contains-duplicate

```
package algorithm.array;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

/**
 * https://leetcode.com/problems/contains-duplicate/
 *
 Given an array of integers, find if the array contains any duplicates.
 Your function should return true if any value appears at least twice in th

 * @author xiaobaoqiu  Date: 16-5-22 Time:  下午6:01
 */
public class ContainsDuplicate {
    public static void main(String[] args) {
        int[] nums = new int[] {1,2,3,4};
        //        int[] nums = new int[]{3, 3};
//        System.out.println(containsDuplicate(nums));
        System.out.println(containsDuplicate_1(nums));
    }

    /**
     * Hash
     *
     * 15 ms
     * Your runtime beats 32.32% of java submissions
     */
    public static boolean containsDuplicate(int[] nums) {
        Set<Integer> set = new HashSet<Integer>();
        for (int v : nums) {
            if(set.contains(v)) return true;
            set.add(v);
        }

        return false;
    }

    /**
     * 排序
     *
     * 6 ms
     * Your runtime beats 81.45% of java submissions
     */
    public static boolean containsDuplicate_1(int[] nums) {
        Arrays.sort(nums);
        for (int i=1; i<nums.length; i++) {
            if (nums[i-1] == nums[i]) return true;
        }
        return false;
    }
}
```

contains-duplicate-ii

```
package algorithm.array;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

/**
 * https://leetcode.com/problems/contains-duplicate-ii/
 *
 Given an array of integers and an integer k,
 find out whether there are two distinct indices i and j in the array
 such that nums[i] = nums[j] and the difference between i and j is at most

 * @author xiaobaoqiu  Date: 16-7-4 Time:  下午10:55
 */
public class ContainsDuplicateII {
    public static void main(String[] args) {

    }

    /**
     * 13 ms
     * Your runtime beats 48.61% of java submissions.
     */
    public boolean containsNearbyDuplicate(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap<Integer, Integer>();

        for (int i = 0; i< nums.length; i++) {
            Integer exist = map.get(nums[i]);
            if(exist != null && exist + k >= i) return true;
            map.put(nums[i], i);
        }

        return false;
    }
}
```

intersection-of-two-arrays

```
package algorithm.array;

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

/**
 * https://leetcode.com/problems/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.

 * @author xiaobaoqiu  Date: 16-5-19 Time:  下午10:04
 */
public class IntersectionOfTwoArrays {
    public static void main(String[] args) {
        int[] nums1 = new int[] {1, 2, 2, 1};
        int[] nums2 = new int[] {2, 2};

        int[] ret = intersection(nums1, nums2);
        for (int v : ret) System.out.print(v + " , ");
    }

    /**
     * 6 ms
     */
    public static int[] intersection(int[] nums1, int[] nums2) {
        Set<Integer> set = new HashSet<Integer>(nums1.length);
        for (int v : nums1) set.add(v);

        Set<Integer> common = new HashSet<Integer>();
        for (int v : nums2) if (set.contains(v)) common.add(v);

        int[] ret = new int[common.size()];
        int i = 0;
        for (int v: common) ret[i++] = v;
        return ret;
    }
}
```

intersection-of-two-arrays-ii

```
package algorithm.array;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

/**
 * https://leetcode.com/problems/intersection-of-two-arrays-ii/
 *
 Given two arrays, write a function to compute their intersection.

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

 Note:
 Each element in the result should appear as many times as it shows in both
 The result can be in any order.
 Follow up:
 What if the given array is already sorted? How would you optimize your alg
 What if nums1's size is small compared to num2's size? Which algorithm is
 What if elements of nums2 are stored on disk, and the memory is limited su

 * @author xiaobaoqiu  Date: 16-5-19 Time:  下午10:04
 */
public class IntersectionOfTwoArraysII {
    public static void main(String[] args) {
//        int[] nums1 = new int[] {1, 2, 2, 1};
//        int[] nums2 = new int[] {2, 2};

        int[] nums1 = new int[] {2};
        int[] nums2 = new int[] {2, 2};

        int[] ret = intersection(nums1, nums2);
        for (int v : ret) System.out.print(v + " , ");
    }

    /**
     * 13 ms
     */
    public static int[] intersection(int[] nums1, int[] nums2) {
        Map<Integer, Integer> map = new HashMap<Integer, Integer>();
        for (int v : nums1) {
            if (!map.containsKey(v)) map.put(v, 1);
            else map.put(v, map.get(v) + 1);
        }

        int total = 0;
        Map<Integer, Integer> common = new HashMap<Integer, Integer>();;
        for (int v : nums2) {
            if (map.containsKey(v)) {
                int count = map.get(v);
                Integer cur = common.get(v);
                if (cur == null || cur == 0) {common.put(v, 1); total += 1
                else if (cur < count) {common.put(v, common.get(v) + 1); to
            }
        }

        int[] ret = new int[total];
        int i = 0;
        for (Map.Entry<Integer, Integer> e : common.entrySet()) {
            for (int j = 0; j < e.getValue(); j++) ret[i++] = e.getKey();
        }
        return ret;
    }
}
```

merge-sorted-array

```
package algorithm.array;

import java.util.Arrays;

/**
 * https://leetcode.com/problems/merge-sorted-array/
 *
 Given two sorted integer arrays nums1 and nums2,
 merge nums2 into nums1 as one sorted array.

 Note:
 You may assume that nums1 has enough space
 (size that is greater or equal to m + n)
 to hold additional elements from nums2.
 The number of elements initialized in nums1 and nums2 are m and n respect

 * @author xiaobaoqiu  Date: 16-7-5 Time:  下午10:47
 */
public class MergeSortedArray {
    public static void main(String[] args) {
        int[] a = new int[]{1, 3, 5, 7, 9, 0, 0 , 0, 0};
        int[] b = new int[]{2, 4, 6, 8};
//        merge(a, 5, b, 4);
        merge_1(a, 5, b, 4);
        for (int v : a) System.out.print(v + "-->");
    }

    /**
     * 从大到小排序，只是放的位置从  m+n 开始往前
     * 1 ms
     * Your runtime beats 7.07% of java submissions
     */
    public static void merge(int[] nums1, int m, int[] nums2, int n) {
        int i = m + n - 1;
        m -= 1;
        n -= 1;
        while (m >= 0 && n >= 0) {
            if (nums1[m] > nums2[n]) nums1[i--] = nums1[m--];
            else nums1[i--] = nums2[n--];
        }
        while(m >= 0) nums1[i--] = nums1[m--];
        while(n >= 0) nums1[i--] = nums2[n--];
    }

    /**
     * 从大到小排序，只是放的位置从  m+n-1 开始往前
     * 1 ms
     * Your runtime beats 7.07% of java submissions
     */
    public static void merge_1(int[] nums1, int m, int[] nums2, int n) {
        int i = m + n - 1;
        --m;
        --n;
        while (m >= 0 || n >= 0) {
            if (m < 0) nums1[i--] = nums2[n--];
            else if (n < 0) nums1[i--] = nums1[m--];
            else if (nums1[m] > nums2[n]) nums1[i--] = nums1[m--];
            else nums1[i--] = nums2[n--];
        }
    }
}
```

missing-number

```
package algorithm.array;

/**
 * https://leetcode.com/problems/missing-number/
 * <p/>
 * Given an array containing n distinct numbers taken from 0, 1, 2, ..., n,
 * find the one that is missing from the array.
 * <p/>
 * For example,
 * Given nums = [0, 1, 3] return 2.
 * <p/>
 * Note:
 * Your algorithm should run in linear runtime complexity.
 * Could you implement it using only constant extra space complexity?
 *
 * @author xiaobaoqiu  Date: 16-5-23 Time:  下午11:15
 */
public class MissingNumber {
    public static void main(String[] args) {
        int[] nums = new int[]{0, 1, 3};
        System.out.println(missingNumber(nums));
    }

    /**
     * 思路：求和，再减去数组中的每个数
     *
     * 1 ms
     * Your runtime beats 37.49% of java submissions.
     */
    public static int missingNumber(int[] nums) {
        int sum = (nums.length + 1) * nums.length / 2;
        for (int v : nums) sum -= v;
        return sum;
    }
}
```

move-zeroes

```
package algorithm.array;

/**
 * https://leetcode.com/problems/move-zeroes/
 *
 Given an array nums, write a function to move all 0's to the end of it wh

 For example, given nums = [0, 1, 0, 3, 12], after calling your function, n

 Note:
 You must do this in-place without making a copy of the array.
 Minimize the total number of operations.

 * @author xiaobaoqiu  Date: 16-5-17 Time:  下午9:48
 */
public class MoveZeroes {
    public static void main(String[] args) {
        int[] num = new int[]{0, 1, 0, 3, 12};
        for (int v : num) System.out.print(v + " , ");
        System.out.println();

        moveZeroes(num);
        for (int v : num) System.out.print(v + " , ");
    }

    /**
     * 思路： 将非0直接放到开始,末尾补0
     *
     * 1 ms
     * Your runtime beats 23.38% of java submissions
     */
    public static void moveZeroes(int[] nums) {
        if (nums == null || nums.length <= 1) return;

        int cur = 0;
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] != 0) nums[cur++] = nums[i];
        }
        while (cur < nums.length) {
            nums[cur++] = 0;
        }
    }
}
```

pascals-triangle

```
package algorithm.array;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.LinkedList;
import java.util.List;

/**
 * https://leetcode.com/problems/pascals-triangle/
 *
 Given numRows, generate the first numRows of Pascal's triangle.

 For example, given numRows = 5,
 Return

 [
 [1],
 [1,1],
 [1,2,1],
 [1,3,3,1],
 [1,4,6,4,1]
 ]

 * @author xiaobaoqiu  Date: 16-6-6 Time:  下午10:07
 */
public class PascalsTriangle {
    public static void main(String[] args) {
        List<List<Integer>> ret = generate(5);
        System.out.println(ret);
    }

    /**
     * 1 ms
     * Your runtime beats 40.47% of java submissions
     */
    public static List<List<Integer>> generate(int numRows) {
        if (numRows <= 0) return Collections.emptyList();

        List<List<Integer>> ret = new ArrayList<List<Integer>>(numRows);
        for (int i = 0; i < numRows; i++) {
            Integer[] temp = new Integer[i + 1];
            temp[0] = 1;
            if (i > 0) {
                temp[i] = 1;
            }
            for (int j = 1; j < i; j++) {
                temp[j] = ret.get(i - 1).get(j - 1) + ret.get(i - 1).get(j
            }
            ret.add(Arrays.asList(temp));
        }
        return ret;
    }
}
```

pascals-triangle-ii

```
package algorithm.array;

import java.util.Arrays;
import java.util.List;

/**
 * https://leetcode.com/problems/pascals-triangle-ii/
 *
 Given an index k, return the kth row of the Pascal's triangle.

 For example, given k = 3,
 Return [1,3,3,1]

 Note:
 Could you optimize your algorithm to use only O(k) extra space?

 * @author xiaobaoqiu  Date: 16-6-28 Time:  下午10:34
 */
public class PascalsTriangleII {
    public static void main(String[] args) {
        int n = 5;
        System.out.println(getRow(0));
        System.out.println(getRow(1));
        System.out.println(getRow(2));
        System.out.println(getRow(3));
        System.out.println(getRow(4));
        System.out.println(getRow(5));
    }

    /**
     * n 个
     * f[i][j] = f[i - 1][j - 1] + f[i - 1][j]
     *
     * 2 ms
     * Your runtime beats 75.23% of java submissions
     */
    public static List<Integer> getRow(int rowIndex) {
        Integer[] f = new Integer[rowIndex + 1];
        for (int i = 0; i <= rowIndex; i++) {
            f[0] = 1;
            if (i > 0) {
                f[i] = 1;
            }
            int pre = 1, next;
            for (int j = 1; j < i; j++) {
                next = f[j];
                f[j] = pre + f[j];
                pre = next;
            }
        }

        return Arrays.asList(f);
    }
}
```

plus-one

```
package algorithm.array;

/**
 * https://leetcode.com/problems/plus-one/
 * <p/>
 * Given a non-negative number represented as an array of digits,
 * plus one to the number.
 * <p/>
 * The digits are stored such that the most significant digit is
 * at the head of the list.
 *
 * @author xiaobaoqiu  Date: 16-6-1 Time:  下午11:06
 */
public class PlusOne {
    public static void main(String[] args) {
//        int[] digit = new int[]{9, 9, 9, 9};
        int[] digit = new int[]{9, 9, 9, 8};
        for (int v : digit) System.out.print(v + " , ");
        System.out.println();

        digit = plusOne(digit);
        for (int v : digit) System.out.print(v + " , ");
        System.out.println();
    }

    /**
     * 0 ms
     * Your runtime beats 36.50% of java submissions
     */
    public static int[] plusOne(int[] digits) {
        int more = 1;
        for (int i = digits.length - 1; i >= 0; i--) {
            digits[i] += more;
            more = 0;
            if (digits[i] > 9) {
                digits[i] -= 10;
                more = 1;
            }
        }
        if (more <= 0) return digits;
        int[] ret = new int[digits.length + 1];
        ret[0] = more;
        System.arraycopy(digits, 0, ret, 1, digits.length);
        return ret;
    }
}
```

product-of-array-except-self

```
package algorithm.array;

/**
 * https://leetcode.com/problems/product-of-array-except-self/
 *
 Given an array of n integers where n > 1, nums, return an array output suc

 Solve it without division and in O(n).

 For example, given [1,2,3,4], return [24,12,8,6].

 Follow up:
 Could you solve it with constant space complexity? (Note: The output array

 * @author xiaobaoqiu  Date: 16-5-18 Time:  下午9:26
 */
public class ProductOfArrayExceptSelf {
    public static void main(String[] args) {
        int[] nums = new int[]{1,2,3,4};
//        int[] ret = productExceptSelf(nums);
        int[] ret = productExceptSelf_1(nums);

        for (int v : ret) System.out.print(v + " , ");
    }

    /**
     * 两个数组,A[i]存放的是 nums[i] 之前的元素的乘积,B[i]存放的是 nums[i] 之后的
     * 假设结果为ret, 则ret[i] = A[i] * B[i]
     *
     * 3 ms
     * Your runtime beats 11.83% of javasubmissions
     */
    public static int[] productExceptSelf(int[] nums) {
        int[] A = new int[nums.length];
        A[0] = 1;
        int[] B = new int[nums.length];
        B[nums.length - 1] = 1;

        for (int i = 1; i < nums.length; i++) A[i] = A[i - 1] * nums[i - 1]
        for (int i = nums.length - 2; i >= 0; i--) B[i] = B[i + 1] * nums[i

        for (int i = 0; i < nums.length; i++) A[i] *= B[i];
        return A;
    }

    /**
     * 在上面的基础上, 我们可以减少一个数组, 直接再原始数据基础上修改
     * 根据题目疑似, 因为输出数组不算在空间复杂度内, 因此空间复杂度为常量
     *
     * 3 ms
     * Your runtime beats 11.83% of java submissions.
     */
    public static int[] productExceptSelf_1(int[] nums) {
        int[] A = new int[nums.length];
        A[0] = 1;

        for (int i = 1; i < nums.length; i++) A[i] = A[i - 1] * nums[i - 1]

        for (int i = nums.length - 2; i >= 0; i--) {
            nums[i] *= nums[i + 1];
            A[i] = A[i] * nums[i + 1];
        }

        return A;
    }

    public static int[] productExceptSelf_2(int[] nums) {
        int[] A = new int[nums.length];
        A[0] = 1;

        for (int i = 1; i < nums.length; i++) A[i] = A[i - 1] * nums[i - 1]

        for (int i = nums.length - 2; i >= 0; i--) {
            nums[i] *= nums[i + 1];
            A[i] = A[i] * nums[i + 1];
        }

        return A;
    }
}
```

remove-duplicates-from-sorted-array

```
package algorithm.array;

/**
 * https://leetcode.com/problems/remove-duplicates-from-sorted-array/
 *
 Given a sorted array, remove the duplicates in place such that each elemen

 Do not allocate extra space for another array, you must do this in place w

 For example,
 Given input array nums = [1,1,2],

 Your function should return length = 2, with the first two elements of nur
 * @author xiaobaoqiu  Date: 16-6-27 Time:  下午10:31
 */
public class RemoveDuplicatesFromSortedArray {
    public static void main(String[] args) {
        int[] nums = new int[]{1, 2, 3, 3, 3, 4, 5, 5};
        int len = removeDuplicates(nums);
        System.out.println(len);
        for (int v : nums) System.out.print(v + " , ");
    }

    /**
     * 1 ms
     * Your runtime beats 51.76% of java submissions
     */
    public static int removeDuplicates(int[] nums) {
        if (nums == null || nums.length == 0) return 0;

        int pos = 0;
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] != nums[pos]) {
                pos++;
                nums[pos] = nums[i];
            }
        }
        return pos + 1;
    }
}
```

remove-element

```
package algorithm.array;

/**
 * https://leetcode.com/problems/remove-element/
 *
 Given an array and a value, remove all instances of that value in place ar

 Do not allocate extra space for another array, you must do this in place w

 The order of elements can be changed. It doesn't matter what you leave bey

 Example:
 Given input array nums = [3,2,2,3], val = 3

 Your function should return length = 2, with the first two elements of nur

 * @author xiaobaoqiu  Date: 16-6-1 Time:  下午10:22
 */
public class RemoveElement {
    public static void main(String[] args) {
        int[] nums = new int[]{3,2,2,3};
        System.out.println(removeElement(nums, 3));
    }

    /**
     * 1 ms
     * Your runtime beats 3.78% of java submissions
     */
    public static int removeElement(int[] nums, int val) {
        int pos = 0;
        for (int i =0; i< nums.length; i++) {
            if (nums[i] != val) nums[pos++] = nums[i];
        }
        return pos;
    }
}
```

rotate-array

```
package algorithm.array;

/**
 * https://leetcode.com/problems/rotate-array/
 *
 Rotate an array of n elements to the right by k steps.

 For example, with n = 7 and k = 3,
 the array [1,2,3,4,5,6,7] is rotated to [5,6,7,1,2,3,4].

 Note:
 Try to come up as many solutions as you can,
 there are at least 3 different ways to solve this problem.

 * @author xiaobaoqiu  Date: 16-7-8 Time:  下午11:56
 */
public class RotateArray {
    public static void main(String[] args) {
//        int[] nums = new int[]{1,2,3,4,5,6,7};
//        int k = 3;
        int[] nums = new int[]{1,2,3,4,5,6};
        int k = 2;
        for (int v : nums) System.out.print(v + " --> ");
        System.out.println();
//        rotate(nums, k);
//        rotate_1(nums, k);
        rotate_2(nums, k);
        for (int v : nums) System.out.print(v + " --> ");
        System.out.println();
    }

    /**
     * 新开一个数组
     * 时间:O(n)
     * 空间:O(1)
     *
     * 1 ms
     * Your runtime beats 13.71% of java submissions.
     */
    public static void rotate(int[] nums, int k) {
        k %= nums.length;
        if (k == 0) return;

        int[] temp = new int[nums.length];
        for (int i = 0; i < nums.length; i++) {
            int pos = i + k;
            if (pos >= nums.length) pos -= nums.length;
            temp[pos] = nums[i];
        }
        System.arraycopy(temp, 0, nums, 0, nums.length);
    }

    /**
     * 模拟一个一个移动
     */
    public static void rotate_1(int[] nums, int k) {
        k %= nums.length;
        if (k == 0) return;
        //i don't want to do it
    }

    /**
     * 两部分翻转
     * (1).翻转后 k 个
     * (2).翻转前 n-k 个
     * (3).翻转全部
     *
     * 时间:O(n)
     * 空间:O(1)
     *
     * 1 ms
     * Your runtime beats 13.27% of javasubmissions
     */
    public static void rotate_2(int[] nums, int k) {
        k %= nums.length;
        if (k == 0) return;

        reverse(nums, 0, nums.length - k - 1);
        reverse(nums, nums.length - k, nums.length - 1);
        reverse(nums, 0, nums.length - 1);
    }
    private static void reverse(int[] nums, int beg, int end) {
        if (beg >= end) return;
        int temp;
        for (int i = beg, j = end; i < j; i++, j--) {
            temp = nums[i];
            nums[i] = nums[j];
            nums[j] = temp;
        }
    }
}
```

two-sum

```
package algorithm.array;

import java.util.Arrays;

/**
 * https://leetcode.com/problems/two-sum/
 *
 Given an array of integers, return indices of the two numbers
 such that they add up to a specific target.

 You may assume that each input would have exactly one solution.

 Example:
 Given nums = [2, 7, 11, 15], target = 9,

 Because nums[0] + nums[1] = 2 + 7 = 9,
 return [0, 1].

 UPDATE (2016/2/13):
 The return format had been changed to zero-based indices.
 Please read the above updated description carefully.

 * @author xiaobaoqiu  Date: 16-7-8 Time:  下午11:11
 */
public class TwoSum {
    public static void main(String[] args) {
        int[] nums = new int[] {2, 7, 11, 15};
//        int[] ret = twoSum(nums, 17);
        int[] ret = twoSum_1(nums, 17);
        for (int v : ret) System.out.print(v + " --> ");
    }

    /**
     * 暴力
     * 时间：O(n^2)
     * 空间：O(1)
     *
     * 42 ms
     * Your runtime beats 23.73% of java submissions
     */
    public static int[] twoSum(int[] nums, int target) {
        int[] ret = new int[2];
        for (int i = 0; i<nums.length - 1; i++) {
            for (int j = i+1; j < nums.length; j++)
                if (nums[i] + nums[j] == target) {
                    ret[0] = i;
                    ret[1] = j;
                    return ret;
                }
        }
        return ret;
    }

    /**
```

```
     * 排序 + 两端搜索
     * 时间：O(nlogn + n)
     * 空间：O(n)
     *
     * 10 ms
     * Your runtime beats 38.72% of java submissions
     */
    public static int[] twoSum_1(int[] nums, int target) {
        Pair[] pairs = new Pair[nums.length];
        for (int i = 0; i < nums.length; i++) {
            pairs[i] = new Pair(nums[i], i);
        }
        Arrays.sort(pairs);

        int[] ret = new int[2];
        for (int i = 0, j = pairs.length - 1; i < j; ) {
            int t = pairs[i].value + pairs[j].value;
            if (t > target) --j;
            else if (t < target) ++i;
            else {
                ret[0] = pairs[i].index;
                ret[1] = pairs[j].index;
                break;
            }
        }
        return ret;
    }

    public static class Pair implements Comparable<Pair> {
        private int value;
        private int index;

        public Pair(int value, int index) {
            this.value = value;
            this.index = index;
        }

        public int compareTo(Pair o) {
            return this.value - o.value;
        }
    }
}
```