# 题目

不使用任何内建的哈希表库设计一个哈希集合

具体地说，你的设计应该包含以下的功能

add(value)：向哈希集合中插入一个值。

contains(value) ：返回哈希集合中是否存在这个值。

remove(value)：将给定值从哈希集合中删除。如果哈希集合中没有这个值，什么也不做。

**示例:**

MyHashSet hashSet = new MyHashSet();

hashSet.add(1);

hashSet.add(2);

hashSet.contains(1);    // 返回 true

hashSet.contains(3);    // 返回 false (未找到)

hashSet.add(2);

hashSet.contains(2);    // 返回 true

hashSet.remove(2);

hashSet.contains(2);    // 返回 false (已经被删除)

**注意：**

所有的值都在 [0, 1000000]的范围内。

操作的总数目在[1, 10000]范围内。

不要使用内建的哈希集合库。

# 分析

## 方法一：数组/布尔数组（推荐）

**分析：**

利用一个 布尔数组（或位数组）来直接存储每个数是否存在。这种方法非常适用于你这种键范围已知并且较小的情况。

**代码：**

class MyHashSet {

private:

static const int SIZE = 1'000'001; // 因为 key <= 10^6

bool data[SIZE];

public:

MyHashSet() {

for (int i = 0; i < SIZE; ++i) data[i] = false;

}

void add(int key) {

data[key] = true;

}

void remove(int key) {

data[key] = false;

}

bool contains(int key) {

return data[key];

}

};

时间复杂度：add/remove/contains 时间复杂度都是 O(1)

空间复杂度：O(N)，其中 N 是键的范围，即约 10^6 个布尔值 ≈ 1 MB，非常合理

## 方法二：数组+链表

**分析：**是用一段有限数组作为容器，使用哈希函数（这里为key%len，len为数组的长度）算出该数字该放的位置（键值）。若已有数字在内（即发生冲突），利用链表在已有数据的后面插入新数据，解决冲突。这种方法为**链地址法**。

**代码：**

struct Node{

int val;

Node \*next;

Node(int val): val(val),next(nullptr){}

};

const int len = 100; //预分配大小

class MyHashSet {

public:

vector<Node\*> arr; //本题题点

/\*\* Initialize your data structure here. \*/

MyHashSet() {

arr = vector<Node\*>(len, new Node(-1));

}

void add(int key) {

int haval = key % len;

Node\* temp = arr[haval];

if(temp -> val != -1){//如果已经设置初始值，这里增加新节点

while(temp){

if(temp -> val == key) return;

if(!(temp -> next)){

Node \*node = new Node(key);

temp -> next = node;

return;

}

temp = temp -> next;

}

}

else{//如果是没有初始化值的节点，这里直接设置初始化的值即可

temp -> val = key;

return;

}

}

void remove(int key) {

int haval = key % len;

Node\* temp = arr[haval];

if(temp -> val != -1){//不等于-1的才删除，等于-1表示不存在数据

while(temp){

if(temp -> val == key){

temp -> val = -1;//设置为初值即表示删除数据

return;

}

temp = temp -> next;

}

}

}

/\*\* Returns true if this set contains the specified element \*/

bool contains(int key) {

int haval = key % len;

Node\* temp = arr[haval];

while(temp){

if(temp -> val == key) return true;

temp = temp -> next;

}

return false;

}

};