> 如果哈希值比较少、特别分散、跨度非常大，使用数组就造成空间的极大浪费！

349. 两个数组的交集

https://leetcode-cn.com/problems/intersection-of-two-arrays/

题意：给定两个数组，编写一个函数来计算它们的交集。

说明： 输出结果中的每个元素一定是唯一的。 我们可以不考虑输出结果的顺序。

思路

这道题目，主要要学会使用一种哈希数据结构：unordered_set，这个数据结构可以解决很多类似的问题。

注意题目特意说明：**输出结果中的每个元素一定是唯一的，也就是说输出的结果的去重的， 同时可以不考虑输出结果的顺序**

这道题用暴力的解法时间复杂度是O(n^2)，那来看看使用哈希法进一步优化。

那么用数组来做哈希表也是不错的选择，例如242. 有效的字母异位词

但是要注意，**使用数组来做哈希的题目，是因为题目都限制了数值的大小。** 是限制了数组的值，还是索引下标的值的范围???

而这道题目没有限制数值的大小，就无法使用数组来做哈希表了。

而且如果哈希值比较少、特别分散、跨度非常大，使用数组就造成空间的极大浪费。

此时就要使用另一种结构体了，set ，关于set，C++ 给提供了如下三种可用的数据结构：

- std::set
- std::multiset
- std::unordered_set

std::set和std::multiset底层实现都是红黑树，std::unordered_set的底层实现是哈希表， 使用unordered_set 读写效率是最高的，并不需要对数据进行排序，而且还不要让数据重复，所以选择unordered_set。

思路如图所示：

C++代码如下：

```
class Solution {
public:
    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
        unordered_set<int> result_set; // 存放结果
        unordered_set<int> nums_set(nums1.begin(), nums1.end());
        for (int num : nums2) {
            // 发现nums2的元素 在nums_set里又出现过
            if (nums_set.find(num) != nums_set.end()) {
                result_set.insert(num);
            }
        }
        return vector<int>(result_set.begin(), result_set.end());
    }
};
```

unordered_set的find和insert两个模板类里的成员函数
vector向量的赋值

拓展

那有同学可能问了，遇到哈希问题我直接都用set不就得了，用什么数组啊。

直接使用set 不仅占用空间比数组大，而且速度要比数组慢，set把数值映射到key上都要做hash计算的。

不要小瞧 这个耗时，在数据量大的情况，差距是很明显的。

其他语言版本

Java：

```
import java.util.HashSet;
import java.util.Set;

class Solution {
    public int[] intersection(int[] nums1, int[] nums2) {
```

```
        if (nums1 == null || nums1.length == 0 || nums2 == null || nums2.length == 0) {
            return new int[0];
        }
        Set<Integer> set1 = new HashSet<>();
        Set<Integer> resSet = new HashSet<>();
        //遍历数组1
        for (int i : nums1) {
            set1.add(i);
        }
        //遍历数组2的过程中判断哈希表中是否存在该元素
        for (int i : nums2) {
            if (set1.contains(i)) {
                resSet.add(i);
            }
        }
        int[] resArr = new int[resSet.size()];
        int index = 0;
        //将结果几何转为数组
        for (int i : resSet) {
            resArr[index++] = i;
        }
        return resArr;
    }
}
```

Python：

```
class Solution:
    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:
        result_set = set()

        set1 = set(nums1)
        for num in nums2:
            if num in set1:
                result_set.add(num) # set1里出现的nums2元素 存放到结果
        return result_set
```

Go：

```
func intersection(nums1 []int, nums2 []int) []int {
    m := make(map[int]int)
    for _, v := range nums1 {
        m[v] = 1
    }
    var res []int
    // 利用count>0，实现重复值只拿一次放入返回结果中
    for _, v := range nums2 {
        if count, ok := m[v]; ok && count > 0 {
            res = append(res, v)
            m[v]--
        }
    }
    return res
}
```

javaScript:

```
/**
 * @param {number[]} nums1
 * @param {number[]} nums2
 * @return {number[]}
 */
var intersection = function(nums1, nums2) {
    // 根据数组大小交换操作的数组
    if(nums1.length < nums2.length) {
        const _ = nums1;
        nums1 = nums2;
        nums2 = _;
    }
    const nums1Set = new Set(nums1);
    const resSet = new Set();
    // for(const n of nums2) {
    //     nums1Set.has(n) && resSet.add(n);
    // }
    // 循环 比 迭代器快
    for(let i = nums2.length - 1; i >= 0; i--) {
        nums1Set.has(nums2[i]) && resSet.add(nums2[i]);
    }
    return Array.from(resSet);
};
```

相关题目

- 350.两个数组的交集 II

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录