

# 栈\_队列 (Stack\_Queue)

@M了个J

<https://github.com/CoderMJLee>

<https://space.bilibili.com/325538782>



实力IT教育 [www.520it.com](http://www.520it.com)



# 回顾一下基本概念

## ■ 栈

- 先进后出，后进先出

- 对称

## ■ 队列、双端队列

- 先进先出，后进后出

- 顺序

## 155. 最小栈

设计一个支持 push, pop, top 操作，并能在常数时间内检索到最小元素的栈。

- push(x) -- 将元素 x 推入栈中。
- pop() -- 删除栈顶的元素。
- top() -- 获取栈顶元素。
- getMin() -- 检索栈中的最小元素。

```
MinStack minStack = new MinStack();  
minStack.push(-2);  
minStack.push(0);  
minStack.push(-3);  
minStack.getMin();    --> 返回 -3.  
minStack.pop();  
minStack.top();        --> 返回 0.  
minStack.getMin();     --> 返回 -2.
```

■ 同样的题目：[面试题 03.02. 栈的最小值](#)

3

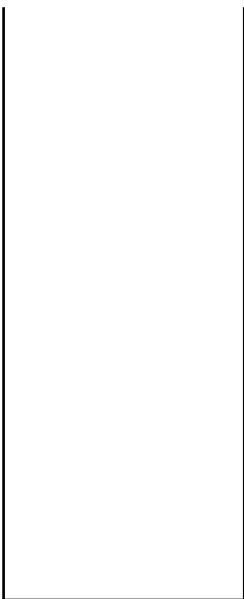
4

2

5

2

1



## 239. 滑动窗口最大值

给定一个数组 *nums*，有一个大小为 *k* 的滑动窗口从数组的最左侧移动到数组的最右侧。你只可以看到在滑动窗口内的 *k* 个数字。滑动窗口每次只向右移动一位。

返回滑动窗口中的最大值。

### 提示：

你可以假设 *k* 总是有效的，在输入数组不为空的情况下， $1 \leq k \leq$  输入数组的大小。

- 时间复杂度： $O(n)$
- 同样的题目：[面试题59 - I. 滑动窗口的最大值](#)
- 此题还可以用【动态规划】的思路

输入：*nums* = [1,3,-1,-3,5,3,6,7]，和 *k* = 3

输出：[3,3,5,5,6,7]

解释：

滑动窗口的位置	最大值
-----	-----
[1 3 -1] -3 5 3 6 7	3
1 [3 -1 -3] 5 3 6 7	3
1 3 [-1 -3 5] 3 6 7	5
1 3 -1 [-3 5 3] 6 7	5
1 3 -1 -3 [5 3 6] 7	6
1 3 -1 -3 5 [3 6 7]	7

← 头 双端队列 (单调队列) 尾

0	1	1	7	2	5	3	3
4	-3	5	-1	6	8	7	2

li				ri			
↓				↓			
0	1	2	3	4	5	6	7
1	7	5	3	-3	-1	8	2

窗口最大值					
0	1	2	3	4	5
7	7	5	3	8	8

li                  ri  
↓                  ↓

0	1	2	3	4	5	6	7
1	7	5	7	-1	-3	-8	2

窗口最大值					
0	1	2	3	4	5
7	7	5	3	8	8

maxIdx 当前窗口最大值的索引	
6	8

# 如何维护双端队列？

① 如果  $\text{nums}[i] \geq \text{nums}[\text{队尾}]$ ，不断删除队尾，直到  $\text{nums}[\text{队尾}] > \text{nums}[i]$

② 将  $i$  加入队尾

③ 如果  $w \geq 0$

- 删除失效的队头（队头  $< w$  就代表失效）
- 更新  $w$  位置的窗口最大值为  $\text{nums}[\text{队头}]$

注意

- 队列中存放的是索引
- 从队头到队尾， $\text{nums}[\text{队列元素}]$ ，是逐渐减小的



## 654. 最大二叉树

给定一个不含重复元素的整数数组。一个以此数组构建的最大二叉树定义如下：

1. 二叉树的根是数组中的最大元素。
2. 左子树是通过数组中最大值左边部分构造出的最大二叉树。
3. 右子树是通过数组中最大值右边部分构造出的最大二叉树。

通过给定的数组构建最大二叉树，并且输出这个树的根节点。

**提示：**

1. 给定的数组的大小在  $[1, 1000]$  之间。

输入：[3,2,1,6,0,5]

输出：返回下面这棵树的根节点：

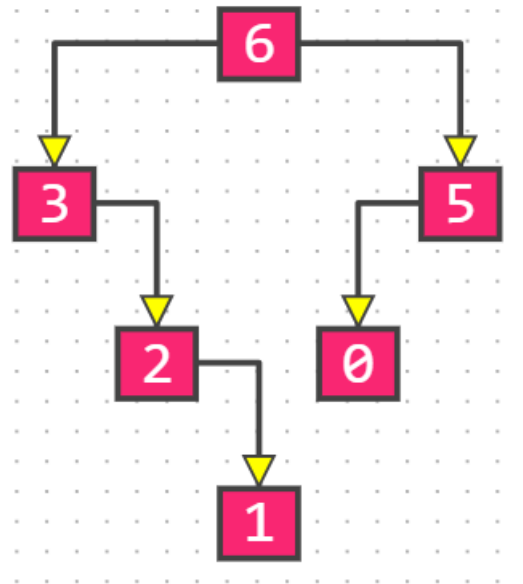


■ 时间复杂度、空间复杂度： $O(n)$

# 题目变种

- 返回一个数组，数组里面存着每个节点的父节点的索引（如果没有父节点，就存-1）

	0	1	2	3	4	5
原数组	3	2	1	6	0	5
返回	3	0	1	-1	5	3



# 利用栈求左、右边第一个比它大的数

0	3	1	2	2	1	3	6	4	0	5	5
---	---	---	---	---	---	---	---	---	---	---	---

单调递减

	0	1	2	3	4	5
	3	2	1	6	0	5
左边第一个比它大的数		0 3	1 2		3 6	3 6
右边第一个比它大的数	3 6	3 6	3 6		5 5	

## 739. 每日温度

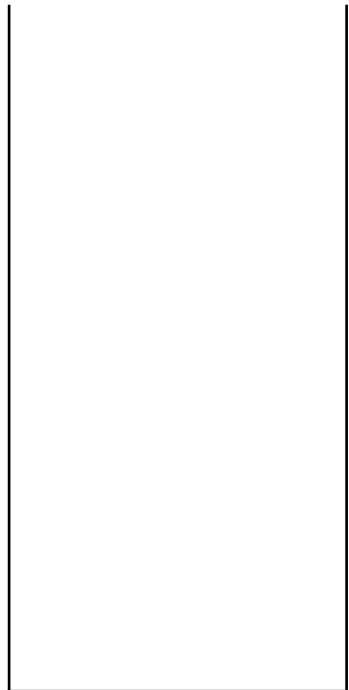
根据每日 `气温` 列表，请重新生成一个列表，对应位置的输入是你需要再等待多久温度才会升高超过该日的天数。如果之后都不会升高，请在该位置用 `0` 来代替。

例如，给定一个列表 `temperatures = [73, 74, 75, 71, 69, 72, 76, 73]`，你的输出应该是 `[1, 1, 4, 2, 1, 1, 0, 0]`。

**提示：** `气温` 列表长度的范围是 `[1, 30000]`。每个气温的值的均为华氏度，都是在 `[30, 100]` 范围内的整数。

0	1	2	3	4	5	6	7
73	74	75	71	69	72	76	73

0	1	2	3	4	5	6	7



0	73	1	74	2	75	3	71
4	69	5	72	6	76	7	73

# 倒推法

	i					j	
	↓					↓	
0	1	2	3	4	5	6	7
73	74	75	75	69	72	76	73

0	1	2	3	4	5	6	7
			3				

- $i$  用来扫描所有的元素，从右往左扫描（ $i$  逐渐递减），一开始  $i$  指向倒数第2个元素
- 对于每一个  $i$ ，一开始令  $j = i + 1$ 
  - ① 如果  $T[i] < T[j]$ ，那么  $values[i] = j - i$ ，然后  $i--$
  - ② 如果  $values[j] == 0$ ，那么  $values[i] == 0$ ，然后  $i--$
  - ③ 否则，设置  $j = j + values[j]$ ，回到步骤①

■  $i$ 用来扫描所有的元素，从右往左扫描（ $i$ 逐渐递减），一开始 $i$ 指向倒数第2个元素



0	1	2	3	4	5	6	7
75	74	75	85	68	69	89	64

0	1	2	3	4	5	6	7
							0

■ 对于每一个 $i$ ，一开始令 $j = i + 1$

① 如果 $T[i] < T[j]$ ，那么 $values[i] = j - i$ ，然后 $i--$

② 如果 $T[i] == T[j]$

① 如果 $values[j] == 0$ ， $values[i] = 0$ ，然后 $i--$

② 如果 $values[j] != 0$ ， $values[i] = values[j] + j - i$ ，然后 $i--$

③ 如果 $T[i] > T[j]$

① 如果 $values[j] == 0$ ， $values[i] = 0$ ，然后 $i--$

② 如果 $values[j] != 0$ ， $j = j + values[j]$ ，重新进入①的判断

## ■ [42. 接雨水](#)



- [20. 有效的括号](#) ([第一季](#)中讲过)
- [232. 用栈实现队列](#)、[面试题09. 用两个栈实现队列](#)、[面试题 03.04. 化栈为队](#) ([第一季](#)中讲过)