双指针

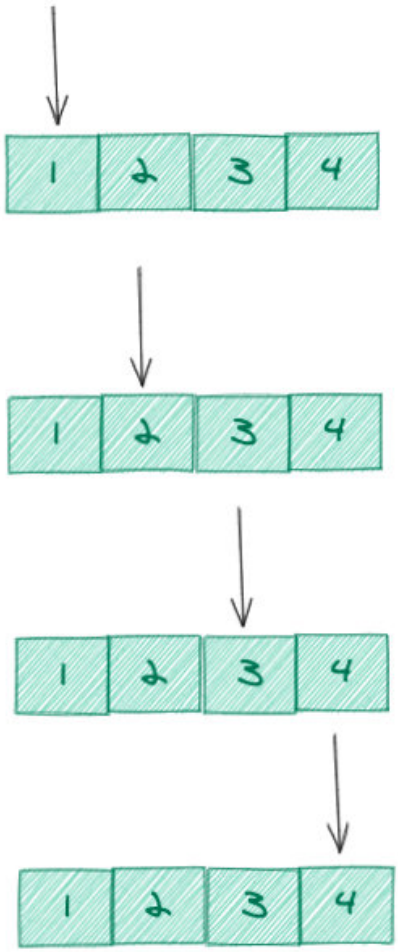
什么是双指针

顾名思议，双指针就是**两个指针**，但是不同于 C，C++ 等语言中的指针， 其是一种**算法思想**。

如果说迭代一个数组，并输出数组每一项需要一个指针来记录当前遍历项，这个过程我们叫单指针的话。

> 这里的“指针”指的是数组的索引

```
for(int i = 0;i < nums.size(); i++) {
  输出(nums[i]);
}
```



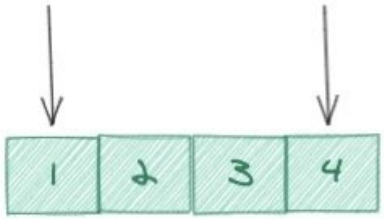
（图 1）

那么双指针实际上就是有两个这样的指针，最为经典的就是二分法中的左右双指针啦。

当然这里的“指针”不仅可以是数组的索引，也可以是别的。不过在大多数情况下，都是数组的索引。因此数组双指针就是有两个这样的数组索引。

```
int l = 0;
int r = nums.size() - 1;

while (l < r) {
    if(一定条件) return 答案
    if(一定条件) l++
    if(一定条件) r--
}
// 由于循环结束的时候 l == r，因此返回 l 和 r 都是一样的
return l
```



（图 2）

读到这里，你发现双指针是一个很宽泛的概念，就好像数组，链表一样，其类型会有很多很多， 比如二分法经常用到 左右端点双指针 。滑动窗口会用到 快慢指针和固定间距指针 。 因此双指针其实是一种综合性很强的类型，类似于数组，栈等。 但是我们这里所讲述的双指针，往往指的是某几种类型的双指针，而不是“只要有两个指针就是双指针了”。

> 有了这样一个**算法框架**，或者算法思维，有很大的好处。它能帮助你理清思路，当你碰到新的问题，在脑海里进行搜索的时候，**双指针**这个词就会在你脑海里闪过，闪过的同时你可以根据**双指针**的所有套路和这道题进行**穷举匹配**，这个思考解题过程本来就像是算法，我会在进阶篇《搜索算法》中详细阐述。

那么究竟我们算法中提到的双指针指的是什么呢？我们一起来看下算法中双指针的常见题型吧。

常见题型有哪些?

这里我将其分为三种类类型，分别是：

1. 快慢指针（两个指针步长不同，一个步长大，一个步长小。典型的是一个步长为 1，另外一个步长为 2）

2. 左右端点指针（两个指针分别指向头尾，并往中间移动，步长关系不确定）

3. 固定间距指针（两个指针间距相同，步长相同）

> 上面是我自己的分类，没有参考别人。可以发现我的分类标准已经覆盖了几乎所有常见的情况。 大家在平时做题的时候一定要养成这样的习惯，将题目类型进行总结，当然这个总结可以是别人总结好的，也可以是自己独立总结的。不管是哪一种，都要进行一定的消化吸收，把它们变成真正属于自己的知识。

不管是哪一种双指针，只考虑双指针部分的话，由于最多还是会遍历整个数组一次，因此时间复杂度取决于步长，如果步长是 1，2 这种常数的话，那么时间复杂度就是 O(N)，如果步长是和数据规模有关，比如二分法每次将数据规模缩小为一半，其时间复杂度就是 O(logN)。实际上，很多二分法都需要两个指针，一个指向上界，一个指向下界， 这个时候其实它就是一种特殊的双指针。

并且由于不管规模多大，我们都只需要最多两个指针，因此空间复杂度是 O(1)。下面我们就来看看双指针的常见套路有哪些。

常见套路

1. 快慢指针

1. 判断链表是否有环

这里给大家推荐两个非常经典的题目，一个是力扣 287 题，一个是 142 题。其中 142 题我在我的 LeetCode 题解仓库中的每日一题板块出过，并且给了很详细的证明和解答。而 287 题相对不直观，比较难以想到，这道题曾被官方选定为每日一题，也是相当经典的。 而这两道题都可以使用快慢双指针解决。

- **287. 寻找重复数**

- **【每日一题】- 2020-01-14 - 142. 环形链表 II · Issue #274 · azl397985856/leetcode**

2. 读写指针。典型的是 删除重复元素

这里推荐我仓库中的一道题， 我给出一个题解，横向对比了几个相似题目，并剖析了这种题目的本质是什么，让你看透题目本质，推荐阅读。

- **80.删除排序数组中的重复项 II**

3. 一次遍历（One Pass）求链表的中点

直观的思路是先进行一次遍历求出链表长度 n ， 然后再次遍历链表，走 n / 2 次即可。 而这需要两次遍历，我们可以使用快慢双指针来优化这个过程。

具体算法是 使用两个指针。快指针每次最两步，慢指针每次走一步，这样当快指针走到链表尾部的时候， 慢指针刚好到达链表中间位置。

2. 左右端点指针

1. 二分查找。

二分查找会在专题篇展开，这里不多说，大家先知道就行了。

2. 暴力枚举中“从大到小枚举”（剪枝）

一个典型的题目是我之前参加官方每日一题的时候给的一个解法，大家可以看下。这种解法是可以 AC 的。同样地，这道题我也给出了三种方法，帮助大家从多个纬度看清这个题目。强烈推荐大家做到一题多解。这对于你做题很多帮助。除了一题多解，还有一个大招是多题同解，这部分我们放在专题篇介绍。

find-the-longest-substring-containing-vowels-in-even

3. 有序数组。

区别于上面的二分查找，这种算法指针移动是连续的，而不是跳跃性的，典型的是 LeetCode 的 两数和 ， 以及 N数和 系列问题。

3. 固定间距指针

1. 一次遍历（One Pass）求链表的倒数第 k 个元素

2. 固定窗口大小的**滑动窗口**

模板(伪代码)

我们来看下上面三种题目的算法框架是什么样的。

这个时候我们没必要纠结具体的语言，这里我直接使用了伪代码，就是防止你掉进细节。当你掌握了这种算法的细节，就应该找几个题目试试。一方面是检测自己是否真的掌握了，另一方面是“细节”，”细节“是人类，尤其是软件工程师最大的敌人，毕竟我们都是 差不多先生 。

1. 快慢指针

```
l = 0
r = 0
while 没有遍历完
  if 一定条件
    l += 1
  r += 1
return 合适的值
```

2. 左右端点指针

```
l = 0
r = n - 1
while l < r
  if 找到了
    return 找到的值
  if 一定条件1
    l += 1
  else if  一定条件2
    r -= 1
return 没找到
```

3. 固定间距指针

```
l = 0
r = k
while 没有遍历完
  自定义逻辑
  l += 1
  r += 1
return 合适的值
```

题目推荐

如果你 差不多 理解了上面的东西，那么可以拿下面的题练练手。Let's Go!

左右端点指针

- 16.3Sum Closest (Medium)
- **42.trapping-rain-water** (Hard)
- 713.Subarray Product Less Than K (Medium)
- 977.Squares of a Sorted Array (Easy)
- Dutch National Flag Problem

> 下面是二分类型

- 33.Search in Rotated Sorted Array (Medium)
- 875.Koko Eating Bananas（Medium）
- 881.Boats to Save People（Medium）

更多二分推荐：

- **search-for-range**
- **search-insert-position**
- **search-a-2d-matrix**
- **first-bad-version**
- **find-minimum-in-rotated-sorted-array**
- **find-minimum-in-rotated-sorted-array-ii**
- **search-in-rotated-sorted-array**
- **search-in-rotated-sorted-array-ii**

快慢指针

- 26.Remove Duplicates from Sorted Array（Easy）
- 141.Linked List Cycle (Easy)
- 142.Linked List Cycle II（Medium）
- 287.Find the Duplicate Number（Medium）
- 202.Happy Number (Easy)

固定间距指针

- 1456.Maximum Number of Vowels in a Substring of Given Length（Medium）

滑动窗口

其实滑动窗口就是借助双指针来完成的，其中两个指针分别是窗口的左右两个边界。

- 如果我使用固定间距的双指针，那就是窗口大小固定的双指针。
- 如果我使用快慢双指针，那就是可变窗口大小的双指针，一般这种题目都是求满足一定条件的窗口的最大或者最小值。

滑动窗口见专题篇的[滑动窗口专题]

可变窗口大小模板（伪代码）

固定窗口大小和上面代码类似，直接使用就行。 因此这里重点看下可变窗口大小模板。

```
初始化慢指针 = 0
初始化 ans

for 快指针 in 可迭代集合
    更新窗口内信息
    while 窗口内不符合题意
        扩展或者收缩窗口
        慢指针移动
    更新答案
返回 ans
```

代码

以下是 209 题目的代码，使用 Python 编写，大家意会即可。

```python
class Solution:
    def minSubArrayLen(self, s: int, nums: List[int]) -> int:
        l = total = 0
        ans = len(nums) + 1
        for r in range(len(nums)):
            total += nums[r]
            while total >= s:
                ans = min(ans, r - l + 1)
                total -= nums[l]
                l += 1
        return  0 if ans == len(nums) + 1 else ans
```

题目列表（有题解）

以下题目有的信息比较直接，有的题目信息比较隐蔽，需要自己发掘

- 【Python，JavaScript】滑动窗口（3. 无重复字符的最长子串）
- 76. 最小覆盖子串
- 209. 长度最小的子数组
- 【Python】滑动窗口（438. 找到字符串中所有字母异位词）

- 【904. 水果成篮】（Python3）
- 【930. 和相同的二元子数组】（Java，Python）
- 【992. K 个不同整数的子数组】滑动窗口（Python）
- 978. 最长湍流子数组
- 【1004. 最大连续 1 的个数 III】滑动窗口（Python3）
- 【1234. 替换子串得到平衡字符串】[Java/C++/Python] Sliding Window
- 【1248. 统计「优美子数组」】滑动窗口（Python）
- 1658. 将 x 减到 0 的最小操作数

如果你理解了滑动窗口，那就快用我的模板试试解决这些问题吧~

小技巧

一般而言，拿到一道题我们首先考虑的是如何暴力地解决，然后再思考如何进行优化。

比如一道题的暴力解法时间复杂度是 $O(n^2)$，不妨假设这里的 $O(n^2)$ 来源于两层的暴力枚举。而根据题目提供的数据规模我们需要 n 的时间复杂度解决。那么我们需要考虑将暴力枚举优化一下。

有多种思路进行优化。比如：

- 使用双指针将两层枚举改为”一层“。比如典型的两数和，以及 Triangle-Triplets 就是这种情况。
- 使用滑动窗口解决。比如我们枚举的两个端点后需要计算连续区间的信息，那么使用滑动窗口就可以减少重复计算。
- 使用二分。先一层枚举，然后利用有序进行二分。如果题目并不有序可以考虑排序或者构造有序序列。关于这部分，二分章节会详细介绍。

当大家做题越来越多，见过题型越来越多，不妨自己总结这种经验，这样做题的时候思路就不断涌现出来了！

扩展阅读

- LeetCode Sliding Window Series Discussion

双数组

除了双指针，还有一个比较类似的技巧是双数组。

通常我们会记录两个数组 left 和 right。进而使用这两个数组来完成我们的算法。

而这里的 left[i] 通常都是记录 i 左侧第一个 xxxxx，相应地 right[i] 记录的是右侧第一个 xxxxx。

> 这里的 xxx 根据题目求解不同而不同

比如 **2055. 蜡烛之间的盘子** 这道题。我们关心的是每次查询 query[i] 中的 query[i][0] 右侧（包含自身）第一个蜡烛的位置，同样我们关系 query[i][1] 左侧（包含自身）第一个蜡烛的位置。根据这两个位置，我们就知道夹在中间的**总位置**有多少。那么中间的区域有多少盘子呢？由于是连续区间求和，我们其实可以用前缀和空间换时间，当然我们也可以使用二分来处理。

再额外推荐两道题目供大家练习消化：

- **42. 接雨水** 这道题使用双数组的核心在于每个柱子下雨后接水量 h[i]等于左右两侧柱子的最大值中的较小值，即 h[i] = Math.min(左边柱子最大值, 右边柱子最大值)
- **Fair-Pay** 这道题可以双数组的核心在于每个人都需要比它相邻（**左和右**）的表现差的多至少一美元。因此仅考虑左和仅考虑右的结果取较大值即可。

总结

广义的双指针是一个非常宽泛的话题，因为其实我只需要有两个指针就可以了。双指针只是一个方便记忆的名字，实际上如果是固定窗口大小的滑动窗口只需要一个指针就够了，用不到双指针，不过我习惯还是将其归于双指针。

而我们讨论的是狭义的双指针，具体来说有以下三种类型：

1. 快慢指针。典型的就是一次遍历找中点，找链表交点，找倒数第几个链表节点。相关题目不多，建议都做一下。
2. 首尾双指针。这个类型题目很多，典型的比如两数和，**Triangle-Triplets**，大家可以用这两道题练习一下。
3. 滑动窗口双指针。（使用两个指针表示一个窗口） 关于这个力扣有标签，大家直接根据标签找就行。

每一种都给了使用场景和模板供大家参考，这节涉及的题目比较多，但是使用我们的思维方式和模板都可以轻松解决，前提是你要花时间理解和练习。

有时候也不能太思维定式，比如 https://leetcode-cn.com/problems/consecutive-characters/ 这道题根本就没必要双指针什么的。再比如：https://lucifer.ren/blog/2020/05/31/101.symmetric-tree/

双指针还有很多其他的类型，比如之后要讲的二分法。二分法的实现大多也是使用两个指针，并通过不断二分解空间实现。

上一页 下一页