# 题目

不使用任何内建的哈希表库设计一个哈希映射

具体地说，你的设计应该包含以下的功能

put(key, value)：向哈希映射中插入(键,值)的数值对。如果键对应的值已经存在，更新这个值。

get(key)：返回给定的键所对应的值，如果映射中不包含这个键，返回-1。

remove(key)：如果映射中存在这个键，删除这个数值对。

**示例：**

MyHashMap hashMap = new MyHashMap();

hashMap.put(1, 1);

hashMap.put(2, 2);

hashMap.get(1);            // 返回 1

hashMap.get(3);            // 返回 -1 (未找到)

hashMap.put(2, 1);         // 更新已有的值

hashMap.get(2);            // 返回 1

hashMap.remove(2);         // 删除键为2的数据

hashMap.get(2);            // 返回 -1 (未找到)

**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库。

# 分析

## 方法一：取模+数组

**分析：**与设计HashSet不同，这里定义节点的时候需要定义key-value，在HashSet中只需要定义value即可。

**代码：**

struct Node{

int nkey;

int nval;

Node\* next;

Node(int key, int val): nkey(key), nval(val), next(nullptr){}

};

int len = 1000;

class MyHashMap {

public:

vector <Node\*> arr;

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

MyHashMap() {

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

}

/\*\* value will always be non-negative. \*/

void put(int key, int value) {

int temp = key % len;

Node\* h = arr[temp];

Node\* prev = h;

while(h){

if(h -> nkey == key){

h -> nval = value;

return;

}

prev = h;

h = h -> next;

}

Node\* node = new Node(key,value);

//头插法，不用遍历链表插入节点到链表尾部

prev -> next = node;

}

/\*\* Returns the value to which the specified key is mapped, or -1 if this map contains no mapping for the key \*/

int get(int key) {

int temp = key % len;

Node\* h = arr[temp];

while(h){

if(h -> nkey == key) return h -> nval;

h = h -> next;

}

return -1;

}

/\*\* Removes the mapping of the specified value key if this map contains a mapping for the key \*/

void remove(int key) {

int temp = key % len;

Node\* h = arr[temp];

while(h){

if(h -> nkey == key){

h -> nval = -1;

}

h = h -> next;

}

}

};

/\*\*

\* Your MyHashMap object will be instantiated and called as such:

\* MyHashMap\* obj = new MyHashMap();

\* obj->put(key,value);

\* int param\_2 = obj->get(key);

\* obj->remove(key);

\*/