Intersection of Two Arrays

Given two arrays, write a function to compute their intersection.

Example:
Given *nums1* = `[1, 2, 2, 1]`, *nums2* = `[2, 2]`, return `[2]`.

Note:

- Each element in the result must be unique.
- The result can be in any order.

Solution 1

Use two hash sets

Time complexity: O(n)

```
public class Solution {
    public int[] intersection(int[] nums1, int[] nums2) {
        Set<Integer> set = new HashSet<>();
        Set<Integer> intersect = new HashSet<>();
        for (int i = 0; i < nums1.length; i++) {
            set.add(nums1[i]);
        }
        for (int i = 0; i < nums2.length; i++) {
            if (set.contains(nums2[i])) {
                intersect.add(nums2[i]);
            }
        }
        int[] result = new int[intersect.size()];
        int i = 0;
        for (Integer num : intersect) {
            result[i++] = num;
        }
        return result;
    }
}
```

Sort both arrays, use two pointers

Time complexity: O(nlogn)

```java
public class Solution {
    public int[] intersection(int[] nums1, int[] nums2) {
        Set<Integer> set = new HashSet<>();
        Arrays.sort(nums1);
        Arrays.sort(nums2);
        int i = 0;
        int j = 0;
        while (i < nums1.length && j < nums2.length) {
            if (nums1[i] < nums2[j]) {
                i++;
            } else if (nums1[i] > nums2[j]) {
                j++;
            } else {
                set.add(nums1[i]);
                i++;
                j++;
            }
        }
        int[] result = new int[set.size()];
        int k = 0;
        for (Integer num : set) {
            result[k++] = num;
        }
        return result;
    }
}
```

Binary search

Time complexity: O(nlogn)

```java
public class Solution {
    public int[] intersection(int[] nums1, int[] nums2) {
        Set<Integer> set = new HashSet<>();
        Arrays.sort(nums2);
        for (Integer num : nums1) {
            if (binarySearch(nums2, num)) {
                set.add(num);
            }
        }
        int i = 0;
        int[] result = new int[set.size()];
        for (Integer num : set) {
            result[i++] = num;
        }
        return result;
    }

    public boolean binarySearch(int[] nums, int target) {
        int low = 0;
        int high = nums.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (nums[mid] == target) {
                return true;
            }
            if (nums[mid] > target) {
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        return false;
    }
}
```

written by divingboy89 original link here

Solution 2

```
class Solution {
public:
    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
        unordered_set<int> m(nums1.begin(), nums1.end());
        vector<int> res;
        for (auto a : nums2)
            if (m.count(a)) {
                res.push_back(a);
                m.erase(a);
            }
        return res;
    }
};
```

written by Kenigma original link here

Solution 3

```
vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
    set<int> s(nums1.begin(), nums1.end());
    vector<int> out;
    for (int x : nums2)
        if (s.erase(x))
            out.push_back(x);
    return out;
}
```

written by StefanPochmann original link here