# Q1. <https://leetcode.com/problems/best-team-with-no-conflicts/>

## Solution:

# Q2. <https://leetcode.com/problems/majority-element-ii/>

## Solution*:*

import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 30/06/2023  
 \*/*public class MajorityElement {  
 public static void main(String[] args) {  
 System.*out*.println(*majorityElement*(new int[]{3, 2, 3}));  
 }  
   
 public static List<Integer> majorityElement(int[] nums) {  
 HashMap<Integer, Integer> map = new HashMap<>();  
 for (int num : nums) {  
 if (map.containsKey(num)) {  
 int count = map.get(num) + 1;  
 map.put(num, count);  
 } else {  
 map.put(num, 1);  
 }  
 }  
 List<Integer> list = new ArrayList<>();  
 for (Map.Entry<Integer, Integer> entry : map.entrySet()) {  
 int value = entry.getValue();  
 int key = entry.getKey();  
 if (value > nums.length / 3) {  
 list.add(key);  
 }  
 }  
 return list;  
 }  
}

# Q3. <https://leetcode.com/problems/merge-intervals>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Comparator;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 30/06/2023  
 \*/  
 /\*Input: intervals = [[1,3],[2,6],[8,10],[15,18]]  
 Output: [[1,6],[8,10],[15,18]]  
 Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].\*/*public class MergeIntervals {  
 public int[][] merge(int[][] intervals) {  
 Arrays.*sort*(intervals, new Comparator<int[]>() {  
 @Override  
 public int compare(int[] o1, int[] o2) {  
 return o1[0] - o2[0];  
 }  
 });  
 *//Arrays.sort(intervals, (o1, o2) -> o1[0] - o2[0]);  
 //Arrays.sort(intervals, Comparator.comparingInt(o -> o[0]));* List<int[]> ans = new ArrayList<>();  
 ans.add(intervals[0]);  
 for (int i = 1; i < intervals.length; i++) {  
 if (ans.get(ans.size() - 1)[1] >= intervals[i][0]) {  
 ans.get(ans.size() - 1)[1] = Math.*max*(ans.get(ans.size() - 1)[1], intervals[i][1]);  
 } else {  
 ans.add(intervals[i]);  
 }  
 }  
 return ans.toArray(new int[ans.size() - 1][]);  
 }  
}

# Q4. <https://leetcode.com/problems/sort-colors/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 27/06/2023  
 \*/*public class SortColors {  
 public static void main(String[] args) {  
 int[] arr1 = {2, 0, 2, 1, 1, 0};  
 *sortColor*(arr1);  
 System.*out*.println(Arrays.*toString*(arr1));  
 }  
  
 static void sortColor(int[] nums) {  
 int zero = 0, one = 0, two = 0;  
  
 for (int num : nums) {  
 if (num == 0) zero++;  
 else if (num == 1) one++;  
 else if (num == 2) two++;  
 }  
  
 for (int i = 0; i < nums.length; i++) {  
 if (zero != 0) {  
 nums[i] = 0;  
 zero--;  
 } else if (one != 0) {  
 nums[i] = 1;  
 one--;  
 } else {  
 nums[i] = 2;  
 }  
 }  
 }  
}

# Q5. <https://leetcode.com/problems/intersection-of-two-arrays/>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 28/06/2023  
 \*/*public class IntersectionOfTwoArray {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*intersection*(new int[]{1, 2, 2, 1}, new int[]{2, 2})));  
 }  
  
 static int[] intersection(int[] nums1, int[] nums2) {  
 List<Integer> list = new ArrayList<>();  
 int[] count = new int[1000];  
 for (int num : nums1) {  
 count[num]++;  
 }  
 for (int num : nums2) {  
 if (count[num] != 0) {  
 list.add(num);  
 count[num] = 0;  
 }  
 }  
 int[] result = new int[list.size()];  
 for (int i = 0; i < result.length; i++) {  
 result[i] = list.get(i);  
 }  
 return result;  
 }  
}