

[首页](#)[新闻](#)[博问](#)[专区](#)[闪存](#)[班级](#)[代码改变世界](#)[注册](#)[登录](#)

Grandyang

仰天长啸仗剑红尘，冬去春来寒暑几更...

[博客园](#)[首页](#)[新随笔](#)[联系](#)[订阅](#)[管理](#)

随笔 - 1566 文章 - 1 评论 - 4413 阅读 - 125

赞助

LeetCode Binary Search Summary 二分搜索法小结

二分查找法作为一种常见的查找方法，将原本是线性时间提升到了对数时间范围，大大缩短了搜索时间，具有很大的应用场景，而在 LeetCode 中，要运用二分搜索法来解的题目也有很多，但是实际上二分查找法的查找目标有很多种，而且在细节写法也有一些变化。之前有网友留言希望博主能针对二分查找法的具体写法做个总结，博主由于之前一直很忙，一直拖着没写，为了树立博主言出必行的正面形象，不能再无限制的拖下去了，那么今天就来做个了断吧，总结写起来~ (以下内容均为博主自己的总结，并不权威，权当参考，欢迎各位大神们留言讨论指正)

根据查找的目标不同，博主将二分查找法主要分为以下五类：

第一类：需查找和目标值完全相等的数

这是最简单的一类，也是我们最开始学二分查找法需要解决的问题，比如我们有数组 $[2, 4, 5, 6, 9]$ ， $target = 6$ ，那么我们可以写出二分查找法的代码如下：

公告



(请关注下方微信公众号，并留言跟博主交流)

Github同步地址，欢迎star♡

github.com/grandyang/leetcode

搜索【shua2sum】或扫描二维码
关注微信公众号【刷尽天下】



```
int find(vector<int>& nums, int target) {  
    int left = 0, right = nums.size();  
    while (left < right) {  
        int mid = left + (right - left) / 2;  
        if (nums[mid] == target) return mid;  
        else if (nums[mid] < target) left = mid + 1;  
        else right = mid;  
    }  
    return -1;  
}
```



会返回3，也就是 target 的在数组中的位置。注意二分查找法的写法并不唯一，主要可以变动地方有四处：

第一处是 right 的初始化，可以写成 nums.size() 或者 nums.size() - 1。

第二处是 left 和 right 的关系，可以写成 left < right 或者 left <= right。

第三处是更新 right 的赋值，可以写成 right = mid 或者 right = mid - 1。

第四处是最后返回值，可以返回 left, right, 或 right - 1。

但是这些不同的写法并不能随机的组合，像博主的那种写法，若 right 初始化为了 nums.size()，那么就必须用 left < right，而最后的 right 的赋值必须用 right = mid。但是如果我们 right 初始化为 nums.size() - 1，那么就必须用 left <= right，并且right的赋值要写成 right = mid - 1，不然就会出错。所以博主的建议是选择一套自己喜欢的写法，并且记住，实在不行就带简单的例子来一步一步执行，确定正确的写法也行。

第一类应用实例：

[Intersection of Two Arrays](#)

第二类：查找第一个不小于目标值的数，可变形为查找最后一个小于目标值的数



使用方法：

- 回复数字【0】随机推送一道题。
- 回复区间【1 - 1090】内任意数字推送对应的题目。
- 回复关键字 例如【Two Sum】推送对应的题目。
- 回复【all】推送题目汇总列表。
- 回复【other】推送相关总结帖。
- 回复任意文字跟博主留言交流^.

赞助

喜欢本博客可以请博主喝杯咖啡~

微信打赏



Venmo 打赏

5, 6, 9] 中查找数字3, 就会返回数字4的位置; 在数组 [0, 1, 1, 1, 1] 中查找数字1, 就会返回第一个数字1的位置。我们可以使用如下代码:



```
int find(vector<int>& nums, int target) {  
    int left = 0, right = nums.size();  
    while (left < right) {  
        int mid = left + (right - left) / 2;  
        if (nums[mid] < target) left = mid + 1;  
        else right = mid;  
    }  
    return right;  
}
```



最后我们需要返回的位置就是 right 指针指向的地方。在 C++ 的 STL 中有专门的查找第一个不小于目标值的数的函数 lower_bound, 在博主的解法中也会时不时的用到这个函数。但是如果面试的时候人家不让使用内置函数, 那么我们只能老老实实写上面这段二分查找的函数。

这一类可以轻松的变形为查找最后一个小于目标值的数, 怎么变呢。我们已经找到了第一个不小于目标值的数, 那么再往前退一位, 返回 right - 1, 就是最后一个小于目标值的数。

第二类应用实例:

[Heaters](#), [Arranging Coins](#), [Valid Perfect Square](#), [Max Sum of Rectangle No Larger Than K](#), [Russian Doll Envelopes](#)

第二类变形应用: [Valid Triangle Number](#)

第三类: 查找第一个大于目标值的数, 可变形为查找最后一个不大于目标值的数



venmo

昵称: Grandyang

园龄: 9年

粉丝: 1275

关注: 36

+加关注

赞助

| 2021年4月 | | | | | | |
|---------|----|----|----|----|----|----|
| 日 | 一 | 二 | 三 | 四 | 五 | 六 |
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

搜索

| | |
|----------------------|------|
| <input type="text"/> | 找找看 |
| <input type="text"/> | 谷歌搜索 |

最新随笔

- 1.[LeetCode] 1093. Statistics from a Large Sample 大样本统计
- 2.[LeetCode] 1092. Shortest Common Supersequence 最短公共超

target, 就这一个小小的变化, 其实直接就改变了搜索的方向, 使得在数组中有很多跟目标值相同的数字存在的情况下, 返回最后一个相同的数字的下一个位置。比如在数组 [2, 4, 5, 6, 9] 中查找数字3, 还是返回数字4的位置, 这跟上面那查找方式返回的结果相同, 因为数字4在此数组中既是第一个不小于目标值3的数, 也是第一个大于目标值3的数, 所以 make sense; 在数组 [0, 1, 1, 1, 1] 中查找数字1, 就会返回坐标5, 通过对比返回的坐标和数组的长度, 我们就知道是否存在这样一个大于目标值的数。参见下面的代码:



```
int find(vector<int>& nums, int target) {  
    int left = 0, right = nums.size();  
    while (left < right) {  
        int mid = left + (right - left) / 2;  
        if (nums[mid] <= target) left = mid + 1;  
        else right = mid;  
    }  
    return right;  
}
```



这一类可以轻松的变形为查找最后一个不大于目标值的数, 怎么变呢。我们已经找到了第一个大于目标值的数, 那么再往前退一位, 返回 right - 1, 就是最后一个不大于目标值的数。比如在数组 [0, 1, 1, 1, 1] 中查找数字1, 就会返回最后一个数字1的位置4, 这在有些情况下是需要这么做的。

第三类应用实例:

Kth Smallest Element in a Sorted Matrix

第三类变形应用示例:

Sqrt(x)

n Binary Matrix 二进制矩阵中的最短路径

4.[LeetCode] 1090. Largest Values From Labels 标签中的最大价值

5.[LeetCode] 1089. Duplicate Zeros 复写零

6.[LeetCode] 1088. Confusing Number II 易混淆数之二

7.[LeetCode] 1057. Campus Bikes 校园自行车

8.[LeetCode] 1056. Confusing Number 混淆的数字

9.[LeetCode] 1055. Shortest Way to Form String 形成字符串的最短方法

10.[LeetCode] 1081. Smallest Subsequence of Distinct Characters 不同字符的最小子序列

赞助

积分与排名

积分 - 3328052

排名 - 16

随笔分类

3D Visualization(12)

Algorithms(8)

Amazon Web Service(4)

C/C++, Java, Python(34)

CareerCup(150)

CUDA/OpenCL(1)

Digital Image Processing(3)

Entertainment(6)

GTK+/VTK/ITK/FLTK(20)

IOS(7)

这定取个博士头衔的一类，而且通常情况下都很难。因为这里住一万且找法里安的比较入小的地方使用到了子函数，并不是之前三类中简单的数字大小的比较，比如 [Split Array Largest Sum](#) 那道题中的解法一，就是根据是否能分割数组来确定下一步搜索的范围。类似的还有 [Guess Number Higher or Lower](#) 这道题，是根据给定函数 guess 的返回值情况来确定搜索的范围。对于这类题目，博主也很无奈，遇到了只能自求多福了。

第四类应用实例：

[Split Array Largest Sum](#), [Guess Number Higher or Lower](#), [Find K Closest Elements](#), [Find K-th Smallest Pair Distance](#), [Kth Smallest Number in Multiplication Table](#), [Maximum Average Subarray II](#), [Minimize Max Distance to Gas Station](#), [Swim in Rising Water](#), [Koko Eating Bananas](#), [Nth Magical Number](#)

第五类：其他（通常 target 值不固定）

有些题目不属于上述的四类，但是还是需要用到二分搜索法，比如这道 [Find Peak Element](#)，求的是数组的局部峰值。由于是求的峰值，需要跟相邻的数字比较，那么 target 就不是一个固定的值，而且这道题的一定要注意的是 right 的初始化，一定要是 `nums.size() - 1`，这是由于算出了 mid 后，`nums[mid]` 要和 `nums[mid+1]` 比较，如果 right 初始化为 `nums.size()` 的话，`mid+1` 可能会越界，从而不能找到正确的值，同时 while 循环的终止条件必须是 `left < right`，不能有等号。

类似的还有一道 [H-Index II](#)，这道题的 target 也不是一个固定值，而是 `len-mid`，这就很有意思了，跟上面的 `nums[mid+1]` 有异曲同工之妙，target 值都随着 mid 值的变化而变化，这里的 right 的初始化，一定要是 `nums.size() - 1`，而 while 循环的终止条件必须是 `left <= right`，这里又必须要有等号，是不是很头大 -.-!!!

其实仔细分析的话，可以发现其实这跟第四类还是比较相似，相似点是都很难 -.-!!!，第四类中虽然是用子函数来判断关系，但大部分时候 mid 也会作为一个参数带入子函数进行计算，这样实际上最终算出的值还是受 mid 的影响，但是 right 却可以初始化为数组长度，循环条件也可以不带等号，大家可以对比区别一下~

第五类应用实例：

[Find Peak Element](#)

[H-Index II](#)

[LintCode\(101\)](#)
[MatLab\(10\)](#)
[Maya / 3ds Max\(10\)](#)
[MySQL\(2\)](#)
[Node.js / JavaScript\(8\)](#)
[OpenCV\(37\)](#)
[Point Grey Research\(11\)](#)
[Qt\(49\)](#)
[更多](#)

随笔档案

[2021年4月\(1\)](#)
[2021年3月\(26\)](#)
[2021年2月\(29\)](#)
[2021年1月\(25\)](#)
[2020年12月\(24\)](#)
[2020年11月\(3\)](#)
[2020年10月\(1\)](#)
[2020年9月\(1\)](#)
[2020年8月\(3\)](#)
[2020年7月\(4\)](#)
[2020年6月\(4\)](#)
[2020年5月\(5\)](#)
[2020年4月\(5\)](#)
[2020年3月\(3\)](#)
[2020年2月\(5\)](#)
[2020年1月\(3\)](#)
[2019年12月\(2\)](#)
[2019年11月\(4\)](#)
[2019年10月\(9\)](#)
[2019年9月\(8\)](#)
[更多](#)

最新评论

赞助



中第一类最简单，第四类和第五类最难，遇到这类，博主也没啥好建议，多多练习吧~

如果有写的有遗漏或者错误的地方，请大家踊跃留言啊，共同进步哈~

[LeetCode All in One 题目讲解汇总\(持续更新中...\)](#)

分类: [Algorithms](#)

好文要顶

关注我

收藏该文



Grandyang

关注 - 36

粉丝 - 1275

+加关注

6

推荐

0

反对

« 上一篇: [\[LeetCode\] Split Array with Equal Sum 分割数组成和相同的子数组](#)

» 下一篇: [\[LeetCode\] Binary Tree Longest Consecutive Sequence II 二叉树最长连续序列之二](#)

posted @ 2017-05-15 08:40 Grandyang 阅读(36248) 评论(32) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论, 立即 [登录](#) 或者 [逛逛](#) [博客园首页](#)

【推荐】开发者藏经阁, 160本电子书免费下载! 阿里工程师实践精华

【推荐】免费下载! Apache Flink 系列电子书: 开源大数据前瞻与应用实战

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!

【推荐】创建 AR 世界, 周周赢 HUAWEI WATCH FIT 等好礼

【推荐】限时秒杀! 国云大数据魔镜, 企业级云分析平台

转有序数组的最小值

@hello_world00 因为这道题比较特殊, 目标值 target 并不是一个确定的值, 而是会变的, 所以博主总结的规律在这里并不是适用。...

--Grandyang

2. Re:[LeetCode] 560. Subarray Sum Equals K 子数组和为K

这题好复杂, 感觉很难理解这个过程。我觉得首先是将 $sum[j, i] == k$ 这个带有未知的转化为 $sum[j] - sum[i-1] == k$, $sum[i]$ 可以通过计算累加和得到, 所以算是已知的, 后续的...

--夜半读核

3. Re:[LeetCode] 29. Divide Two Integers 两数相除

那个while循环为啥这么干呢?

--夜半读核

4. Re:[LeetCode] 233. Number of Digit One 数字1的个数

@苏芒 楼主, 如果三位数是200~209的话, 这10个数的区间不是才1个1吗? 考察的范围是[200,299], 这个区间1的个数是20。...

--swords poet

5. Re:[LeetCode] 153. Find Minimum in Rotated Sorted Array 寻找旋转有序数组的最小值

@Grandyang 好久没来了, 最近重新捡起来刷题 又回到这个问题。其实这个题跟很多二分法的模板是不一样的 若 right 初始化为 nu

赞助

[首页](#)[新闻](#)[博文](#)[专区](#)[闪存](#)[班级](#)[代码改变世界](#)[注册](#)[登录](#)

华人家长请注意

还有价值588元的免费课程等你领取，快来体验吧



悟空中文

园子动态:

- 发起一个开源项目: 博客引擎 fluss
- 云计算之路-新篇章-出海记: 开篇
- 博客园2005年6月1日首页截图

最新新闻:

- 雷军造车 小米手机的三个错误不可再犯
- 高德地图发布2021清明节出行预测: 全国高速拥堵预计较去年同期上涨47%
- 拼多多腾讯, 互攻对方腹地
- B站回港二次上市, 能再次挟年轻人“芜湖”起飞吗?
- 高管离职、亏损百亿、合规难题: 首汽约车的努力配不上野心

[» 更多新闻...](#)

--hello_world00

阅读排行榜

1. LeetCode All in One 题目讲解汇总(持续更新中...)(1042595)
2. [LeetCode] 1. Two Sum 两数之和(138557)
3. [LeetCode] 15. 3Sum 三数之和(82288)
4. Manacher's Algorithm 马拉车算法(76127)
5. [LeetCode] 4. Median of Two Sorted Arrays 两个有序数组的中位数(66197)
6. [LeetCode] 5. Longest Palindromic Substring 最长回文子串(65840)
7. [LeetCode] 3. Longest Substring Without Repeating Characters 最长无重复字符的子串(57848)
8. Qt qDebug() 的使用方法(53771)
9. [LeetCode] 10. Regular Expression Matching 正则表达式匹配(51912)
10. [LeetCode] 2. Add Two Numbers 两个数字相加(45749)

评论排行榜

1. LeetCode All in One 题目讲解汇总(持续更新中...)(155)
2. [LeetCode] 4. Median of Two Sorted Arrays 两个有序数组的中位数(38)

赞助



eft, Center and Right) 大黄蜂立体相机保存捕获的视频到左中右三个不同的文件(35)

4. LeetCode Binary Search Summary 二分搜索法小结(32)

5. [LeetCode] 1. Two Sum 两数之和(32)

推荐排行榜

- 1. LeetCode All in One 题目讲解汇总(持续更新中...)(111)
- 2. Manacher's Algorithm 马拉车算法(23)
- 3. [LeetCode] 1. Two Sum 两数之和(12)
- 4. Reward List 赏金列表(8)
- 5. [LeetCode] 407. Trapping Rain Water II 收集雨水之二(7)

赞助