链滴

ARTS 006

作者：lucianolixin

原文链接：https://ld246.com/article/1569253511942

来源网站：链滴

ARTS是由左耳朵耗子陈皓在极客时间专栏《左耳听风》中发起的一个每周学习打卡计划。

Algorithm：至少做一个 LeetCode 的算法题。主要为了编程训练和学习。

Review ： 阅读并点评至少一篇英文技术文章。主要为了学习英文，如果你英文不行，很难成为技术手。

Tip：学习至少一个技术技巧。主要是为了总结和归纳你日常工作中所遇到的知识点。

Share：分享一篇有观点和思考的技术文章。主要为了输出你的影响力，能够输出你的价值观。

Algorithm

搜索旋转序列

搜索一个给定的目标值，如果数组中存在这个目标值，则返回它的索引，否则返回 -1 。

你可以假设数组中不存在重复的元素。

你的算法时间复杂度必须是 O(log n) 级别。

示例 1:

输入: nums = [4,5,6,7,0,1,2], target = 0
输出: 4
示例 2:

输入: nums = [4,5,6,7,0,1,2], target = 3
输出: -1

解法1

```
func search(nums []int, target int) int {
    return findNum(nums,0,len(nums)-1,target)
}

func findNum(nums []int, low int, high int, target int) int {

    mid := low + ((high - low) >> 2)
    if low == high {
        if nums[low] == target {
            return low
        } else {
            return -1
        }
    }
    if nums[low] < nums[mid] {
        r := binarySearch(nums, low, mid, target)
        if r == -1 {
            return findNum(nums, mid+1, high, target)
        } else {
            return r
        }
    } else {
        r := binarySearch(nums, mid+1, high, target)
        if r == -1 {
            return findNum(nums, low, mid, target)
        } else {
            return r
        }
    }
    return -1
}

func binarySearch(n []int, low int, high int, target int) int {
    mid := low + ((high - low) >> 2)
    for low <= high {
        if n[mid] > target {
            high = mid - 1
        } else if n[mid] < target {
            low = mid + 1
        } else {
            return mid
        }
    }
    return -1
}
```

解法2

```
func search(nums []int, target int) int {
    n := len(nums)
    l, r := 0, n
```

```
    var m int
    for l < r {
        m = l + (r-l) >> 1
        if nums[m] == target {
            return m
        } else if nums[l] == target {
            return l
        } else if nums[r-1] == target {
            return r-1
        }
        if nums[l] < nums[m] {
            if target > nums[m] || nums[l] > target {
                l = m+1
            } else {
                r = m
            }
        } else {
            if nums[m] > target || target > nums[r-1] {
                r = m
            } else {
                l = m+1
            }
        }
    }
    return -1
}
```

Review

在微服务架构中的服务发现

本篇文章来自于Nginx官方博客，这是介绍微服务架构一系列文章中的一篇，本文先由Resful API、thrift API的调用，引出服务发现的必要性，在过去我们用一个静态的配置文件保存需要调用服务的地址在现代微服务架构中其暴露的问题越来越明显，微服务的拆分，意味着更多的服务的实例，随之而来是服务的管理，包括自动扩容、下线、升级等维护工作。所以我们需要一个动态的发现服务变更的机，这就是服务发现。服务发现包括3种模式。

- 客户端发现模式
- 服务端发现模式
- 第三方注册模式

Tip

Linux 常用命令行快捷键

- Ctrl + w 删除当前光标之前的字符
- Ctrl + u 删除当前光标至初始的字符
- Ctrl + b 向前移动光标 back
- Ctrl + f 想后移动光标 forward

- Ctrl + e 移动到行位
- Ctrl + a 移动到行首

Share

近期工作不是很忙，开发工作出于尾声，也恰好总结一下这段时间的工作。从今年一月份到现在为止直持续学习，并且新工作中面临的一些挑战，也让我有肉眼可见的成长。在这段时间内，成功的从PH转向Golang的开发，让自己的职业生涯柳暗花明，在对接银行支付的项目中对业务的理解更加深刻并且熟悉了RabbitMQ消息队列，开始学习并且在日常开发中使用vim开发。自学的有mysql、消息列、计算机组成原理、通信协议、架构的基础知识。

同时也为后面的几个月做一个简单的计划和学习方向

- 持续ARTS，多总结
- 学习英语，多阅读英文资料
- 算法与数据结构
- linux性能优化
- 网路编程
- 微服务架构和系统设计