

极客时间算法训练营

第四课

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

前綴和、差分思想

前缀和

- 一维数组 A
- 前缀和数组 S

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

	1	2	1	3	2
	1	2	1	3	2
	1	2	1	3	2
0	1	3	4	7	9

- 子段和 —— A 中第 l 个数到第 r 个数的和

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

- 当 A 中都是非负数时，前缀和数组 S 单调递增

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

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

- 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

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

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 $B_1 = A_1, B_i = A_i - A_{i-1} (2 \leq i \leq n)$

	1	1	-1	2	-1
			1-2=		
	1	2	1	3	2
					9-7=
0	1	3	4	7	9

- 差分数组 B 的前缀和数组就是原数组 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

	+1	+1	+1			+1			-1	
		-2	-2				-2		+2	
						0	1	-2	0	1
0	1	-1	-1	0		0	1	-1	-1	0

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

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 ($l \leq r$)

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，看左端点的取值范围
 - 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，看另一个端点的变化情况
 - 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”

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

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

Homework

和为K的子数组

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

THANKS

 极客时间 | 训练营