**1 (刘炜晨)**

#66

### Problem: Plus One

**Description:**

Given a non-negative number represented as an array of digits, plus one to the number.

The digits are stored such that the most significant digit is at the head of the list.

**2 （张晨阳）**

#119

**Problem:** Pascal's Triangle II

**Description:**

Given an index *k*, return the *k*th 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?

**3 （李虹桥）**

#189

### Problem: Rotate Array

### Description:

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.

**4 （杨楠）**

#217

### Problem: Contains Duplicates

**Description:**

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 the array, and it should return false if every element is distinct.

**5 （王卓思）**

#55

**Problem:** Jump Game

**Description:**

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Determine if you are able to reach the last index.

For example:  
A = [2,3,1,1,4], return true.

A = [3,2,1,0,4], return false.

**6 （梁嘉义）**

#1

### Problem: Two Sum

**Description:**

Given an array of integers, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.

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

**Input:** numbers={2, 7, 11, 15}, target=9  
**Output:** index1=1, index2=2

**7 （王开宇）**

#35

### Problem: Search Insert Position

**Description:**

Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may assume no duplicates in the array.

Here are few examples.  
[1,3,5,6], 5 → 2  
[1,3,5,6], 2 → 1  
[1,3,5,6], 7 → 4  
[1,3,5,6], 0 → 0

### 8 （陈嘉曦）

#62

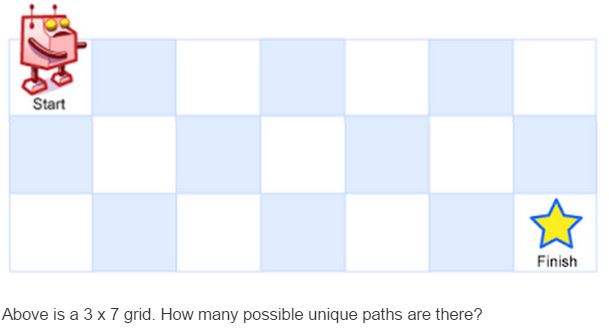
### Problem: Unique Paths

**Description:**

A robot is located at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

How many possible unique paths are there?



**Note:** *m* and *n* will be at most 100.

**9 （黄思琦）**

#81

### Problem: Search in Rotated Sorted Array II

### Description:

Follow up for "Search in Rotated Sorted Array":  
What if *duplicates* are allowed?

Would this affect the run-time complexity? How and why?

Write a function to determine if a given target is in the array.

**10 （邓宇）**

#31

### Problem: Next Permutation

**Description:**

Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numbers.

If such arrangement is not possible, it must rearrange it as the lowest possible order (ie, sorted in ascending order).

The replacement must be in-place, do not allocate extra memory.

Here are some examples. Inputs are in the left-hand column and its corresponding outputs are in the right-hand column.  
1,2,3 → 1,3,2  
3,2,1 → 1,2,3  
1,1,5 → 1,5,1