

# CẤU TRÚC DỮ LIỆU & THUẬT TOÁN CHUYÊN SÂU

## COMPUTER SCIENCE 3

# Bài toán thực tế

# Bài toán thực tế



\* Tìm kiếm trong hàng ngàn, triệu sản phẩm

\* Đáng giá và đề xuất sản phẩm tương tự

\* Quản lý kho hàng, người dùng đặt hàng

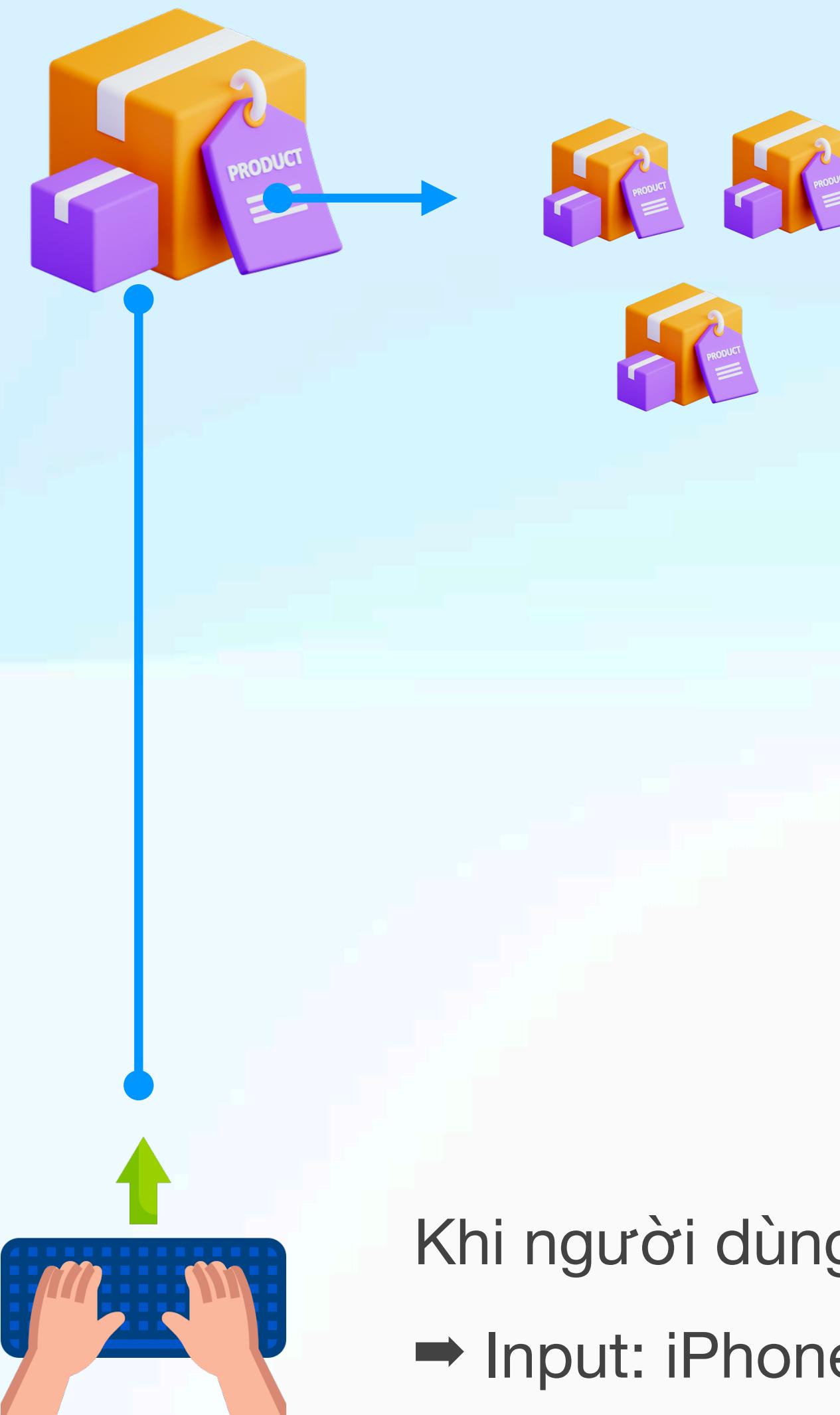
\* Gợi ý tự động điền khi tìm kiếm tên sản phẩm

## \* Đánh giá và đề xuất sản phẩm tương tự

Tên sản phẩm

Danh mục

Đơn giá



Hệ thống trả danh sách sản phẩm tương tự.

→ Output: Galaxy Z Fold, Nokia lumia, Oppo Neo, ...

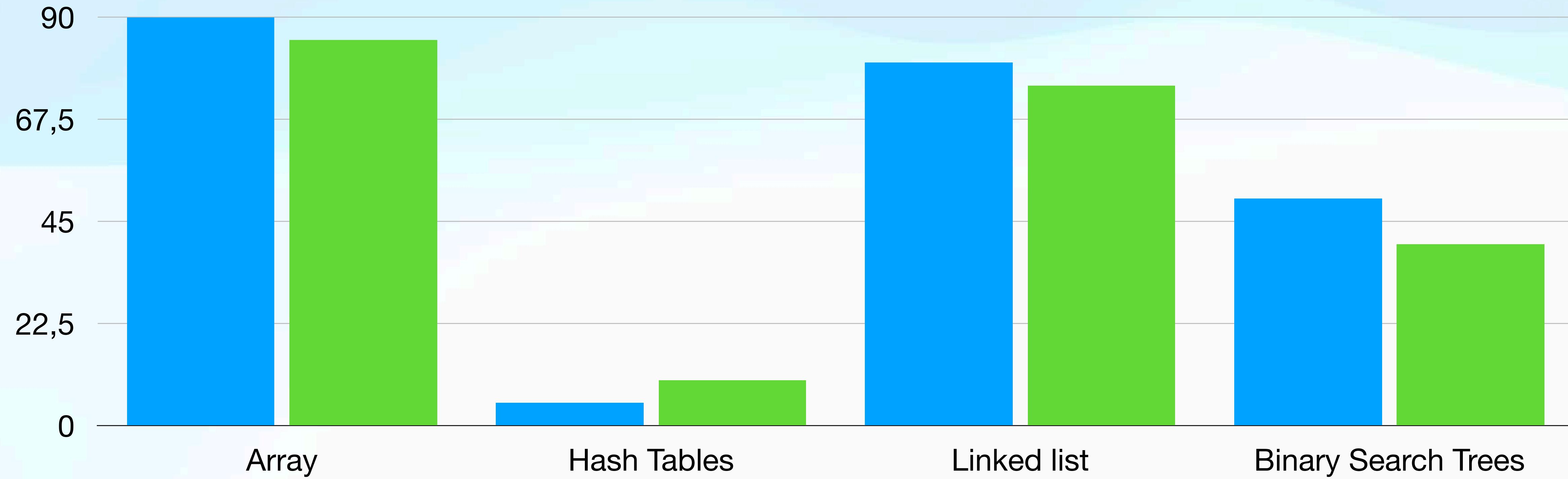
Khi người dùng nhập vào tên sản phẩm

→ Input: iPhone 15 Promax

## \* Đánh giá và đề xuất sản phẩm tương tự

### Tốc độ thực thi

- Thời gian chạy
- Số lần truy xuất



# Hash Table

# HASHING ?

Băm

# HASH FUNCTION ?

Hàm Băm



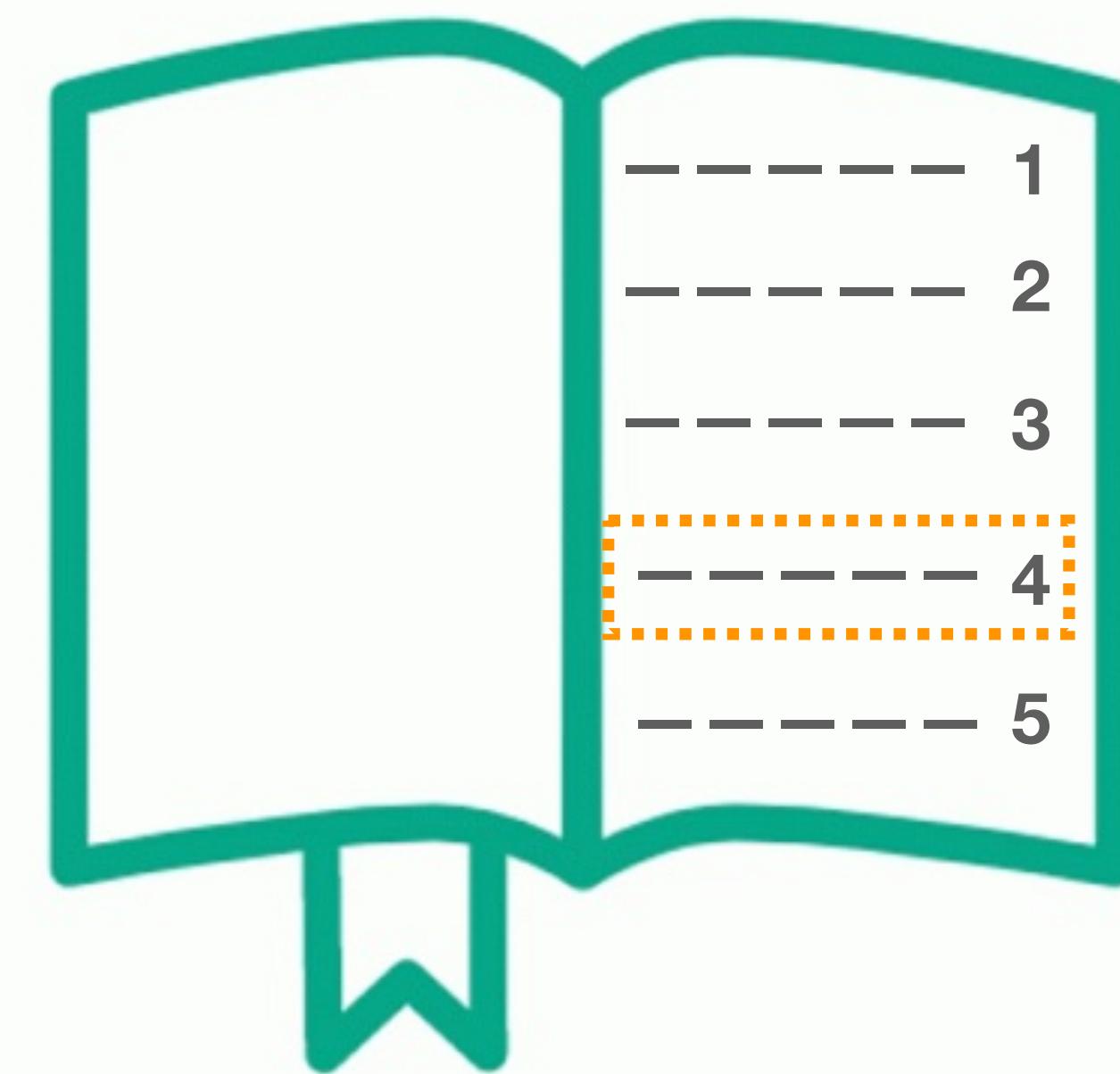
# COLLISION ?

Đụng độ

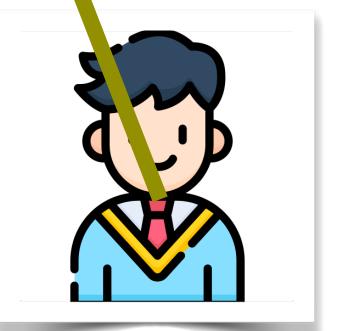
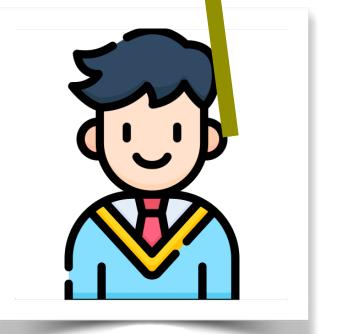
# ❖Quản lý dữ liệu



## ❖ Quản lý dữ liệu



**{ Key : Value }**



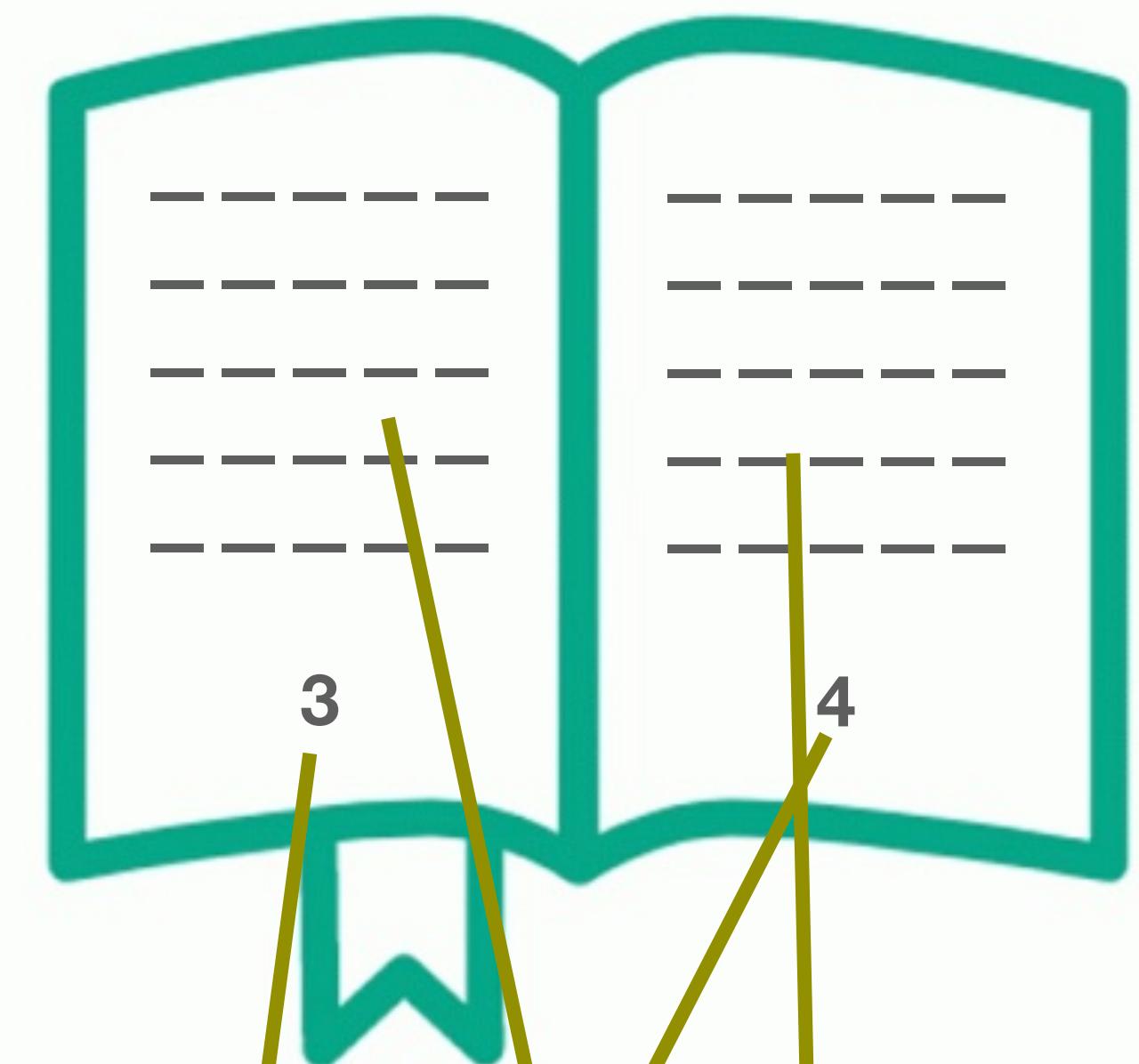
1

2

**{ Key : Value }**

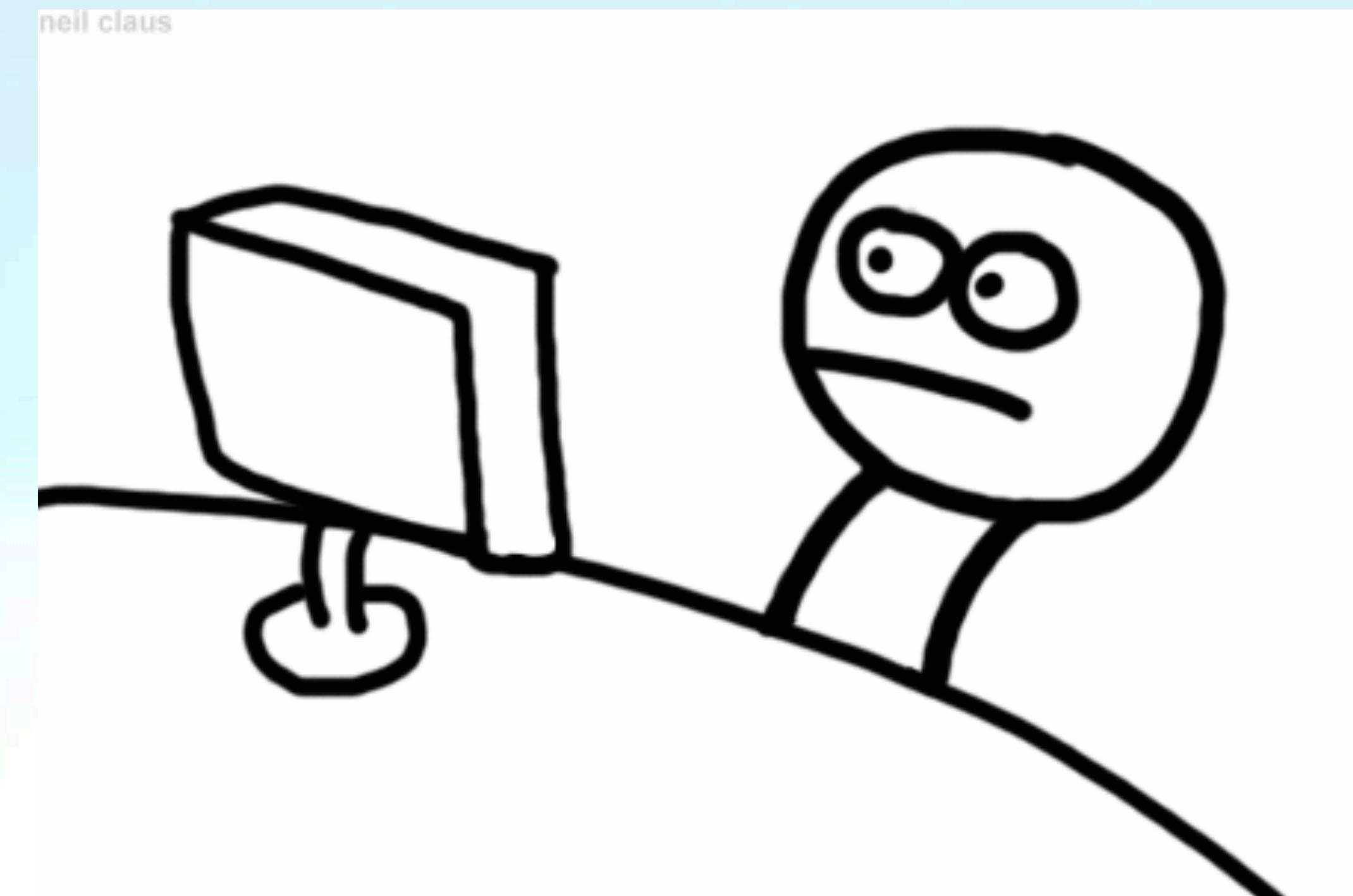
3

4





# Tại sao lại là Hash Table ??



## Tuyến tính (Linear search)



## Nhi phân (Binary search)



Key = 2

Key = 21

Key = 62

3

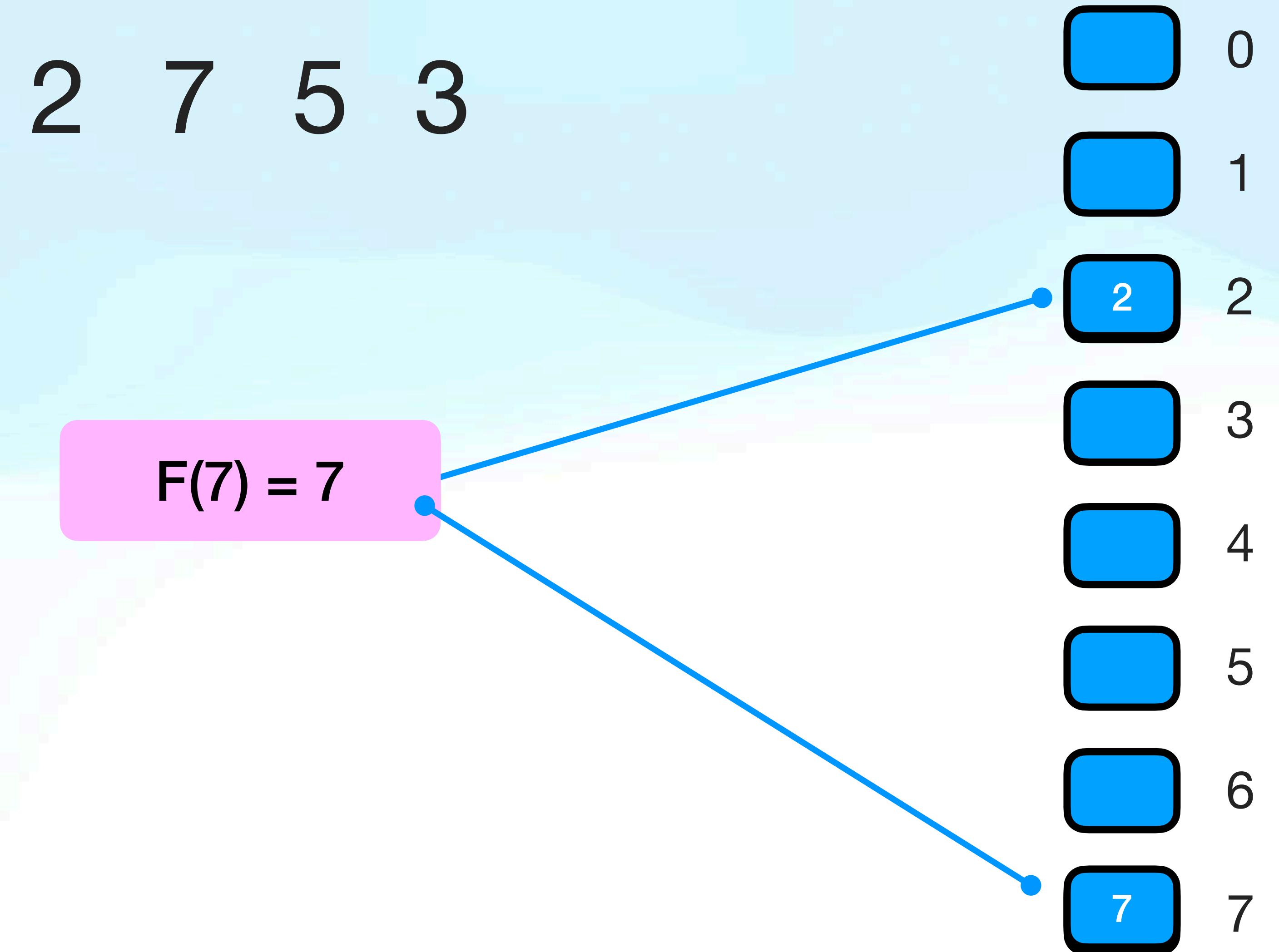
TH: Xấu nhất  $O(N) \Rightarrow O(7)$ TH: Xấu nhất  $O(\log_2 N) \Rightarrow O(3)$

# Tại sao lại là Hash Table ??

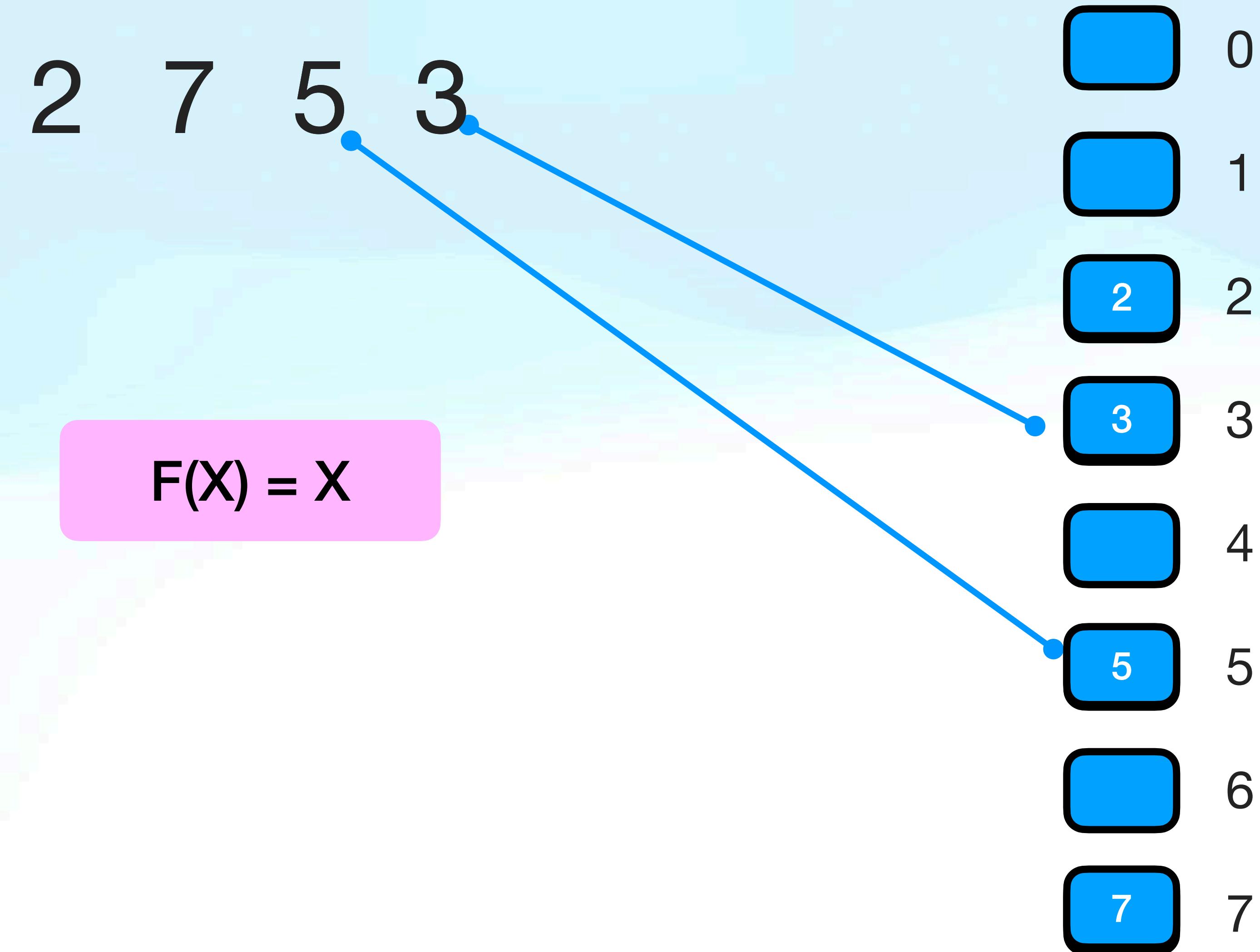


2 7 5 3

## ❖ Hash function



## ❖ Hash function



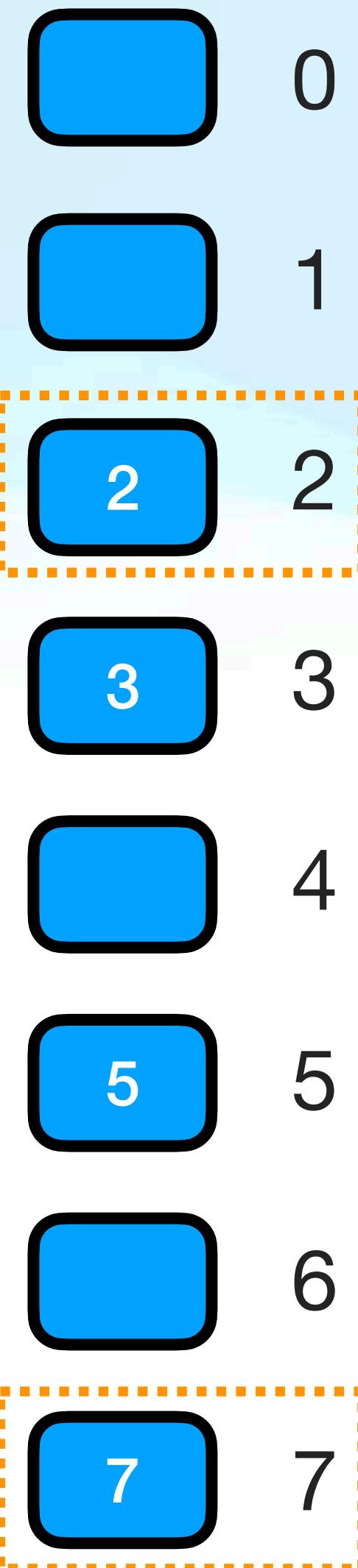
## ❖ Hash function

2 7 5 3

$$F(X) = X$$

Key = 2

Key = 7

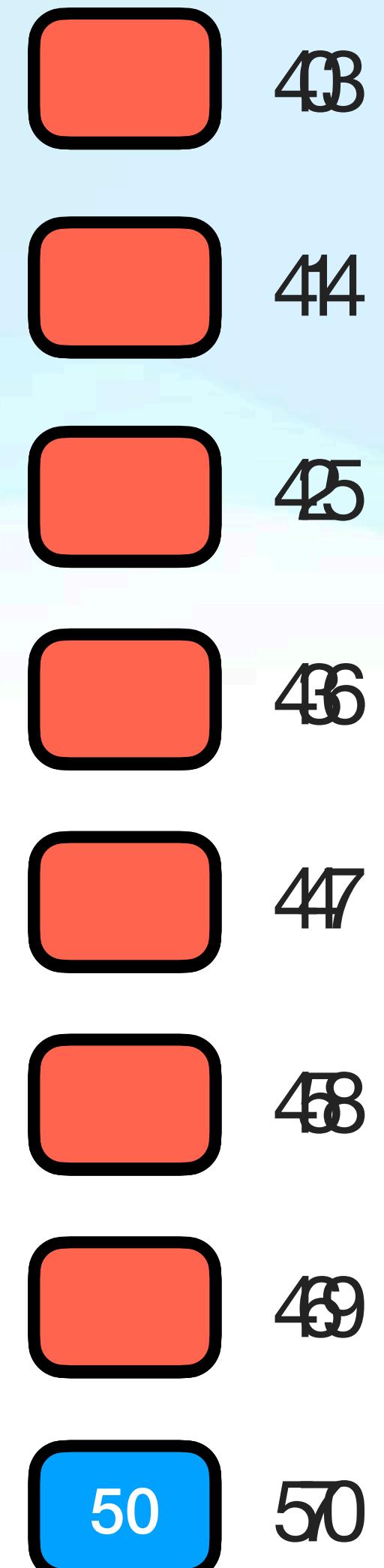


## ❖ Hash function

2 7 5 3 50

Hảo tồn tài nguyên bộ nhớ

$$F(50) = 50$$

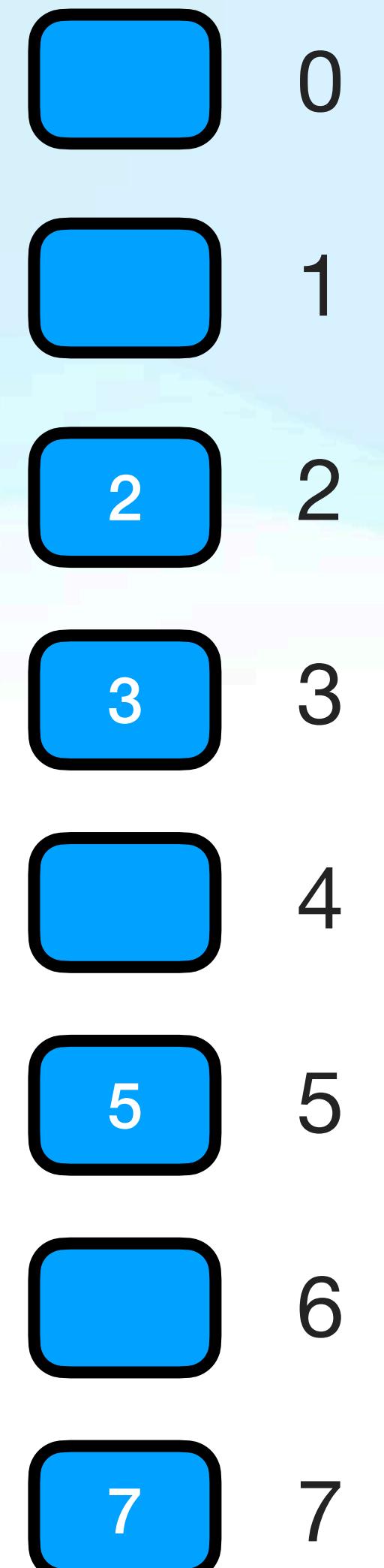


## ❖ Hash function

2 7 5 3

$$F(X) = X$$

$$H(X) = X$$



## ❖ Hash function

Giá trị phần tử

Số lượng phần tử

$$F(X) = M \% N$$

$$M \% N = 0, 1, 2, 3, \dots (N - 1)$$

## ❖ Hash function

2 7 5 3

N = 4

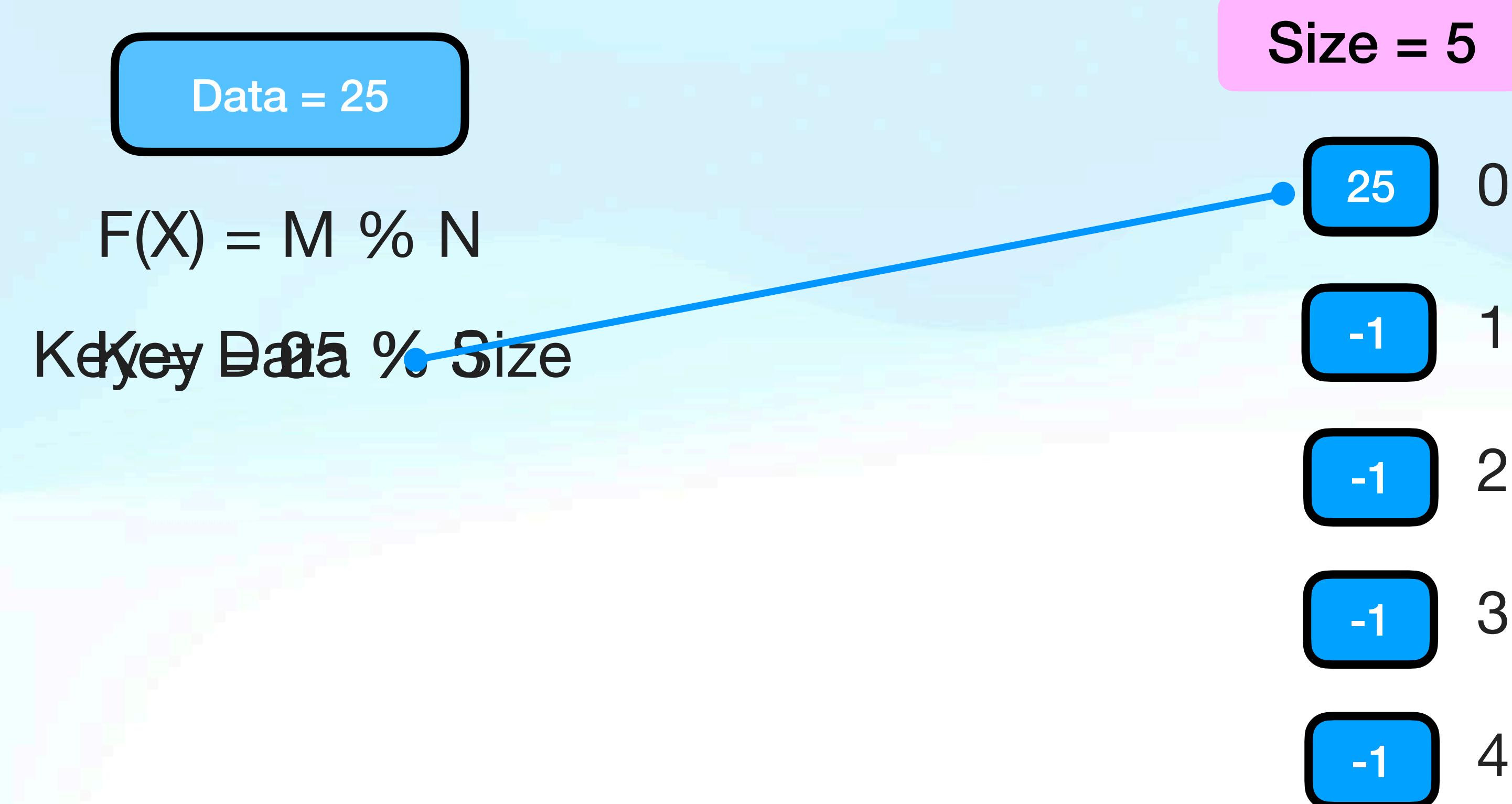
$$\begin{array}{rcl} 2 \% 4 &=& 2 \\ 7 \% 4 &=& 3 \\ 5 \% 4 &=& 1 \\ 3 \% 4 &=& 3 \\ 8 \% 4 &=& 0 \end{array}$$

## ❖ Hash function

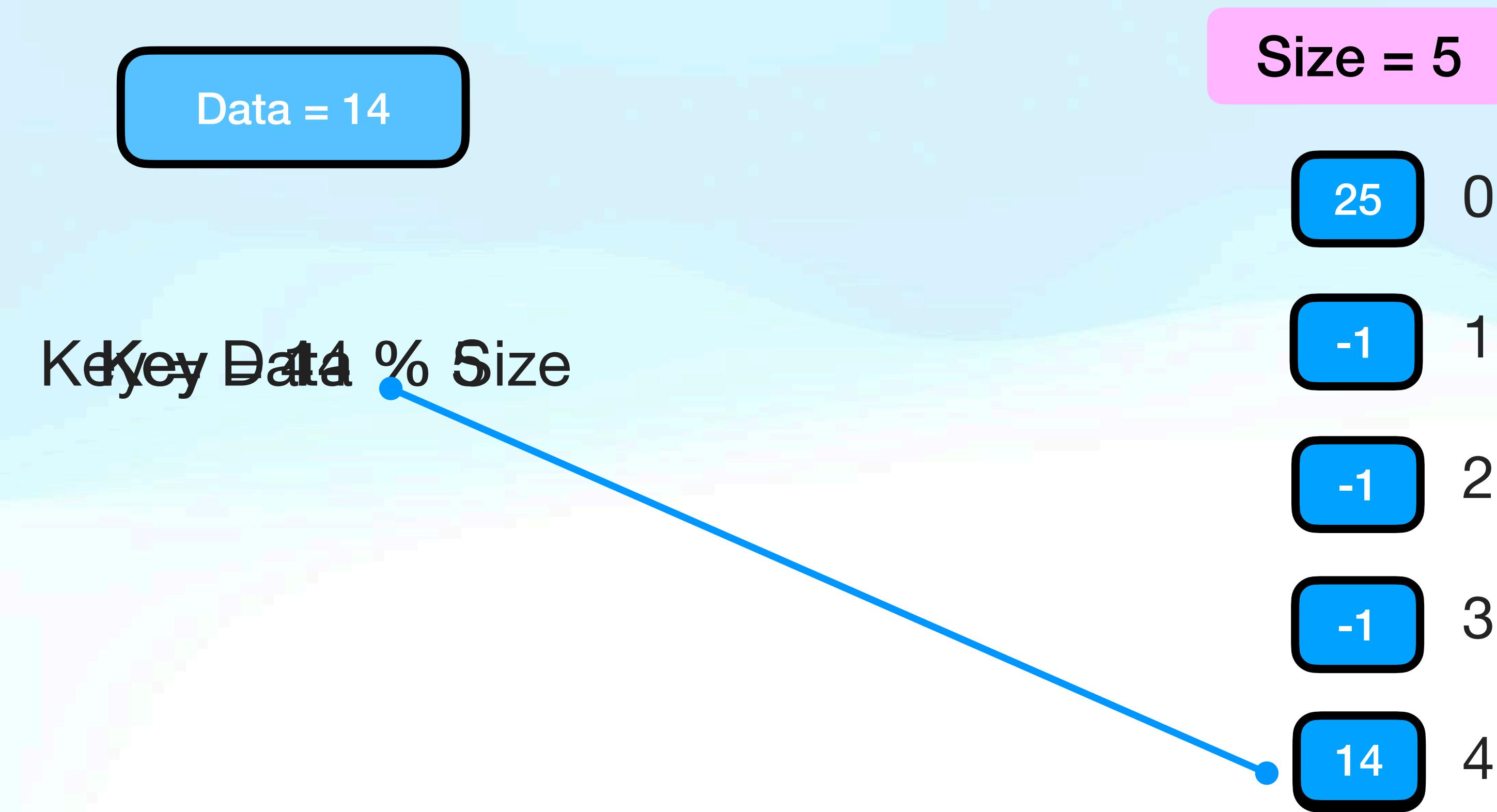
Size = 5

|    |   |
|----|---|
| -1 | 0 |
| -1 | 1 |
| -1 | 2 |
| -1 | 3 |
| -1 | 4 |

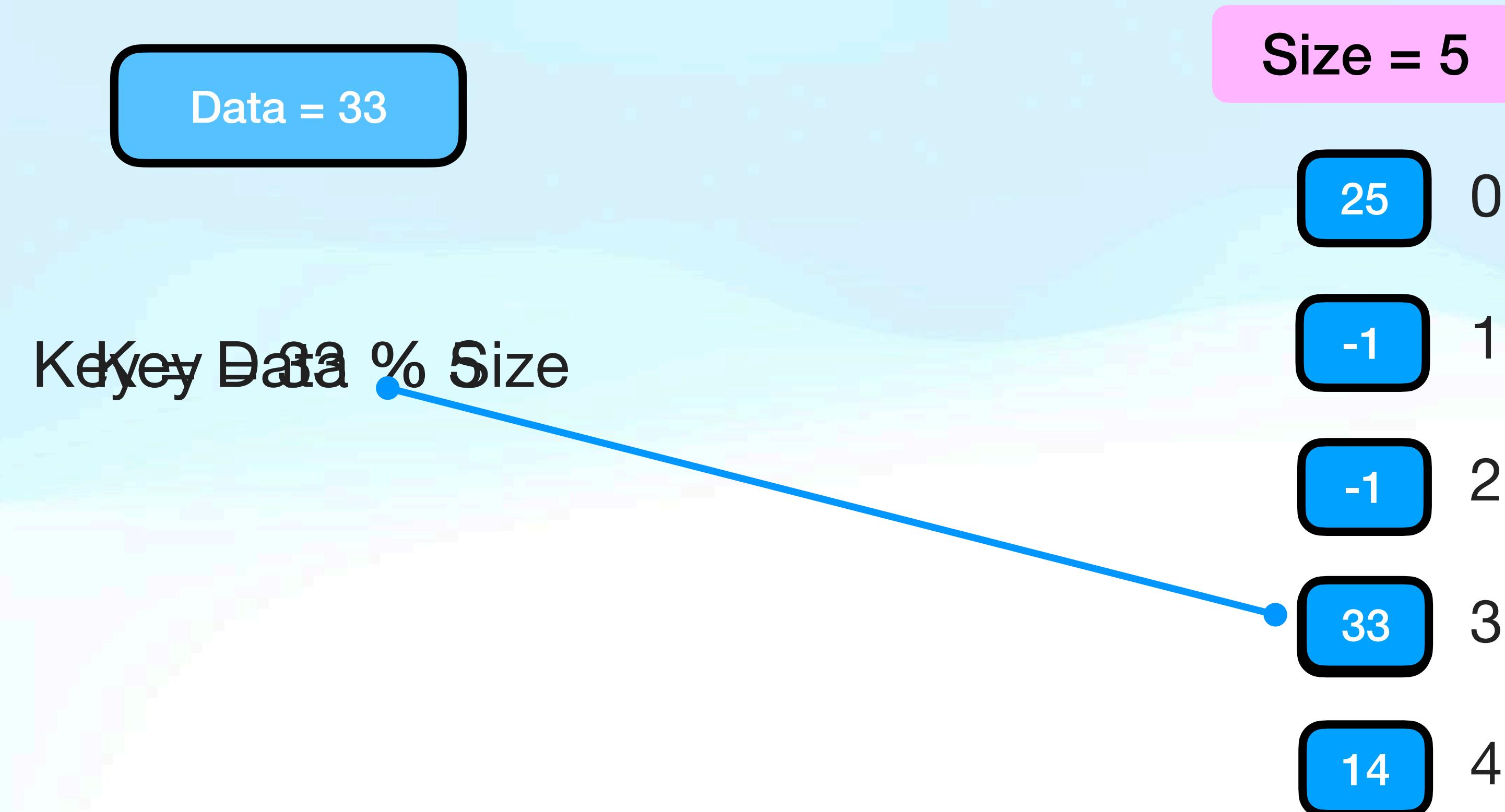
## ❖ Hash function



## ❖ Hash function



## ❖ Hash function

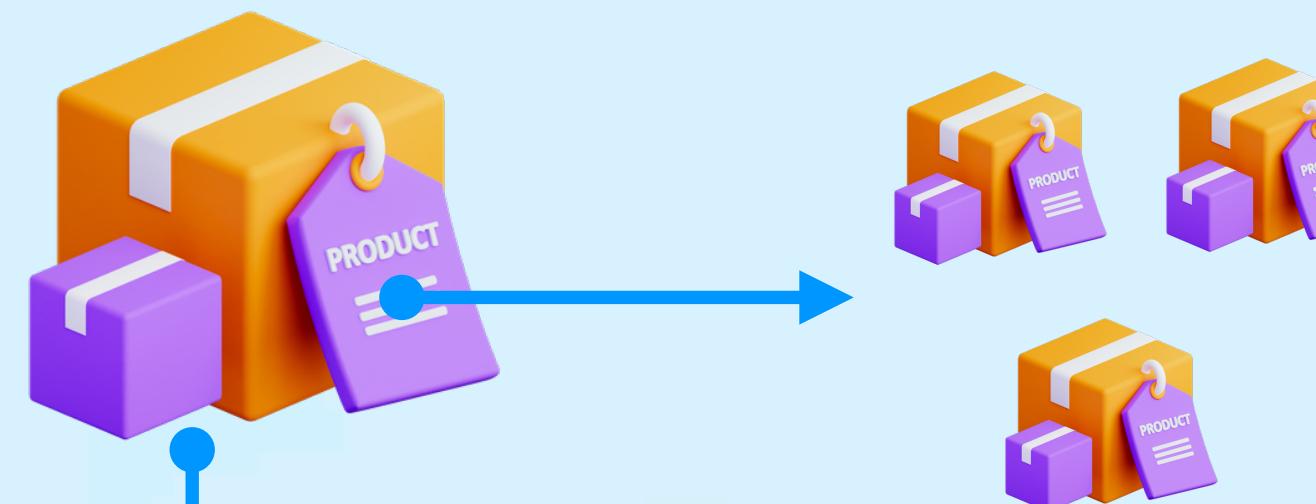


## \* Đánh giá và đề xuất sản phẩm tương tự

Tên sản phẩm

Danh mục

Đơn giá



Hệ thống trả danh sách sản phẩm tương tự.

→ Output: Galaxy Z Fold, Nokia lumia, Oppo Neo, ...

▶ Khởi tạo một danh sách **Product** chứa các đối tượng sản phẩm.

▶ Xây dựng hash table:

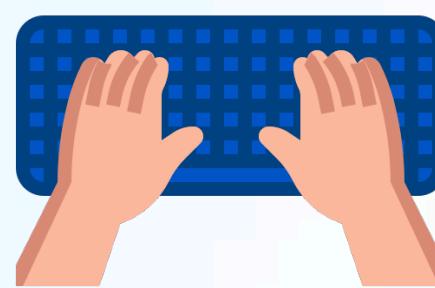
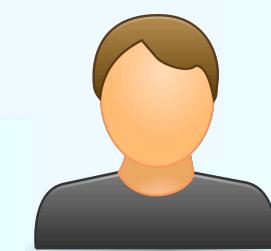
→ Khởi tạo hash function

→ Khởi tạo hàm thêm theo hash function

→ Khởi tạo hàm tìm kiếm theo bài toán

Khi người dùng nhập vào tên sản phẩm

→ Input: iPhone 15 Promax



- Khởi tạo một danh sách **Product** chứa các đối tượng sản phẩm.

Tên sản phẩm

Danh mục

Đơn giá



**class Product:**

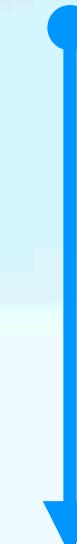
```
def __init__(self, name, price, category):  
    self.name = name  
    self.price = price  
    self.category = category
```

## ► Xây dựng hash table:

→ Khởi tạo hash function

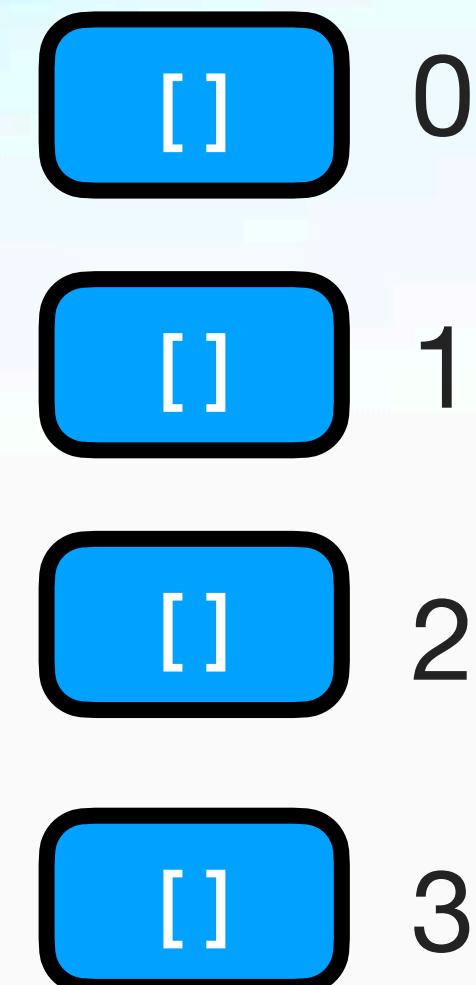
```
class HashTable:  
    def __init__(self, size):  
        self.size = size  
        self.table = [ [] for _ in range(self.size) ]
```

```
def hash_function(self, key):  
    return sum( ord(char) for char in key ) % self.size
```



```
ord("a") => 97
```

```
hash_table = HashTable(4)
```



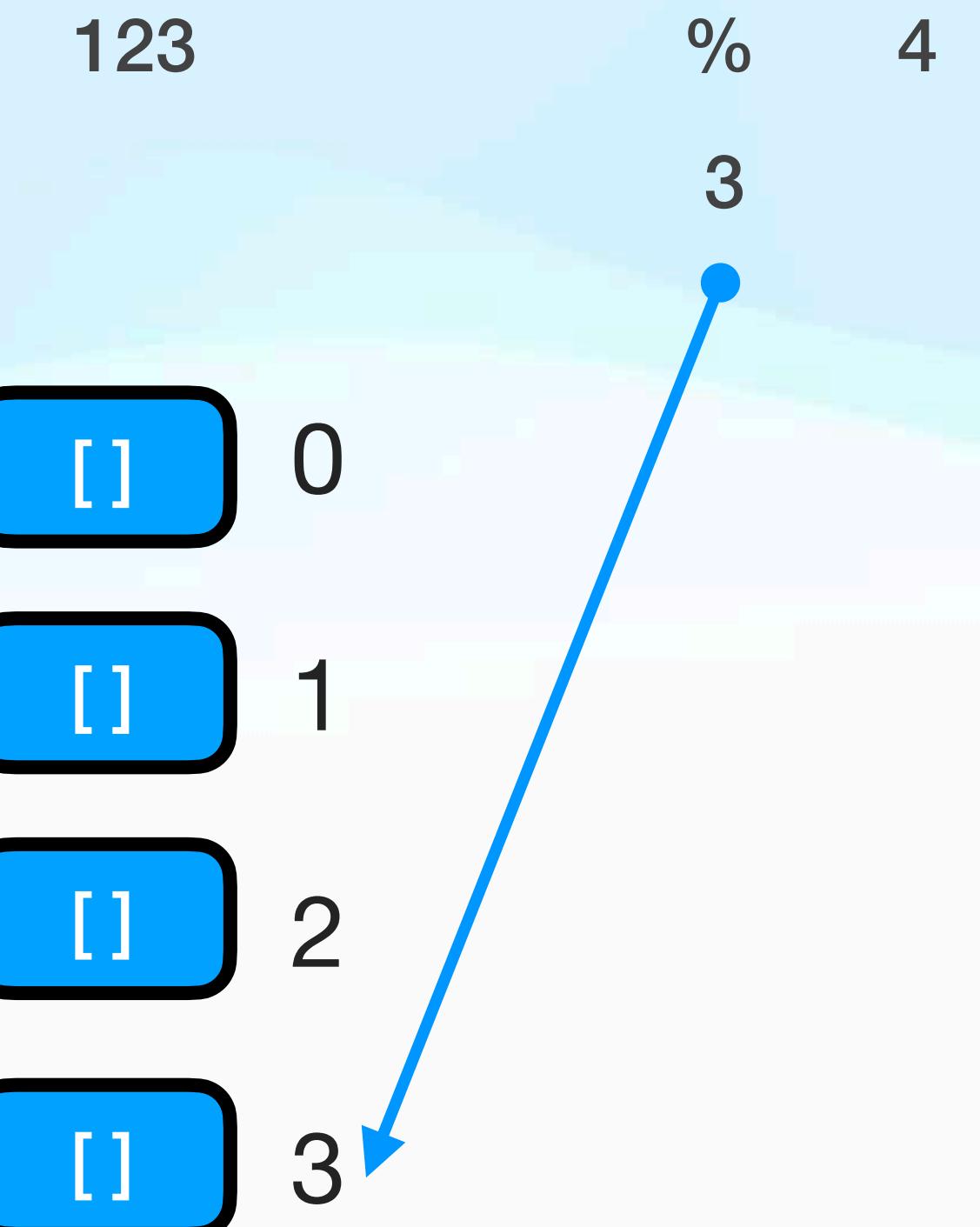
## → Khởi tạo hàm thêm theo hash function

```
def insert(self, product):
    index = self.hash_function(product.category)
    self.table[index].append(product)
```

=> Input: Product("iPhone 15", 199, "Phone") => Category: Phone

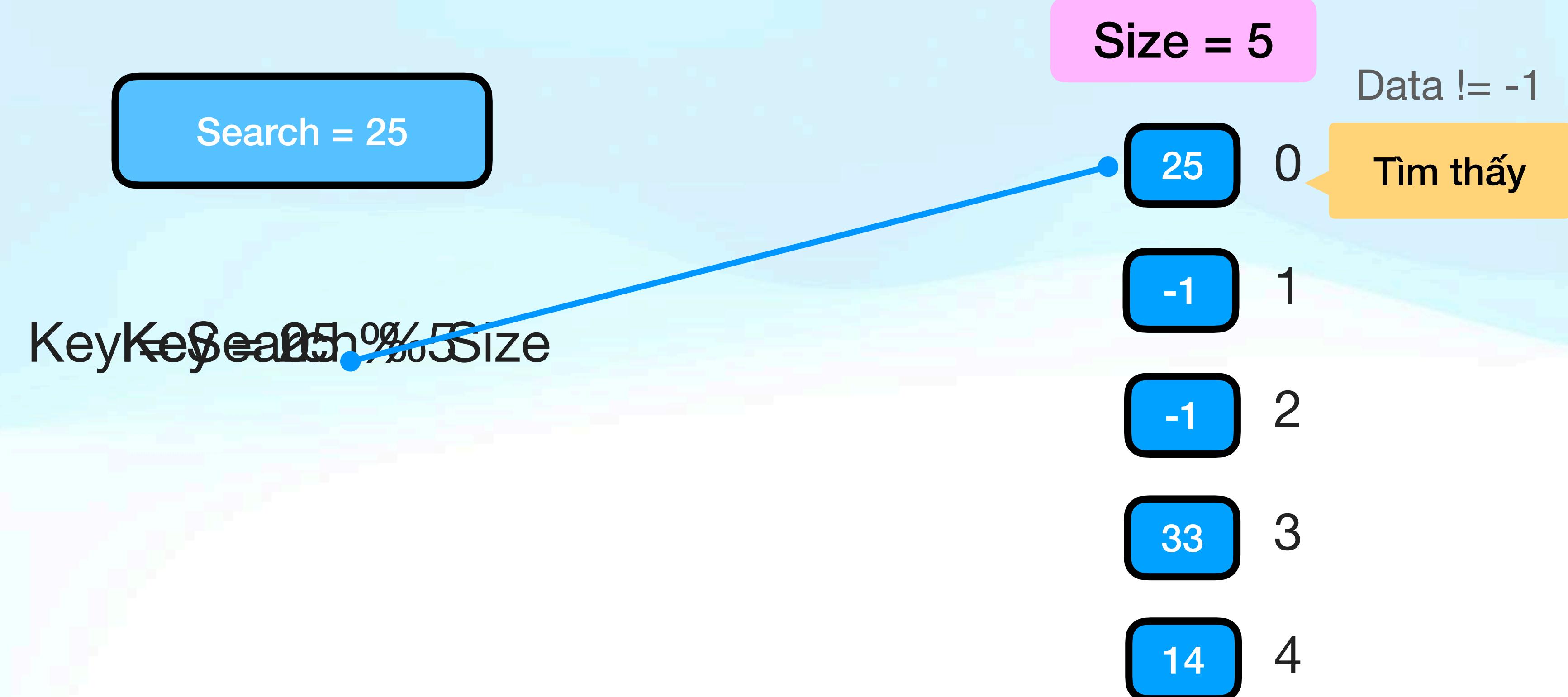
[ Prod ]

```
def hash_function(self, key):
    return sum( ord(char) for char in key ) % self.size
```

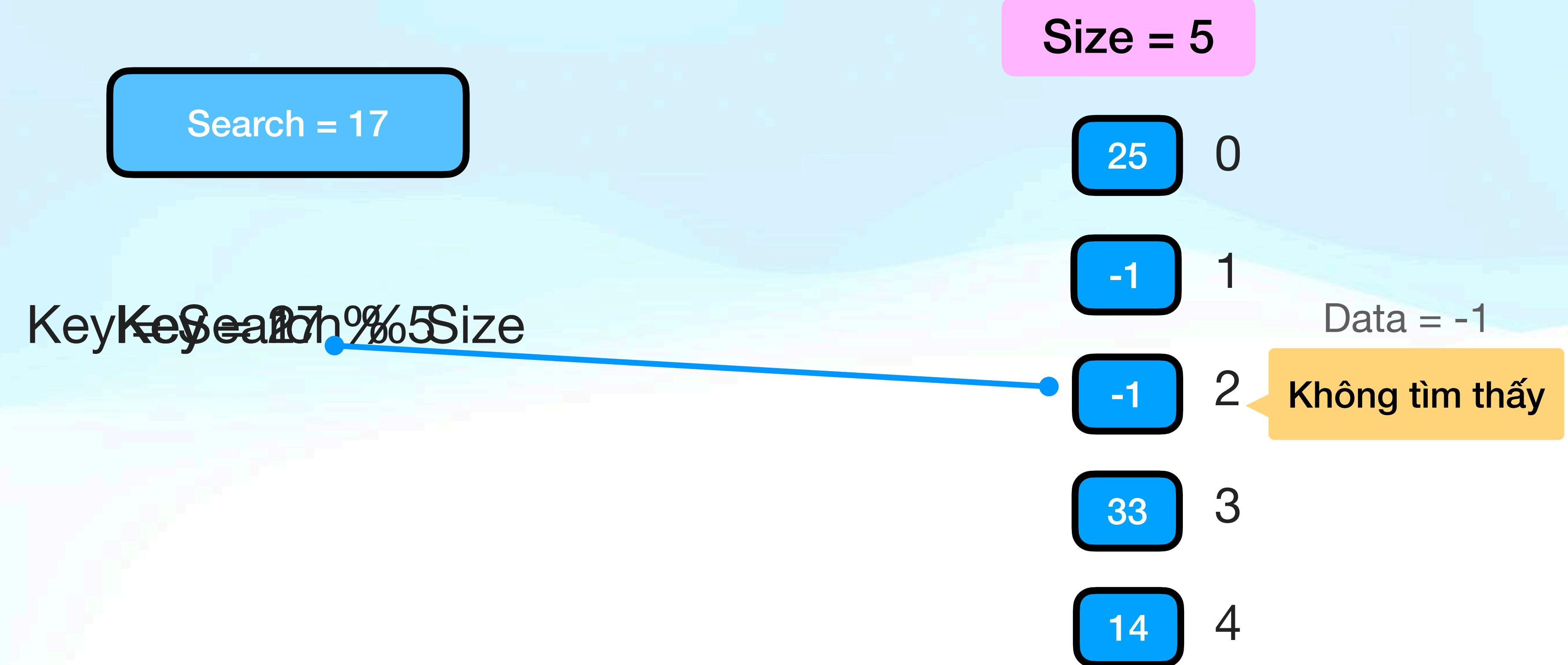


Tìm kiếm phần tử

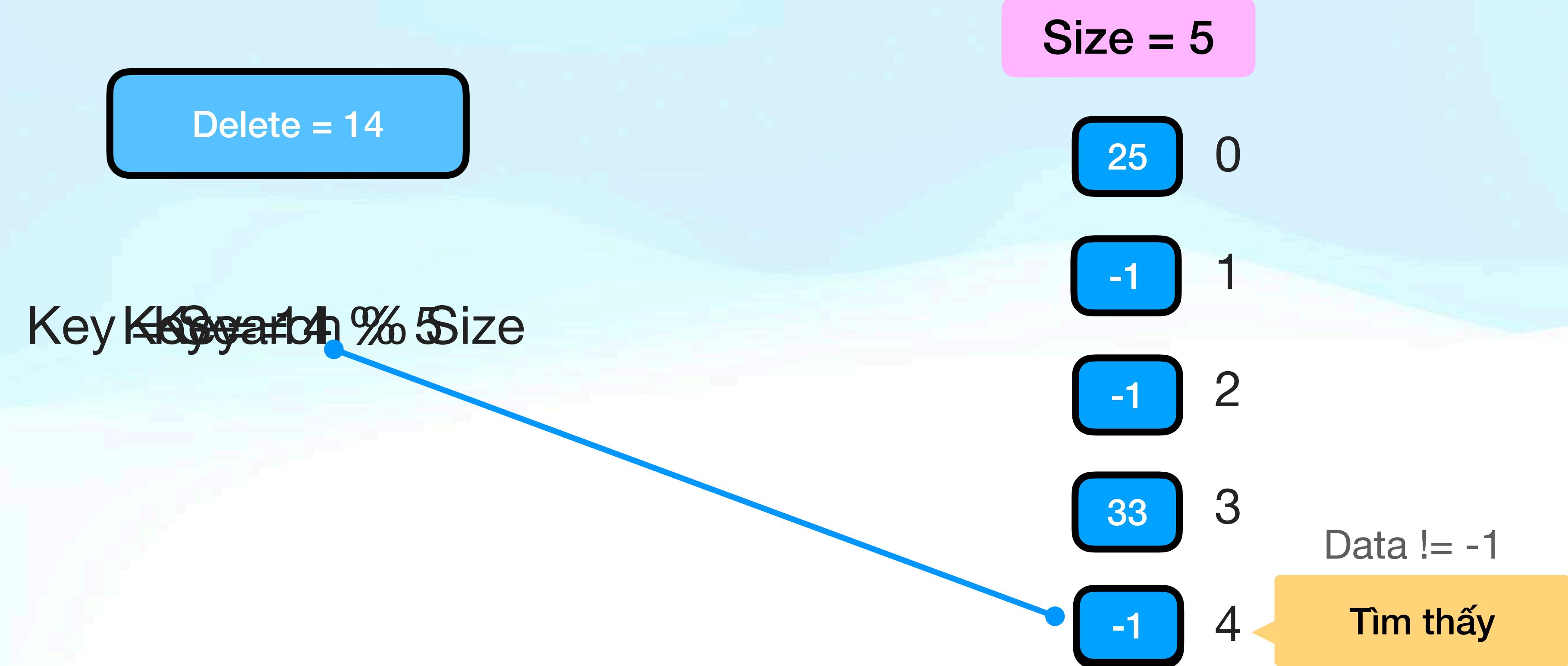
## ❖ Hash function



## ❖ Hash function



## ❖ Hash function

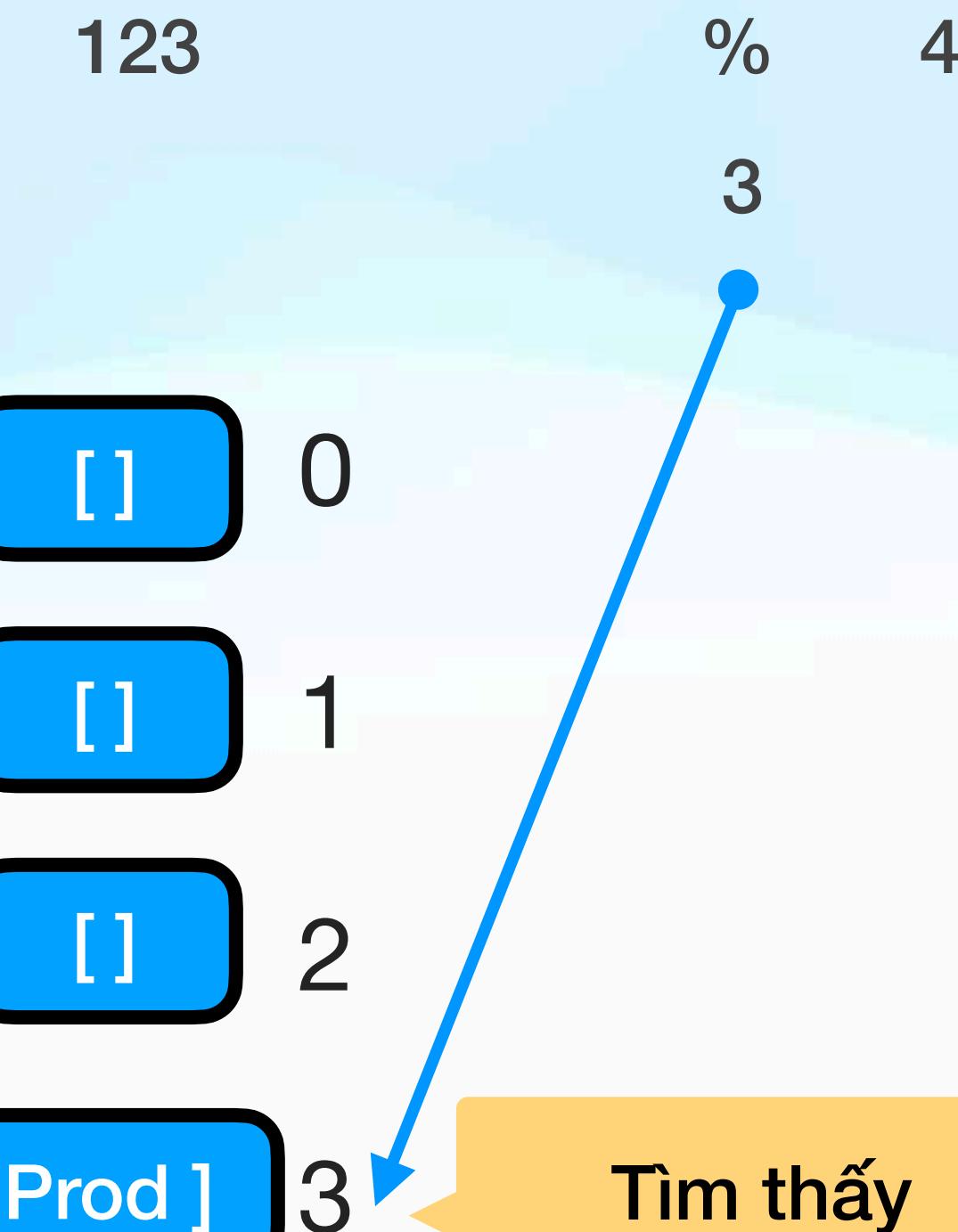


## → Khởi tạo tìm kiếm theo hash function

```
def search(self, name):  
    index = self.hash_function(name)  
    return self.table[index]
```

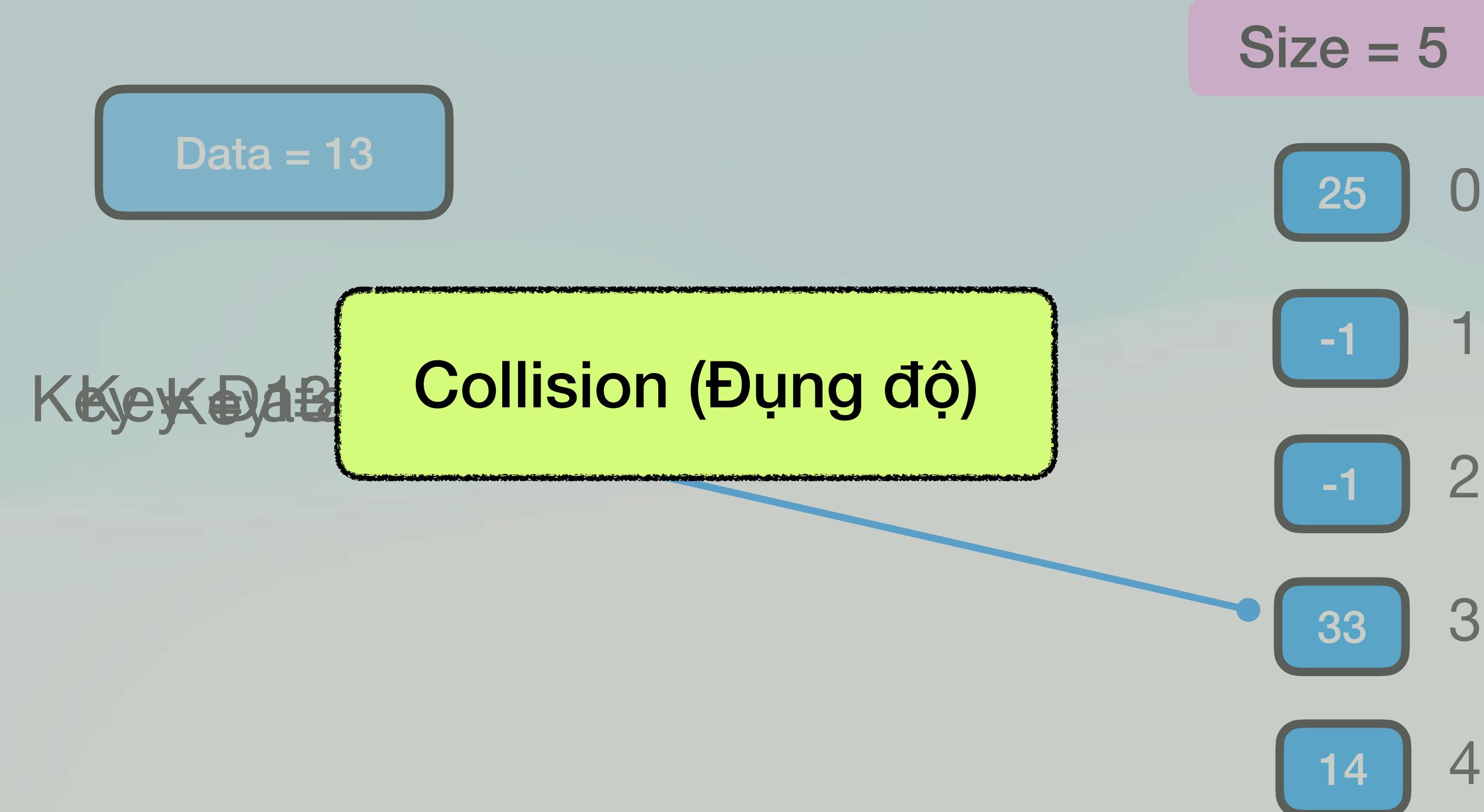
=> Input: Product("iPhone 15", 199, "Phone") => Category: Phone

```
def hash_function(self, key):  
    return sum( ord(char) for char in key ) % self.size
```



# Collision Đụng độ





Open addressing (địa chỉ mở)

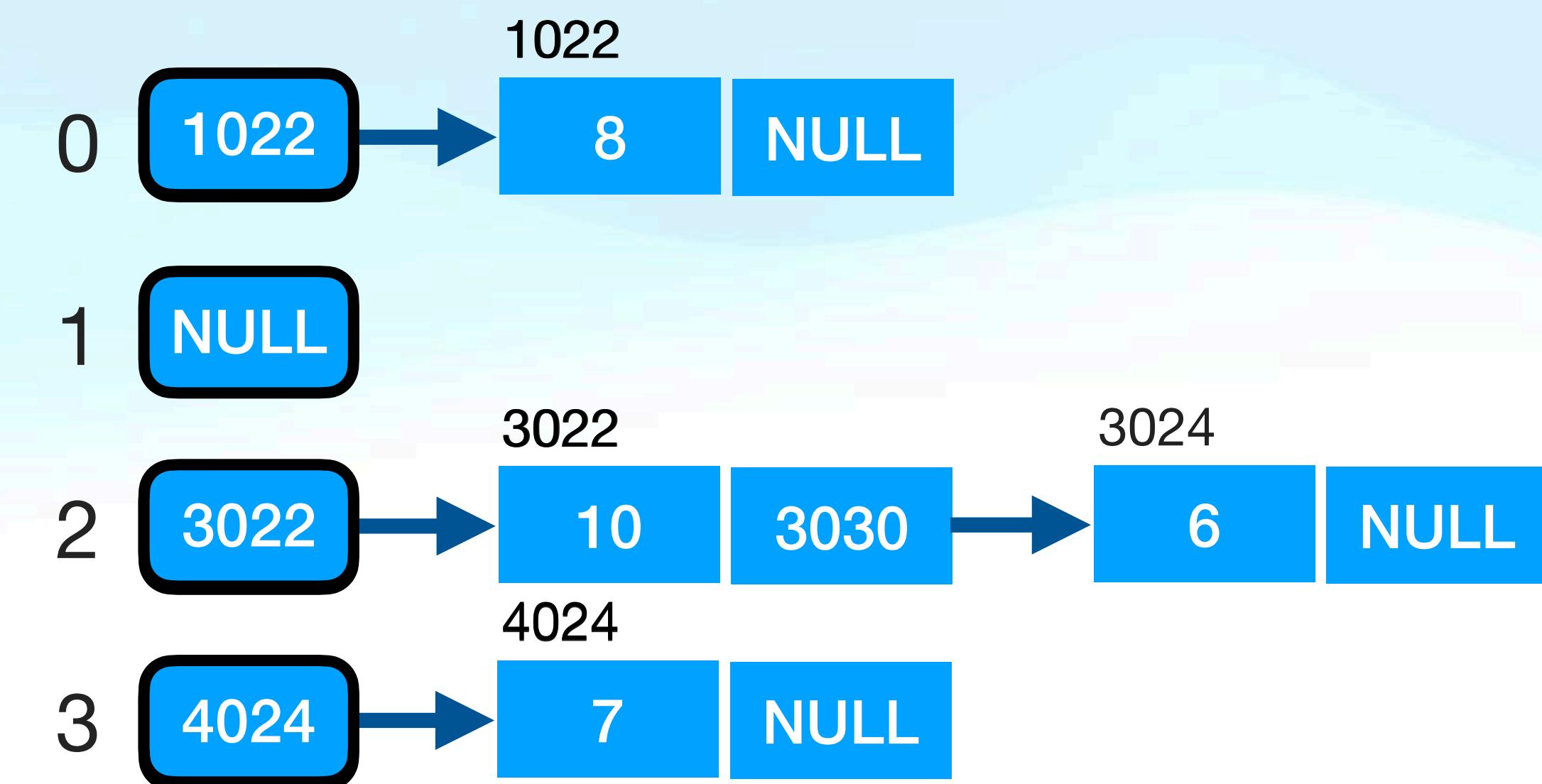
- Linear Probing (Thăm dò tuyến tính)
- Quadratic Probing (Thăm dò bậc hai)

Separate chaining  
(xâu chuỗi)

# Separate Chaining

## Separate Chaining

Linked List (danh sách liên kết)  
Array (Mảng)



## Separate Chaining

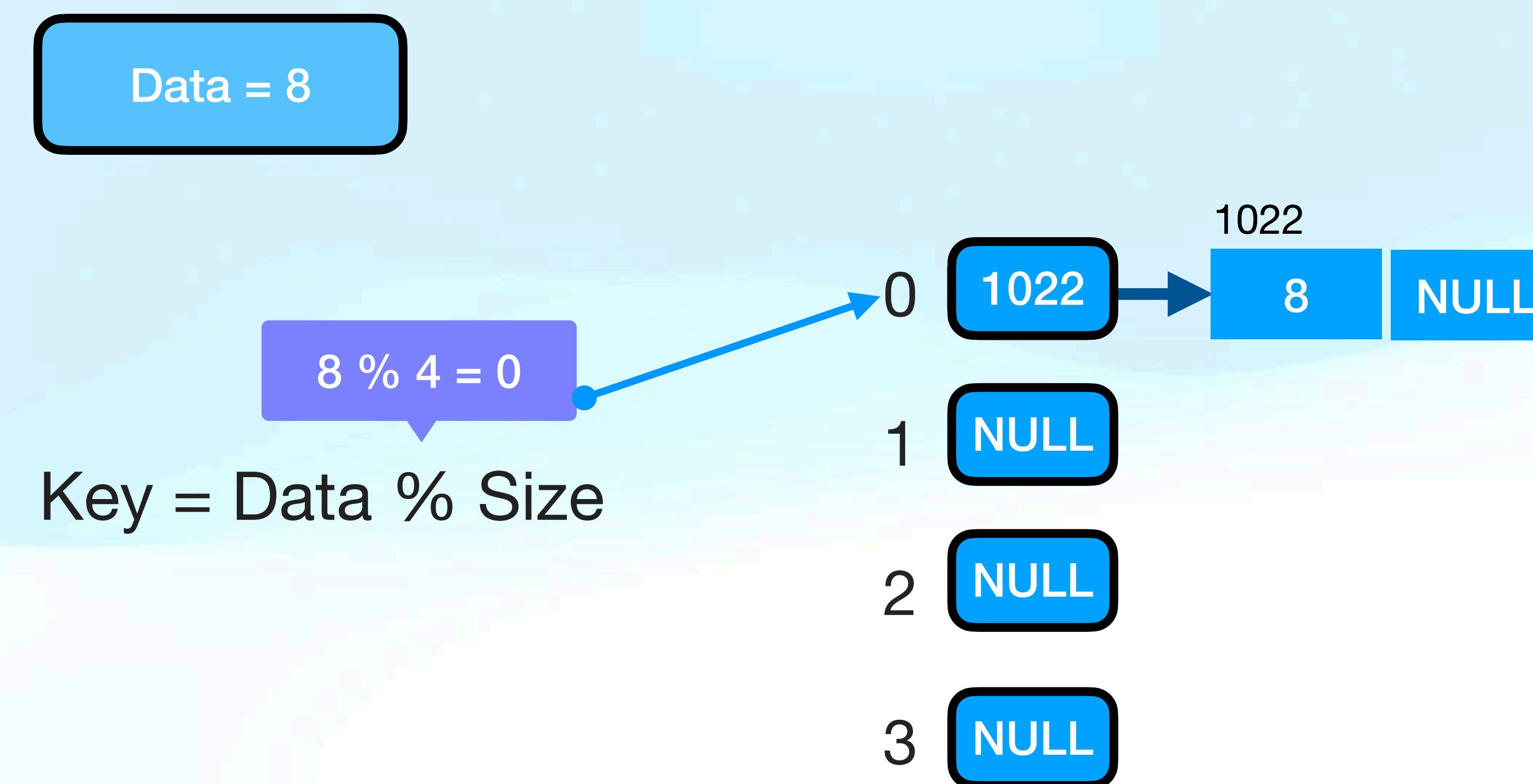
Data = 8

$$8 \% 4 = 0$$

Key = Data % Size



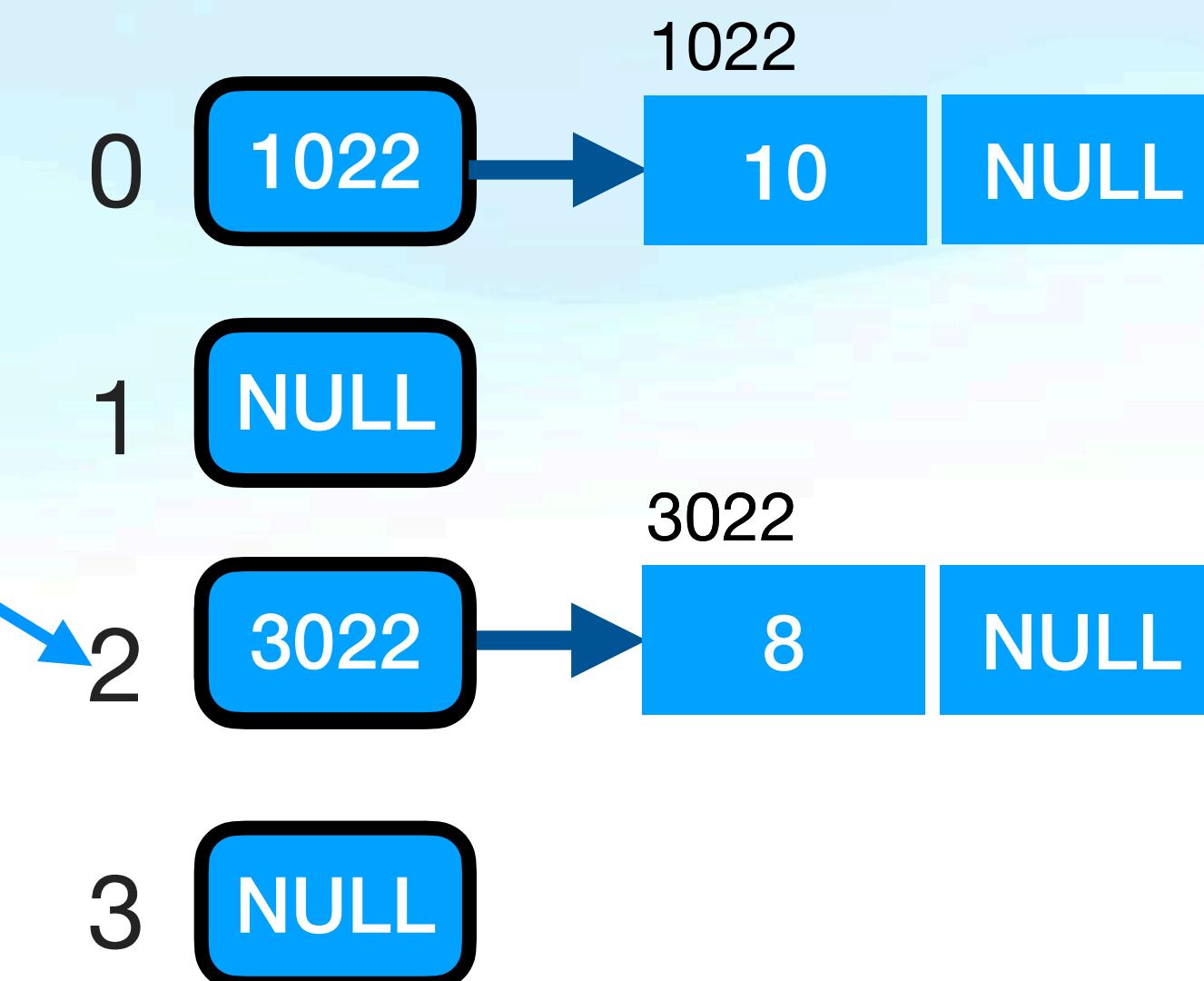
## Separate Chaining



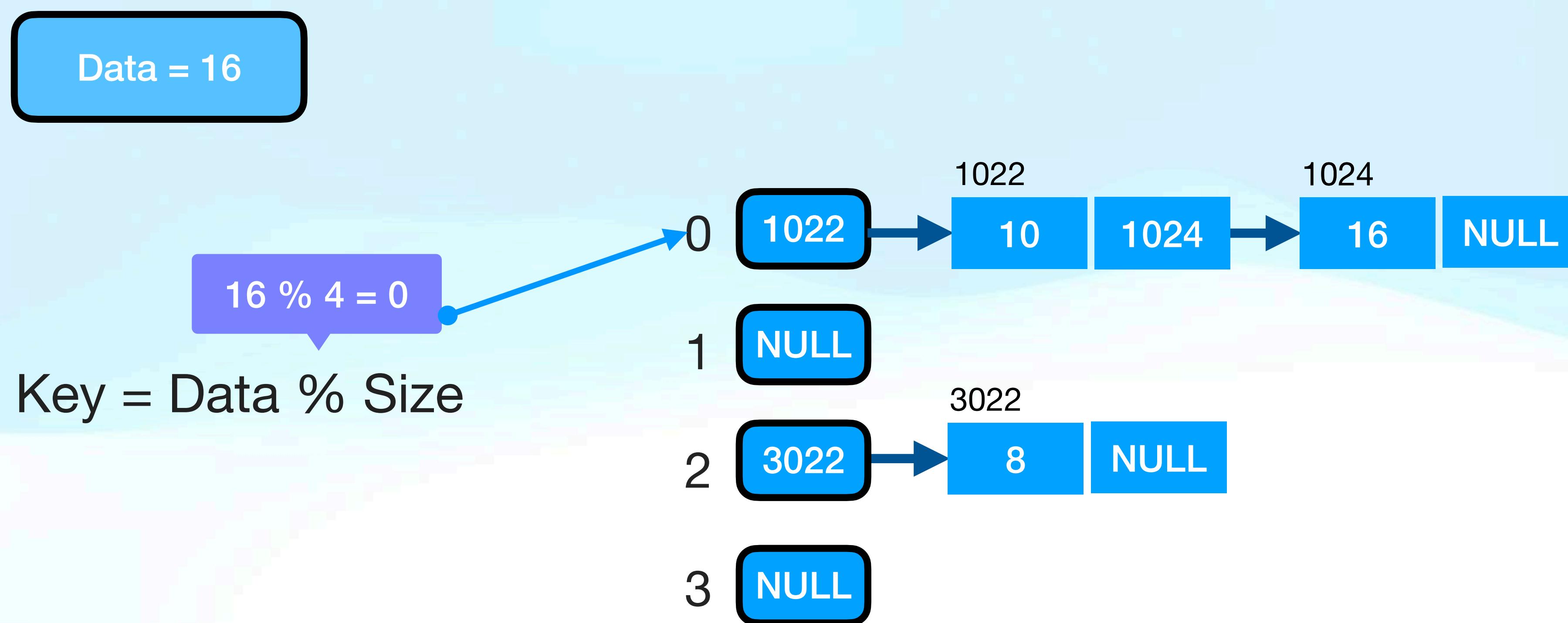
## Separate Chaining

Data = 10

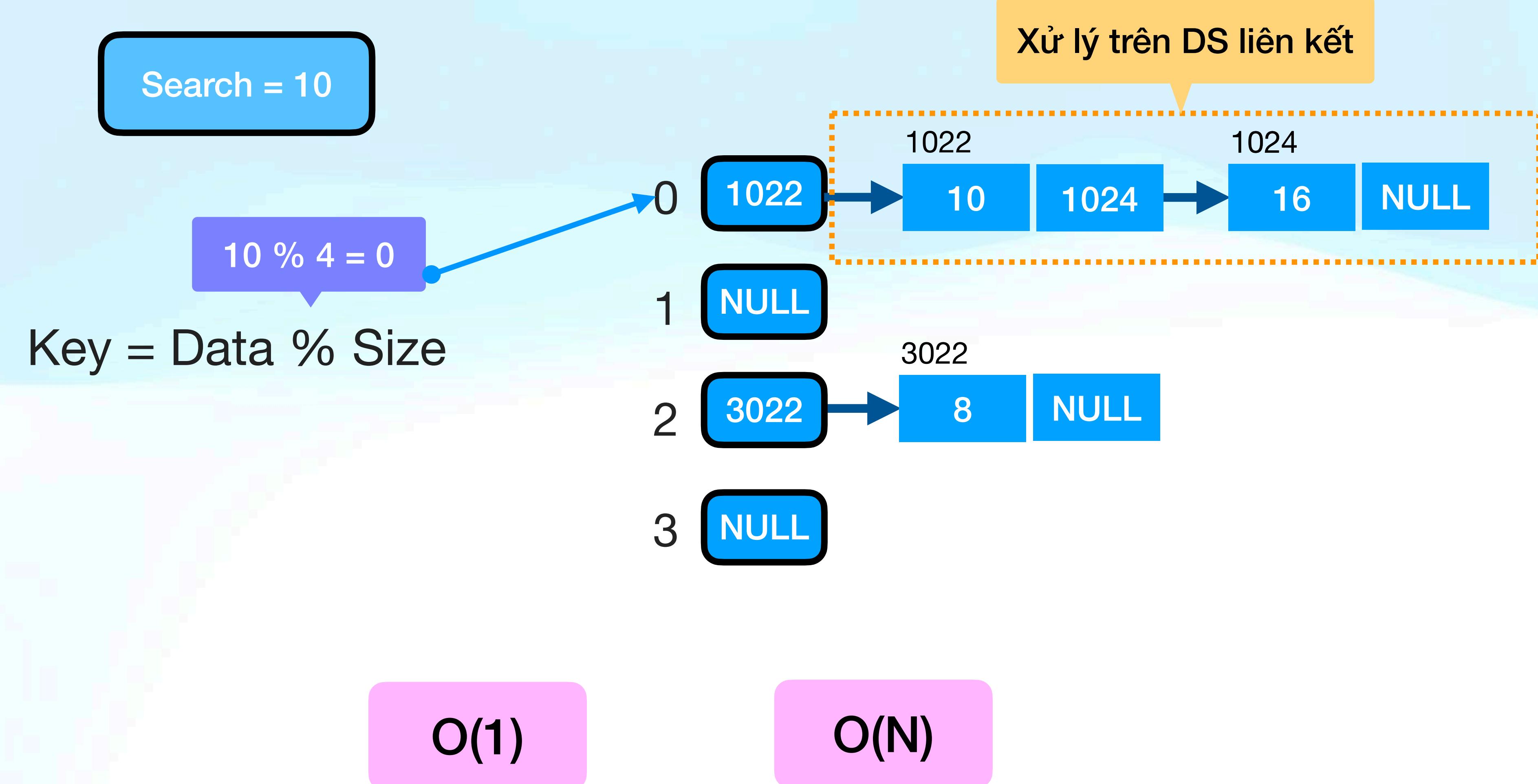
Key = Data % Size  
 $10 \% 4 = 2$



## Separate Chaining



# Separate Chaining



## ➡ Khởi tạo hàm thêm theo hash function

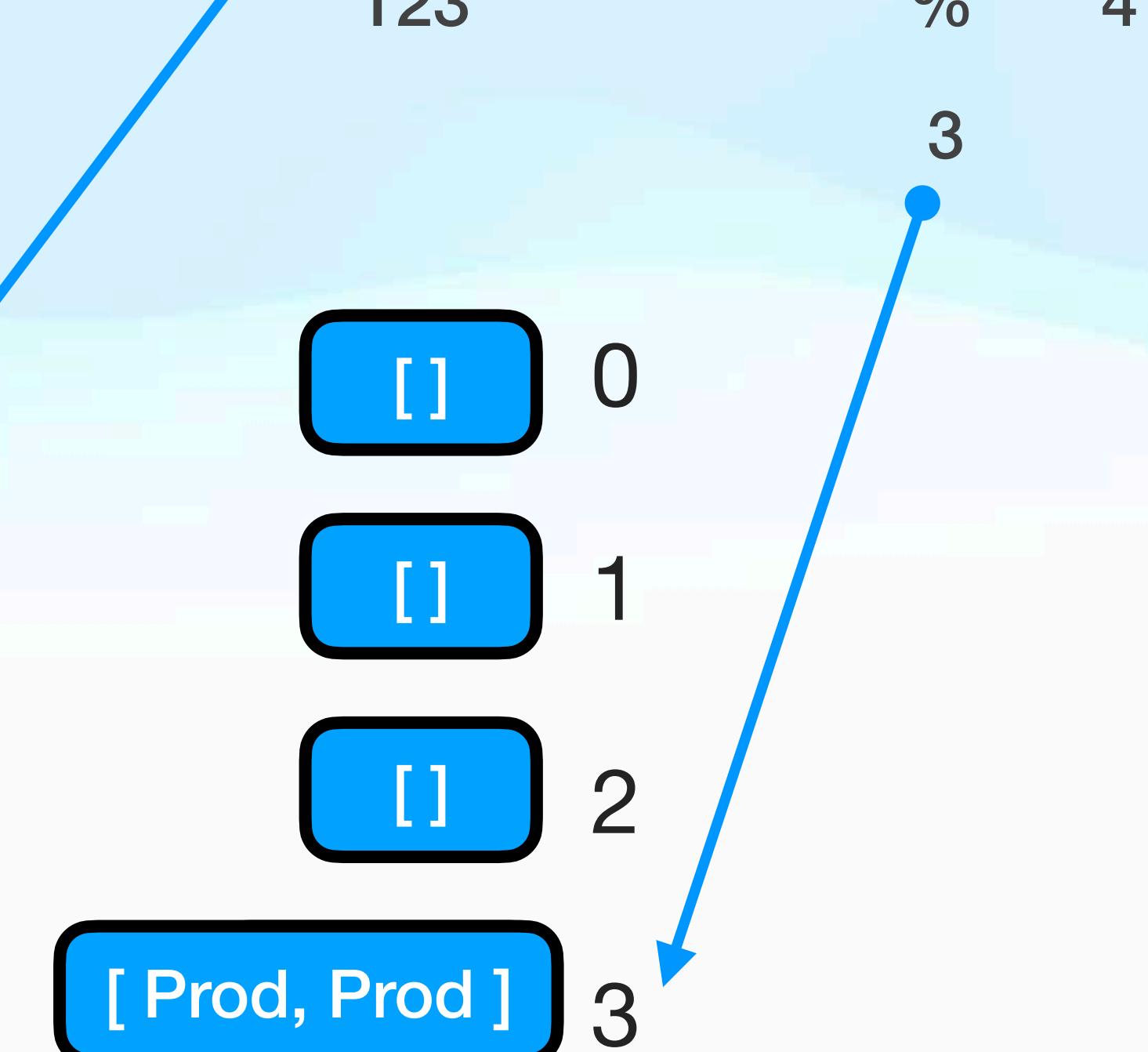
```
def insert(self, product):
    index = self.hash_function(product.category)
    self.table[index].append(product)
```

Prod

```
def hash_function(self, key):
    return sum( ord(char) for char in key ) % self.size
```

=> Input: Product("iPhone 15", 199, "Phone") => Category: Phone

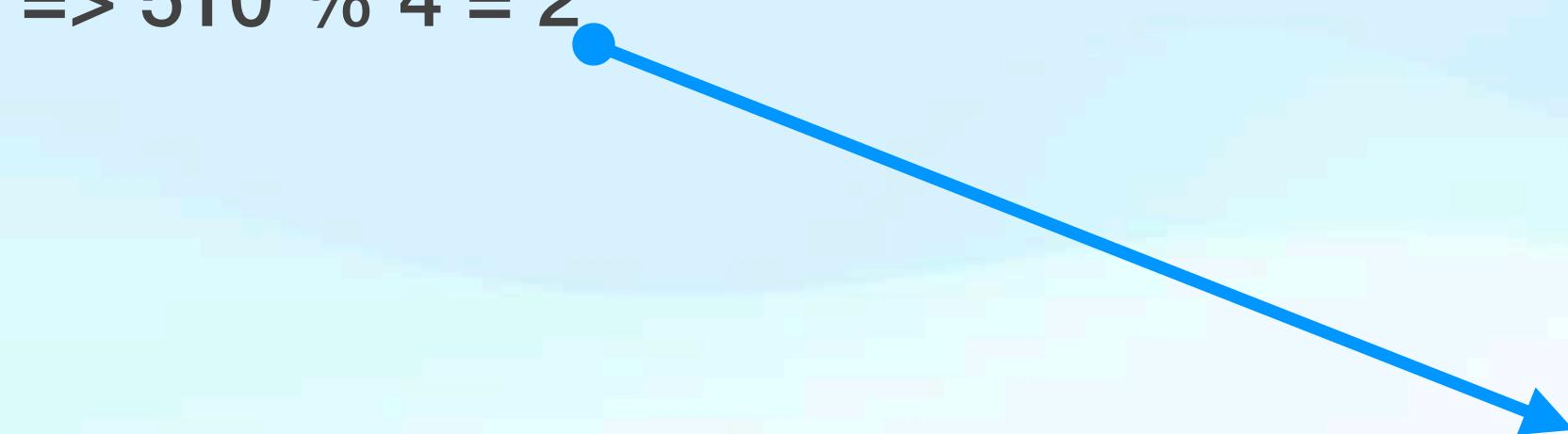
=> Input: Product("iPhone 14", 99, "Phone") => Category: Phone



# Implement

```
products = [  
    Product("Novel", 20, "Books"),  
    Product("History_Book", 25, "Books"),  
    Product("Apple", 1, "Fruits"),  
    Product("Doll", 25, "Toys")  
]  
hash_table = HashTable(len(products))  
for product in products:  
    hash_table.insert(product)
```

$$\Rightarrow 510 \% 4 = 2$$

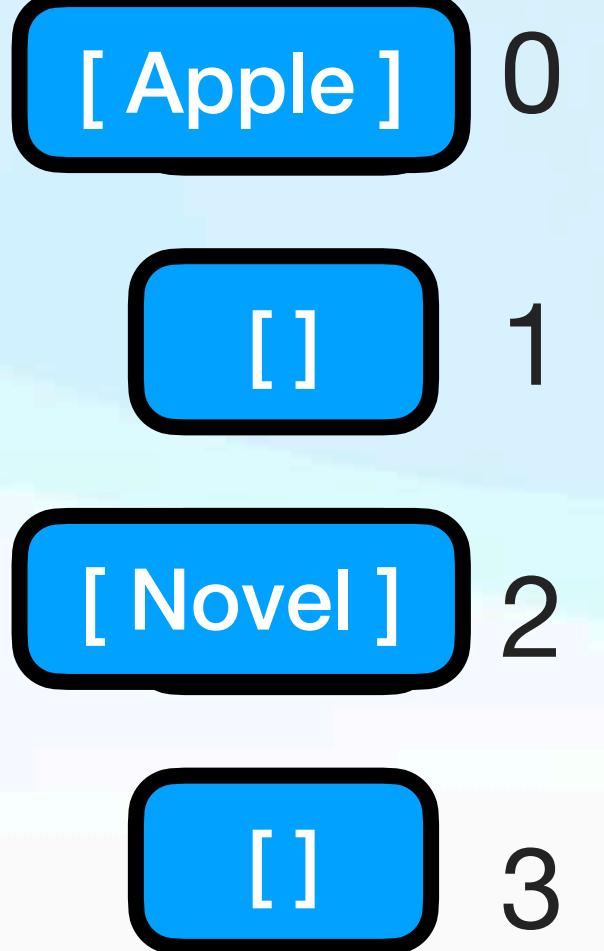


|           |   |
|-----------|---|
| []        | 0 |
| []        | 1 |
| [ Novel ] | 2 |
| []        | 3 |

```
products = [  
    Product("Novel", 20, "Books"),  
    Product("History_Book", 25, "Books"),  
    Product("Apple", 1, "Fruits"),  
    Product("Doll", 25, "Toys")  
]  
hash_table = HashTable(len(products))  
for product in products:  
    hash_table.insert(product)
```

$$\Rightarrow 510 \% 4 = 2$$

$$\Rightarrow 637 \% 4 = 1$$

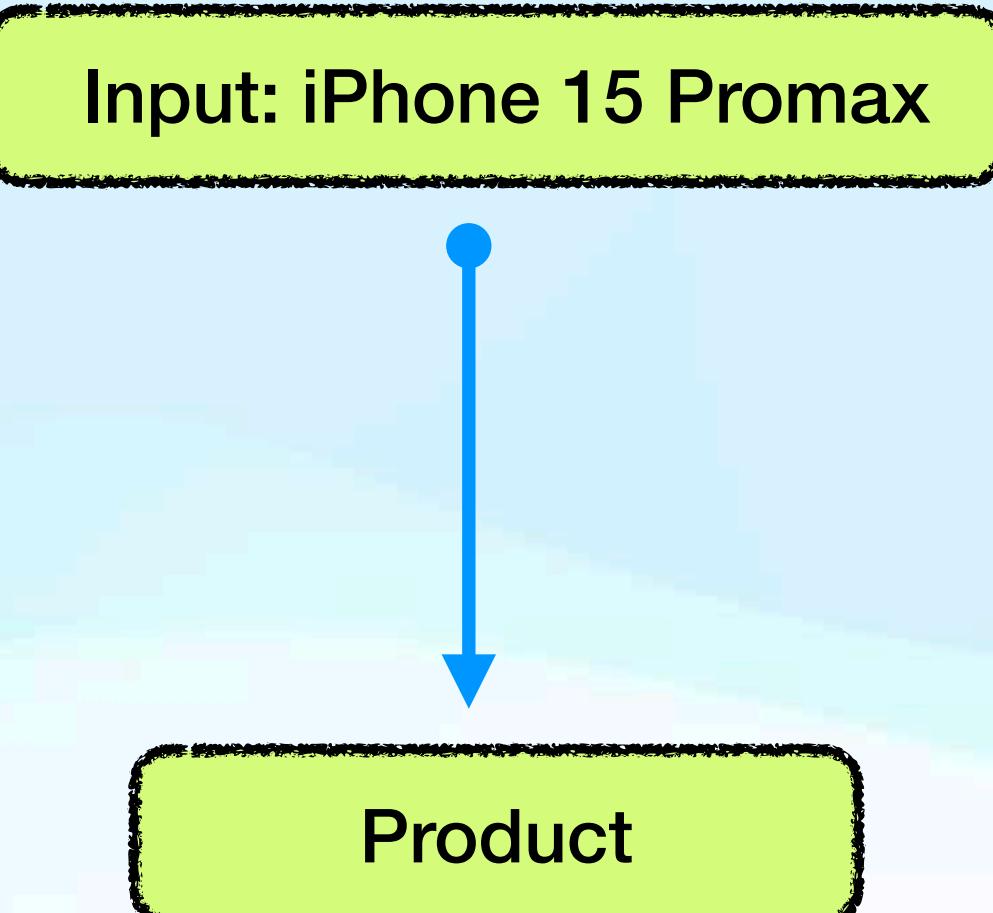


```
stop = 1  
while int(stop) != 0:
```

```
    product_name = input("Enter product name: ")  
    print(f"Recommendations for {product_name}:")  
    product = next(filter(lambda product: product.name == product_name, products), "")
```

```
    if product != "":  
        print(hash_table.recommend_products(product))
```

```
stop = input("Press 0: Stop! \nPress 1 continue!: ")
```



```
def recommend_products(self, product):
```

```
    target_product = self.search(product.category)
```

```
    if len(target_product) == 0:  
        return []
```

```
    recommendations = []
```

```
    for product_hash in target_product:  
        if product.name != product_hash.name:  
            recommendations.append(product_hash.name)
```

```
    return recommendations
```

Product (iPhone 15 Promax)

Category: Phone

```
def search(self, name):  
    index = self.hash_function(name)  
    return self.table[index]
```

[ Prod ] 0

[ Prod ] 1

[] 2

[ Prod , Prod ] 3

Tìm thấy

| Ưu điểm   | Nhược điểm  |
|---|---|
| <ul style="list-style-type: none"><li>✓ <b>Truy cập nhanh:</b> Trong trường hợp tốt nhất và trung bình Hash Table có độ phức tạp là O(1).</li><li>✓ <b>Linh hoạt:</b> Lưu trữ bất kỳ loại dữ liệu nào, từ số nguyên, chuỗi, đến các đối tượng phức tạp.</li><li>✓ <b>Khả năng tự động mở rộng:</b> Một số triển khai của Hash Table có khả năng tự động mở rộng kích thước khi độ lấp đầy tăng lên, giúp duy trì hiệu suất tốt.</li></ul> | <ul style="list-style-type: none"><li>● <b>Xung đột:</b> Khi hai hoặc nhiều khóa có cùng giá trị băm, xảy ra xung đột.</li><li>● <b>Tiêu tốn bộ nhớ:</b> Hash Table thường tiêu tốn nhiều bộ nhớ hơn.</li><li>● <b>Thứ tự không xác định:</b> Hash Table không duy trì thứ tự của các phần tử. Điều này có thể gây ra vấn đề nếu bạn cần truy cập dữ liệu theo một thứ tự cụ thể.</li></ul> |

## Bài toán

- ✓ Giỏ hàng
- ✓ Danh bạ, sổ địa chỉ
- ✓ Các thống kê, tìm kiếm và đếm dữ liệu trùng lặp
- ✓ Hệ thống cơ sở dữ liệu

The image is a composite of three elements. At the top, there's a screenshot of a mobile payment interface with a blue header bar containing a left arrow and the text 'Thanh toán'. Below this is a close-up of a computer keyboard where the standard function keys are replaced by various icons and text: a question mark on the F1 key, a stack of four silver cylinders on the F2 key, and the word 'Database' on the F3 key. In the bottom right corner, there's a pie chart divided into two segments: a large blue segment labeled '35.7%' and a smaller red segment. At the very bottom, there's a snippet of a payment summary: 'Dùng 2000 Shopee Xu' with a toggle switch, 'Tổng thanh toán ₫3.361.800', and a prominent orange 'Đặt hàng' button.

## ❖ Sơ lược

✓ Hashing (băm)

→ Định nghĩa và cấu trúc

→ Hash function -  $F(x)$  (Hàm băm)

→ Các vấn đề xảy ra

✓ Collision (đụng độ)

→ Separate chaining

→ Open Addressing

✓ So sánh

✓ Ứng dụng



Data = 15

Linear probing

Size = 5

25

-1

-1

33

14

0

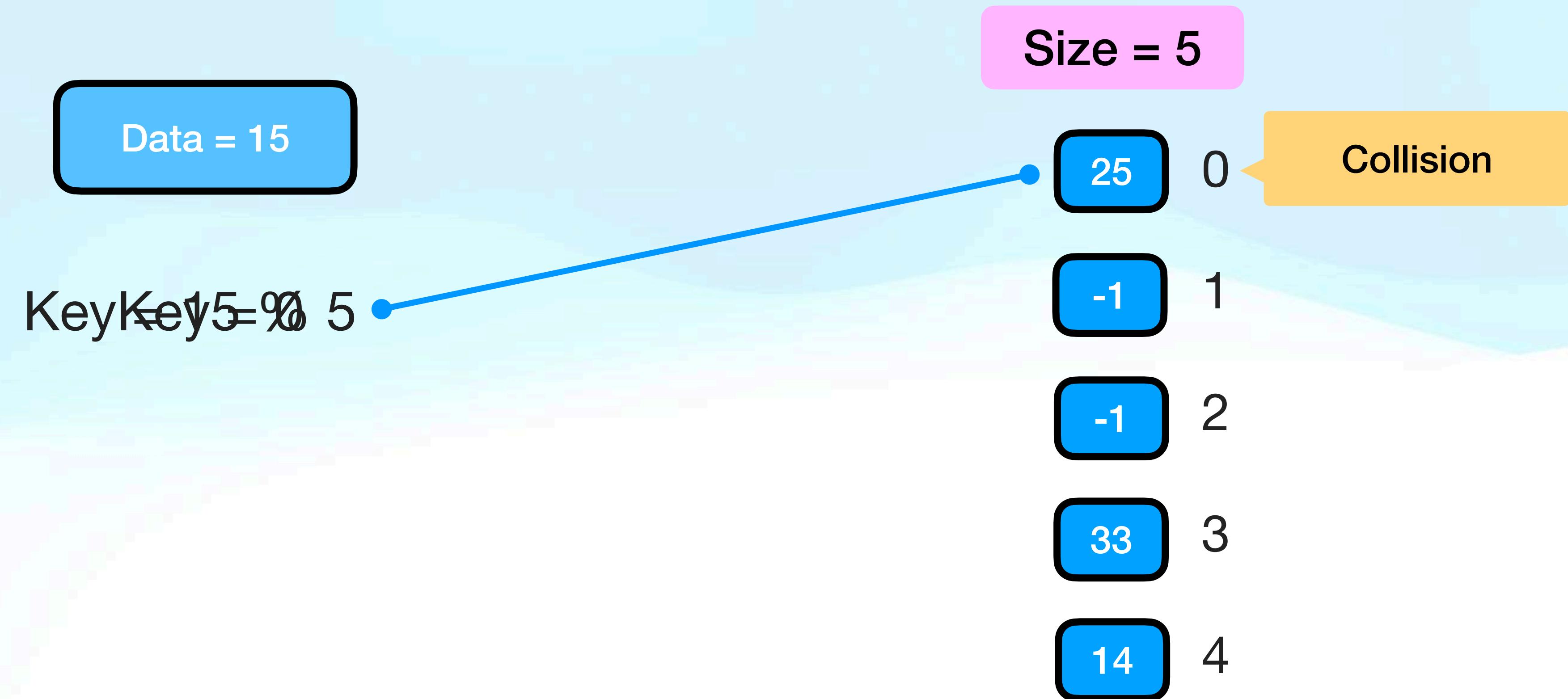
1

2

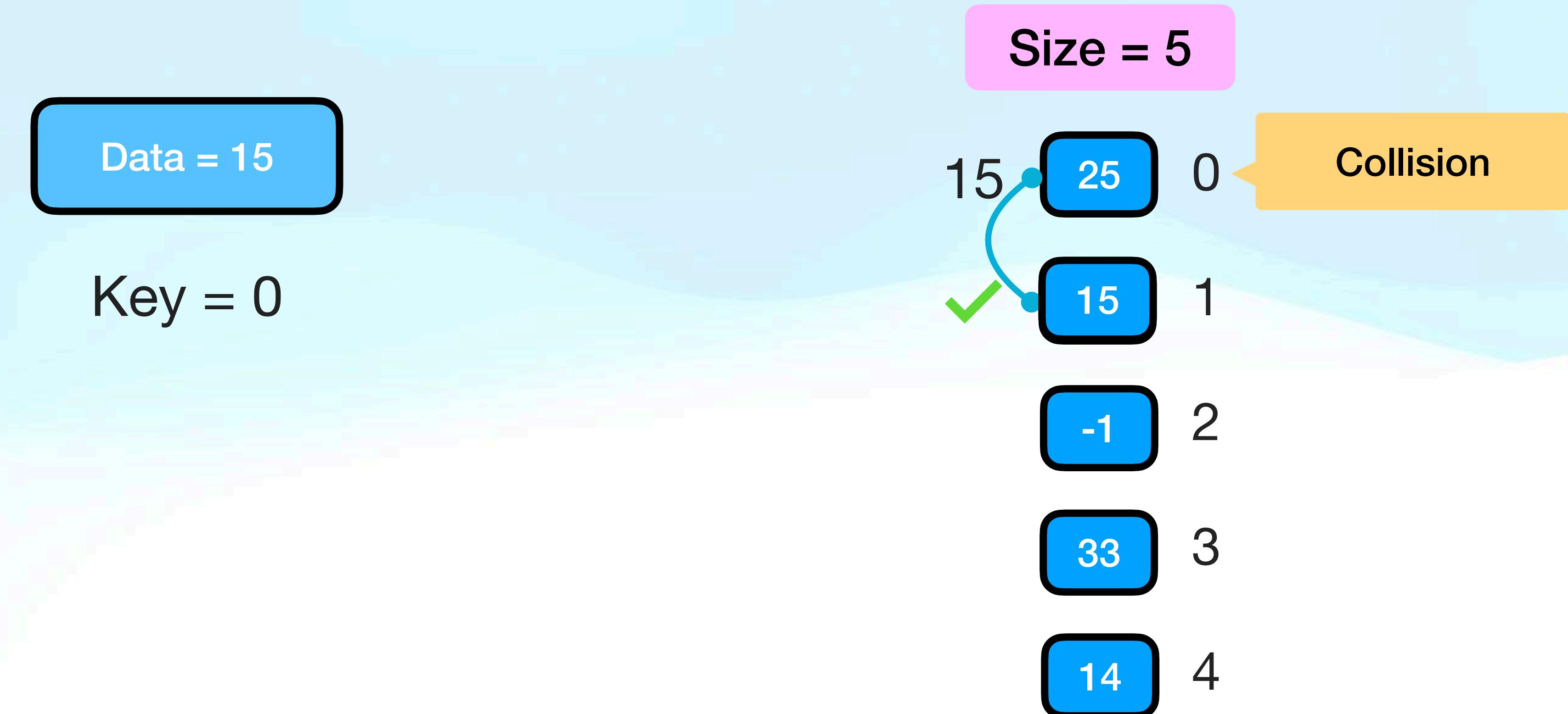
3

4

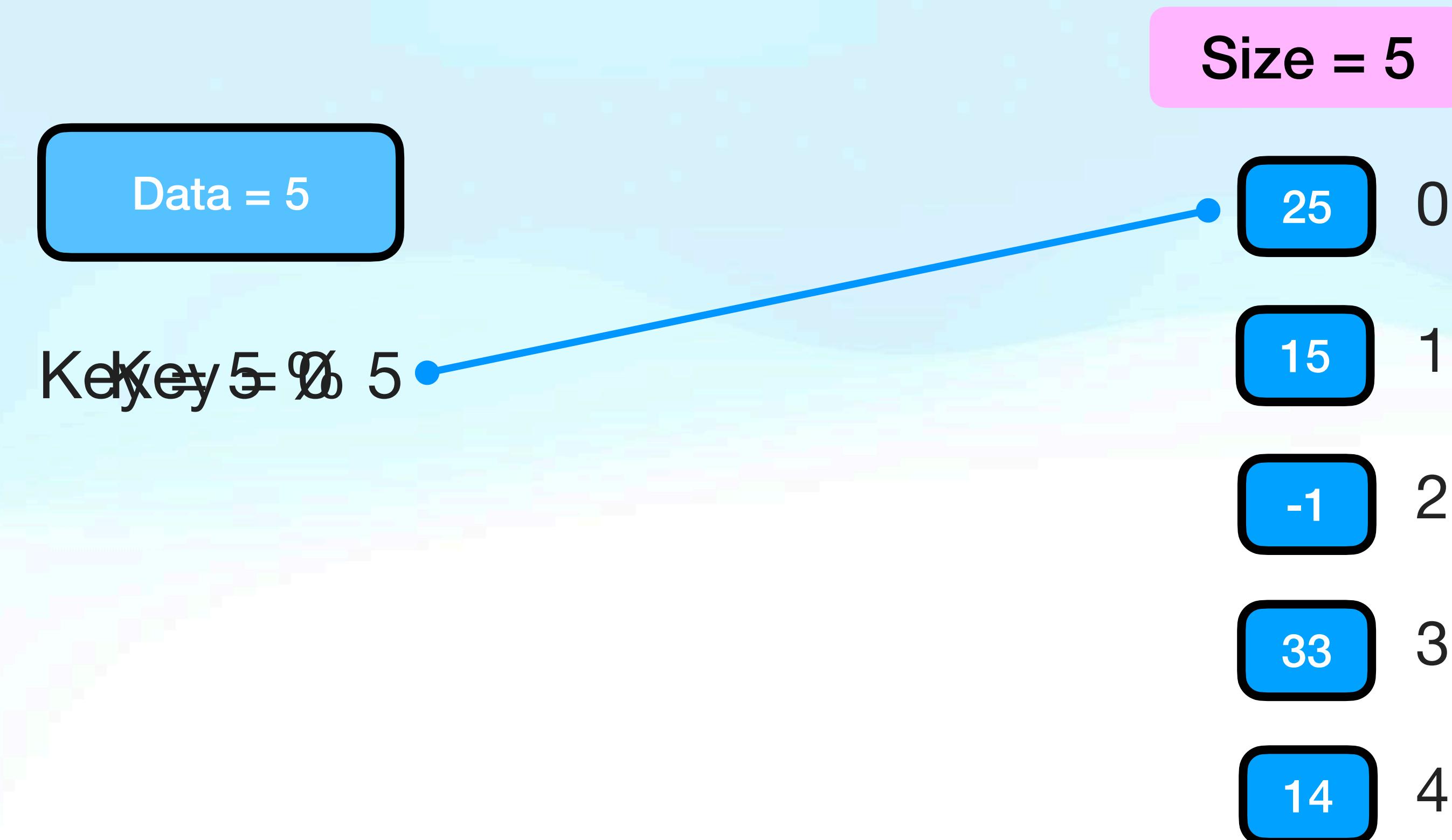
## ❖ Collision - linear probing (Open addressing)



## ❖ Collision - linear probing (Open addressing)



## ❖ Collision - linear probing (Open addressing)

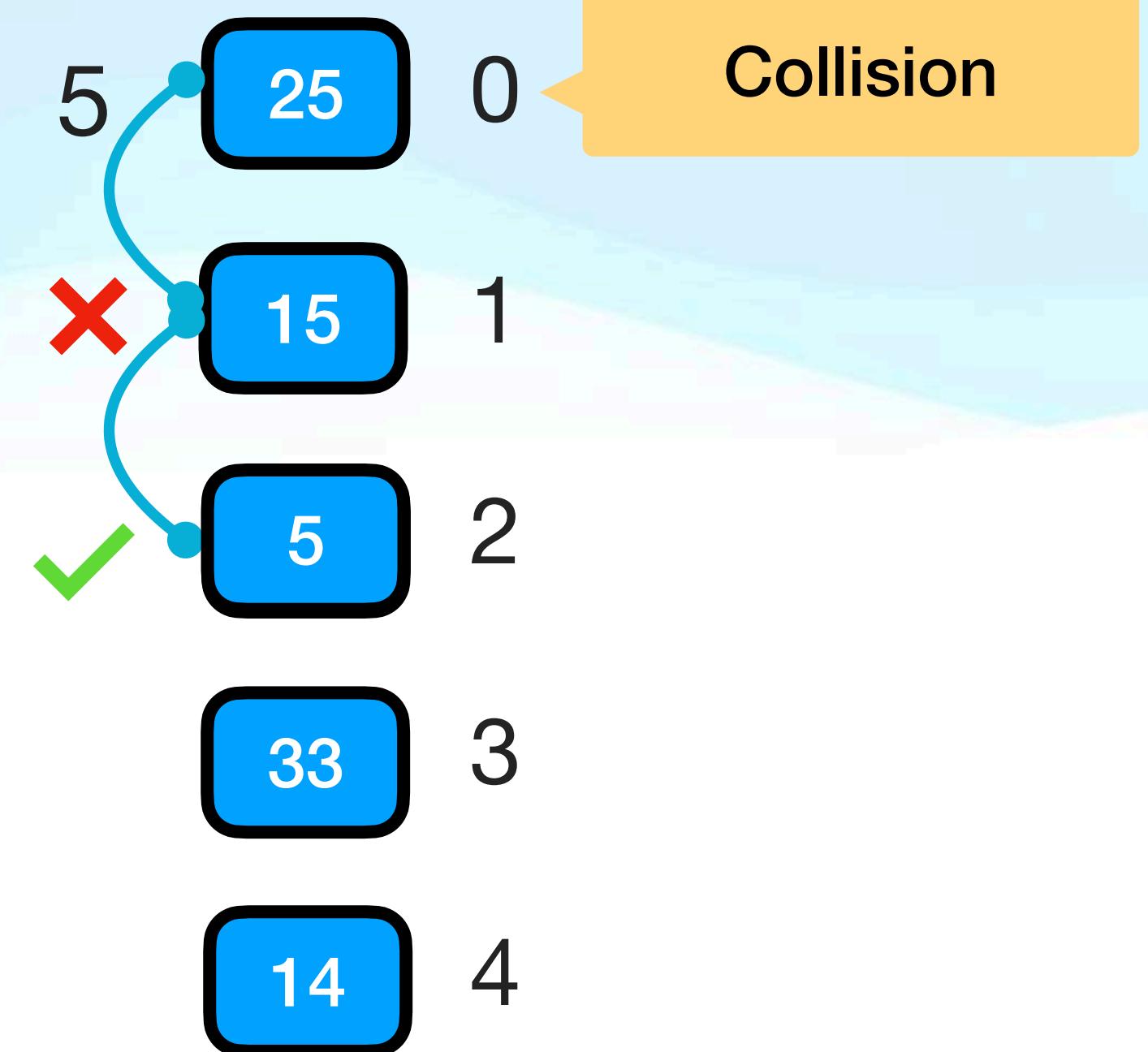


## ❖ Collision - linear probing (Open addressing)

Data = 5

Key = 0

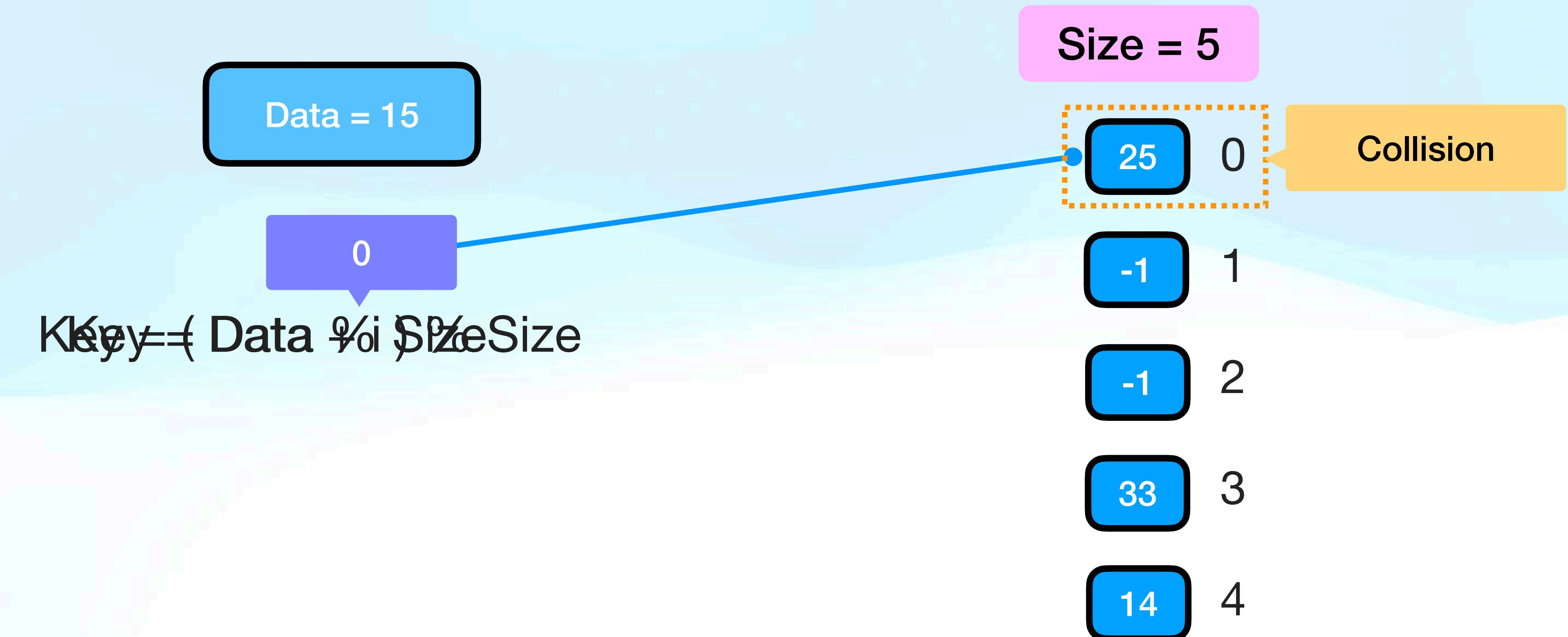
Size = 5



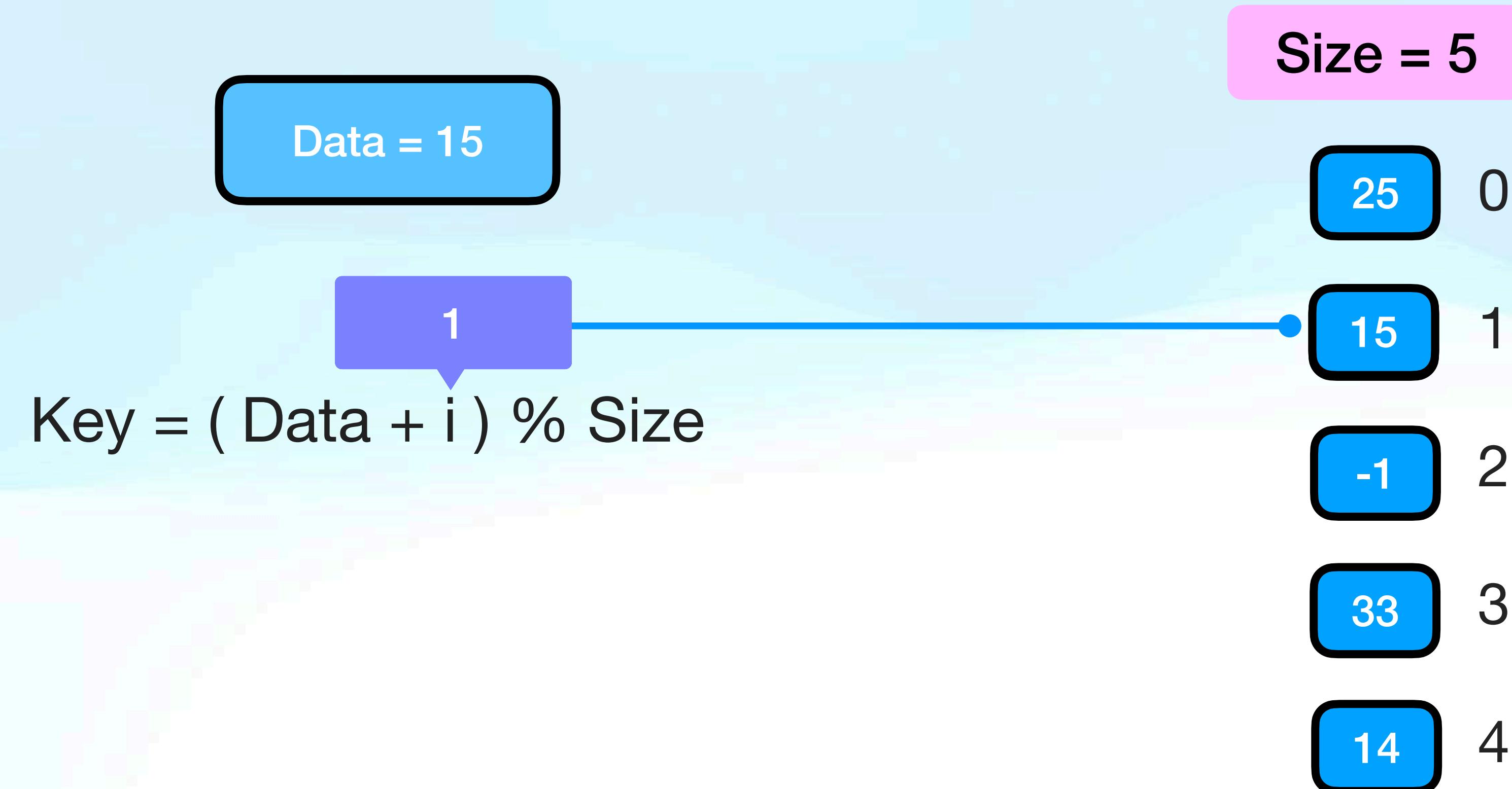
# Linear Probing

$F(X) = (\text{Data} + i) \% \text{Size}$      $i = 1, 2, 3, \dots$

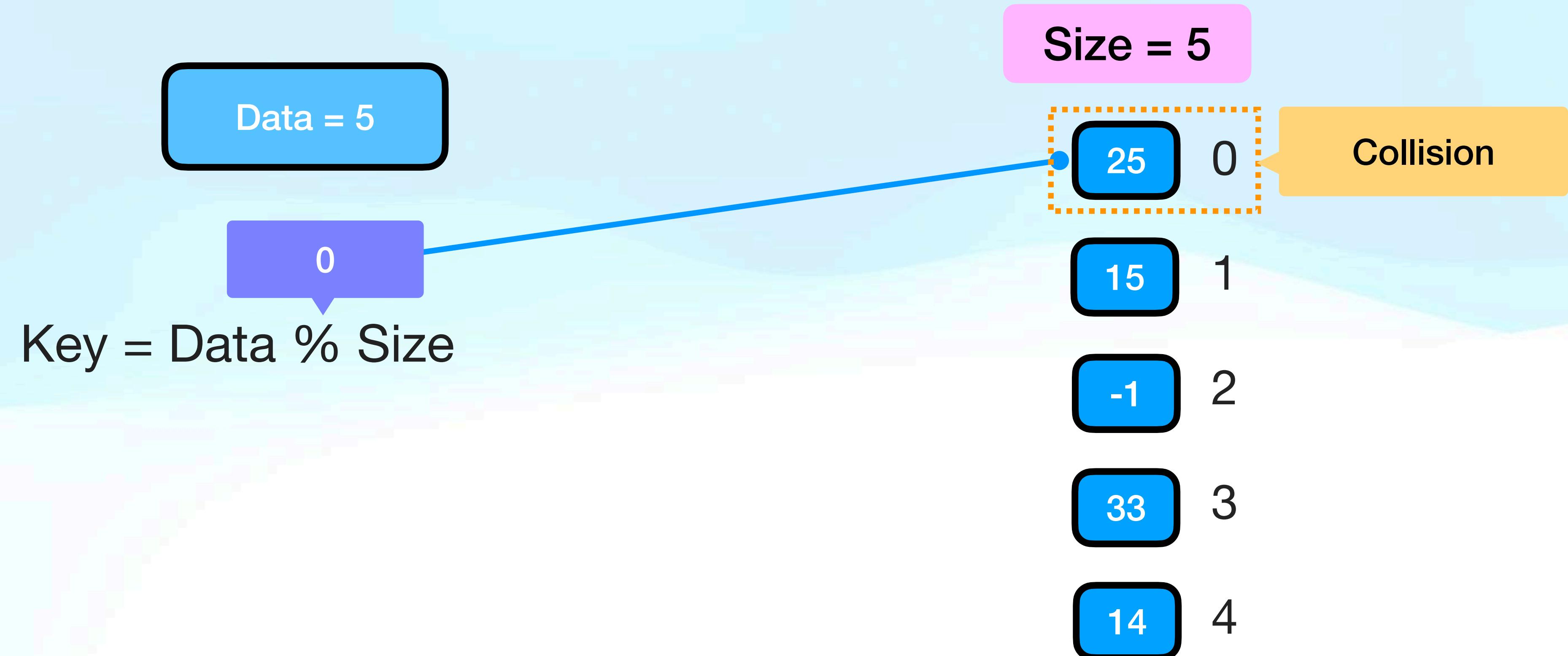
## ❖ Collision - linear probing (Open addressing)



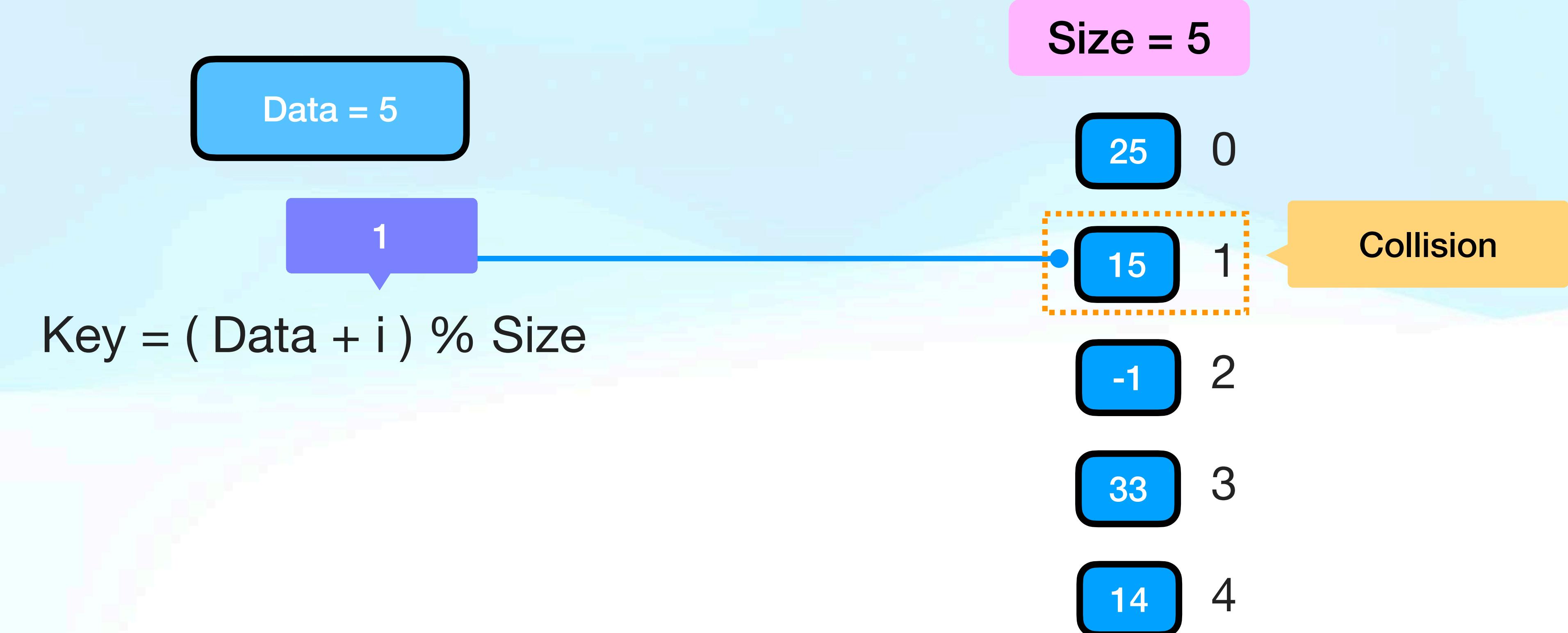
## ❖ Collision - linear probing (Open addressing)



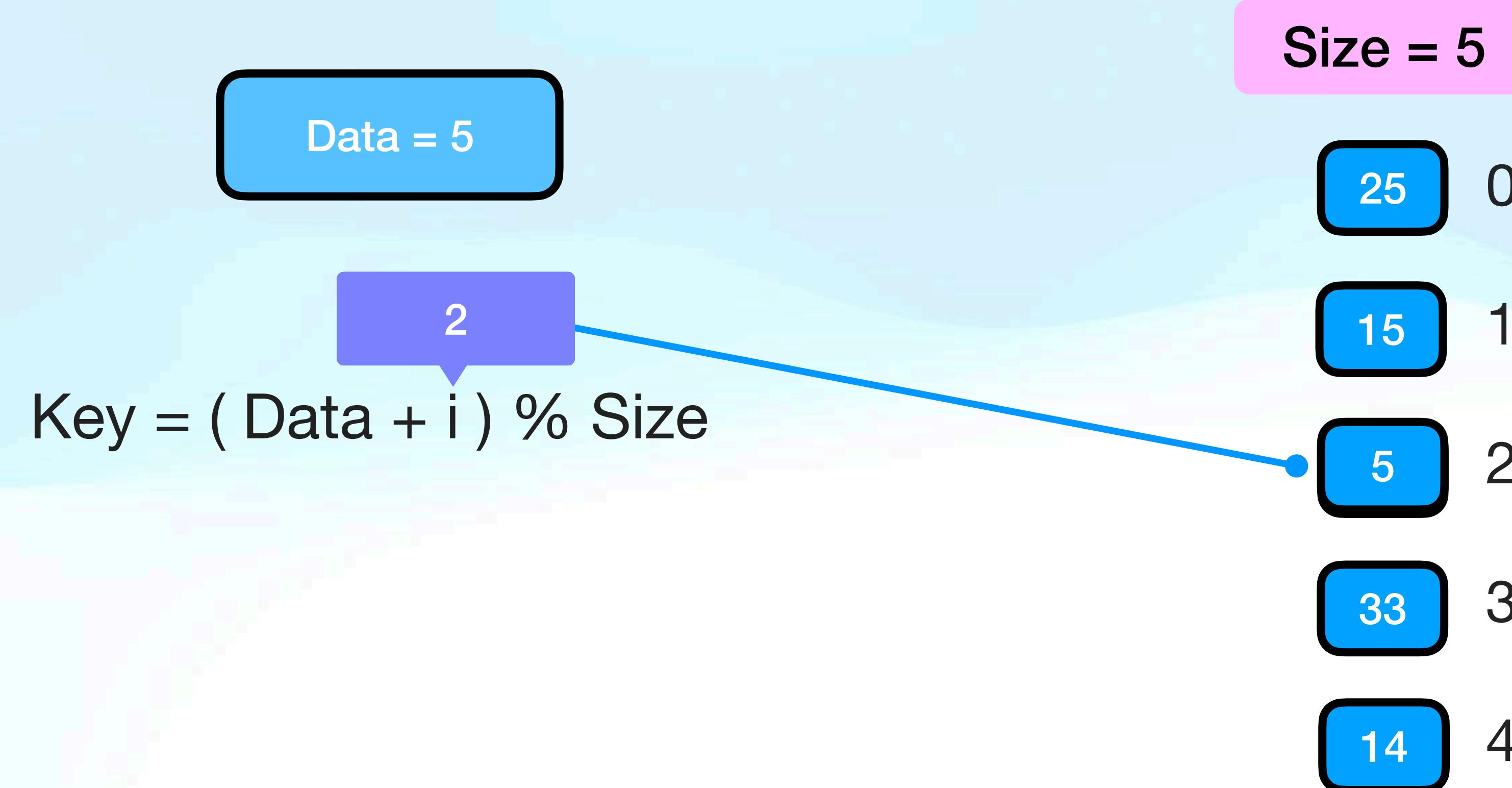
## ❖ Collision - linear probing (Open addressing)



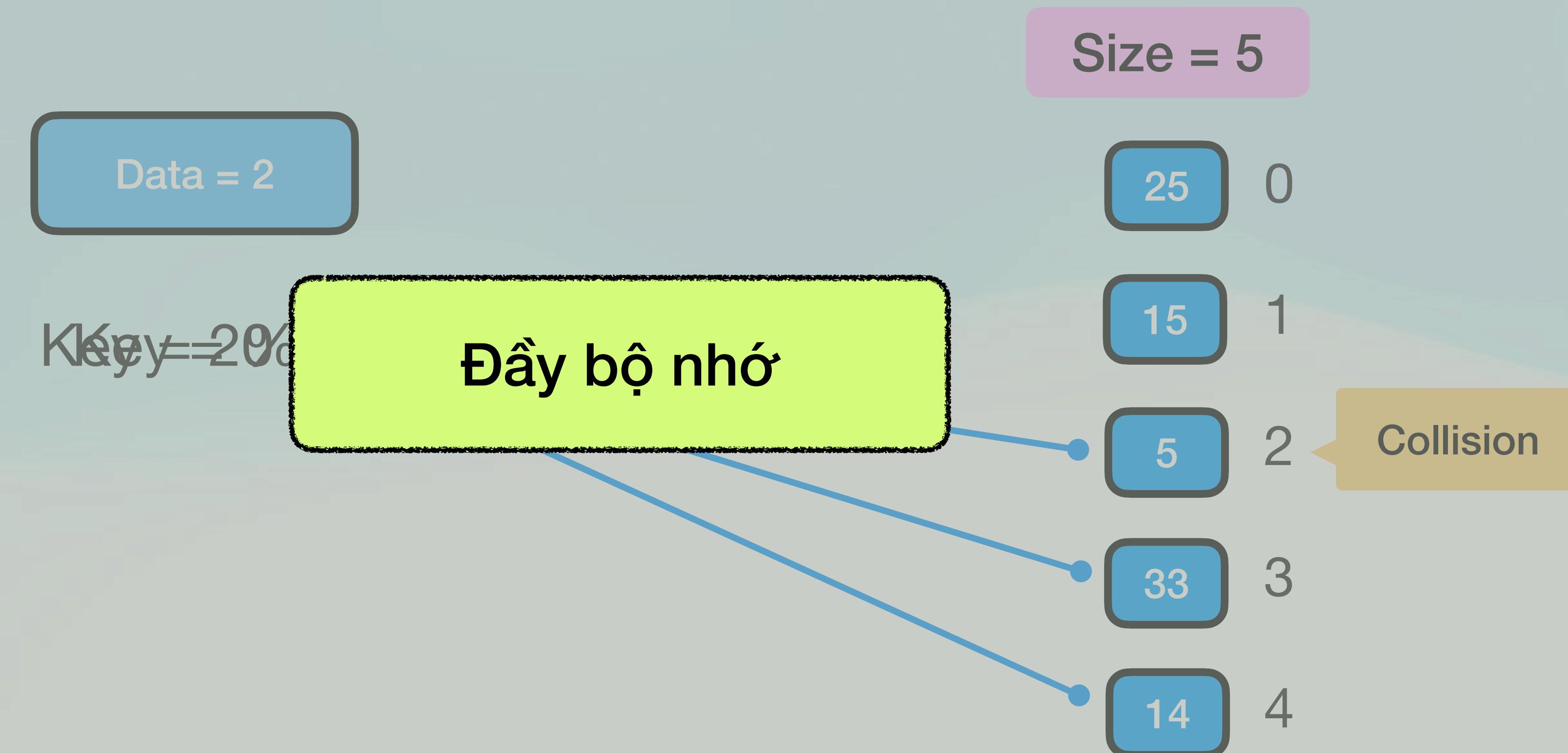
## ❖ Collision - linear probing (Open addressing)



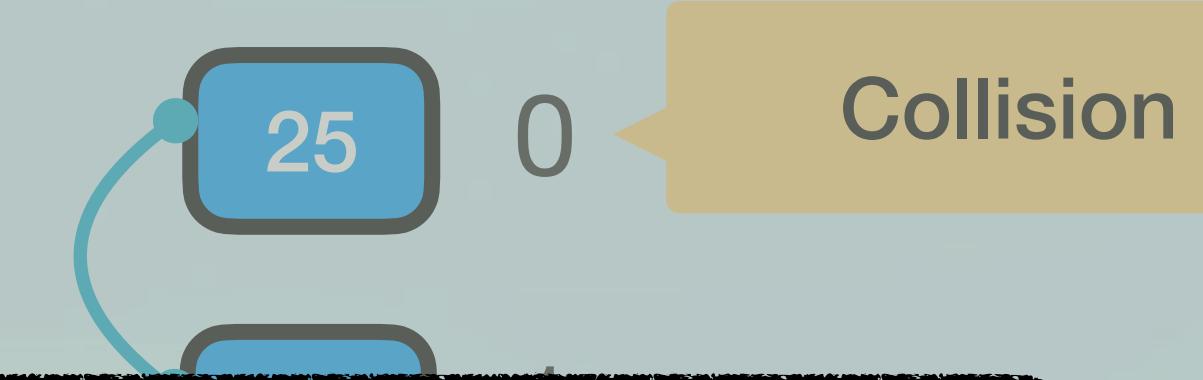
## ❖ Collision - linear probing (Open addressing)



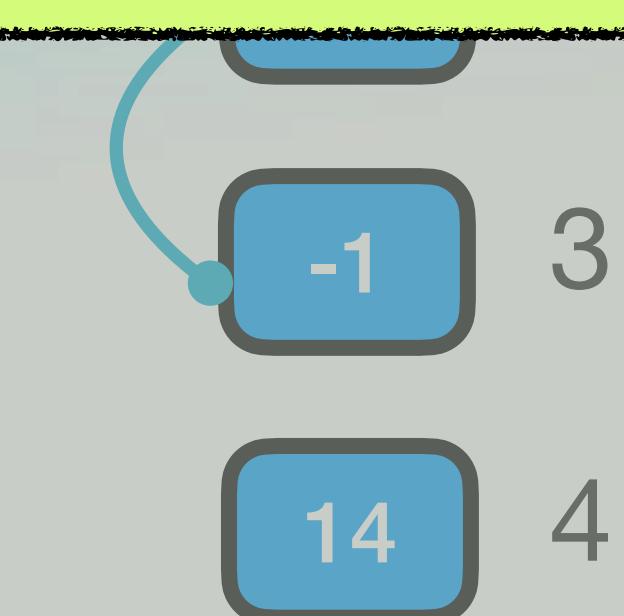
## ❖ Collision - linear probing (Open addressing)



## Linear Probing

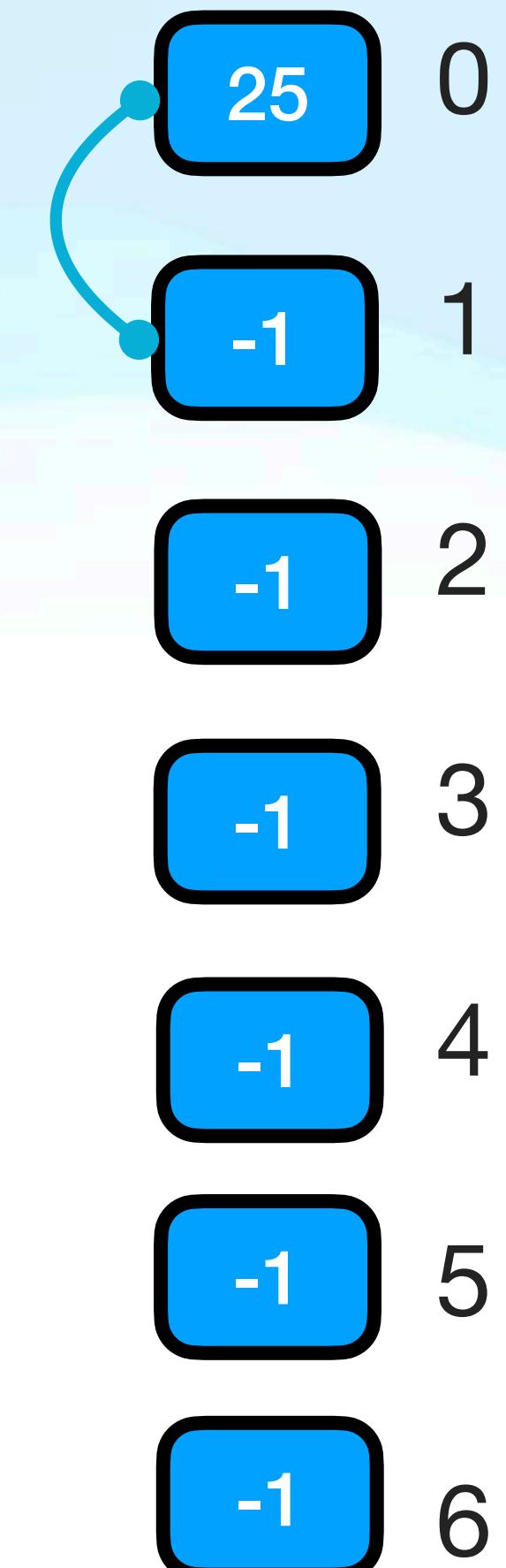


## Quadratic Probing



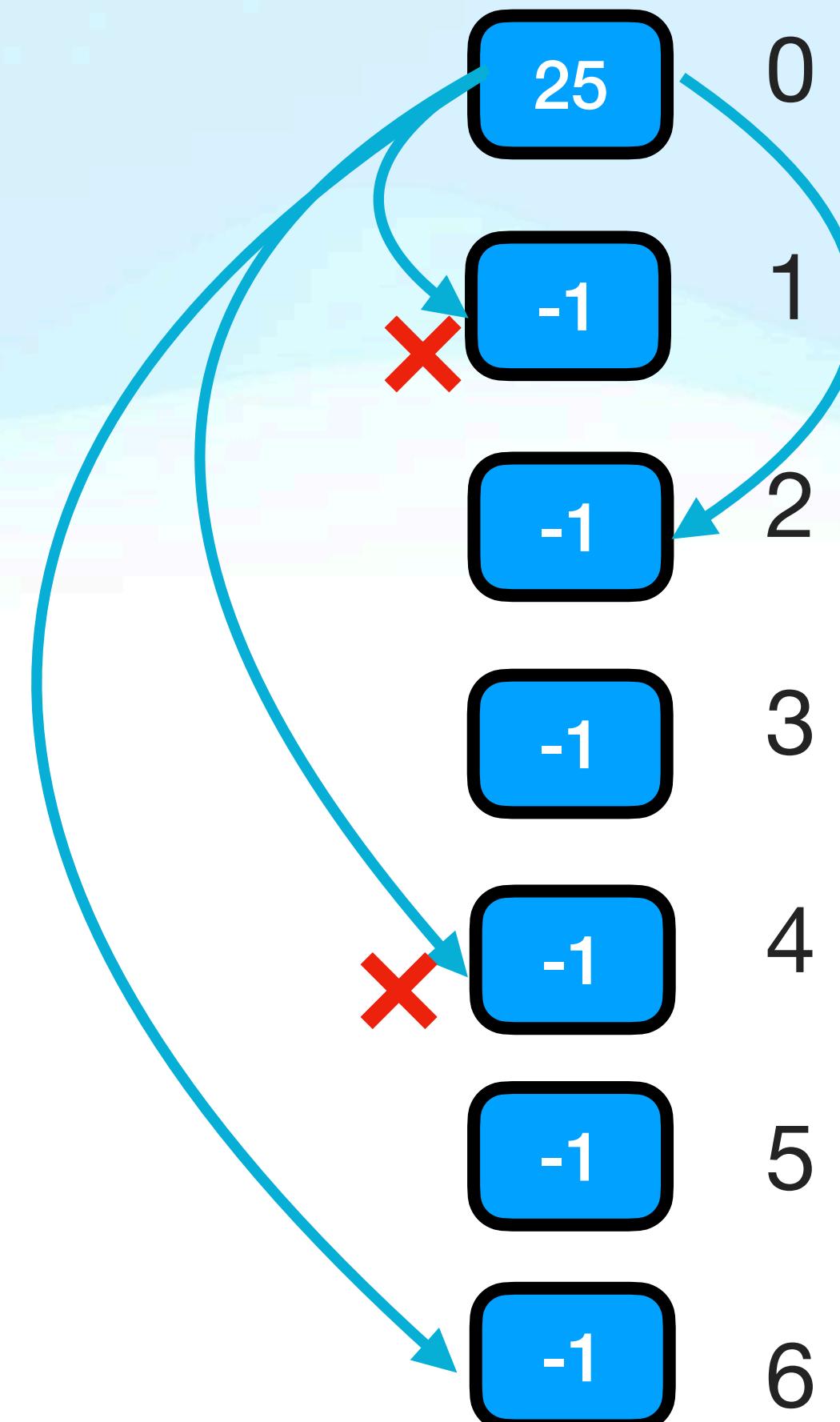
## Quadratic Probing

$H(X) = ((Data + i^2) \% \text{Size}$   
 $i = 1, 2, 3, 4, \dots$



## Quadratic Probing

$F(X) = (\text{Data} + i^2) \% \text{Size}$   
 $i = 1, 2, 3, 4, \dots$

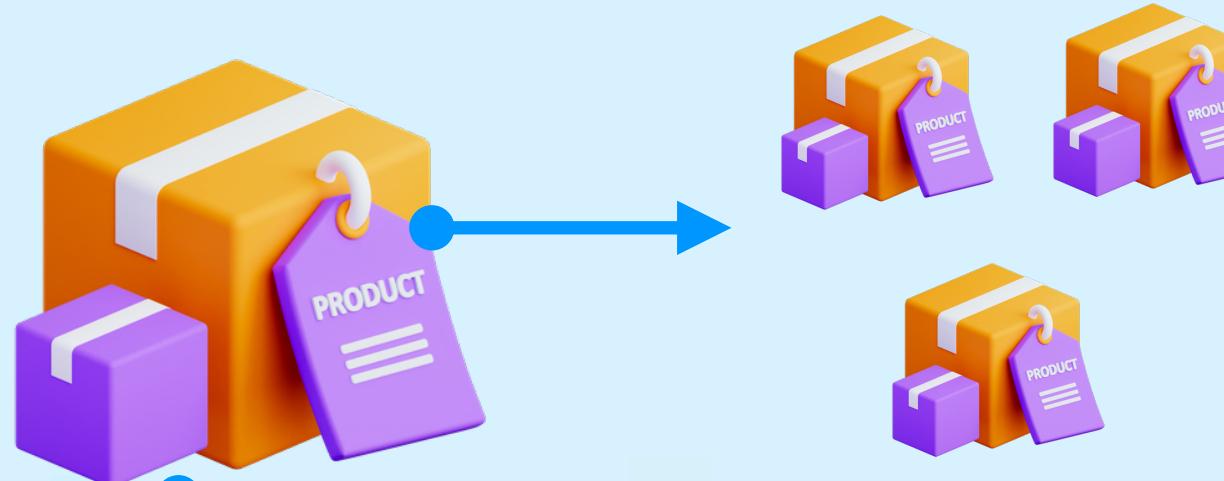


## \* Đánh giá và đề xuất sản phẩm tương tự

Tên sản phẩm

Danh mục

Đơn giá



Hệ thống trả danh sách sản phẩm tương tự.

→ Output: Galaxy Z Fold, Nokia lumia, Oppo Neo, ...

► Khởi tạo một danh sách **Product** chứa các đối tượng sản phẩm

► Xây dựng hash table:

→ Khởi tạo hash function

→ Khởi tạo hàm thêm, tìm kiếm theo hash function

→ Khởi tạo hàm tìm kiếm theo nghiệp vụ

► Chèn dữ liệu vào hash table

► Tìm kiếm và trả kết quả sản phẩm

Khi người dùng nhập vào tên sản phẩm

→ Input: iPhone 15 Promax

## ❖ Bài tập

### Bài toán: Quản lý danh bạ điện thoại:

Giả sử bạn muốn xây dựng một ứng dụng quản lý danh bạ điện thoại. Mỗi mục trong danh bạ sẽ chứa tên của người và số điện thoại của họ. Bạn muốn ứng dụng của mình có khả năng thực hiện các chức năng sau một cách nhanh chóng:

- Thêm một mục mới vào danh bạ.
- Tìm kiếm số điện thoại của một người dựa trên tên của họ.
- Xóa một mục khỏi danh bạ dựa trên tên của người đó.