

极客时间算法训练营

第二课

前缀和、差分、双指针扫描

李煜东

《算法竞赛进阶指南》作者



目录

1. 前缀和、差分思想
2. 双指针扫描、滑动窗口
3. 单调栈、单调队列

前綴和、差分思想

前缀和

- 一维数组 A
- 前缀和数组 S

```
class Prob {
    Prob(A) { ... }
    int sum(int l, int r) { }
```

$$S[i] = \sum_{j=1}^i A[j] = S[i-1] + A[i]$$

A 的 i 个数相加

	0	1	2	3	4	5
A	0	1	2	1	3	2
			$l=2$		$r=4$	
\Rightarrow		1	2	1	3	2
\Rightarrow		1	2	1	3	2
			$1+2$	$1+2+1$		
S	0	1	3	4	7	9

- 子段和 —— A 中第 l 个数到第 r 个数的和

$$\text{sum}(l, r) = \sum_{i=l}^r A[i] = S[r] - S[l-1]$$

$O(1)$

$(r-l+1) = O(n)$

- 当 A 中都是非负数时，前缀和数组 S 单调递增

$1 \sim r - 1 \sim l - 1$

实战

统计「优美子数组」

<https://leetcode-cn.com/problems/count-number-of-nice-subarrays/>

实战

统计「优美子数组」

<https://leetcode-cn.com/problems/count-number-of-nice-subarrays/>

奇数看作 1, 偶数看作 0, 求前缀和数组 S

连续子数组 $[l, r]$ 中的奇数个数为 $S[r] - S[l - 1]$

枚举右端点 i , 只需找到 ~~i~~ 前面有多少个 j 满足 $S[i] - S[j] = k$

由于 S 单调递增, 只要满足 $S[i] - S[j] = k$ ($k > 0$), j 必然在 i 前面

所以只需用一个计数数组统计 S 中每个值的个数

枚举右端点 i , 看一下等于 $S[i] - k$ 的值有几个就行了

二维前缀和

- 二维数组 A

- 前缀和数组 S

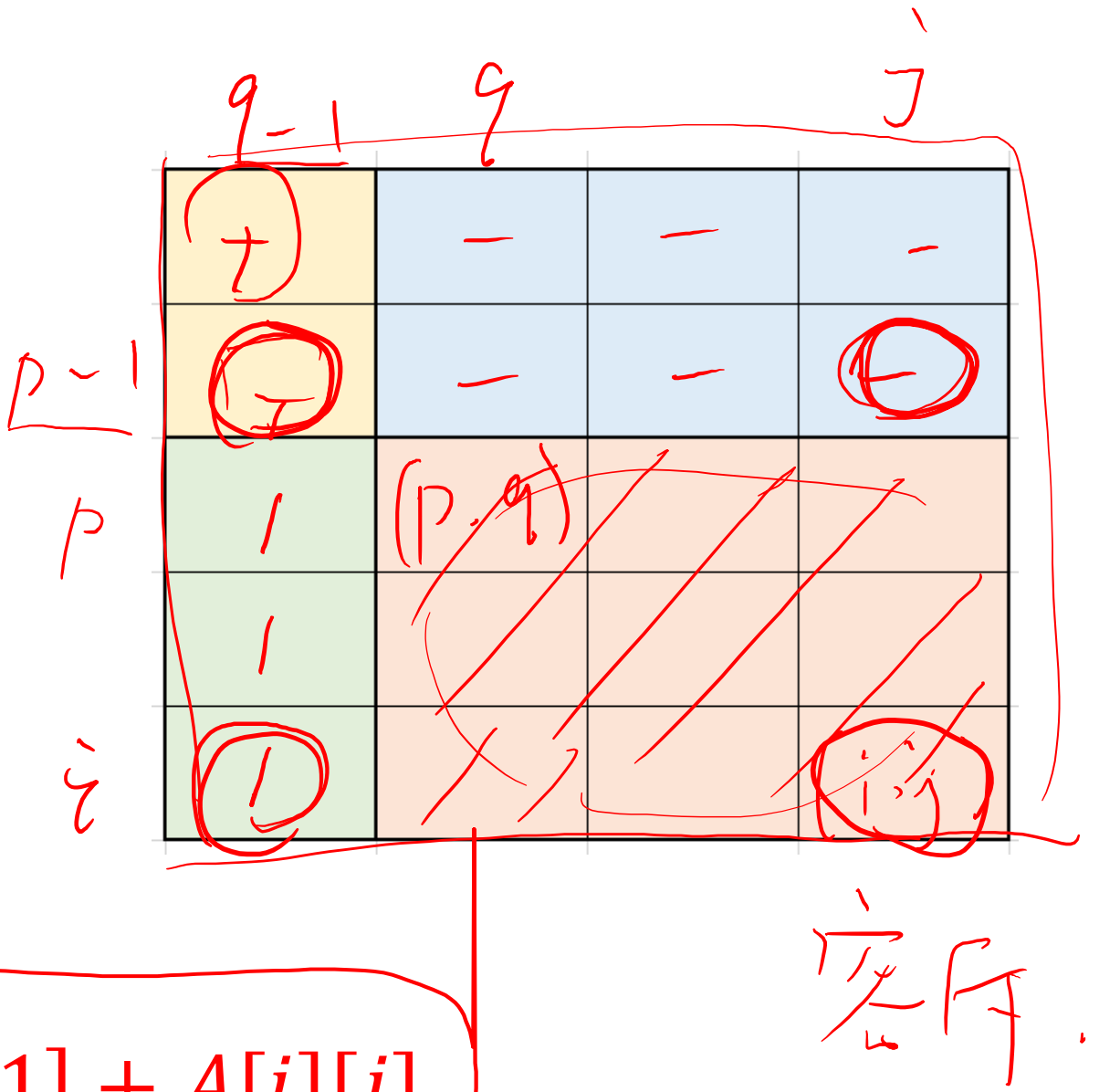
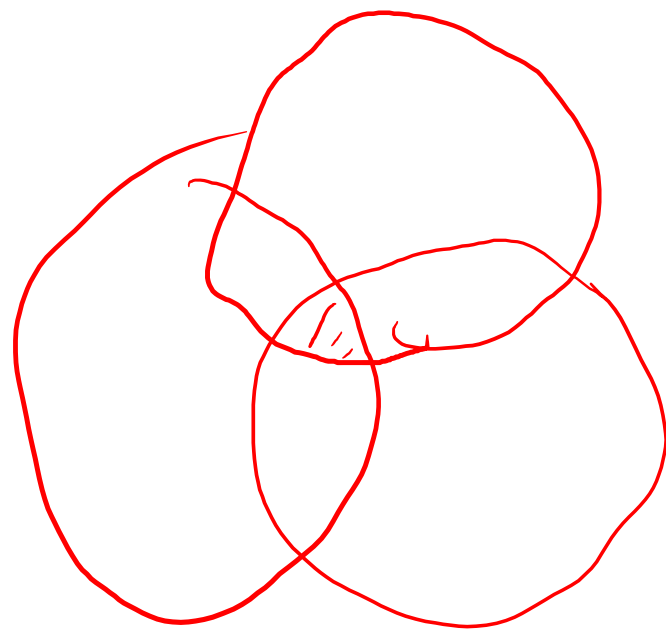
$O(n^2)$

$$S[i][j] = \sum_{x=1}^i \sum_{y=1}^j A[x][y] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i][j]$$

- 子矩阵和 —— 以 (p, q) 为左上角、 (i, j) 为右下角的 A 的子矩阵中数的和

$O(1)$

$$\text{sum}(p, q, i, j) = \sum_{x=p}^i \sum_{y=q}^j A[x][y] = S[i][j] - S[i][q-1] - S[p-1][j] + S[p-1][q-1]$$



实战

二维区域和检索 - 矩阵不可变

<https://leetcode-cn.com/problems/range-sum-query-2d-immutable/>

模板题

差分

- 一维数组 A
- 差分数组 B
- 其中 $B_1 = A_1$, $B_i = A_i - A_{i-1}$ ($2 \leq i \leq n$)

定义

- 差分数组 B 的前缀和数组就是原数组 A

$$4 \text{ calc_b}(\text{calc_sum}(A)) = A$$

- 把 A 的第 l 个数到第 r 个数加 d , B 的变化为: B_l 加 d , B_{r+1} 减 d

Handwritten notes and arrows are present on the left side of the image, indicating the sequence of operations:

- A large red arrow points from the top row (B) down to the bottom row (S).
- A smaller red arrow points from the bottom row (S) up to the middle row (A).
- Red handwritten labels **B**, **A**, and **S** are placed next to their respective rows.
- Red handwritten annotations include:
 - $1-0$ above the first column.
 - $2-1$ above the second column.
 - $1-2$ above the third column.
 - $3-1$ above the fourth column.
 - $2-3$ above the fifth column.
 - $1-2=$ in the empty cell between row B and row A, under the third column.
 - $9-7=$ in the empty cell between row A and row S, under the fifth column.

	2	~	4	
0	+1	+1	+1	
		-2	-2	
		3	~	4
0	1	1	1	0
0	1	-1	-1	0

Handwritten solution for a 2D grid problem. The grid is 2x5. The top row has cells (0,1) with +1, (0,3) with 0, and (0,4) with -1. The bottom row has cells (1,2) with -2 and (1,4) with +2. Below the grid, the values 0, +1, 0, 0, -1 are written. A red line separates this from the next row of values: 0, 1, -2, 0, 1. Another red line separates this from the final row of values: 0, 1, -1, -1, 0. Red circles and lines highlight specific parts of the solution.

	+1	0	0	-1
		-2		+2

0 +1 0 0 -1

0 1 -2 0 1

0 1 -1 -1 0

2
Lug 13 4m

$$c(m, n)$$
$$U(2m + h)$$

实战

航班预定统计

<https://leetcode-cn.com/problems/corporate-flight-bookings/>

模板题

实战

最大子序和

<https://leetcode-cn.com/problems/maximum-subarray/>

实战

最大子序和

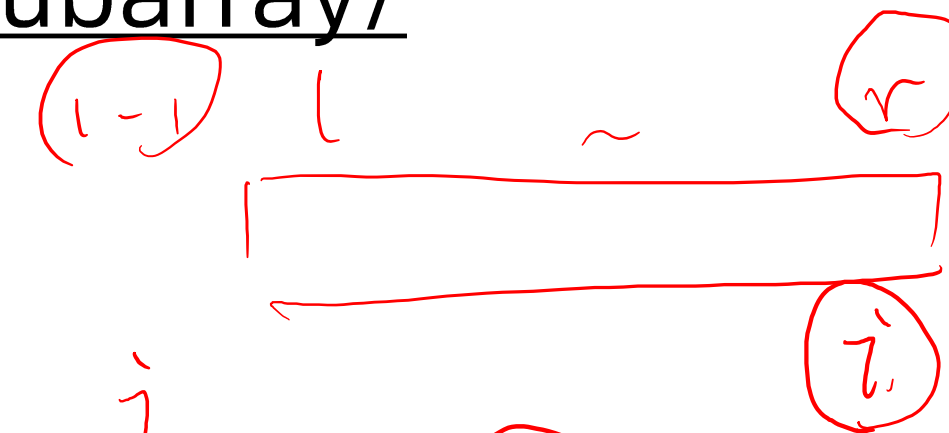
<https://leetcode-cn.com/problems/maximum-subarray/>

解法一：前缀和 + 前缀最小值

求出前缀和数组 S ，枚举右端点 i ，需要找到 i 之前的一个 j 使得 $S[i] - S[j]$ 最大
也就是要让 $S[j]$ 最小，再维护一个 S 的前缀最小值即可

“前缀和”算法不只局限于求和，也可以扩展到前缀最小值、最大值等

$$\begin{aligned} S[r] - S[l-1] &= a[l] + a[l+1] \\ &\quad + \dots + a[r] \end{aligned}$$



↓
 $S[j]$ 越小

实战

最大子序和

<https://leetcode-cn.com/problems/maximum-subarray/>

解法二：贪心

只要“和”是正的，就不断向右扩展

一旦发现“和”是负的，立即舍弃

如果需要方案，用双指针维护当前取的范围即可

双指针扫描、滑动窗口

双指针扫描

用于解决一类基于“子段”的统计问题

子段：数组中连续的一段（下标范围可以用一个闭区间来表示）

这类题目的朴素做法都是两重循环的枚举，枚举左端点 l 、右端点 r ($l \leq r$) $O(n^2)$
优化手法都是找到枚举中的冗余部分，将其去除

优化策略通常有：

- 固定右端点 ^{j} ，看左端点 ^{i} 的取值范围
 - 例如左端点的取值范围是一个前缀，可以用“前缀和”等算法维护前缀信息
- 移动一个端点 ^{i} ，看另一个端点 ^{j} 的变化情况
 - 例如一个端点跟随另一个端点单调移动，像一个“滑动窗口”
 - 此时可以考虑“双指针扫描”

冗余部分

$$\text{num}[i] + \text{num}[j] = t$$

实战

两数之和

<https://leetcode-cn.com/problems/two-sum/>

<https://leetcode-cn.com/problems/two-sum-ii-input-array-is-sorted/>

- Hash?
- 排序 + 双指针扫描?

三数之和

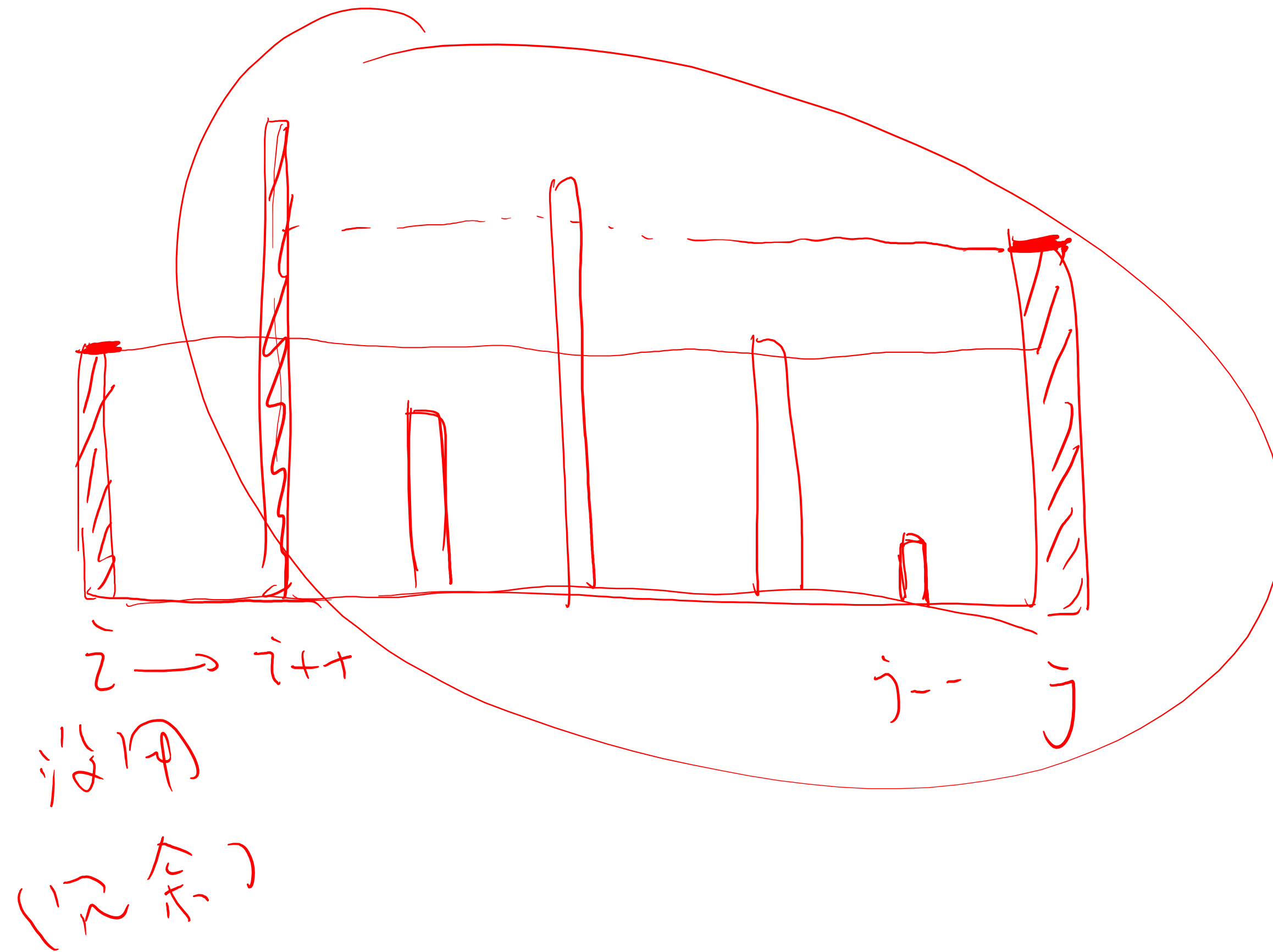
<https://leetcode-cn.com/problems/3sum/>

- 如何避免重复? 相同的数不二次枚举
- 如何简化程序实现? 模块化! 利用“两数之和”

实战

盛最多水的容器

<https://leetcode-cn.com/problems/container-with-most-water/>



实战

盛最多水的容器

<https://leetcode-cn.com/problems/container-with-most-water/>

解题步骤：

1. 两重循环枚举，找冗余
2. 发现关键——盛多少水是由短的那一根决定的，短的算完了就没用了
3. 双指针——两个指针从头尾向中间移动，每次移动短的那根

单调栈、单调队列

单调栈

柱状图中最大的矩形

<https://leetcode-cn.com/problems/largest-rectangle-in-histogram/>

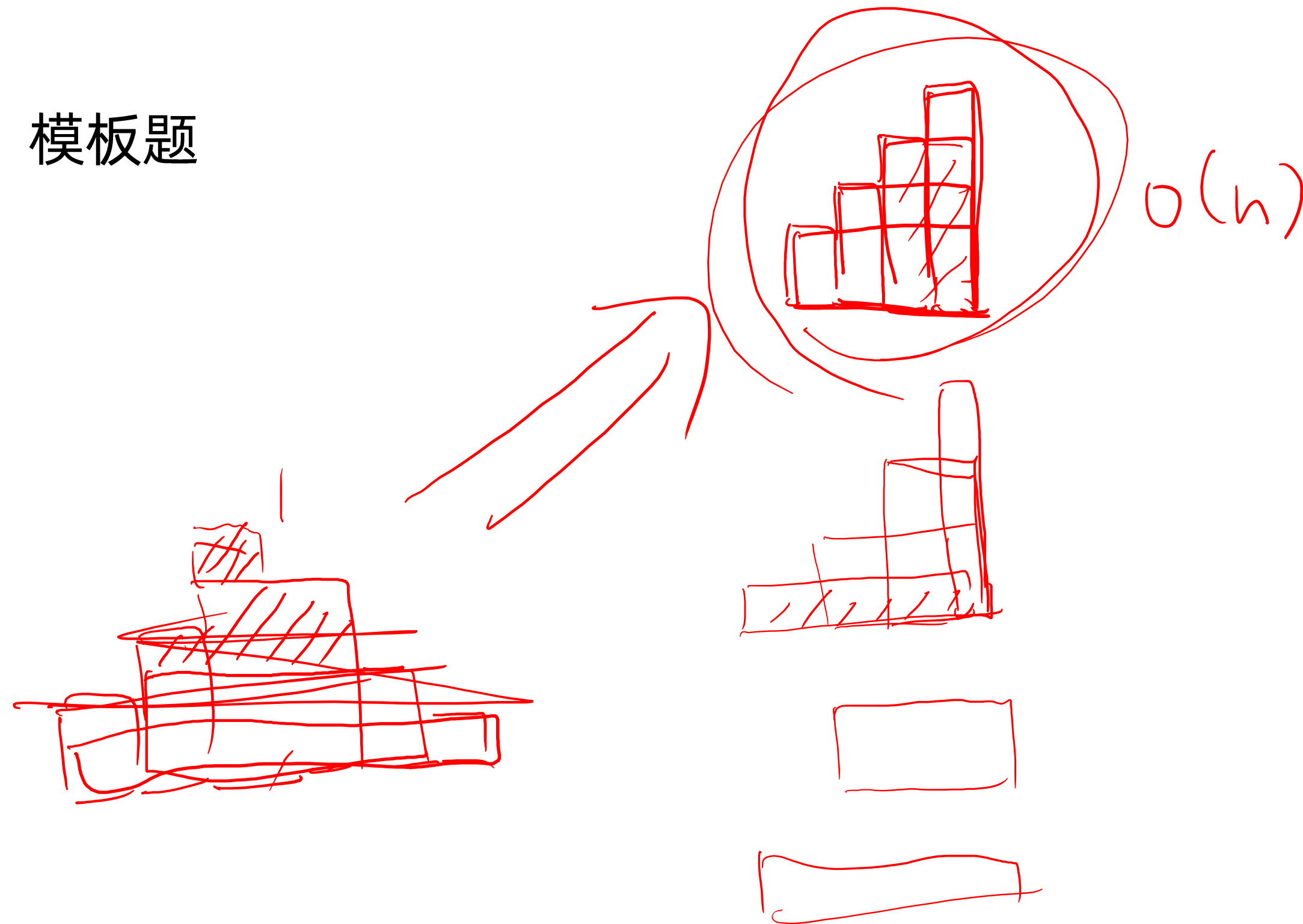
模板题



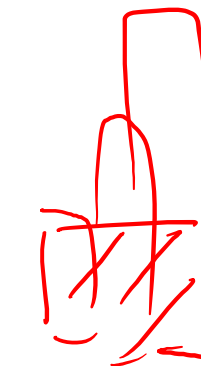
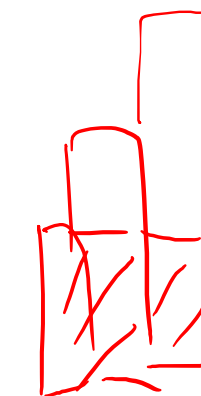
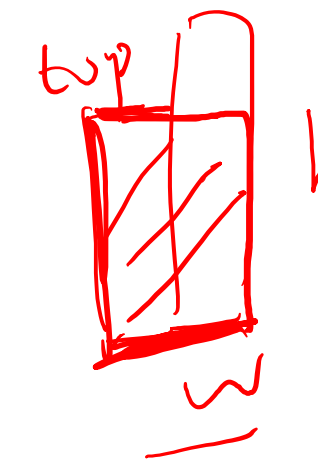
① update ans



高度 ↑



~~h^2~~



单调栈

单调栈题目思维套路：

- 确定递增递减——关键在于考虑“前面不能影响到后面”的条件
- 本题中若 $h[i - 1] > h[i]$ ，则 $h[i - 1]$ 这个高度就无法影响到更后面，自然可以单独计算了

单调栈题目代码套路：

- for 每个元素
- while (栈顶与新元素不满足单调性) { 弹栈，更新答案，累加“宽度” }
- 入栈

单调队列

滑动窗口最大值

<https://leetcode-cn.com/problems/sliding-window-maximum/>

模板题

单调队列

滑动窗口最大值

<https://leetcode-cn.com/problems/sliding-window-maximum/>

单调队列题目思维套路：

- 单调队列维护的是一个候选集合，前面的比较旧，后面的比较新（**时间有单调性**）
- 候选项的某个**属性也具有单调性**
- 确定递增递减的方法——考虑任意两个候选项 $j_1 < j_2$ ，写出 j_1 比 j_2 优的条件

排除冗余的关键：若 j_1 比 j_2 差， j_1 的生命周期还比 j_2 短，那 j_1 就没卵用了

下标

$$\text{num}[j_1] > \text{num}[j_2]$$

单调队列

滑动窗口最大值

<https://leetcode-cn.com/problems/sliding-window-maximum/>

单调队列题目代码套路：

- for 每个元素
- (1) while (队头过期) 队头出队 *合法*
- (2) 取队头为最佳选项，计算答案
- (3) while (队尾与新元素不满足单调性) 队尾出队
- (3) 新元素入队

(2) (3) 的顺序取决于：
 i 是不是候选项

算法对比

思考：

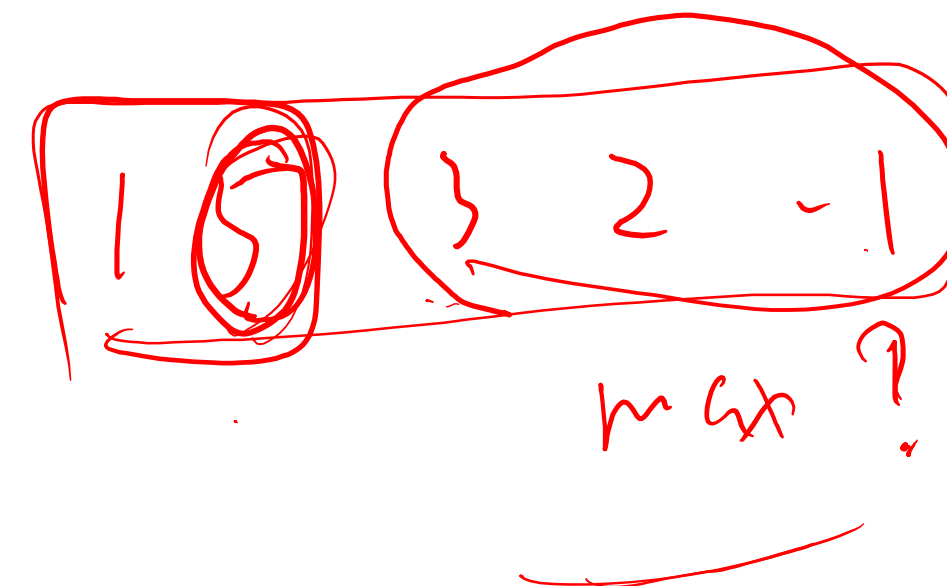
为什么求“子段和”（窗口求和）可以用前缀和？

为什么求“滑动窗口最大值”要用单调队列？

遇到一道跟“子段”（窗口）有关的题，什么时候用前缀和，什么时候用双指针扫描，什么时候用单调队列？

区间减法性质

- 指的是 $[l, r]$ 的信息可以由 $[1, r]$ 和 $[1, l-1]$ 的信息导出
- 满足区间减法，可以用前缀和



维护的信息是关于一个点的，还是一整个候选集合（多个点）的

- 前者用双指针扫描，后者用单调队列

实战

接雨水

<https://leetcode-cn.com/problems/trapping-rain-water/>

前缀/后缀最大值?

单调栈?



Homework + 预习

和为K的子数组

<https://leetcode-cn.com/problems/subarray-sum-equals-k/>

THANKS

 极客时间 | 训练营