

21 | 队列与单调队列：滑动区间最大值

2020-03-05 胡光

人人都能学会的编程入门课

[进入课程 >](#)



讲述：胡光

时长 13:48 大小 12.65M



你好，我是胡光，欢迎回来。

上节课呢，我们学习了二分查找的基本思想，以及明确了二分答案所使用的问题模型，你会发现，正因为问题具有单调性，我们才可以使用二分查找算法对问题求解过程进行加速。

今天呢，我将带你学习一种性质有趣、简单且高效的数据结构，叫做：单调队列。学习这个数据结构的时候呢，我们还是要强调一下那句话：数据结构，就是定义一种性质，并且维护这种性质。




今日任务

在正式开始学习之前呢，先来看一下今天这 10 分钟的任务吧。

滑动区间最大值，就是指在固定区间长度的前提下，在一个序列上，从前到后滑动这个区间窗口，每次窗口内部的最大值，就组成了滑动区间最大值。

例如，给你如下包含 8 个数字的序列，区间长度设置为 3：

 复制代码

```
1  [6 4 2] 10 3 8 5 9 -> 6
2  6 [4 2 10] 3 8 5 9 -> 10
3  6 4 [2 10 3] 8 5 9 -> 10
4  6 4 2 [10 3 8] 5 9 -> 10
5  6 4 2 10 [3 8 5] 9 -> 8
6  6 4 2 10 3 [8 5 9] -> 9
```

滑动区间从数字 6 开始出发，每次向右移动一个数字，同时把左边的一个数字丢出去，保持区间长度为 3，最后移动到数字 9 停止。可以看到，这个序列共包含 8 个数字，所以最后形成的滑动区间最大值共有 6 个，依次是 6、10、10、10、8、9。

面对这个问题，你很容易采用 $O(nm)$ 的算法来完成， n 是区间长度， m 是窗口长度，就是枚举区间的终止位置，每次扫描区间内部，获得最大值。

而我今天要给你讲的这种方法，能让时间复杂度降低到 $O(n)$ ，你可以认为是对原序列扫描一遍，就能得到问题的答案。这究竟是什么样神奇的方法呢？带着这份好奇，我们开始今天的课程吧！

必知必会，查缺补漏

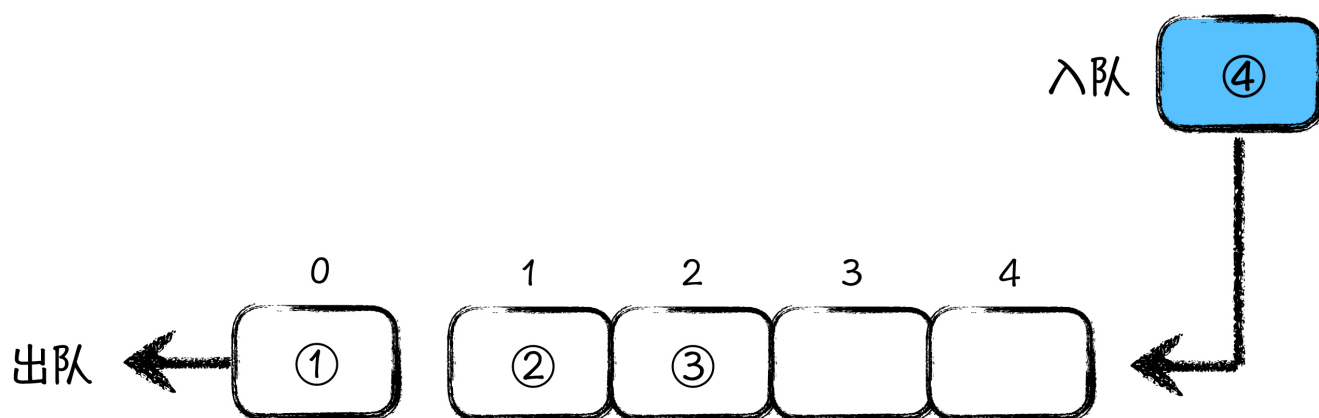
想要完成今天这个任务呢，你必须掌握今天我将要教给你的一种新的结构，就是：单调队列。

1. 初识队列

首先让我们来认识一下最简单的队列结构，举一个生活中最常见的例子：火车站排队买票，你应该都经历过吧？售票员坐在窗口里面，每次只能服务队列中排在最前面的那个人，每当

有人买完票，都会从队列的头部离开，后面的人上前一步，接替离开的人向售票员购票，当有其他人想要来买票的时候，必须从队列的末尾开始排队。这种结构就是典型的队列结构。

我们计算机中的队列，和买票的这个队列是一样的，先到先得，先入先出，每个元素都是从队列尾部入队，在头部被处理完后再出队。如下图所示：



队列结构示意图

如图所示，队列就像一个数组一样，每个元素从数组尾部进入数组，从头部出数组。这种结构很简单，你应该很容易理解它的工作顺序。任何事物，往往就是看起来越简单，想要掌握其真谛就越难。就像是我给你一把锤子，你知道这东西大概可以干什么，而我要是给你一块铁，你可能就懵了。其实队列就是这种表面简单，可作用却不简单的数据结构。

想要理解队列，你就必须理解一句话，叫做：计算机是很专注的。什么意思呢，我们回忆一下之前讲到的链表判环，作为人类的你和我，可以一眼就看出来链表中是否有环。而对于计算机程序来说，只有指针指向的地方，才是它能看得到的地方。所以，我们才费了很大的力气，为计算机设计了一个快慢指针的算法，来判断链表中是否有环。

实际上，当我们在实现程序的时候，我们不仅要把数据存储计算机中，我们还要规定计算机处理这些数据的顺序。想一想，我们之前设计的所有的循环程序，不就是在规定计算机的处理顺序么？

而今天我们学习的这个队列，你可以把其中的元素，看成是计算机要处理的一个个的任务，那么队列结构，其实就是规定了这些任务的处理顺序，程序只从队列头部取任务，先到先得处理，后到的任务，需要在队列后面排着，直到轮到它，这样就可以把计算机的专注与高效发挥到极致。

学习编程，与其说是将我们的思维转换成代码，不如说是将我们的思维，锻炼成计算机的思维。注意，计算机的处理逻辑，是有顺序的。今天，我们所说的队列，就代表了一种顺序。后面，我们还会介绍另外一种顺序的代表，就是“栈”，到时候我再详细讲解。

2. 队列升级：单调队列

讲完了最简单的队列以后，下面就来让我们学习一种队列的升级产物：单调队列。在正式讲解单调队列之前，让我们来讲一个现实中的单调队列的例子。

假设你的学长张三，作为乒乓球体育生，很幸运地进了高中学校的校队。学校规定当前校队中，能力最强的人，才有可能代表学校去参加比赛。那么在校的这三年，张三都有机会代表学校参加国家赛。

高一的时候张三战斗力 85，比他能力强的有两个人，一个是高二的孔令辉，战斗力 93，一个是高三刘国梁，战斗力 98。那么此时，能代表学校参加比赛的只有最高战力的刘国梁。

刘国梁（高三）	孔令辉（高二）	张三（高一）
98	93	85

过了一年，张三上了高二，原高二的孔令辉上了高三，刘国梁毕业了，如果张三的战力比后面上来的新生要强，那么他再等一年，有可能在高三的时候代表学校参赛。然而这时作为高一乒乓球体育生的你也进入了校队，战斗力 88。悲剧出现了，因为只要你在学校，张三永远不可能参加国家赛了。

孔令辉（高三）	你（高一）	
93	88	

这个时候你欣喜若狂，因为再熬一年，如果新学弟战斗力没你高的话，你就能代表学校去参加国家赛了。很快一年又过去了，你终于熬到了高二，同时也迎来了一名新学弟张继科，战斗力 90。悲剧再次上演，你和张三一样，也失去了代表学校参赛的机会了，此时你的心情，是不是五味杂陈？

张继科（高一）		
90		

上面的几个校队名单呢，就是我们所谓的单调队列，如果把学生从高年级到低年级排列，随着时间的流逝，这本身就是一个队列结构，高年级的同学从队列头部毕业，低年级的同学从队列尾部进入。

而这个校队名单，记录的是最有可能代表学校参加比赛队员的名字。刘国梁毕业了，最有可能接班的是孔令辉，孔令辉毕业了，最有可能接班的是张三。而当你进入队列的那一刻，张三尽管比你入队早，但战力没你高，所以张三就永远失去了机会。后来张继科进入队列，你遭遇了和张三一样的悲剧。


如果你要是仔细观察校队名单，你会发现校队名单上，永远是按照能力值的从高到低，来记录学校里面的种子选手。这个名单，既有队列的样子，又有单调的性质，所以称为“单调队列”。

单调队列的作用，就是用来维护在**队列处理顺序**中的区间最大值。就像上面所说的校队名单，维护的就是区间长度为 3 时候的最大值。当一个新的元素入队的时候，它会把其前面违反单调性的元素，都从队列中踢掉，就像张继科的入学，把你踢出了校队名单，最终他成为了队列里的最大值。

滑动区间最大值

让我们回到开始的求“滑动窗口最大值”的任务。其实，滑动窗口每次向后滑动一位，会有一个元素从队首出队，同时也会有一个元素从队尾入队，所以滑动窗口的过程，就遵照了我们所谓的队列处理顺序。

而这个任务，本身就是求区间最大值的，所以也符合了单调队列应用的场景：维护在**队列处理顺序**中的区间最大值。下面呢，我们就来看一下具体代码：

 复制代码

```
1 #define MAX_N 1000
2 int q[MAX_N + 5], head, tail;
3 void interval_max_number(int *a, int n, int m) {
4     head = tail = 0;
5     for (int i = 0; i < n; i++) {
6         // a[i] 入队，将违反单调性的从队列 q 中踢出
7         while (head < tail && a[q[tail - 1]] < a[i]) tail--;
8         q[tail++] = i; // i 入队
9         // 判断队列头部元素是否出了窗口范围
10        if (i - m == q[head]) head++;
11        // 输出区间内最大值
12        if (i + 1 >= m) {
13            printf("interval(%d, %d)", i - m + 1, i);
14            printf(" = %d\n", a[q[head]]);
15        }
16    }
17    return ;
18 }
```

如代码所示，interval_max_number 函数，传入三个参数，数组首地址 a，元素数量 n 以及 区间长度 m。代码中的 q 数组，后续的作用就是模拟单调队列，head 与 tail 代表了队

列的头尾下标，这里我们采用左闭右开式的表示方法，也就是 head 和 tail 所指示的区间范围内，包含 head 所指位置，但不包含 tail 所指位置。

函数内部，依次处理数组中的每个元素，每次处理相应元素的时候，涉及到两个过程：

第一个过程，是将当前元素入队。在入队之前，将队列尾部违反单调性的元素都从队列中踢出，这个就是第 7 行 while 过程的作用，之后就是将编号 i 入队即可。这里注意，单调队列里面，存储的是 a 数组的下标，而不是 a 数组的值。其实存储了下标，我们就可以索引到值，而在上一节二分查找的课里面，我们也见识过了，要是存储了值，想要反向索引下标是比较困难的。

第二个过程呢，就是判断单调队列头部的元素是否超出了窗口范围，也就是前面我们例子中你的学长毕业的过程，如果元素下标已经超出了窗口范围，就将队列头部元素出队。

这样我们就可以保证，我们每次输出的，就都是滑动窗口内部的区间最大值了。

课程小结

以上就是我们今天要学习的单调队列的内容，关于单调队列的知识，你在理解其处理过程的时候，更应该记住单调队列应用的场景：**就是维护队列处理顺序中的区间最大值。**

这个里面，需要重点强调一个**队列处理顺序**。也就是说，如果你可以把一个问题的求解顺序，抽象成队列求解顺序，并且在这个过程中，你还需要维护区间最大值，那么翻出“单调队列”，准能帮助你大幅度提升处理速度！而单调队列，无论是入队，还是出队，操作完以后，一定要保证队列内部满足单调性，这就是开头我们说的：定义一种性质，并且维护这种性质。单调队列，维护的就是单调性。

最后，我们来简单说一下单调队列处理单个元素的平均时间复杂度为什么是 $O(1)$ 的。假设我们要处理 n 个元素，从整体上来看，每个元素会入队列 1 次，出队列最多也是 1 次，那么 n 个元素的总操作次数不会超过 $2 \times n$ 次，平均到一个元素上就是 2 次，也就是常数次，记作 $O(1)$ 时间复杂度。由此得知，处理 n 个元素的总时间复杂度，就是 $O(n)$ 。

今天没有思考题，因为这节课的内容只是作为一个铺垫，下节课关于“栈”的知识才是重头戏。我也希望你对这节课的内容认真学习体会，可以的话，在留言区说说你的看法和思考。

好了，单调队列的知识，就讲到这里了，我是胡光，我们下期见。

课程学习计划

关注极客时间服务号 每日学习签到

月领 25+ 极客币

【点击】保存图片，打开【微信】扫码>>>



© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 20 | 二分查找：提升程序的查找效率

下一篇 22 | 栈与单调栈：最大矩形面积

精选留言 (3)

写留言



胖胖胖

2020-03-08

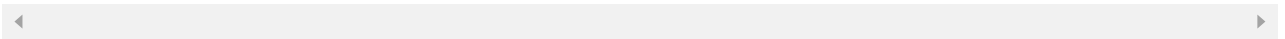
老师，我目前在北京某985读书，平时实验室做得机器学习，大部分代码就是简单得把矩阵运算翻译成matlab和python，但感觉除了matlab其他得编程语言都写得很业余，想来是自己非科班不够努力，同时其他语言对计算得封装是不如matlab完善的。但应该什么样的训练方法可以快速提高呢？本科偏数学，毕业后想去大厂面试（机器学习&深度学习）算法岗，大概还有5年的硕博学习时间，除了数学上的持续学习，（计算机科班培养计划这...
展开 ∨

作者回复: 你好，听了你的情况，我对你的判断是，你的理论知识应该是很扎实的，只是缺少实践经验，建议如下：

- 1、平时多在 Linux 系统下学习和工作，可以选择 CentOS 或者 Ubuntu 均可
- 2、试着使用 github 上传自己用 python 封装的机器学习模型算法

3、如果有条件，尽量在分布式环境下跑跑算法，课外有时间的话，可以读一读分布式计算与并行计算方面的东西。

只能给你如上建议了，仅供参考。



HappyJoo

2020-03-05

老师不好意思，我知道我代码出错的概率大很多很多：

输入：nums = [1,3,-1,-3,5,3,6,7], n = 8, m = 3

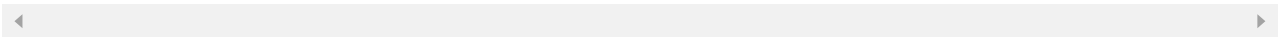
错误输出：[3, 3, 5, 3, 6, 7]

正确输出：[3, 3, 5, 5, 6, 7]

第四项不知道为啥错了，是copy您的代码然后输出的，不知道是不是您的代码哪里写错...

展开 ▾

作者回复: 不是你的错误，你的是对的！是我代码的问题，队列中最后一个元素应该是，q[tail - 1]，最近疫情，都把我在家里呆傻了。_-||| sorry



1



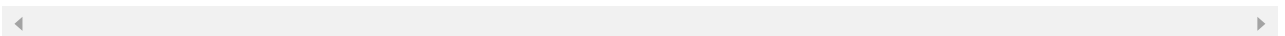
HappyJoo

2020-03-05

老师你好啊~我现在不知道要看啥，c专家与编程好像还早了点，课程其他内容虽然还没有完全消化，但是也差不多了，有点不知道应该干什么，去leetcode刷题吗？还是看一些例如C与指针，啃一下大部头？还是去看看《数据结构与算法之美》，或者《深入了解计算机原理》，或者《编译原理之美》？又或者去看看python？有点不知道要走哪个方向，迷茫~

展开 ▾

作者回复: 你的目标是啥呢？



9



