## 模拟面试题1

### 1. 调整数组顺序使奇数位于偶数前面

输入一个int数组，返回一个数组，其中奇数都在左边，偶数都在右边。附加条件 不使用额外的空间

解答：

这道题是的leetcode地址地址：剑指offer21 https://leetcode-cn.com/problems/diao-zheng-shu-zu-shun-xu-shi-qi-shu-wei-yu-ou-shu-qian-mian-lcof/

这道题我首先想到的是排序，但是超时了

然后想到了双指针

双指针也可以用两种，一种是首尾指针

class Solution:
 def exchange(self, nums: List[int]) -> List[int]:
 i, j = 0, len(nums) - 1
 while i < j:
 if nums[i] % 2 == 1:
 i+=1
 continue

 if nums[j] % 2 == 0:
 j -=1
 continue
 nums[i], nums[j] = nums[j], nums[i]
 return nums

一种是快慢指针

class Solution:
 def exchange(self, nums: List[int]) -> List[int]:
 i, j = 0, 0
 while j < len(nums):
 if nums[j] % 2 == 1:
 nums[i], nums[j] = nums[j], nums[i]
 i += 1
 j += 1
 return nums

#### 建议

1. 熟悉各类排序：冒泡，插入，归并，快速，桶
2. 熟悉双指针法，这在很多数组题里能派上用场

### 2. 数据库索引

todo

### 3. 数据库分库分表

数据库如何分库分表？

### 4. 找到一个文件里出现次数最多的的数字，文件大小远大于内存容量

### 5. 两数之和

给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。

这道题的leetcode地址是：https://leetcode-cn.com/problems/two-sum/

非常经典的一道题，在我的面试经历中，也出现过。

在面试过程中，尽可能逐渐深入，把解法由浅入深依次讲出来，并且比较出它们的优劣

首先是暴力法，双层循环，复杂度O(n^2)

class Solution:
 def twoSum(self, nums: List[int], target: int) -> List[int]:
 for i in range(len(nums)-1):
 for j in range(i+1, len(nums)):
 if nums[i] + nums[j] == target:
 return [i, j]

用字典存放之前的结果，空间换时间，复杂度O(n)

class Solution:
 def twoSum(self, nums: List[int], target: int) -> List[int]:
 data\_dict = dict()
 for i, element in enumerate(nums):
 if target - element in data\_dict:
 return [data\_dict[target-element], i]
 data\_dict[element] = i

#### 建议

1. 算法题要经常刷，不要只刷一遍，过遍数，五毒神掌
2. 同一道题，可以多看看别人的解法，并且知道各个解法的优劣

### 6. MySQL两种存储引擎的差异

简单介绍区别：

* MyISAM是非事务安全的，而InnoDB是事务安全的
* MyISAM锁的粒度是表级的，而InnoDB支持行级锁
* MyISAM支持全文类型索引，而InnoDB不支持全文索引
* MyISAM相对简单，效率上要优于InnoDB，小型应用可以考虑使用MyISAM
* MyISAM表保存成文件形式，跨平台使用更加方便

1、MyISAM管理非事务表，提供高速存储和检索以及全文搜索能力，如果再应用中执行大量select操作，应该选择MyISAM 2、InnoDB用于事务处理，具有ACID事务支持等特性，如果在应用中执行大量insert和update操作，应该选择InnoDB

参考资料：https://juejin.cn/post/6903101301429796871

### 7. where a>1 and b>1；where a = 1；where b = 2；如何为这种条件语句建立索引

table(a,b) table(b)

### 8. LRU的实现

leetcode原题地址：https://leetcode-cn.com/problems/lru-cache/

python可以用collections.OrderedDict轻松实现，但肯定不符合面试官要求。

正解是用哈希表+双向链表

* 双向链表按照被使用的顺序存储了这些键值对，靠近头部的键值对是最近使用的，而靠近尾部的键值对是最久未使用的。
* 哈希表即为普通的哈希映射（HashMap），通过缓存数据的键映射到其在双向链表中的位置。

class DLinkedNode:
 def \_\_init\_\_(self, key=0, value=0):
 self.key = key
 self.value = value
 self.prev = None
 self.next = None

class LRUCache1:

 def \_\_init\_\_(self, capacity: int):
 self.cache = dict()
 # 使用伪头部和伪尾部节点
 self.head = DLinkedNode()
 self.tail = DLinkedNode()
 self.head.next = self.tail
 self.tail.prev = self.head
 self.capacity = capacity
 self.size = 0

 def get(self, key: int) -> int:
 if key not in self.cache:
 return -1
 # 如果 key 存在，先通过哈希表定位，再移到头部
 node = self.cache[key]
 self.moveToHead(node)
 return node.value

 def put(self, key: int, value: int) -> None:
 if key not in self.cache:
 # 如果 key 不存在，创建一个新的节点
 node = DLinkedNode(key, value)
 # 添加进哈希表
 self.cache[key] = node
 # 添加至双向链表的头部
 self.addToHead(node)
 self.size += 1
 if self.size > self.capacity:
 # 如果超出容量，删除双向链表的尾部节点
 removed = self.removeTail()
 # 删除哈希表中对应的项
 self.cache.pop(removed.key)
 self.size -= 1
 else:
 # 如果 key 存在，先通过哈希表定位，再修改 value，并移到头部
 node = self.cache[key]
 node.value = value
 self.moveToHead(node)

 def addToHead(self, node):
 node.prev = self.head
 node.next = self.head.next
 self.head.next.prev = node
 self.head.next = node

 def removeNode(self, node):
 node.prev.next = node.next
 node.next.prev = node.prev

 def moveToHead(self, node):
 self.removeNode(node)
 self.addToHead(node)

 def removeTail(self):
 node = self.tail.prev
 self.removeNode(node)
 return node

### 9. 介绍一个项目中遇到的问题

我好像没遇到特别有挑战性的难题，业务开发居多，很难说出亮点。

我整理了白海飞老师《面试现场》的“经历没有亮点可讲？你需要做份详历”这一节的思路，希望对你能有所启发。

如何从过去的经历中发现亮点：

对于面试官而言，亮点是你打动他的能力素质，以及有价值的成果。能力素质包括经验、技能、潜力和动机，成果包括做出的产品、服务，总结的方法、实践，培养的人才和团队，以及各种形式的认可和奖励。

很多人都说难于发现自己经历的亮点。这是个普遍的现象，类似于“知识的诅咒”：你对做过的项目虽然了如指掌，却不容易看到哪些是别人认为了不起的价值，也总结不出当初用了哪些醒目的技能。正所谓“不识庐山真面目，只缘身在此山中”。

下面我们就一起来看看如何跳出这座山，换一下视角，尝试找找那些有价值有亮点的地方。

1. 转换角度找价值。工作的价值，可以从公司、客户、团队和自己的角度去寻找。
2. 量化结果找提高。
3. 复盘过程找创新。
4. 回顾挑战找动机

### 10. 一个100G的文件，内存只有8G，如何给文件排序

运用归并排序的思想

多路归并

先拆成一个个小块，分别排序，然后使用归并排序的思想

参考链接： https://www.cnblogs.com/chengxiao/p/6194356.html http://blog.itpub.net/31561269/viewspace-2564096/ https://blog.csdn.net/sdmxdzb/article/details/104409250