1. 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, 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].
```

使用结构体+sort+双指针

2. Add Two Numbers

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:

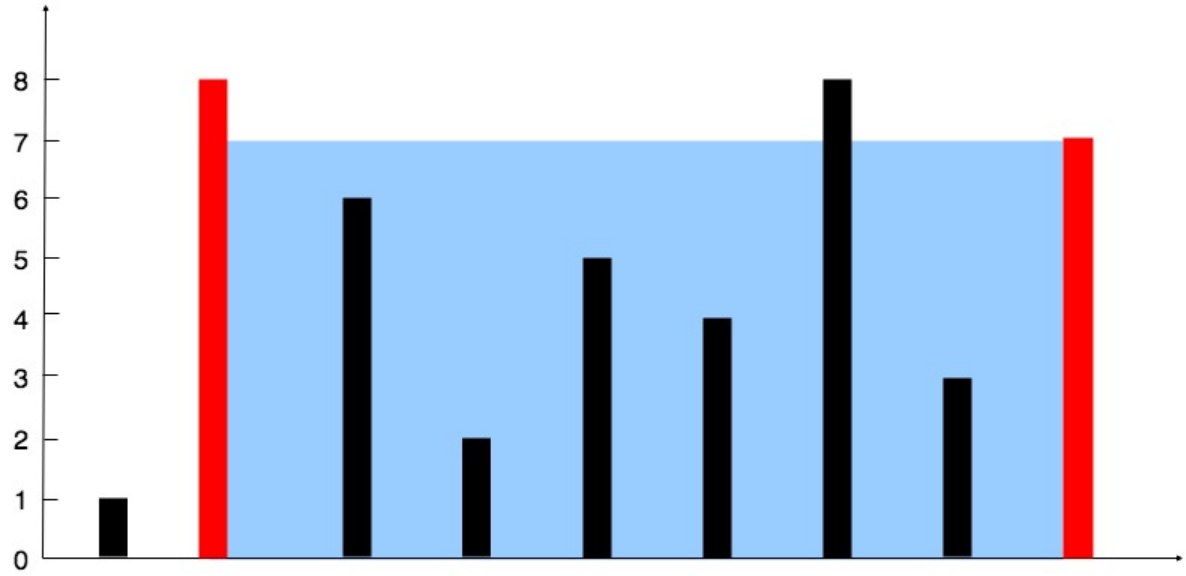
```
Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.
```

使用d偏移量+尾插法

11. Container With Most Water

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.



The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

Example:

```
Input: [1,8,6,2,5,4,8,3,7]
Output: 49
```

双指针

26. Remove Duplicates from Sorted Array

Given a sorted array *nums*, remove the duplicates **in-place** (https://en.wikipedia.org/wiki/In-place_algorithm) 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 (https://en.wikipedia.org/wiki/In-place_algorithm)** 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, and 4 respectively.

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]);
}
```

原地删除重复项，使用flag记录当前长度

---

27. Remove Element

Given an array *nums* and a value *val*, remove all instances of that value **in-place** (https://en.wikipedia.org/wiki/In-place_algorithm) and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array in-place (https://en.wikipedia.org/wiki/In-place_algorithm)** 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]);
}
```

使用flag记录当前长度

---

33. Search in Rotated Sorted Array

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
```

分类讨论+找到旋转点+二分法

34. Find First and Last Position of Element in Sorted Array

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]
```

二分查找+mid左右遍历（记得）

75. Sort Colors

Given an array with *n* objects colored red, white or blue, sort them **in-place (https://en.wikipedia.org/wiki/In-place_algorithm)** 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?

使用三个常量分别存储0，1，2的个数+使用flag重新赋值

---

78. Subsets

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],
  []
]
```

使用二进制模拟进行遍历

---

136. Single Number

Given a **non-empty** array of integers, every element appears *twice* except for one. Find that single one.

Note:

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

Example 1:

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

Example 2:

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

异或运算，相同为false，不同为true

169. Majority Element

Given an array of size *n*, find the majority element. The majority element is the element that appears **more than** ⌊ n/2 ⌋ times.

You may assume that the array is non-empty and the majority element always exist in the array.

Example 1:

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

Example 2:

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

使用flag进行标记，相同则+1，不同则-1，当为0时，更新res，flag置为1

191. Number of 1 Bits

Write a function that takes an unsigned integer and return the number of '1' bits it has (also known as the Hamming weight (http://en.wikipedia.org/wiki/Hamming_weight)).

Example 1:

```
Input: 00000000000000000000000000001011
Output: 3
Explanation: The input binary string 00000000000000000000000000001011 has a total
 of three '1' bits.
```

Example 2:

```
Input: 00000000000000000000000010000000
Output: 1
Explanation: The input binary string 00000000000000000000000010000000 has a total of
```

Example 3:

```
Input: 11111111111111111111111111111101
Output: 31
Explanation: The input binary string 11111111111111111111111111111101 has a total of
```

Note:

- Note that in some languages such as Java, there is no unsigned integer type. In this case, the input will be given as signed integer type and should not affect your implementation, as the internal binary representation of the integer is the same whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using 2's complement notation (https://en.wikipedia.org/wiki/Two%27s_complement). Therefore, in **Example 3** above the input represents the signed integer `-3`.

Follow up:

If this function is called many times, how would you optimize it?

进制转换模板 while(n!=0){ res=n%t; n/=t; }

---

215. Kth Largest Element in an Array

Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Example 1:

```
Input: [3,2,1,5,6,4] and k = 2
Output: 5
```

Example 2:

```
Input: [3,2,3,1,2,4,5,5,6] and k = 4
Output: 4
```

Note:
You may assume k is always valid, 1 ≤ k ≤ array's length.

---

sort+对应序号返回

---

322. Coin Change

You are given coins of different denominations and a total amount of money *amount*. Write a function to compute the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return `-1`.

Example 1:

```
Input: coins = [1, 2, 5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1
```

Example 2:

```
Input: coins = [2], amount = 3
Output: -1
```

Note:
You may assume that you have an infinite number of each kind of coin.

动态规划+背包数组dp[amount+1]+dp[i]=min(dp[i],dp[i-coins[j]]+1)

---

338. Counting Bits

Given a non negative integer number **num**. For every numbers **i** in the range **0 ≤ i ≤ num** calculate the number of 1's in their binary representation and return them as an array.

Example 1:

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

Example 2:

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

Follow up:

- It is very easy to come up with a solution with run time **O(n*sizeof(integer))**. But can you do it in linear time **O(n)** /possibly in a single pass?
- Space complexity should be **O(n)**.
- Can you do it like a boss? Do it without using any builtin function like **__builtin_popcount** in c++ or in any other language.

暴力法：使用进制转换 进阶：dp[i]=dp[i/2]+i%2

---

387. First Unique Character in a String

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

Examples:

```
s = "leetcode"
return 0.

s = "loveleetcode",
return 2.
```

Note: You may assume the string contain only lowercase letters.

使用map+遍历

---

389. Find the Difference

Given two strings ***s*** and ***t*** which consist of only lowercase letters.

String ***t*** is generated by random shuffling string ***s*** and then add one more letter at a random position.

Find the letter that was added in ***t***.

Example:

```
Input:
s = "abcd"
t = "abcde"

Output:
e

Explanation:
'e' is the letter that was added.
```

使用map来标记数组元素的个数，先判断mp[t[i]]==0，else mp[t[i]]--

455. Assign Cookies

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie. Each child i has a greed factor g_i, which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s_j. If $s_j >= g_i$, we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

Note:
You may assume the greed factor is always positive.
You cannot assign more than one cookie to one child.

Example 1:

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

Output: 1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are
And even though you have 2 cookies, since their size is both 1, you could only make
You need to output 1.
```

Example 2:

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

Output: 2

Explanation: You have 2 children and 3 cookies. The greed factors of 2 children are
You have 3 cookies and their sizes are big enough to gratify all of the children,
You need to output 2.
```

贪心+sort+加两个变量标记进度

---

693. Binary Number with Alternating Bits

Given a positive integer, check whether it has alternating bits: namely, if two adjacent bits will always have different values.

Example 1:

```
Input: 5
Output: True
Explanation:
The binary representation of 5 is: 101
```

Example 2:

```
Input: 7
Output: False
Explanation:
The binary representation of 7 is: 111.
```

Example 3:

```
Input: 11
Output: False
Explanation:
The binary representation of 11 is: 1011.
```

Example 4:

```
Input: 10
Output: True
Explanation:
The binary representation of 10 is: 1010.
```

使用res数组来存储n的二进制表示，整数的进制转换

1200. Minimum Absolute Difference

Given an array of **distinct** integers `arr`, find all pairs of elements with the minimum absolute difference of any two elements.

Return a list of pairs in ascending order(with respect to pairs), each pair `[a, b]` follows

- `a, b` are from `arr`
- `a < b`
- `b - a` equals to the minimum absolute difference of any two elements in `arr`

Example 1:

```
Input: arr = [4,2,1,3]
Output: [[1,2],[2,3],[3,4]]
Explanation: The minimum absolute difference is 1. List all pairs with difference eq
```

Example 2:

```
Input: arr = [1,3,6,10,15]
Output: [[1,3]]
```

Example 3:

```
Input: arr = [3,8,-10,23,19,-4,-14,27]
Output: [[-14,-10],[19,23],[23,27]]
```

Constraints:

- `2 <= arr.length <= 10^5`
- `-10^6 <= arr[i] <= 10^6`

---

sort+遍历

LCP 1. Guess Numbers

English description is not available for the problem. Please switch to Chinese.

遍历
