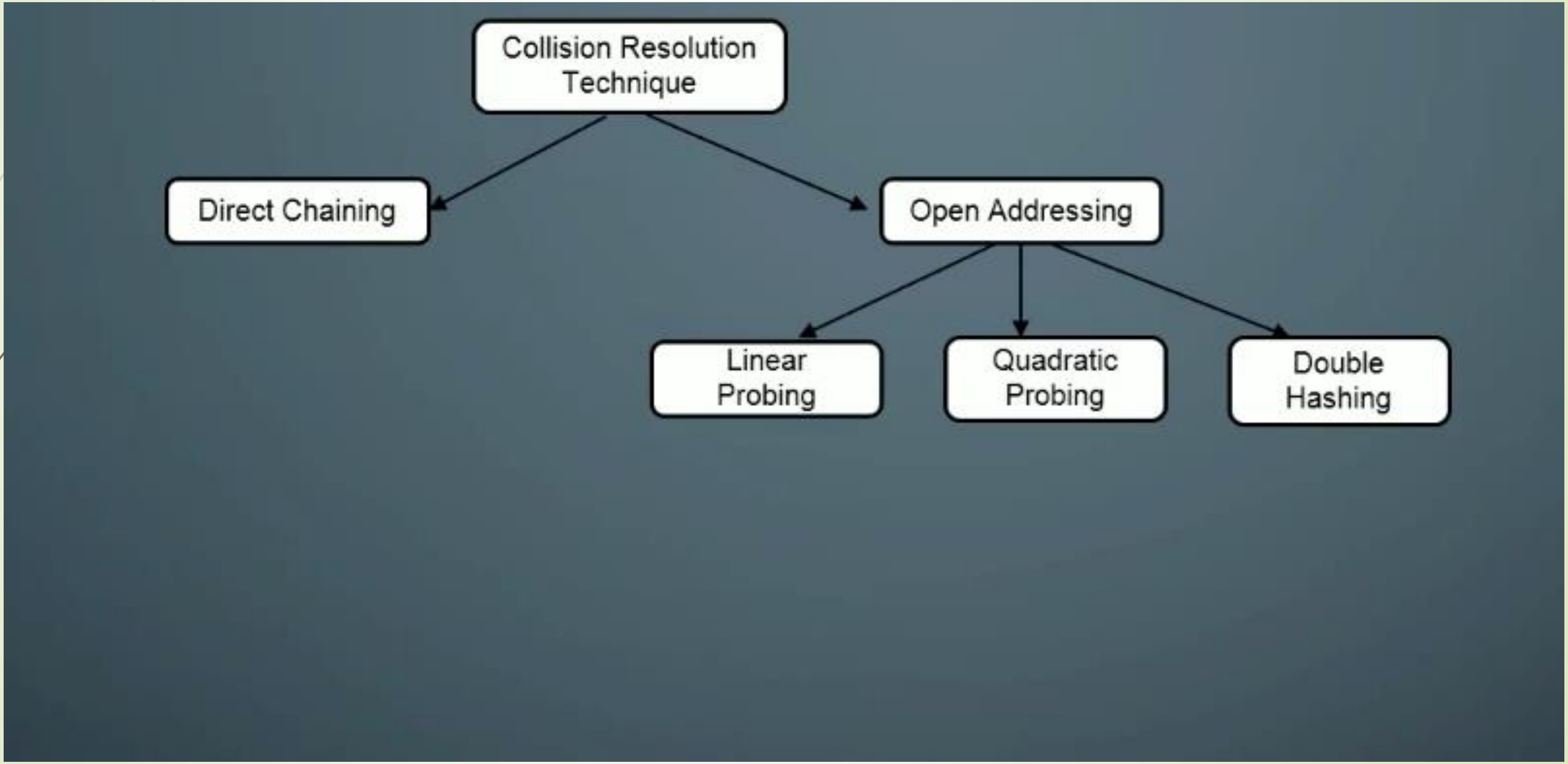
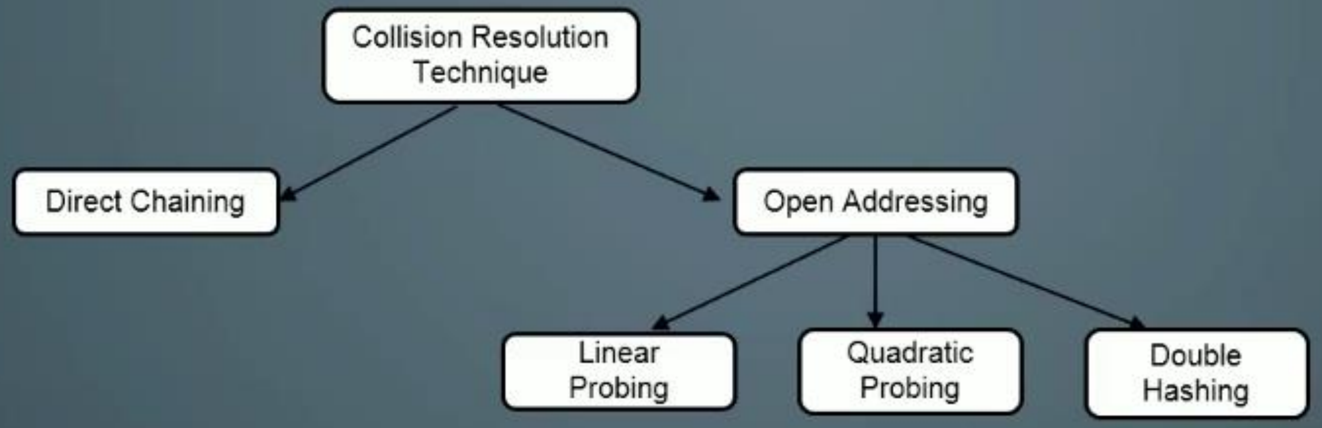
Data Structure and Algorithms

Session-31

Dr. Subhra Rani Patra
SCOPE, VIT Chennai


Collision Resolution Technique
Direct Chaining
Open Addressing
Linear Probing
Quadratic Probing
Double Hashing

Collision Resolution Techniques:



Direct Chaining: Implements the buckets as linked lists. Colliding elements are stored in these lists.

Open Addressing: Colliding elements are stored in other vacant buckets. During storage and lookup, these are found through so-called ‘probing’

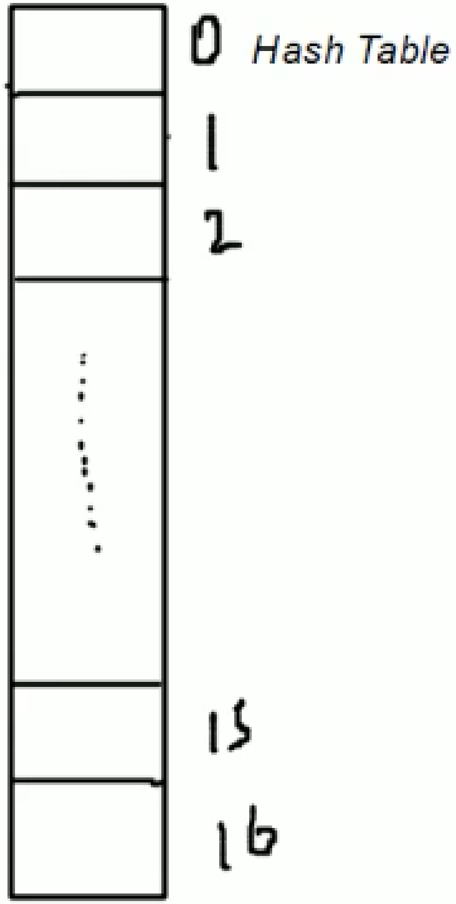
- Linear Probing:
  - Linear probing is a strategy for resolving collisions, by placing the new key into the closest following empty cell

LINEAR PROBING

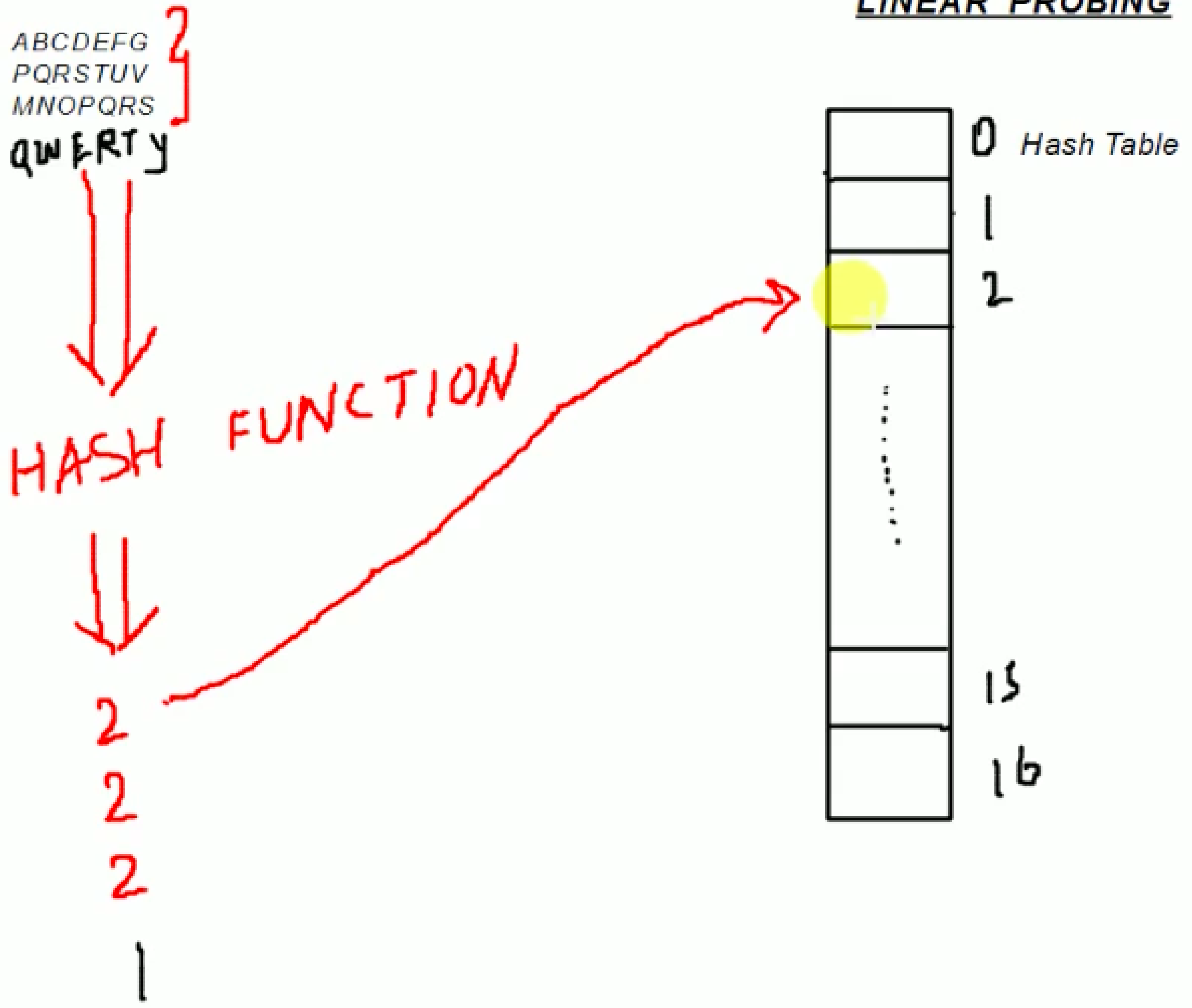
ABCDEFG
PQRSTUV
MNOPQRS
qwERTy

HASH FUNCTION

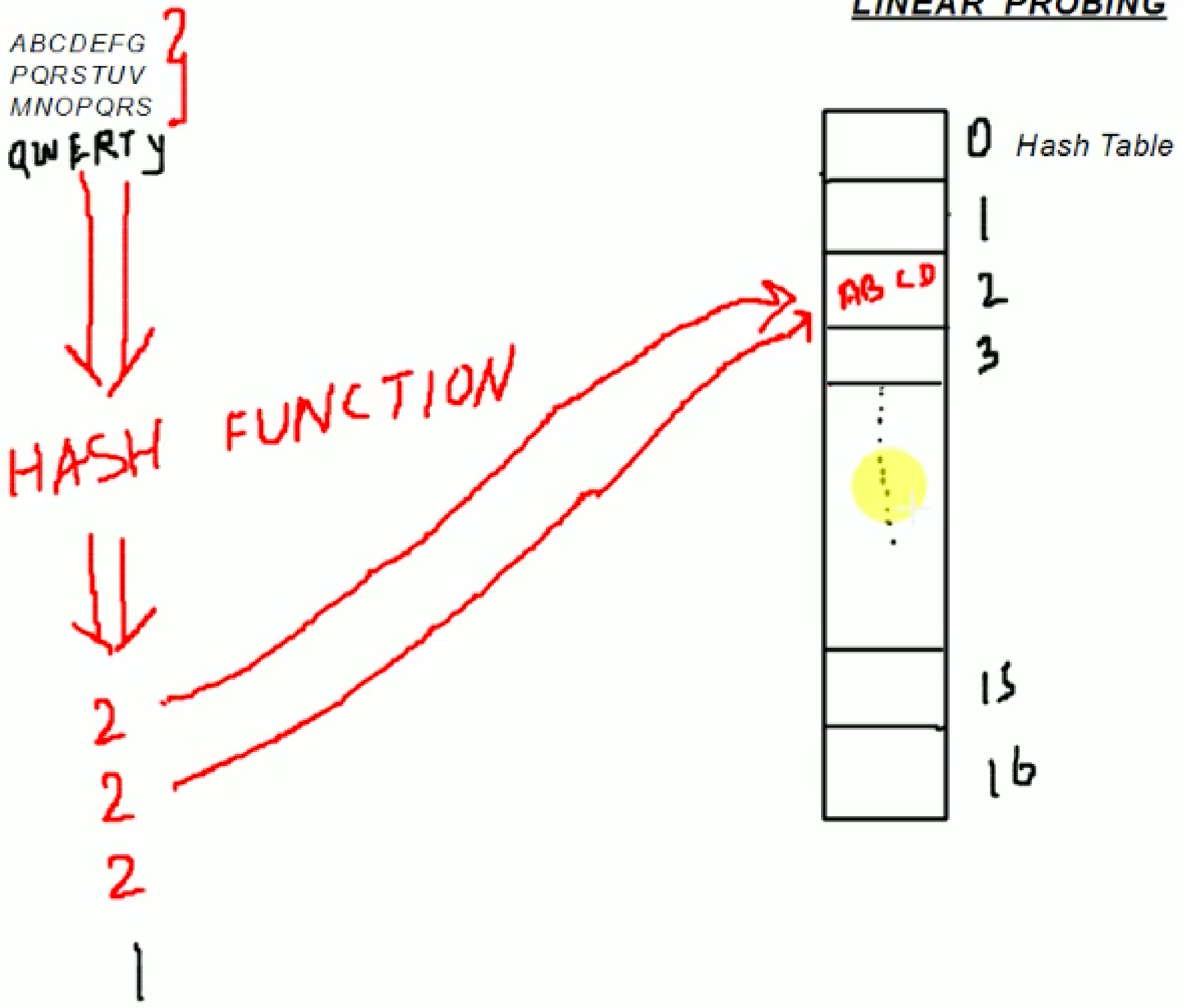
2
2
2
1

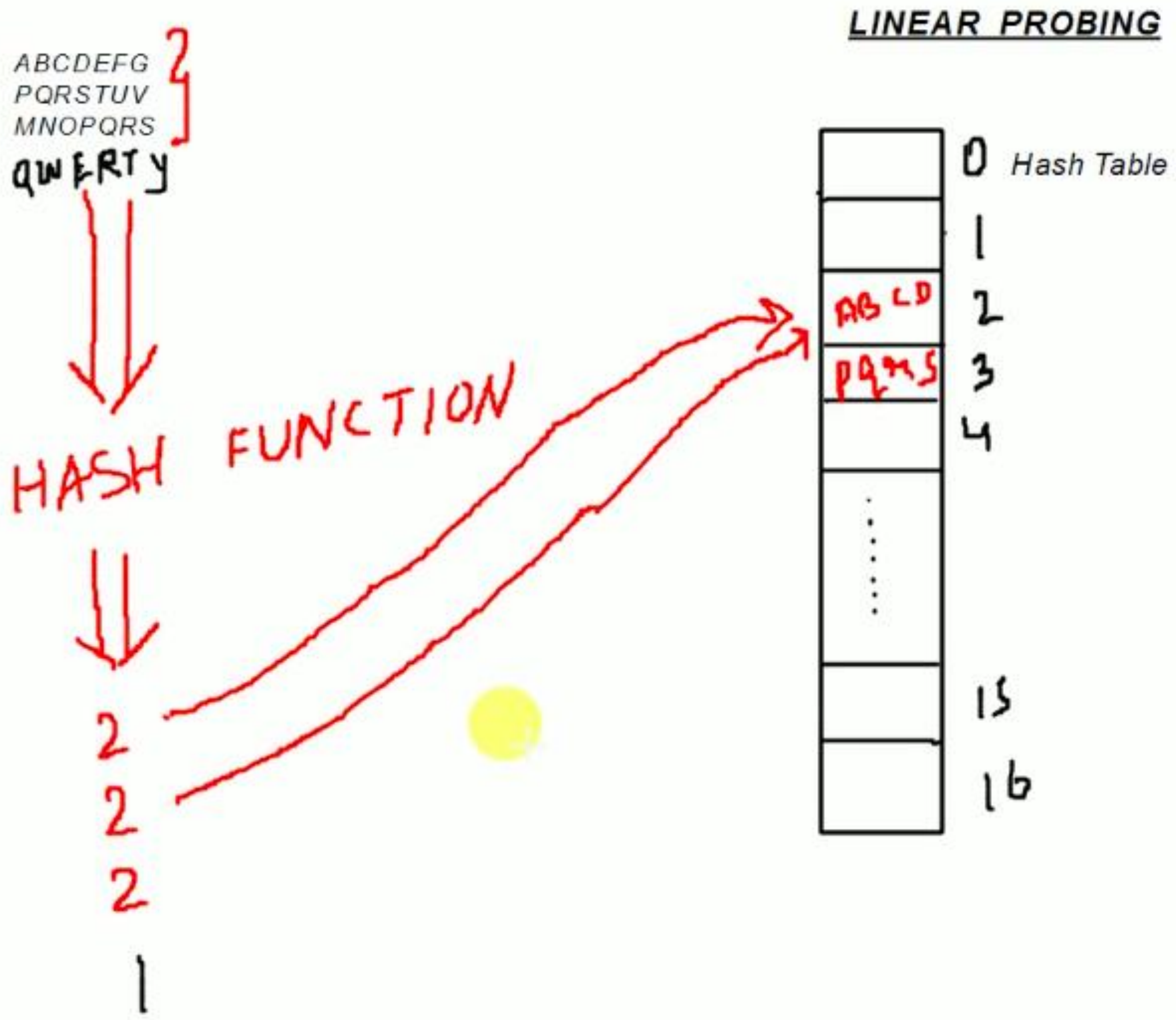


LINEAR PROBING

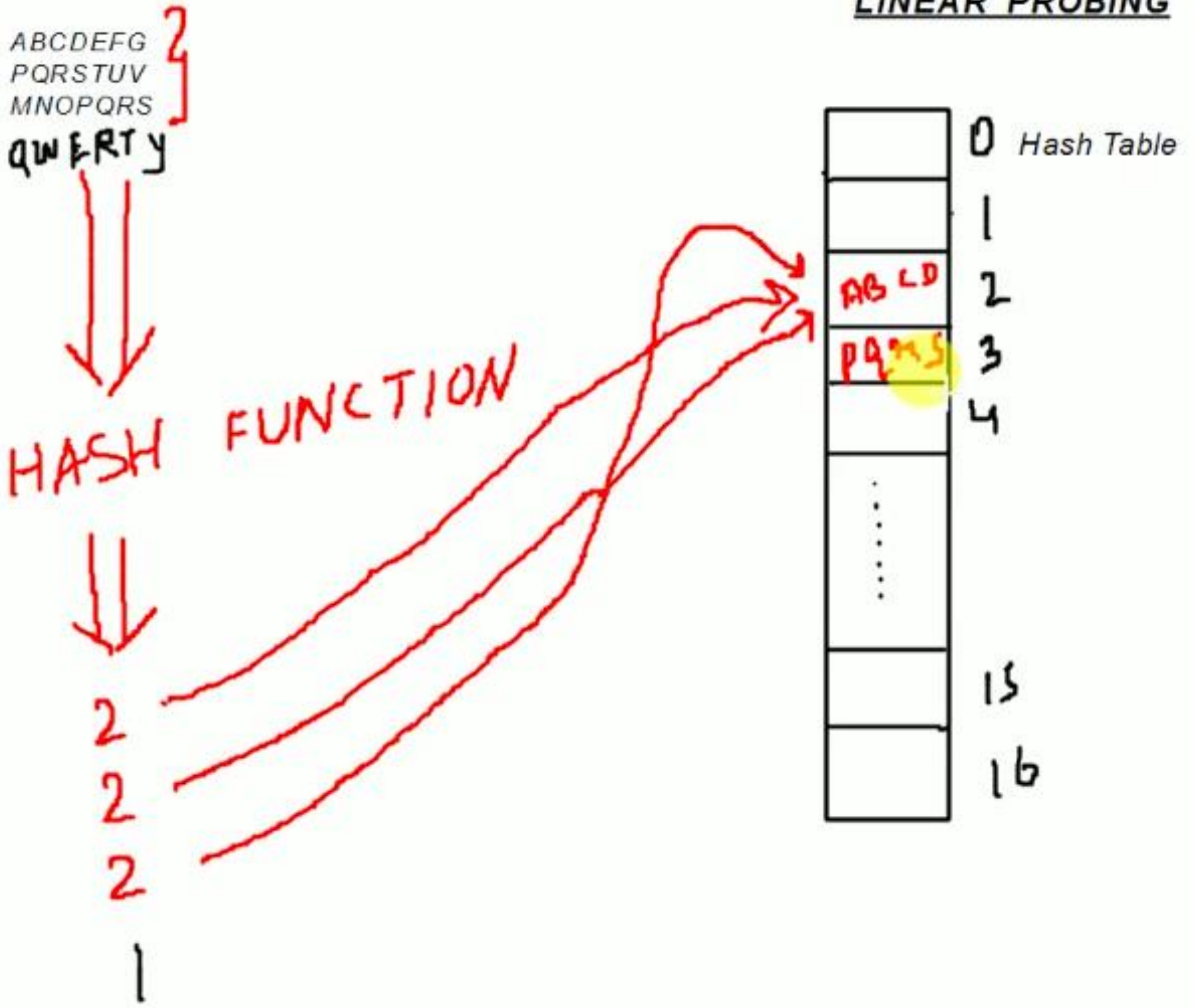


LINEAR PROBING

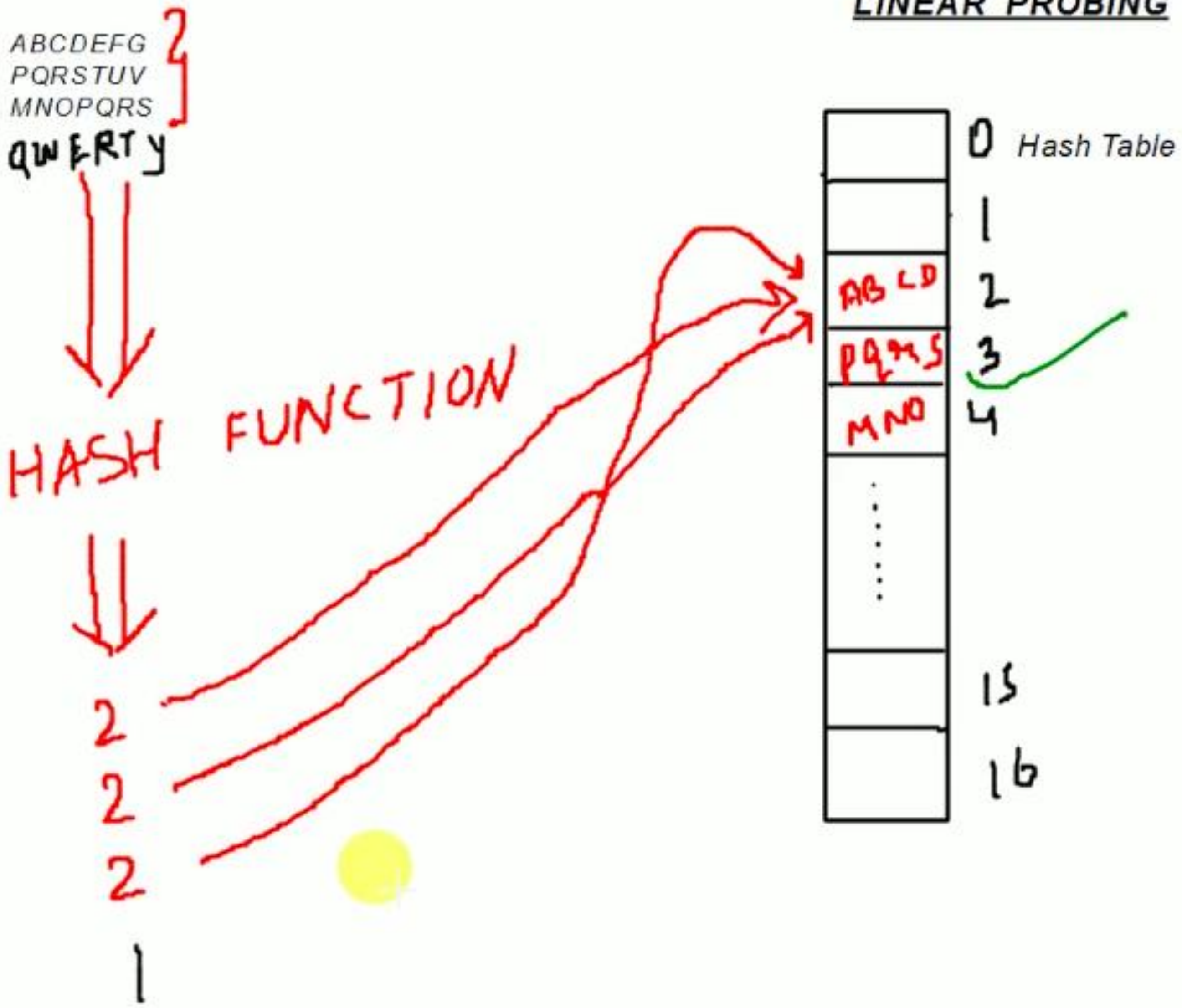



LINEAR PROBING
ABCDEFG
PQRSTUV
MNOPQRS
QWERTY
HASH FUNCTION
2
2
2
1
Hash Table
0
1
2
3
4
15
16

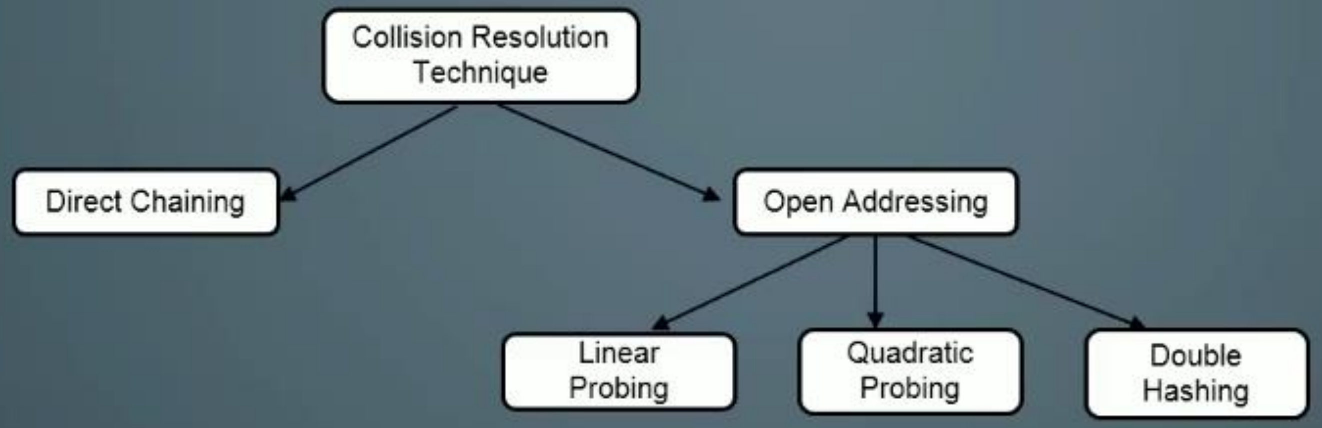
LINEAR PROBING



LINEAR PROBING



Collision Resolution Techniques:

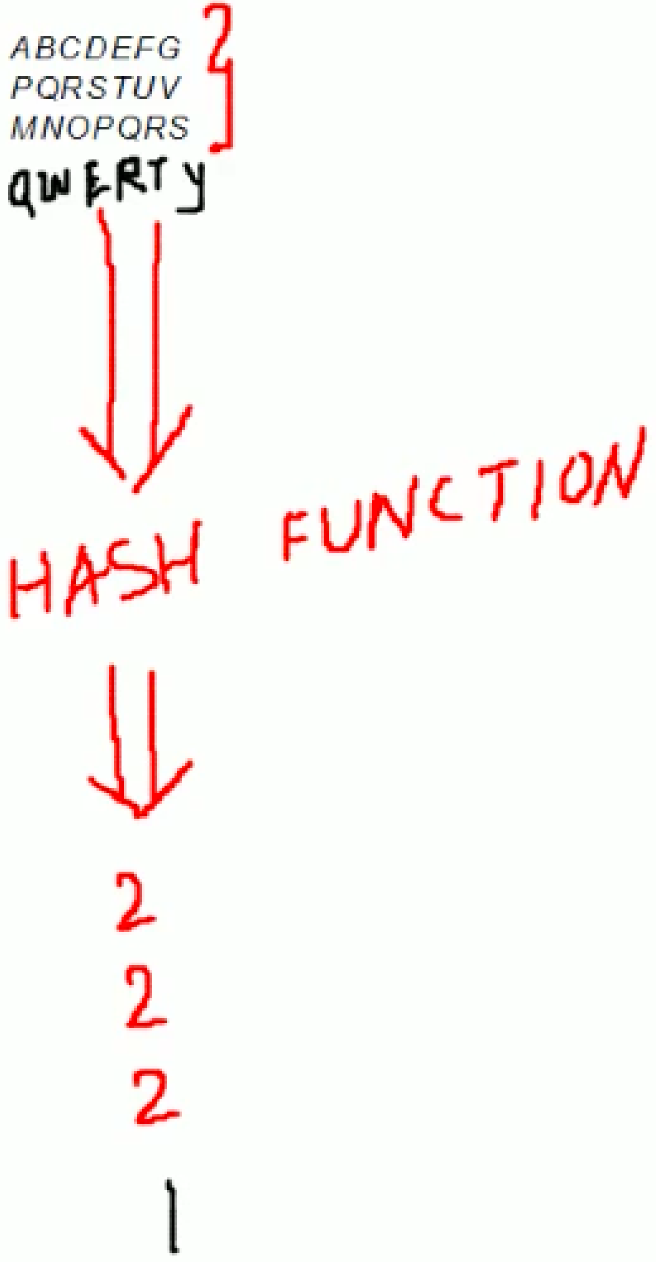


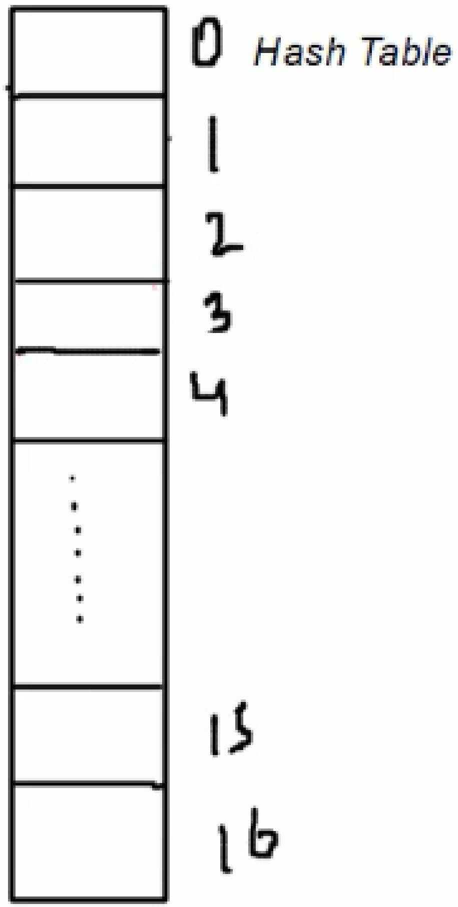
Direct Chaining: Implements the buckets as linked lists. Colliding elements are stored in these lists.

Open Addressing: Colliding elements are stored in other vacant buckets. During storage and lookup, these are found through so-called 'probing'

- Linear Probing:
  - Linear probing is a strategy for resolving collisions, by placing the new key into the closest following empty cell
- Quadratic Probing:
  - Quadratic probing operates by taking the original hash index and adding successive values of an arbitrary quadratic polynomial until an open slot is found.

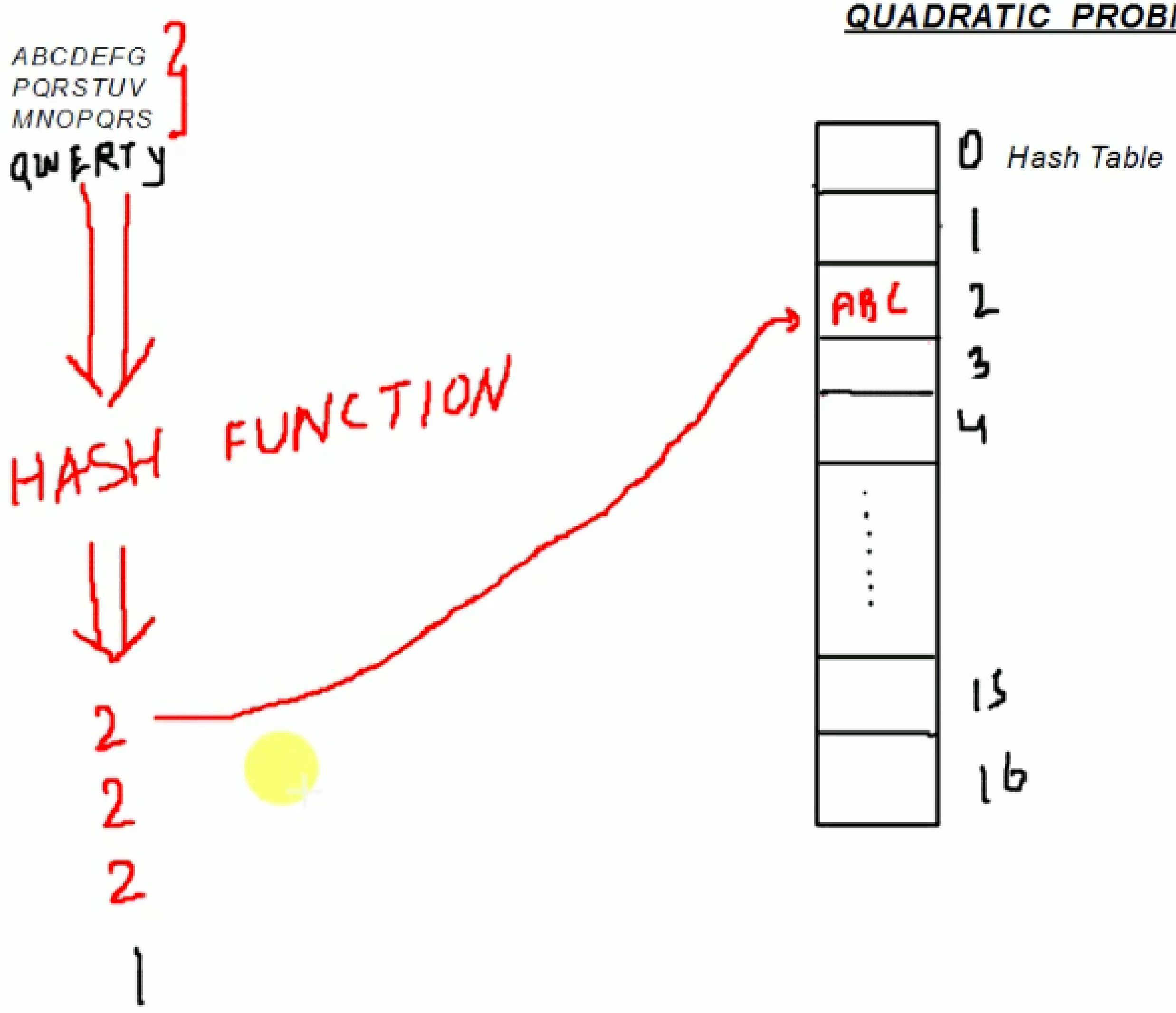
QUADRATIC PROBING





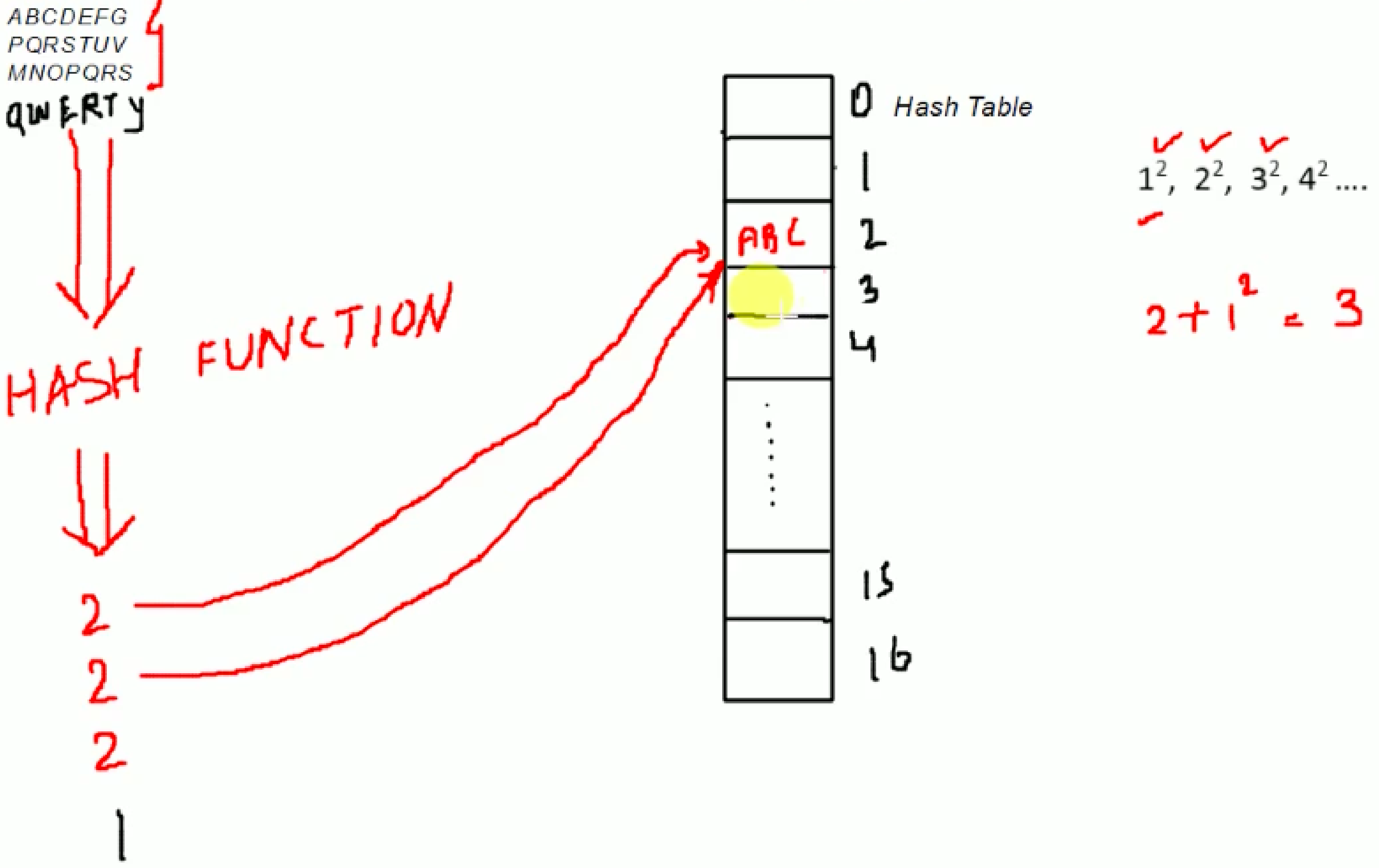
$1^2, 2^2, 3^2, 4^2 ....$

QUADRATIC PROBING

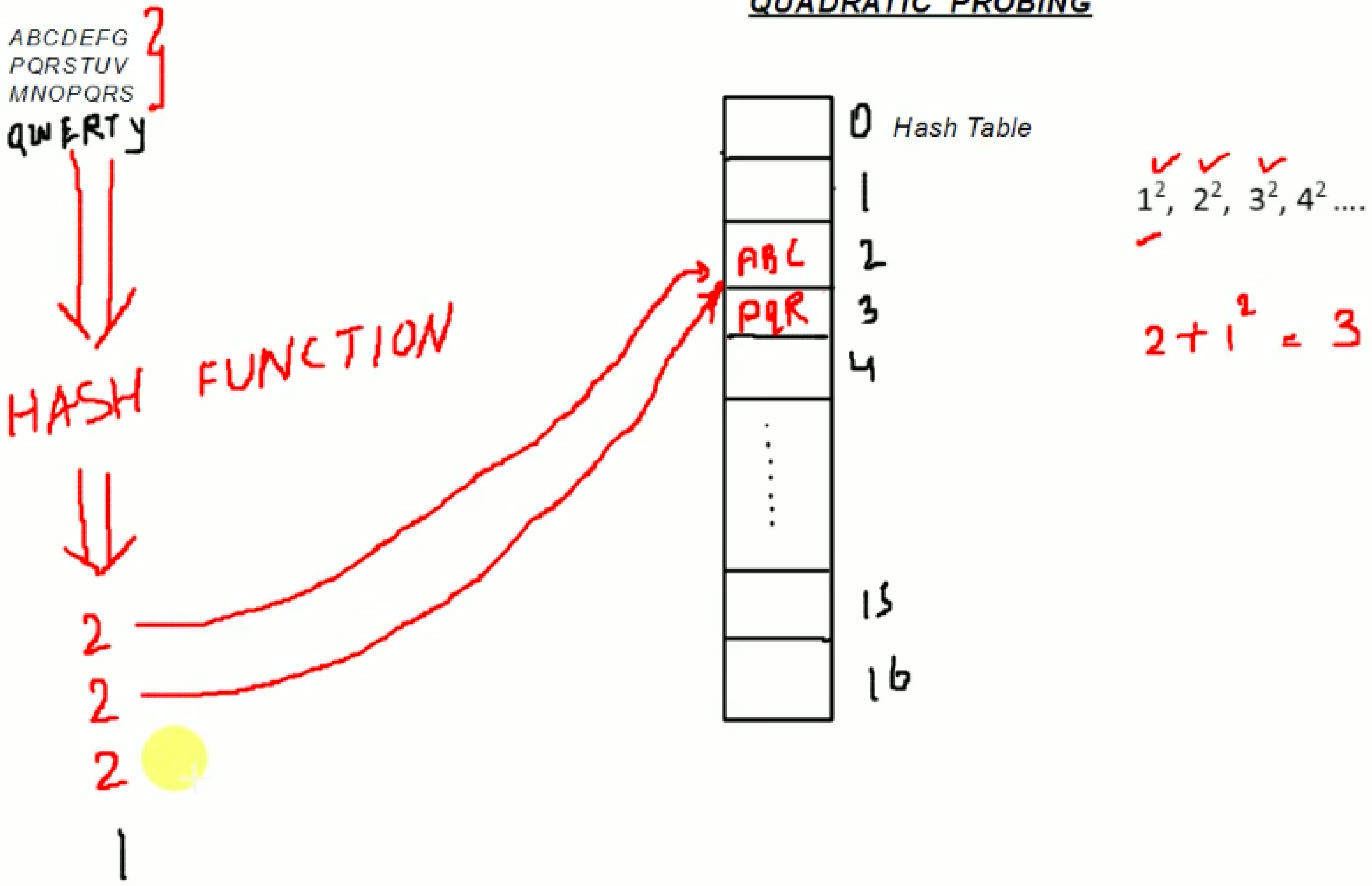


$1^2, 2^2, 3^2, 4^2 \ldots$

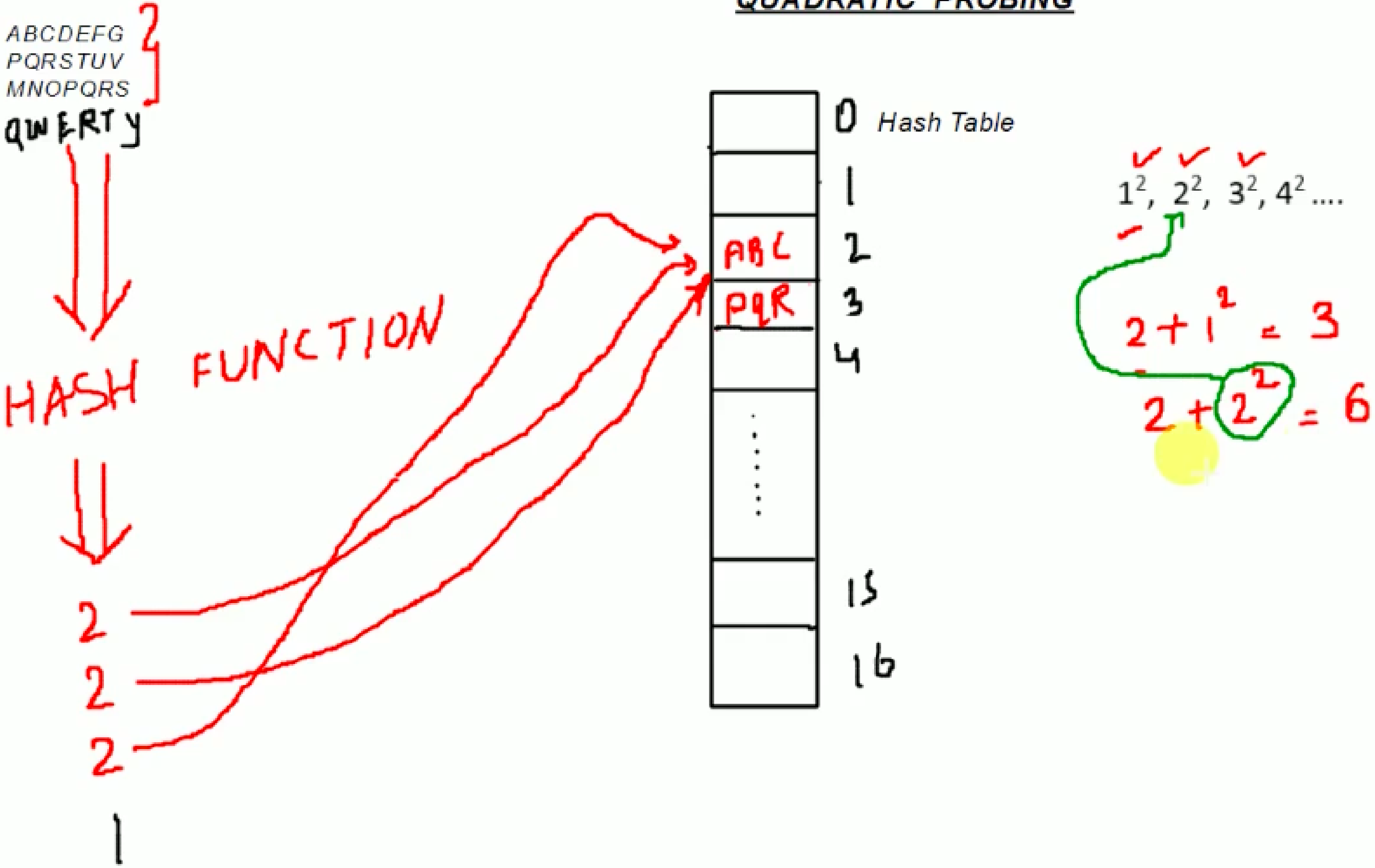
QUADRATIC PROBING



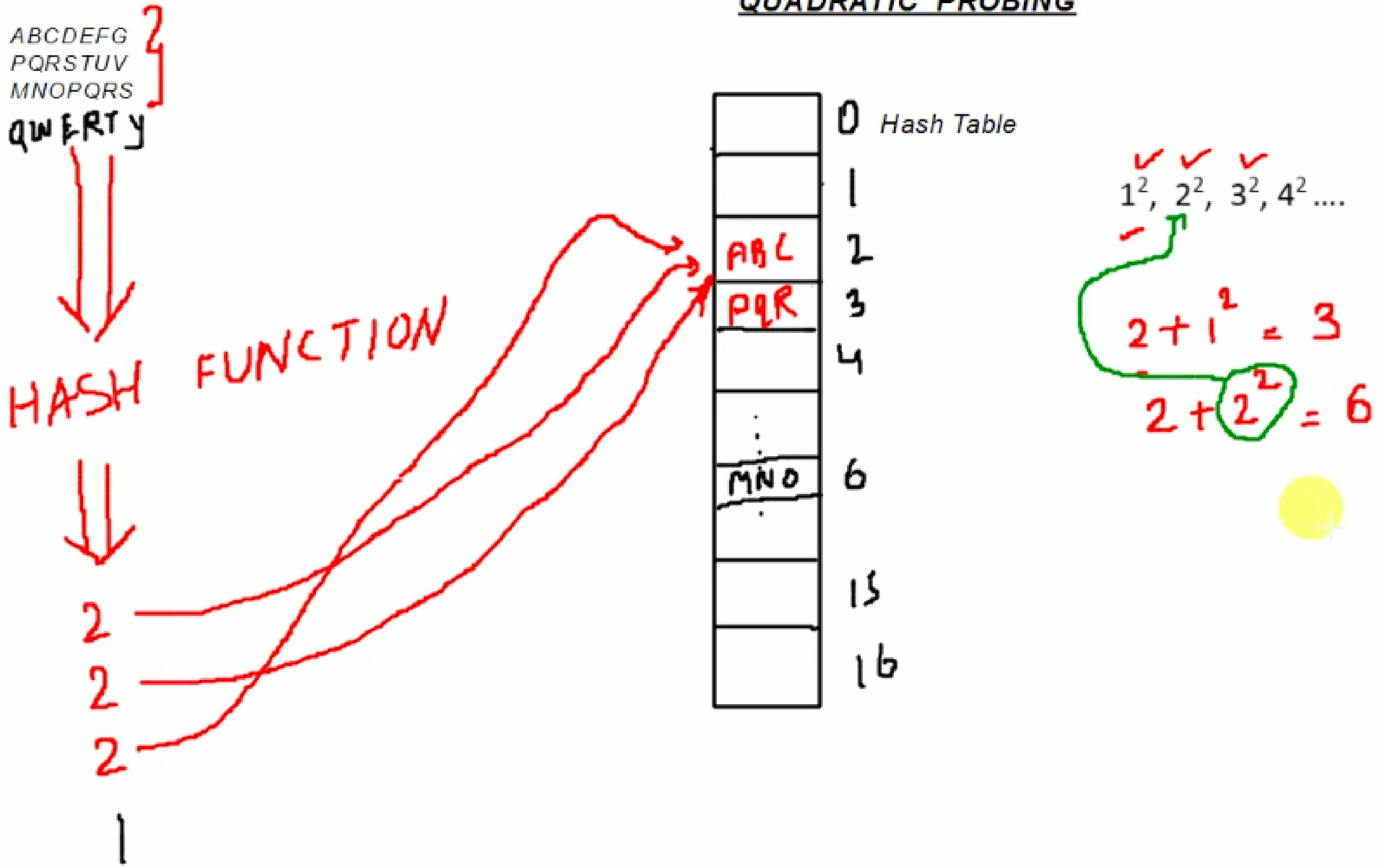
QUADRATIC PROBING



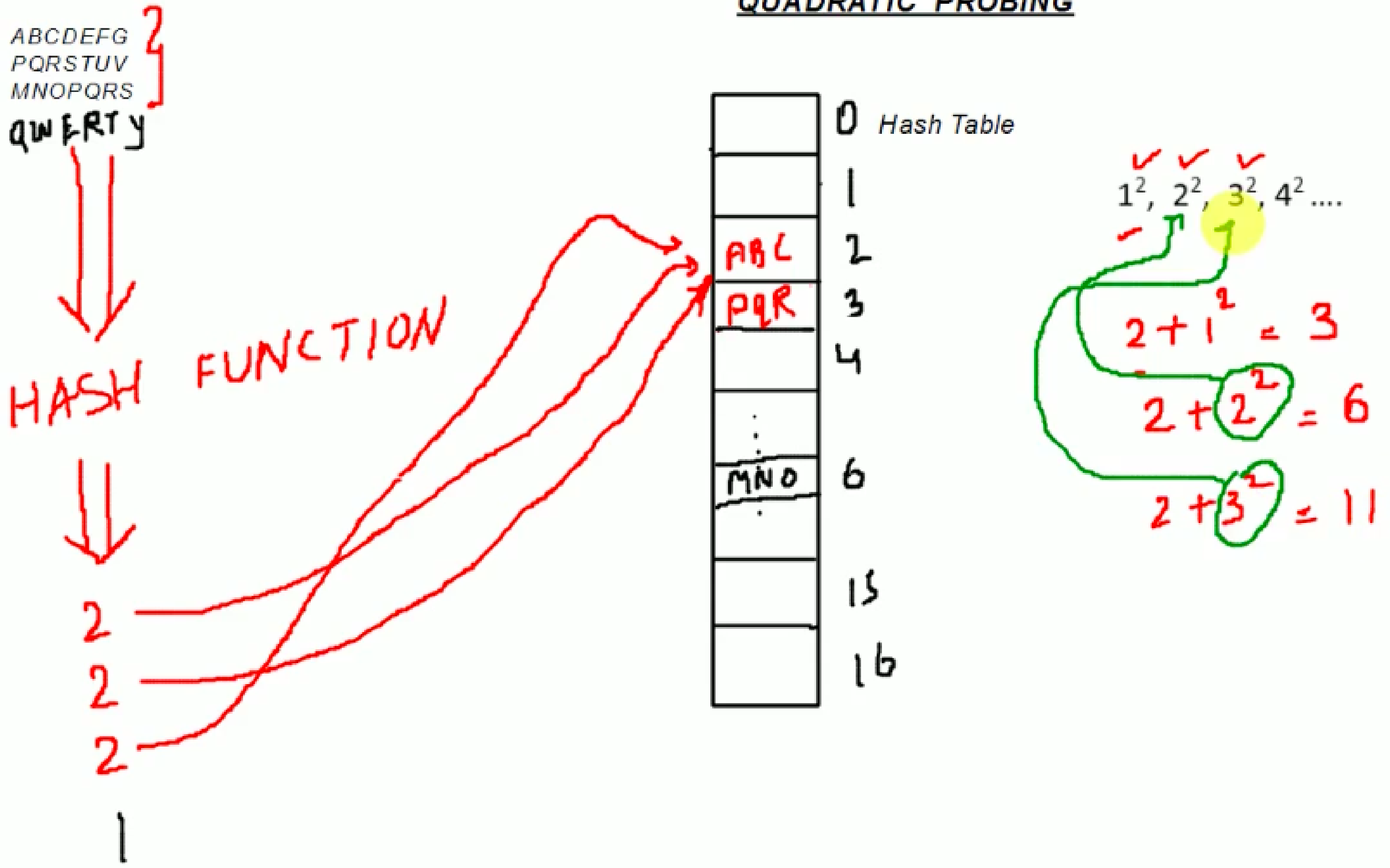
QUADRATIC PROBING



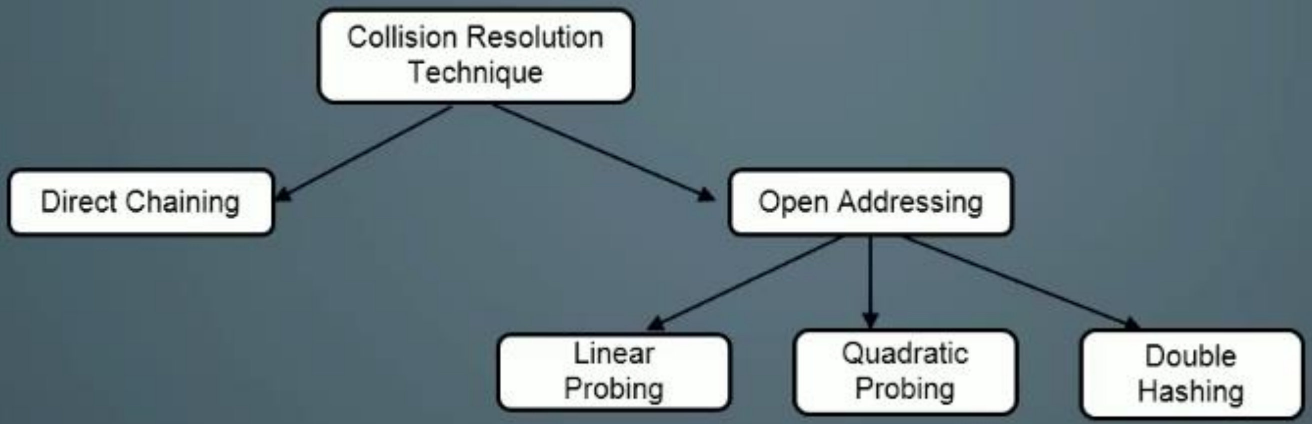
QUADRATIC PROBING



QUADRATIC PROBING



Collision Resolution Techniques:

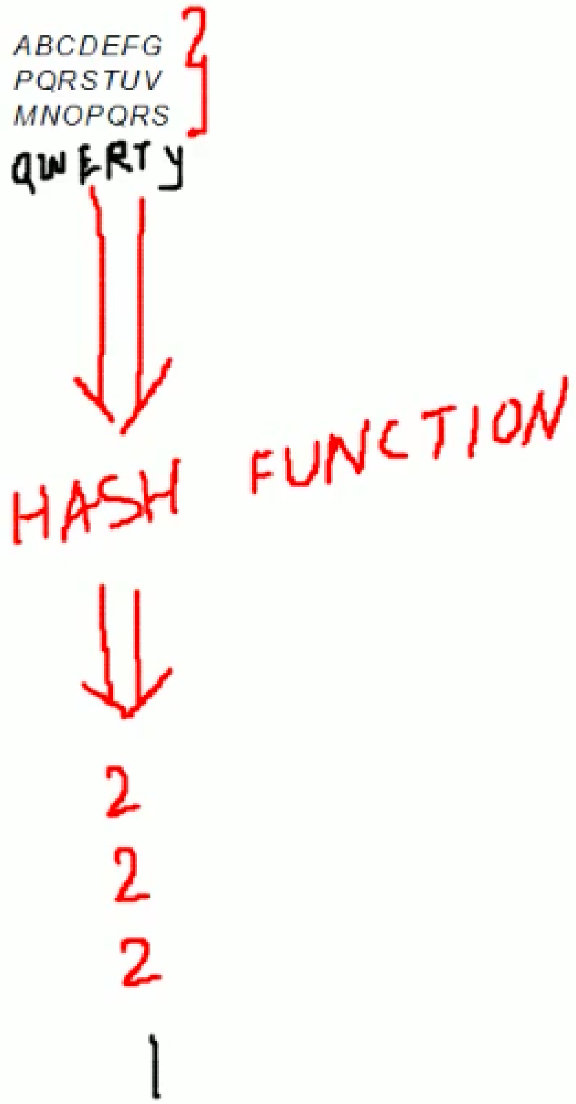


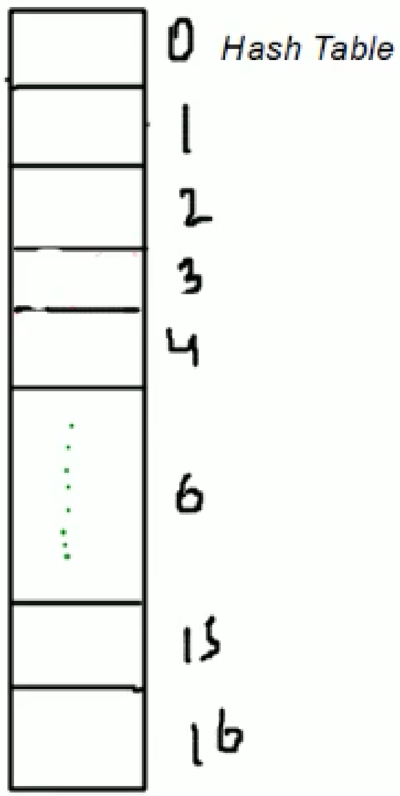
Direct Chaining: Implements the buckets as linked lists. Colliding elements are stored in these lists.

Open Addressing: Colliding elements are stored in other vacant buckets. During storage and lookup, these are found through so-called 'probing'

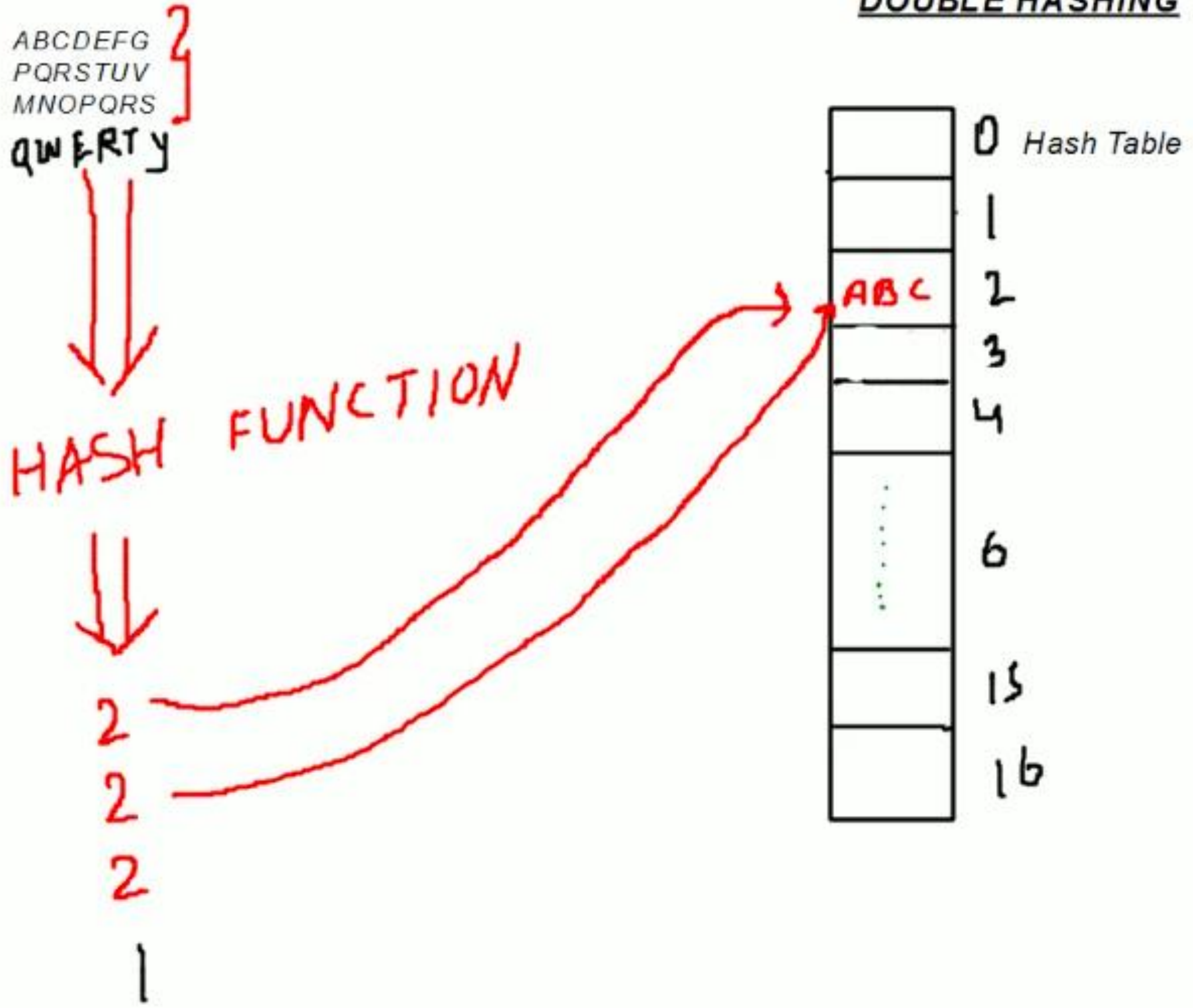
- Linear Probing:
  - Linear probing is a strategy for resolving collisions, by placing the new key into the closest following empty cell
- Quadratic Probing:
  - Quadratic probing operates by taking the original hash index and adding successive values of an arbitrary quadratic polynomial until an open slot is found.
- Double Hashing:
  - Interval between probes is computed by another hash function.

DOUBLE HASHING

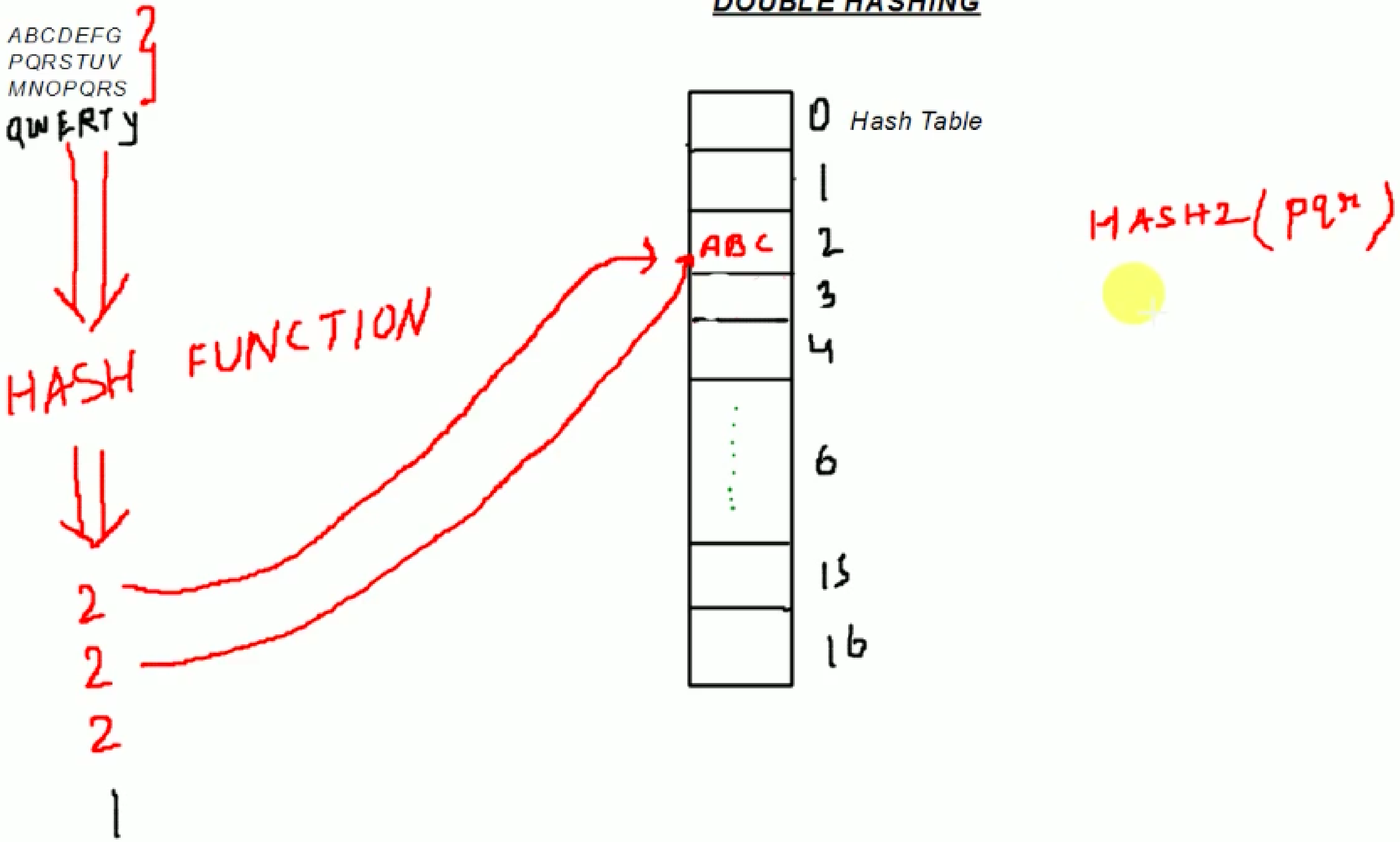




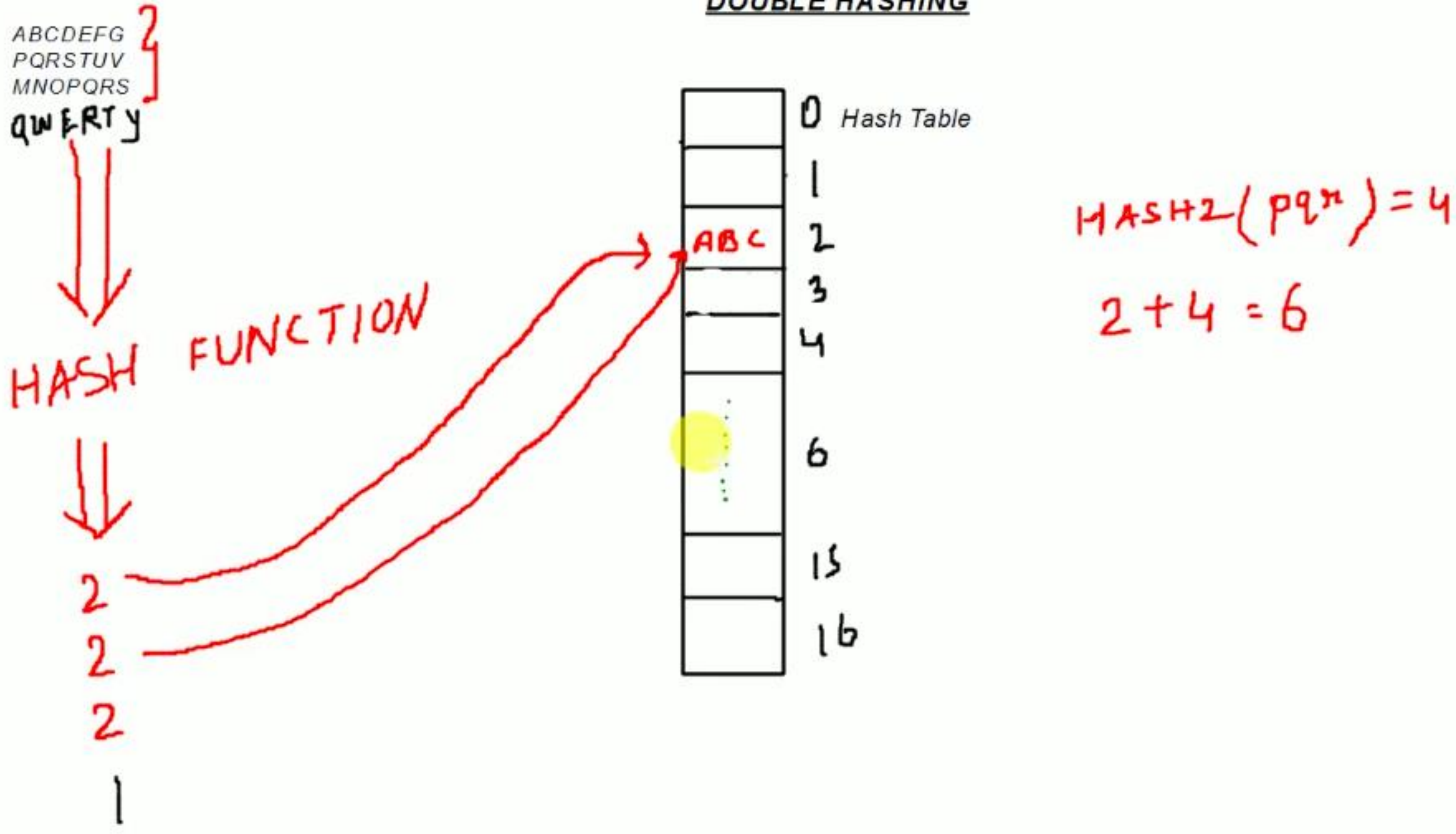
DOUBLE HASHING



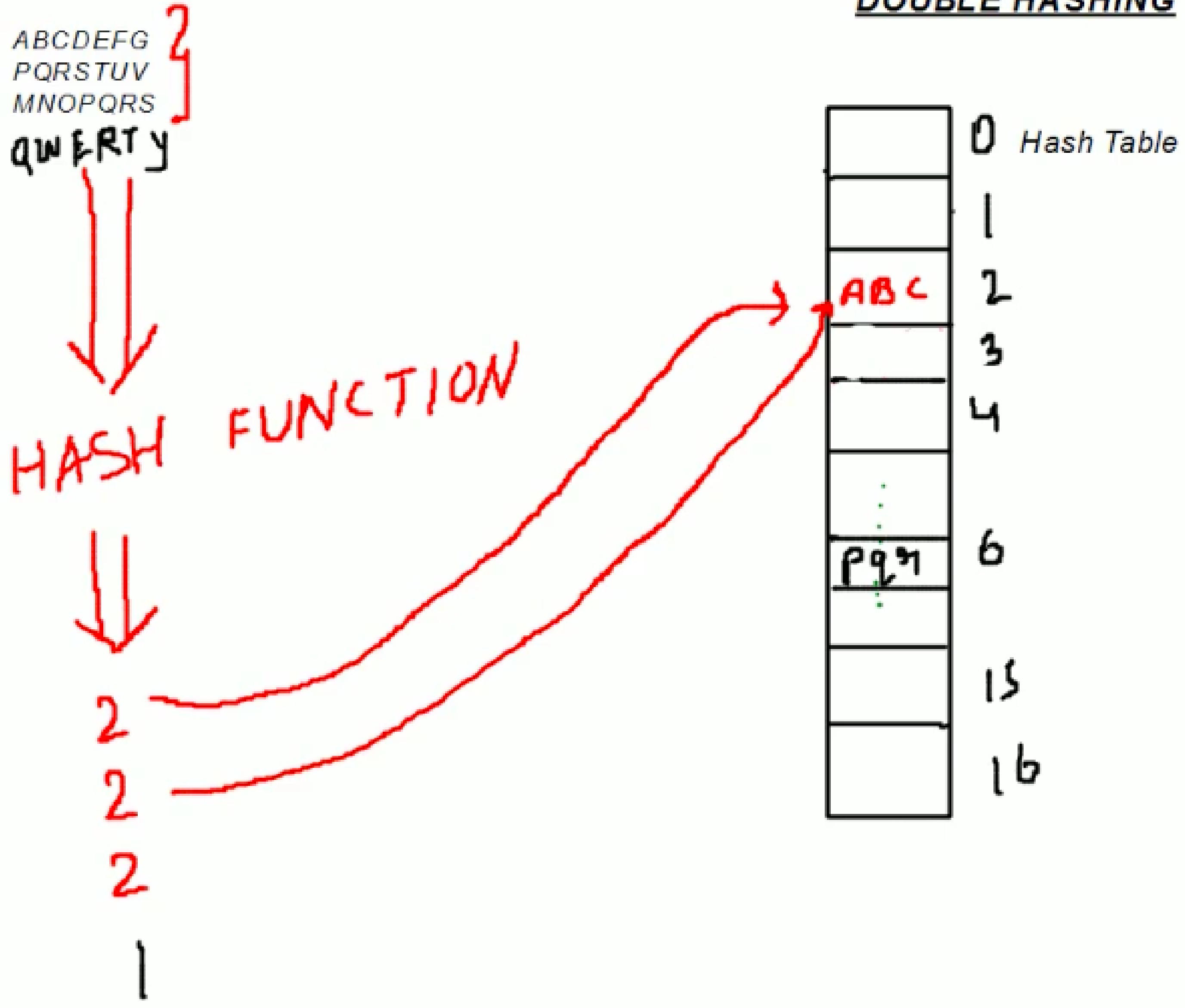
DOUBLE HASHING



DOUBLE HASHING



DOUBLE HASHING

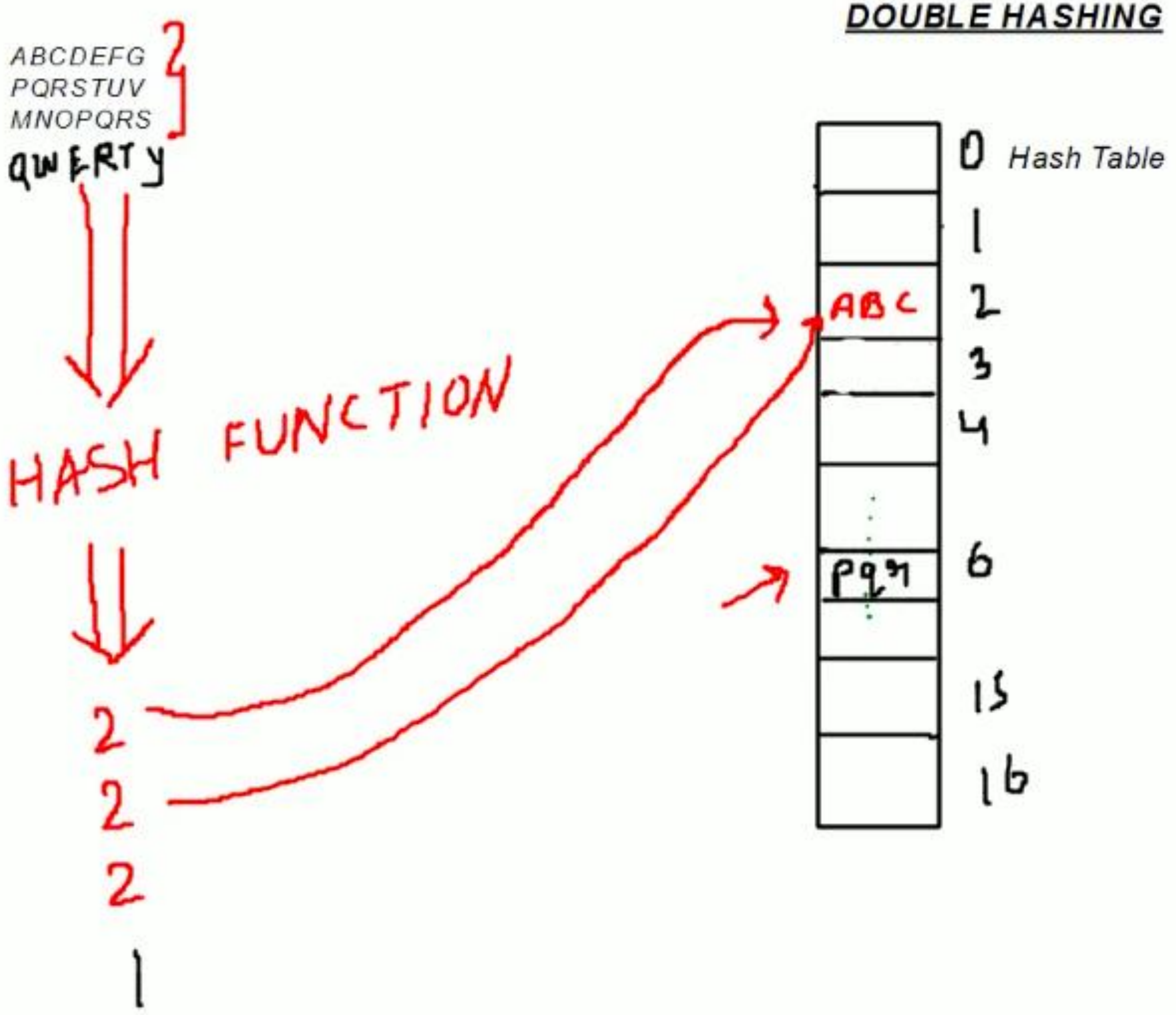


HASH2(pqr) = 4

2 + 4 = 6

HASH2(MNOP) = 4

DOUBLE HASHING



$HASH2(pqr) = 4$

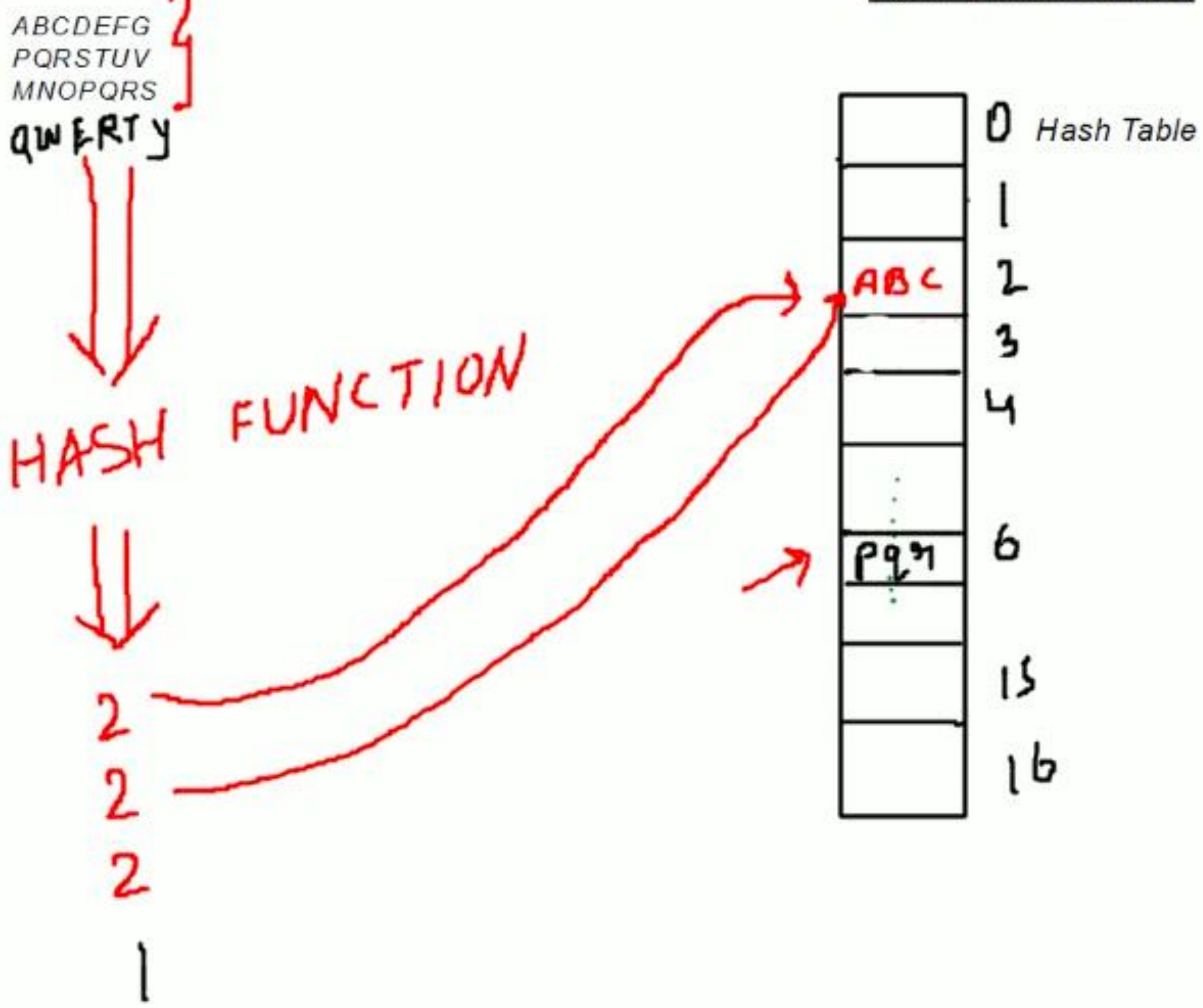
$2 + 4 = 6$

$HASH2(MNOP) = 4$

$2 + 4 = 6$

$2 + (2 \times 4) = 10$

DOUBLE HASHING



$HASH2(pqr) = 4$

$2 + 4 = 6$

$HASH2(MNOP) = 4$

$2 + 4 = 6$

$2 + (2 \times 4) = 10$

$2 + (3 \times 4) = 14$

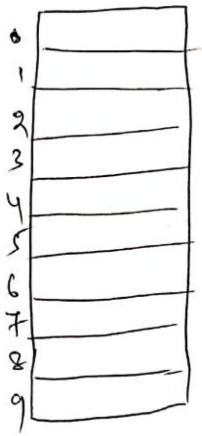
$A = 3, 2, 9, 6, 11, 13, 7, 12$

$h_1(k) = 2k+3 \quad m = 10$

$h_2(k) = 3k+1$

Insert k_i at first free place from $(u + v \times i) \% m$ where $[i = 0 \text{ to } (m-1)]$

$v = h_2(k) \% m$



Key	Location (u)	v	probe
3	$[2\times3+3]\%10 = 9$	–	1
2	$[2\times2+3]\%10 = 7$	–	1
9	$[2\times9+3]\%10 = 1$	–	1
6	$[2\times6+3]\%10 = 5$	–	1
11	$[2\times11+3]\%10 = 5$	4	3
13	$[2\times13+3]\%10 = 9$		
7	$[2\times7+3]\%10 = 7$	2	
12	$[2\times12+3]\%10 = 7$		

$[3\times13+1]\%10 = 0$

$[9+0\times0]\%10 = 9$

$[9+0\times9]\%10 = 9$

$v(11) = [3\times11+1]\%10 = 4$

$(u + v\times i)\%m$

$[5+4*0]\%10 = 5$

$[5+4*1]\%10 = 9$

$[5+4*2]\%10 = 3$ ✓

v

$[3\times7+1]\%10 = 2$

$[7+2\times0]\%10 = 7$

$[7+2\times1]\%10 = 9$

$[7+2\times2]\%10 = 1$

$[7+2\times3]\%10 = 3$

$[7+2\times4]\%10 = 5$

$[7+2\times5]\%10 = 7$

$[7+2\times6]\%10 = 9$

$[7+2\times7]\%10 = 1$

$[7+2\times8]\%10 = 3$

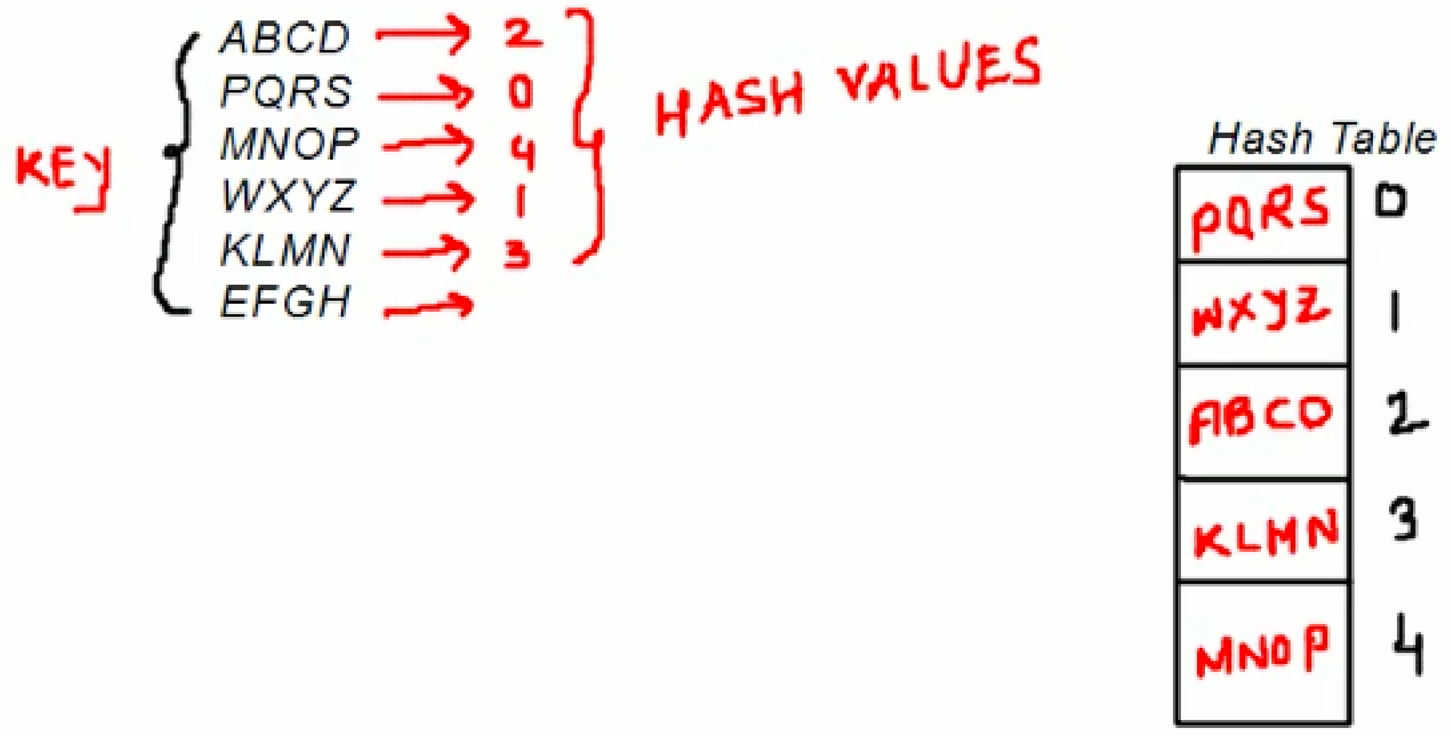
$[3\times12+1]\%10 = 7$

$[7+7\times0]\%10 = 7$

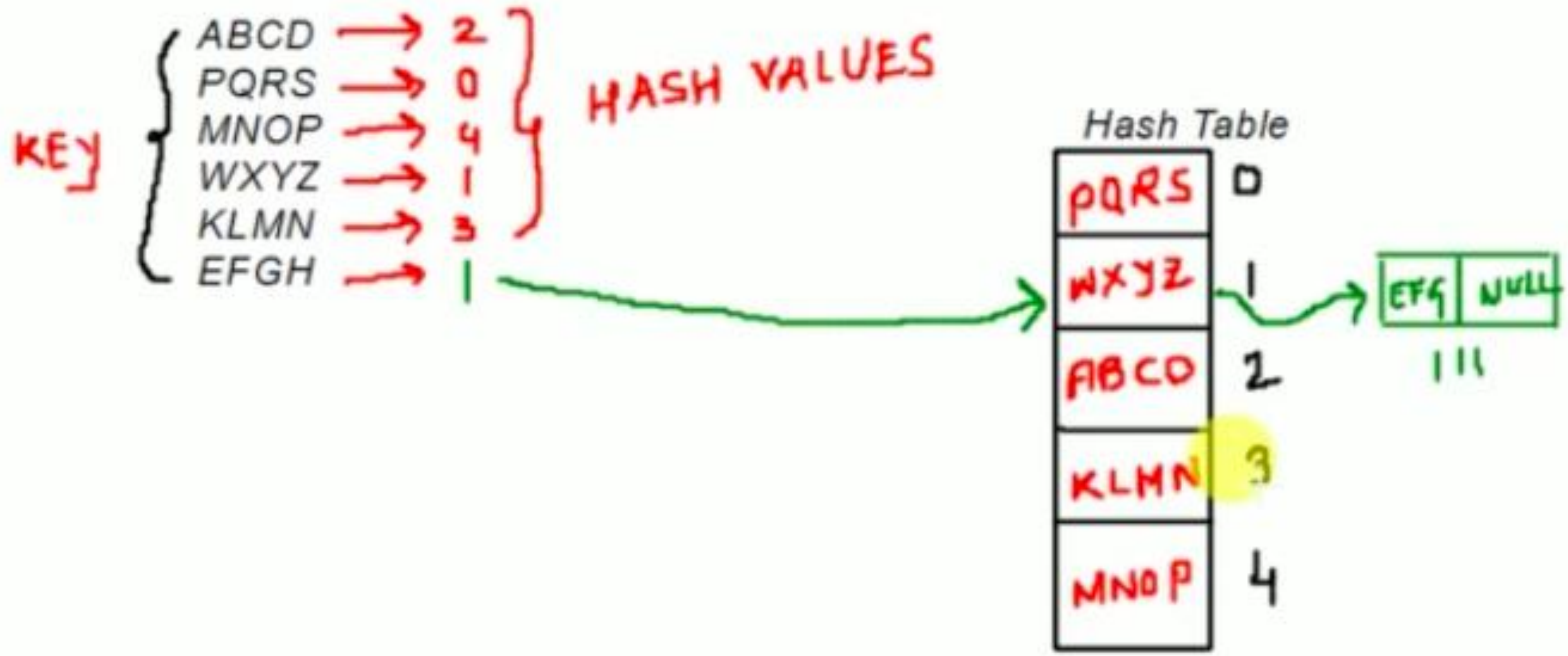
$[7+7\times1]\%10 = 4$

What happens when Hash Table is full ?

- Direct Chaining:
  - This situation will never arise.



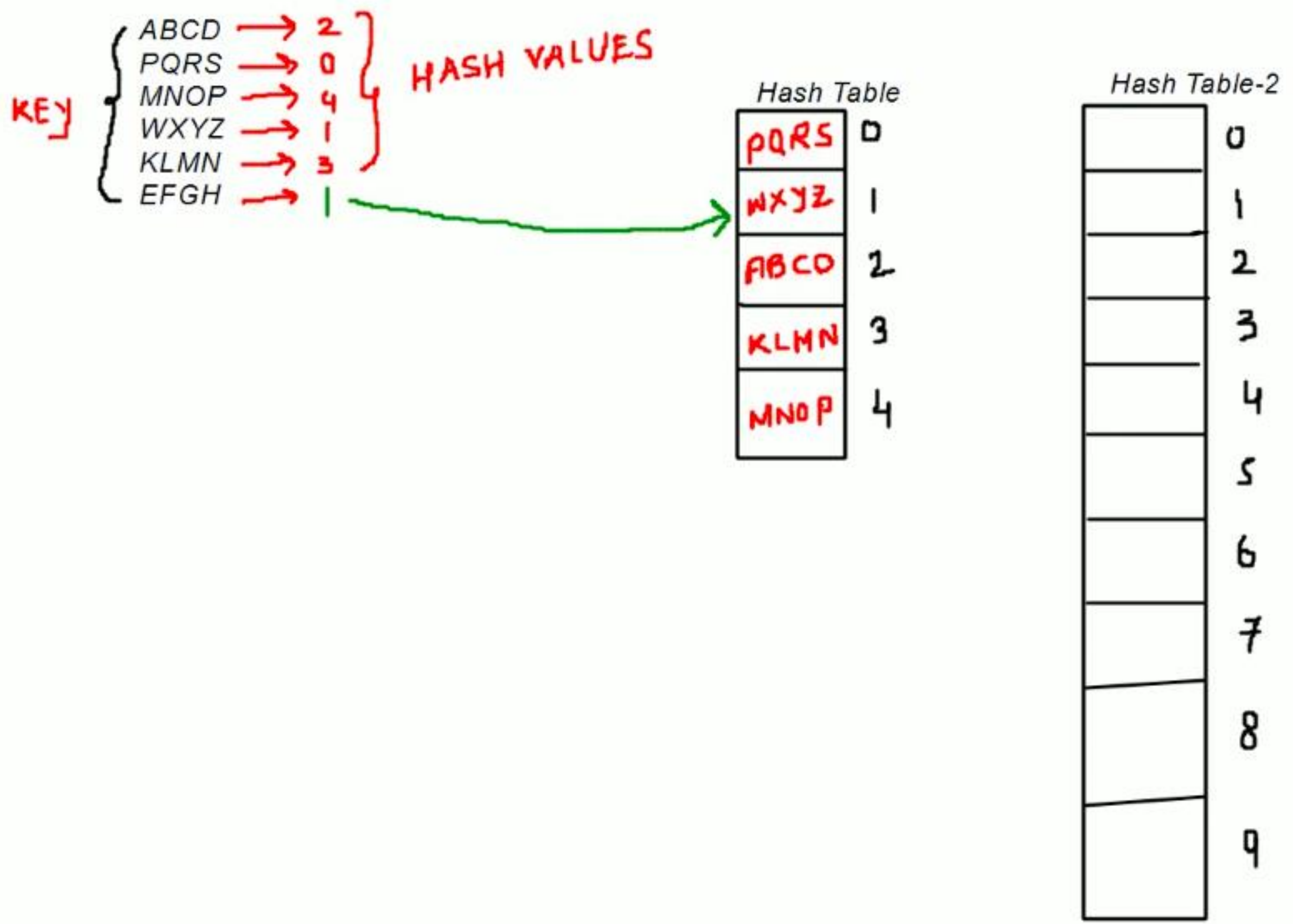
KEY
ABCD
PQRS
MNOP
WXYZ
KLMN
EFGH
2
0
4
1
3
HASH VALUES
Hash Table
PQRS
WXYZ
ABCD
KLMN
MNOP
0
1
2
3
4



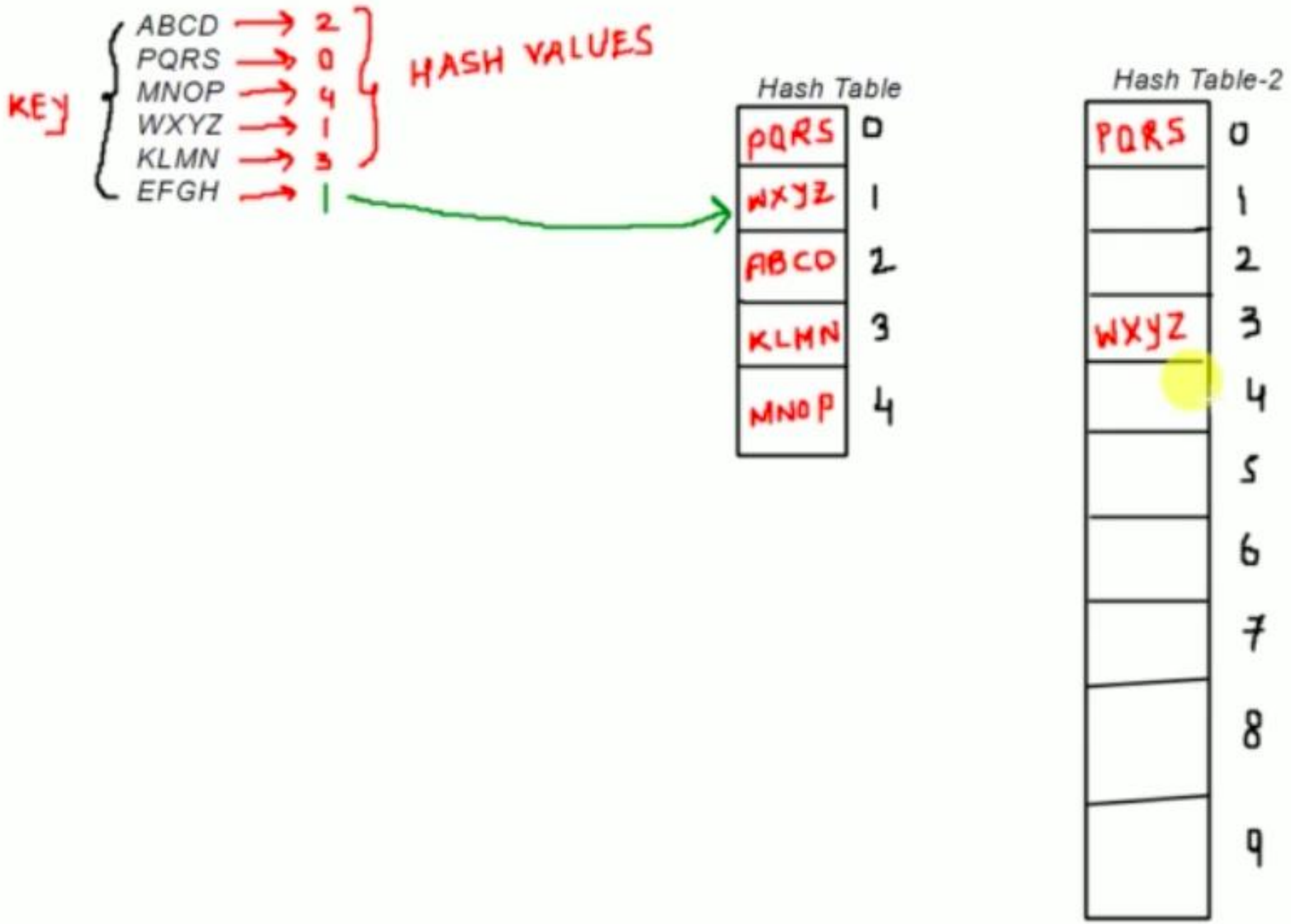
ABCD → 2
PQRS → 0
MNOP → 4
WXYZ → 1
KLMN → 3
EFGH → 1
KEY
HASH VALUES
Hash Table
PQRS 0
WXYZ 1
ABCD 2
KLMN 3
MNOP 4
EFG NULL
111

What happens when Hash Table is full ?

- Direct Chaining:
  - This situation will never arise.
- Open Addressing:
  - Need to create 2X size of current Hash Table and redo Hashing for existing keys.



KEY
ABCD → 2
PQRS → 0
MNOP → 4
WXYZ → 1
KLMN → 3
EFGH → 1
HASH VALUES
Hash Table
PQRS 0
WXYZ 1
ABCD 2
KLMN 3
MNOP 4
Hash Table-2
0
1
2
3
4
5
6
7
8
9

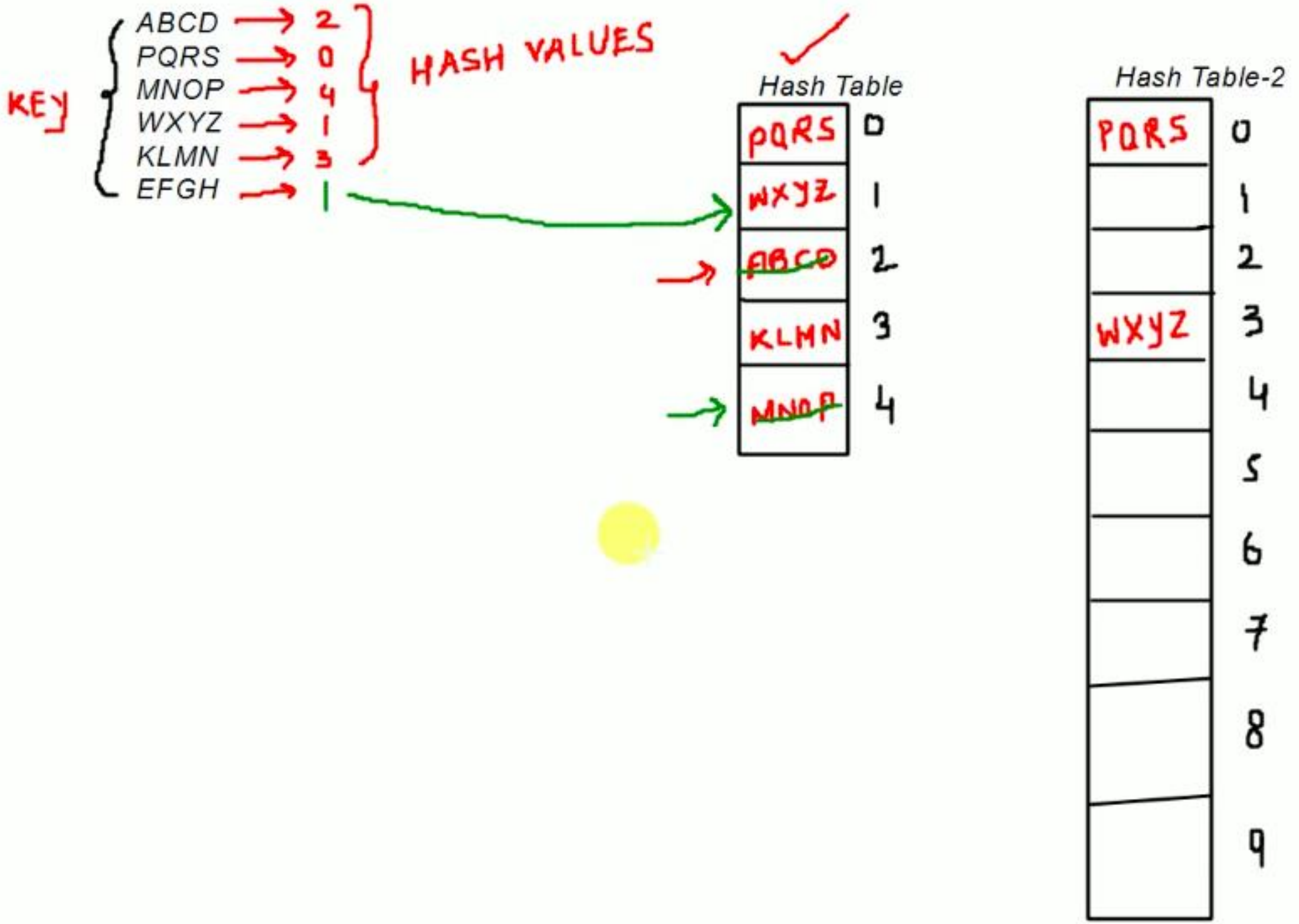


KEY
ABCD → 2
PQRS → 0
MNOP → 4
WXYZ → 1
KLMN → 3
EFGH → 1
HASH VALUES
Hash Table
PQRS 0
WXYZ 1
ABCD 2
KLMN 3
MNOP 4
Hash Table-2
PQRS 0
1
2
WXYZ 3
4
5
6
7
8
9

Pros & Cons of Collision Resolution Technique:

- Direct Chaining:
  - No fear of exhausting Hash Table buckets.
  - Fear of big Linked Lists (can effect performance big time).
- Open Addressing:
  - Easy implementation.
  - Fear of exhausting Hash Table buckets

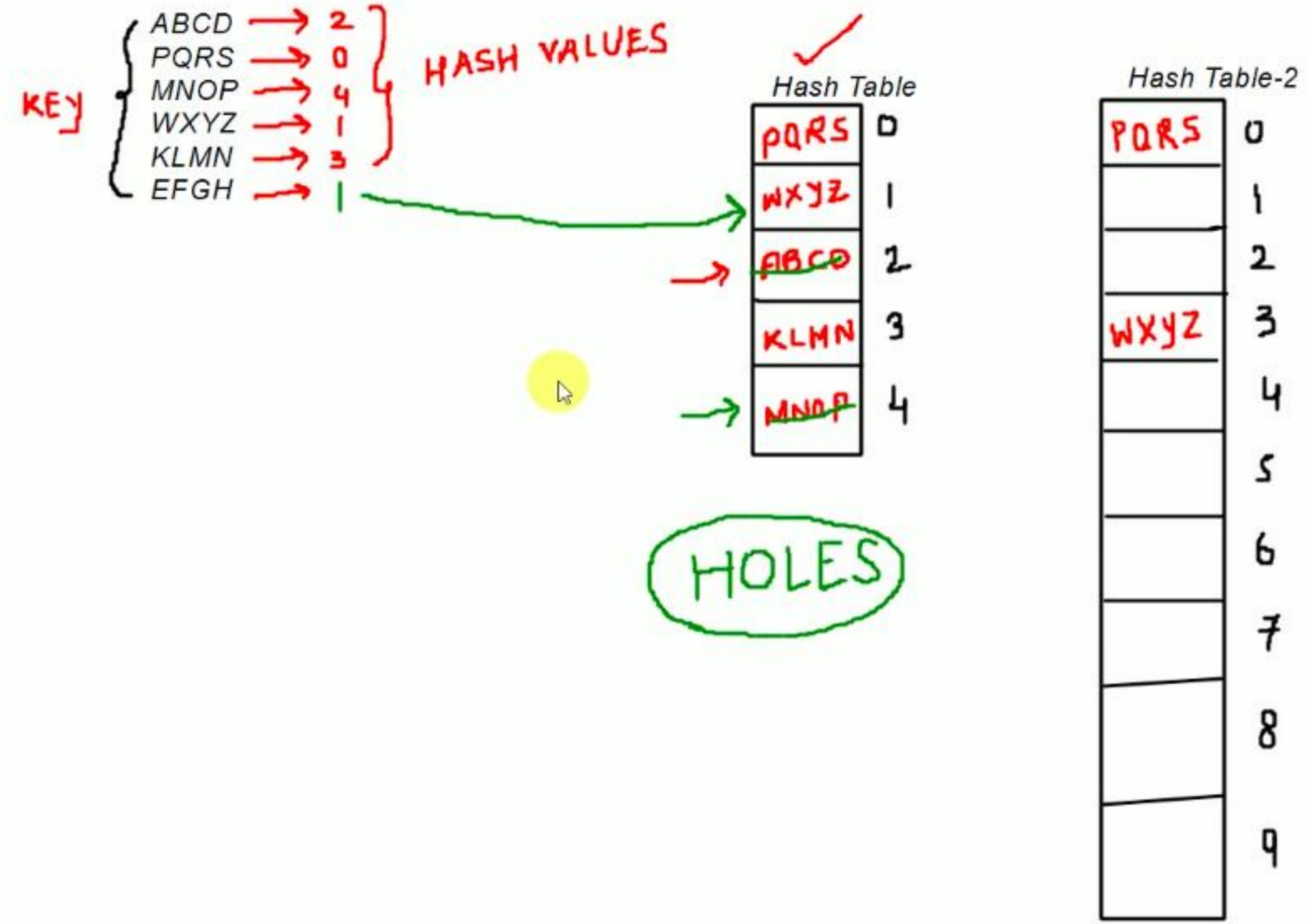
- If Input size is known then always use “Open Addressing”, else can use any of the two.


KEY
ABCD → 2
PQRS → 0
MNOP → 4
WXYZ → 1
KLMN → 3
EFGH → 1
HASH VALUES
Hash Table
PQRS 0
WXYZ 1
ABCD 2
KLMN 3
MNOP 4
Hash Table-2
PQRS 0
1
2
WXYZ 3
4
5
6
7
8
9

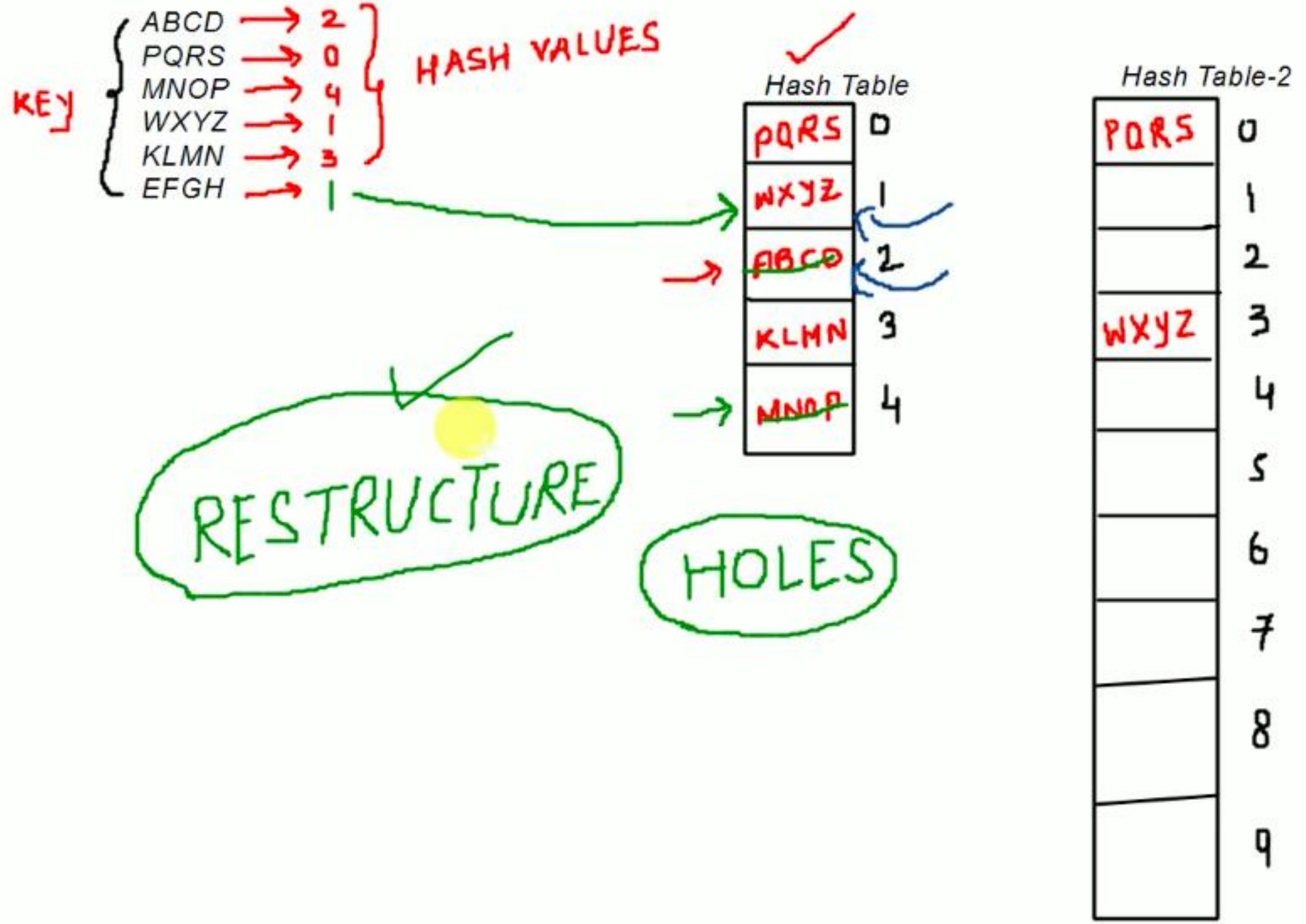
Pros & Cons of Collision Resolution Technique:

- Direct Chaining:
    - No fear of exhausting Hash Table buckets.
    - Fear of big Linked Lists (can effect performance big time).
- Open Addressing:
    - Easy implementation.
    - Fear of exhausting Hash Table buckets

- If Input size is known then always use "Open Addressing", else can use any of the two.
- If Deletion is very high, then we should always go for 'Direct Chaining'.



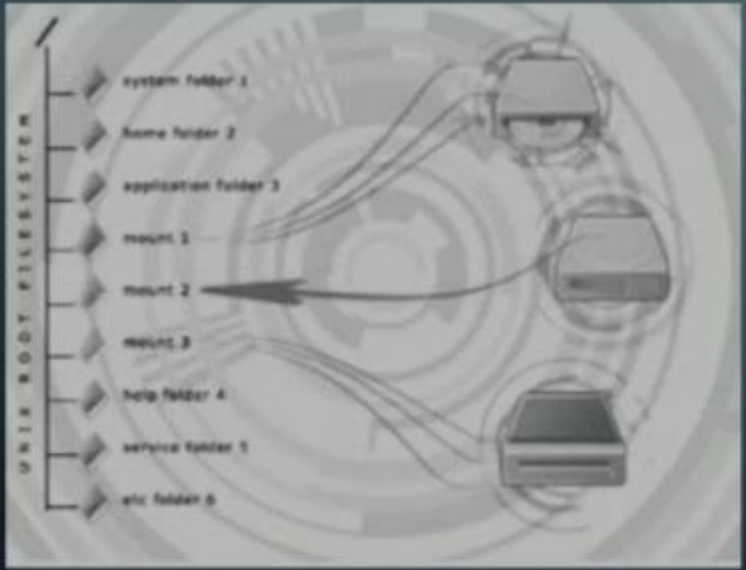
ABCD → 2
PQRS → 0
MNOP → 4
WXYZ → 1
KLMN → 3
EFGH → 1
KEY
HASH VALUES
Hash Table
PQRS 0
WXYZ 1
ABCD 2
KLMN 3
MNOP 4
HOLES
Hash Table-2
PQRS 0
1
2
WXYZ 3
4
5
6
7
8
9


KEY
ABCD → 2
PQRS → 0
MNOP → 4
WXYZ → 1
KLMN → 3
EFGH → 1
HASH VALUES
Hash Table
PQRS 0
WXYZ 1
ABCD 2
KLMN 3
MNOP 4
RESTRUCTURE
HOLES
Hash Table-2
PQRS 0
1
2
WXYZ 3
4
5
6
7
8
9

Practical Use of Hashing:

- ✓ Password Verification:

- ✓ File Systems : File path is mapped to physical location on disk



Personal Computer

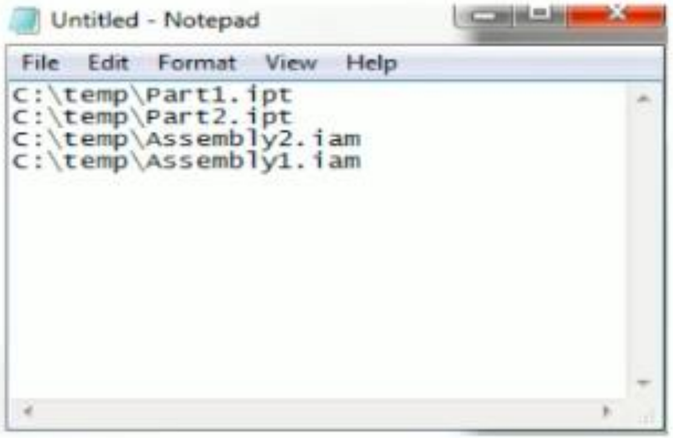
Username: abc@bla.com
Password: 123456

FACEBOOK's SERVER

1. Save the password at it is. i.e 123456

2. Convert the Key (password) into hash value and save the hash value instead of password. say ruh67#87Fg6yhe@^%!

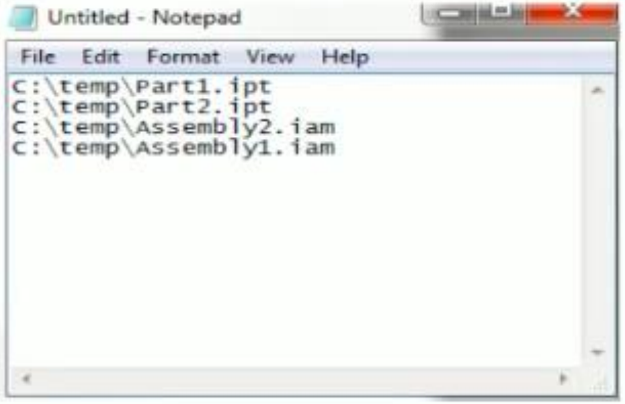
List of Files in temp folder



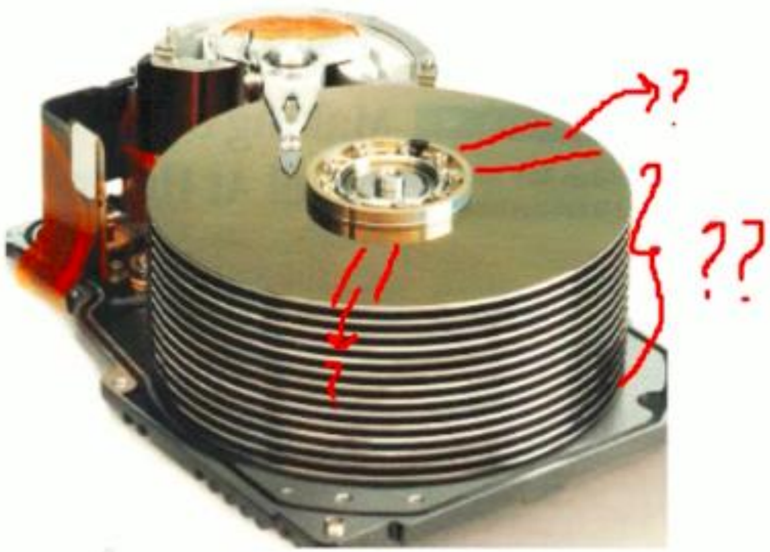
Hard Disk

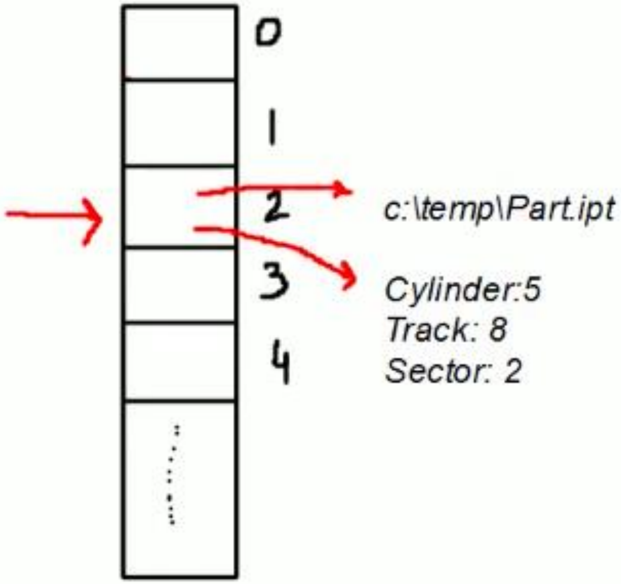
List of Files in temp folder

KEY


Untitled - Notepad
File Edit Format View Help
```
C:\temp\Part1.ipt
C:\temp\Part2.ipt
C:\temp\Assembly2.iam
C:\temp\Assembly1.iam
```

Hard Disk





Pros & Cons of Hashing:

- Pros:
  - On an average Insertion/Deletion/Search operation takes O(1) time.
- Cons:
  - In the worst case Insertion/Deletion/Search might take O(n) time (when hash functions is not good enough).

Hashing vs other DS:

Particulars	Array	Linked List	Tree	Hashing
Insertion	O(n)	O(n)	O(Log n)	O(1) / O(n)
Deletion	O(n)	O(n)	O(Log n)	O(1) / O(n)
Searching	O(n)	O(n)	O(Log n)	O(1) / O(n)


Thank you