算法

数组

二分查找

704. 二分查找

已解答

简单 相关标签 相关企业

给定一个 `n` 个元素有序的（升序）整型数组 `nums` 和一个目标值 `target` ，写一个函数搜索 `nums` 中的 `target`，如果目标值存在返回下标，否则返回 `-1`。

示例 1:

> **输入:** `nums = [-1,0,3,5,9,12], target = 9`
> **输出:** 4
> **解释:** 9 出现在 `nums` 中并且下标为 4

示例 2:

> **输入:** `nums = [-1,0,3,5,9,12], target = 2`
> **输出:** -1
> **解释:** 2 不存在 `nums` 中因此返回 -1

Go 智能模式

```go
func search(nums []int, target int) int {
    left:=0
    right:=len(nums)-1

    for left<=right{
        mid:=left+(right-left)/2
        if nums[mid]<target{
            left=mid+1
        }else if nums[mid]>target{
            right=mid-1
        }else{
            return mid
        }
    }
    return -1
}
```

35. 搜索插入位置

已解答

简单 相关标签 相关企业

给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。

请必须使用时间复杂度为 `O(log n)` 的算法。

示例 1:

> **输入:** nums = [1,3,5,6], target = 5
> **输出:** 2

示例 2:

> **输入:** nums = [1,3,5,6], target = 2
> **输出:** 1

示例 3:

> **输入:** nums = [1,3,5,6], target = 7
> **输出:** 4

Go 智能模式

```go
func searchInsert(nums []int, target int) int {
    left:=0
    right:=len(nums)-1
    for left<=right{
        mid :=left+(right-left)/2
        if nums[mid]<target{
            left=mid+1
        }else if nums[mid]>target{
            right=mid-1
        }else{
            return mid
        }
    }
    return right+1
}
```

已存储

34. 在排序数组中查找元素的第一个和最后一个位置

已解答

中等 相关标签 相关企业

给你一个按照非递减顺序排列的整数数组 `nums`，和一个目标值 `target`。请你找出给定目标值在数组中的开始位置和结束位置。

如果数组中不存在目标值 `target`，返回 `[-1, -1]`。

你必须设计并实现时间复杂度为 `O(log n)` 的算法解决此问题。

示例 1：

> **输入：** `nums = [5,7,7,8,8,10], target = 8`
> **输出：** `[3,4]`

示例 2：

> **输入：** `nums = [5,7,7,8,8,10], target = 6`
> **输出：** `[-1,-1]`

示例 3：

> **输入：** `nums = [], target = 0`
> **输出：** `[-1,-1]`

提示：

- `0 <= nums.length <= 10^5`
- `-10^9 <= nums[i] <= 10^9`
- `nums` 是一个非递减数组
- `-10^9 <= target <= 10^9`

面试中遇到过这道题？ 1/5

Go 智能模式

```go
func searchRange(nums []int, target int) []int {
    leftBorder:=leftBorder(nums,target)
    rightBorder:=rightBorder(nums,target)
    if leftBorder==-2&&rightBorder==-2{
        return []int{-1,-1}
    }
    if (rightBorder-leftBorder)>1{
        return []int{leftBorder+1,rightBorder-1}
    }
    return []int{-1,-1}
}

func leftBorder(nums []int, target int) int {
    left:=0
    right:=len(nums)-1
    leftBorder:=-2
    for left<=right{
        mid :=left+(right-left)/2
        if nums[mid]<target{
            left=mid+1
        }else {
            right=mid-1
            leftBorder=right
        }
    }
    return leftBorder
}

func rightBorder(nums []int, target int) int {
    left:=0
    right:=len(nums)-1
    rightBorder:=-2
    for left<=right{
        mid :=left+(right-left)/2
        if nums[mid]<target{
            left=mid+1
            rightBorder=left
        }else {
            right=mid-1
        }
    }
    return rightBorder
}
```

69. x 的平方根

简单 相关标签 相关企业 提示

给你一个非负整数 `x`，计算并返回 `x` 的 **算术平方根**。

由于返回类型是整数，结果只保留 **整数部分**，小数部分将被 **舍去**。

注意： 不允许使用任何内置指数函数和算符，例如 `pow(x, 0.5)` 或者 `x ** 0.5`。

示例 1：

> **输入：** `x = 4`
> **输出：** `2`

示例 2：

> **输入：** `x = 8`
> **输出：** `2`
> **解释：** `8` 的算术平方根是 `2.82842...`，由于返回类型是整数，小数部分将被舍去。

Go 智能模式

```go
func mySqrt(x int) int {
    left:=0
    right:=x
    for left<=right{
        mid:=left+(right-left)/2
        if mid*mid>x{
            right=mid-1
        } else if mid*mid<x{
            left=mid+1
        }else{
            return mid
        }
    }
    return right
}
```

367. 有效的完全平方数

已解答

简单 相关标签 相关企业

给你一个正整数 `num`。如果 `num` 是一个完全平方数，则返回 `true`，否则返回 `false`。

完全平方数 是一个可以写成某个整数的平方的整数。换句话说，它可以写成某个整数和自身的乘积。

不能使用任何内置的库函数，如 `sqrt`。

示例 1：

> **输入：** `num = 16`
> **输出：** `true`
> **解释：** 返回 `true`，因为 `4 * 4 = 16` 且 `4` 是一个整数。

示例 2：

> **输入：** `num = 14`
> **输出：** `false`
> **解释：** 返回 `false`，因为 `3.742 * 3.742 = 14` 但 `3.742` 不是一个整数。

Go 智能模式

```go
func isPerfectSquare(num int) bool {
    left:=0
    right:=num
    for left<=right{
        mid:=left+(right-left)/2
        if mid*mid>num{
            right=mid-1
        }else if mid*mid<num{
            left=mid+1
        }else{
            return true
        }
    }
    return false
}
```

已存储 升级云端代码存储

移除元素-双指针

示例 1：

> **输入：** nums = [3,2,2,3], val = 3
> **输出：** 2, nums = [2,2,_,_]
> **解释：** 你的函数函数应该返回 k = 2, 并且 nums 中的前两个元素均为 2。
> 你在返回的 k 个元素之外留下了什么并不重要（因此它们并不计入评测）。

示例 2：

> **输入：** nums = [0,1,2,2,3,0,4,2], val = 2
> **输出：** 5, nums = [0,1,4,0,3,_,_,_]
> **解释：** 你的函数应该返回 k = 5, 并且 nums 中的前五个元素为 0,0,1,3,4。
> 注意这五个元素可以任意顺序返回。
> 你在返回的 k 个元素之外留下了什么并不重要（因此它们并不计入评测）。

Go 智能模式

```go
func removeElement(nums []int, val int) int {
    slow:=0
    for fast:=0;fast<len(nums);fast++{
        if nums[fast]!=val{
            nums[slow]=nums[fast]
            slow++
        }
    }
    return slow
}
```

示例 1：

> **输入：** nums = [1,1,2]
> **输出：** 2, nums = [1,2,_]
> **解释：** 函数应该返回新的长度 `2` ，并且原数组 *nums* 的前两个元素被修改为 `1`, `2` 。不需要考虑数组中超出新长度后面的元素。

示例 2：

> **输入：** nums = [0,0,1,1,1,2,2,3,3,4]
> **输出：** 5, nums = [0,1,2,3,4]
> **解释：** 函数应该返回新的长度 `5` ， 并且原数组 *nums* 的前五个元素被修改为 `0`, `1`, `2`, `3`, `4` 。不需要考虑数组中超出新长度后面的元素。

Go 智能模式

```go
func removeDuplicates(nums []int) int {
    n:=len(nums)
    if n<1{
        return 0
    }
    slow := 1
    for fast := 1; fast < len(nums); fast++ {
        if nums[fast] != nums[fast-1] {
            nums[slow] = nums[fast]
            slow++
        }
    }
    return slow
}
```

283. 移动零

已解答

简单 | 相关标签 | 相关企业 | 提示 | A文

给定一个数组 `nums`，编写一个函数将所有 `0` 移动到数组的末尾，同时保持非零元素的相对顺序。

请注意 ，必须在不复制数组的情况下原地对数组进行操作。

示例 1：

> **输入：** nums = `[0,1,0,3,12]`
> **输出：** `[1,3,12,0,0]`

示例 2：

> **输入：** nums = `[0]`
> **输出：** `[0]`

Go 智能模式

```go
func moveZeroes(nums []int)  {
    slow := 0
    for fast := 0; fast < len(nums); fast++ {
        if nums[fast] != 0 {
            nums[slow] = nums[fast]
            slow++
        }
    }
    for i := slow; i < len(nums); i++ {
        nums[i] = 0
    }
}
```

844. 比较含退格的字符串

已解答

简单 相关标签 相关企业

给定 `s` 和 `t` 两个字符串，当它们分别被输入到空白的文本编辑器后，如果两者相等，返回 `true` 。`#` 代表退格字符。

注意：如果对空文本输入退格字符，文本继续为空。

示例 1：

> **输入：** s = "ab#c", t = "ad#c"
> **输出：** true
> **解释：** s 和 t 都会变成 "ac"。

示例 2：

> **输入：** s = "ab##", t = "c#d#"
> **输出：** true
> **解释：** s 和 t 都会变成 ""。

示例 3：

> **输入：** s = "a#c", t = "b"
> **输出：** false
> **解释：** s 会变成 "c"，但 t 仍然是 "b"。

Go 智能模式

```go
func backspaceCompare(s string, t string) bool {
    resultS := simplify(s)
    resultT := simplify(t)
    return resultS == resultT
}

func simplify(s string) string {
    bs:=[]byte(s)
    slow := 0
        for fast := 0; fast < len(bs); fast++ {
            if bs[fast] != '#' {
                bs[slow] = bs[fast]
                slow++
            } else {
                if slow > 0 {
                    slow--
                }
            }
        }
return string(bs[:slow])
}

// func simplify(s string) string {
```

已存储

测试用例 测试结果

977. 有序数组的平方

已解答

简单 相关标签 相关企业

给你一个按 **非递减顺序** 排序的整数数组 `nums`，返回 **每个数字的平方** 组成的新数组，要求也按 **非递减顺序** 排序。

示例 1：

> **输入：** nums = [-4,-1,0,3,10]
> **输出：** [0,1,9,16,100]
> **解释：** 平方后，数组变为 [16,1,0,9,100]
> 排序后，数组变为 [0,1,9,16,100]

示例 2：

> **输入：** nums = [-7,-3,2,3,11]
> **输出：** [4,9,9,49,121]

Go 智能模式

```go
func sortedSquares(nums []int) []int {
    front := 0
    backend, count := len(nums)-1, len(nums)-1
    sortNums := make([]int, count+1)
    for front <= backend {
        if nums[front]*nums[front] < nums[backend]*nums[backend] {
            sortNums[count] = nums[backend] * nums[backend]
            backend--
        } else {
            sortNums[count] = nums[front] * nums[front]
            front++
        }
        count--
    }
    return sortNums
}
```

长度最小的子数组-滑动窗口

209. 长度最小的子数组

已解答

中等 相关标签 相关企业

给定一个含有 `n` 个正整数的数组和一个正整数 `target` 。

找出该数组中满足其总和大于等于 `target` 的长度最小的 **子数组** $[num s_l, num s_{l+1}, ..., num s_{r-1}, num s_r]$ ，并返回其长度。如果不存在符合条件的子数组，返回 `0` 。

示例 1：

> **输入：** target = 7, nums = [2,3,1,2,4,3]
> **输出：** 2
> **解释：** 子数组 [4,3] 是该条件下的长度最小的子数组。

示例 2：

> **输入：** target = 4, nums = [1,4,4]
> **输出：** 1

Go 智能模式

```go
func minSubArrayLen(target int, nums []int) int {
    fast, slow := 0, 0
    sum := 0
    minLen := math.MaxInt
    for fast < len(nums) {
        sum += nums[fast]
        for sum >= target {
            sum -= nums[slow]
            minLen = min(minLen, fast-slow+1)
            slow++
        }
        fast++
    }

    if minLen == math.MaxInt {
        return 0
    }
    return minLen
}
```

满足条件的最大数组

输入： fruits = [**1,2,1**]
输出： 3
解释： 可以采摘全部 3 棵树。

示例 2：

输入： fruits = [0,**1,2,2**]
输出： 3
解释： 可以采摘 [1,2,2] 这三棵树。
如果从第一棵树开始采摘，则只能采摘 [0,1] 这两棵树。

示例 3：

输入： fruits = [1,**2,3,2,2**]
输出： 4
解释： 可以采摘 [2,3,2,2] 这四棵树。
如果从第一棵树开始采摘，则只能采摘 [1,2] 这两棵树。

示例 4：

输入： fruits = [3,3,3,**1,2,1,1,2**,3,3,4]
输出： 5
解释： 可以采摘 [1,2,1,1,2] 这五棵树。

Go 智能模式

```go
func totalFruit(fruits []int) int {
    left, right, res := 0, 0, 0
    n := len(fruits)
    m := make(map[int]int)
    for right < n {
        m[fruits[right]]++
        if len(m) > 2 {
            m[fruits[left]]--
            if m[fruits[left]] == 0 {
                delete(m, fruits[left])
            }
            left++
        }
        res = max(res, right-left+1)
        right++
    }
    return res
}
```

最小覆盖子串

76. 最小覆盖子串

已解答

困难 相关标签 相关企业 提示 Az

给你一个字符串 `s` 、一个字符串 `t` 。返回 `s` 中涵盖 `t` 所有字符的最小子串。如果 `s` 中不存在涵盖 `t` 所有字符的子串，则返回空字符串 `""` 。

注意：

- 对于 `t` 中重复字符，我们寻找的子字符串中该字符数量必须不少于 `t` 中该字符数量。
- 如果 `s` 中存在这样的子串，我们保证它是唯一的答案。

示例 1：

> **输入：** s = "ADOBECODEBANC", t = "ABC"
> **输出：** "BANC"
> **解释：** 最小覆盖子串 "BANC" 包含来自字符串 t 的 'A'、'B' 和 'C'。

示例 2：

> **输入：** s = "a", t = "a"
> **输出：** "a"
> **解释：** 整个字符串 s 是最小覆盖子串。

示例 3：

> **输入：** s = "a", t = "aa"
> **输出：** ""
> **解释：** t 中两个字符 'a' 均应包含在 s 的子串中，
> 因此没有符合条件的子字符串，返回空字符串。

提示：

Go 智能模式

```go
func minWindow(s string, t string) string {
    left, right, match := 0, 0, 0
    need := make(map[byte]int)
    win := make(map[byte]int)
    minLen := math.MaxInt
    start := 0
    for i := range t {
        need[t[i]]++
    }

    for right < len(s) {
        c := s[right]
        right++
        if _, ok := need[c]; ok {
            win[c]++
            if win[c] == need[c] {
                match++
            }
        }

        for match == len(need) {
            if right-left < minLen {
                minLen = right - left
                start = left
            }
            c = s[left]
            left++
            if _, ok := need[c]; ok{
                if win[c] == need[c] {
                    match--
                }
                win[c]--
            }
        }
    }
    if minLen == math.MaxInt {
        return ""
    }
    return s[start : start+minLen]
}
```

螺旋矩阵

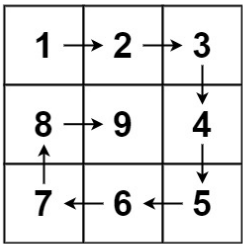
59. 螺旋矩阵 II

已解答

中等 相关标签 相关企业

给你一个正整数 `n` ，生成一个包含 `1` 到 n^2 所有元素，且元素按顺时针顺序螺旋排列的 `n x n` 正方形矩阵 `matrix` 。

示例 1：




```
输入：n = 3
输出：[[1,2,3],[8,9,4],[7,6,5]]
```

示例 2：

```
输入：n = 1
输出：[[1]]
```

提示：

```go
func generateMatrix(n int) [][]int {
    top, right, bottom, left := 0, n-1, n-1, 0
    arr := make([][]int, n)
    for i := range arr {
        arr[i] = make([]int, n)
    }
    var i int
    count := 1
    for i < n*n {
        for j := left; j <= right && i < n*n; j++ {
            arr[top][j] = count
            count++
            i++
        }
        top++
        for j := top; j <= bottom && i < n*n; j++ {
            arr[j][right] = count
            count++
            i++
        }
        right--
        for j := right; j >= left && i < n*n; j-- {
            arr[bottom][j] = count
            count++
            i++
        }
        bottom--
        for j := bottom; j >= top && i < n*n; j-- {
            arr[j][left] = count
            count++
            i++
        }
        left++
    }
    return arr
}
```

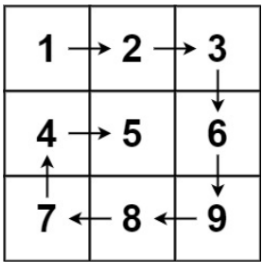
54. 螺旋矩阵

已解答

中等 相关标签 相关企业 提示

给你一个 `m` 行 `n` 列的矩阵 `matrix` ，请按照 **顺时针螺旋顺序** ，返回矩阵中的所有元素。

示例 1：




```
输入：matrix = [[1,2,3],[4,5,6],[7,8,9]]
输出：[1,2,3,6,9,8,7,4,5]
```

示例 2：

Go 智能模式

```go
func spiralOrder(matrix [][]int) []int {
    a := len(matrix)
    b := len(matrix[0])
    top, left, bottom, right := 0, 0, a-1, b-1
    var i int
    var arr []int
    for i < a*b {
        for j := left; j <= right && i < a*b; j++ {
            arr = append(arr, matrix[top][j])
            i++
        }
        top++
        for j := top; j <= bottom && i < a*b; j++ {
            arr = append(arr, matrix[j][right])
            i++
        }
        right--
        for j := right; j >= left && i < a*b; j-- {
            arr = append(arr, matrix[bottom][j])
            i++
        }
        bottom--
        for j := bottom; j >= top && i < a*b; j-- {
            arr = append(arr, matrix[j][left])
            i++
        }
        left++
    }
    return arr
}
```

LCR 146. 螺旋遍历二维数组

已解答

简单 相关标签 相关企业

给定一个二维数组 `array`，请返回「**螺旋遍历**」该数组的结果。

螺旋遍历：从左上角开始，按照 **向右**、**向下**、**向左**、**向上** 的顺序 **依次** 提取元素，然后再进入内部一层重复相同的步骤，直到提取完所有元素。

示例 1：

```
输入：array = [[1,2,3],[8,9,4],[7,6,5]]
输出：[1,2,3,4,5,6,7,8,9]
```

示例 2：

```
输入：array  = [[1,2,3,4],[12,13,14,5],[11,16,15,6],[10,9,8,7]]
输出：[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
```

限制：

- `0 <= array.length <= 100`
- `0 <= array[i].length <= 100`

注意：本题与主站 54 题相同：https://leetcode-cn.com/problems/spiral-matrix/

```go
func spiralArray(array [][]int) []int {
    if len(array) == 0 || (len(array) != 0 && len(array[0]) == 0) {
        return []int{}
    }
    a := len(array)
    b := len(array[0])
    top, right, bottom, left := 0, b-1, a-1, 0
    var i int
    var nums []int
    for i < a*b {
        for j := left; j <= right && i < a*b; j++ {
            nums = append(nums, array[top][j])
            i++
        }
        top++
        for j := top; j <= bottom && i < a*b; j++ {
            nums = append(nums, array[j][right])
            i++
        }
        right--
        for j := right; j >= left && i < a*b; j-- {
            nums = append(nums, array[bottom][j])
            i++
        }
        bottom--
        for j := bottom; j >= top && i < a*b; j-- {
            nums = append(nums, array[j][left])
            i++
        }
        left++
    }
    return nums
}
```

从键盘输入

```go
// 用go实现第一行输入为整数数组 Array 的长度 n，接下来 n 行，每行一个整数，表示数组的元素。随后的输入为需要计算总和的区间下标：a，b （b > = a），直至文
func main1() {  no usages
    scanner := bufio.NewScanner(os.Stdin)

    // 读取数组长度
    scanner.Scan()
    lenArr, err := strconv.Atoi(scanner.Text())
    if err != nil {
        fmt.Println( a...: "Error reading array length:", err)
        return
    }

    // 读取数组元素
    array := make([]int, lenArr)
    for i := 0; i < lenArr; i++ {
        scanner.Scan()
        num, err := strconv.Atoi(scanner.Text())
        if err != nil {
            fmt.Println( a...: "Error reading array element:", err)
            return
        }
        array[i] = num
    }

    //下标
    for scanner.Scan() {
        line := scanner.Text()
        indices := strings.Split(line,  sep: " ")
        scanner := 0
```

链表

移除链表元素

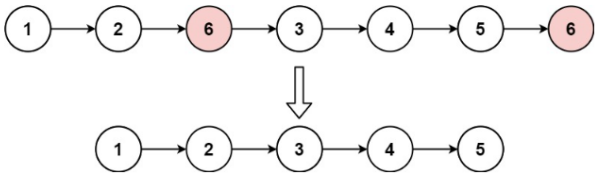
203. 移除链表元素

已解答

简单 相关标签 相关企业

给你一个链表的头节点 `head` 和一个整数 `val` ，请你删除链表中所有满足 `Node.val == val` 的节点，并返回 **新的头节点** 。

示例 1：



输入： head = [1,2,6,3,4,5,6], val = 6
输出： [1,2,3,4,5]

示例 2：

输入： head = [], val = 1
输出： []

Go 智能模式

```go
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func removeElements(head *ListNode, val int) *ListNode {
    dummyHead:=&ListNode{}
    dummyHead.Next=head
    cur:=dummyHead
    for cur.Next!=nil{
        if cur.Next.Val==val{
            cur.Next=cur.Next.Next
        } else {
            cur = cur.Next
        }
    }
    return dummyHead.Next
}
```

设计链表

- `void addAtIndex(int index, int val)` 将一个值为 `val` 的节点插入到链表中下标为 `index` 的节点之前。如果 `index` 等于链表的长度，那么该节点会被追加到链表的末尾。如果 `index` 比长度更大，该节点将 **不会插入** 到链表中。
- `void deleteAtIndex(int index)` 如果下标有效，则删除链表中下标为 `index` 的节点。

示例：

输入
```
["MyLinkedList", "addAtHead", "addAtTail", "addAtIndex", "get", "deleteAtIndex", "get"]
[[], [1], [3], [1, 2], [1], [1], [1]]
```
输出
```
[null, null, null, null, 2, null, 3]
```

解释
```
MyLinkedList myLinkedList = new MyLinkedList();
myLinkedList.addAtHead(1);
myLinkedList.addAtTail(3);
myLinkedList.addAtIndex(1, 2);    // 链表变为 1->2->3
myLinkedList.get(1);              // 返回 2
myLinkedList.deleteAtIndex(1);    // 现在，链表变为 1->3
myLinkedList.get(1);              // 返回 3
```

提示：

- `0 <= index, val <= 1000`
- 请不要使用内置的 LinkedList 库。
- 调用 `get`、`addAtHead`、`addAtTail`、`addAtIndex` 和 `deleteAtIndex` 的次数不超过 `2000` 。

面试中遇到过这道题？ 1/5

是 否

Go 智能模式

```go
type Node struct {
    Val  int
    Pre  *Node
    Next *Node
}

type MyLinkedList struct {
    Size      int
    DummyHead *Node
    DummyTail *Node
}

func Constructor() MyLinkedList {
    dummyHead := &Node{-1, nil, nil}
    dummyTail := &Node{-1, dummyHead,nil}
    dummyHead.Next = dummyTail //指向尾结点

    return MyLinkedList{0, dummyHead, dummyTail}
}

func (this *MyLinkedList) Get(index int) int {
    if index < 0 || index > this.Size {
        return -1
    }
    cur := this.DummyHead
    for i := 0; i <= index; i++ {
        cur = cur.Next
    }
    return cur.Val
}

func (this *MyLinkedList) AddAtHead(val int) {
    cur := this.DummyHead
    temp := &Node{Val: val}
    temp.Next = cur.Next
    temp.Pre = cur
    cur.Next.Pre = temp
    cur.Next = temp
    this.Size++
}
```

```go
func (this *MyLinkedList) AddAtTail(val int) {
    cur := this.DummyTail
    temp := &Node{val, nil, nil}
    temp.Next = cur
    temp.Pre = cur.Pre
    cur.Pre.Next = temp
    cur.Pre = temp
    this.Size++
}

func (this *MyLinkedList) AddAtIndex(index int, val int) {
    if index < 0 || index > this.Size {
        return
    }
    temp := &Node{Val: val}
    cur := this.DummyHead
    for i := 0; i < index; i++ {
        cur = cur.Next
    }
    temp.Next = cur.Next
    temp.Pre = cur
    cur.Next.Pre = temp
    cur.Next = temp
    this.Size++
}

func (this *MyLinkedList) DeleteAtIndex(index int) {
    if index < 0 || index >= this.Size {
        return
    }
    cur := this.DummyHead
    for i := 0; i < index; i++ {
        cur = cur.Next
    }
    cur.Next.Next.Pre = cur
    cur.Next = cur.Next.Next

    this.Size--
}
```

翻转链表

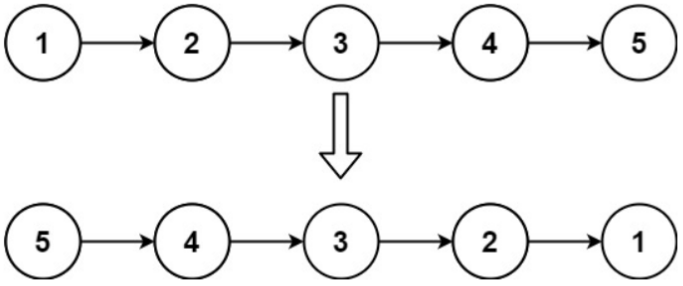
206. 反转链表

已解答

简单 相关标签 相关企业

给你单链表的头节点 `head` ，请你反转链表，并返回反转后的链表。

示例 1：



输入： head = [1,2,3,4,5]
输出： [5,4,3,2,1]

Go 智能模式

```go
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func reverseList(head *ListNode) *ListNode {
    cur:=head
    // pre:=&ListNode{}
    var pre *ListNode

    for cur!=nil{
        temp:=cur.Next
        cur.Next=pre
        //移动pre、cur到下一位
        pre=cur
        cur=temp
    }
    return pre
}
```

两两交换链表中的节点

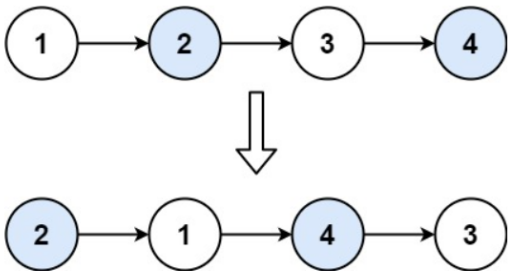
24. 两两交换链表中的节点

已解答

中等 相关标签 相关企业

给你一个链表，两两交换其中相邻的节点，并返回交换后链表的头节点。你必须在不修改节点内部的值的情况下完成本题（即，只能进行节点交换）。

示例 1：



输入： head = [1,2,3,4]
输出： [2,1,4,3]

Go 智能模式

```go
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func swapPairs(head *ListNode) *ListNode {
    dummyHead := &ListNode{Next: head}
    cur:=dummyHead
    for cur.Next != nil && cur.Next.Next != nil {
        temp := cur.Next
        temp2 := cur.Next.Next.Next

        cur.Next = cur.Next.Next //->2
        cur.Next.Next = temp     //2->1
        temp.Next = temp2        //1->3
        cur=cur.Next.Next
    }
    return dummyHead.Next
}
```

删除链表的倒数第N个节点

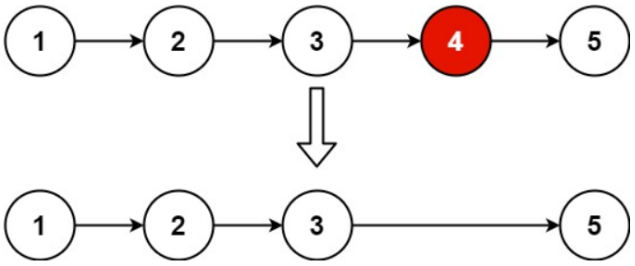
19. 删除链表的倒数第 N 个结点

已解答

中等 相关标签 相关企业 提示

给你一个链表，删除链表的倒数第 `n` 个结点，并且返回链表的头结点。

示例 1：



输入： head = [1,2,3,4,5], n = 2
输出： [1,2,3,5]

Go 智能模式

```go
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func removeNthFromEnd(head *ListNode, n int) *ListNode {
    dummyHead := &ListNode{Next: head}
    slow, fast := dummyHead, dummyHead
    for i := 0; i < n; i++ {
        fast = fast.Next //fast移动n
    }
    for fast.Next != nil {
        //fast、slow同步移动直到fast为nil
        fast = fast.Next
        slow = slow.Next
    }
    slow.Next = slow.Next.Next
    return dummyHead.Next
}
```

链表相交

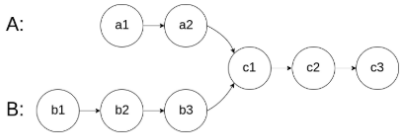
面试题 02.07. 链表相交

已解答

简单 相关标签 相关企业 提示

给你两个单链表的头节点 `headA` 和 `headB` ，请你找出并返回两个单链表相交的起始节点。如果两个链表没有交点，返回 `null` 。

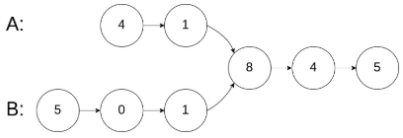
图示两个链表在节点 `c1` 开始相交：



题目数据 **保证** 整个链式结构中不存在环。

注意，函数返回结果后，链表必须 **保持其原始结构** 。

示例 1：



> **输入：** intersectVal = 8, listA = [4,1,8,4,5], listB = [5,0,1,8,4,5], skipA = 2, skipB = 3
> **输出：** Intersected at '8'

```go
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func getIntersectionNode(headA, headB *ListNode) *ListNode {
    curA, curB := headA, headB
    lenA, lenB := 0, 0
    for curA != nil {
        lenA++
        curA = curA.Next
    }
    for curB != nil {
        lenB++
        curB = curB.Next
    }
    pA, pB := headA, headB
    //尾节点一致，移动到长度一致的地方
    if lenA > lenB {
        for i := 0; i < lenA-lenB; i++ {
            pA = pA.Next
        }
    } else {
        for i := 0; i < lenB-lenA; i++ {
            pB = pB.Next
        }
    }
    //比较节点
    for pA != pB {
        pA = pA.Next
        pB = pB.Next
    }
    return pB
}
```

环形链表

142. 环形链表 II

已解答

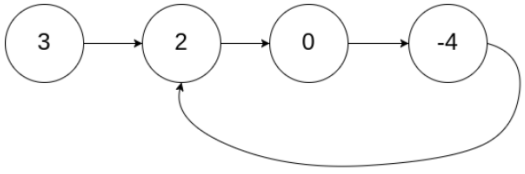
中等 相关标签 相关企业

给定一个链表的头节点 `head` ，返回链表开始入环的第一个节点。 *如果链表无环，则返回* `null`。

如果链表中有某个节点，可以通过连续跟踪 `next` 指针再次到达，则链表中存在环。 为了表示给定链表中的环，评测系统内部使用整数 `pos` 来表示链表尾连接到链表中的位置（**索引从 0 开始**）。如果 `pos` 是 `-1`，则在该链表中没有环。**注意：** `pos` **不作为参数进行传递**，仅仅是为了标识链表的实际情况。

不允许修改 链表。

示例 1：



> **输入：** head = [3,2,0,-4], pos = 1
> **输出：** 返回索引为 1 的链表节点
> **解释：** 链表中有一个环，其尾部连接到第二个节点。

```go
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func detectCycle(head *ListNode) *ListNode {
    fast, slow := head, head
    for fast != nil && fast.Next != nil { //设定快慢指针，同时快指针比满指针快一步
        fast = fast.Next.Next
        slow = slow.Next
        if fast == slow { //快慢指针相等，找到环入口
            temp1 := fast
            for temp1 != head {
                temp1 = temp1.Next
                head = head.Next
            }
            return temp1
        }
    }
    return nil
}
```

哈希

有效的字母异位词-数组/切片

242. 有效的字母异位词

已解答

简单 相关标签 相关企业

给定两个字符串 `s` 和 `t` ，编写一个函数来判断 `t` 是否是 `s` 的 字母异位词。

示例 1:

> **输入:** s = "anagram", t = "nagaram"
> **输出:** true

示例 2:

> **输入:** s = "rat", t = "car"
> **输出:** false

Go 智能模式

```go
func isAnagram(s string, t string) bool {
    var num [26]int
    for _, v := range s {
        num[v-rune('a')]++
    }
    for _, v := range t {
        num[v-rune('a')]--
    }
    if num == [26]int{}{
        return true
    }

    return false
}
```

383. 赎金信

已解答

简单 相关标签 相关企业

给你两个字符串：`ransomNote` 和 `magazine` ，判断 `ransomNote` 能不能由 `magazine` 里面的字符构成。

如果可以，返回 `true` ；否则返回 `false` 。

`magazine` 中的每个字符只能在 `ransomNote` 中使用一次。

示例 1:

> **输入:** ransomNote = "a", magazine = "b"
> **输出:** false

示例 2:

> **输入:** ransomNote = "aa", magazine = "ab"
> **输出:** false

示例 3:

> **输入:** ransomNote = "aa", magazine = "aab"
> **输出:** true

Go 智能模式

```go
func canConstruct(ransomNote string, magazine string) bool {
    var record [26]int
    for _, v := range magazine {
        record[v-'a']++
    }
    for _, v := range ransomNote {
        record[v-'a']--
        if record[v-'a'] < 0 {
            return false
        }
    }

    return true
}
```

49. 字母异位词分组

已解答

中等 相关标签 相关企业

给你一个字符串数组，请你将 **字母异位词** 组合在一起。可以按任意顺序返回结果列表。

字母异位词 是由重新排列源单词的所有字母得到的一个新单词。

示例 1:

> **输入:** strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
> **输出:** [["bat"],["nat","tan"],["ate","eat","tea"]]

示例 2:

> **输入:** strs = [""]
> **输出:** [[""]]

示例 3:

> **输入:** strs = ["a"]
> **输出:** [["a"]]

Go 智能模式

```go
func groupAnagrams(strs []string) [][]string {
    m := make(map[[26]int][]string)
    for _, str := range strs {
        var num [26]int
        for _, v := range str {
            num[v-'a']++
        }
        m[num] = append(m[num], str)
    }

    ans := make([][]string, 0, len(strs))
    for _, v := range m {
        ans = append(ans, v)
    }
    return ans
}
```

438. 找到字符串中所有字母异位词

已解答

中等 相关标签 相关企业

给定两个字符串 `s` 和 `p`，找到 `s` 中所有 `p` 的 **异位词** 的子串，返回这些子串的起始索引。不考虑答案输出的顺序。

示例 1:

> **输入:** s = "cbaebabacd", p = "abc"
> **输出:** [0,6]
> **解释:**
> 起始索引等于 0 的子串是 "cba", 它是 "abc" 的异位词。
> 起始索引等于 6 的子串是 "bac", 它是 "abc" 的异位词。

示例 2:

> **输入:** s = "abab", p = "ab"
> **输出:** [0,1,2]
> **解释:**
> 起始索引等于 0 的子串是 "ab", 它是 "ab" 的异位词。
> 起始索引等于 1 的子串是 "ba", 它是 "ab" 的异位词。
> 起始索引等于 2 的子串是 "ab", 它是 "ab" 的异位词。

Go 智能模式

```go
func findAnagrams(s string, p string) []int {
    var num [26]int
    for _, v := range p {
        num[v-'a']++
    }
    //滑动窗口
    left := 0
    var result []int
    for right, v := range s {
        num[v-'a']--
        for num[v-'a'] < 0 {
            num[s[left]-'a']++
            left++
        }
        if right-left+1 == len(p) {
            result = append(result, left)
        }
    }
    return result
}
```

两个数组的交集-map

349. 两个数组的交集

已解答

简单 相关标签 相关企业

给定两个数组 `nums1` 和 `nums2` ，返回 *它们的* *交集*。输出结果中的每个元素一定是 **唯一** 的。我们可以 **不考虑输出结果的顺序** 。

示例 1:

> **输入:** nums1 = [1,2,2,1], nums2 = [2,2]
> **输出:** [2]

示例 2:

> **输入:** nums1 = [4,9,5], nums2 = [9,4,9,8,4]
> **输出:** [9,4]
> **解释:** [4,9] 也是可通过的

Go 智能模式

```go
func intersection(nums1 []int, nums2 []int) []int {
    record := make(map[int]int)
    var result []int
    for _, v := range nums1 {
        record[v]=1
    }

    for _, v := range nums2 {
        if record[v] == 1 {
            result = append(result, v)
            record[v]++
        }
    }

    return result
}
```

350. 两个数组的交集 II

已解答

简单 相关标签 相关企业

给你两个整数数组 `nums1` 和 `nums2` ，请你以数组形式返回两数组的交集。返回结果中每个元素出现的次数，应与元素在两个数组中都出现的次数一致（如果出现次数不一致，则考虑取较小值）。可以不考虑输出结果的顺序。

示例 1:

> **输入:** nums1 = [1,2,2,1], nums2 = [2,2]
> **输出:** [2,2]

示例 2:

> **输入:** nums1 = [4,9,5], nums2 = [9,4,9,8,4]
> **输出:** [4,9]

Go 智能模式

```go
func intersect(nums1 []int, nums2 []int) []int {
    record := make(map[int]int)
    var result []int
    if len(nums1) < len(nums2) {
        nums1, nums2 = nums2, nums1
    }
    for _, v := range nums1 {
        record[v]++
    }

    for _, v := range nums2 {
        if record[v] > 0 {
            result = append(result, v)
            record[v]--
        }
    }
    return result
}
```

快乐数

202. 快乐数

已解答

简单 相关标签 相关企业

编写一个算法来判断一个数 `n` 是不是快乐数。

「快乐数」 定义为：

- 对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和。
- 然后重复这个过程直到这个数变为 1，也可能是 **无限循环** 但始终变不到 1。
- 如果这个过程 **结果为** 1，那么这个数就是快乐数。

如果 `n` 是 *快乐数* 就返回 `true` ；不是，则返回 `false` 。

示例 1：

> **输入：** n = 19
> **输出：** true
> **解释：**
> $1^2 + 9^2 = 82$
> $8^2 + 2^2 = 68$
> $6^2 + 8^2 = 100$
> $1^2 + 0^2 + 0^2 = 1$

Go 智能模式

```go
func isHappy(n int) bool {
    numMap := make(map[int]bool)
    for{
        sum := 0
        for n != 0 {
            sum += (n % 10) * (n % 10)
            n = n / 10
        }
        if sum == 1 {
            return true
        }
        if numMap[sum] {
            return false
        }
        numMap[sum] = true
        n = sum
    }
}
```

两数之和

1. 两数之和

已解答

简单 相关标签 相关企业 提示

给定一个整数数组 `nums` 和一个整数目标值 `target`，请你在该数组中找出 **和为目标值** *target* 的那 **两个** 整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案，并且你不能使用两次相同的元素。

你可以按任意顺序返回答案。

示例 1：

> **输入：** nums = [2,7,11,15], target = 9
> **输出：** [0,1]
> **解释：** 因为 nums[0] + nums[1] == 9 ，返回 [0, 1] 。

示例 2：

> **输入：** nums = [3,2,4], target = 6
> **输出：** [1,2]

Go 智能模式

```go
func twoSum(nums []int, target int) []int {
    numMap := make(map[int]int)
    for index, num := range nums {
        value := target - num
        valueIndex, ok := numMap[value]
        if ok {
            return []int{index, valueIndex}
        } else {
            numMap[num] = index
        }
    }
    return []int{}
}
```

四数相加2

454. 四数相加 II

已解答

中等 相关标签 相关企业

给你四个整数数组 `nums1`、`nums2`、`nums3` 和 `nums4` ，数组长度都是 `n` ，请你计算有多少个元组 `(i, j, k, l)` 能满足：

- `0 <= i, j, k, l < n`
- `nums1[i] + nums2[j] + nums3[k] + nums4[l] == 0`

示例 1：

> **输入：** nums1 = [1,2], nums2 = [-2,-1], nums3 = [-1,2], nums4 = [0,2]
> **输出：** 2
> **解释：**
> 两个元组如下：
> 1. (0, 0, 0, 1) -> nums1[0] + nums2[0] + nums3[0] + nums4[1] = 1 + (-2) + (-1) + 2 = 0
> 2. (1, 1, 0, 0) -> nums1[1] + nums2[1] + nums3[0] + nums4[0] = 2 + (-1) + (-1) + 0 = 0

Go 智能模式

```go
func fourSumCount(nums1 []int, nums2 []int, nums3 []int, nums4 []int) int {
    m := make(map[int]int)
    count:=0
    for _, a := range nums1 {
        for _, b := range nums2 {
            m[a+b]++
        }
    }
    for _, c := range nums3 {
        for _, d := range nums4 {
            value, ok := m[-c-d]
            if ok {
                count+=value
            }
        }
    }
    return count
}
```

三数之和

15. 三数之和

已解答

中等 | 相关标签 | 相关企业 | 提示 | 文A

给你一个整数数组 `nums` ，判断是否存在三元组 `[nums[i], nums[j], nums[k]]` 满足 `i != j`、`i != k` 且 `j != k` ，同时还满足 `nums[i] + nums[j] + nums[k] == 0` 。请你返回所有和为 `0` 且不重复的三元组。

注意：答案中不可以包含重复的三元组。

示例 1：

> **输入：** nums = [-1,0,1,2,-1,-4]
> **输出：** [[-1,-1,2],[-1,0,1]]
> **解释：**
> 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 。
> 不同的三元组是 [-1,0,1] 和 [-1,-1,2] 。
> 注意，输出的顺序和三元组的顺序并不重要。

示例 2：

> **输入：** nums = [0,1,1]
> **输出：** []
> **解释：** 唯一可能的三元组和不为 0 。

Go 智能模式

```go
for i := 0; i < len(nums)-2; i++ {
    // 排序之后如果第一个元素已经大于零，那么无论如何组合都不可能凑成三元组，直接返回结果就可以了
    n1 := nums[i]
    if n1 > 0 {
        break
    }
    // 去重a
    if i > 0 && n1 == nums[i-1] {
        continue
    }
    l, r := i+1, len(nums)-1
    for l < r {
        n2, n3 := nums[l], nums[r]
        if n1+n2+n3 == 0 {
            res = append(res, []int{n1, n2, n3})
            // 去重逻辑应该放在找到一个三元组之后，对b 和 c去重
            for l < r && nums[l] == n2 {
                l++
            }
            for l < r && nums[r] == n3 {
                r--
            }
        } else if n1+n2+n3 < 0 {
            l++
        } else {
            r--
        }
    }
}
return res
}
```

四数之和

18. 四数之和

已解答

中等 | 相关标签 | 相关企业 | 文A

给你一个由 `n` 个整数组成的数组 `nums` ，和一个目标值 `target` 。请你找出并返回满足下述全部条件且**不重复**的四元组 `[nums[a], nums[b], nums[c], nums[d]]` （若两个四元组元素一一对应，则认为两个四元组重复）：

- `0 <= a, b, c, d < n`
- `a`、`b`、`c` 和 `d` **互不相同**
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

你可以按 **任意顺序** 返回答案 。

示例 1：

> **输入：** nums = [1,0,-1,0,-2,2], target = 0
> **输出：** [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

示例 2：

> **输入：** nums = [2,2,2,2,2], target = 8
> **输出：** [[2,2,2,2]]

提示：

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

Go 智能模式

```go
func fourSum(nums []int, target int) [][]int {
    sort.Ints(nums)
    result := [][]int{}
    for i := 0; i < len(nums)-3; i++ {
        if i > 0 && nums[i] == nums[i-1] {
            continue
        }
        for j := i + 1; j < len(nums)-2; j++ {
            if j > i+1 && nums[j] == nums[j-1] {
                continue
            }
            left := j + 1
            right := len(nums) - 1
            for right > left {
                if nums[i]+nums[j]+nums[left]+nums[right] > target {
                    right--
                } else if nums[i]+nums[j]+nums[left]+nums[right] < target {
                    left++
                } else {
                    result = append(result, []int{nums[i], nums[j], nums[left], nums[right]})
                    for right > left && nums[right] == nums[right-1] {
                        right--
                    }
                    for right > left && nums[left] == nums[left+1] {
                        left++
                    }
                    right--
                    left++
                }
            }
        }
    }
    return result
}
```

字符串

反转字符串-双指针

344. 反转字符串

已解答

简单 | 相关标签 | 相关企业 | 提示 | 文A

编写一个函数，其作用是将输入的字符串反转过来。输入字符串以字符数组 `s` 的形式给出。

不要给另外的数组分配额外的空间，你必须**原地修改输入数组**、使用 O(1) 的额外空间解决这一问题。

示例 1：

> **输入：** s = ["h","e","l","l","o"]
> **输出：** ["o","l","l","e","h"]

示例 2：

> **输入：** s = ["H","a","n","n","a","h"]
> **输出：** ["h","a","n","n","a","H"]

Go 智能模式

```go
func reverseString(s []byte) {
    if s == nil {
        return
    }
    slow, fast := 0, len(s)-1
    for slow < fast {
        // temp := s[slow]
        // s[slow] = s[fast]
        // s[fast] = temp
        s[slow],s[fast]=s[fast],s[slow]
        slow++
        fast--
    }
}
```

反转字符串2

541. 反转字符串 II

已解答

简单 相关标签 相关企业

给定一个字符串 `s` 和一个整数 `k`，从字符串开头算起，每计数至 `2k` 个字符，就反转这 `2k` 字符中的前 `k` 个字符。

- 如果剩余字符少于 `k` 个，则将剩余字符全部反转。
- 如果剩余字符小于 `2k` 但大于或等于 `k` 个，则反转前 `k` 个字符，其余字符保持原样。

示例 1:

> **输入:** s = "abcdefg", k = 2
> **输出:** "bacdfeg"

示例 2:

> **输入:** s = "abcd", k = 2
> **输出:** "bacd"

Go 智能模式

```go
func reverseStr(s string, k int) string {
    if len(s) == 0 {
        return ""
    }
    str := []byte(s)
    for i := 0; i < len(str); i = i + 2*k {
        if i+k < len(s) {
            reverse(str[i : i+k])
        } else {
            reverse(str[i : len(str)])
        }
    }
    return string(str)
}

func reverse(s []byte) {
    slow, fast := 0, len(s)-1
    for slow < fast {
        s[slow], s[fast] = s[fast], s[slow]
        slow++
        fast--
    }
}
```

替换字符串

```go
func main() {
    /.../

    /.../

    /.../

    scanner := bufio.NewScanner(os.Stdin)
    for scanner.Scan() {
        ss := scanner.Bytes()
        result := make([]string, 0)
        for _, value := range ss {
            if value >= '0' && value <= '9' {
                result = append(result, elems...: "number")
            } else {
                result = append(result, string(value))
            }
        }
        fmt.Println(strings.Join(result, sep: ""))
    }
}
```

实现str()-kmp

28. 找出字符串中第一个匹配项的下标

已解答

简单 相关标签 相关企业

给你两个字符串 `haystack` 和 `needle` ，请你在 `haystack` 字符串中找出 `needle` 字符串的第一个匹配项的下标（下标从 0 开始）。如果 `needle` 不是 `haystack` 的一部分，则返回 `-1` 。

示例 1:

> **输入:** haystack = "sadbutsad", needle = "sad"
> **输出:** 0
> **解释:** "sad" 在下标 0 和 6 处匹配。
> 第一个匹配项的下标是 0 ，所以返回 0 。

示例 2:

> **输入:** haystack = "leetcode", needle = "leeto"
> **输出:** -1
> **解释:** "leeto" 没有在 "leetcode" 中出现，所以返回 -1 。

提示:

- `1 <= haystack.length, needle.length <= 10^4`
- `haystack` 和 `needle` 仅由小写英文字符组成

面试中遇到过这道题? 1/5

是 否

通过次数 1.2M | 提交次数 2.7M | 通过率 44.4%

Go 智能模式

```go
func strStr(haystack string, needle string) int {
    pattern := []byte(needle)
    str := []byte(haystack)
    lps := getNext(pattern) //模式串的前缀表
    i, j := 0, 0            //i是主串要找的，j是模式串
    for i < len(str) && j < len(pattern) {
        if str[i] == pattern[j] {
            i++
            j++
        } else {
            if j != 0 {
                j = lps[j-1]
            } else {
                i++
            }
        }
    }
    if j == len(pattern) {
        return i - len(pattern)
    }
    return -1
}
func getNext(pattern []byte) []int {
    lps := make([]int, len(pattern))
    index := 0
    for i := 1; i < len(pattern); {
        if pattern[i] == pattern[index] {
            lps[i] = index + 1
            index++
            i++
        } else { //不相等，如果下标不为0，index是前一位下标的前缀表对应的值
            if index != 0 {
                index = lps[index-1]
            } else {
                lps[i] = 0
                i++
            }
        }
    }
    return lps
}
```

重复的子字符串

459. 重复的子字符串

已解答

简单 相关标签 相关企业

给定一个非空的字符串 `s` ，检查是否可以通过由它的一个子串重复多次构成。

示例 1:

> **输入:** s = "abab"
> **输出:** true
> **解释:** 可由子串 "ab" 重复两次构成。

示例 2:

> **输入:** s = "aba"
> **输出:** false

示例 3:

> **输入:** s = "abcabcabcabc"
> **输出:** true
> **解释:** 可由子串 "abc" 重复四次构成。 (或子串 "abcabc" 重复两次构成。)

Go 智能模式

```go
func repeatedSubstringPattern(s string) bool {
    ss := []byte(s)
    if len(ss) == 1 {
        return false
    }
    n := len(ss)
    next := make([]int, len(ss))
    i := 0
    for j := 1; j < len(ss); {
        if ss[i] == ss[j] {
            next[j] = i + 1
            j++
            i++
        } else {
            if i != 0 {
                i = next[i-1]
            } else {
                next[j] = 0
                j++
            }
        }
    }
    if next[n-1] != 0 && n%(n-next[n-1]) == 0 {
        return true
    }
    return false
}
```