

Fork me on GitHub

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 - 1429 文章 - 1 评论 - 4

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，并留言跟博主联系)

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 - 919】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【Imm】推送iOS APP [Leetcode Meet Me]
- 回复任意文字跟博主留言交流^_^

昵称：Grandyang

园龄：7年10个月

粉丝：1112

关注：36

+加关注

赞助

2020年2月						
<	一	二	三	四	五	六
26	27	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
1	2	3	4	5	6	7

搜索

找找看

这是比较常见的一类，因为我们要查找的目标值不一定会在数组中出现，也有可能是跟目标值相等的数在数组中并不唯一，而是有多个，那么这种情况下 `nums[mid] == target` 这条判断语句就没有必要存在。比如在数组 `[2, 4, 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

最新随笔

- 1.[LeetCode] 932. Beautiful Array 漂亮数组
- 2.[LeetCode] 931. Minimum Falling Path Sum 下降路径最小和
- 3.[LeetCode] 930. Binary Subarrays With Sum 二元子数组之和
- 4.[LeetCode] 929. Unique Email Addresses 独特的邮件地址
- 5.[LeetCode] 928. Minimize Malware Spread II 最大程度上减少恶意软件的传播之二
- 6.[LeetCode] 927. Three Equal Parts 三个相等的部分
- 7.[LeetCode] 926. Flip String to Monotone Increasing 翻转字符串到单调递增
- 8.[LeetCode] 925. Long Pressed Name 长按键入的名字
- 9.[LeetCode] 924. Minimize Malware Spread 最大程度上减少恶意软件的传播
- 10.[LeetCode] 923. 3Sum With Multiplicity 三数之和的多种情况

积分与排名

积分 - 2949993
排名 - 16

随笔分类

3D Visualization(12)
Algorithms(8)
C/C++, Java, Python(34)
CareerCup(150)

赞助

这一类也比较常见，尤其是查找第一个大于目标值的数，在 C++ 的 STL 也有专门的函数 `upper_bound`，这里跟上面的那种情况的写法上很相似，只需要添加一个等号，将之前的 `nums[mid] < target` 变成 `nums[mid] <= 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)

CUDA/OpenCL(1)
Digital Image Processing(3)
Entertainment(6)
GTK+/VTK/ITK/FLTK(20)
IOS(7)
LaTeX(3)
LeetCode(903)
LintCode(101)
MatLab(10)
Maya / 3ds Max(10)
MySQL(2)
Node.js / JavaScript(8)
OpenCV(37)
Point Grey Research(11)
Qt(49)
Software/Tools(3)
Useful Links(34)

随笔档案

2020年2月(2)
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)
2019年8月(10)
2019年7月(8)
2019年6月(13)
2019年5月(16)
2019年4月(14)
2019年3月(10)
2019年2月(12)
2019年1月(10)
2018年12月(8)
2018年11月(19)
2018年10月(9)

赞助

第四类：用子函数当作判断关系（通常由 mid 计算得出）

这是最令博主头疼的一类，而且通常情况下都很难。因为这里在二分查找法重要的比较大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](#)

2018年9月(6)
2018年8月(8)
2018年7月(11)
2018年6月(10)
2018年5月(11)
2018年4月(13)
2018年3月(15)
2018年2月(14)
2018年1月(17)
2017年12月(12)
2017年11月(16)
2017年10月(29)
2017年9月(21)
2017年8月(10)
2017年7月(12)
2017年6月(21)
2017年5月(26)
2017年4月(18)
2017年3月(22)
2017年2月(23)
2017年1月(13)
2016年12月(26)
2016年11月(30)
2016年10月(30)
2016年9月(24)
2016年8月(40)
2016年7月(31)
2016年6月(33)
2016年5月(30)
2016年4月(70)
2016年3月(32)
2016年2月(32)
2016年1月(25)
2015年12月(3)
2015年11月(36)
2015年10月(43)
2015年9月(51)

赞助

综上所述，博主大致将二分搜索法的应用场景分成了主要这五类，其中第二类和第三类还有各自的扩展。根据目前博主的经验来看，第二类和第三类的应用场景最多，也是最重要的两类。第一类，第四类，和第五类较少，其中第一类最简单，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 - 1112

+加关注

4

0

推荐

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(23295) 评论(31) 编辑 收藏

评论列表

#1楼 2017-06-24 15:31 icewating

顶一个。博主是否已经工作了，仍然在更新着LeetCode。

支持(0) 反对(0)

#2楼 [楼主] 2017-06-25 05:29 Grandyang

@ icewating
燃鹅并没有，博主在实习哈~

支持(0) 反对(0)

#3楼 2018-01-03 10:55 DMU_LZH

我刷leetcode主要就是在百度大神你的解析了，明年秋季找工作，现在还没有找实习。能认识下呗大神，我在大连

支持(0) 反对(0)

- 2015年8月(46)
- 2015年7月(45)
- 2015年6月(29)
- 2015年5月(28)
- 2015年4月(42)
- 2015年3月(55)
- 2015年2月(61)
- 2015年1月(27)
- 2014年12月(8)
- 2014年11月(27)
- 2014年10月(35)
- 2014年9月(5)

最新评论

1. Re:[LeetCode] 272. Closest Binary Search Tree Value II 最近的二分搜索树的值之二
vector删掉第一个元素不是O(1), 如果要这样做，用list或者dq。
--kingming0811

2. Re:[LeetCode] 312. Burst Balloons 打气球游戏
@ hao1987我感觉这里意思是有点问题，其实这里事最后打爆#2...
--adnii8130

3. Re:[LeetCode] 96. Unique Binary Search Trees 独一无二的二叉搜索树
class Solution { public int numTrees(int n) { int[] dp = new int[n + 1]; Arrays.fill(dp, -1); helper...
--mikecheng

4. Re:[LeetCode] Longest Absolute File Path 最长的绝对文件路径

赞

#4楼 [楼主] 2018-01-03 11:12 Grandyang

@ DMU_LZH

可在微信打赏留言中留下你的微信号;)

支持(0) 反对(0)

#5楼 2018-01-04 04:05 edyyy

博主网红17夏在哪里实习?

支持(0) 反对(0)

#6楼 [楼主] 2018-01-04 07:46 Grandyang

@ edyyy

博主跟一家经纪公司签约, 在进行网红培训。。

支持(0) 反对(0)

#7楼 2018-01-04 13:57 edyyy

@ Grandyang

狗家还训网红? 哈哈

支持(0) 反对(0)

#8楼 2018-04-23 16:52 wtybill

博主能问下你关于第二三类的测试例子leetcode好像没有欸。。不知道博主知道有啥oj可以直接测试么

支持(0) 反对(0)

#9楼 [楼主] 2018-05-02 23:26 Grandyang

@ wtybill

博主不是在各类下方都列举了几个应用实例么

支持(0) 反对(0)

#10楼 2018-05-02 23:32 wtybill

@ Grandyang

嗯嗯, 我找到了, 后面Search For A range这个题可以变相测试不严格上下界二分, 谢谢博主T_T

支持(0) 反对(0)

#11楼 2018-05-11 14:07 ddlyq

@ wtybill

方丈是你吗? -_-

支持(0) 反对(0)

#12楼 2018-05-11 22:43 edyyy

@ leo2019不会有问题的, 解法二是按照每一层来处理的。处理完一层的所有文件, 处理下一层的时候, 会更新directory的长度的。...

--mikecheng

5. Re:[LeetCode] 100. Same Tree 判断相同树

@ Cass998已添加, 多谢指出~ ...

--Grandyang

阅读排行榜

1. LeetCode All in One 题目讲解汇总(持续更新中...)(767649)
2. [LeetCode] 1. Two Sum 两数之和(115269)
3. [LeetCode] 15. 3Sum 三数之和(67741)
4. Manacher's Algorithm 马拉车算法(62096)
5. [LeetCode] 4. Median of Two Sorted Arrays 两个有序数组的中位数(53635)
6. [LeetCode] 5. Longest Palindromic Substring 最长回文子串(53366)
7. [LeetCode] 3. Longest Substring Without Repeating Characters 最长无重复字符的子串(47271)
8. [LeetCode] 10. Regular Expression Matching 正则表达式匹配(43244)
9. Qt qDebug() 的使用方法(38917)
10. [LeetCode] 2. Add Two Numbers 两个数字相加(36849)

评论排行榜

赞
助

1. LeetCode All in One 题目讲解汇总(持续更新中...)(153)
2. [FlyCapture2] Bumblebee XB3 Save Images to Three AVI Files (Left, 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)
3. [LeetCode] 4. Median of Two Sorted Arrays 两个有序数组的中位数(35)
4. [LeetCode] 1. Two Sum 两数之和(32)
5. LeetCode Binary Search Summary 二分搜索法小结(31)

推荐排行榜

1. LeetCode All in One 题目讲解汇总(持续更新中...)(90)
2. Manacher's Algorithm 马拉车算法(17)
3. Reward List 赏金列表(8)
4. [LeetCode] 1. Two Sum 两数之和(7)
5. [LeetCode] 94. Binary Tree Inorder Traversal 二叉树的中序遍历(6)
6. [LeetCode] 15. 3Sum 三数之和(6)
7. [LeetCode] 289. Game of Life 生命游戏(6)
8. [LeetCode] 407. Trapping Rain Water II 收集雨水之二(5)
9. LeetCode Binary Search Summary 二分搜索法小结(4)
10. [LeetCode] 287. Find the Duplicate Number 寻找重复数(4)

赞助

@ddlyq
@wtybill
你们是二师兄吗?

支持(0) 反对(0)

#13楼 [楼主] 2018-05-11 23:10 Grandyang

@ ddlyq
敢问大师法号?

支持(0) 反对(0)

#14楼 2018-06-03 16:38 ddlyq

博主，第三类问题的变种的描述是不是有问题啊？我觉得应该是最后一个不大于才对吧。[1,4,5,6,7,9]，target = 8, return right -1 之后，返回的是4，也就是7,7是最后一个不大于8的数吧。难道是我理解错了吗？

支持(0) 反对(0)

#15楼 [楼主] 2018-06-04 06:06 Grandyang

@ ddlyq
嗯嗯，已改正，多谢指出~

支持(0) 反对(0)

#16楼 2018-06-19 11:24 Omfg

第二类 返回left right 都可以吧，因为跳出while的时候left==right。
第一个小于目标值的数 不就是第一个吗？应该是最后一个小于目标值的数吧

支持(0) 反对(0)

#17楼 2018-06-19 11:39 Omfg

另外 while 条件要不要=，我觉得跟 left right初值没关系，跟每次移动时, 是否包含mid 有关系。

支持(0) 反对(0)

#18楼 2018-06-19 19:38 wtybill

@ ddlyq
@edyyy
你们看我屌么。。

支持(0) 反对(0)

#19楼 [楼主] 2018-06-20 23:26 Grandyang

@ Omfg
嗯嗯，是的，返回left和right都行。
其实博主当时是从右往左来看的，所以写的是第一个。不过还是从左往右吧，已改正，多谢指出~

支持(0) 反对(0)

#20楼 2018-07-26 02:52 Lisa114

感觉arranging coins应该是第三类变形

支持(0) 反对(0)

#21楼 [楼主] 2018-08-01 12:45 Grandyang

@ Omfg

我觉得应该是有关系的，比如只有一个数字的话，如果使用`right=nums.size()-1` 那么left和right都是0，此时如果while中不加等号，那么不会进入循环，如果此时要找数组中第一个大于目标值的数时，而数组中那个唯一的数字正好是目标值的时候，就会出错~

支持(0) 反对(0)

#22楼 2018-11-06 23:01 Lisa114

博主，能详细说一下第一类return中的left, right, 或right - 1都代表啥吗？多谢~

支持(0) 反对(0)

#23楼 [楼主] 2018-11-07 23:57 Grandyang

@ Lisa114

left和right就是搜索范围的左右边界，根据while中的判定条件的不同，最后的值也不同。如果是while(left < right)，最后left和right会相同，如果while(left <= right)，最后left和right值会不同。而right-1一般是根据要求的不同，要查找的位置会在终止值的前一个位置~

支持(0) 反对(0)

#24楼 2018-11-08 02:19 Lisa114

@Grandyang

多谢，多谢~

支持(0) 反对(0)

#25楼 2019-01-19 00:12 梅斯特菠萝

想问问呢博主大大

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。这是为什么？

支持(0) 反对(0)

#26楼 [楼主] 2019-01-21 11:21 Grandyang

@ 梅斯特菠萝

当只有一个数字的时候，如果right初始化为 `nums.size() - 1`，那么此时left和right都是0了，如果还使用 `left < right`，那么循环根本进不去，如果是第三类，查找第一个大于目标的数，而恰好数组中唯一的那个数小于目标值，我们还是会返回那个数，因为right此时是0，就会出错~

支持(0) 反对(0)

#27楼 2019-01-24 22:41 hadeser

楼主你好，使用第一种方式中 right初始化为`nums.size()`。right赋值如果采用 `mid - 1` 的话，会找不到把（为什么减一或者不减都可以呢）？如果`nums={1, 2,3},target = 1`。还是我哪里算错了吗

支持(0) 反对(0)

赞
助

#28楼 [楼主] 2019-01-25 05:41 Grandyang

@ 不靠谱

嗯嗯，已改正，多谢指出~

支持(0) 反对(0)

#29楼 2019-05-29 22:38 动手想明白再编码

博主可不可以解释一下，为什么在第五类中while循环中left和right比大小有时候需要等号，有时候不能有等号，这该如何判断什么时候该有什么时候不该有呢？

支持(0) 反对(0)

#30楼 [楼主] 2019-10-21 07:54 Grandyang

@ 动手想明白再编码

这个感觉没有规律可循，因为 target 不是固定值，而且变换方式也不同，只能具体题目具体分析了。

支持(0) 反对(0)

#31楼 [楼主] 2019-10-21 07:54 Grandyang

@ hadeser

为啥找不到呢？

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 **注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，访问 [网站首页](#)。**

【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】开发者上云福利，腾讯云1核4G云服务器11元/月起

【推荐】开年盛典，百度智能云1核1G云服务器84元/年

【推荐】精品问答：大数据常见问题之 flink 五十问

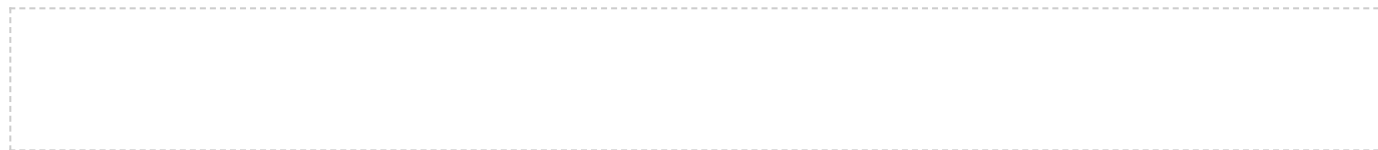
【推荐】阿里技术3年大合集免费电子书一键下载

赞助

**相关博文:**

- Leetcode中几道二分查找(Binary Search)的算法题总结
 - 二分法 Binary Search
 - LeetCodeMonotoneStackSummary单调栈小结
 - [LeetCode] K Empty Slots K个空槽
 - [LeetCode] Find K-th Smallest Pair Distance 找第K小的数对儿距离
- » 更多推荐...

免费下载《阿里工程师的自我修养》

**最新 IT 新闻:**

- Windows 10 本月例行更新存在 bug
 - 小米钱包发布两大新功能：智能选卡和非NFC机型Mi Pay扫码支付
 - 爱的起源与演化
 - 马云和钟南山团队合作研发药物，前者提供资金和AI算力支持
 - 中芯的喜与忧：第二代FinFET逼近台积电7nm，惟疫情将延后8寸进度
- » 更多新闻...

赞助

历史上的今天:

2016-05-15 [LintCode] Maximal Rectangle 最大矩形

2015-05-15 Use auto keyword in C++11

Copyright © 2020 Grandyang
Powered by .NET Core 3.1.1 on Linux

赞助