Data Structure & Algorithm

Data Structure

Hash Table - Bảng Băm

1. Định nghĩa

➢ **Bài toán**: Ta có một tập **tên người** và **số điện thoại**. Làm sao **từ tên người** ta có thể **tra được số điện thoại nhanh nhất** có thể?

Name	PhoneNumber
Jonh Smith	521-8976
Lisa Smith	521-1234
Sandra Dee	521-9655
…	…

1. Định nghĩa

Name | Key
Jonh Smith
Lisa Smith
Sandra Dee
…

Hash Function

Hash Value
01
02
14
…

Bucket

Phone Number Array	
index	**PhoneNumber**
01	521-8976
02	521-1234
…	…
14	521-9655
…	…

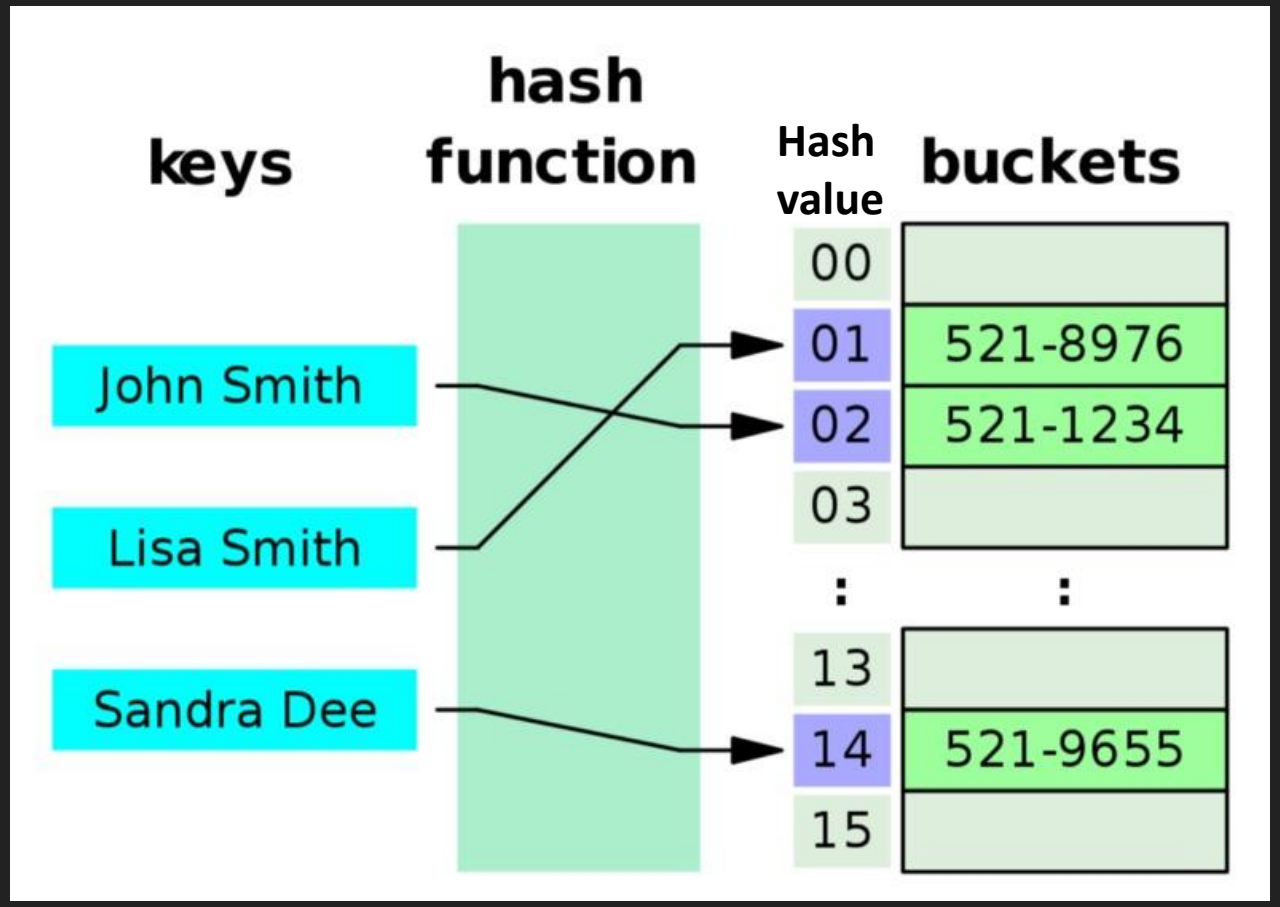
1. Định nghĩa

➢ Hash Table:

Key	Hash Value
Jonh Smith	01
Lisa Smith	02
Sandra Dee	14
…	…

1. Định nghĩa

- **Hash Table**: is a data structure which **organizes data** using **hash functions** in order to support quick insertion and search



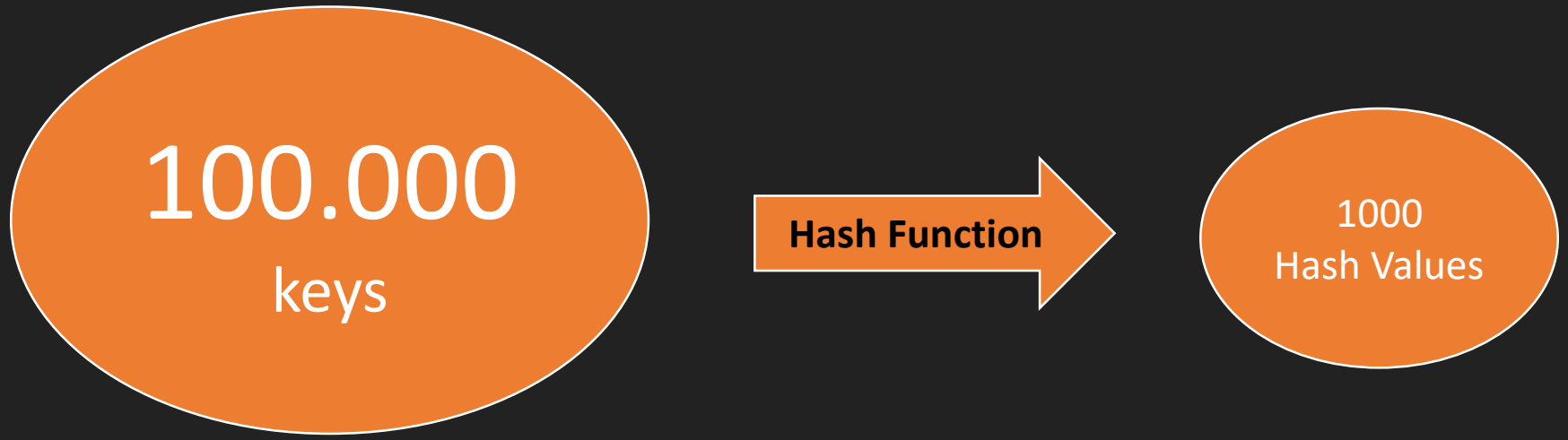
2. Design a Hash Function

- The hash function will depend on the range of key values and the number of buckets.

Key Type	Key Range	Number Of Buckets	Hash Function Example
integer	0 to 100,000	1000	y = x % 1000
char	'a' to 'z'	26	y = x – 'a'
array of integer	size <= 10, xi = [0,1]	1024	$y = x0*2^0 + x1*2^1 + \ldots + x9*2^9$
array of integer	size <= 5, xi = [0..3]	1024	$y = x0*4^0 + x1*4^1 + \ldots + x9*4^4$

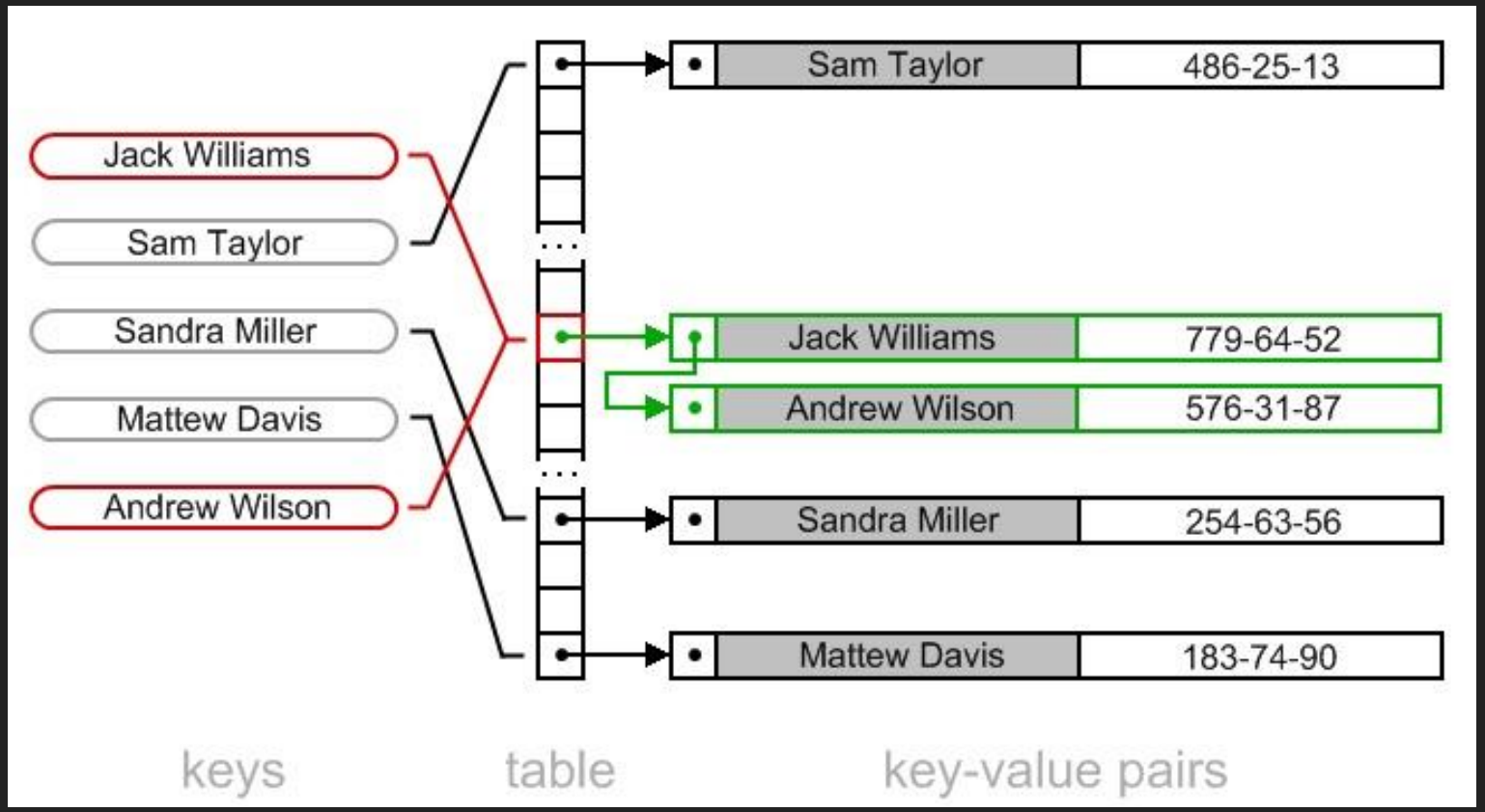
3. Collision Resolution

Key Type	Key Range	Number Of Buckets	Hash Function Example
integer	0 to 100,000	1000	y = x % 1000



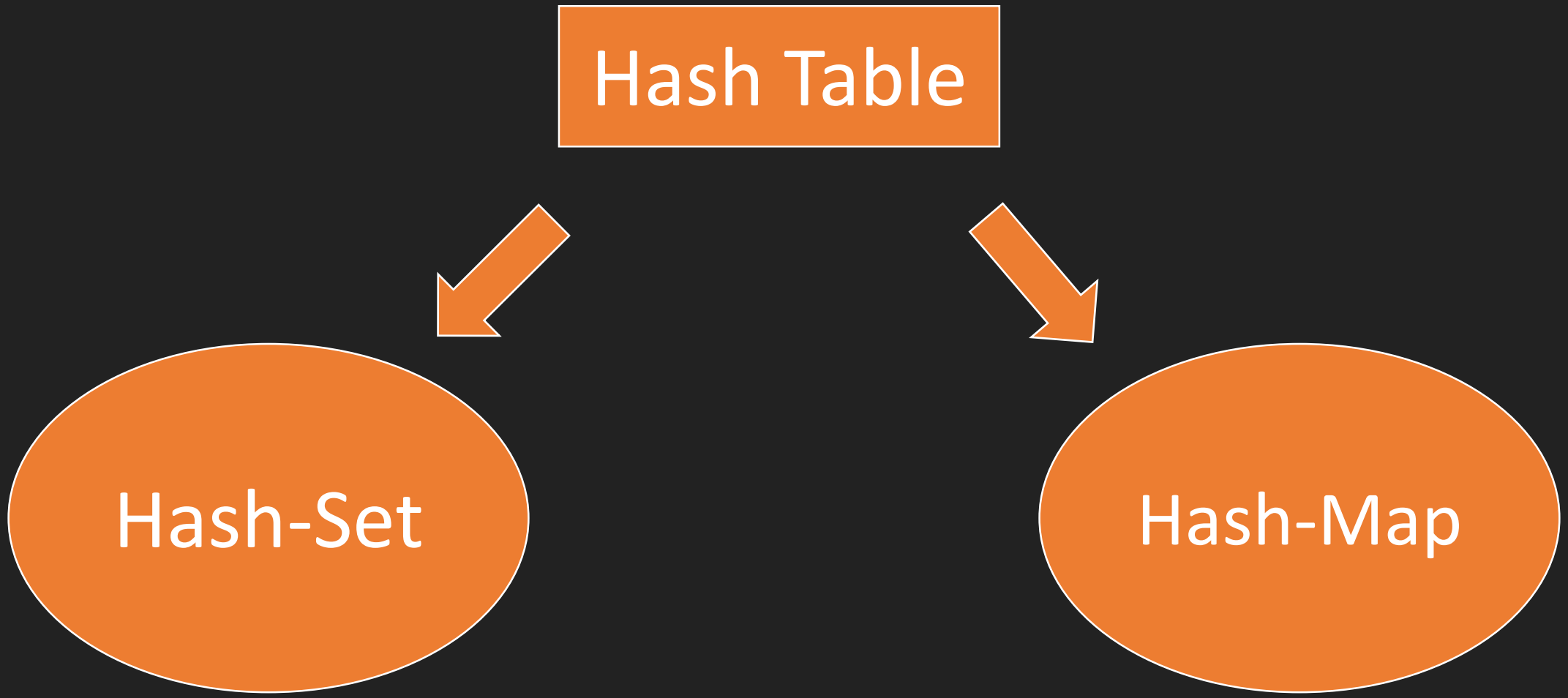
- Trường hợp **lý tưởng** hash function có thể tạo ra maping **1-1**, nhưng đa phần sẽ không được => **Collision**.

3. Collision Resolution



- Để xử lý các collision tạo ra bởi hash function, ta sử dụng linked list hoặc array tại mỗi bucket.

4. Ứng dụng Hash Table.



4. Ứng dụng Hash Table.

- **HashSet**: Thường để quản lý **1 tập các keys**.
  (Khi chúng ta chỉ cần quản lý khoá - key)

Name
Jonh Smith
Lisa Smith
Sandra Dee
…

Hash Function

Hash Value
01
02
14
…

Bucket	
index	
01	Jonh Smith
02	Lisa Smith
…	…
14	Sandra Dee
…	…

=> Bài toán cơ bản nhất:
Kiểm phần tử (key) đã tồn tại hay chưa.

4. Ứng dụng Hash Table.

- **HashMap**: Thường để quản lý **1 tập các cặp key-value**.
  (Khi chúng ta cần quản lý một thông tin nào đó dựa vào khoá - key)

Name
Jonh Smith
Lisa Smith
Sandra Dee
…

Hash Function

Hash Value
01
02
14
…

Bucket	
index	
01	Jonh Smith 521-8976
02	Lisa Smith 521-1234
…	…
14	Sandra Dee 521-9655
…	…

=> Bài toán cơ bản nhất:
Đếm số lượng xuất hiện của các phần tử.

5. Design a HashSet

❖ Bài toán:

Nhập vào n số x.

Tại mỗi thao tác, kiểm tra xem số x đã xuất hiện trước đó hay chưa?

❖ Input:

10

1

2

1

…

❖ Output:

false

false

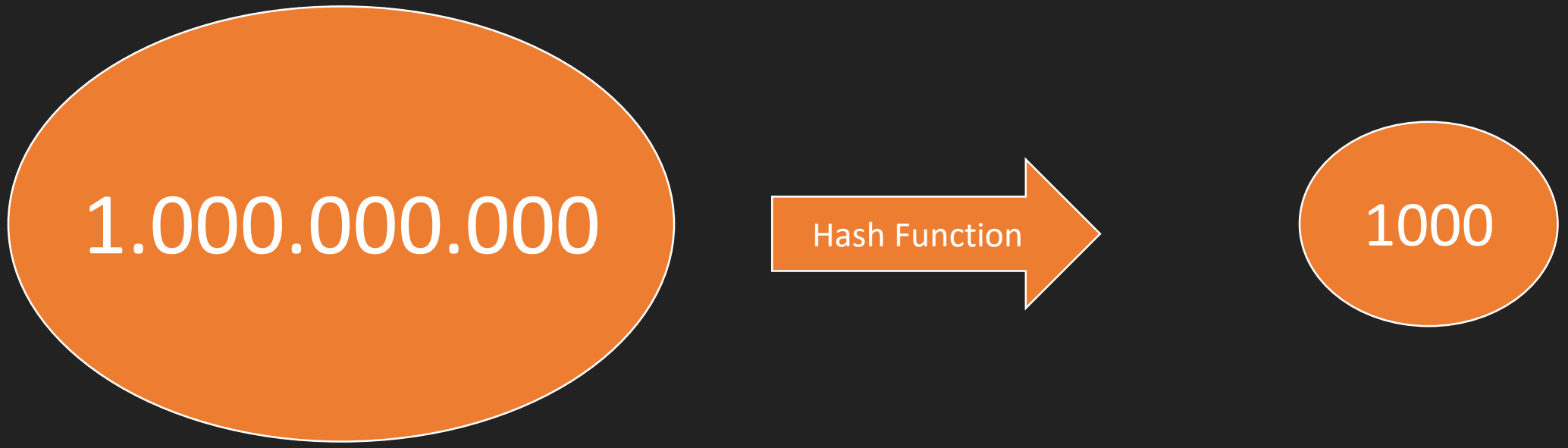
true

…

❖ Đặc điểm:

0 <= x <= 10^9

5. Design a HashSet



❖ **Hash Function:** **y = x % 1000**

5. Design a HashSet

❖ **Hash Function:** y = x % 1000

❖ **Hash Table:**

Key	Hash Function	Hash Value	Bucket
1	1 % 1000	1	1
5000	5000 % 1000	0	5000
5001	5001 % 1000	1	1, 5001
10.000	10.000 % 1000	0	5000, 10.000
…	…	…	…

5. Design a HashSet

❖ **HashSet operations:**

- ✓ hashFunction(int key): Chuyển từ key về hashValue.
- ✓ add(int key): Thêm 1 key vào Set.
- ✓ remove(int key): Xoá key trong Set.
- ✓ contains(int key): Kiểm tra xem có key đó trong Set hay không.

705. Design HashSet

6. Design a HashMap

❖ **Bài toán:**

Nhập vào n số x.

Sau khi nhập xong, in ra danh sách các số x và số lần xuất hiện tương ứng của chúng.

❖ **Input:**

10
1
2
1
…

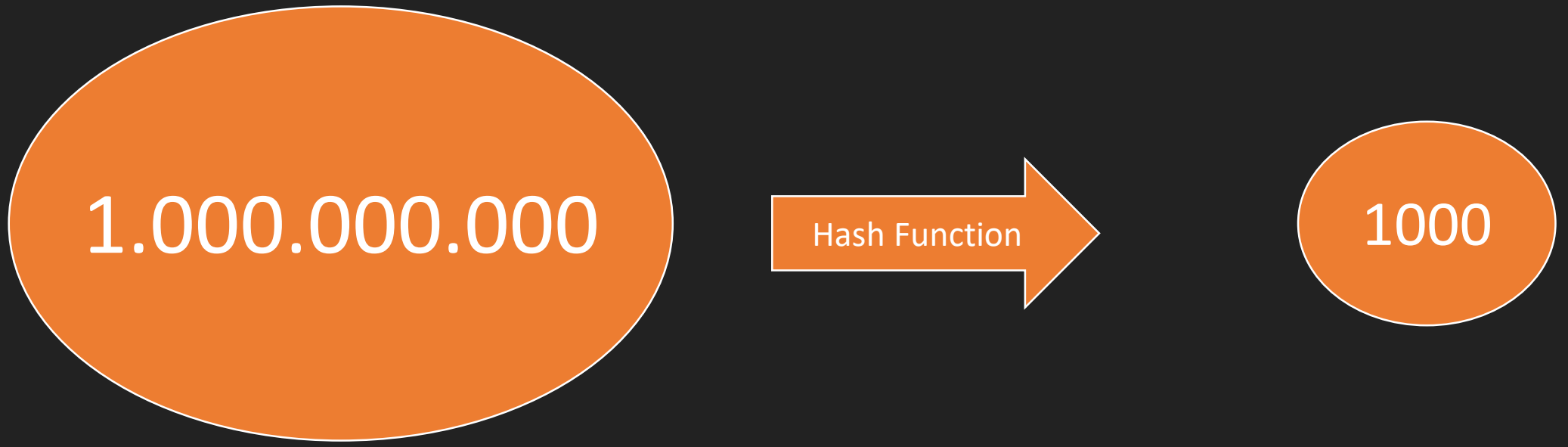
❖ **Output:**

1 2
2 1
…

❖ **Đặc điểm:**

0 <= x <= 10^9

6. Design a HashMap



❖ **Hash Function:** $y = x \% 1000$

6. Design a HashMap

❖ **Hash Function:** y = x % 1000

key value
{ 1 : 1 }

❖ **Hash Table:**

Key	Hash Function	Hash Value	Bucket
1	1 % 1000	1	{1 : 1}
5000	5000 % 1000	0	{5000:1}
5001	5001 % 1000	1	{1:1}, {5001:1}
1	1 % 1000	1	{1 : 2}
2	2 % 1000	2	{2 : 1}
5001	5001 % 1000	1	{1:2}, {5001:2}
…	…	…	…

6. Design a HashMap

❖ **HashMap operations:**

- ✓ hashFunction(int key): Chuyển từ key về hashValue.
- ✓ **put**(int key, int value): Thêm 1 cặp key-value vào Map.
- ✓ remove(int key): Xoá cặp key-value trong Map.
- ✓ **get**(int key): Trả về value của phần tử có khoá key trong Map.

706. Design HashMap