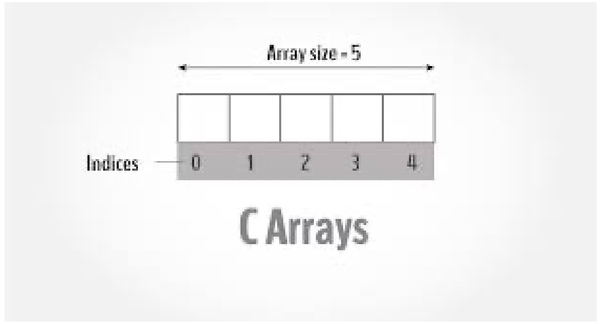
极客时间算法训练营
2022 第七期
领教直播
第一周: 数组 链表 栈 队列

领教：Ellie

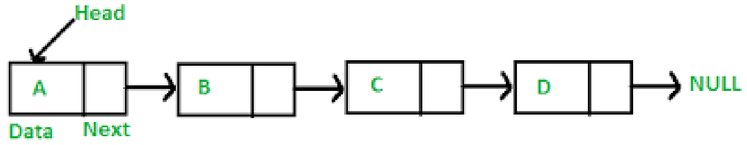
目录

数组 链表 栈 队列

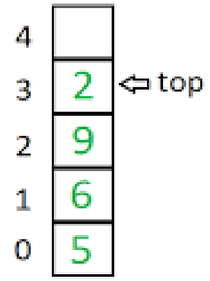
- 数组 array
  - 连续储存空间
  - 一次性定位



- 链表 linked list
  - 离散储存空间
  - 通过next 节点定位下一个list node



- 栈 stack
  - 先进后出 （last in first out）



- 队列 queue
  - 先进先出 （first in first out）

作业1：加一

- 链接： https://leetcode-cn.com/problems/plus-one/
- 描述： 给定一个由 整数 组成的 非空 数组所表示的非负整数，在该数的基础上加一。最高位数字存放在数组的首位， 数组中每个元素只存储单个数字。你可以假设除了整数 0 之外，这个整数不会以零开头。

  输入：digits = [1,2,3] 输出：[1,2,4]

  解释：输入数组表示数字 123。
- 数组想象为数字

  1 + 1 = 2

  9 + 1 = 10

  99 + 1 = 100
- 从最后一个数字开始遍历
  - 最后一位 + 1
  - 剩余的算上进位
- 需要一个变量储存进位值

【1， 2， 9】

carry = 0

9 + 1 = 10

- carry = 1， cur = 0， 【0】

2 + carry = 3

- carry = 0， cur = 3， 【3， 0】

1 + carry = 1

- carry = 0 ， cur = 1， 【1，3，0】

作业2：合并两个有序链表

- 链接：https://leetcode-cn.com/problems/merge-two-sorted-lists/
- 描述： 将两个升序链表合并为一个新的 升序 链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

- 输入：l1 = [1,2,4], l2 = [1,3,4] 输出：[1,1,2,3,4,4]

- 方法：两个指针， 指向两个lists 的头部节点（链表头节点）
  - 比较两个指针指向的值
  - 将更小的节点串起来， 并且移动该指针
  - 持续比较，直到所有的点都遍历结束
  - 需要用一个dummy node 作为串联开始的头部节点

作业3：设计循环双端队列

- 链接：https://leetcode-cn.com/problems/design-circular-deque/
- 你的实现需要支持以下操作：
    - MyCircularDeque(k)：构造函数,双端队列的大小为k。
    - insertFront()：将一个元素添加到双端队列头部。 如果操作成功返回 true。
    - insertLast()：将一个元素添加到双端队列尾部。如果操作成功返回 true。
    - deleteFront()：从双端队列头部删除一个元素。 如果操作成功返回 true。
    - deleteLast()：从双端队列尾部删除一个元素。如果操作成功返回 true。
    - getFront()：从双端队列头部获得一个元素。如果双端队列为空，返回 -1。
    - getRear()：获得双端队列的最后一个元素。 如果双端队列为空，返回 -1。
    - isEmpty()：检查双端队列是否为空。
    - isFull()：检查双端队列是否满了。

- 方法： 头尾方便读取， 类似链表，可以实现一个双向链表

作业3：设计循环双端队列

- 双向链表节点的设计

```python
class Node:
    def __init__(self, val):
        self.next = None
        self.pre = None
        self.val = val
```

- 双向链表初始数据结构

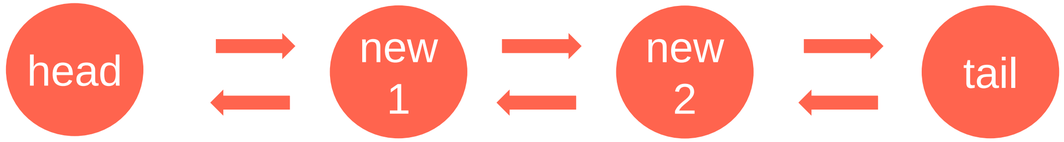
```python
class MyCircularDeque:

    def __init__(self, k: int):
        """
        Initialize your data structure here.
        Set the size of the deque to be k.
        """
        self.capacity = k
        self.size = 0
        self.head = Node(-1)
        self.tail = Node(-1)
        self.head.next = self.tail
        self.tail.pre = self.head
```

作业3：设计循环双端队列

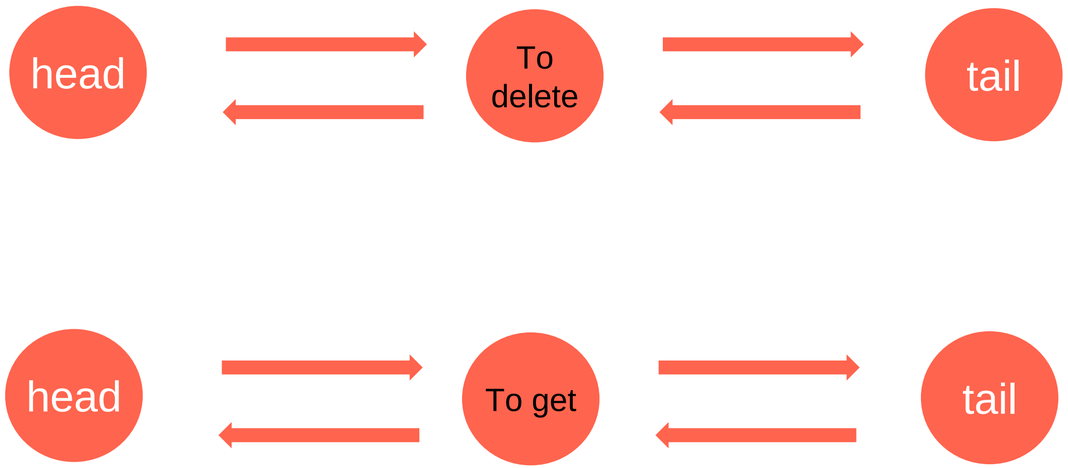
insertFront()：将一个元素添加到双端队列头部。 如果操作成功返回 true。创造一个新的node new1， 插入head 后面

insertLast()：将一个元素添加到双端队列尾部。如果操作成功返回 true。创造一个新的节点插入tail 之前



作业3：设计循环双端队列

- deleteFront()：从双端队列头部删除一个元素。 如果操作成功返回 true：
  - 删除 head 后面的一个node
- deleteLast()：从双端队列尾部删除一个元素。如果操作成功返回 true
  - 删除 tail 前的node
- getFront()：从双端队列头部获得一个元素。如果双端队列为空，返回 -1
  - Head 后面一个node
- getRear()：获得双端队列的最后一个元素。 如果双端队列为空，返回 -1
  - Tail 前一个node



作业3：设计循环双端队列

- isEmpty()：检查双端队列是否为空。
  - 直接计数比较
- isFull()：检查双端队列是否满了：
  - 直接计数比较

```
49  class MyCircularDeque:
50
51      def __init__(self, k: int):
52          """
53          Initialize your data structure here.
54          Set the size of the deque to be k.
55          """
56          self.capacity = k
57          self.size = 0
58          self.head = Node(-1)
59          self.tail = Node(-1)
60          self.head.next = self.tail
61          self.tail.pre = self.head
62
63
```

作业4：最大矩形

- 链接： https://leetcode-cn.com/problems/maximal-rectangle/
- 描述： 给定一个仅包含 0 和 1 、大小为 rows x cols 的二维二进制矩阵，找出只包含 1 的最大矩形，并返回其面积。

输入：matrix =

[["1","0","1","0","0"],

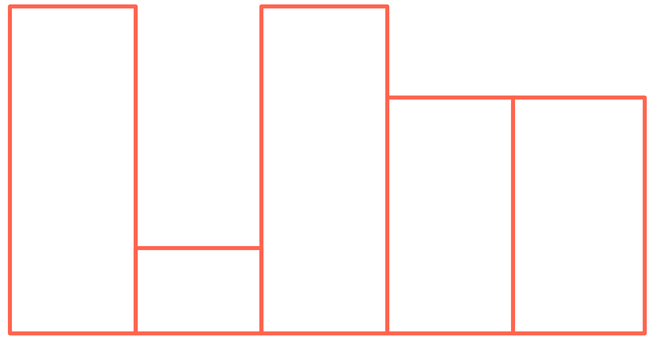
["1","0","1","1","1"],

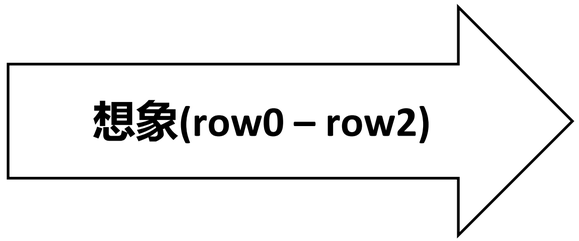
["1","1","1","1","1"],

["1","0","0","1","0"]]

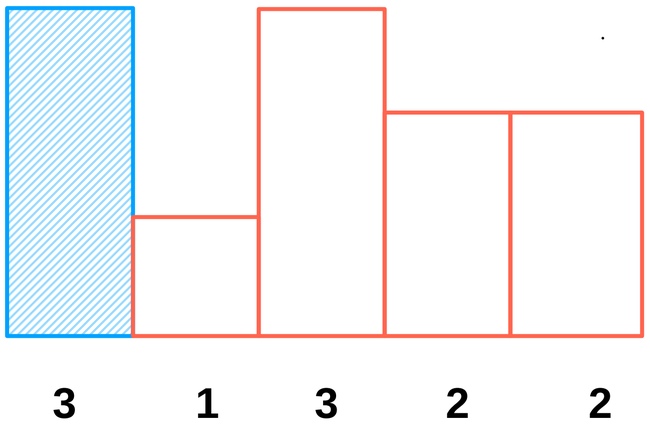
输出：6

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0





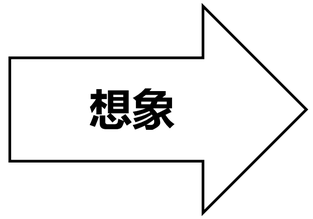
作业4：最大矩形

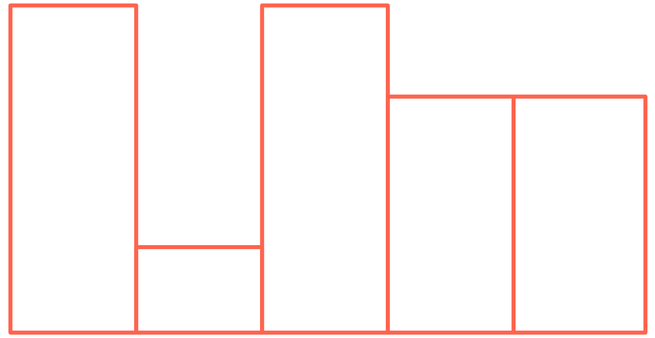


- 找每个柱子的左右边界？时间复杂度很大o（n^2）
- 有没有办法一次遍历全部找到？
  - 有， 需要单调栈
  - 本质是寻找局部最大值 【1 3 2】
  - 如果用一个递增的栈维护，发现下个进入的数值比 stack 顶部的要小， 我们知道，此刻已经找到局部最d 大值， 找到了两个边界，可以开始计算
  - 比如现在栈里面是【3】， 然后1 可能是下一个进入的，反正1 比 3 小，那么说明 3 左边的比它自己小（我们是个单调递增的栈， 栈顶元素最大）。1 又比 3 小，那么，我们可以计算 以 3 为高的长方形，得到局部答案。 同时 pop 3，因为3已经没有利用价值了
  - 最后把1 塞进stack 里面

作业4：最大矩形

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0





- 原始问题： 如何把二位数组转为一维的数列？
- 两重循环遍历每一排， 计算 row 0 和 row x (x 可以是0 ，1，2，3 ...) 之间的每一列的高度
- 再用单调栈计算那个区间的最大值
- 用 res 变量记录历史最大

作业4：最大矩形

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

Row 0 – Row 0: 1 0 1 0 0

1	0	1	0	0

Row 0 – Row 1: 2 0 2 1 1

1	0	1	0	0
1	0	1	1	1

Row 0 – Row 2: 3 1 3 2 2

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1

Row 0 – Row 3: 4 0 0 3 0

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

总结

- 数组

  支持随机访问

- 链表
  - 根据前驱，后继 自由访问前后节点
  - 利用链表的特性， 自由增加删除节点
  - 可以配合hash， 进行o1 的查找删除插入， lru cache
- 栈
  - 先进后出， 顶部 object 是最近访问的 object
  - 利用单调栈寻找局部最大最小或者边界
- 队列
  - 先进先出， 队列的头部是最先到达的
  - 一般应用在 bfs 广度优先搜索中

Debug

- Print 大法
  - 面试中
  - 平时练习
  - 工作中的log

- Debugger
  - Ide， intellij demo
  - Leetcode 自带

- 逐行对比
  - 和答案的每一行进行比较，并且改成一样的

优秀作业

- 基本
  - 思路清晰， 实现简洁
  - 清晰的命名。 left，right or a， b
  - code style，空格的位置
  - 注释
  - 复杂度分析

- 更好
  - 多种方法完成
  - Try catch block
  - Throw exception

- Reference
  - Clean code
  - https://google.github.io/styleguide/javaguide.html

其他问题 Q & A

THANKS