1. Two Sum

Easy Topics Companies Hint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

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

Constraints:

- $2 \le nums.length \le 10^4$
- $-10^9 \le nums[i] \le 10^9$
- $-10^9 \le target \le 10^9$
- **Only one valid answer exists.**

Follow-up: Can you come up with an algorithm that is less than $O(n^2)$ time complexity?

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
      HashMap<Integer,Integer> map = new HashMap<>();
      for(int i=0; i<nums.length; i++) {
        int key = target - nums[i];
        if(map.containsKey(key)){
          return new int[]{map.get(key),i};
        }
        map.put(nums[i],i);
      }
      return null;

    }
}
```

Saved to local

Testcase Test Result

You must run your code first

Description Editorial Solutions Submissions

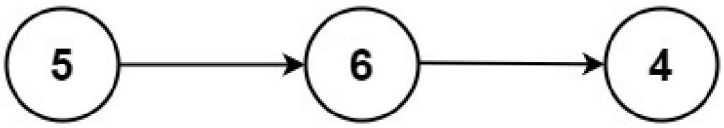
2. Add Two Numbers

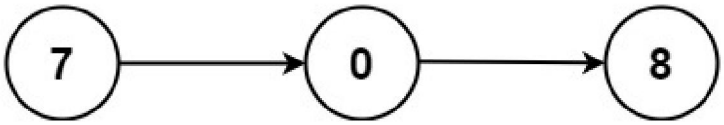
Medium Topics Companies

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, an nodes contains a single digit. Add the two numbers and return the sum as a linked list.

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

Example 1:






```
Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7,0,8]
Explanation: 342 + 465 = 807.
```

Example 2:

```
Input: l1 = [0], l2 = [0]
Output: [0]
```

Example 3:

28.9K 435

Code

Java Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        int carry=0;
        ListNode head = new ListNode(0);
        ListNode cur=head;
        while(l1!=null || l2!=null) {
            int val=carry;
            if(l1!=null) {
                val+=l1.val;
                l1=l1.next;
            }
            if(l2!=null) {
                val+=l2.val;
                l2=l2.next;
            }
            int tmp=val%10;
            carry=(val-tmp)/10;

            cur.next = new ListNode(tmp);
            cur=cur.next;


        }
        if(carry>0) {
            cur.next=new ListNode(carry);
        }
        return head.next;

    }
}
```

Saved to local

Testcase Test Result

You must run your code first

3. Longest Substring Without Repeating Characters

Medium
Topics
Companies

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

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
```

Constraints:

- `0 <= s.length <= 5 * 10^4`
- `s` consists of English letters, digits, symbols and spaces.

Seen this question in a real interview before? 1/4

Yes No

Accepted 5.1M Submissions 14.9M Acceptance Rate 34.1%

37.8K 239

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public int sol1(String s) {

        if (s.length()==0) return 0;
        HashMap<Character, Integer> lastPos = new HashMap<Character, Integer>();
        int max=0;
        int left=0;
        for (int i=0; i<s.length(); ++i){
            if (lastPos.containsKey(s.charAt(i))){
                left = Math.max(left,lastPos.get(s.charAt(i))+1);
            }
            lastPos.put(s.charAt(i),i);
            max = Math.max(max,i-left+1);
        }
        return max;
    }

    public int sol2(String s) {
        if (s == null || s.length() == 0) {
            return 0;
        }

        int max = 0;
        int left = 0;
        int[] last = new int[256]; //last[c]+1 is the last position of c
        for (int j = 0; j < s.length(); j++) {
            left = Math.max(left, last[s.charAt(j)]);
            max = Math.max(max, j - left + 1);
            last[s.charAt(j)] = j + 1;
        }
        return max;
    }

    public int lengthOfLongestSubstring(String s) {
        // return sol1(s);

        return sol2(s);

    }


}
```

Saved to local

Testcase Test Result

You must run your code first

4. Median of Two Sorted Arrays

Hard Topics Companies

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be `O(log (m+n))`.

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```

Constraints:

- `nums1.length == m`
- `nums2.length == n`
- `0 <= m <= 1000`
- `0 <= n <= 1000`
- `1 <= m + n <= 2000`
- `-10^6 <= nums1[i], nums2[i] <= 10^6`

Seen this question in a real interview before? 1/4

Yes No

Run Submit

Code

Java Auto

```java
class Solution {
    //============== recursive ==============================
    public double sol1(int[] nums1, int[] nums2) {
      int n1 = nums1.length;
      int n2 = nums2.length;
      int k1 = (n1+n2+1) / 2;
      int k2 = (n1+n2+2) / 2;

      int ret1 = find_k(nums1, nums2, 0, 0, k1);
      int ret2 = find_k(nums1, nums2, 0, 0, k2);

      return (ret1 + ret2) / 2.0;
    }

    public int find_k(int[] nums1, int[] nums2, int begin1, int begin2, int k) {
      if (begin1 >= nums1.length) return nums2[begin2 + k -1];
      if (begin2 >= nums2.length) return nums1[begin1 + k -1];
      if (k==1) return Math.min(nums1[begin1], nums2[begin2]);
      int mid1 = Integer.MAX_VALUE;
      int mid2 = Integer.MAX_VALUE;
      if (begin1+k/2-1 < nums1.length) mid1 = nums1[begin1+k/2-1 ];
      if (begin2+k/2-1 < nums2.length) mid2 = nums2[begin2+k/2-1 ];
      if (mid1 < mid2)
        return find_k(nums1, nums2, begin1 + k/2, begin2, k-k/2);
      else
        return find_k(nums1, nums2, begin1, begin2+ k/2, k-k/2);
    }


    //============== iterative ==============================
    public double sol2(int[] nums1, int[] nums2) {
      int m = nums1.length;
      int n = nums2.length;

      if (m > n) {
        return findMedianSortedArrays(nums2, nums1);
      }

      int i = 0, j = 0, imin = 0, imax = m, half = (m + n + 1) / 2;
      double maxLeft = 0, minRight = 0;
      while(imin <= imax){
        i = (imin + imax) / 2;
        j = half - i;
        if(j > 0 && i < m && nums2[j - 1] > nums1[i]){
          imin = i + 1;
        }else if(i > 0 && j < n && nums1[i - 1] > nums2[j]){
          imax = i - 1;
        }else{
```

Saved to local

Testcase Test Result

Code

Java Auto

```java

    //============== iterative ==============================
    public double sol2(int[] nums1, int[] nums2) {
      int m = nums1.length;
      int n = nums2.length;

      if (m > n) {
        return findMedianSortedArrays(nums2, nums1);
      }

      int i = 0, j = 0, imin = 0, imax = m, half = (m + n + 1) / 2;
      double maxLeft = 0, minRight = 0;
      while(imin <= imax){
        i = (imin + imax) / 2;
        j = half - i;
        if(j > 0 && i < m && nums2[j - 1] > nums1[i]){
          imin = i + 1;
        }else if(i > 0 && j < n && nums1[i - 1] > nums2[j]){
          imax = i - 1;
        }else{
          if(i == 0){
            maxLeft = (double)nums2[j - 1];
          }else if(j == 0){
            maxLeft = (double)nums1[i - 1];
          }else{
            maxLeft = (double)Math.max(nums1[i - 1], nums2[j - 1]);
          }
          break;
        }
      }
      if((m + n) % 2 == 1){
        return maxLeft;
      }
      if(i == m){
        minRight = (double)nums2[j];
      }else if(j == n){
        minRight = (double)nums1[i];
      }else{
        minRight = (double)Math.min(nums1[i], nums2[j]);
      }

      return (double)(maxLeft + minRight) / 2;
    }

    //============================================
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
      //return sol1(nums1, nums2);
      return sol2(nums1, nums2);
    }
```

Saved to local

Testcase Test Result

Description Editorial Solutions Submissions

5. Longest Palindromic Substring

Medium Topics Companies Hint

Given a string `s`, return *the longest palindromic substring* in `s`.

Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

Constraints:

- `1 <= s.length <= 1000`
- `s` consist of only digits and English letters.

Seen this question in a real interview before? 1/4

Yes No

Accepted 2.7M Submissions 8.2M Acceptance Rate 33.2%

Topics

Companies

Hint 1

28K 225

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    private int maxLen = 0;
    private int start = 0;
    public String sol1(String s) {
      for (int i = 0; i < s.length(); i++) {
        checkValid(s, i, i);
        checkValid(s, i, i+1);
      }
      return s.substring(start, start + maxLen);
    }

    public void checkValid(String s, int left, int right) {
      while(left >= 0 && right <= s.length() - 1 && s.charAt(left) == s.charAt(right)) {
        left--;
        right++;
      }
      int newLen = right - left - 1;
      if (newLen > maxLen) {
        maxLen = newLen;
        start = left + 1;
      }
    }

    public String longestPalindrome(String s) {
      //return sol1(s);
      return sol2(s);
    }
    public String sol2(String s) {
      int N = s.length();
      if (N == 0)
        return "";
      boolean[][] dp = new boolean[N][N];
      int maxLen = 1;
      int start = 0;
      for (int i = 0; i < N; i++) {
        dp[i][i] = true;
        for (int j = 0; j < i; j++) {
          if (s.charAt(i) == s.charAt(j) && ( i-j <= 1 || dp[j+1][i-1]))
              dp[j][i] = true;
          if (dp[j][i] && i-j+1 > maxLen) {
            maxLen = i-j+1;
            start = j;
          }

        }
      }
      return s.substring(start, start + maxLen);
    }
```

Saved to local

Testcase Test Result

6. Zigzag Conversion

Medium Topics Companies

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

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

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

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

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

Example 1:

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

Example 2:

```
Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
Explanation:
P     I    N
A   L S  I G
Y A   H R
P     I
```

Example 3:

```
Input: s = "A", numRows = 1
Output: "A"
```

Constraints:

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public String convert(String s, int numRows) {
        if(numRows==1)
            return s;
        StringBuilder[] sbs=new StringBuilder[numRows] ;
        for(int i=0; i<numRows; i++)
            sbs[i]=new StringBuilder();
        int curRow=0;
        int direction=1;
        for(int i=0; i<s.length(); i++) {
            char c = s.charAt(i);
            sbs[curRow].append(c);
            if(direction==1 && curRow==numRows-1) {

                direction=-1;
            } else if(direction==-1 && curRow==0) {

                direction=1;
            }
            curRow += direction;

        }
        StringBuilder ret = new StringBuilder();
        for(int i=0; i<numRows; i++)
            ret.append(sbs[i].toString());
        return ret.toString();
    }
}
```

Saved to local

Testcase Test Result

7. Reverse Integer

Medium Topics Companies

Given a signed 32-bit integer `x`, return `x` *with its digits reversed*. If reversing `x` causes the value to go outside the signed 32 range `[-2^31, 2^31 - 1]`, then return `0`.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

```
Input: x = 123
Output: 321
```

Example 2:

```
Input: x = -123
Output: -321
```

Example 3:

```
Input: x = 120
Output: 21
```

Constraints:

- `-2^31 <= x <= 2^31 - 1`

Seen this question in a real interview before? 1/4

Yes No

Accepted 2.9M Submissions 10.3M Acceptance Rate 28.0%

Topics

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public int reverse(int x) {
        int ret = 0;
        int newret = 0;
        while( x != 0 ) {
            newret = ret * 10 + x % 10;
            if( ( newret - x % 10 ) / 10 != ret )
                return 0;
            ret = newret;
            x = x / 10;

        }
        return ret;

    }
}
```

Saved to local

Testcase Test Result

8. String to Integer (atoi)

Medium Topics Companies

Implement the `myAtoi(string s)` function, which converts a string to a 32-bit signed integer (similar to C/C++'s `atoi` func

The algorithm for `myAtoi(string s)` is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is `'-'` or `'+'`. Read this character in if it is either. This c if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is igr
4. Convert these digits into an integer (i.e. `"123" -> 123`, `"0032" -> 32`). If no digits were read, then the integer is `0`. Char as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range `[-2^31, 2^31 - 1]`, then clamp the integer so that it remains in the ra Specifically, integers less than `-2^31` should be clamped to `-2^31`, and integers greater than `2^31 - 1` should be clamped to
6. Return the integer as the final result.

Note:

- Only the space character `' '` is considered a whitespace character.
- **Do not ignore** any characters other than the leading whitespace or the rest of the string after the digits.

Example 1:

```
Input: s = "42"
Output: 42
Explanation: The underlined characters are what is read in, the caret is the current read
position.
Step 1: "42" (no characters read because there is no leading whitespace)
         ^
Step 2: "42" (no characters read because there is neither a '-' nor '+')
         ^
Step 3: "42" ("42" is read in)
           ^
The parsed integer is 42.
```

Run Submit

Code

Java Auto

```java
class Solution {
    public int myAtoi(String str) {
      int i=0;
      while(i<str.length() && str.charAt(i)==' ') {

          i++;
      }
      if(i==str.length())
        return 0;
      int sign=1;
      if(str.charAt(i)=='-') {
        sign=-1;
        i++;
      }else if(str.charAt(i)=='+') {
        i++;
      }
      int ret=0;

      while(i<str.length()) {
        int n=str.charAt(i)-'0';
        if(n<0 || n>9)
          break;

        if(ret>Integer.MAX_VALUE/10 || ret==Integer.MAX_VALUE/10 && n>7){


          if(sign==1)
            return Integer.MAX_VALUE;
          else
            return Integer.MIN_VALUE;
        }
        ret=ret*10+n;

        i++;
      }
      return ret*sign;

    }
}
```

Saved to local

Testcase Test Result

9. Palindrome Number

Easy Topics Companies Hint

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

Example 2:

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

Example 3:

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

Constraints:

- $-2^{31} <= x <= 2^{31} - 1$

Follow up: Could you solve it without converting the integer to a string?

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public boolean isPalindrome(int x) {
      if (x<0 || (x!=0 && x%10==0)) return false;
      int rev = 0;
      while (x>rev){
        rev = rev*10 + x%10;
        x = x/10;
      }
      return (x==rev || x==rev/10);
    }
}
```

Saved to local

Testcase Test Result

10. Regular Expression Matching

Hard Topics Companies

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

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

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

Example 1:

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

Example 2:

```
Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeatin
it becomes "aa".
```

Example 3:

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

Constraints:

- `1 <= s.length <= 20`
- `1 <= p.length <= 20`
- `s` contains only lowercase English letters.
- `p` contains only lowercase English letters, `'.'`, and `'*'`.

Run Submit

Code

Java Auto

```java
class Solution {
    public boolean sol1(String s, String p) {
      //System.out.println(s+" "+p);
      if(p.isEmpty()) return s.isEmpty();

      boolean firstMatch=false;
      if(!s.isEmpty() && (s.charAt(0)==p.charAt(0) || p.charAt(0)=='.'))
        firstMatch=true;

      if(p.length()>=2 && p.charAt(1) =='*' ) {
        return isMatch(s, p.substring(2)) ||
          (firstMatch && isMatch(s.substring(1),p));
      }else {
        return firstMatch && isMatch(s.substring(1), p.substring(1));
      }
    }
    public boolean sol2(String s, String p) {
      boolean[][] dp = new boolean[s.length()+1][p.length()+1];
      dp[0][0]=true;
      for (int i = 0; i < p.length(); i++) {
        if (p.charAt(i) == '*' && dp[0][i-1]) {
            dp[0][i+1] = true;
        }
      }

      for(int i=0; i<s.length(); i++) {
        for(int j=0; j<p.length(); j++) {
          if(j>0 && p.charAt(j)=='*') {
            dp[i+1][j+1]=dp[i+1][j-1] //skip .* in pattern
              || (dp[i][j+1] && (s.charAt(i)==p.charAt(j-1) || p.charAt(j-1)=='.'));  //skip in S because of ma
          }//*
          else {
            dp[i+1][j+1]=dp[i][j] && (s.charAt(i)==p.charAt(j) || p.charAt(j)=='.');
          }//not *
        }//for j
      }//for i

      return dp[s.length()][p.length()];
    }

    public boolean isMatch(String s, String p) {
      //return sol1(s,p);
      return sol2(s,p);
    }

}
```

Saved to local

Testcase Test Result

Description Editorial Solutions Submissions

11. Container With Most Water

Medium Topics Companies Hint

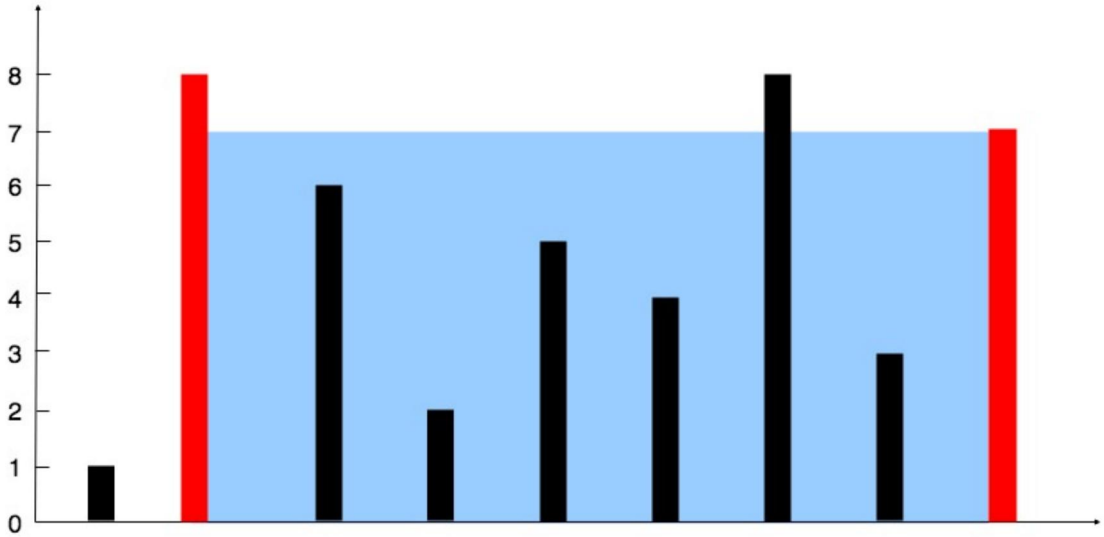
You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `i^th` `0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*.

Notice that you may not slant the container.

Example 1:




```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In th
the max area of water (blue section) the container can contain is 49.
```

Example 2:

```
Input: height = [1,1]
Output: 1
```

27.2K 174

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public int maxArea(int[] height) {
      int ret = 0;
      int left = 0;
      int right=height.length-1;
      while(left<right) {
        int area=Math.min(height[left], height[right]) * (right-left);
        if(area>ret)
          ret=area;
        if(height[left]<height[right])
          left++;
        else
          right--;
      }

      return ret;

    }
}
```

Saved to local

Testcase Test Result

12. Integer to Roman

Medium Topics Companies

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

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

For example, `2` is written as `II` in Roman numeral, just two one's added together. `12` is written as `XII`, which is simply `X +` number `27` is written as `XXVII`, which is `XX + V + II`.

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

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

Given an integer, convert it to a roman numeral.

Example 1:

```
Input: num = 3
Output: "III"
Explanation: 3 is represented as 3 ones.
```

Example 2:

```
Input: num = 58
Output: "LVIII"
Explanation: L = 50, V = 5, III = 3.
```

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public String intToRoman(int num) {
      String[] syms=new String[]{ "M", "D", "C", "L", "X", "V", "I"};
      int[] nums=new int[]{1000,500,100,50,10, 5,1};
      StringBuilder sb = new StringBuilder();
      for(int i=0; i<nums.length; i+=2) {

        int n=num/nums[i];
        if(n>0 && n<4) {
          for(int j=0; j<n; j++)
            sb.append(syms[i]);
        }else if (n==4 ) {
          sb.append(syms[i]);
          sb.append(syms[i-1]);
        }else if (n>4 && n <9 ) {
          sb.append(syms[i-1]);
          for(int j=0; j<n-5; j++)
            sb.append(syms[i]);
        }else if (n==9 ) {
          sb.append(syms[i]);
          sb.append(syms[i-2]);
        }
        num=num%nums[i];

      }
      return sb.toString();

    }
}
```

Saved to local

Testcase Test Result

13. Roman to Integer

Easy Topics Companies Hint

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

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

For example, `2` is written as `II` in Roman numeral, just two ones added together. `12` is written as `XII`, which is simply `X +` number `27` is written as `XXVII`, which is `XX + V + II`.

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

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

Given a roman numeral, convert it to an integer.

Example 1:

```
Input: s = "III"
Output: 3
Explanation: III = 3.
```

Example 2:

```
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.
```

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public int romanToInt(String s) {
      Map<Character, Integer> map = new HashMap<>();
      map.put('I',1);
      map.put('V',5);
      map.put('X',10);
      map.put('L',50);
      map.put('C',100);
      map.put('D',500);
      map.put('M',1000);
      int ret=0;
      for(int i=0; i<s.length(); i++) {
        if(i== s.length()-1 || map.get(s.charAt(i+1) ) <= map.get(s.charAt(i))) {
          ret+=map.get(s.charAt(i));
        }else{
          ret-=map.get(s.charAt(i));
        }
      }
      return ret;

    }
}
```

Saving...

Testcase Test Result

Description Editorial Solutions Submissions

14. Longest Common Prefix

Easy Topics Companies

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

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

Example 1:

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

Example 2:

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

Constraints:

- `1 <= strs.length <= 200`
- `0 <= strs[i].length <= 200`
- `strs[i]` consists of only lowercase English letters.

Seen this question in a real interview before? 1/4

Yes No

Accepted 2.8M Submissions 6.8M Acceptance Rate 41.8%

Topics

Companies

16.3K 179

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public String longestCommonPrefix(String[] strs) {
      if(strs.length==0) return "";
      StringBuilder sb=new StringBuilder()    ;
      int i=0;
      boolean done=false;
      while(!done) {
        boolean same=true;
        char c=0;
        for(int j=0; j<strs.length; j++) {
          String s=strs[j];
          if(i==s.length()) {
            same=false;
            break;
          }
          if(j==0) {
            c=s.charAt(i);
          }else{
            if(c!=s.charAt(i)) {
              same=false;
              break;
            }
          }

        }
        if(same) {
          sb.append(c);
          i++;
        }
        else {
          done=true;
        }


      }


      return sb.toString();

    }
}
```

Saved to local

Testcase Test Result

15. 3Sum

Medium Topics Companies Hint

Given an integer array nums, return all the triplets `[nums[i], nums[j], nums[k]]` such that `i != j`, `i != k`, and `j != k`, `+ nums[j] + nums[k] == 0`.

Notice that the solution set must not contain duplicate triplets.

Example 1:

```
Input: nums = [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
Explanation:
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-1,2].
Notice that the order of the output and the order of the triplets does not matter.
```

Example 2:

```
Input: nums = [0,1,1]
Output: []
Explanation: The only possible triplet does not sum up to 0.
```

Example 3:

```
Input: nums = [0,0,0]
Output: [[0,0,0]]
Explanation: The only possible triplet sums up to 0.
```

Constraints:

- `3 <= nums.length <= 3000`
- `-10^5 <= nums[i] <= 10^5`

Code

Java Auto

```java
class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
      List<List<Integer>> ret = new ArrayList<>();
      Arrays.sort(nums);
      int n = nums.length;
      int left = 0;
      int right = 0;
      for(int i=0; i<n-2; i++) {

        if(nums[i]>0) break;//all rest are positive numbers, impossible to have answers.

        if(i>0 && nums[i-1] == nums[i] ) continue; //remove duplicate answers, as the 1st element has been process


        left = i+1;
        right = n-1;
        while(left<right) {
          int sum=nums[i]+nums[left]+nums[right];
          if(sum==0) {
            ret.add(Arrays.asList(nums[i],nums[left],nums[right]));

            //avoid dups
            while(left<right&&nums[left]==nums[left+1])
              left++;
            //avoid dups
            while(left<right&&nums[right]==nums[right-1])
              right--;

            left++;
            right--;
          }
          if(sum<0) left++;
          if(sum>0) right--;
        }
      }
      return ret;

    }
}
```

Saved to local

Testcase Test Result

16. 3Sum Closest

Medium Topics Companies

Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to

Return *the sum of the three integers*.

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

Example 1:

```
Input: nums = [-1,2,1,-4], target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

Example 2:

```
Input: nums = [0,0,0], target = 1
Output: 0
Explanation: The sum that is closest to the target is 0. (0 + 0 + 0 = 0).
```

Constraints:

- `3 <= nums.length <= 500`
- `-1000 <= nums[i] <= 1000`
- `-10^4 <= target <= 10^4`

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public int threeSumClosest(int[] nums, int target) {
      int n = nums.length;
      int ret = 0;
      int diff = Integer.MAX_VALUE;
      Arrays.sort(nums);
      for(int i=0; i<n-2; i++) {
        int left=i+1;
        int right=n-1;

        while(left<right) {
          int sum =  nums[i] + nums[left] + nums[right];
          if(Math.abs(sum-target)<diff) {
            ret = sum;
            diff=Math.abs(sum-target);

          }
          if(sum<target) left++;
          if(sum>target) right--;
          if(sum==target) return sum;
        }
      }
      return ret;

    }
}
```

Saved to local

Testcase Test Result

17. Letter Combinations of a Phone Number

Medium Topics Companies

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

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

Example 1:

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

Example 2:

```
Input: digits = ""
Output: []
```

Example 3:

```
Input: digits = "2"
Output: ["a","b","c"]
```

Constraints:

Code

Java Auto

```java
class Solution {
    public List<String> letterCombinations(String digits) {
       return sol1(digits);


      //return sol2(digits);
    }
    public List<String> sol2(String digits) {
      LinkedList<String> ans = new LinkedList<String>();
      if(digits.isEmpty()) return ans;
      String[] mapping = new String[] {"0", "1", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
      ans.add("");
      while(ans.peek().length()!=digits.length()){
        String remove = ans.remove();
        String map = mapping[digits.charAt(remove.length())-'0'];
        for(char c: map.toCharArray()){
          ans.addLast(remove+c);
        }
      }
      return ans;

    }

    public List<String> sol1(String digits) {
      List<String> ret = new ArrayList<>();
      if(digits.length()==0)
        return ret;

      String[] map = new String[]{"0", "1", "abc","def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
      helper(ret, map, digits, 0, "");
      return ret;
    }
    public void helper(List<String> ret, String[] map, String digits, int index, String curStr ) {
      if(index==digits.length()) {
        ret.add(curStr);
        return;
      }
      int n = digits.charAt(index)-'0';
      String s=map[n];
      for(int i=0; i<s.length(); i++){
        helper(ret, map, digits, index+1, curStr+s.charAt(i));
      }

    }
}
```

Saving...

Testcase Test Result

Description Editorial Solutions Submissions

18. 4Sum

Medium Topics Companies

Given an array `nums` of `n` integers, return *an array of all the **unique** quadruplets* `[nums[a], nums[b], nums[c], nums[d]]`

- `0 <= a, b, c, d < n`
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Example 1:

```
Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
```

Example 2:

```
Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]
```

Constraints:

- `1 <= nums.length <= 200`
- `-10^9 <= nums[i] <= 10^9`
- `-10^9 <= target <= 10^9`

Seen this question in a real interview before? 1/4

Yes No

Accepted 838.6K Submissions 2.3M Acceptance Rate 35.8%

10.7K 112

Problem List

Run Submit

Code

Java Auto

```java
class Solution {
    public List<List<Integer>> fourSum(int[] nums, int target) {
      List<List<Integer>> ret = new ArrayList<>();
      Arrays.sort(nums);
      System.out.println(Arrays.toString(nums));
      for(int i=0;i<nums.length-3;i++){
        if(i>0 && nums[i] == nums[i-1])
          continue;
        for(int j=i+1;j<nums.length-2;j++) {
          if(j>i+1 && nums[j]==nums[j-1])
            continue;
          int left = j+1;
          int right = nums.length -1;
          while(left<right) {
            int tempVal = nums[i]+nums[j]+nums[left]+nums[right];
            if(tempVal == target) {
              // ret.add(new ArrayList<>(Arrays.asList(nums[i], nums[j], nums[left], nums[right])));
              ret.add(Arrays.asList(nums[i], nums[j], nums[left], nums[right]));
              while(left<right && nums[left]==nums[left+1])
                ++left;
              while(left<right && nums[right]==nums[right-1])
                --right;
              ++left;
              --right;
            }else if(tempVal < target) {
              left++;
            }else {
              right--;
            }

          }
        }
      }
      return ret;


    }
}


//K sum problem: change to K-1 sum problem, until k=2

    // public class Solution {
    //     int len = 0;
    //     public List<List<Integer>> fourSum(int[] nums, int target) {
    //         len = nums.length;
    //         Arrays.sort(nums);
    //         return kSum(nums, target, 4, 0);
```

Saved to local

Testcase Test Result

Run Submit

Code

Java Auto

```
//K sum problem: change to K-1 sum problem, until k=2
    // public class Solution {
    //     int len = 0;
    //     public List<List<Integer>> fourSum(int[] nums, int target) {
    //         len = nums.length;
    //         Arrays.sort(nums);
    //         return kSum(nums, target, 4, 0);
    //     }
    //    private ArrayList<List<Integer>> kSum(int[] nums, int target, int k, int index) {
    //         ArrayList<List<Integer>> res = new ArrayList<List<Integer>>();
    //         if(index >= len) {    return res;    }
    //         if(k == 2) {
    //          int i = index, j = len - 1;
    //          while(i < j) {
    //                 //find a pair
    //              if(target - nums[i] == nums[j]) {
    //                List<Integer> temp = new ArrayList<>();
    //                  temp.add(nums[i]);
    //                  temp.add(target-nums[i]);
    //                     res.add(temp);
    //                     //skip duplication
    //                     while(i<j && nums[i]==nums[i+1]) i++;
    //                     while(i<j && nums[j-1]==nums[j]) j--;
    //                     i++;
    //                     j--;
    //                 //move left bound
    //              } else if (target - nums[i] > nums[j]) {                  i++;
    //                 //move right bound
    //              } else {                  j--;
    //              }
    //          }
    //         } else{
    //             for (int i = index; i < len - k + 1; i++) {
    //                 //use current number to reduce ksum into k-1sum
    //                 ArrayList<List<Integer>> temp = kSum(nums, target - nums[i], k-1, i+1);
    //                 if(temp != null){
    //                     //add previous results
    //                     for (List<Integer> t : temp) {
    //                         t.add(0, nums[i]);
    //                     }
    //                     res.addAll(temp);
    //                 }
    //                 while (i < len-1 && nums[i] == nums[i+1]) {
    //                     //skip duplicated numbers
    //                     i++;
    //                 }
    //             }
    //         }
    //         return res;
```

Saved to local

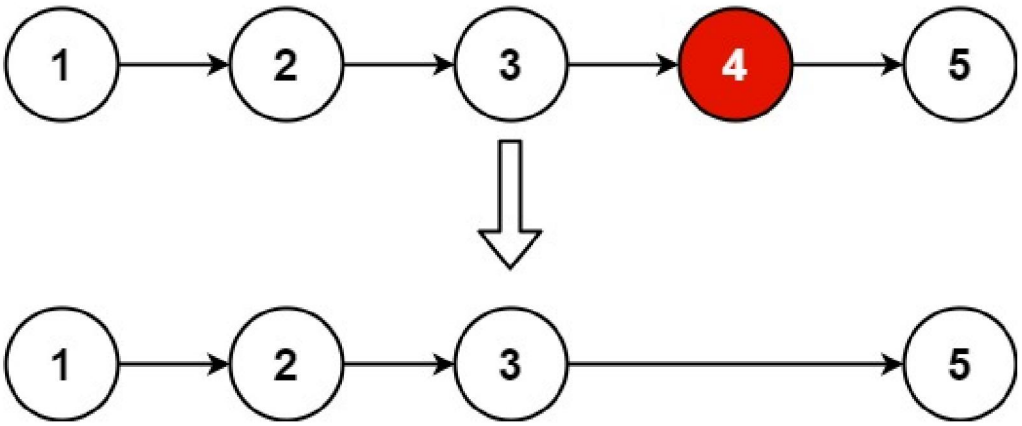
Testcase Test Result

19. Remove Nth Node From End of List

Medium Topics Companies Hint

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

Example 1:




```
Input: head = [1,2,3,4,5], n = 2
Output: [1,2,3,5]
```

Example 2:

```
Input: head = [1], n = 1
Output: []
```

Example 3:

```
Input: head = [1,2], n = 1
Output: [1]
```

Constraints:

- The number of nodes in the list is `sz`.
- `1 <= sz <= 30`

Run Submit

Code

Java Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode removeNthFromEnd(ListNode head, int n) {
      ListNode right = head;
      int i=1;
      while(i++<=n && right!=null) {
        right=right.next;

      }

      if(right == null && i>=n)  {
        return head.next;
      }

      if(right == null && i<n)  {
        return head;
      }
      ListNode left=head;
      while(right.next!=null) {
        right=right.next;
        left=left.next;
      }

      left.next=left.next.next;
      return head;

    }
}
```

Saving...

Testcase Test Result