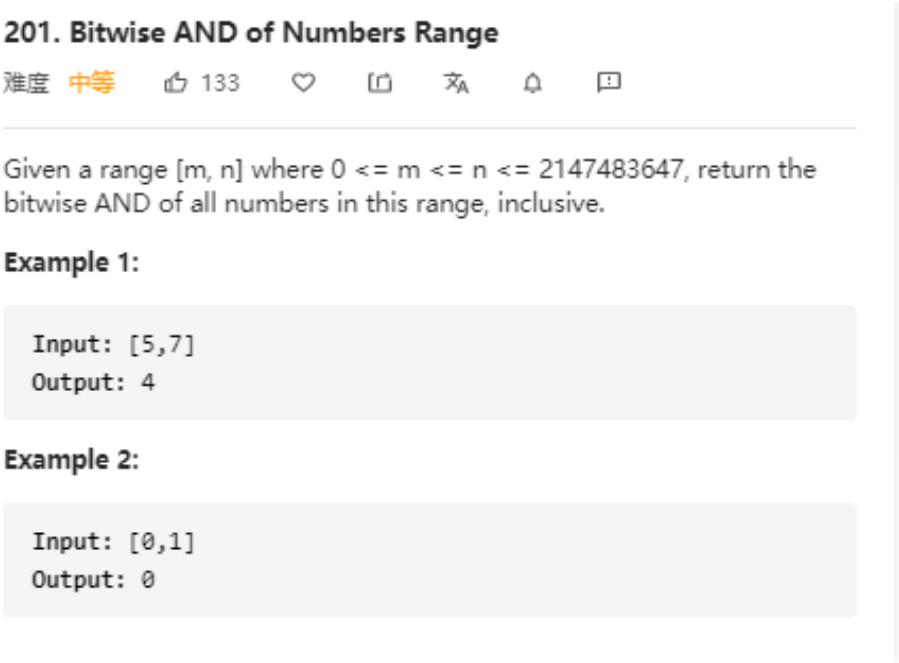
201-300

201 Bitwise AND of Numbers Range



201. Bitwise AND of Numbers Range

难度 中等 133

Given a range [m, n] where 0 <= m <= n <= 2147483647, return the bitwise AND of all numbers in this range, inclusive.

Example 1:

```
Input: [5,7]
Output: 4
```

Example 2:

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

```java
/*
  注意审题，他说的是从a~b 比如9 11
  那么就是9 & 10 & 11
  如果只是暴力与，会超出时间限制
  9   0 0 0 0 1 | 0 0 1
  10      0 0 0 0 1 | 0 1 0
  11      0 0 0 0 1 | 0 1 1
  12      0 0 0 0 1 | 1 0 0
  在对上述数字进行完与后，剩余部分是这些字符串的公共前缀
  换句话说，剩下的部分，因为至少存在1个0，都会消失，最后的结果只取决于公共前缀的大小

  因此本题求得是公共前缀的大小

  将两个数字向右移动，直到数字相等，即数字被缩减为它们的公共前缀。然后，通过将公共前缀向左移动，将零附加到公共前缀后面以获得最终结果。
*/
class Solution {
  public int rangeBitwiseAnd(int m, int n) {
    int shift = 0;
    // find the common 1-bits
    while (m < n) {
```

```
        m >>= 1;
        n >>= 1;
        ++shift;
      }
      return m << shift;
    }
}

作者：LeetCode
链接：https://leetcode-cn.com/problems/bitwise-and-of-numbers-range/solution/shu-zi-fan-wei-an-wei-yu-by-leetcode/
```

202 Happy Number

202. Happy Number

难度 简单 389

Write an algorithm to determine if a number `n` is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1. Those numbers for which this process **ends in 1** are happy numbers.

Return True if `n` is a happy number, and False if not.

Example:

```
Input: 19
Output: true
Explanation:
1^2 + 9^2 = 82
8^2 + 2^2 = 68
6^2 + 8^2 = 100
1^2 + 0^2 + 0^2 = 1
```

执行结果: 通过 显示详情 >

执行用时: 1 ms , 在所有 Java 提交中击败了 99.97% 的用户

内存消耗: 36.8 MB , 在所有 Java 提交中击败了 8.33% 的用户

炫耀一下:

```
/*
```

```java
  使用HashSet记录每次的n
  这样当出现重复的时候，直接return false
*/
public boolean isHappy(int n) {
    HashSet<Integer> set = new HashSet<>();

    while(n != 1)
    {
        if(set.contains(n))
            return false;
        int res = 0;
        set.add(n);
        while(n > 0)
        {
            int c = n % 10;
            res += c * c;
            n /= 10;
        }
        n = res;

    }
    return true;
}
```

203 Remove Linked List Elements

203. Remove Linked List Elements

难度 简单 408

Remove all elements from a linked list of integers that have value **val**.

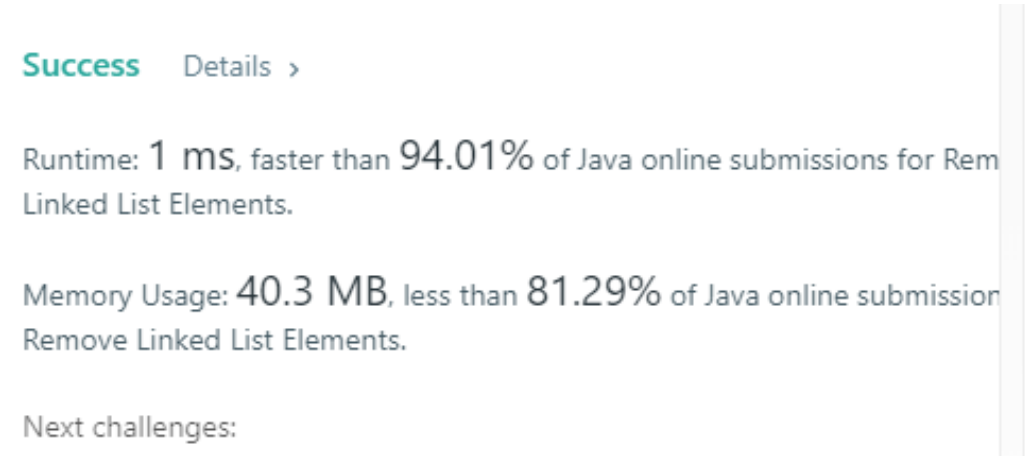
Example:

```
Input:  1->2->6->3->4->5->6, val = 6
Output: 1->2->3->4->5
```

```go
func removeElements(head *ListNode, val int) *ListNode {
  head = del(head, val)
  return head
}

func del(head *ListNode, target int) *ListNode{
  if head == nil{
    return nil
```

```
  }

  if head.Val == target{
    return del(head.Next, target)
  }else{
    head.Next = del(head.Next, target)
    return head
  }
}
```


```
public ListNode removeElements(ListNode head, int val) {
    if(head == null)        return null;
    while(head != null && head.val == val)      head = head.next;
    if(head == null)        return null;

    ListNode cur = head;
    while(cur != null && cur.next != null)
    {
        if(cur.next.val == val)
        {
            ListNode cleaner = cur.next;
            while(cleaner != null && cleaner.val == val)
                cleaner = cleaner.next;
            cur.next = cleaner;
        }
        cur = cur.next;

    }
    return head;
}
```




```java
public ListNode removeElements(ListNode head, int val) {
    if(head == null)        return null;

    ListNode dummy = new ListNode(0);
    ListNode dummyCur = dummy;

    ListNode cur = head;
    ListNode temp = null;
    while(cur != null)
    {
        temp = cur.next;
        cur.next = null;
        if(cur.val != val){
            dummyCur.next = cur;
            dummyCur = dummyCur.next;
            cur = temp;
        }

        cur = temp;
    }

    return dummy.next;
}
```

204 Count Primes

204. Count Primes

难度 简单 382

Count the number of prime numbers less than a non-negative number, **n**.

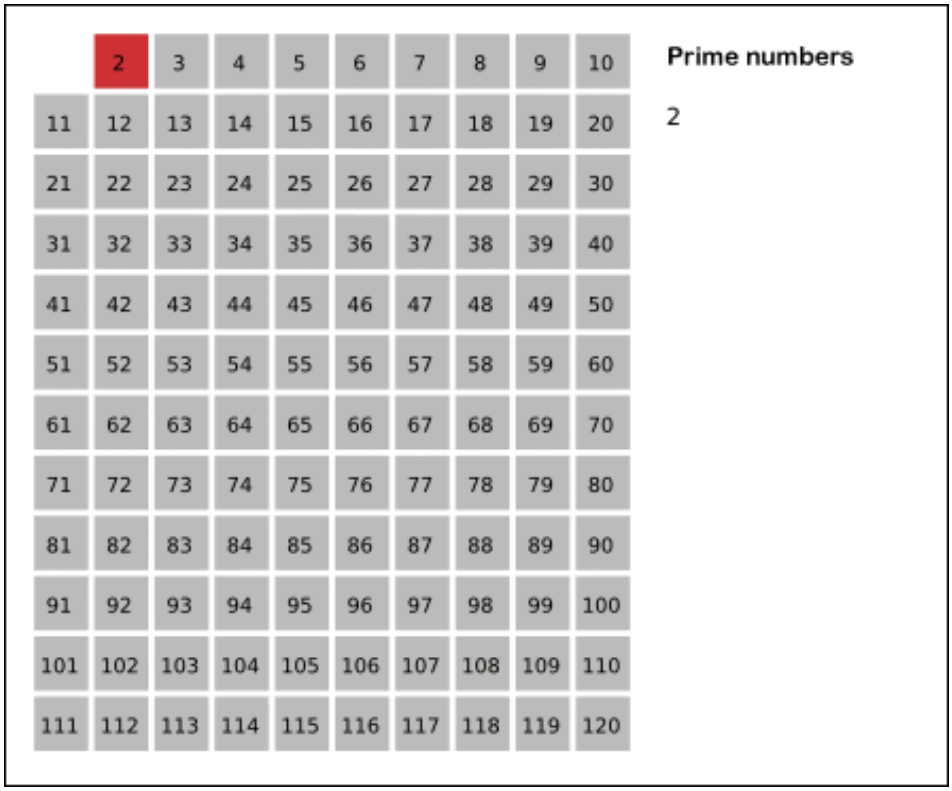
Example:

```
Input: 10
Output: 4
Explanation: There are 4 prime numbers less than 10,
they are 2, 3, 5, 7.
```

```java
//二刷
  public int countPrimes(int n) {
        if(n <= 1)
            return 0;

        boolean[] nonPrimes = new boolean[n + 1];

        for(int i = 2; i <= n; i++){
            for(int j = 2; j <= i; j++){
                int num = i * j;
                if(num > n)
                    break;
                else
                    nonPrimes[num] = true;
            }
        }

        int count = 0;
        for(int i = 2; i < n; i++){
            if(!nonPrimes[i])
                count++;
        }

        return count;
    }
```

```java
/*
  通过17/19
  超出时间限制
  准备拿备忘录优化
*/
public int countPrimes(int n) {
```

```
    if(n == 0 || n == 1)    return 0;
    if(n == 2)      return 0;

    boolean flag = false;

    int count = 1;
    for(int i = 3; i < n; i++)
    {
        for(int j = 2; j < i; j++)
        {
            if(i % j == 0)
            {
                flag = true;
                break;
            }

        }
        count += flag? 0 : 1;
        flag = false;
    }

    return count;
}
```




```
/*
  用notPrime 数组进行优化，节省时间
*/
public class Solution {
```

```
    public int countPrimes(int n) {
        boolean[] notPrime = new boolean[n];
        int count = 0;
        for (int i = 2; i < n; i++)
        {
            if (notPrime[i] == false)
            {
                count++;
                for (int j = 2; i*j < n; j++)
                    notPrime[i*j] = true;
            }
        }

        return count;
    }
}
```

205 Isomorphic Strings


```
//更简单的逻辑判断
class Solution {
    public boolean isIsomorphic(String s, String t) {
        HashMap<Character, Character> mapS = new HashMap<>();
        HashMap<Character, Character> mapT = new HashMap<>();

        if(s.length() != t.length())    return false;

        for(int i = 0; i < s.length(); i++){
            char s1 = s.charAt(i);
            char t1 = t.charAt(i);

            if(mapS.containsKey(s1)){
                if(mapS.get(s1) != t1)
                    return false;
```

```
            }else if(mapT.containsKey(t1)){
                return false;
            }else{
                mapS.put(s1, t1);
                mapT.put(t1, s1);
            }
        }

        return true;
    }
}
```

205. Isomorphic Strings

难度 简单 215

Given two strings **s** and **t**, determine if they are isomorphic.

Two strings are isomorphic if the characters in **s** can be replaced to get **t**.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character but a character may map to itself.

Example 1:

```
Input: s = "egg", t = "add"
Output: true
```

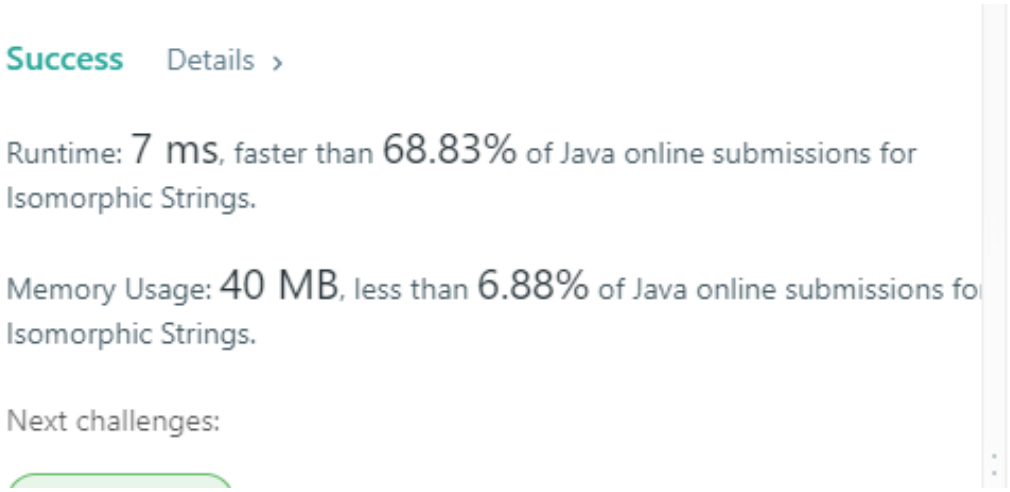
Example 2:

```
Input: s = "foo", t = "bar"
Output: false
```

Example 3:

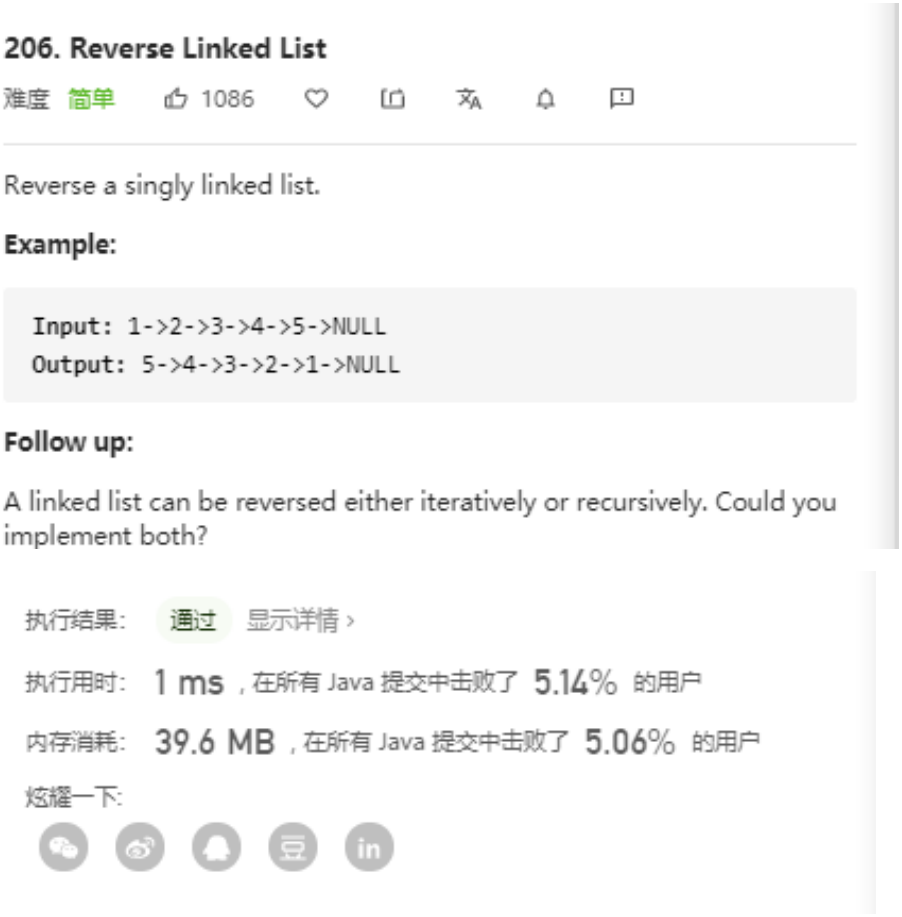
```
Input: s = "paper", t = "title"
Output: true
```

Note:
You may assume both **s** and **t** have the same length.




```java
public boolean isIsomorphic(String s, String t) {
    HashMap<Character, Character> mapS = new HashMap<>();
    HashMap<Character, Character> mapT = new HashMap<>();
    if(s.length() != t.length())        return false;

    for(int i = 0; i < s.length(); i++)
    {
        char ch1 = s.charAt(i);
        char ch2 = t.charAt(i);
        if(!mapS.containsKey(ch1) && !mapT.containsKey(ch2))
        {
            mapS.put(ch1, ch2);
            mapT.put(ch2, ch1);
        }
        else if(mapS.containsKey(ch1))
        {
            if(mapS.get(ch1) != ch2)
                return false;
        }
        else if(mapT.containsKey(ch2))
            if(mapT.get(ch2) != ch1)
                return false;
        else
        {
            if(mapS.get(ch1) != ch2 || mapT.get(ch2) != ch1)
                return false;
        }
    }
    return true;
}
```

206 Reverse Linked List



206. Reverse Linked List

难度 简单 1086

Reverse a singly linked list.

Example:

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

Follow up:

A linked list can be reversed either iteratively or recursively. Could you implement both?

执行结果: 通过 显示详情 >

执行用时: 1 ms , 在所有 Java 提交中击败了 5.14% 的用户

内存消耗: 39.6 MB , 在所有 Java 提交中击败了 5.06% 的用户

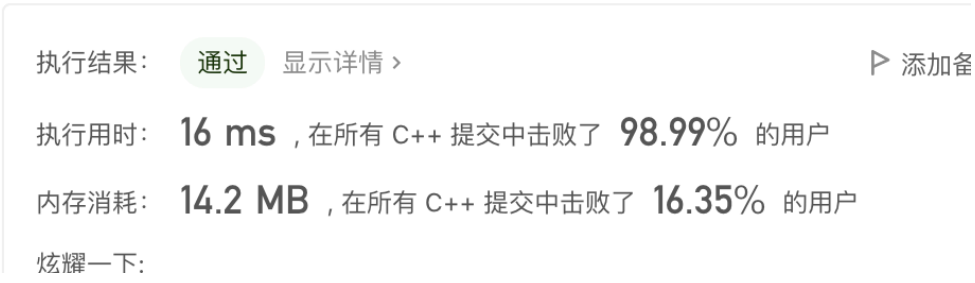
炫耀一下:

```java
public ListNode reverseList(ListNode head) {
    ListNode cur = head;
    ListNode pre = null, temp = null;

    while(cur != null)
    {
        temp = cur.next;
        cur.next = pre;
        pre = cur;
        cur = temp;
    }
    return pre;
}
```

```java
/*
  递归解法
```

```
*/
class Solution {
  public ListNode reverseList(ListNode head) {
    //递归终止条件是当前为空，或者下一个节点为空
    if(head==null || head.next==null) {
      return head;
    }
    //这里的cur就是最后一个节点
    ListNode cur = reverseList(head.next);
    //这里请配合动画演示理解
    //如果链表是 1->2->3->4->5，那么此时的cur就是5
    //而head是4，head的下一个是5，下下一个是空
    //所以head.next.next 就是5->4
    head.next.next = head;
    //防止链表循环，需要将head.next设置为空
    head.next = null;
    //每层递归函数都返回cur，也就是最后一个节点
    return cur;
  }
}

作者：wang_ni_ma
链接：https://leetcode-cn.com/problems/reverse-linked-list/solution/dong-hua-yan-shi-206-fan-zhuan-lian-biao-by-user74/
```

207 Course Schedule




```
class Solution {
public:
    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
        vector<int>       prerequisiteCourse(numCourses, 0);
        vector<set<int>>  outDegree(numCourses, set<int>());

        for(vector<int>& v : prerequisites){
```

```
            prerequisiteCourse[v[0]] += 1;
            outDegree[v[1]].insert(v[0]);
        }

        queue<int> myQueue;
        set<int> learned;

        for(int i = 0; i < numCourses; i++){
            if(prerequisiteCourse[i] == 0)
                myQueue.push(i);
        }

        while(!myQueue.empty()){
            int size = myQueue.size();
            for(int i = 0; i < size; i++){
                int curCourse = myQueue.front();
                myQueue.pop();
                learned.insert(curCourse);

                for(int nextCourse : outDegree[curCourse]){
                    prerequisiteCourse[nextCourse]--;
                    if(prerequisiteCourse[nextCourse] == 0)
                        myQueue.push(nextCourse);
                }
            }
        }


        return learned.size() == numCourses;
    }
};

```

执行结果： 通过 显示详情 >

执行用时： 132 ms , 在所有 Java 提交中击败了 5.03% 的用户

内存消耗： 39.5 MB , 在所有 Java 提交中击败了 24.31% 的用户

```
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        int[] indegree = new int[numCourses];
        List<List<Integer>> adj = new ArrayList<>();
        for(int i = 0; i < numCourses; i++)
            adj.add(new ArrayList<Integer>());

```

```
        for(int[] pre : prerequisites) {
            indegree[pre[0]]++;
            adj.get(pre[0]).add(pre[1]);
        }

        Deque<Integer> queue = new ArrayDeque<>();

        for(int i = 0; i < indegree.length; i++)
            if(indegree[i] == 0)
                queue.add(i);

        HashSet<Integer> finished = new HashSet<>();
        while(!queue.isEmpty()){
            int cur = queue.pollFirst();
            finished.add(cur);
            for(int i = 0; i < numCourses; i++){
                if(adj.get(i).contains(cur)){
                    indegree[i]--;
                    if(indegree[i] == 0)
                        queue.addLast(i);
                }
            }
        }

        return finished.size() == numCourses;
    }
```

208 Implement Trie(Prefix Tree)

```
//可以学习这种写法，尤其是 智能指针之类
class Trie {
public:
    /** Initialize your data structure here. */
    Trie() : root(new TrieNode) {}

    /** Inserts a word into the trie. */
    void insert(string word) {
        TrieNode* cur = root.get();

        for(const char ch : word){
            if(cur->children[ch - 'a'] == nullptr)
```

```
                cur->children[ch - 'a'] = new TrieNode();
            cur = cur->children[ch - 'a'];
        }

        cur->isWord = true;
    }

    /** Returns if the word is in the trie. */
    bool search(string word) {
        TrieNode* res = find(word);

        return (res != nullptr && res->isWord);
    }

    /** Returns if there is any word in the trie that starts with the given prefix.
*/
    bool startsWith(string prefix) {
        return find(prefix) != nullptr;
    }

private:
    class TrieNode{
    public:
        bool isWord;
        vector<TrieNode*> children;
        TrieNode() : isWord(false), children(26, nullptr){}

        ~TrieNode(){
            for(TrieNode* child : children){
                delete child;
            }
        }

    };

    TrieNode* find(string& str){
        TrieNode* cur = root.get();
        for(const char ch : str){
            if(cur->children[ch - 'a'] == nullptr)
                return nullptr;
            cur = cur->children[ch - 'a'];
        }

        return cur;
    }

    std::unique_ptr<TrieNode> root;

};
```

209 Minimum Size Subarray Sum

209. Minimum Size Subarray Sum

难度 中等 376

Given an array of **n** positive integers and a positive integer **s**, find the minimal length of a **contiguous** subarray of which the sum ≥ **s**. If there isn't one, return 0 instead.

Example:

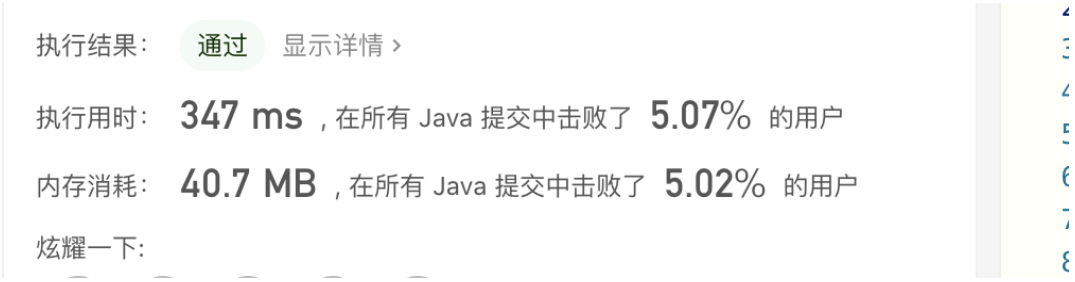
```
Input: s = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: the subarray [4,3] has the minimal length
under the problem constraint.
```

Follow up:
If you have figured out the O(n) solution, try coding another solution of which the time complexity is O(n log n).


```java
public int minSubArrayLen(int s, int[] nums) {
    int len = nums.length;
    if(len == 0)        return 0;
    if(len == 1)        return nums[0] == s ? 1 : 0;

    int left = 0;
    int right = 0;
    int window = 0;
    int res = Integer.MAX_VALUE;
    while(right < len)
    {
        for(;right < len && window < s; right++)
            window += nums[right];


        while(window >= s)
```

```
        {
            res = Math.min(res, right - left);

            window -= nums[left];
            left++;
        }
    }

    return res == Integer.MAX_VALUE ? 0 : res;
}
```

210 Course Schedule II




```
    public int[] findOrder(int numCourses, int[][] prerequisites) {
        List<Integer> res = new ArrayList<>();


        int len = numCourses;
        List<List<String>> adj = new ArrayList<>();
        for(int i = 0; i < len; i++)
            adj.add(new ArrayList<>());

        for(int[] pre : prerequisites)
            adj.get(pre[0]).add(pre[1] + "");

        Deque<Integer> queue = new ArrayDeque<>();
        for(int i = 0;  i < len; i++)
            if(adj.get(i).size() == 0)
                queue.addLast(i);

        while(!queue.isEmpty()){
```

```
            int cur = queue.pollFirst();
            res.add(cur);

            for(int i = 0; i < len; i++){
                if(adj.get(i).contains(cur + "")){
                    adj.get(i).remove(cur + "");
                    if(adj.get(i).size() == 0)
                        queue.addLast(i);
                }
            }
        }
        if(res.size() != numCourses)    return new int[]{};

        int[] ans = new int[res.size()];
        int index = 0;
        for(Integer num : res)
            ans[index++] = num;
        return ans;
    }

```

212 Word Search II

212. Word Search II

难度 困难 187

Given a 2D board and a list of words from the dictionary, find all words in the board.

Each word must 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 in a word.

Example:

```
Input:
board = [
  ['o','a','a','n'],
  ['e','t','a','e'],
  ['i','h','k','r'],
  ['i','f','l','v']
]
words = ["oath","pea","eat","rain"]

Output: ["eat","oath"]
```

Note:

1. All inputs are consist of lowercase letters `a-z`.
2. The values of `words` are distinct.

通过次数 16,188 | 提交次数 39,190

在真实的面试中遇到过这道题?

是 否


```java
/*
  这个题其实和wordbreak I 没什么本质区别
  记得案例要去重，用set
*/
List<String> res;
private int[][] pos = new int[][]{{-1, 0},{0, -1},{1, 0},{0, 1}};
private char[][] board;
private boolean[][] marked;
public List<String> findWords(char[][] board, String[] words) {
    res = new ArrayList<>();
    this.board = board;
    int row = board.length;
    int column = board[0].length;
    int index = 0;

    for(int k = 0; k < words.length; k++)
    {
        marked = new boolean[row][column];
        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(backtrack(i, j, words[k], 0))
                    if(!res.contains(words[k]))
                        res.add(words[k]);
    }

    return res;
}

private boolean backtrack(int i, int j, String s, int checkpoint)
{
    if(checkpoint == s.length() -1)
        return board[i][j] == s.charAt(checkpoint);

    if(s.charAt(checkpoint) == board[i][j])
    {
        marked[i][j] = true;
        for(int k = 0; k < 4; k++)
```

```
        {
            int newX = i + pos[k][0];
            int newY = j + pos[k][1];

            if(isInRange(newX, newY) && !marked[newX][newY])
                if(backtrack(newX, newY, s, checkpoint+1))
                    return true;
        }
        marked[i][j] = false;
    }

    return false;
}

private boolean isInRange(int m, int n)
{
    return m >= 0 && n >= 0 && m < board.length && n < board[0].length;
}
```

214 Shortest Palindrome KMP解法待研究

214. Shortest Palindrome

难度 困难 158

Given a string **s**, you are allowed to convert it to a palindrome by adding characters in front of it. Find and return the shortest palindrome you can find by performing this transformation.

Example 1:

```
Input: "aacecaaa"
Output: "aaacecaaa"
```

Example 2:

```
Input: "abcd"
Output: "dcbabcd"
```

```
/*
  思路：可以从字符串开头找到更大的回文子串，然后反转剩余的子串并附加到开头
  比如"abcbabcab"  从开头找到的最大回文子串 abcba
  将剩余的部分"bcab" 反转并拼接到后面

```

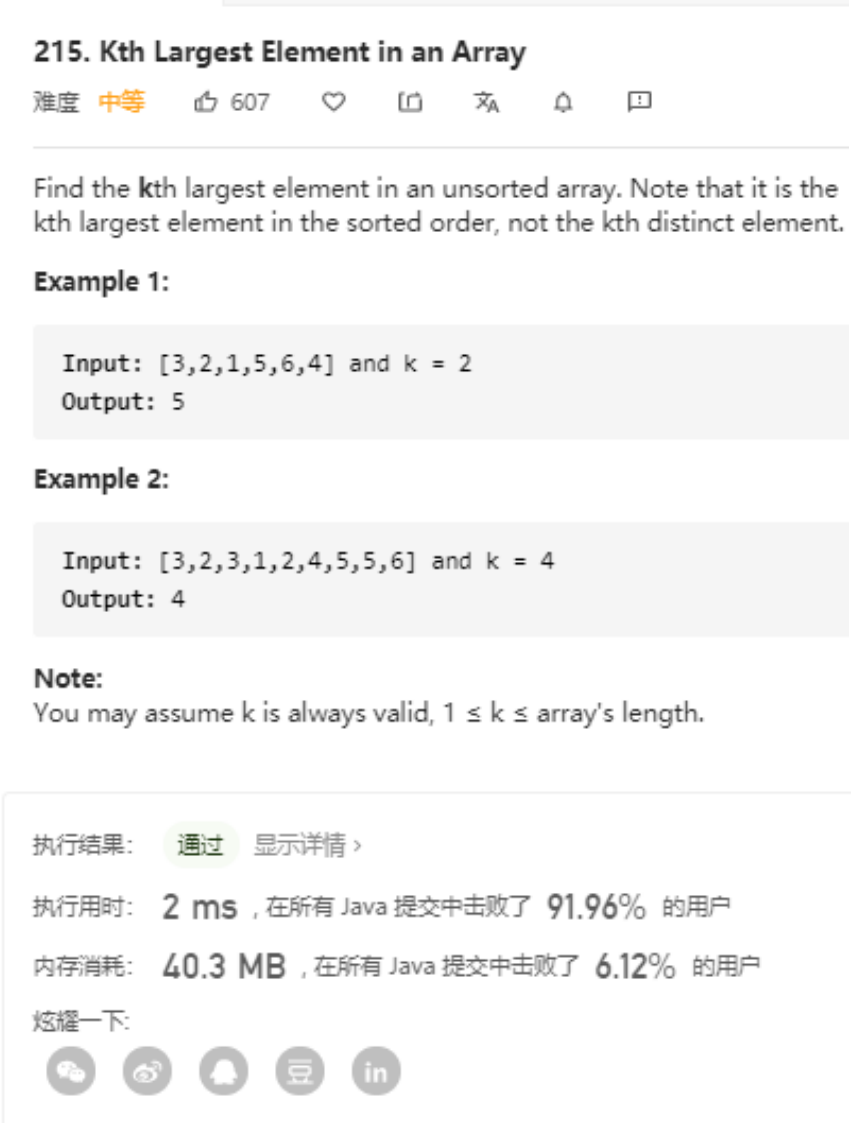
```java
  但是会超出时间限制 119/120
*/
public String shortestPalindrome(String s)
{
    if(s.length() == 0)     return "";
    int index = 0;
    String longestPalindrome = "";
    for(int i = 0; i < s.length(); i++)
    {
        String frac = s.substring(0, i+1);
        if(!isPalindrome(frac))
            continue;
        else
        {
            index = i;
            longestPalindrome = frac;
        }
    }
    StringBuilder sb = new StringBuilder(s.substring(index+1));
    String str = sb.reverse().toString();
    return str + longestPalindrome + sb.reverse().toString();

}

private boolean isPalindrome(String frac) {
    int left = 0, right = frac.length()-1;
    while(left < right)
    {
        if(frac.charAt(left) != frac.charAt(right))
            return false;
        else
        {left++; right--;}
    }
    return true;
}

//-------------------------------------------------------------
//CLEAN VERSION
//这种情况下可以过
public String shortestPalindrome(String s) {
    int i = 0, end = s.length() - 1, j = end; char chs[] = s.toCharArray();
    while(i < j) {
         if (chs[i] == chs[j]) {
             i++; j--;
         } else {
             i = 0; end--; j = end;
         }
    }
    return new StringBuilder(s.substring(end+1)).reverse().toString() + s;
```

```
}
```

```
public String shortestPalindrome(String s) {
    String temp = s + "#" + new StringBuilder(s).reverse().toString();
    int[] table = getTable(temp);

    //get the maximum palin part in s starts from 0
    return new StringBuilder(s.substring(table[table.length -
1])).reverse().toString() + s;
}

public int[] getTable(String s){
    //get lookup table
    int[] table = new int[s.length()];

    //pointer that points to matched char in prefix part

    int index = 0;
    //skip index 0, we will not match a string with itself
    for(int i = 1; i < s.length(); i++){
        if(s.charAt(index) == s.charAt(i)){
            //we can extend match in prefix and postfix
            table[i] = table[i-1] + 1;
            index ++;
        }else{
            //match failed, we try to match a shorter substring

            //by assigning index to table[i-1], we will shorten the match string
length, and jump to the
            //prefix part that we used to match postfix ended at i - 1
            index = table[i-1];

            while(index > 0 && s.charAt(index) != s.charAt(i)){
                //we will try to shorten the match string length until we revert to
the beginning of match (index 1)
                index = table[index-1];
            }

            //when we are here may either found a match char or we reach the
boundary and still no luck
            //so we need check char match
            if(s.charAt(index) == s.charAt(i)){
                //if match, then extend one char
                index ++ ;
            }

            table[i] = index;
```

```
        }

    }

    return table;
}
https://leetcode.com/problems/shortest-palindrome/discuss/60113/Clean-KMP-solution-with-super-detailed-explanation
```

215 Kth Largest Element in An array 快排 Partition



215. Kth Largest Element in an Array

难度 中等 607

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.

执行结果: 通过 显示详情 >

执行用时: 2 ms , 在所有 Java 提交中击败了 91.96% 的用户

内存消耗: 40.3 MB , 在所有 Java 提交中击败了 6.12% 的用户

炫耀一下:

```
public int findKthLargest(int[] nums, int k) {
    Arrays.sort(nums);
    int count = 0;
    for(int i = 0; i < nums.length; i++)
    {
```

```
        count++;
        if(count == nums.length - k + 1)
            return nums[i];
    }
    return -1;
}

public class Solution {

    public int findKthLargest(int[] nums, int k) {
        int len = nums.length;
        Arrays.sort(nums);
        return nums[len - k];
    }
}
```
作者：liweiwei1419

```
//用了快排中的partition
public int findKthLargest(int[] nums, int k) {
    int len = nums.length;
    int left = 0;
    int right = len - 1;

    // 转换一下，第 k 大元素的索引是 len - k
    int target = len - k;

    while (true) {
        int index = partition(nums, left, right);
        if (index == target) {
            return nums[index];
        } else if (index < target) {
            left = index + 1;
        } else {
            right = index - 1;
        }
    }
}


public int partition(int[] nums, int lo, int hi) {
    int i = lo, j = hi + 1;
    int v = nums[lo];
    while(true)
    {
        //这里对快排在算法四中进行一点修改，以便能够通过测试，否则会越界
        while(++i < nums.length && nums[i] < v) if(i == hi)   break;
        while(nums[--j] > v)                    if(j == lo)   break;
```

```
31          if(i >= j)                                  break;
32          exch(nums, i, j);
33      }
34      exch(nums, lo, j);
35      return j;
36  }
37
38  private void exch(int[] nums, int index1, int index2) {
39      int temp = nums[index1];
40      nums[index1] = nums[index2];
41      nums[index2] = temp;
42  }
```

217 Contains Duplicate

217. Contains Duplicate

难度 简单 266

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.

Example 1:

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

Example 2:

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

Example 3:

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

执行结果: 通过 显示详情 >

执行用时: 9 ms , 在所有 Java 提交中击败了 51.78% 的用户

内存消耗: 46 MB , 在所有 Java 提交中击败了 6.98% 的用户

炫耀一下:

```java
public boolean containsDuplicate(int[] nums) {
    HashSet<Integer> set = new HashSet<Integer>();
    for(int i = 0; i < nums.length; i++)
        if(set.contains(nums[i]))
            return true;
    else
        set.add(nums[i]);
    return false;
}
```

219 Contains Duplicate II 哈希表实现滑动窗口

219. Contains Duplicate II

难度 简单 181

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 **absolute** difference between i and j is at most k.

Example 1:

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

Example 2:

```
Input: nums = [1,0,1,1], k = 1
Output: true
```

Example 3:

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

执行结果: 通过 显示详情 >

执行用时: 2089 ms , 在所有 Java 提交中击败了 5.02% 的用户

内存消耗: 42.2 MB , 在所有 Java 提交中击败了 42.86% 的用户

炫耀一下:

```java
public boolean containsNearbyDuplicate(int[] nums, int k) {
    for(int i = 0; i < nums.length; i++)
        for(int j = i+1; j <= i+k; j++)
            if(j < nums.length && nums[j] == nums[i])
                return true;
    return false;
}
```


```java
public boolean containsNearbyDuplicate(int[] nums, int k) {
    HashMap<Integer, List<Integer>> map = new HashMap<>();

    for(int i = 0; i < nums.length; i++)
    {
        if(!map.containsKey(nums[i]))
            map.put(nums[i], new ArrayList<>());
        map.get(nums[i]).add(i);
    }

    for(List<Integer> jar : map.values())
    {
        if(jar.size() <= 1)     continue;

        Collections.sort(jar);
        for(int i = 0; i < jar.size()-1; i++)
            if(jar.get(i+1) - jar.get(i) <= k )
                return true;
    }
    return false;
}
```

220 Contains Duplicate III 桶排序的应用

220. Contains Duplicate III

难度 中等 195

Given an array of integers, find out whether there are two distinct indices i and j in the array such that the **absolute** difference between **nums[i]** and **nums[j]** is at most t and the **absolute** difference between i and j is at most k.

Example 1:

```
Input: nums = [1,2,3,1], k = 3, t = 0
Output: true
```

Example 2:

```
Input: nums = [1,0,1,1], k = 1, t = 2
Output: true
```

Example 3:

```
Input: nums = [1,5,9,1,5,9], k = 2, t = 3
Output: false
```

```java
//暴力解法
public boolean containsNearbyAlmostDuplicate(int[] nums, int k, int t) {
    long[] numsl = new long[nums.length];
    for(int i = 0; i < nums.length; i++)
        numsl[i] = (long)nums[i];

    for(int i = 0; i < nums.length; i++)
        for(int j = i + 1; j <= i + k && j < nums.length; j++)
            if(Math.abs(numsl[i] - numsl[j]) <= t)
                return true;
    return false;
}
```

```java
public boolean containsNearbyAlmostDuplicate(int[] nums, int k, int t) {
    if (t < 0) {
        return false;
    }
    HashMap<Long, Long> map = new HashMap<>();
    int n = nums.length;
    long w = t + 1; // 一个桶里边数字范围的个数是 t + 1
    for (int i = 0; i < n; i++) {
        //删除窗口中第一个数字
```

```
        if (i > k) {
            map.remove(getId(nums[i - k - 1], w));
        }
        //得到当前数的桶编号
        long id = getId(nums[i], w);
        if (map.containsKey(id)) {
            return true;
        }
        if (map.containsKey(id + 1) && map.get(id + 1) - nums[i] < w) {
            return true;
        }

        if (map.containsKey(id - 1) && nums[i] - map.get(id - 1) < w) {
            return true;
        }
        map.put(id, (long) nums[i]);
    }
    return false;
}

private long getId(long num, long w) {
    if (num >= 0) {
        return num / w;
    } else {
        return (num + 1) / w - 1;
    }
}

作者：windliang
链接：https://leetcode-cn.com/problems/contains-duplicate-iii/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by-46/
```

221 Maximal Rectangle

```
public int maximalSquare(char[][] matrix) {
    int row = matrix.length, column = row == 0 ? 0 :matrix[0].length;
    if(column == 0)     return 0;
    int[][] dp = new int[row][column];

    for(int i = 0; i < row; i++)
        for(int j = 0; j < column; j++)
        {
            if(i == 0)
            {
                if(matrix[i][j] == '1')
```

```
                    dp[i][j] = 1;
            }
            else if(j == 0)
            {
                if(matrix[i][j] == '1')
                    dp[i][j] = 1 + dp[i-1][j];
            }
            else
                if(matrix[i][j] == '1')
                    dp[i][j] = 1 + dp[i-1][j];
        }

    int maxArea = 0, curWidth = 0, curHeight = 0;
    for(int i = 0; i < row; i++)
        for(int j = 0; j < column; j++)
        {
            if(dp[i][j] == 0)       continue;

            curHeight = dp[i][j];
            for(int k = j; k >= 0; k--)
            {
                if(dp[i][k] == 0)       break;
                curHeight = Math.min(curHeight, dp[i][k]);
                curWidth = j - k + 1;
                int side = Math.min(curHeight, curWidth);
                maxArea = Math.max(side * side, maxArea);
            }
        }
    return maxArea;
}
```

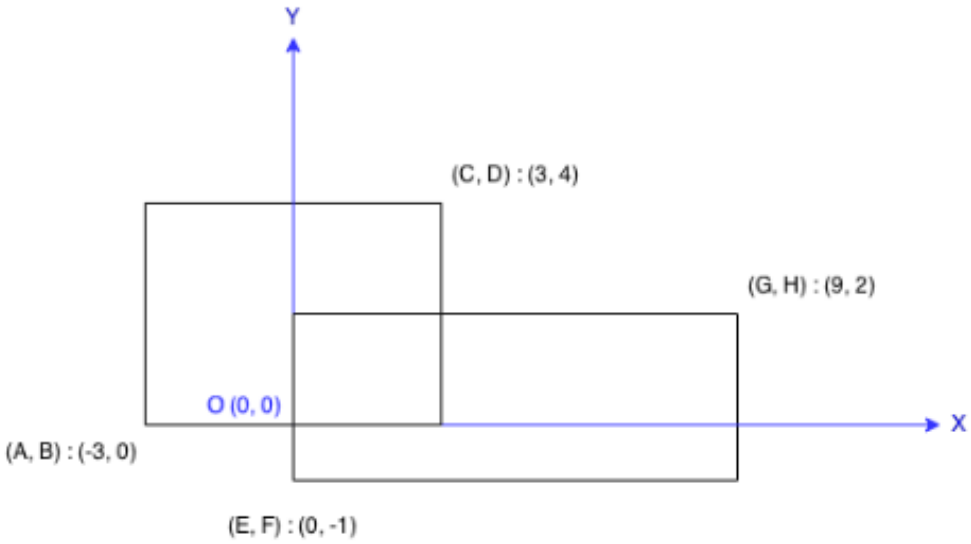
223 Rectangle Area

223. Rectangle Area

难度 中等 78

Find the total area covered by two **rectilinear** rectangles in a **2D** plane.

Each rectangle is defined by its bottom left corner and top right corner as shown in the figure.



Example:

```
Input: A = -3, B = 0, C = 3, D = 4, E = 0, F = -1, G = 9, H = 2
Output: 45
```

Note:

Assume that the total area is never beyond the maximum possible value of **int**.

通过次数 10,272 | 提交次数 23,638

在真实的面试中遇到过这道题? 是 否

```java
class Solution {
    public int computeArea(int A, int B, int C, int D, int E, int F, int G, int H)
{
        int leftTopX = Math.max(A, E);
        int leftTopY = Math.min(D, H);
        int leftBotX = leftTopX;
        int leftBotY = Math.max(B,F);

        int rightTopX = Math.min(C, G);
        int rightTopY = leftTopY;
        int rightBotX = rightTopX;
        int rightBotY = leftBotY;
```

```
12
13          int total = Math.abs(A-C) * Math.abs(B - D) + Math.abs(E - G) * Math.abs(F - H);
14
15          if(C <= E || G <= A || D <= F || H <= B)
16              return total;
17
18          return total - (rightTopX - leftTopX) * (rightTopY - rightBotY);
19      }
20  }
```

224 Basic Calculator简易计算器

224. Basic Calculator

难度 困难 223

Implement a basic calculator to evaluate a simple expression string.

The expression string may contain open `(` and closing parentheses `)`, the plus `+` or minus sign `-`, **non-negative** integers and empty spaces ` `.

Example 1:

```
Input: "1 + 1"
Output: 2
```

Example 2:

```
Input: " 2-1 + 2 "
Output: 3
```

Example 3:

```
Input: "(1+(4+5+2)-3)+(6+8)"
Output: 23
```

Note:

- You may assume that the given expression is always valid.
- **Do not** use the `eval` built-in library function.

```
1  class Solution {
2  public:
3      int calculate(string s) {
4          long res = 0;
5          long num = 0;
6          int sign = 1;
7
8          stack<int> myStack;
```

```
        int size = s.size();

        for(int i = 0; i < size; i++){
            if(s[i] == ' ')
                continue;
            else if(s[i] >= '0' && s[i] <= '9'){
                num = num * 10 + s[i] - '0';

                if(i + 1 < size && s[i + 1] <= '9' && s[i + 1] >= '0')
                    continue;
            }else if(s[i] == '+' || s[i] == '-'){
                num = 0;
                sign = s[i] == '+' ? 1 : -1;
            }else if(s[i] == '('){
                myStack.push(res);
                myStack.push(sign);

                res = 0;

                sign = 1;
            }else if(s[i] == ')'){
                sign = myStack.top(); myStack.pop();
                num = res;
                res = myStack.top(); myStack.pop();
            }

            res += sign * num;
        }

        return (int)res;
    }
};
```

```
/*
思路：  中缀转后缀表达式 + 逆波兰计算

中缀转后缀
1）如果遇到操作数，我们就直接将其加入到后缀表达式。

2）如果遇到左括号，则我们将其放入到栈中。

3）如果遇到一个右括号，则将栈元素弹出，将弹出的操作符加入到后缀表达式直到遇到左括号为止，接着将左括号弹出，但不加入到结果中。

```

```
4）如果遇到其他的操作符，如（“+”， “-”）等，从栈中弹出元素将其加入到后缀表达式，直到栈顶的元素优先级比当前的优先级低（或者遇到左括号或者栈为空）为止。弹出完这些元素后，最后将当前遇到的操作符压入到栈中。

5）如果我们读到了输入的末尾，则将栈中所有元素依次弹出。
*/

public int calculate(String s) {
    String[] polish = getPolish(s); //转后缀表达式
    return evalRPN(polish);
}

//中缀表达式转后缀表达式
private String[] getPolish(String s) {
    List<String> res = new ArrayList<>();
    Stack<String> stack = new Stack<>();
    char[] array = s.toCharArray();
    int n = array.length;
    int temp = -1; //累加数字，-1 表示当前没有数字
    for (int i = 0; i < n; i++) {
        if (array[i] == ' ') {
            continue;
        }
        //遇到数字
        if (isNumber(array[i])) {
            //进行数字的累加
            if (temp == -1) {
                temp = array[i] - '0';
            } else {
                temp = temp * 10 + array[i] - '0';
            }
        }
        else {
            //遇到其它操作符，将数字加入到结果中
            if (temp != -1) {
                res.add(temp + "");
                temp = -1;
            }
            if (isOperation(array[i] + "")) {
                //遇到操作符将栈中的操作符加入到结果中
                while (!stack.isEmpty()) {
                    //遇到左括号结束
                    if (stack.peek().equals("(")) {
                        break;
                    }
                    res.add(stack.pop());
                }
                //当前操作符入栈
                stack.push(array[i] + "");
```

```
            } else {
                //遇到左括号，直接入栈
                if (array[i] == '(') {
                    stack.push(array[i] + "");
                }
                //遇到右括号，将出栈元素加入到结果中，直到遇到左括号
                if (array[i] == ')') {
                    while (!stack.peek().equals("(")) {
                        res.add(stack.pop());
                    }
                    //左括号出栈
                    stack.pop();
                }

            }
        }
    }
    //如果有数字，将数字加入到结果
    if (temp != -1) {
        res.add(temp + "");
    }
    //栈中的其他元素加入到结果
    while (!stack.isEmpty()) {
        res.add(stack.pop());
    }
    String[] sArray = new String[res.size()];
    //List 转为 数组
    for (int i = 0; i < res.size(); i++) {
        sArray[i] = res.get(i);
    }
    return sArray;
}

// 下边是 150 题的代码，求后缀表达式的值
public int evalRPN(String[] tokens) {
    Stack<String> stack = new Stack<>();
    for (String t : tokens) {
        if (isOperation(t)) {
            int a = stringToNumber(stack.pop());
            int b = stringToNumber(stack.pop());
            int ans = eval(b, a, t.charAt(0));
            stack.push(ans + "");
        } else {
            stack.push(t);
        }
    }
    return stringToNumber(stack.pop());
}

```

```java
private int eval(int a, int b, char op) {
    switch (op) {
        case '+':
            return a + b;
        case '-':
            return a - b;
        case '*':
            return a * b;
        case '/':
            return a / b;
    }
    return 0;
}

private int stringToNumber(String s) {
    int sign = 1;
    int start = 0;
    if (s.charAt(0) == '-') {
        sign = -1;
        start = 1;
    }
    int res = 0;
    for (int i = start; i < s.length(); i++) {
        res = res * 10 + s.charAt(i) - '0';
    }
    return res * sign;
}

private boolean isNumber(char c) {
    return c >= '0' && c <= '9';
}

private boolean isOperation(String t) {
    return t.equals("+") || t.equals("-") || t.equals("*") || t.equals("/");
}
作者：windliang
链接：https://leetcode-cn.com/problems/basic-calculator/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by--47/
```

225 Implement Stack Using Queues

225. Implement Stack using Queues

难度 简单 195

Implement the following operations of a stack using queues.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- empty() -- Return whether the stack is empty.

Example:

```
MyStack stack = new MyStack();

stack.push(1);
stack.push(2);
stack.top();   // returns 2
stack.pop();   // returns 2
stack.empty(); // returns false
```

Notes:

- You must use only standard operations of a queue -- which means only `push to back`, `peek/pop from front`, `size`, and `is empty` operations are valid.
- Depending on your language, queue may not be supported natively. You may simulate a queue by using a list or deque (double-ended queue), as long as you use only standard operations of a queue.
- You may assume that all operations are valid (for example, no pop or top operations will be called on an empty stack).

执行结果: 通过 显示详情 >

执行用时: 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 37.4 MB , 在所有 Java 提交中击败了 7.41% 的用户

炫耀一下:

```java
int num;
Deque<Integer> queue;

/** Initialize your data structure here. */
public MyStack() {
    num = 0;
    queue = new ArrayDeque<>();
}

/** Push element x onto stack. */
public void push(int x) {
```

```
    queue.addLast(x);
    num++;
}

/** Removes the element on top of the stack and returns that element. */
public int pop() {
    num--;
    return queue.removeLast();
}

/** Get the top element. */
public int top() {
    return queue.getLast();
}

/** Returns whether the stack is empty. */
public boolean empty() {
    return num == 0;
}

```

226 Invert Binary Tree

```
    public TreeNode invertTree(TreeNode root) {
        if(root == null)
            return null;

        TreeNode temp = root.left;
        root.left = root.right;
        root.right = temp;

        invertTree(root.left);
        invertTree(root.right);

        return root;
    }
```

227 Basic calculator

执行结果： 通过 显示详情 >

执行用时： 52 ms ，在所有 C++ 提交中击败了 5.66% 的用

内存消耗： 24 MB ，在所有 C++ 提交中击败了 5.02% 的用

炫耀一下：

```cpp

class Solution {
public:
    int calculate(string s) {
        vector<string> vec = getRPN(s);
        int res = evl(vec);

        return res;
    }

    vector<string> getRPN(string& s){
        vector<string> res;
        stack<string> myStack;
        int num = -1;

        for(int i = 0; i < s.size(); i++){
            if(s[i] == ' ')
                continue;

            if(s[i] >= '0' && s[i] <= '9'){
                if(num == -1)
                    num = s[i] - '0';
                else{
                    num *= 10;
                    num += s[i] - '0';
                }
            }else{
                if(num != -1){
                    res.push_back(to_string(num));
                    num = -1;
                }

                if(s[i] == '(')
                    myStack.push("(");
                else if(s[i] == ')'){
                    while(!myStack.empty()){
                        if(myStack.top() == "("){
                            myStack.pop();
                            break;
                        }

```

```
                        res.push_back(myStack.top());       myStack.pop();
                    }
                }else{
                    while(!myStack.empty()){
                        if(myStack.top() == "("){
                            break;
                        }

                        if(getPriority(myStack.top()[0]) <= getPriority(s[i])){
                            res.push_back(myStack.top());       myStack.pop();
                        }else{
                            break;
                        }
                    }

                    myStack.push(string(1, s[i]));
                }

            }
        }

        if(num != -1)
            myStack.push(to_string(num));
        while(!myStack.empty()){
            res.push_back(myStack.top()); myStack.pop();
        }

        return res;
    }



    int getPriority(char ch){
        if(ch == '+' || ch == '-')
            return 1;
        else
            return 0;
    }

    int evl(vector<string>& vec) {
        stack<int> myStack;
        int res = 0;
        for(string& str : vec){
            if(str[0] >= '0' && str[0] <= '9'){ //number
                myStack.push(parseInt(str));
            }else{  //ops
                int op1 = myStack.top();    myStack.pop();
                int op2 = myStack.top();    myStack.pop();

```

```
                int temp = 0;
                if(str == "+")
                    temp =  op1 + op2;
                else if(str == "-")
                    temp =  op2 - op1;
                else if(str == "*")
                    temp =  op1 * op2;
                else
                    temp =  op2 / op1;

                myStack.push(temp);
            }
        }

        return myStack.top();
    }

    int parseInt(string s){
        int res = 0;
        for(int i = 0; i < s.size(); i++){
            if(res > INT_MAX / 10 || (res == INT_MAX / 10 && s[i] - '0' > INT_MAX % 10))
                return INT_MAX;

            res *= 10;
            res += s[i] - '0';
        }

        return res;
    }
};
```

228 Summary Ranges

228. Summary Ranges

难度 中等 53

Given a sorted integer array without duplicates, return the summary of its ranges.

Example 1:

```
Input:  [0,1,2,4,5,7]
Output: ["0->2","4->5","7"]
Explanation: 0,1,2 form a continuous range; 4,5 form a continuous
range.
```

Example 2:

```
Input:  [0,2,3,4,6,8,9]
Output: ["0","2->4","6","8->9"]
Explanation: 2,3,4 form a continuous range; 8,9 form a continuous
range.
```

执行结果: 通过 显示详情 >

执行用时: 5 ms , 在所有 Java 提交中击败了 84.49% 的用户

内存消耗: 37.4 MB , 在所有 Java 提交中击败了 16.67% 的用户

炫耀一下:

```java
public List<String> summaryRanges(int[] nums) {
    List<String> res = new ArrayList<>();
    for(int i = 0; i < nums.length; )
    {
        int index = i;
        while(index + 1< nums.length && nums[index] + 1 == nums[index+1])
            index++;
        if(index == i)
            res.add(nums[i] + "");
        else
        {
            StringBuilder sb = new StringBuilder();
            sb.append(nums[i]).append("->").append(nums[index]);
            res.add(sb.toString());
        }
        i = index + 1;
    }

    return res;
```

```
}
```

229 Majority Element II 摩尔投票法

229. Majority Element II

难度 中等 215

Given an integer array of size n, find all elements that appear more than `⌊ n/3 ⌋` times.

Note: The algorithm should run in linear time and in O(1) space.

Example 1:

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

Example 2:

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

执行结果: 通过 显示详情 >

执行用时: 20 ms , 在所有 Java 提交中击败了 7.52% 的用户

内存消耗: 42.4 MB , 在所有 Java 提交中击败了 11.11% 的用户

炫耀一下:

```java
/*
  内存消耗为 O(n) 不符合要求
*/
public List<Integer> majorityElement(int[] nums) {
    List<Integer> res = new ArrayList<>();
    HashMap<Integer, Integer> map = new HashMap<>();
    for(int i = 0; i < nums.length; i++)
    {
        map.put(nums[i], map.getOrDefault(nums[i], 0) + 1);
        if(map.get(nums[i]) > nums.length / 3)
            if(!res.contains(nums[i]))
                res.add(nums[i]);
    }
    return res;
}
```

```java
/*
  看这个题之前，首先要看一道简单题， 就是求众数 169
  "该元素出现频率 >n / 2"
  之所以摩尔投票法可以成功的原因是因为
    投票法是遇到相同的则票数 + 1， 遇到不同的则票数 - 1
    且"多数元素" 的个数 - 其余元素的个数综合， 结果肯定 >= 1

  类似于， "多数元素" - "其余元素" 两两抵消， 抵消到最后， 肯定至少还剩余 1个多数元素
*/

public int majorityElememnt(int[] nums)
{
    int cand_num = nums[0], count = 1;
    for(int i = 1; i < nums.length; i++)
    {
        if(cand_num == nums[i])
            count++;
        else if(--count == 0)
        {
            cand_num = nums[i];
            count = 1;
        }
    }
    return cand_num;
}
```

```java
/*
  KWOK NOTE:
  本题说要找出多个众数，其中众数的出现频率一定是大于n / 3
  那么推论是，数量最多两个，原因是
    如果众数有3个， 每个出现频率大于 n / 3，那么最后的数组个数为 3 * (more than n /3) > n
    不符合题意，因此众数的个数只可能是k - 1 个， 其中k 为 n / k的k

  算法具体步骤
    if      如果投A， A++
    else if 如果投B， B++
    else 如果投其他人， 检查A B 票数是否 == 0，
      如果是，则当前元素成为新的候选元素
      如果不是， 那么A B 的候选人票数全部减1

    最后的情况是 有两个人大于n/3，有1个人大于n/3，没有众数
    同时在最后，还要遍历两个候选人，确保他们满足 > n / 3 条件，找出两个候选人的具体票数
      因为题目并没有保证一定有

*/
class Solution {
```

```java
    public List<Integer> majorityElement(int[] nums) {
        List<Integer> res = new ArrayList<>();
        if (nums == null || nums.length == 0)
            return res;

        // 定义两个候选者和它们的票数
        int cand1 = 0,cand2 = 0;
        int cnt1 = 0, cnt2 = 0;
        // 投票过程
        for (int num : nums) {
            if (num == cand1) {
                cnt1++;
                // 一遍遍历，如果你不想写continue，你写多个else if也可以
                continue;
            }
            else if (num == cand2) {
                cnt2++;
                continue;
            }
            // 既不是cand1也不是cand2，如果cnt1为0，那它就去做cand1
            else if (cnt1 == 0) {
                cand1 = num;
                cnt1++;
                continue;
            }
            // 如果cand1的数量不为0但是cand2的数量为0，那他就去做cand2
            else if (cnt2 == 0) {
                cand2 = num;
                cnt2++;
                continue;
            }
            // 如果cand1和cand2的数量都不为0，那就都-1
            cnt1--;
            cnt2--;
        }

        // 检查两个票数符不符合
        cnt1 = cnt2 = 0;
        for (int num : nums) {
            if (num == cand1)
                cnt1++;

            else if (num == cand2)
                // 这里一定要用else if
                // 因为可能出现[0,0,0]这种用例，导致两个cand是一样的，写两个if结果就变为[0,0]了
                cnt2++;

        }
```

```
        int n = nums.length;
        if (cnt1 > n / 3)
            res.add(cand1);

        if (cnt2 > n / 3)
            res.add(cand2);

        return res;
    }
}

作者：jerry_nju
    链接：https://leetcode-cn.com/problems/majority-element-ii/solution/169ti-sheng-ji-ban-xiang-jie-zhu-xing-jie-shi-tong/

//revised version
public List<Integer> majorityElement(int[] nums) {
    ArrayList<Integer> res = new ArrayList<>();

    if(nums.length == 0 || nums == null)        return res;
    if(nums.length == 1){
        res.add(nums[0]);
        return res;
    }

    int cand_1 = 0, cand_2 = 0;
    int cnt_1 = 0, cnt_2 = 0;

    for(int i = 0; i < nums.length; i++)
    {
        if      (cand_1 == nums[i])   cnt_1++;
        else if (cand_2 == nums[i])   cnt_2++;
        else if (cnt_1 == 0)          {cand_1 = nums[i];    cnt_1++;}
        else if (cnt_2 == 0)          {cand_2 = nums[i];    cnt_2++;}
        else    {cnt_1--;   cnt_2--;}
    }

    cnt_1 = 0;  cnt_2 = 0;
    for(int num : nums)
        if(cand_1 == num)
            cnt_1++;
        else if(cand_2 == num)
            cnt_2++;

    if(cnt_1 > nums.length / 3)     res.add(cand_1);
    if(cnt_2 > nums.length / 3)     res.add(cand_2);

    return res;
}
```

231 Power of Two

231. Power of Two

难度 简单 218

Given an integer, write a function to determine if it is a power of two.

Example 1:

```
Input: 1
Output: true
Explanation: 2^0 = 1
```

Example 2:

```
Input: 16
Output: true
Explanation: 2^4 = 16
```

Example 3:

```
Input: 218
Output: false
```

```java
//通过全部测试案例，就是一个位运算
public boolean isPowerOfTwo(int n) {
    long nn = n;
    if(n == 0)      return false;
    if(n == Integer.MIN_VALUE)      return false;
    int count = 0;
    for(int i = 0; i < 32; i++)
    {
        long res = (n & 1);
        if(res == 0)    {}
        else            count++;

        if(count >= 2)      return false;
        n >>= 1;
    }
    return true;
}
```

232 Implement Queue using Stacks

232. Implement Queue using Stacks

难度 简单 197

Implement the following operations of a queue using stacks.

- push(x) -- Push element x to the back of queue.
- pop() -- Removes the element from in front of queue.
- peek() -- Get the front element.
- empty() -- Return whether the queue is empty.

Example:

```
MyQueue queue = new MyQueue();

queue.push(1);
queue.push(2);
queue.peek();  // returns 1
queue.pop();   // returns 1
queue.empty(); // returns false
```

Notes:

- You must use only standard operations of a stack -- which means only `push to top`, `peek/pop from top`, `size`, and `is empty` operations are valid.
- Depending on your language, stack may not be supported natively. You may simulate a stack by using a list or deque (double-ended queue), as long as you use only standard operations of a stack.
- You may assume that all operations are valid (for example, no pop or peek operations will be called on an empty queue).

```java
import java.util.Stack;

public class MyQueue {

    private Stack<Integer> stackPush;
    private Stack<Integer> stackPop;

    /**
     * Initialize your data structure here.
     */
    public MyQueue() {
        stackPush = new Stack<>();
        stackPop = new Stack<>();
    }

    /**
     * Push element x to the back of queue.
```

```
     */
    public void push(int x) {
        stackPush.push(x);
    }

    /**
     * 辅助方法：一次性将 stackPush 里的所有元素倒入 stackPop
     * 注意：1、该操作只在 stackPop 里为空的时候才操作，否则会破坏出队入队的顺序
     * 2、在 peek 和 pop 操作之前调用该方法
     */
    private void shift() {
        if (stackPop.isEmpty()) {
            while (!stackPush.isEmpty()) {
                stackPop.push(stackPush.pop());
            }
        }
    }

    /**
     * Removes the element from in front of queue and returns that element.
     */
    public int pop() {
        shift();
        if (!stackPop.isEmpty()) {
            return stackPop.pop();
        }
        throw new RuntimeException("队列里没有元素");
    }

    /**
     * Get the front element.
     */
    public int peek() {
        shift();
        if (!stackPop.isEmpty()) {
            return stackPop.peek();
        }
        throw new RuntimeException("队列里没有元素");
    }

    /**
     * Returns whether the queue is empty.
     */
    public boolean empty() {
        return stackPush.isEmpty() && stackPop.isEmpty();
    }
}

/**
```

```
 * Your MyQueue object will be instantiated and called as such:
 * MyQueue obj = new MyQueue();
 * obj.push(x);
 * int param_2 = obj.pop();
 * int param_3 = obj.peek();
 * boolean param_4 = obj.empty();
 */

作者：liweiwei1419
链接：https://leetcode-cn.com/problems/implement-queue-using-stacks/solution/shi-yong-liang-ge-zhan-yi-ge-zhuan-men-ru-dui-yi-g/
```

233 Number of Digit One 未完成

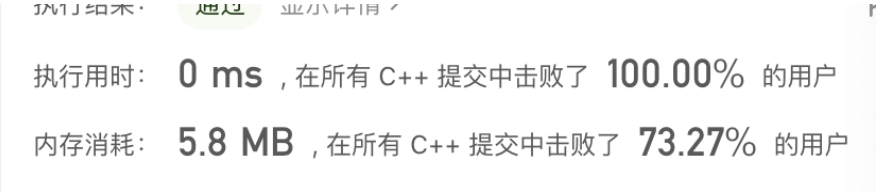
233. Number of Digit One

难度 困难 139

Given an integer n, count the total number of digit 1 appearing in all non-negative integers less than or equal to n.

Example:

```
Input: 13
Output: 6
Explanation: Digit 1 occurred in the following numbers: 1, 10, 11, 12, 13.
```




```cpp
//source : https://leetcode.wang/leetcode-233-Number-of-Digit-One.html
class Solution {
public:
    int countDigitOne(int n) {
        int count = 0;

        //xyzdabc
        for(int k = 1; k <= n; k *= 10){
            int abc = n % k;
            int xyzd = n / k;
            int d = xyzd % 10;
            int xyz = xyzd / 10;
            count += xyz * k;
```

```
            if(d > 1){
                count += k;
            }else if(d == 1){
                count += abc + 1;
            }

            if(xyz == 0)
                break;

        }

        return count;
    }
};
```

```
class Solution {
    public int countDigitOne(int n) {
        //求每个位的数字所用
        int index = 1;
        //记录1的个数
        int count = 0;
        int high = n,cur = 0,low = 0;
        //由于high = n /(index*10) 中index *10 很容易越位
        //特修改如下
        while(high > 0){
            high /= 10;
            cur = (n / index) % 10;
            low = n - (n / index) * index;
            //以下是计算的公式
            if(cur == 0) count += high * index;
            if(cur == 1) count += high * index + low + 1;
            if(cur > 1) count += (high+1) * index;
            index *= 10;
        }
        return count;
    }
}

作者：xyx1273930793
链接：https://leetcode-cn.com/problems/number-of-digit-one/solution/java100fu-si-lu-shuo-ming-by-xyx1273930793/
```

234 Palindrome Linked List

234. Palindrome Linked List

难度 简单 559

Given a singly linked list, determine if it is a palindrome.

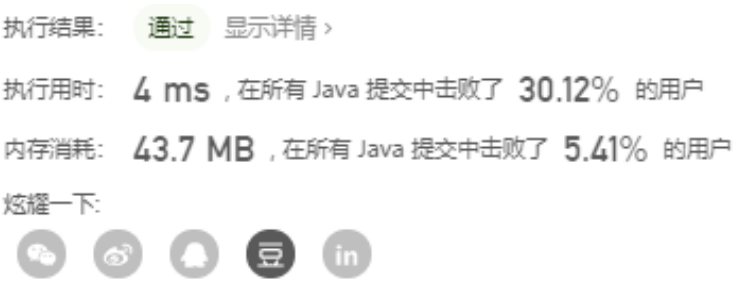
Example 1:

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

Example 2:

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

Follow up:
Could you do it in O(n) time and O(1) space?




```java
/*
  不足点是用了O(n)的额外空间
*/
public boolean isPalindrome(ListNode head) {
    List<Integer> list = new ArrayList<>();
    ListNode cur = head;
    if(head == null)        return true;
    while(cur != null)
    {
        list.add(cur.val);
        cur = cur.next;
    }
    int left = 0, right = list.size()-1;
    while(left < right)
    {
        if(!list.get(left).equals(list.get(right)))
            return false;
        left++;
```

```
        right--;
    }
    return true;

}
```


```
/*
  快慢指针走到一半，然后反转后半段链表进行判断
  O(1) space
*/
public boolean isPalindrome(ListNode head) {
    ListNode slow = head;
    ListNode fast = head;
    if(head == null)        return true;

    //拿到中间位置
    while(fast != null && fast.next != null)
    {
        slow = slow.next;
        fast = fast.next.next;
    }
  //slow 此时指向的位置，就是链表的中间部分

    //-----------反转链表开始-----------------
    ListNode cur = slow;
    ListNode temp = null, pre = null;

    while(cur != null)
    {
        temp = cur.next;
        cur.next = pre;
        pre = cur;
        cur = temp;
    }
   //-----------反转链表结束-----------------

    //开始判断
    while(pre != null)
    {
        if(pre.val != head.val)
            return false;
```

```
            pre = pre.next;
            head = head.next;
        }
        return true;
    }
```

236 Lowest Common Ancestor of a Binary Tree 未完成

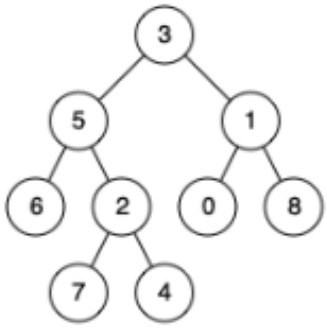
236. Lowest Common Ancestor of a Binary Tree

难度 中等 648

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**)."

Given the following binary tree: root = [3,5,1,6,2,0,8,null,null,7,4]



Example 1:

```
Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1
Output: 3
Explanation: The LCA of nodes 5 and 1 is 3.
```

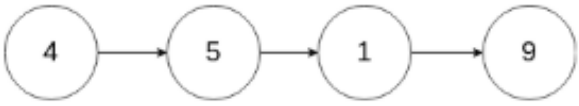
237 Delete Node in a Linked List

237. Delete Node in a Linked List

难度 简单 715

Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Given linked list -- head = [4,5,1,9], which looks like following:



Example 1:

```
Input: head = [4,5,1,9], node = 5
Output: [4,1,9]
Explanation: You are given the second node with value 5, the linked
list should become 4 -> 1 -> 9 after calling your function.
```

Example 2:

```
Input: head = [4,5,1,9], node = 1
Output: [4,5,9]
Explanation: You are given the third node with value 1, the linked
list should become 4 -> 5 -> 9 after calling your function.
```

Note:

- The linked list will have at least two elements.
- All of the nodes' values will be unique.
- The given node will not be the tail and it will always be a valid node of the linked list.
- Do not return anything from your function.

执行结果: 通过 显示详情 >

执行用时: 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 39.2 MB , 在所有 Java 提交中击败了 6.52% 的用户

炫耀一下:

```java
public void deleteNode(ListNode node) {
    if(node == null)        return;

    ListNode cur = node;
    while(cur != null && cur.next != null)
    {
```

```
        if(cur.next.next == null)
        {
            cur.val = cur.next.val;
            cur.next = null;
            return;
        }

        cur.val = cur.next.val;
        cur = cur.next;

    }
```

```
/*
  更加简洁的办法
*/
class Solution {
    public void deleteNode(ListNode node) {

        node.val = node.next.val;

        node.next = node.next.next;

    }
}
```

238 Product of Array Except Self

238. Product of Array Except Self

难度 中等 525

Given an array `nums` of n integers where n > 1, return an array `output` such that `output[i]` is equal to the product of all the elements of `nums` except `nums[i]`.

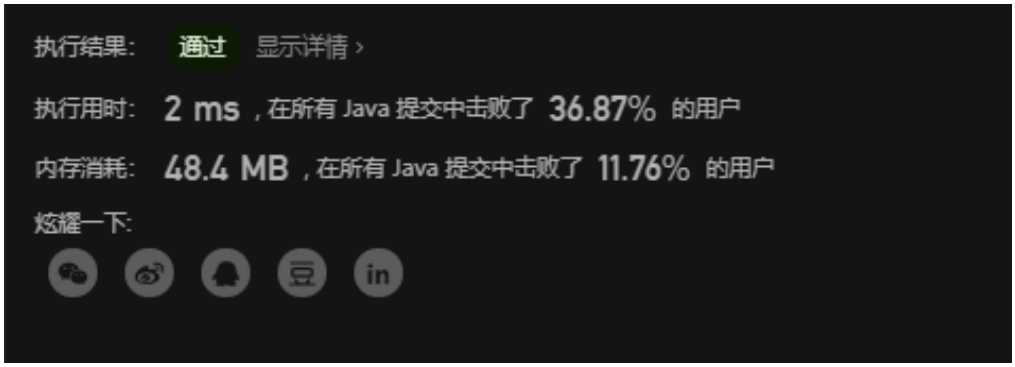
Example:

```
Input:  [1,2,3,4]
Output: [24,12,8,6]
```

Constraint: It's guaranteed that the product of the elements of any prefix or suffix of the array (including the whole array) fits in a 32 bit integer.

Note: Please solve it **without division** and in O(n).

Follow up:
Could you solve it with constant space complexity? (The output array **does not** count as extra space for the purpose of space complexity analysis.)

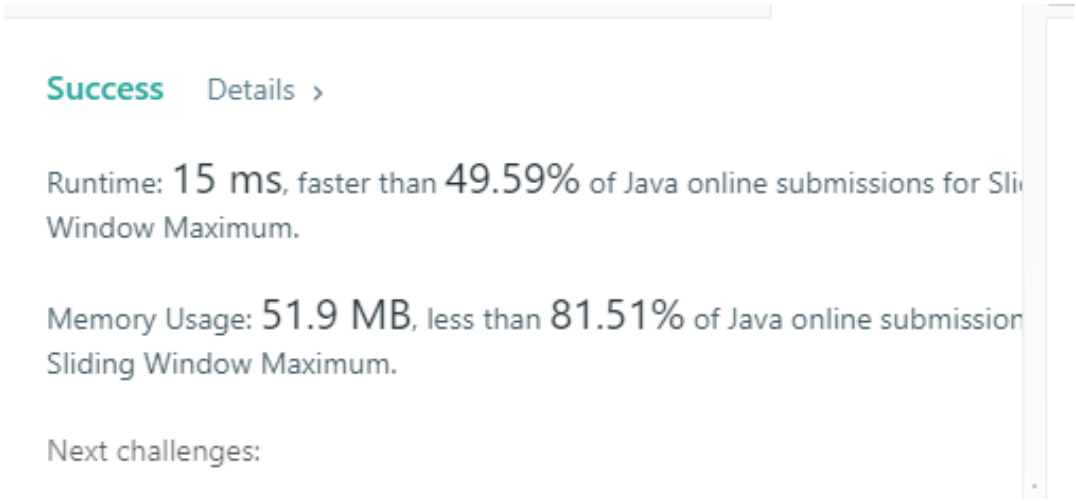



```java
/*
  讲个题外话，看到题目描述中说prefix 和suffix 就算是提示了
  所以才能解出这个题
*/
public int[] productExceptSelf(int[] nums) {
    int[] prefix = new int[nums.length];
    int[] suffix = new int[nums.length];

    for(int i = 0; i < nums.length; i++)
    {
        if(i == 0)       prefix[i] = nums[0];
        else             prefix[i] = prefix[i-1] * nums[i];
    }
    for(int i = nums.length - 1; i >= 0; i--)
    {
        if(i == nums.length -1)       suffix[i] = nums[nums.length-1];
        else                         suffix[i] = suffix[i+1] * nums[i];
    }

    int[] Output = new int[nums.length];
```

```
    for(int i = 0; i < nums.length; i++)
    {
        if      (i == 0)              Output[i] = suffix[1];
        else if (i == nums.length-1)  Output[i] = prefix[nums.length-2];
        else                          Output[i] = prefix[i-1] * suffix[i+1];
    }
    return Output;

}
```

```
//符合空间复杂度的方法
public int[] productExceptSelf(int[] nums) {
    int[] res   = new int[nums.length];

    for(int i = 0; i < nums.length; i++)
    {
        if(i == 0)
            res[0] = 1;
        else
            res[i] = res[i-1] * nums[i-1];
    }

    int right = 1;
    for(int i = nums.length - 1; i >= 0; i --)
    {
        res[i] = res[i] * right;
        right *= nums[i];
    }
    return res;

}
```

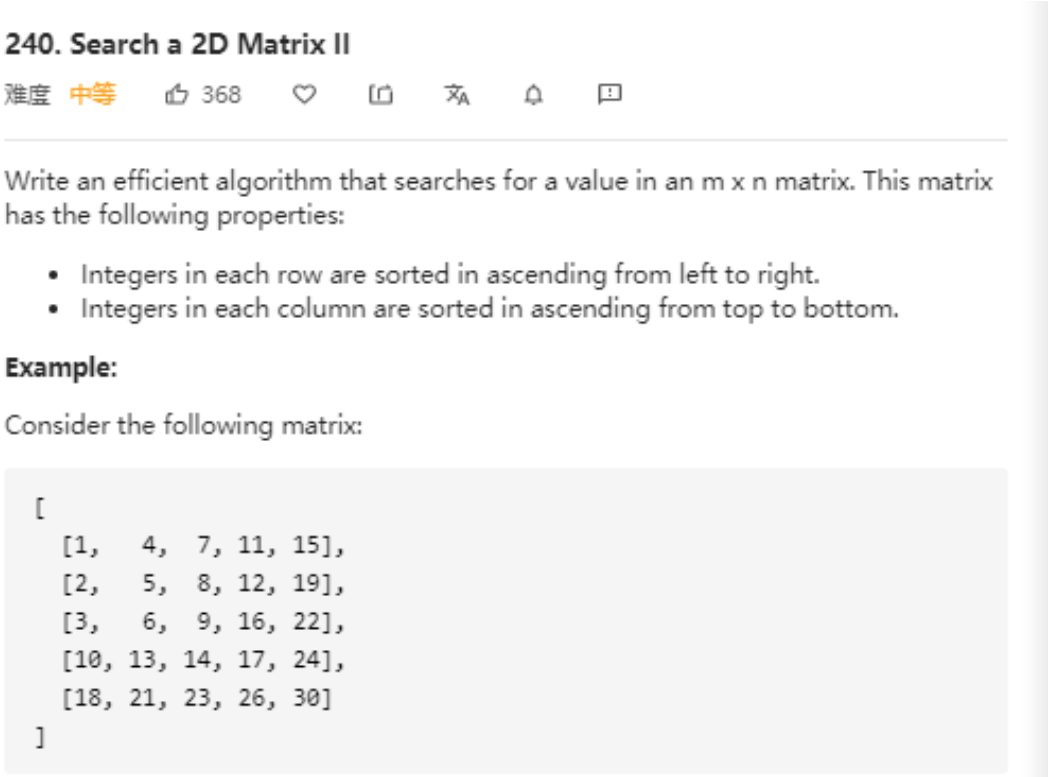
239 Sliding Window Maximum




```java
//单调队列解法

/*
  郭郭版本的单调队列解法
*/
    public int[] maxSlidingWindow(int[] nums, int k) {
        Deque<Integer> queue = new ArrayDeque<>();
        int index = 0;
        int[] res = new int[nums.length - k + 1];
        for(int i = 0; i < nums.length; i++){
            while(!queue.isEmpty() && nums[i] > queue.peekLast()){
                queue.pollLast();
            }
            queue.addLast(nums[i]);


            if(i >= k - 1) {
                res[index++] = queue.peekFirst();
                if(i >= k - 1 && nums[i - k + 1] == queue.peekFirst())
                    queue.removeFirst();
            }
        }

        return res;
    }


//用pq 的话，时间复杂度太高，超不过一个案例
public int[] maxSlidingWindow(int[] nums, int k) {
    PriorityQueue<Integer> pq = new PriorityQueue<>((o1, o2) -> o2- o1);
    int[] output = new int[nums.length - k + 1];

    for(int i = 0; i < k - 1; i++)
        pq.add(nums[i]);

    for(int i = k - 1; i < nums.length; i++)
```

```
    {
        pq.add(nums[i]);
        output[i - k + 1] = pq.peek();
        pq.remove(nums[i - (k - 1)]);
    }

    return output;
}
```

240 Search a 2D Matrix II



240. Search a 2D Matrix II

难度 中等 368

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 in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Example:

Consider the following matrix:

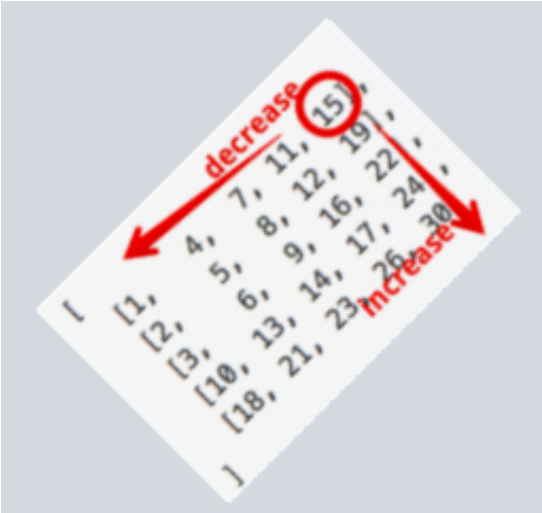
```
[
  [1,   4,  7, 11, 15],
  [2,   5,  8, 12, 19],
  [3,   6,  9, 16, 22],
  [10, 13, 14, 17, 24],
  [18, 21, 23, 26, 30]
]
```

```
public boolean searchMatrix(int[][] matrix, int target) {
    if(matrix.length == 0 || matrix[0].length == 0)      return false;
    for(int i = 0; i < matrix.length; i++)
    {
        if(matrix[i][0] > target)   break;

        if(matrix[i][matrix[i].length-1] < target) continue;

        int col = binarySearch(matrix[i], target);

        if(col != -1)       return true;
    }
```

```
    return false;
}

//单独对一行进行二分查找
private int binarySearch(int[] nums, int target)
{
    int left = 0;
    int right = nums.length -1;

    while(left <= right)
    {
        int mid = (left + right)/ 2;
        if(nums[mid] == target)
            return mid;
        else if (nums[mid] > target)
            right = mid - 1;
        else
            left = mid + 1;
    }
    return -1;
}
参考
    作者：windliang
    链接：https://leetcode-cn.com/problems/search-a-2d-matrix-ii/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by-5-4/
```

```
/*
  解法二：
  注意题目条件，每一行被sorted，  每一列被sorted
  我们从右上角开始搜索
    如果target > 当前元素 ，那么这一行就被排除掉
    如果target < 当前元素，  那么这一列都被排除掉，因为列也被sorted
  时间复杂度为O(M+N)，十分类似BST where 往左变小，往右变大
*/
public boolean searchMatrix(int[][] matrix, int target) {
    if(matrix.length == 0 || matrix[0].length == 0)   return false;
    int row = matrix.length;
    int column = matrix[0].length;
    int curRow = 0, curCol = column-1;
    while(curRow != row && curCol != -1)
    {
        if     (matrix[curRow][curCol] == target)        return true;
        else if(matrix[curRow][curCol] > target)         curCol--;
        else if(matrix[curRow][curCol] < target)         curRow++;
    }

    return false;
```

```
22  }
```



241 Different Ways to Add Parentheses 递归的又一应用

241. Different Ways to Add Parentheses

难度 中等 211

Given a string of numbers and operators, return all possible results from computing all the different possible ways to group numbers and operators. The valid operators are `+`, `-` and `*`.

Example 1:

```
Input: "2-1-1"
Output: [0, 2]
Explanation:
((2-1)-1) = 0
(2-(1-1)) = 2
```

Example 2:

```
Input: "2*3-4*5"
Output: [-34, -14, -10, -10, 10]
Explanation:
(2*(3-(4*5))) = -34
((2*3)-(4*5)) = -14
((2*(3-4))*5) = -10
(2*((3-4)*5)) = -10
(((2*3)-4)*5) = 10
```

```java
/*
  整体思路:
    * 先对全是数字的进行处理
    * 对每一个运算符的左右两边进行处理
*/
public List<Integer> diffWaysToCompute(String input) {
    if (input.length() == 0)
        return new ArrayList<>();

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

    //----------------如果该字符串是数字的处理----------------------------------------
    int index = 0;
    int num = 0;
    while (index < input.length() && !isOperation(input.charAt(index)))
        num = num * 10 + input.charAt(index++) - '0';

    //将全数字的情况直接返回
    if (index == input.length()) {
        result.add(num);
        return result;
    }
    //---------------结束全是数字的处理阶段-------------------------------

    for (int i = 0; i < input.length(); i++) {
        //通过运算符将字符串分成两部分
        if (isOperation(input.charAt(i))) {
            List<Integer> result1 = diffWaysToCompute(input.substring(0, i));
            List<Integer> result2 = diffWaysToCompute(input.substring(i + 1));
            //将两个结果依次运算
            for (int j = 0; j < result1.size(); j++) {
                for (int k = 0; k < result2.size(); k++) {
                    char op = input.charAt(i);
                    result.add(caculate(result1.get(j), op, result2.get(k)));
                }
            }
        }
    }
    return result;
}

private int caculate(int num1, char c, int num2) {
    switch (c) {
        case '+':
            return num1 + num2;
        case '-':
            return num1 - num2;
        case '*':
            return num1 * num2;
```

```
    }
    return -1;
}

private boolean isOperation(char c) {
    return c == '+' || c == '-' || c == '*';
}

作者：windliang
链接：https://leetcode-cn.com/problems/different-ways-to-add-parentheses/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by-5-5/
```

```
/*
  优化递归，  使用备忘录
*/
//添加一个 map
HashMap<String,List<Integer>> map = new HashMap<>();
public List<Integer> diffWaysToCompute(String input) {
    if (input.length() == 0) {
        return new ArrayList<>();
    }
    //如果已经有当前解了，直接返回
    if(map.containsKey(input)){
        return map.get(input);
    }
    List<Integer> result = new ArrayList<>();

    int num = 0;
    int index = 0;
    while (index < input.length() && !isOperation(input.charAt(index))) {
        num = num * 10 + input.charAt(index) - '0';
        index++;
    }
    if (index == input.length()) {
        result.add(num);
        //存到 map
        map.put(input, result);
        return result;
    }
    for (int i = 0; i < input.length(); i++) {
        if (isOperation(input.charAt(i))) {
            List<Integer> result1 = diffWaysToCompute(input.substring(0, i));
            List<Integer> result2 = diffWaysToCompute(input.substring(i + 1));
            for (int j = 0; j < result1.size(); j++) {
                for (int k = 0; k < result2.size(); k++) {
                    char op = input.charAt(i);
                    result.add(caculate(result1.get(j), op, result2.get(k)));
```

```
                }
            }
        }
    }
     //存到 map
    map.put(input, result);
    return result;
}

private int caculate(int num1, char c, int num2);
    //same
private boolean isOperation(char c) ;
   //SAME


作者：windliang
链接：https://leetcode-cn.com/problems/different-ways-to-add-parentheses/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by-5-5/
```

242 Valid Anagram

242. Valid Anagram

难度 简单 216

Given two strings s and t , write a function to determine if t is an anagram of s.

Example 1:

```
Input: s = "anagram", t = "nagaram"
Output: true
```

Example 2:

```
Input: s = "rat", t = "car"
Output: false
```

Note:
You may assume the string contains only lowercase alphabets.

Follow up:
What if the inputs contain unicode characters? How would you adapt your solution to such case?


```java
public boolean isAnagram(String s, String t) {
    if(s.length() != t.length())        return false;
    HashMap<Character, Integer> mapS = new HashMap<>();
    HashMap<Character, Integer> mapT = new HashMap<>();

    for(int i = 0; i < s.length(); i++)
    {
        mapS.put(s.charAt(i), mapS.getOrDefault(s.charAt(i), 0) + 1);
        mapT.put(t.charAt(i), mapT.getOrDefault(t.charAt(i), 0) + 1);
    }

    if(mapS.equals(mapT))       return true;
    else                        return false;
}
```

243 Shortest Word Distance

243. Shortest Word Distance

难度 简单 22

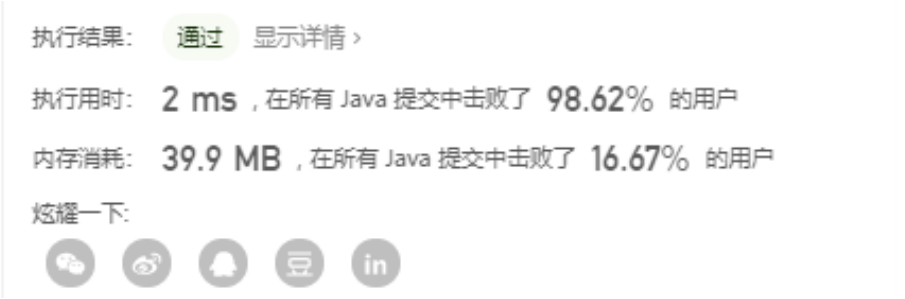
Given a list of words and two words word1 and word2, return the shortest distance between these two words in the list.

Example:
Assume that words = `["practice", "makes", "perfect", "coding", "makes"]`.

```
Input: word1 = “coding”, word2 = “practice”
Output: 3
```

```
Input: word1 = "makes", word2 = "coding"
Output: 1
```

Note:
You may assume that word1 **does not equal to** word2, and word1 and word2 are both in the list.




```java
/*
  典型的滑动窗口题目
*/
public int shortestDistance(String[] words, String word1, String word2) {
    int left = 0, right = 0;
    int res = Integer.MAX_VALUE;
    while(left < words.length && right < words.length)
    {
        while(left < words.length && !words[left].equals(word1))
            left++;
        while(right < words.length && !words[right].equals(word2))
            right++;
        if(left == words.length || right == words.length)    break;

        if(right - left == 0)       continue;
        else
            res = Math.min(res, Math.abs(right - left));
        if(left < right)    left++;
        else                right++;
    }

    return res;
}
```

244 Shortest Word Distance II

244. Shortest Word Distance II

难度 中等 15

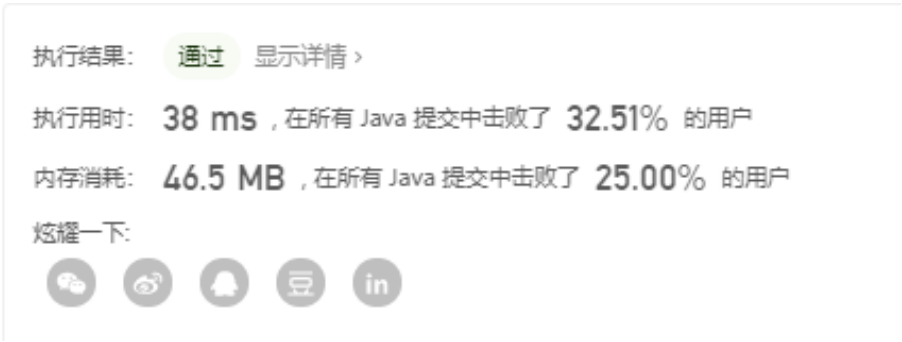
Design a class which receives a list of words in the constructor, and implements a method that takes two words word1 and word2 and return the shortest distance between these two words in the list. Your method will be called repeatedly many times with different parameters.

Example:
Assume that words = `["practice", "makes", "perfect", "coding", "makes"]`.

```
Input: word1 = “coding”, word2 = “practice”
Output: 3
```

```
Input: word1 = "makes", word2 = "coding"
Output: 1
```

Note:
You may assume that word1 **does not equal to** word2, and word1 and word2 are both in the list.




```java
private HashMap<String, List<Integer>> map;
public WordDistance(String[] words) {
    map = new HashMap<>();
    for(int i = 0; i < words.length; i++)
    {
        if(!map.containsKey(words[i]))
            map.put(words[i], new ArrayList<>());

        map.get(words[i]).add(i);
    }
}

public int shortest(String word1, String word2) {
    int res = Integer.MAX_VALUE;
    List<Integer> jar1 = map.get(word1);
    List<Integer> jar2 = map.get(word2);
    for(Integer i : jar1)
        for(Integer j : jar2)
```

```
            res = Math.min(res, Math.abs(i-j));

    return res;
}
```

245 Shortest word distanceIII

245. Shortest Word Distance III

难度 中等 15

Given a list of words and two words word1 and word2, return the shortest distance between these two words in the list.

word1 and word2 may be the same and they represent two individual words in the list.

Example:
Assume that words = `["practice", "makes", "perfect", "coding", "makes"]`.

```
Input: word1 = “makes”, word2 = “coding”
Output: 1
```

```
Input: word1 = "makes", word2 = "makes"
Output: 3
```

Note:
You may assume word1 and word2 are both in the list.

执行结果: 通过 显示详情 >

执行用时: 1 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 40 MB , 在所有 Java 提交中击败了 16.67% 的用户

炫耀一下:

```
public int shortestWordDistance(String[] words, String word1, String word2) {
    int left = 0, right = 0;
    int res = Integer.MAX_VALUE;
    while(left < words.length && right < words.length)
    {
        while(left < words.length && !words[left].equals(word1))
            left++;
```

```
        while(right < words.length && !words[right].equals(word2))
            right++;
        if(left == words.length || right == words.length)    break;

        if(right - left == 0)       {}
        else
            res = Math.min(res, Math.abs(right - left));

        if(left < right)    left++;
        else                right++;
    }

    return res;

}
```

246 Strobogrammatic number

246. Strobogrammatic Number

难度 简单 14

A strobogrammatic number is a number that looks the same when rotated 180 degrees (looked at upside down).

Write a function to determine if a number is strobogrammatic. The number is represented as a string.

Example 1:

```
Input: num = "69"
Output: true
```

Example 2:

```
Input: num = "88"
Output: true
```

Example 3:

```
Input: num = "962"
Output: false
```

Example 4:

```
Input: num = "1"
Output: true
```

执行结果: 通过 显示详情 >

执行用时: 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

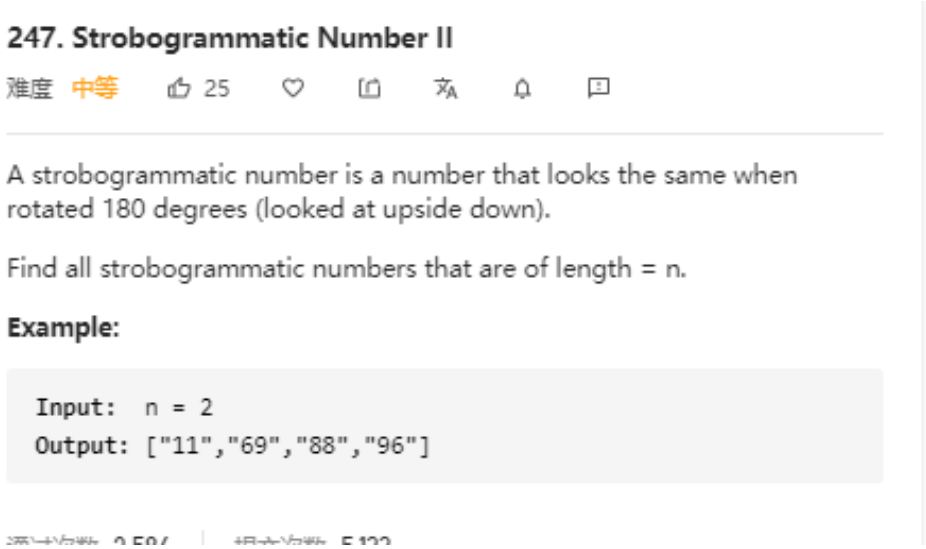
内存消耗: 37.7 MB , 在所有 Java 提交中击败了 100.00% 的用户

炫耀一下:

```java
public boolean isStrobogrammatic(String num) {
    HashMap<Integer, Integer> map = new HashMap<>();
    map.put(6,9);
    map.put(9,6);
    map.put(8,8);
    map.put(1,1);
    map.put(0,0);

```

```
    StringBuilder sb = new StringBuilder();
    for(int i = 0; i < num.length(); i++)
    {
        if(map.containsKey(num.charAt(i) - '0'))
            sb.append(map.get(num.charAt(i)- '0'));
        else
            return false;
    }

    return sb.reverse().toString().equals(num);
}
```

247 Strobogrammatic Number II 递归的应用



247. Strobogrammatic Number II

难度 中等 25

A strobogrammatic number is a number that looks the same when rotated 180 degrees (looked at upside down).

Find all strobogrammatic numbers that are of length = n.

Example:

```
Input:  n = 2
Output: ["11","69","88","96"]
```

```
## 代码实现
class Solution{
    public List<String> findStrobogrammatic(int n) {
        return helper(n,n);
    }
    // n表示，当前循环中，求得字符串长度； m表示题目中要求的字符串长度
    public List<String> helper(int n, int m){
        // 第一步：判断输入或者状态是否合法
        if(n<0 || m<0 || n>m){
            throw  new IllegalArgumentException("invalid input");
        }
        // 第二步：判断递归是否应当结束
        if (n==0)
```

```java
            return new ArrayList<>(Arrays.asList(""));
        if (n==1)
            return new ArrayList<>(Arrays.asList("0","1","8"));

        // 第三步：缩小问题规模
        List<String> list = helper(n-2, m);

        // 第四步：整合结果
        List<String> res = new ArrayList<>();
        for (String s : list){
            if (n!=m)
                // n=m时，表示最外层处理。
                // 例如：原始需求n=m=2，'00'不合法
                // 若原始需求n=m=4，内层循环n=2,m=4,'00';最外层循环，n=m=4时，'1001'
                res.add("0"+s+"0");
            res.add("1"+s+"1");
            res.add("6"+s+"9");
            res.add("8"+s+"8");
            res.add("9"+s+"6");
        }
        return res;
    }
}

作者：tang-yuan-27
链接：https://leetcode-cn.com/problems/strobogrammatic-number-ii/solution/zhong-xin-dui-cheng-shu-ii-247-by-tang-yuan-27/
```

248 Strobogrammatic Number III 未完成

248. Strobogrammatic Number III

难度 困难 20

A strobogrammatic number is a number that looks the same when rotated 180 degrees (looked at upside down).

Write a function to count the total strobogrammatic numbers that exist in the range of low <= num <= high.

Example:

```
Input: low = "50", high = "100"
Output: 3
Explanation: 69, 88, and 96 are three strobogrammatic numbers.
```

Note:
Because the range might be a large number, the low and high numbers are represented as string.

```cpp
class Solution {
public:
    char dl[5] = {'0','1','6','8','9'};
    char dr[5] = {'0','1','9','8','6'};

    bool compare(string & a,string & b){
        if(a.size() != b.size()){return a.size() > b.size();}
        return a >= b;
    }

    bool checkValid(string & num,string & low, string & high){
        return compare(num,low)&&compare(high,num);
    }

    int strobogrammaticInRange(string low, string high) {
        int res = 0;
        queue<string> qu;

        /*intial*/
        qu.push("");
        qu.push("0");
        qu.push("1");
        qu.push("8");

        /*BFS*/
        while(!qu.empty()){
            string curr = qu.front();
            qu.pop();
```

```
29
30              if(curr.size() >= low.size() && curr.size() <= high.size()){
31                  /*skip the num which start with zero*/
32                  if(!(curr[0] == '0' && curr.size() > 1)){
33                      if(checkValid(curr,low,high)){
34                          res++;
35                      }
36                  }
37              }
38
39              if(curr.size() > high.size()){ continue;}
40
41              for(int i = 0 ; i < 5; ++i){
42                  string next = dl[i] + curr + dr[i];
43                  if(next.size() <= high.size()){
44                      qu.push(next);
45                  }
46              }
47          }
48
49          return res;
50      }
51  };
52
53  作者：mike-meng
54  链接：https://leetcode-cn.com/problems/strobogrammatic-number-iii/solution/fei-
    chang-jian-dan-qing-xi-de-bfs-by-mike-meng/
```

249 Group Shifted Strings

249. Group Shifted Strings

难度 中等 20

Given a string, we can "shift" each of its letter to its successive letter, for example: `"abc" -> "bcd"`. We can keep "shifting" which forms the sequence:

```
"abc" -> "bcd" -> ... -> "xyz"
```

Given a list of **non-empty** strings which contains only lowercase alphabets, group all strings that belong to the same shifting sequence.

Example:

```
Input: ["abc", "bcd", "acef", "xyz", "az", "ba", "a",
"z"],
Output:
[
  ["abc","bcd","xyz"],
  ["az","ba"],
  ["acef"],
  ["a","z"]
]
```

```cpp
/*
    ref: https://leetcode-cn.com/problems/group-shifted-strings/solution/c-ha-xi-biao-shi-xian-yong-shi-4ms-ji-bai-9661-nei/
    本质上是把所有字符串转移到 以a 开头，  对齐然后比较
*/
class Solution {
public:
    vector<vector<string>> groupStrings(vector<string>& strings) {
        vector<vector<string>> res;
        unordered_map<string, int> map;

        int size = strings.size();
        for(int i = 0; i < size; i++){
            string ss = strings[i];

            if(ss[0] != 'a'){
                int diff = (ss[0] - 'a' + 26) % 26;
                for(int j = 1; j < ss.size(); j++){
                    ss[j] = (ss[j] - ss[0] + 26) % 26 + 'a';
                }

```

```
                ss[0] = 'a';
            }

            if(map.count(ss) != 0){
                int pos = map[ss];
                res[pos].emplace_back(strings[i]);
            }else{
                map.insert({ss, res.size()});
                res.push_back({strings[i]});
            }
        }

        return res;
    }
};

作者：starfirz
链接：
来源：力扣（LeetCode）
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。
```


```
public List<List<String>> groupStrings(String[] strings) {
    List<List<String>> res = new ArrayList<>();
    boolean[] isAdded   = new boolean[strings.length];
    for(int i = 0; i < strings.length; i++)
    {
        List<String> path = new ArrayList<>();
        if(isAdded[i])  continue;
        else            path.add(strings[i]);

        for(int j = i+1; j < strings.length; j++)
        {
            if(!isAdded[j] && isShifted(strings[i], strings[j]))
            {
                path.add(strings[j]);
                isAdded[j] = true;
            }
```

```
17
18          }
19
20          res.add(path);
21      }
22
23      return res;
24  }
25  //helper function
26  private boolean isShifted(String s, String t)
27  {
28      if(s.length() != t.length())        return false;
29      if(s.length() == 1)                 return true;
30
31      int gap = 0;
32      //z a
33      if(t.charAt(0) - s.charAt(0) < 0)
34          gap = 26 + t.charAt(0) - s.charAt(0);
35      else
36          gap = t.charAt(0) - s.charAt(0);
37      for(int i = 0; i < s.length(); i++)
38      {
39          int curgap =  0;
40          if(t.charAt(i) - s.charAt(i) < 0)
41              curgap = 26 + t.charAt(i) - s.charAt(i);
42          else
43              curgap = t.charAt(i) - s.charAt(i);
44
45          if(curgap != gap)       return false;
46      }
47      return true;
48  }
```

250 Count Univalue Subtrees 递归的典型应用

250. Count Univalue Subtrees

难度 中等 23

Given a binary tree, count the number of uni-value subtrees.

A Uni-value subtree means all nodes of the subtree have the same value.

Example :

```
Input:  root = [5,1,5,5,5,null,5]

              5
             / \
            1   5
           / \   \
          5   5   5

Output: 4
```

执行结果: 通过 显示详情 >

执行用时: 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 37.2 MB , 在所有 Java 提交中击败了 17.92% 的用户

炫耀一下:

```java
int res = 0;
public int countUnivalSubtrees(TreeNode root) {
  dfs(root);

  return res;
}

//to justify the given root is a univalue subtree;
private boolean dfs(TreeNode root){
  boolean status = false;
  if(root == null)        return !status;

  boolean isOkLeft  = dfs(root.left);
  boolean isOkRight = dfs(root.right);

  if(isOkLeft && isOkRight){
    boolean statusL = (root.left != null && root.val == root.left.val) ||
(root.left == null);
```

```
    boolean statusR = (root.right != null && root.val == root.right.val) ||
(root.right == null);
    status = statusL && statusR;
    if(status)    res += 1;
  }

  return status && isOkLeft && isOkRight;
}
```

leecode 251-300

251 Flatten 2D Vector

```
class Vector2D {
    Deque<Integer> queue;
    public Vector2D(int[][] v) {
        queue = new ArrayDeque<Integer>();
        for(int i= 0; i <v.length; i++)
            for(int j = 0; j < v[i].length; j++)
                queue.addLast(v[i][j]);
    }

    public int next() {
        return queue.removeFirst();
    }

    public boolean hasNext() {
        return queue.size() >= 1;
    }
}
```

252 Meeting Rooms 数组的排列

执行结果： 通过 显示详情 >

执行用时： 16 ms ，在所有 C++ 提交中击败了 95.28% 的用户

内存消耗： 11.4 MB ，在所有 C++ 提交中击败了 45.52% 的用户

炫耀一下:

```cpp
class Solution {
public:
    bool canAttendMeetings(vector<vector<int>>& intervals) {
        int size = intervals.size();

        sort(intervals.begin(), intervals.end(), [](vector<int>& v1, vector<int>& v2){ return v1[0] == v2[0] ? v1[1] > v2[1] : v1[0] < v2[0];});

        for(int i = 0; i < size - 1; i++){
            if(intervals[i][1] > intervals[i + 1][0]){
                return false;
            }
        }

        return true;
    }
};
```


```java
public boolean canAttendMeetings(int[][] intervals) {
    List<int[]> list = Arrays.asList(intervals);
    list.sort((o1, o2) -> o1[0] - o2[0]);
    for(int i = 0; i < list.size()-1; i++)
        if(list.get(i)[1] > list.get(i+1)[0])
            return false;
    return true;
}
```

253 Meeting Rooms II 堆排序

```java
/*
  一个显而易见的思路是，每当有新会议时，就遍历所有房间，查看是否有空闲房间。

  把第一个数组的结束时间加入优先队列中
    -如果下一个的开始时间早于优先队列的最早结束时间，那就加一个房间
    -如果下一个的开始时间晚于头元素的最早结束时间，那就不用加了，直接把他替换出去

    Amazing!!!!
*/
class Solution {
    public int minMeetingRooms(int[][] intervals) {

    // Check for the base case. If there are no intervals, return 0
    if (intervals.length == 0)
      return 0;

    // Min heap
    PriorityQueue<Integer> allocator = new PriorityQueue<Integer>(intervals.length, (a,b) -> a-b);

    // Sort the intervals by start time
    Arrays.sort(intervals,(a,b) -> a[0]- b[0]);

    // Add the first meeting
    allocator.add(intervals[0][1]);

    // Iterate over remaining intervals
    for (int i = 1; i < intervals.length; i++) {
      // If the room due to free up the earliest is free, assign that room to this meeting.
      if (intervals[i][0] >= allocator.peek())
        allocator.poll();


      // If a new room is to be assigned, then also we add to the heap,
      // If an old room is allocated, then also we have to add to the heap with updated end time.
      allocator.add(intervals[i][1]);
    }

    // The size of the heap tells us the minimum rooms required for all the meetings.
    return allocator.size();
  }
}

```

```
作者：LeetCode
链接：https://leetcode-cn.com/problems/meeting-rooms-ii/solution/hui-yi-shi-ii-by-leetcode/
```

254 Factor Combinations

```java
    public List<List<Integer>> getFactors(int n) {
        return dfs(n, 2);
    }

    private List<List<Integer>> dfs(int n, int start) {
        List<List<Integer>> res = new ArrayList<>();
        for(int i = start; i * i <= n; i++){
            if(n % i == 0){
                List<Integer> temp = new ArrayList<>();
                temp.add(n / i);
                temp.add(i);
                res.add(new ArrayList<>(temp));

                for(List<Integer> list : dfs(n / i, i)){
                    list.add(i);
                    res.add(new ArrayList<>(list));
                }
            }
        }

        return  res;
    }

```

255 Verify Preorder Sequence in Binary Serach Tree 单调栈

```cpp
class Solution {
public:
    bool verifyPreorder(vector<int>& preorder) {
        int size = preorder.size();

        stack<int> myStack;
        int curMin = INT_MIN;

        for(int i = 0; i < size; i++){
            if(preorder[i] < curMin)
                return false;

            while(!myStack.empty() && preorder[i] > preorder[myStack.top()]){
                curMin = max(curMin, preorder[myStack.top()]);
                myStack.pop();
            }

            myStack.push(i);
        }

        return true;
    }
};
```

```java
/*
  暴力法
*/

public boolean verifyPreorder(int[] preorder) {
    // 遍历每个元素，右侧发现比这个元素大之后，后面必须都要比这个元素大
    int len = preorder.length;
    for (int i = 0; i < len; i++) {
        boolean isBeginBigger = false;
        for (int j = i+1; j < len; j++) {
            if (isBeginBigger && preorder[j] < preorder[i]) {
                return false;
            }
            if (preorder[j] > preorder[i]) {
                isBeginBigger = true;
            }
        }
```

```
    }

    return true;
}

作者：yuruiyin
链接：https://leetcode-cn.com/problems/verify-preorder-sequence-in-binary-search-tree/solution/java-liang-chong-jie-fa-by-npe_tle/
```

```
// 用单调栈的方式，递减栈，当碰到一个数比栈顶元素大的时候，说明从左子树到了右子树。
// 此时要删掉左子树的所有节点，并且保留子树的根为最小值，此时遍历的所有右子树的节点都必须大于这个根，否则非法
    public boolean verifyPreorder(int[] preorder) {
        int len = preorder.length;
        int[] stack = new int[len];
        int top = -1;
        int min = Integer.MIN_VALUE;

        for (int value : preorder) {
            if (value < min) return false;

            while (top > -1 && value > stack[top]) {
                min = stack[top];
                top--;
            }

            stack[++top] = value;
        }

        return true;
    }

作者：yuruiyin
链接：https://leetcode-cn.com/problems/verify-preorder-sequence-in-binary-search-tree/solution/java-liang-chong-jie-fa-by-npe_tle/
```

257 Binayr Tree paths

执行用时： 1 ms ，在所有 Java 提交中击败了 100.00% 的用户

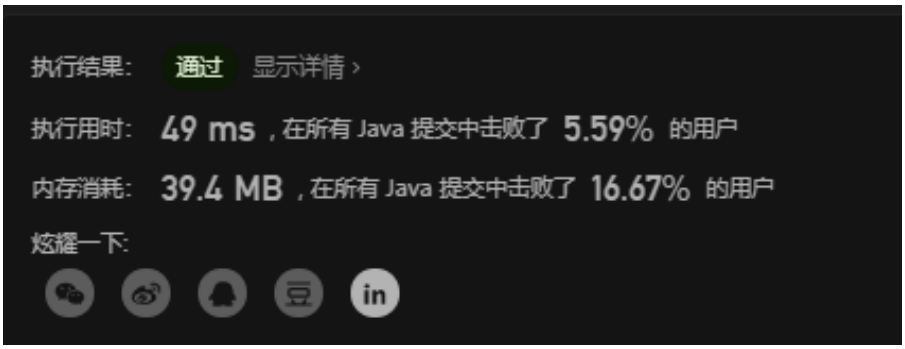
内存消耗： 38.5 MB ，在所有 Java 提交中击败了 84.39% 的用户

```java
class Solution {
    List<String> res = new ArrayList<>();
    public List<String> binaryTreePaths(TreeNode root) {
        if(root == null)
            return res;
        dfs(root, new StringBuilder());
        return res;
    }


    public void dfs(TreeNode root, StringBuilder path){
        if(root == null)    return;

        if(root.left == null && root.right == null){
            path.append(root.val);
            res.add(path.toString());
            return;
        }

        path.append(root.val).append("->");

        //这里最好new 个新的
        dfs(root.left, new StringBuilder(path));
        dfs(root.right, new StringBuilder(path));
    }
}

```

258 Add Digits


```java
public int addDigits(int num) {
    while((num+"").length() != 1)
    {
        int sum = 0;
        char[] chars = (num + "").toCharArray();
        for(char ch : chars)
            sum += ch-'0';
        num = sum;
    }
    return num;
}
```

259 3Sum Smaller 双指针解法




```java
/*
  暴力解法,  O(n3)
*/
public int threeSumSmaller(int[] nums, int target) {
    Arrays.sort(nums);

    int count = 0;
    for(int i = 0; i < nums.length-2; i++)
        for(int j = i+1; j < nums.length-1;j++)
```

```
            for(int k = j+1; k < nums.length; k++)
                if(nums[i] + nums[j] + nums[k] < target)
                    count++;
    return count;

}
```

```
/*
  双指针的解法
*/
public int threeSumSmaller(int[] nums, int target) {
    int count = 0;
    if(nums.length < 3)     return count;
    Arrays.sort(nums);
    int left = 0; int right = 0;

    for(int i = 0; i < nums.length-2; i++)
    {
        left = i+1;
        right = nums.length - 1;

        while(left < right)
        {
            int total = nums[i] + nums[left] + nums[right];

            if(total < target)
            {
                int slots = right - left;
                count += slots;
                left++;
            }
            else
                right--;
        }
    }
    return count;
}
```

260 Single Number III


```java
public int[] singleNumber(int[] nums) {
    int[] res = new int[2];
    int index = 0;
    HashMap<Integer, Integer> map = new HashMap<>();
    for(int i = 0;i < nums.length; i++)
        map.put(nums[i], map.getOrDefault(nums[i], 0) + 1);
    for(Integer i : map.keySet())
        if(map.get(i) == 1)
            res[index++] = i;

    return res;
}
```

```java
/*
  采用O(1)的空间复杂度
  其中思路参考136题，找到落单的数字
  算法思路：
    - 将原数组分成两组，只出现过一次的两个数字分别放到两个组里面
    - 因为要找的两个数组不同，至少有一位二进制不同，将这一位作为分类标准
    - 最后异或的结果实质上就是两个数字的异或，那么出现了1就证明是不同位
    - 构造一个数，恰好就是不同的那一位写成1，其他位写成0
    - 之后分组异或
*/
public int[] singleNumber(int[] nums) {
    int diff = 0;
    for (int n : nums) {
        diff ^= n;
    }
    diff = Integer.highestOneBit(diff);
    int[] result = { 0, 0 };
    for (int n : nums) {
        //当前位是 0 的组，然后组内异或
        if ((diff & n) == 0) {
            result[0] ^= n;
        //当前位是 1 的组
        } else {
            result[1] ^= n;
```

```
        }
    }
    return result;
}

作者：windliang
链接：https://leetcode-cn.com/problems/single-number-iii/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by-5-8/
```

261 Graphic Valid Tree


```java
    public boolean validTree(int n, int[][] edges) {
        if(n != edges.length + 1)       return false;

        HashSet<Integer> visited = new HashSet<>();
        List<List<Integer>> adj = new ArrayList<>();
        for(int i = 0; i < n; i++)
            adj.add(new ArrayList<>());

        for(int[] edge : edges){
            adj.get(edge[0]).add(edge[1]);
            adj.get(edge[1]).add(edge[0]);
        }

        Deque<Integer> queue = new ArrayDeque<>();
        queue.addLast(0);
        while(!queue.isEmpty()){
            int cur = queue.pollFirst();
            visited.add(cur);
            for(int num : adj.get(cur)){
                if(!visited.contains(num))
                    queue.addLast(num);
            }
        }

```

```
        return visited.size() == n;
    }
```

执行结果： 通过 显示详情 >

执行用时： 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 39.1 MB , 在所有 Java 提交中击败了 81.27% 的用户

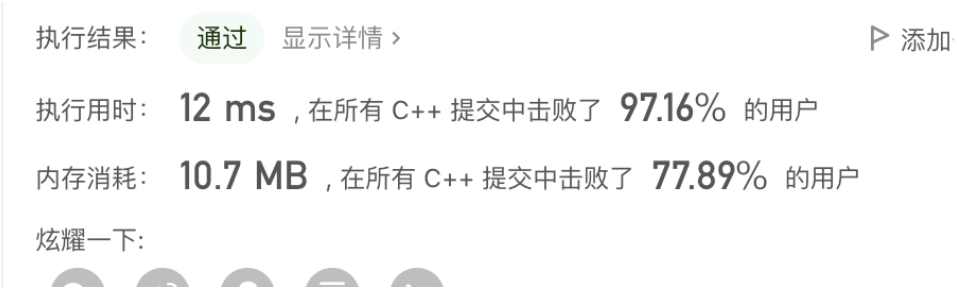
```java
public class Solution {
    public boolean validTree(int n, int[][] edges) {
        if(n != edges.length + 1)       return false;

        WeighedUnionFind wuf = new WeighedUnionFind(n);

        for(int[] edge : edges){
            wuf.union(edge[0], edge[1]);
        }

        return wuf.count() == 1;
    }
}

class WeighedUnionFind{
    private int[] id;
    private int[] sz;
    private int count;

    public WeighedUnionFind(int N){
        id = new int[N];
        sz = new int[N];
        count = N;

        for(int i = 0; i < N; i++){
            id[i] = i;
            sz[i] = 1;
        }
    }

    public int find(int p){
        while(p != id[p]){
            id[p] = id[id[p]];
            p = id[p];
        }

```

```
        return p;
    }

    public void union(int p, int q){
        int pRoot = find(p);
        int qRoot = find(q);

        if(pRoot == qRoot)  return;

        if(sz[pRoot] > sz[qRoot]){
            sz[pRoot] += sz[qRoot];
            id[qRoot] = pRoot;
        }else{
            sz[qRoot] += sz[pRoot];
            id[pRoot] = qRoot;
        }

        count--;
    }

    public boolean connected(int p, int q){return find(p) == find(q);}

    public int count(){return count;}

}
```

```
class Solution {
public:
    bool validTree(int n, vector<vector<int>>& edges) {
        /**1. 节点数==变数+1*/
        if(edges.size() +1 != n) return false;
        /**2. 连通性*/
        vector<vector<int>> Graph(n,vector<int>());
        //构造邻接表
        for(auto edge: edges){
            Graph[edge[0]].push_back(edge[1]);
            Graph[edge[1]].push_back(edge[0]);
        }
        set<int> visited;//访问过的节点放在visited数组中
        //广度优先,看是否连通
        queue<int> q;
        q.push(0);
        visited.insert(0);
        while(!q.empty()){
            int sz = q.size();
```

```
            while(sz){
                int v = q.front();
                q.pop();
                for(auto v_a : Graph[v]){
                    if(visited.find(v_a) != visited.end())//访问过了
                        continue;
                    visited.insert(v_a);
                    q.push(v_a);
                }
                sz--;
            }
        }
        return visited.size() == n;  //访问过的节点数<n,则不连通
    }
};

作者：shubulan
链接：https://leetcode-cn.com/problems/graph-valid-tree/solution/shen-du-you-xian-yan-du-you-xian-bing-cha-ji-de-zo/
```

265 Paint House

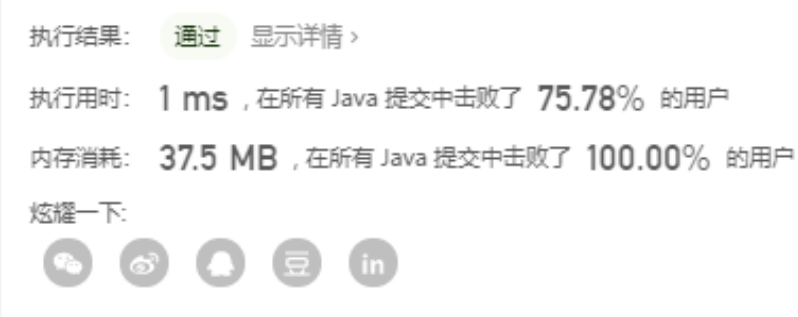



```
//author 郭郭
class Solution {
public:
    int minCostII(vector<vector<int>>& costs) {
        int colors = costs[0].size();
        int row = costs.size();

        if(colors == 1){
            int sum = 0;
            for(vector<int>& v : costs)
                sum += v[0];
            return sum;
        }

        int firstMin    = INT_MAX;
```

```
        int firstIndex  = -1;
        int secondMin   = INT_MAX;
        int secondIndex = -1;

        for(int i = 0; i < row; i++){
            if(i == 0){
                for(int j = 0; j < colors; j++){
                    if(firstMin == INT_MAX){
                        firstMin = costs[i][j];
                        firstIndex = j;
                    }else if(costs[i][j] < firstMin){
                        secondMin = firstMin;
                        firstMin = costs[i][j];
                        secondIndex = firstIndex;
                        firstIndex = j;
                    }else if(costs[i][j] < secondMin){
                        secondMin = costs[i][j];
                        secondIndex = j;
                    }
                }

//                cout << firstMin << " " << secondMin << endl;
//                cout << firstIndex << " " << secondIndex << endl;
            }else{
                int tempFirstMin = INT_MAX;
                int tempFirstIndex = -1;
                int tempSecondMin = INT_MAX;
                int tempSecondIndex = -1;
                for(int j = 0; j < colors; j++){
                    int curCost = costs[i][j] + (firstIndex == j ? secondMin :
firstMin);
                    if(tempFirstMin == INT_MAX){
                        tempFirstMin = curCost;
                        tempFirstIndex = j;
                    }
                    else if(curCost < tempFirstMin){
                        tempSecondMin = tempFirstMin;
                        tempFirstMin = curCost;
                        tempSecondIndex = tempFirstIndex;
                        tempFirstIndex = j;
                    }else if(curCost < tempSecondMin){
                        tempSecondMin = curCost;
                        tempSecondIndex = j;
                    }
                }

                firstMin = tempFirstMin;
                secondMin = tempSecondMin;
                firstIndex = tempFirstIndex;
```

```
                secondIndex = tempSecondIndex;

                // cout << "----" << i << endl;
                // cout << firstMin << " " << secondMin << endl;
                // cout << firstIndex << " " << secondIndex << endl;


            }

            if(i == row - 1)
                return firstMin;
        }

        return -1;
    }
};
```

266 Palindrome Permutation




```
public boolean canPermutePalindrome(String s) {
    int index = 0;
    HashMap<Character, Integer> map = new HashMap<>();
    while(index < s.length())
    {
        map.put(s.charAt(index), map.getOrDefault(s.charAt(index), 0) + 1);
        index++;
    }

    int count = 0;
    for(char ch : map.keySet())
    {
        if(map.get(ch) % 2 != 0)
            count++;
        if(count > 1)       return false;
    }
    return true;
}
```

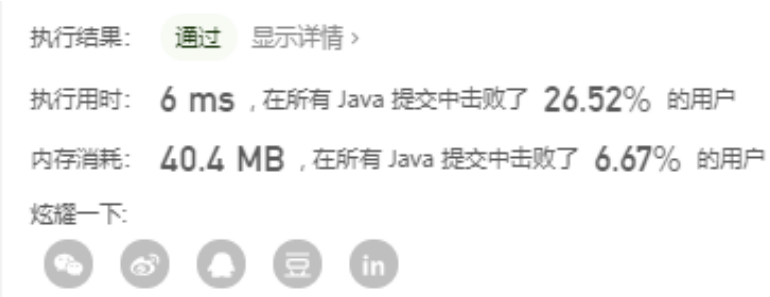
263 Ugly Number

```java
public boolean isUgly(int num) {
    if(num < 1)     return false;
    if(num == 1)    return true;

    while(num > 1)
    {
        if(num % 2 == 0)    num /= 2;
        else if(num % 3 == 0)    num /= 3;
        else if(num % 5 == 0)    num /= 5;
        else
            return false;
    }

    return true;
}
```

268 Missing Number

```java
class Solution {
    public int missingNumber(int[] nums) {
        for(int i = 0 ; i < nums.length; i++){
            if(nums[i] == nums.length)
                continue;

            while(nums[i] != i){
                if(nums[i] == nums.length)
                    break;
                int temp = nums[nums[i]];
                nums[nums[i]] = nums[i];
                nums[i] = temp;
            }
        }

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

```
        }

        return nums.length;
    }
}
```




```java
public int missingNumber(int[] nums) {
    HashSet<Integer> set = new HashSet<>();
    for(int num : nums)
        set.add(num);

    for(int i=0;;i++)
        if(!set.contains(i))
            return i;

}
```

269 Alien Dictionary 拓扑排序

```
class Solution {
public:
    string alienOrder(vector<string>& words) {
        unordered_map<char, unordered_set<int>> map;
        int size = words.size();
        for(int i = 0; i < size - 1; i++){
            if(words[i].size() > words[i + 1].size() && words[i].find(words[i + 1])
== 0)
                return "";
            for(int j = 0; j < words[i].size() && j < words[i + 1].size(); j++){
                if(words[i][j] == words[i + 1][j])
```

```
                    continue;

                map[words[i][j]].insert(words[i + 1][j]);
                break;
            }
        }

        vector<int> degree = vector<int>(26, -1);
        for(string& str : words){
            for(char ch : str)
                degree[ch - 'a'] = 0;
        }

        for(auto it = map.begin(); it != map.end(); it++){
            char key = it->first;
            for(char ch : map[key])
                degree[ch - 'a']++;

        }

        string res;
        queue<char> myQueue;
        int count = 0;
        for(int i = 0; i < 26; i++){
            if(degree[i] == 0)
                myQueue.push((char)(i + 'a'));

            if(degree[i] != -1)
                count++;
        }

        while(!myQueue.empty()){
            char ch = myQueue.front(); myQueue.pop();

            res += ch;
            if(map.count(ch)){
                auto set = map[ch];
                for(char ch : set) {
                    degree[ch - 'a']--;
                    if(degree[ch - 'a'] == 0)
                        myQueue.push(ch);
                }

            }
        }

        return count == res.size() ? res : "";
    }
};
```

```java
/*
  将字符串的优先级构建为图，然后进行拓扑排序。
  如果图中无环，则将拓扑排序输出，否则顺序是非法的。
*/
class Solution {
    public String alienOrder(String[] words) {
        //1.构建图
        Map<Character, Set<Character>> map = new HashMap<>();
        for (int i = 0; i < words.length - 1; i++) {
            for (int j = 0; j < words[i].length() && j < words[i + 1].length(); j++) {
                //如果字符相同，比较下一个
                if (words[i].charAt(j) == words[i + 1].charAt(j)) continue;
                //保存第一个不同的字符顺序
                Set<Character> set = map.getOrDefault(words[i].charAt(j), new HashSet<>());
                set.add(words[i + 1].charAt(j));
                map.put(words[i].charAt(j), set);
                break;
            }
        }

        //2.拓扑排序
        //创建保存入度的数组
        int[] degrees = new int[26];
        Arrays.fill(degrees, -1);
        //注意，不是26字母都在words中出现，所以出度分为两种情况：没有出现的字母出度为-1，出现了的字母的出度为非负数
        for (String str : words)
            //将出现过的字符的出度设定为0
            for (char c : str.toCharArray())
                degrees[c - 'a'] = 0;

        for (char key : map.keySet())
            for (char val : map.get(key))
                degrees[val - 'a']++;


        //创建StringBuilder保存拓扑排序
        StringBuilder sb = new StringBuilder();
        //创建一个Queue保存入度为0的节点
        Queue<Character> list = new LinkedList<>();

        int count = 0;//计算图中节点数
        for (int i = 0; i < 26; i++) {
```

```
            if (degrees[i] != -1) count++;
            if (degrees[i] == 0) {
                list.add((char) ('a' + i));
            }
        }

        while (!list.isEmpty()) {
            Character cur = list.poll();
            sb.append(cur);
            //将邻接点出度-1
            if (map.containsKey(cur)) {
                Set<Character> set = map.get(cur);
                for (Character c : set) {
                    degrees[c - 'a']--;
                    if (degrees[c - 'a'] == 0) list.add(c);
                }
            }
        }

        //判断是否有环
        if (sb.length() != count) return "";
        else return sb.toString();

    }
}

作者：mmmmmJCY
链接：https://leetcode-cn.com/problems/alien-dictionary/solution/java-tuo-bu-pai-xu-by-zxy0917/
```

270 Closest Binary Search Tree Value

```
//二分查找
class Solution {
  public int closestValue(TreeNode root, double target) {
    int val, closest = root.val;
    while (root != null) {
      val = root.val;
      closest = Math.abs(val - target) < Math.abs(closest - target) ? val :
closest;
      root =  target < root.val ? root.left : root.right;
    }
    return closest;
```

```
  }
}

作者：LeetCode
链接：https://leetcode-cn.com/problems/closest-binary-search-tree-value/solution/zui-jie-jin-de-er-cha-sou-suo-shu-zhi-by-leetcode/
```

```java
/*
  暴力法，写的行云流水
*/
public void inorder(TreeNode root, List<Integer> nums) {
    if (root == null) return;
    inorder(root.left, nums);
    nums.add(root.val);a
    inorder(root.right, nums);
}

public int closestValue(TreeNode root, double target) {
    List<Integer> nums = new ArrayList();
    inorder(root, nums);
    return Collections.min(nums, new Comparator<Integer>() {
        @Override
        public int compare(Integer o1, Integer o2) {
            return Math.abs(o1 - target) < Math.abs(o2 - target) ? -1 : 1;
        }
    });
}
```

272 Closet Binary Search Tree Value II

```java
/*
    采用优先队列，比较牛逼

 */
class Solution {
    double t;
    int n;
    Queue<Integer> queue;
    public List<Integer> closestKValues(TreeNode root, double target, int k) {
        // 用一个优先队列来保存值，然后遍历树，找到
        t=target;
        n=k;
```

```
        queue = new PriorityQueue<>((o1, o2) -> Double.compare(Math.abs(o2 - t), Math.abs(o1 - t)));
        inOrder(root);
        return new ArrayList<>(queue);
    }

    private void inOrder(TreeNode root) {
        if (root==null) return;
        inOrder(root.left);
        int val=root.val;
        if (queue.size()<n){
            queue.add(val);
        }else {
            double t1=Math.abs(val-t);
            Integer peek = queue.peek();
            double t2=Math.abs(peek-t);
            if (t2>t1){
                queue.poll();
                queue.add(val);
            }
        }
        inOrder(root.right);
    }
}


作者：lifengcai_fans
链接：https://leetcode-cn.com/problems/closest-binary-search-tree-value-ii/solution/qing-xi-jian-dan-by-xiaoweixiang/
```

275 H-index || 实际上我还不清楚这是什么玩意

```
public int hIndex(int[] citations) {
    int H_index = 0;
    int n = 1;
    for(int i = citations.length - 1; i >= 0; i--)
        if(n > citations[i])
            break;
    else
        n++;

    return n-1;
}
```

278 First Bad Version


```java
/*
  典型二分
*/
public int firstBadVersion(int n) {
    long left = 1;
    long right = n;

    while(left < right)
    {
        long mid = (left + right) /2;

        if(isBadVersion((int)mid))
            right = mid;
        else
            left = mid + 1;
    }
    return (int)left;
}
```

280 Wiggle Sort

282 Expression Add Operations 未解决

283 Move Zeroes 快排

```java
/*
  每次遍历，看看有几个零，非零的赋值
  最后把末尾附上0
*/
public void moveZeroes(int[] nums) {
    int j = 0;

    for(int i = 0; i < nums.length; i++)
        if(nums[i] != 0)
            nums[j++] = nums[i];

    for(int i = j; i < nums.length; i++)
        nums[i] = 0;
}
```

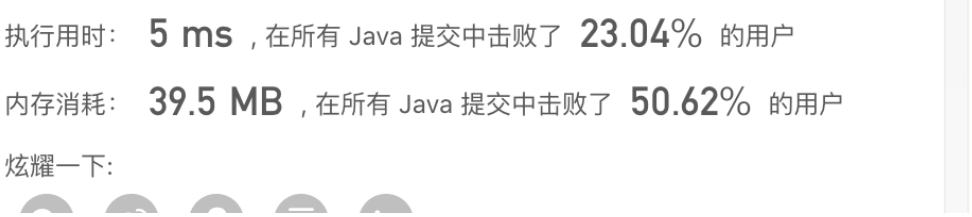
```java
// 快速排序的又一另类应用
class Solution {
  public void moveZeroes(int[] nums) {
    if(nums==null) {
      return;
    }
    //两个指针i和j
    int j = 0;
    for(int i=0;i<nums.length;i++) {
      //当前元素!=0，就把其交换到左边，等于0的交换到右边
      if(nums[i]!=0) {
        int tmp = nums[i];
        nums[i] = nums[j];
        nums[j++] = tmp;
      }
    }
  }
}

作者：wang_ni_ma
链接：https://leetcode-cn.com/problems/move-zeroes/solution/dong-hua-yan-shi-283yi-dong-ling-by-wang_ni_ma/
```

284 Peeking Iterator


```java
// Java Iterator interface reference:
// https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html

class PeekingIterator implements Iterator<Integer> {
    private Deque<Integer> queue;

  public PeekingIterator(Iterator<Integer> iterator) {
      // initialize any member here.
      queue = new ArrayDeque<>();
        while(iterator.hasNext())
            queue.addLast(iterator.next());
  }

    // Returns the next element in the iteration without advancing the iterator.
  public Integer peek() {
        return queue.peek();
  }

  // hasNext() and next() should behave the same as in the Iterator interface.
  // Override them if needed.
  @Override
  public Integer next() {
      return queue.poll();
  }

  @Override
  public boolean hasNext() {
      return queue.size() > 0;
  }
}
```

285 Inorder Successor in BST


```java
/*
  笨办法
*/
public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {
    List<Integer> list = new ArrayList<>();
    inOrder(root, list);
    for(int i = 0; i < list.size()-1; i++)
        if(list.get(i) == p.val)
            return new TreeNode(list.get(i+1));
    return null;
}

private void inOrder(TreeNode root, List<Integer> list)
{
    if(root == null)        return;
    inOrder(root.left, list);
    list.add(root.val);
    inOrder(root.right, list);
}
```




```java
//套用框架， 中序遍历
//Author by Wenchao Guo
    public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {
```

```
        Deque<TreeNode> stack = new ArrayDeque<>();

        TreeNode pre = null;
        while(root != null || !stack.isEmpty()){
            while(root != null){
                stack.push(root);
                root = root.left;
            }

            TreeNode cur = stack.pop();
            if(pre != null && pre.val <= p.val && cur.val > p.val)
                return cur;
            else
                pre = cur;
            root = cur.right;

        }

        return null;
    }
```

286 Walls and Gates

执行结果： 通过 显示详情 >

执行用时： 22 ms , 在所有 Java 提交中击败了 18.25% 的用户

内存消耗： 45.3 MB , 在所有 Java 提交中击败了 5.07% 的用户

```

/*
    记得利用好 单点BFS 最短的特性
*/
class Solution {
    int[][] dir = {{-1, 0},{1, 0},{0, 1},{0, -1}};
    int row;
    int column;
    public void wallsAndGates(int[][] rooms) {
        row    = rooms.length;
        column = row == 0 ? 0 : rooms[0].length;

        if(column == 0)     return;
        Deque<int[]> queue = new ArrayDeque<>();
        HashSet<int[]> visited = new HashSet<>();

```

```
        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(rooms[i][j] == 0){
                    queue.addLast(new int[]{i, j});
                    visited.add(new int[]{i, j});
                }
        int record = 1;
        while(!queue.isEmpty()){
            int size = queue.size();
            for(int i = 0; i < size; i++){
                int[] cur = queue.pollFirst();

                for(int k = 0 ; k < 4; k++){
                    int newX = cur[0] + dir[k][0];
                    int newY = cur[1] + dir[k][1];

                    if(!isInRange(newX, newY)
                        || rooms[newX][newY] == -1 || rooms[newX][newY] !=
Integer.MAX_VALUE ||
                        visited.contains(new int[]{newX, newY}))
                        continue;

                    rooms[newX][newY] = record;
                    queue.add(new int[]{newX, newY});
                    visited.add(new int[]{newX, newY});
                }
            }

            record++;
        }
    }

    private boolean isInRange(int i, int j){
        return i >= 0 && j >= 0 && i < row && j < column;
    }
}
```

287 Find the Duplicate Number 另类的二分法运用 + 牛逼双指针

```java
/*
  典型的时间换空间的想法

  采用二分法， 思路是
  每次拿到中间数 mid
  然后在所有元素遍历一遍，看看数量少于mid的元素个数
  如果多，说明在区间[1,mid]一定有重复的
  反之就在另一半

  属实是二分法的另类应用
*/
public int findDuplicate(int[] nums) {
    int len = nums.length;
    int left = 1; //这里left == 1 是因为数组元素从1-n的限制
    int right = len - 1;
    while(left < right)
    {
        int mid = (left + right) >>> 1;

        int cnt = 0;
        for(int num : nums)
            if(num <= mid)
                cnt++;
        if(cnt > mid)
            right = mid;
        else
            left = mid + 1;
    }
    return left;
}
```

```java
//对比链表找环的题
public int findDuplicate(int[] nums) {
    int slow = nums[0], fast = nums[nums[0]];

    while(slow != fast)
    {
        slow = nums[slow];
        fast = nums[nums[fast]];
```

```
    }

    slow = 0;

    while(slow != fast)
    {
        slow = nums[slow];
        fast = nums[fast];
    }
    return slow;
}
```

289 Game of Life

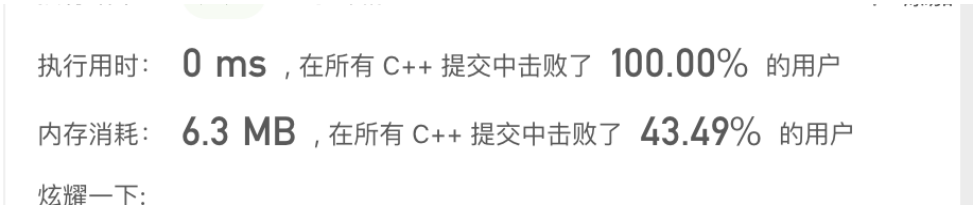

```
/*
  不太符合题目的follow up 属于 non in-place
*/
public void gameOfLife(int[][] board) {
    if(board.length == 0 || board[0].length == 0)       return;
    int[][] newBoard = new int[board.length][board[0].length];
    for(int i = 0; i < board.length; i++)
        for(int j = 0; j < board[0].length; j++)
            newBoard[i][j] = board[i][j];

    for(int i = 0; i < board.length; i++)
        for(int j = 0; j < board[0].length; j++)
        {
            int ln = getNumber(board, i, j);
            if      (board[i][j] == 1 && ln < 2)              newBoard[i][j] = 0;
            else if (board[i][j] == 1 && ln >= 2 && ln <= 3)  newBoard[i][j] = 1;
            else if (board[i][j] == 1 && ln > 3)              newBoard[i][j] = 0;
            else if (board[i][j] == 0 && ln == 3)             newBoard[i][j] = 1;

        }

    for(int i = 0; i < board.length; i++)
        for(int j = 0; j < board[0].length; j++)
```

```
            board[i][j] = newBoard[i][j];
}

private int getNumber(int[][] board, int i, int j)
{
    int count = 0;
    if(i+1 < board.length)                                  count += board[i+1][j] == 1?
1 : 0;
    if(i-1 >= 0)                                            count += board[i-1][j] == 1?
1 : 0;
    if(j + 1 < board[0].length)                             count += board[i][j+1] == 1?
1 : 0;
    if(j - 1>= 0)                                           count += board[i][j-1] == 1?
1 : 0;
    if(i+1 < board.length && j + 1 < board[0].length)  count += board[i+1][j+1] ==
1? 1 : 0;
    if(i+1 < board.length && j - 1>= 0 )                count += board[i+1][j-1] ==
1? 1 : 0;
    if(i-1 >= 0 && j + 1 < board[0].length)             count += board[i-1][j+1] ==
1? 1 : 0;
    if(i-1 >= 0 && j - 1>= 0)                           count += board[i-1][j-1] ==
1? 1 : 0;
    return count;
}

private void copyBoard(int[][] oldBoard, int[][] newBoard)
{
    for(int i = 0; i < oldBoard.length; i++)
        for(int j = 0; j < oldBoard[0].length; j++)
            newBoard[i][j] = oldBoard[i][j];
}
```

```
/*
  in-place 的算法
  用10作为一个测量周围有没有1的方案，实现拿到周围到底有几个1
  相当于 立了一个flag = 10
  并且，相较于上面一个方法，找到该cell中的1的个数
  我们这种方法是找1，然后从1这个cell扩散周围8个格子
*/
public void gameOfLife(int[][] board)
{
    if(board.length == 0 || board[0].length==0) return;
    int m = board.length, n = board[0].length;

    int[] neighbor = {0, 1, -1};
    for(int i = 0; i < m; i++)
        for(int j = 0; j < n; j++)
```

```
            if(board[i][j] % 10 == 1)
                for(int k = 0; k < 3; k++)
                    for(int l = 0; l < 3; l++)
                    {
                        int left = i + neighbor[k];
                        int right = j + neighbor[l];
                        if(neighbor[k] == 0 && neighbor[l] == 0)
                            continue;
                        if(left >= 0 && right >= 0 && left < m && right < n)
                            board[left][right] += 10;
                    }
    for(int i = 0; i < m; i++)
        for(int j = 0; j < n; j++)
            if(board[i][j] == 21 || board[i][j] == 31)
                board[i][j] = 1;
            else if(board[i][j] == 30)
                board[i][j] = 1;
            else
                board[i][j] = 0;


}
```

290 Word Pattern




```
class Solution {
public:
    bool wordPattern(string pattern, string s) {
        stringstream ss(s);
        vector<string> vec;
        string token;
        while(ss >> token){
            vec.push_back(token);
        }

        if(pattern.size() != vec.size())
            return false;

        unordered_map<char, string> m1;
        unordered_set<string> hasMapped;

```

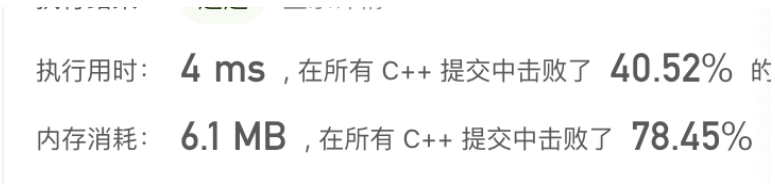
```
        for(int i = 0; i < pattern.size(); i++){
            char ch = pattern[i];
            if(hasMapped.count(vec[i]) != 0){
                if(m1.count(ch) == 0 || m1[ch] != vec[i])
                    return false;
            }else{
                if(m1.count(ch) != 0)
                    return false;

                m1.emplace(make_pair(ch, vec[i]));
                hasMapped.emplace(vec[i]);
            }
        }

        return true;
    }
};
```

```
func wordPattern(pattern string, s string) bool {
  m := make(map[uint8]string)
  n := make(map[string]uint8)

  words := strings.Fields(s)

  index1 := 0
  index2 := 0

  for ; index1 != len(pattern) && index2 != len(words); {
    ch := pattern[index1]

    if old, ok := m[ch]; ok{
      if words[index2] != old{
        return false
      }
    }else{
      if _, ok := n[words[index2]]; ok{
        return false
      }
      m[ch] = words[index2]
      n[words[index2]] = ch
    }

    index1++
    index2++
  }

  return index1 == len(pattern) && index2 == len(words)
```

```
}
```


```java
public boolean wordPattern(String pattern, String str) {
    String[] words = str.split(" ");
    HashMap<Character, String> map = new HashMap<>();
    if(pattern.length() != words.length)        return false;

    for(int i = 0; i < pattern.length(); i++)
    {
        if(!map.containsKey(pattern.charAt(i)))
            if(map.values().contains(words[i]))
                return false;
            else
                map.put(pattern.charAt(i), words[i]);

        if(!(map.get(pattern.charAt(i)).equals(words[i])))
            return false;
    }
    return true;
}
```

291 Word Pattern II




```cpp

class Solution {
public:
    unordered_map<char, string> map1;
    unordered_map<string, char> map2;
    bool wordPatternMatch(string pattern, string s) {
```

```
        return dfs(pattern, s, 0, 0);
    }

    bool dfs(string pattern, string s, int index1, int index2){
        if(index1 == pattern.size() || index2 == s.size()){
            if(index1 == pattern.size() && index2 == s.size())
                return true;
            return false;
        }

        char frac1 = pattern[index1];
        for(int j = index2 + 1; j <= s.size(); j++){
            string frac2 = s.substr(index2, j - index2);

            if(map1.count(frac1) == 0){
                if(map2.count(frac2) != 0)
                    continue;
                map1.insert({frac1, frac2});
                map2.insert({frac2, frac1});

                if(dfs(pattern, s, index1 + 1, j))
                    return true;

                map1.erase(frac1);
                map2.erase(frac2);
            }else{
                if(map1[frac1] != frac2)
                    continue;
                if(dfs(pattern, s, index1 + 1, j))
                    return true;
            }
        }

        return false;
    }
};
```

292 Nim Game

```
/*
  通过50/60个案例， 卡在1348820612这个数字
*/
private HashMap<Integer, Boolean> map = new HashMap<>();

public boolean canWinNim(int n) {
    if(n <= 3)      return true;
```

```
    if(map.containsKey(n))      return map.get(n);
    map.put(1, true);
    map.put(2, true);
    map.put(3, true);

    boolean a = map.containsKey(n-1) ? !map.get(n-1) : !canWinNim(n-1);
    boolean b = map.containsKey(n-2) ? !map.get(n-2) : !canWinNim(n-2);
    boolean c = map.containsKey(n-3) ? !map.get(n-3) : !canWinNim(n-3);

    map.put(n, a || b || c);
    return map.get(n);
}
```

```
//无法通过 通过50/60个案例， 卡在1348820612这个数字
public boolean canWinNim(int n) {
    if(n <= 3)      return true;
    boolean[] canWin = new boolean[n+1];
    canWin[1] = true;
    canWin[2] = true;
    canWin[3] = true;

    for(int i = 4; i <= n; i++)
        canWin[i] = !canWin[i-1] || !canWin[i-2] || !canWin[i-3];
    return canWin[n];
}
```

```
//52/60个案例， 卡在730530679这个数字
public boolean canWinNim(int n) {
    if(n <= 3)      return true;
    boolean canWin = false;
    boolean a = true;
    boolean b = true;
    boolean c = true;

    for(int i = 4; i <= n; i++)
    {
        canWin = !a || !b || !c;
        a = b;
        b = c;
        c = canWin;
    }
    return canWin;
}
```


```java
  public boolean canWinNim(int n) {
       if(n <= 3)       return true;
       return !(n % 4 == 0);
    }
```

293 Flip Game 1


```java
public List<String> generatePossibleNextMoves(String s) {
    List<String> res = new ArrayList<>();
    int index = 0;

    while(index < s.length())
    {
        char[] chars = s.toCharArray();
        while((index + 1 < chars.length) && (chars[index] != '+' || chars[index+1] != '+'))
            index++;
        if(index >= s.length() -1)      break;
        chars[index] = '-';
        chars[index + 1] = '-';
        res.add(String.valueOf(chars));
        index++;
    }
    return res;
}
```

298 Binary Tree longest Consecutive Sequence

```java
/*
  方法一：自顶向下DFS
  注意，之所以在参数中引入多一个parent的argument
  主要是为了方便判断连续性
*/
private int maxLength = 0;
public int longestConsecutive(TreeNode root)
{
    dfs(root, null, 0);
    return maxLength;
}

private void dfs(TreeNode p, TreeNode parent, int length)
{
    if(p == null)   return;
    length = (parent != null && p.val == parent.val + 1) ? length + 1: 1;
    maxLength = Math.max(maxLength, length);
    dfs(p.left, p, length);
    dfs(p.right, p, length);
}
```

```java
/*
  自底向上，类似后序遍历
*/

private int maxLength = 0;
public int longestConsecutive(TreeNode root)
{
    dfs(root);
    return maxLength;
}

private int dfs(TreeNode p)
{
    if(p == null) return 0;
    int L = dfs(p.left) + 1;
    int R = dfs(p.right) + 1;
    if(p.left != null && p.val + 1 != p.left.val)
```

```
        L = 1;
    if(p.right != null && p.val + 1 != p.right.val)
        R = 1;

    int length = Math.max(L, R);
    maxLength = Math.max(maxLength, length);
    return length;
}
```

执行结果: 通过 显示详情 >

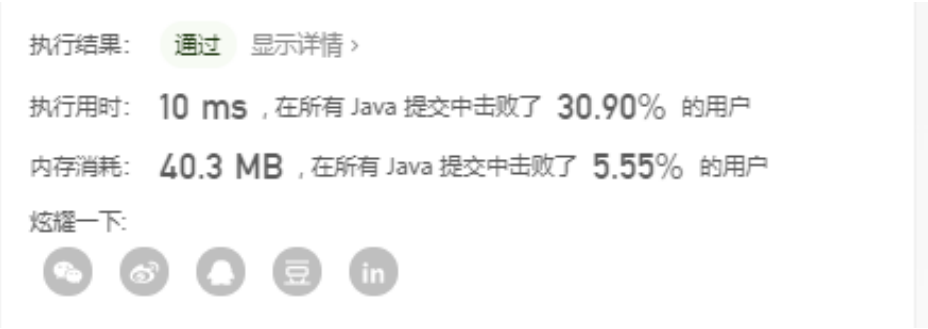
执行用时: 1 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 40.2 MB , 在所有 Java 提交中击败了 41.38% 的用户

炫耀一下:

```
//Author: WenchaoGuo
int res = 0;
public int longestConsecutive(TreeNode root) {
  dfs(root, 0);
  return res;
}

private void dfs(TreeNode root, int sum){
  if(root == null)      return;
  res = Math.max(res, sum + 1);

  if(root.left != null && root.val + 1 == root.left.val)
    dfs(root.left, sum + 1);
  else
    dfs(root.left, 0);

  if(root.right != null && root.val + 1 == root.right.val)
    dfs(root.right, sum + 1);
  else
    dfs(root.right, 0);

  return;
}

```

299 Bulls And Cows




```java
public String getHint(String secret, String guess) {
    int A = 0, B = 0;
    HashMap<Character, Integer> mapA = new HashMap<>();
    HashMap<Character, Integer> mapB = new HashMap<>();

    for(int i = 0; i < secret.length(); i++)
        if(secret.charAt(i) == guess.charAt(i))
            A++;
        else
        {
            mapA.put(secret.charAt(i), mapA.getOrDefault(secret.charAt(i),0)+1);
            mapB.put(guess.charAt(i), mapB.getOrDefault(guess.charAt(i),0)+1);
        }
    for(char ch : mapA.keySet())
        if(mapA.get(ch) <= mapB.getOrDefault(ch,0))
            B += mapA.get(ch);
        else
            B += mapB.getOrDefault(ch,0);

    return A + "A" + B + "B";
}
```