```
/*
给定两个数组，编写一个函数来计算它们的交集。

示例 1:

输入: nums1 = [1,2,2,1], nums2 = [2,2]
输出: [2,2]

示例 2:

输入: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
输出: [4,9]

说明:

    输出结果中每个元素出现的次数，应与元素在两个数组中出现的次数一致。
    我们可以不考虑输出结果的顺序。

进阶:

    如果给定的数组已经排好序呢？你将如何优化你的算法？
    如果 nums1 的大小比 nums2 小很多，哪种方法更优？
    如果 nums2 的元素存储在磁盘上，磁盘内存是有限的，并且你不能一次加载所有的元素到内存中，你该怎么办？

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/intersection-of-two-arrays-ii
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 方法一: 使用map数据结果统计两个数组每个元素出现的次数,然后遍历其中一个map与另外一个map对照即可(与349 两个数组的交集的解法基本一致).
- 其他进阶方法等二刷.

方法一:C++_map数据库统计查询法

```
class Solution
{
    public:
        vector<int> intersect(vector<int>& nums1, vector<int>& nums2)
        {
            map<int,int>    mii1            ;
            map<int,int>    mii2            ;
            vector<int>     ret_val         ;
            int             i       =   0   ;
            int             j       =   0   ;
            int             temp    =   0   ;


```

```
            for(i = 0 ; i < nums1.size() ; i++)
            {
                if(mii1.count(nums1[i]))
                {
                    mii1[nums1[i]] ++;

                }
                else
                {
                    mii1[nums1[i]] = 1;
                }
            }


            for(i = 0 ; i < nums2.size() ; i++)
            {
                if(mii2.count(nums2[i]))
                {
                    mii2[nums2[i]] ++;

                }
                else
                {
                    mii2[nums2[i]] = 1;
                }
            }


            map<int,int>::iterator iter = mii1.begin();
            while(iter!=mii1.end())
            {
                temp = iter->first;
                if(mii2.count(temp))
                {
                    j = min(mii1[temp],mii2[temp]);
                    while(j--)
                    {
                        ret_val.push_back(temp);
                    }
                }
                iter++;
            }
            return ret_val;
        }
};


/*
执行结果:
通过
显示详情
执行用时 :20 ms, 在所有 cpp 提交中击败了20.76% 的用户
内存消耗 :9.8 MB, 在所有 cpp 提交中击败了5.03%的用户
*/
```

2019.11.11