# 题目

给你一个下标从0开始的整数数组nums，同时给你一个整数key，它在nums出现过。

统计在nums数组中紧跟着key后面出现的不同整数target的出现次数。换言之，target的出现次数为满足以下条件的i的数目：

0 <= i <= n - 2

nums[i] == key且

nums[i + 1] == target。

请你返回出现最多次数的target。测试数据保证出现次数最多的target是唯一的。

示例 1：

输入：nums = [1,100,200,1,100], key = 1

输出：100

解释：对于target = 100，在下标1和4处出现过2次，且都紧跟着key。

没有其他整数在key后面紧跟着出现，所以我们返回100。

示例 2：

输入：nums = [2,2,2,2,3], key = 2

输出：2

解释：对于target = 2，在下标1，2和3处出现过3次，且都紧跟着key。

对于target = 3，在下标4出出现过1次，且紧跟着key。

target = 2是紧跟着key之后出现次数最多的数字，所以我们返回2。

提示：

2 <= nums.length <= 1000

1 <= nums[i] <= 1000

测试数据保证答案是唯一的。

# 分析

## 方法一：哈希表

思路：

unordered\_map用于存储目标数字（target）以及它们各自出现的次数。我们遍历nums数组，寻找紧跟在key后面的目标数字，并更新它们的计数。我们还维护了两个变量maxCount和mostFrequentTarget来跟踪出现次数最多的目标数字及其次数。最终，我们返回mostFrequentTarget作为结果。

代码：

class Solution {

public:

int mostFrequent(std::vector<int>& nums, int key) {

// 使用unordered\_map来存储每个目标数字及其出现次数

std::unordered\_map<int, int> targetCount;

// 记录出现次数最多的次数

int maxCount = 0;

// 记录出现次数最多的目标数字

int mostFrequentTarget = 0;

// 遍历数组nums，除了最后一个元素

for (size\_t i = 0; i < nums.size() - 1; ++i) {

// 如果当前元素等于key

if (nums[i] == key) {

// 获取下一个元素作为目标数字

int target = nums[i + 1];

// 在targetCount中增加目标数字的出现次数

targetCount[target]++;

// 如果当前目标数字的出现次数超过了最大次数

if (targetCount[target] > maxCount) {

// 更新最大次数

maxCount = targetCount[target];

// 更新出现次数最多的目标数字

mostFrequentTarget = target;

}

}

}

// 返回出现次数最多的目标数字

return mostFrequentTarget;

}

};

或：

class Solution {

public:

int mostFrequent(vector<int>& nums, int key) {

int n = nums.size();

unordered\_map<int, int> freq; // 统计出现次数的哈希表

for (int i = 0; i < n - 1; ++i) {

if (nums[i] == key) {

freq[nums[i+1]]++; // 记录target的数目

}

}

// 计算并返回最高频元素

int maxfreq = 0;

int res = 0;

for (const auto&it : freq) {

if (it.second > maxfreq) {

res = it.first;

maxfreq = it.second;

}

}

return res;

}

};