class Solution {

public List<List<Integer>> threeSum(int[] nums) {

Arrays.sort(nums);

List<List<Integer>> myList = new ArrayList<>();

for(int i=0;i<nums.length;i++){

myList.addAll(twoSum(i,nums));

if (i!=nums.length-1)

while(nums[i]==nums[i+1]) {

i++;

if (i==nums.length-1) break;

}

}

return myList;

}

private List<List<Integer>> twoSum(int start,int[] nums){

int target=-nums[start];

HashMap<Integer,Integer> mymap=new HashMap();

List<List<Integer>> myList = new ArrayList<>();

for (int i=start+1;i<nums.length;i++){

int diff=target-nums[i];

if(mymap.containsKey(diff)&&mymap.get(diff)<i){

List<Integer> temp=Arrays.asList(nums[start],diff,nums[i]);

if(!myList.contains(temp))

myList.add(temp);

}

mymap.put(nums[i],i);

}

return myList;

}

}

class Solution {

public List<List<Integer>> threeSum(int[] nums) {

List<List<Integer>> result = new ArrayList<>();

Arrays.sort(nums);

for(int i = 0; i < nums.length - 2; i++){

if(i == 0 || (i > 0 && nums[i] != nums[i - 1])){

int left = i + 1;

int right = nums.length - 1;

int sum = 0 - nums[i];

while(left < right){

if(nums[left] + nums[right] == sum){

result.add(Arrays.asList(nums[i], nums[left], nums[right]));

while(left < right && nums[left] == nums[left + 1]) left++;

while(left < right && nums[right] == nums[right - 1]) right--;

left++;

right--;

}

else if(nums[left] + nums[right] < sum) left++;

else right--;

}

}

}

return result;

}

}