1 Two Sum
Easy

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, and you may not use the *same* element twice.

Example:

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

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

2 Add Two Numbers
Medium

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order** and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example

```
Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.
```

3 Longest Substring Without Repeating Characters
Medium

Given a string, find the length of the **longest substring** without repeating characters.

Examples:

Given `"abcabcbb"`, the answer is `"abc"`, which the length is 3.

Given `"bbbbb"`, the answer is `"b"`, with the length of 1.

Given `"pwwkew"`, the answer is `"wke"`, with the length of 3. Note that the answer must be a **substring**, `"pwke"` is a *subsequence* and not a substring.

4 Median of Two Sorted Arrays
Hard

There are two sorted arrays **nums1** and **nums2** of size m and n respectively.

Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

Example 1:

```
nums1 = [1, 3]
nums2 = [2]

The median is 2.0
```

Example 2:

```
nums1 = [1, 2]
nums2 = [3, 4]

The median is (2 + 3)/2 = 2.5
```

5 Longest Palindromic Substring

Medium

Given a string **s**, find the longest palindromic substring in **s**. You may assume that the maximum length of **s** is 1000.

Example 1:

```
Input: "babad"
Output: "bab"
Note: "aba" is also a valid answer.
```

Example 2:

```
Input: "cbbd"
Output: "bb"
```

6 ZigZag Conversion

Medium

The string `"PAYPALISHIRING"` is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

```
P   A   H   N
A P L S I I G
Y   I   R
```

And then read line by line: `"PAHNAPLSIIGYIR"`

Write the code that will take a string and make this conversion given a number of rows:

```
string convert(string s, int numRows);
```

Example 1:

```
Input: s = "PAYPALISHIRING", numRows = 3
Output: "PAHNAPLSIIGYIR"
```

Example 2:

```
Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
Explanation:

P     I    N
A   L S  I G
Y A   H R
P     I
```

7 Reverse Integer

Easy

Given a 32-bit signed integer, reverse digits of an integer.

Example 1:

```
Input: 123
Output: 321
```

Example 2:

```
Input: -123
Output: -321
```

Example 3:

```
Input: 120
Output: 21
```

Note:

Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: $[-2^{31},\ 2^{31} - 1]$. For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.

8 String to Integer (atoi)

Medium

Implement `atoi` which converts a string to an integer.

The function first discards as many whitespace characters as necessary until the first non-whitespace character is found. Then, starting from this character, takes an optional initial plus or minus sign followed by as many numerical digits as possible, and interprets them as a numerical value.

The string can contain additional characters after those that form the integral number, which are ignored and have no effect on the behavior of this function.

If the first sequence of non-whitespace characters in str is not a valid integral number, or if no such sequence exists because either str is empty or it contains only whitespace characters, no conversion is performed.

If no valid conversion could be performed, a zero value is returned.

Note:

- Only the space character `' '` is considered as whitespace character.
- Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: $[-2^{31}, 2^{31} - 1]$. If the numerical value is out of the range of representable values, INT_MAX ($2^{31} - 1$) or INT_MIN (-2^{31}) is returned.

Example 1:

```
Input: "42"
Output: 42
```

Example 2:

```
Input: "   -42"
Output: -42
Explanation: The first non-whitespace character is '-', which is the minus sign.
             Then take as many numerical digits as possible, which gets 42.
```

Example 3:

```
Input: "4193 with words"
Output: 4193
Explanation: Conversion stops at digit '3' as the next character is not a numerical digit.
```

Example 4:

```
Input: "words and 987"
Output: 0
Explanation: The first non-whitespace character is 'w', which is not a numerical
             digit or a +/- sign. Therefore no valid conversion could be performed.
```

Example 5:

```
Input: "-91283472332"
Output: -2147483648
Explanation: The number "-91283472332" is out of the range of a 32-bit signed integer.
             Thefore INT_MIN (-2^31) is returned.
```

9 Palindrome Number

Easy

Determine whether an integer is a palindrome. An integer is a palindrome when it reads the same backward as forward.

Example 1:

```
Input: 121
Output: true
```

Example 2:

```
Input: -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a
```

Example 3:

```
Input: 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.
```

Follow up:

Coud you solve it without converting the integer to a string?

10 Regular Expression Matching

Hard

Given an input string (`s`) and a pattern (`p`), implement regular expression matching with support for `'.'` and `'*'`.

```
'.' Matches any single character.
'*' Matches zero or more of the preceding element.
```

The matching should cover the **entire** input string (not partial).

Note:

- `s` could be empty and contains only lowercase letters `a-z`.
- `p` could be empty and contains only lowercase letters `a-z`, and characters like `.` or `*`.

Example 1:

```
Input:
s = "aa"
p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input:
s = "aa"
p = "a*"
Output: true
Explanation: '*' means zero or more of the precedeng element, 'a'. Therefore, by repeating 'a' once, it bec
```

Example 3:

```
Input:
s = "ab"
p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".
```

Example 4:

```
Input:
s = "aab"
p = "c*a*b"
Output: true
Explanation: c can be repeated 0 times, a can be repeated 1 time. Therefore it matches "aab".
```

Example 5:

```
Input:
s = "mississippi"
p = "mis*is*p*."
Output: false
```

11 Container With Most Water

Medium

Given n non-negative integers a_1, a_2, ..., a_n, where each represents a point at coordinate (i, a_i). n vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

12 Integer to Roman

Medium

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

```
Symbol       Value
I             1
V             5
X             10
L             50
C             100
D             500
M             1000
```

For example, two is written as `II` in Roman numeral, just two one's added together. Twelve is written as, `XII`, which is simply `X` + `II`. The number twenty seven is written as `XXVII`, which is `XX` + `V` + `II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral. Input is guaranteed to be within the range from 1 to 3999.

Example 1:

```
Input: 3
Output: "III"
```

Example 2:

```
Input: 4
Output: "IV"
```

Example 3:

```
Input: 9
Output: "IX"
```

Example 4:

```
Input: 58
Output: "LVIII"
Explanation: C = 100, L = 50, XXX = 30 and III = 3.
```

Example 5:

```
Input: 1994
Output: "MCMXCIV"
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

13 Roman to Integer

Easy

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, two is written as `II` in Roman numeral, just two one's added together. Twelve is written as, `XII`, which is simply `X` + `II`. The number twenty seven is written as `XXVII`, which is `XX` + `V` + `II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer. Input is guaranteed to be within the range from 1 to 3999.

Example 1:

```
Input: "III"
Output: 3
```

Example 2:

```
Input: "IV"
Output: 4
```

Example 3:

```
Input: "IX"
Output: 9
```

Example 4:

```
Input: "LVIII"
Output: 58
Explanation: C = 100, L = 50, XXX = 30 and III = 3.
```

Example 5:

```
Input: "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

14 Longest Common Prefix

Easy

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string `""`.

Example 1:

```
Input: ["flower","flow","flight"]
Output: "fl"
```

Example 2:

```
Input: ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.
```

Note:

All given inputs are in lowercase letters `a-z`.

15 3Sum

Medium

Given an array `nums` of *n* integers, are there elements *a*, *b*, *c* in `nums` such that *a* + *b* + *c* = 0? Find all unique triplets in the array which gives the sum of zero.

Note:

The solution set must not contain duplicate triplets.

Example:

```
Given array nums = [-1, 0, 1, 2, -1, -4],

A solution set is:
[
  [-1, 0, 1],
  [-1, -1, 2]
]
```

16 3Sum Closest

Medium

Given an array `nums` of *n* integers and an integer `target`, find three integers in `nums` such that the sum is closest to `target`. Return the sum of the three integers. You may assume that each input would have exactly one solution.

Example:

```
Given array nums = [-1, 2, 1, -4], and target = 1.

The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

17 Letter Combinations of a Phone Number

Medium

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example:

```
Input: "23"
Output: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].
```

Note:

Although the above answer is in lexicographical order, your answer could be in any order you want.

18 4Sum

Medium

Given an array `nums` of *n* integers and an integer `target`, are there elements *a*, *b*, *c*, and *d* in `nums` such that *a* + *b* + *c* + *d* = `target`? Find all unique quadruplets in the array which gives the sum of `target`.

Note:

The solution set must not contain duplicate quadruplets.

Example:

```
Given array nums = [1, 0, -1, 0, -2, 2], and target = 0.

A solution set is:
[
  [-1,  0, 0, 1],
  [-2, -1, 1, 2],
  [-2,  0, 0, 2]
]
```

19 Remove Nth Node From End of List

Medium

Given a linked list, remove the *n*-th node from the end of list and return its head.

Example:

```
Given linked list: 1->2->3->4->5, and n = 2.

After removing the second node from the end, the linked list becomes 1->2->3->5.
```

Note:

Given *n* will always be valid.

Follow up:

Could you do this in one pass?

20 Valid Parentheses
Easy

Given a string containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

Note that an empty string is also considered valid.

Example 1:

```
Input: "()"
Output: true
```

Example 2:

```
Input: "()[]{}"
Output: true
```

Example 3:

```
Input: "(]"
Output: false
```

Example 4:

```
Input: "([)]"
Output: false
```

Example 5:

```
Input: "{[]}"
Output: true
```

21 Merge Two Sorted Lists
Easy

Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists.

Example:

```
Input: 1->2->4, 1->3->4
Output: 1->1->2->3->4->4
```

22 Generate Parentheses
Medium

Given *n* pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

For example, given *n* = 3, a solution set is:

```
[
  "((()))",
  "(()())",
  "(())()",
  "()(())",
  "()()()"
]
```

23 Merge k Sorted Lists
Hard

Merge *k* sorted linked lists and return it as one sorted list. Analyze and describe its complexity.

Example:

```
Input:
[
  1->4->5,
  1->3->4,
  2->6
]
Output: 1->1->2->3->4->4->5->6
```

24 Swap Nodes in Pairs
Medium

Given a linked list, swap every two adjacent nodes and return its head.

Example:

```
Given 1->2->3->4, you should return the list as 2->1->4->3.
```

Note:

- Your algorithm should use only constant extra space.
- You may **not** modify the values in the list's nodes, only nodes itself may be changed.

25 Reverse Nodes in k-Group

Hard

Given a linked list, reverse the nodes of a linked list *k* at a time and return its modified list.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of *k* then left-out nodes in the end should remain as it is.

Example:

Given this linked list: `1->2->3->4->5`

For *k* = 2, you should return: `2->1->4->3->5`

For *k* = 3, you should return: `3->2->1->4->5`

Note:

- Only constant extra memory is allowed.
- You may not alter the values in the list's nodes, only nodes itself may be changed.

26 Remove Duplicates from Sorted Array

Easy

Given a sorted array *nums*, remove the duplicates **in-place** such that each element appear only *once* and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array in-place** with O(1) extra memory.

Example 1:

```
Given nums = [1,1,2],

Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively.

It doesn't matter what you leave beyond the returned length.
```

Example 2:

```
Given nums = [0,0,1,1,1,2,2,3,3,4],

Your function should return length = 5, with the first five elements of nums being modified to 0, 1, 2, 3,

It doesn't matter what values are set beyond the returned length.
```

Clarification:

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

```
// nums is passed in by reference. (i.e., without making a copy)
int len = removeDuplicates(nums);

// any modification to nums in your function would be known by the caller.
// using the length returned by your function, it prints the first len elements.
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

27 Remove Element

Easy

Given an array *nums* and a value *val*, remove all instances of that value **in-place** and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array in-place** with O(1) extra memory.

The order of elements can be changed. It doesn't matter what you leave beyond the new length.

Example 1:

```
Given nums = [3,2,2,3], val = 3,

Your function should return length = 2, with the first two elements of nums being 2.

It doesn't matter what you leave beyond the returned length.
```

Example 2:

```
Given nums = [0,1,2,2,3,0,4,2], val = 2,

Your function should return length = 5, with the first five elements of nums containing 0, 1, 3, 0, and 4.

Note that the order of those five elements can be arbitrary.

It doesn't matter what values are set beyond the returned length.
```

Clarification:

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

```
// nums is passed in by reference. (i.e., without making a copy)
int len = removeElement(nums, val);

// any modification to nums in your function would be known by the caller.
// using the length returned by your function, it prints the first len elements.
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

28 Implement strStr()

Easy

Implement strStr().

Return the index of the first occurrence of needle in haystack, or **-1** if needle is not part of haystack.

Example 1:

```
Input: haystack = "hello", needle = "ll"
Output: 2
```

Example 2:

```
Input: haystack = "aaaaa", needle = "bba"
Output: -1
```

Clarification:

What should we return when `needle` is an empty string? This is a great question to ask during an interview.

For the purpose of this problem, we will return 0 when `needle` is an empty string. This is consistent to C's strstr() and Java's indexOf().

29 Divide Two Integers
Medium

Given two integers `dividend` and `divisor`, divide two integers without using multiplication, division and mod operator.

Return the quotient after dividing `dividend` by `divisor`.

The integer division should truncate toward zero.

Example 1:

```
Input: dividend = 10, divisor = 3
Output: 3
```

Example 2:

```
Input: dividend = 7, divisor = -3
Output: -2
```

Note:

- Both dividend and divisor will be 32-bit signed integers.
- The divisor will never be 0.
- Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: $[-2^{31}, 2^{31} - 1]$. For the purpose of this problem, assume that your function returns $2^{31} - 1$ when the division result overflows.

30 Substring with Concatenation of All Words
Hard

You are given a string, **s**, and a list of words, **words**, that are all of the same length. Find all starting indices of substring(s) in **s** that is a concatenation of each word in **words** exactly once and without any intervening characters.

Example 1:

```
Input:
  s = "barfoothefoobarman",
  words = ["foo","bar"]
Output: [0,9]
Explanation: Substrings starting at index 0 and 9 are "barfoor" and "foobar" respectively.
The output order does not matter, returning [9,0] is fine too.
```

Example 2:

```
Input:
  s = "wordgoodstudentgoodword",
  words = ["word","student"]
Output: []
```

31 Next Permutation
Medium

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** and use only constant 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`

32 Longest Valid Parentheses
Hard

Given a string containing just the characters `'('` and `')'`, find the length of the longest valid (well-formed) parentheses substring.

Example 1:

```
Input: "(()"
Output: 2
Explanation: The longest valid parentheses substring is "()"
```

Example 2:

```
Input: ")()())"
Output: 4
Explanation: The longest valid parentheses substring is "()()"
```

33 Search in Rotated Sorted Array
Medium

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

You are given a target value to search. If found in the array return its index, otherwise return `-1`.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

34 Search for a Range

Medium

Given an array of integers `nums` sorted in ascending order, find the starting and ending position of a given `target` value.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

If the target is not found in the array, return `[-1, -1]`.

Example 1:

```
Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
```

Example 2:

```
Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
```

35 Search Insert Position

Easy

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.

Example 1:

```
Input: [1,3,5,6], 5
Output: 2
```

Example 2:

```
Input: [1,3,5,6], 2
Output: 1
```

Example 3:

```
Input: [1,3,5,6], 7
Output: 4
```

Example 4:

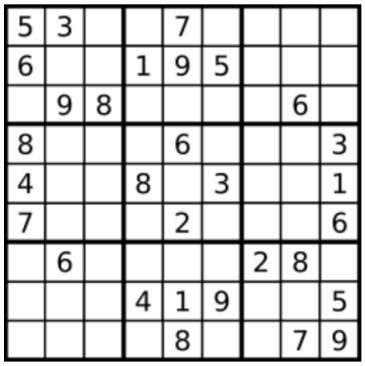
```
Input: [1,3,5,6], 0
Output: 0
```

36 Valid Sudoku

Medium

Determine if a 9x9 Sudoku board is valid. Only the filled cells need to be validated **according to the following rules**:

1. Each row must contain the digits `1-9` without repetition.
2. Each column must contain the digits `1-9` without repetition.
3. Each of the 9 `3x3` sub-boxes of the grid must contain the digits `1-9` without repetition.



A partially filled sudoku which is valid.

The Sudoku board could be partially filled, where empty cells are filled with the character `'.'`.

Example 1:

```
Input:
[
  ["5","3",".",".","7",".",".",".","."],
  ["6",".",".","1","9","5",".",".","."],
  [".","9","8",".",".",".",".","6","."],
  ["8",".",".",".","6",".",".",".","3"],
  ["4",".",".","8",".","3",".",".","1"],
  ["7",".",".",".","2",".",".",".","6"],
  [".","6",".",".",".",".","2","8","."],
  [".",".",".","4","1","9",".",".","5"],
  [".",".",".",".","8",".",".","7","9"]
]
Output: true
```

Example 2:

```
Input:
[
  ["8","3",".",".","7",".",".",".","."],
  ["6",".",".","1","9","5",".",".","."],
  [".","9","8",".",".",".",".","6","."],
  ["8",".",".",".","6",".",".",".","3"],
  ["4",".",".","8",".","3",".",".","1"],
  ["7",".",".",".",".","2",".",".",".","6"],
  [".","6",".",".",".",".",".","2","8","."],
  [".",".",".","4","1","9",".",".","5"],
  [".",".",".",".","8",".",".","7","9"]
]
Output: false
Explanation: Same as Example 1, except with the 5 in the top left corner being
    modified to 8. Since there are two 8's in the top left 3x3 sub-box, it is invalid.
```

Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.
- The given board contain only digits `1-9` and the character `'.'`.
- The given board size is always `9x9`.

37 Sudoku Solver

Hard

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

1. Each of the digits `1-9` must occur exactly once in each row.
2. Each of the digits `1-9` must occur exactly once in each column.
3. Each of the the digits `1-9` must occur exactly once in each of the 9 `3x3` sub-boxes of the grid.

Empty cells are indicated by the character `'.'`.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

A sudoku puzzle...

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

...and its solution numbers marked in red.

Note:

- The given board contain only digits `1-9` and the character `'.'`.
- You may assume that the given Sudoku puzzle will have a single unique solution.
- The given board size is always `9x9`.

38 Count and Say

Easy

The count-and-say sequence is the sequence of integers with the first five terms as following:

```
1.     1
2.     11
3.     21
4.     1211
5.     111221
```

`1` is read off as `"one 1"` or `11`.

`11` is read off as `"two 1s"` or `21`.

`21` is read off as `"one 2`, then `one 1"` or `1211`.

Given an integer *n*, generate the n^{th} term of the count-and-say sequence.

Note: Each term of the sequence of integers will be represented as a string.

Example 1:

```
Input: 1
Output: "1"
```

Example 2:

```
Input: 4
Output: "1211"
```

39 Combination Sum

Medium

Given a **set** of candidate numbers (`candidates`) **(without duplicates)** and a target number (`target`), find all unique combinations in `candidates` where the candidate numbers sums to `target`.

The **same** repeated number may be chosen from `candidates` unlimited number of times.

Note:

- All numbers (including `target`) will be positive integers.
- The solution set must not contain duplicate combinations.

Example 1:

```
Input: candidates = [2,3,6,7], target = 7,
A solution set is:
[
  [7],
  [2,2,3]
]
```

Example 2:

```
Input: candidates = [2,3,5], target = 8,
A solution set is:
[
  [2,2,2,2],
  [2,3,3],
  [3,5]
]
```

40 Combination Sum II

Medium

Given a collection of candidate numbers (`candidates`) and a target number (`target`), find all unique combinations in `candidates` where the candidate numbers sums to `target` .

Each number in `candidates` may only be used **once** in the combination.

Note:

- All numbers (including `target`) will be positive integers.
- The solution set must not contain duplicate combinations.

Example 1:

```
Input: candidates = [10,1,2,7,6,1,5], target = 8,
A solution set is:
[
  [1, 7],
  [1, 2, 5],
  [2, 6],
  [1, 1, 6]
]
```

Example 2:

```
Input: candidates = [2,5,2,1,2], target = 5,
A solution set is:
[
  [1,2,2],
  [5]
]
```

41 First Missing Positive

Hard

Given an unsorted integer array, find the smallest missing positive integer.

Example 1:

```
Input: [1,2,0]
Output: 3
```

Example 2:

```
Input: [3,4,-1,1]
Output: 2
```

Example 3:

```
Input: [7,8,9,11,12]
Output: 1
```

Note:

Your algorithm should run in *O*(*n*) time and uses constant extra space.

42 Trapping Rain Water
Hard

Given *n* non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped. **Thanks Marcos** for contributing this image!

Example:

```
Input: [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
```

43 Multiply Strings
Medium

Given two non-negative integers `num1` and `num2` represented as strings, return the product of `num1` and `num2`, also represented as a string.

Example 1:

```
Input: num1 = "2", num2 = "3"
Output: "6"
```

Example 2:

```
Input: num1 = "123", num2 = "456"
Output: "56088"
```

Note:

1. The length of both `num1` and `num2` is < 110.
2. Both `num1` and `num2` contain only digits `0-9`.
3. Both `num1` and `num2` do not contain any leading zero, except the number 0 itself.
4. You **must not use any built-in BigInteger library** or **convert the inputs to integer** directly.

44 Wildcard Matching
Hard

Given an input string (`s`) and a pattern (`p`), implement wildcard pattern matching with support for `'?'` and `'*'` .

```
'?' Matches any single character.
'*' Matches any sequence of characters (including the empty sequence).
```

The matching should cover the **entire** input string (not partial).

Note:

- `s` could be empty and contains only lowercase letters `a-z` .
- `p` could be empty and contains only lowercase letters `a-z` , and characters like `?` or `*` .

Example 1:

```
Input:
s = "aa"
p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input:
s = "aa"
p = "*"
Output: true
Explanation: '*' matches any sequence.
```

Example 3:

```
Input:
s = "cb"
p = "?a"
Output: false
Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.
```

Example 4:

```
Input:
s = "adceb"
p = "*a*b"
Output: true
Explanation: The first '*' matches the empty sequence, while the second '*' matches the substring "dce".
```

Example 5:

```
Input:
s = "acdcb"
p = "a*c?b"
Output: false
```

45 Jump Game II

Hard

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.

Your goal is to reach the last index in the minimum number of jumps.

Example:

```
Input: [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2.
    Jump 1 step from index 0 to 1, then 3 steps to the last index.
```

Note:

You can assume that you can always reach the last index.

46 Permutations
Medium

Given a collection of **distinct** integers, return all possible permutations.

Example:

```
Input: [1,2,3]
Output:
[
  [1,2,3],
  [1,3,2],
  [2,1,3],
  [2,3,1],
  [3,1,2],
  [3,2,1]
]
```

47 Permutations II
Medium

Given a collection of numbers that might contain duplicates, return all possible unique permutations.

Example:

```
Input: [1,1,2]
Output:
[
  [1,1,2],
  [1,2,1],
  [2,1,1]
]
```

48 Rotate Image
Medium

You are given an *n* x *n* 2D matrix representing an image.

Rotate the image by 90 degrees (clockwise).

Note:

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:

```
Given input matrix =
[
  [1,2,3],
  [4,5,6],
  [7,8,9]
],

rotate the input matrix in-place such that it becomes:
[
  [7,4,1],
  [8,5,2],
  [9,6,3]
]
```

Example 2:

```
Given input matrix =
[
  [ 5, 1, 9,11],
  [ 2, 4, 8,10],
  [13, 3, 6, 7],
  [15,14,12,16]
],

rotate the input matrix in-place such that it becomes:
[
  [15,13, 2, 5],
  [14, 3, 4, 1],
  [12, 6, 8, 9],
  [16, 7,10,11]
]
```

49 Group Anagrams

Medium

Given an array of strings, group anagrams together.

Example:

```
Input: ["eat", "tea", "tan", "ate", "nat", "bat"],
Output:
[
  ["ate","eat","tea"],
  ["nat","tan"],
  ["bat"]
]
```

Note:

- All inputs will be in lowercase.
- The order of your output does not matter.

50 Pow(x, n)

Medium

Implement pow(*x*, *n*), which calculates *x* raised to the power *n* (x^n).

Example 1:

```
Input: 2.00000, 10
Output: 1024.00000
```

Example 2:

```
Input: 2.10000, 3
Output: 9.26100
```

Example 3:

```
Input: 2.00000, -2
Output: 0.25000
Explanation: 2^-2 = 1/2^2 = 1/4 = 0.25
```

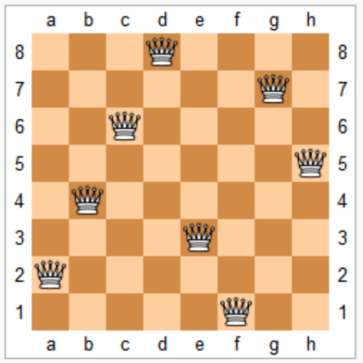
Note:

- -100.0 < *x* < 100.0
- *n* is a 32-bit signed integer, within the range [-2^{31}, $2^{31} - 1$]

51 N-Queens

Hard

The *n*-queens puzzle is the problem of placing *n* queens on an *n*×*n* chessboard such that no two queens attack each other.



One solution to the eight queens puzzle

Given an integer *n*, return all distinct solutions to the *n*-queens puzzle.

Each solution contains a distinct board configuration of the *n*-queens' placement, where `'Q'` and `'.'` both indicate a queen and an empty space respectively.

Example:

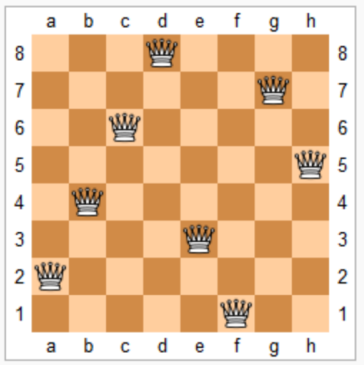
```
Input: 4
Output: [
 [".Q..",  // Solution 1
  "...Q",
  "Q...",
  "..Q."],

 ["..Q.",  // Solution 2
  "Q...",
  "...Q",
  ".Q.."]
]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above.
```

52 N-Queens II

Hard

The *n*-queens puzzle is the problem of placing *n* queens on an *n*×*n* chessboard such that no two queens attack each other.



One solution to the eight queens puzzle

Given an integer *n*, return the number of distinct solutions to the *n*-queens puzzle.

Example:

```
Input: 4
Output: 2
Explanation: There are two distinct solutions to the 4-queens puzzle as shown below.
[
 [".Q..",  // Solution 1
  "...Q",
  "Q...",
  "..Q."],

 ["..Q.",  // Solution 2
  "Q...",
  "...Q",
  ".Q.."]
]
```

53 Maximum Subarray

Easy

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

```
Input: [-2,1,-3,4,-1,2,1,-5,4],
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

Follow up:

If you have figured out the O(*n*) solution, try coding another solution using the divide and conquer approach, which is more subtle.

54 Spiral Matrix

Medium

Given a matrix of *m* x *n* elements (*m* rows, *n* columns), return all elements of the matrix in spiral order.

Example 1:

```
Input:
[
 [ 1, 2, 3 ],
 [ 4, 5, 6 ],
 [ 7, 8, 9 ]
]
Output: [1,2,3,6,9,8,7,4,5]
```

Example 2:

```
Input:
[
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9,10,11,12]
]
Output: [1,2,3,4,8,12,11,10,9,5,6,7]
```

55 Jump Game

Medium

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.

Example 1:

```
Input: [2,3,1,1,4]
Output: true
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.
```

Example 2:

```
Input: [3,2,1,0,4]
Output: false
Explanation: You will always arrive at index 3 no matter what. Its maximum
             jump length is 0, which makes it impossible to reach the last index.
```

56 Merge Intervals

Medium

Given a collection of intervals, merge all overlapping intervals.

Example 1:

```
Input: [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlaps, merge them into [1,6].
```

Example 2:

```
Input: [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considerred overlapping.
```

57 Insert Interval
Hard

Given a set of *non-overlapping* intervals, insert a new interval into the intervals (merge if necessary).

You may assume that the intervals were initially sorted according to their start times.

Example 1:

```
Input: intervals = [[1,3],[6,9]], newInterval = [2,5]
Output: [[1,5],[6,9]]
```

Example 2:

```
Input: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8]
Output: [[1,2],[3,10],[12,16]]
Explanation: Because the new interval [4,8] overlaps with [3,5],[6,7],[8,10].
```

58 Length of Last Word
Easy

Given a string *s* consists of upper/lower-case alphabets and empty space characters `' '`, return the length of last word in the string.

If the last word does not exist, return 0.

Note: A word is defined as a character sequence consists of non-space characters only.

Example:

```
Input: "Hello World"
Output: 5
```

59 Spiral Matrix II
Medium

Given a positive integer *n*, generate a square matrix filled with elements from 1 to n^2 in spiral order.

Example:

```
Input: 3
Output:
[
 [ 1, 2, 3 ],
 [ 8, 9, 4 ],
 [ 7, 6, 5 ]
]
```

60 Permutation Sequence

Medium

The set `[1,2,3,...,n]` contains a total of *n*! unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for *n* = 3:

1. `"123"`
2. `"132"`
3. `"213"`
4. `"231"`
5. `"312"`
6. `"321"`

Given *n* and *k*, return the k^{th} permutation sequence.

Note:

- Given *n* will be between 1 and 9 inclusive.
- Given *k* will be between 1 and *n*! inclusive.

Example 1:

```
Input: n = 3, k = 3
Output: "213"
```

Example 2:

```
Input: n = 4, k = 9
Output: "2314"
```

61 Rotate List

Medium

Given a linked list, rotate the list to the right by *k* places, where *k* is non-negative.

Example 1:

```
Input: 1->2->3->4->5->NULL, k = 2
Output: 4->5->1->2->3->NULL
Explanation:
rotate 1 steps to the right: 5->1->2->3->4->NULL
rotate 2 steps to the right: 4->5->1->2->3->NULL
```

Example 2:

```
Input: 0->1->2->NULL, k = 4
Output: 2->0->1->NULL
Explanation:
rotate 1 steps to the right: 2->0->1->NULL
rotate 2 steps to the right: 1->2->0->NULL
rotate 3 steps to the right: 0->1->2->NULL
rotate 4 steps to the right: 2->0->1->NULL
```

62 Unique Paths

Medium

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?

Above is a 7 x 3 grid. How many possible unique paths are there?

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

Example 1:

```
Input: m = 3, n = 2
Output: 3
Explanation:
From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:
1. Right -> Right -> Down
2. Right -> Down -> Right
3. Down -> Right -> Right
```

Example 2:

```
Input: m = 7, n = 3
Output: 28
```

63 Unique Paths II

Medium

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).

Now consider if some obstacles are added to the grids. How many unique paths would there be?

An obstacle and empty space is marked as `1` and `0` respectively in the grid.

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

Example 1:

```
Input:
[
  [0,0,0],
  [0,1,0],
  [0,0,0]
]
Output: 2
Explanation:
There is one obstacle in the middle of the 3x3 grid above.
There are two ways to reach the bottom-right corner:
1. Right -> Right -> Down -> Down
2. Down -> Down -> Right -> Right
```

64 Minimum Path Sum

Medium

Given a *m* x *n* grid filled with non-negative numbers, find a path from top left to bottom right which *minimizes* the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.

Example:

```
Input:
[
  [1,3,1],
  [1,5,1],
  [4,2,1]
]
Output: 7
Explanation: Because the path 1→3→1→1→1 minimizes the sum.
```

65 Valid Number
Hard

Validate if a given string is numeric.

Some examples:
`"0"` => `true`
`" 0.1 "` => `true`
`"abc"` => `false`
`"1 a"` => `false`
`"2e10"` => `true`

Note: It is intended for the problem statement to be ambiguous. You should gather all requirements up front before implementing one.

Update (2015-02-10):
The signature of the `C++` function had been updated. If you still see your function signature accepts a `const char *` argument, please click the reload button to reset your code definition.

66 Plus One
Easy

Given a **non-empty** array of digits representing a non-negative integer, plus one to the integer.

The digits are stored such that the most significant digit is at the head of the list, and each element in the array contain a single digit.

You may assume the integer does not contain any leading zero, except the number 0 itself.

Example 1:

```
Input: [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
```

Example 2:

```
Input: [4,3,2,1]
Output: [4,3,2,2]
Explanation: The array represents the integer 4321.
```

67 Add Binary
Easy

Given two binary strings, return their sum (also a binary string).

The input strings are both **non-empty** and contains only characters `1` or `0`.

Example 1:

```
Input: a = "11", b = "1"
Output: "100"
```

Example 2:

```
Input: a = "1010", b = "1011"
Output: "10101"
```

68 Text Justification

Hard

Given an array of words and a width *maxWidth*, format the text such that each line has exactly *maxWidth* characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces `' '` when necessary so that each line has exactly *maxWidth* characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line do not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left justified and no **extra** space is inserted between words.

Note:

- A word is defined as a character sequence consisting of non-space characters only.
- Each word's length is guaranteed to be greater than 0 and not exceed *maxWidth*.
- The input array `words` contains at least one word.

Example 1:

```
Input:
words = ["This", "is", "an", "example", "of", "text", "justification."]
maxWidth = 16
Output:
[
   "This    is    an",
   "example  of text",
   "justification.  "
]
```

Example 2:

```
Input:
words = ["What","must","be","acknowledgment","shall","be"]
maxWidth = 16
Output:
[
  "What   must   be",
  "acknowledgment  ",
  "shall be        "
]
Explanation: Note that the last line is "shall be    " instead of "shall     be",
             because the last line must be left-justified instead of fully-justified.
             Note that the second line is also left-justified becase it contains only one word.
```

Example 3:

```
Input:
words = ["Science","is","what","we","understand","well","enough","to","explain",
         "to","a","computer.","Art","is","everything","else","we","do"]
maxWidth = 20
Output:
[
  "Science  is  what we",
  "understand      well",
  "enough to explain to",
  "a  computer.  Art is",
  "everything  else  we",
  "do                  "
]
```

69 Sqrt(x)
Easy

Implement `int sqrt(int x)`.

Compute and return the square root of *x*, where *x* is guaranteed to be a non-negative integer.

Since the return type is an integer, the decimal digits are truncated and only the integer part of the result is returned.

Example 1:

```
Input: 4
Output: 2
```

Example 2:

```
Input: 8
Output: 2
Explanation: The square root of 8 is 2.82842..., and since
             the decimal part is truncated, 2 is returned.
```

70 Climbing Stairs

Easy

You are climbing a stair case. It takes *n* steps to reach to the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Note: Given *n* will be a positive integer.

Example 1:

```
Input: 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps
```

Example 2:

```
Input: 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step
```

71 Simplify Path

Medium

Given an absolute path for a file (Unix-style), simplify it.

For example,
path = `"/home/"` , => `"/home"`
path = `"/a/./b/../../c/"` , => `"/c"`

click to show corner cases.

Corner Cases:

- Did you consider the case where **path** = `"/../"` ?
  In this case, you should return `"/"` .
- Another corner case is the path might contain multiple slashes `'/'` together, such as `"/home//foo/"` .
  In this case, you should ignore redundant slashes and return `"/home/foo"` .

72 Edit Distance
Hard

Given two words *word1* and *word2*, find the minimum number of operations required to convert *word1* to *word2*.

You have the following 3 operations permitted on a word:

1. Insert a character
2. Delete a character
3. Replace a character

Example 1:

```
Input: word1 = "horse", word2 = "ros"
Output: 3
Explanation:
horse -> rorse (replace 'h' with 'r')
rorse -> rose (remove 'r')
rose -> ros (remove 'e')
```

Example 2:

```
Input: word1 = "intention", word2 = "execution"
Output: 5
Explanation:
intention -> inention (remove 't')
inention -> enention (replace 'i' with 'e')
enention -> exention (replace 'n' with 'x')
exention -> exection (replace 'n' with 'c')
exection -> execution (insert 'u')
```

73 Set Matrix Zeroes
Medium

Given a *m* x *n* matrix, if an element is 0, set its entire row and column to 0. Do it **in-place**.

Example 1:

```
Input:
[
  [1,1,1],
  [1,0,1],
  [1,1,1]
]
Output:
[
  [1,0,1],
  [0,0,0],
  [1,0,1]
]
```

Example 2:

```
Input:
[
  [0,1,2,0],
  [3,4,5,2],
  [1,3,1,5]
]
Output:
[
  [0,0,0,0],
  [0,4,5,0],
  [0,3,1,0]
]
```

Follow up:

- A straight forward solution using O(*mn*) space is probably a bad idea.
- A simple improvement uses O(*m* + *n*) space, but still not the best solution.
- Could you devise a constant space solution?

74 Search a 2D Matrix

Medium

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Example 1:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 3
Output: true
```

Example 2:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 13
Output: false
```

75 Sort Colors

Medium

Given an array with *n* objects colored red, white or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

Note: You are not suppose to use the library's sort function for this problem.

Example:

```
Input: [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

Follow up:

- A rather straight forward solution is a two-pass algorithm using counting sort.
  First, iterate the array counting number of 0's, 1's, and 2's, then overwrite array with total number of 0's, then 1's and followed by 2's.
- Could you come up with a one-pass algorithm using only constant space?

76 Minimum Window Substring

Hard

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

Example:

```
Input: S = "ADOBECODEBANC", T = "ABC"
Output: "BANC"
```

Note:

- If there is no such window in S that covers all characters in T, return the empty string `""`.
- If there is such window, you are guaranteed that there will always be only one unique minimum window in S.

77 Combinations

Medium

Given two integers *n* and *k*, return all possible combinations of *k* numbers out of 1 ... *n*.

Example:

```
Input: n = 4, k = 2
Output:
[
  [2,4],
  [3,4],
  [2,3],
  [1,2],
  [1,3],
  [1,4],
]
```

78 Subsets

Medium

Given a set of **distinct** integers, *nums*, return all possible subsets (the power set).

Note: The solution set must not contain duplicate subsets.

Example:

```
Input: nums = [1,2,3]
Output:
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]
```

79 Word Search

Medium

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

Example:

```
board =
[
  ['A','B','C','E'],
  ['S','F','C','S'],
  ['A','D','E','E']
]

Given word = "ABCCED", return true.
Given word = "SEE", return true.
Given word = "ABCB", return false.
```

80 Remove Duplicates from Sorted Array II

Medium

Given a sorted array *nums*, remove the duplicates **in-place** such that duplicates appeared at most *twice* and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array in-place** with O(1) extra memory.

Example 1:

```
Given nums = [1,1,1,2,2,3],

Your function should return length = 5, with the first five elements of nums being 1, 1, 2, 2 and 3 respect

It doesn't matter what you leave beyond the returned length.
```

Example 2:

```
Given nums = [0,0,1,1,1,1,2,3,3],

Your function should return length = 7, with the first seven elements of nums being modified to 0, 0, 1, 1,

It doesn't matter what values are set beyond the returned length.
```

Clarification:

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

```
// nums is passed in by reference. (i.e., without making a copy)
int len = removeDuplicates(nums);

// any modification to nums in your function would be known by the caller.
// using the length returned by your function, it prints the first len elements.
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

81 Search in Rotated Sorted Array II

Medium

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,0,1,2,2,5,6]` might become `[2,5,6,0,0,1,2]`).

You are given a target value to search. If found in the array return `true`, otherwise return `false`.

Example 1:

```
Input: nums = [2,5,6,0,0,1,2], target = 0
Output: true
```

Example 2:

```
Input: nums = [2,5,6,0,0,1,2], target = 3
Output: false
```

Follow up:

- This is a follow up problem to Search in Rotated Sorted Array, where `nums` may contain duplicates.
- Would this affect the run-time complexity? How and why?

82 Remove Duplicates from Sorted List II

Medium

Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only *distinct* numbers from the original list.

Example 1:

```
Input: 1->2->3->3->4->4->5
Output: 1->2->5
```

Example 2:

```
Input: 1->1->1->2->3
Output: 2->3
```

83 Remove Duplicates from Sorted List

Easy

Given a sorted linked list, delete all duplicates such that each element appear only *once*.

Example 1:

```
Input: 1->1->2
Output: 1->2
```

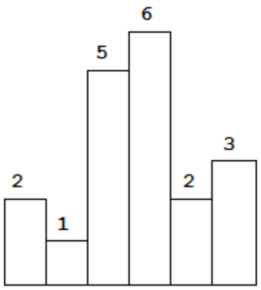
Example 2:

```
Input: 1->1->2->3->3
Output: 1->2->3
```

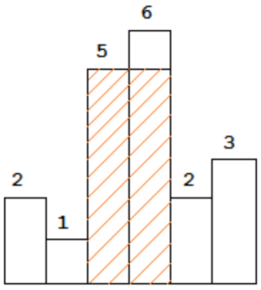
84 Largest Rectangle in Histogram

Hard

Given *n* non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.



Above is a histogram where width of each bar is 1, given height = `[2,1,5,6,2,3]`.



The largest rectangle is shown in the shaded area, which has area = `10` unit.

Example:

```
Input: [2,1,5,6,2,3]
Output: 10
```

85 Maximal Rectangle

Hard

Given a 2D binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area.

Example:

```
Input:
[
  ["1","0","1","0","0"],
  ["1","0","1","1","1"],
  ["1","1","1","1","1"],
  ["1","0","0","1","0"]
]
Output: 6
```

86 Partition List

Medium

Given a linked list and a value *x*, partition it such that all nodes less than *x* come before nodes greater than or equal to *x*.

You should preserve the original relative order of the nodes in each of the two partitions.

Example:

```
Input: head = 1->4->3->2->5->2, x = 3
Output: 1->2->2->4->3->5
```

87 Scramble String

Hard

Given a string *s1*, we may represent it as a binary tree by partitioning it to two non-empty substrings recursively.

Below is one possible representation of *s1* = `"great"`:

```
    great
   /    \
  gr    eat
 / \    /  \
g   r  e   at
           / \
          a   t
```

To scramble the string, we may choose any non-leaf node and swap its two children.

For example, if we choose the node `"gr"` and swap its two children, it produces a scrambled string `"rgeat"`.

```
    rgeat
   /    \
  rg    eat
 / \    /  \
r   g  e   at
           / \
          a   t
```

We say that `"rgeat"` is a scrambled string of `"great"`.

Similarly, if we continue to swap the children of nodes `"eat"` and `"at"`, it produces a scrambled string `"rgtae"`.

```
    rgtae
   /    \
  rg    tae
 / \    /  \
r   g  ta  e
       / \
      t   a
```

We say that `"rgtae"` is a scrambled string of `"great"`.

Given two strings *s1* and *s2* of the same length, determine if *s2* is a scrambled string of *s1*.

Example 1:

```
Input: s1 = "great", s2 = "rgeat"
Output: true
```

Example 2:

```
Input: s1 = "abcde", s2 = "caebd"
Output: false
```

88 Merge Sorted Array

Easy

Given two sorted integer arrays *nums1* and *nums2*, merge *nums2* into *nums1* as one sorted array.

Note:

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

Example:

```
Input:
nums1 = [1,2,3,0,0,0], m = 3
nums2 = [2,5,6],       n = 3

Output: [1,2,2,3,5,6]
```

89 Gray Code
Medium

The gray code is a binary numeral system where two successive values differ in only one bit.

Given a non-negative integer *n* representing the total number of bits in the code, print the sequence of gray code. A gray code sequence must begin with 0.

For example, given *n* = 2, return `[0,1,3,2]`. Its gray code sequence is:

```
00 - 0
01 - 1
11 - 3
10 - 2
```

Note:
For a given *n*, a gray code sequence is not uniquely defined.

For example, `[0,2,3,1]` is also a valid gray code sequence according to the above definition.

For now, the judge is able to judge based on one instance of gray code sequence. Sorry about that.

90 Subsets II
Medium

Given a collection of integers that might contain duplicates, ***nums***, return all possible subsets (the power set).

Note: The solution set must not contain duplicate subsets.

Example:

```
Input: [1,2,2]
Output:
[
  [2],
  [1],
  [1,2,2],
  [2,2],
  [1,2],
  []
]
```

91 Decode Ways
Medium

A message containing letters from `A-Z` is being encoded to numbers using the following mapping:

```
'A' -> 1
'B' -> 2
...
'Z' -> 26
```

Given a **non-empty** string containing only digits, determine the total number of ways to decode it.

Example 1:

```
Input: "12"
Output: 2
Explanation: It could be decoded as "AB" (1 2) or "L" (12).
```

Example 2:

```
Input: "226"
Output: 3
Explanation: It could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).
```

92 Reverse Linked List II
Medium

Reverse a linked list from position *m* to *n*. Do it in one-pass.

Note: 1 ≤ *m* ≤ *n* ≤ length of list.

Example:

```
Input: 1->2->3->4->5->NULL, m = 2, n = 4
Output: 1->4->3->2->5->NULL
```

93 Restore IP Addresses
Medium

Given a string containing only digits, restore it by returning all possible valid IP address combinations.

Example:

```
Input: "25525511135"
Output: ["255.255.11.135", "255.255.111.35"]
```

94 Binary Tree Inorder Traversal
Medium

Given a binary tree, return the *inorder* traversal of its nodes' values.

Example:

```
Input: [1,null,2,3]
   1
    \
     2
    /
   3

Output: [1,3,2]
```

Follow up: Recursive solution is trivial, could you do it iteratively?

95 Unique Binary Search Trees II
Medium

Given an integer *n*, generate all structurally unique **BST's** (binary search trees) that store values 1 ... *n*.

Example:

```
Input: 3
Output:
[
  [1,null,3,2],
  [3,2,null,1],
  [3,1,null,null,2],
  [2,1,3],
  [1,null,2,null,3]
]
Explanation:
The above output corresponds to the 5 unique BST's shown below:

   1         3     3      2      1
    \       /     /      / \      \
     3     2     1      1   3      2
    /     /       \                 \
   2     1         2                 3
```

96 Unique Binary Search Trees
Medium

Given *n*, how many structurally unique **BST's** (binary search trees) that store values 1 ... *n*?

Example:

```
Input: 3
Output: 5
Explanation:
Given n = 3, there are a total of 5 unique BST's:

   1         3     3      2      1
    \       /     /      / \      \
     3     2     1      1   3      2
    /     /       \                 \
   2     1         2                 3
```

97 Interleaving String

Hard

Given *s1*, *s2*, *s3*, find whether *s3* is formed by the interleaving of *s1* and *s2*.

Example 1:

```
Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac"
Output: true
```

Example 2:

```
Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbbaccc"
Output: false
```

98 Validate Binary Search Tree

Medium

Given a binary tree, determine if it is a valid binary search tree (BST).

Assume a BST is defined as follows:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:

```
Input:
    2
   / \
  1   3
Output: true
```

Example 2:

```
    5
   / \
  1   4
     / \
    3   6
Output: false
Explanation: The input is: [5,1,4,null,null,3,6]. The root node's value
             is 5 but its right child's value is 4.
```

99 Recover Binary Search Tree

Hard

Two elements of a binary search tree (BST) are swapped by mistake.

Recover the tree without changing its structure.

Example 1:

```
Input: [1,3,null,null,2]

   1
  /
 3
  \
   2

Output: [3,1,null,null,2]

   3
  /
 1
  \
   2
```

Example 2:

```
Input: [3,1,4,null,null,2]

  3
 / \
1   4
   /
  2

Output: [2,1,4,null,null,3]

  2
 / \
1   4
   /
  3
```

Follow up:

- A solution using O(*n*) space is pretty straight forward.
- Could you devise a constant space solution?

100 Same Tree

Easy

Given two binary trees, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical and the nodes have the same value.

Example 1:

```
Input:     1         1
          / \       / \
         2   3     2   3

        [1,2,3],   [1,2,3]

Output: true
```

Example 2:

```
Input:     1         1
          /           \
         2             2

        [1,2],     [1,null,2]

Output: false
```

Example 3:

```
Input:     1         1
          / \       / \
         2   1     1   2

        [1,2,1],   [1,1,2]

Output: false
```