

# 线段树

游坦之老师

扫描二维码关注微信/小程序，获取最新面试题及权威解答



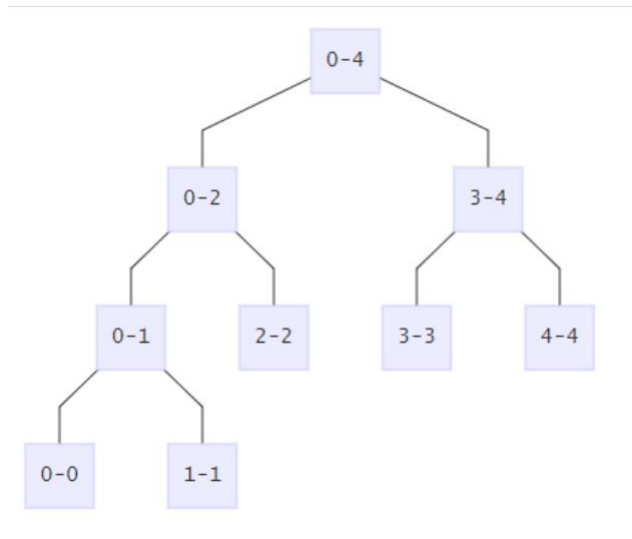
微信扫一扫，使用小程序



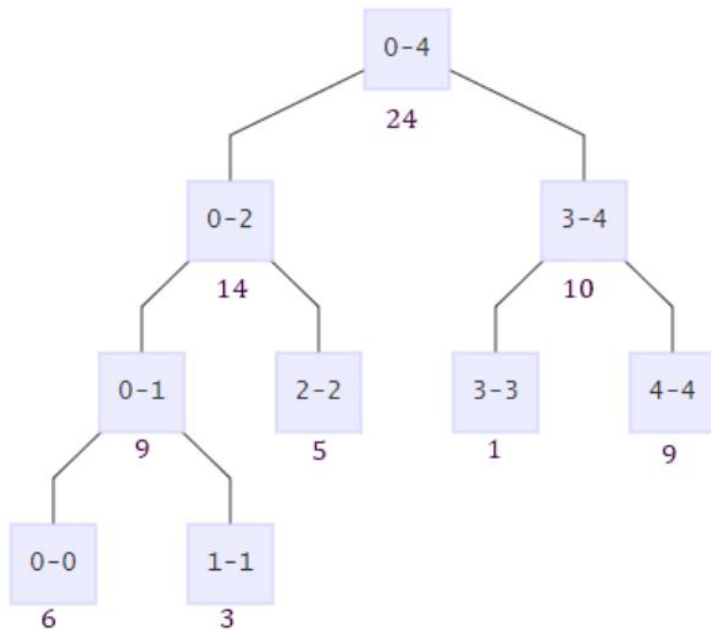
微信扫一扫，关注微信公众号

# 1、什么是线段树

线段树就是一个二叉树，二叉树中的每个节点代表一个区间。



# 1、什么是线段树——range-sum问题



# 1、什么是线段树——课程内容

---

线段树的作用、可以解决的问题

线段树的结构 线段树的构建

线段树的修改

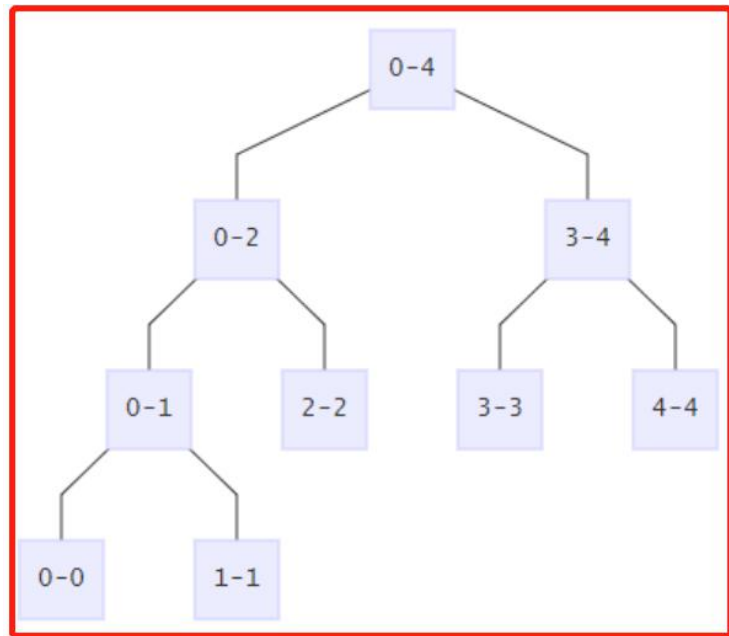
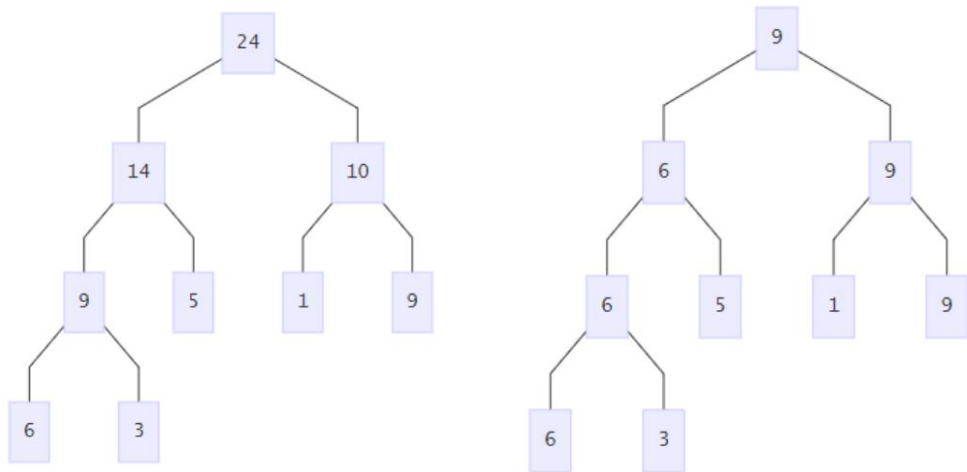
线段树的查询

## 2、线段树有什么用

线段树主要问题对象是区间。

求解区间和、区间最值以及其它区间上的问题。

根据问题的需要定义node的属性。



### 3、线段树适用题型

我们经常会遇到需要维护一个序列的问题：

给定一个整数序列，每次操作会修改序列某个位置上的数，或是询问你序列中某个区间内所有数的和。

暴力：修改时间 $O(1)$  查询时间 $O(n)$  空间 $O(1)$

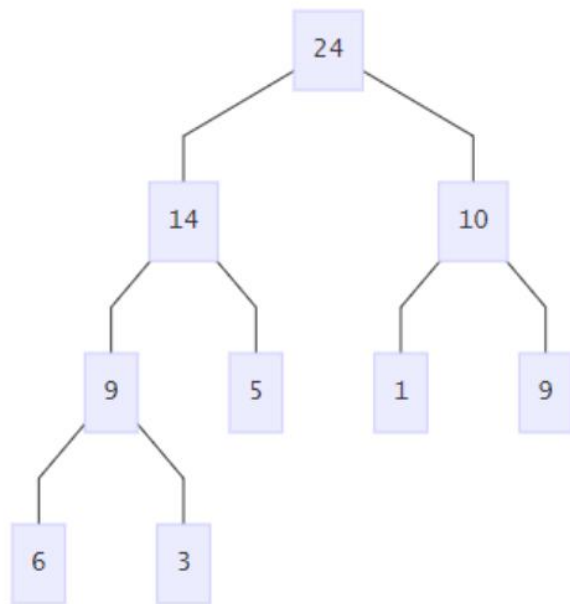
前缀和数组：修改时间 $O(n)$  查询时间 $O(1)$  空间 $O(n)$

### 3、线段树适用题型

在序列上单点/区间修改，然后对区间进行查询——线段树

修改和查询的时间复杂度都是 $O(\log n)$

空间复杂度是 $O(n)$



### 3、线段树适用题型

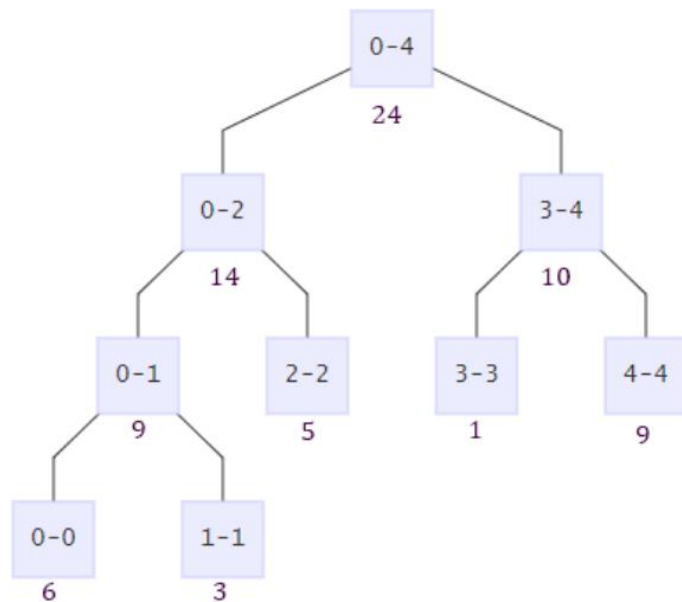
---

如果仅涉及区间上的查询，而不涉及修改，那么用前缀和即可。



## 4、线段树三个基本操作：构建修改查询

构建：自上向下，将大区间一切两半，递归调用。

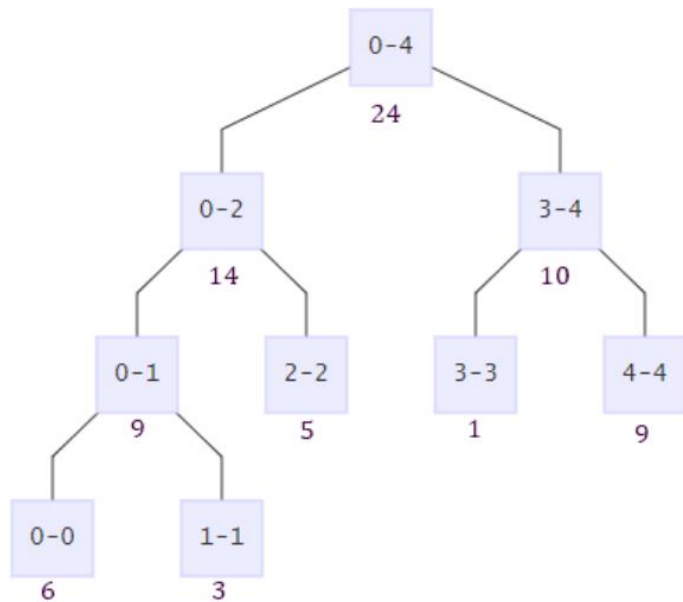


## 4、线段树三个基本操作：构建修改查询

修改：递归调用，一路向下然后触底反弹。

一路向下是为了找到最小区间。

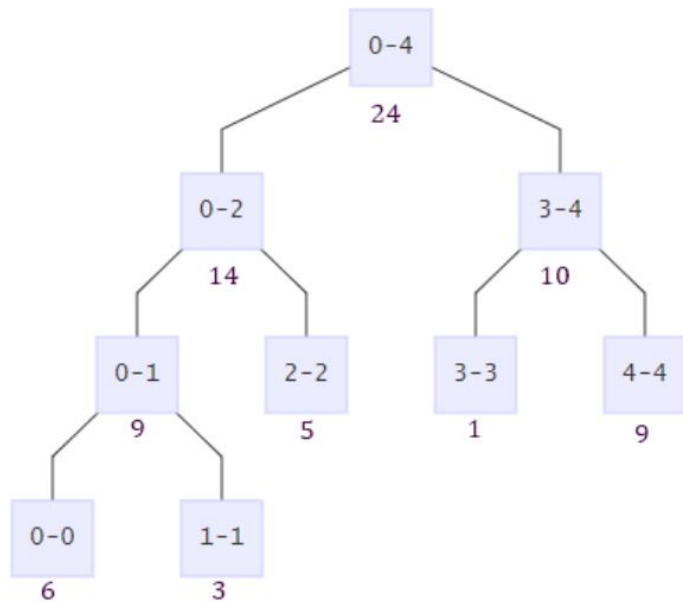
触底反弹的时候才去修改node。



## 4、线段树三个基本操作：构建修改查询



查询：0-3 and 2-3



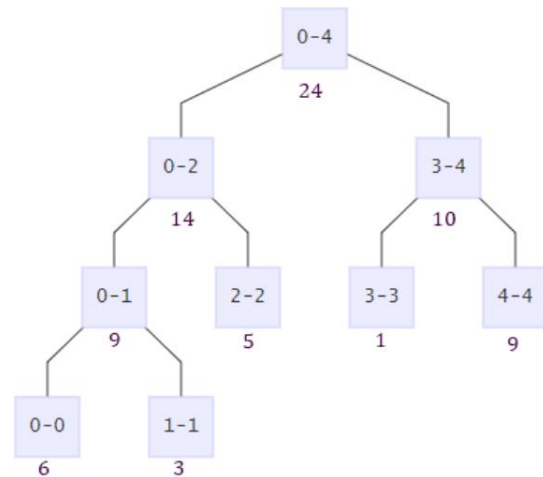
## 5、线段树的样子

性质:

除表示单点的一个节点是叶子结点外，其它每一个表示区间的节点都有两颗子树。

每一个节点分出了左右节点的区间长度为父亲节点长度的一半（左边向上取整，右边向下取整）。

每一个节点存储的值都是左右节点进行对应运算得出的。这个运算是根据要求而定的。如：求和的是和，求最大值的是max。



## 5、线段树的样子

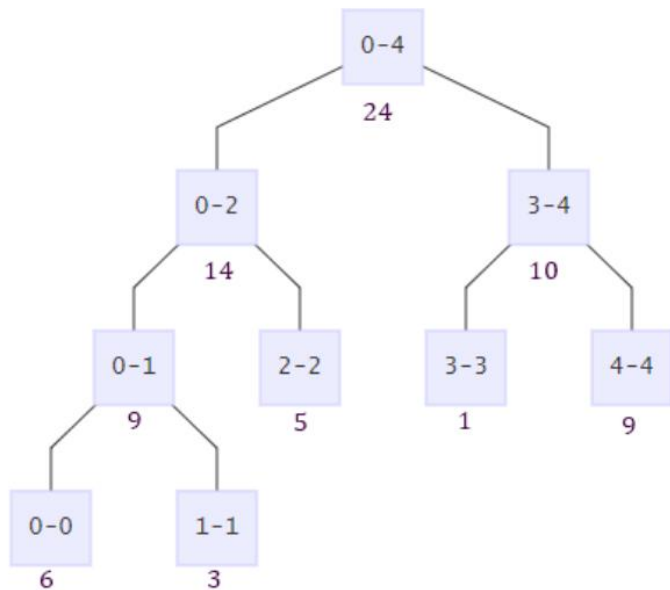
结点定义:

左端点、右端点

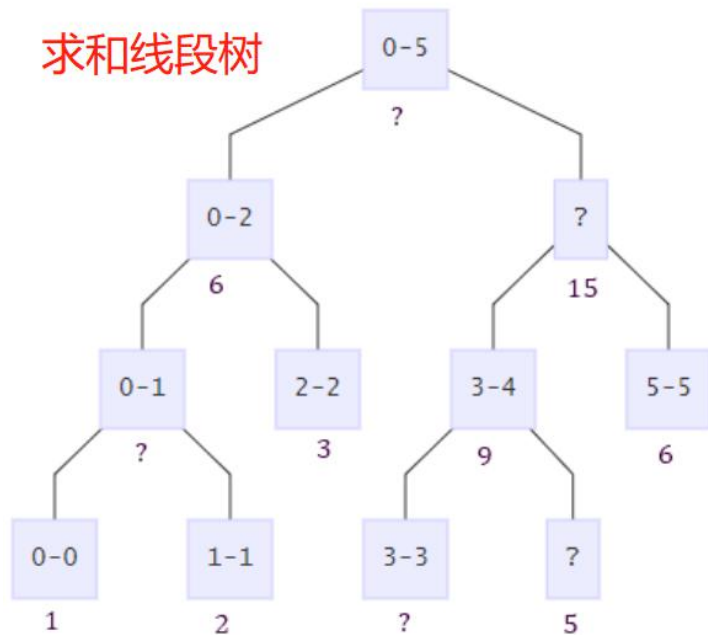
左孩子、右孩子

val (sum、max)

```
public class SegmentTreeNode {  
    public int start, end, max;  
    public SegmentTreeNode left, right;  
    public SegmentTreeNode(int start, int end, int max) {  
        this.start = start;  
        this.end = end;  
        this.max = max;  
        this.left = this.right = null;  
    }  
}
```



## 5、线段树的样子——大家来填空

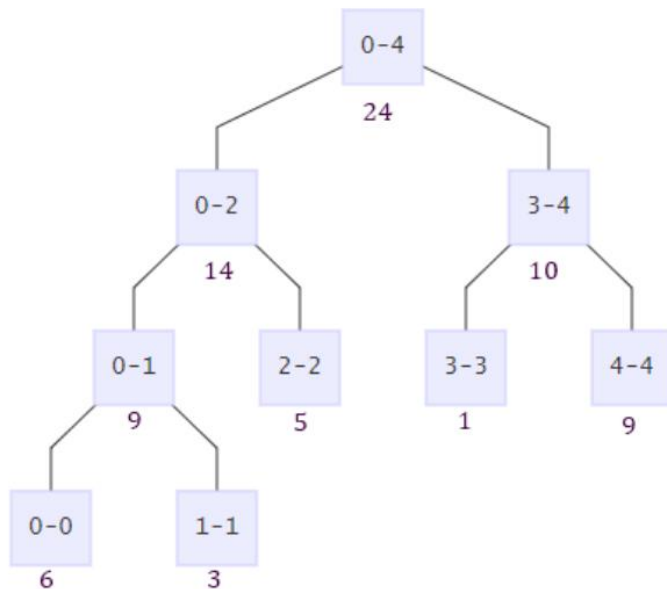


## 6、分析线段树构建、查询、修改时间复杂度

构建  $O(n)$

查询  $O(\log n)$

修改  $O(\log n)$




## 7、如何去根据问题去构建线段树

建一个区间为0-5的线段树:



## 7、如何去根据问题去构建线段树

```
public SegmentTreeNode build(int start, int end) {  
    if(start > end) {  
        return null;  
    }  
    if (start == end) {  
        return new SegmentTreeNode(start, end);  
    }  
    SegmentTreeNode root = new SegmentTreeNode(start, end);  
  
    if(start != end) {  
        int mid = (start + end) / 2;   
        root.left = build(start, mid);  
        root.right = build(mid+1, end);  
    }  
    return root;  
}
```

## 7、如何去根据问题去构建线段树

针对[9,7,4,5,2,62]构造max-range线段树:

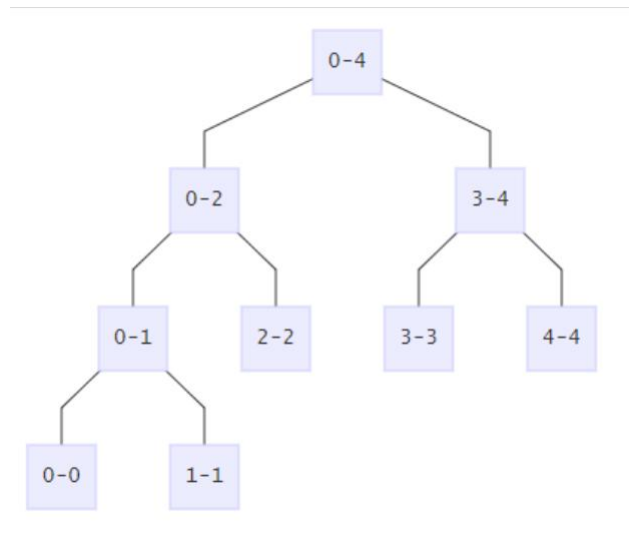
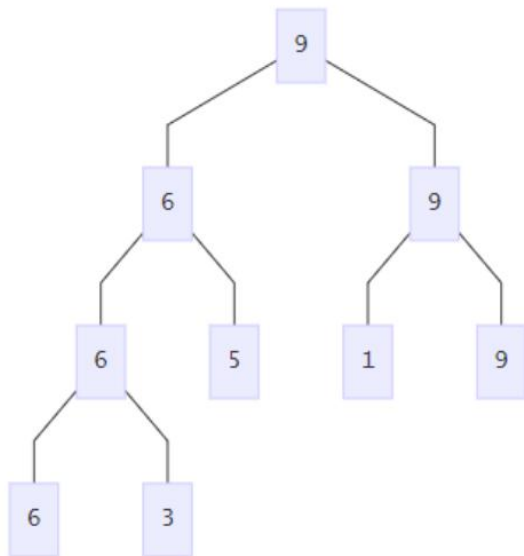
## 7、如何去根据问题去构建线段树

```
public SegmentTreeNode buildTree(int start, int end, int[] A) {
    if (start > end)
        return null;

    if (start == end) {
        return new SegmentTreeNode(start, end, A[start]);
    }
    SegmentTreeNode node = new SegmentTreeNode(start, end,
A[start]);
    if(start != end){
        int mid = (start + end) / 2;
        node.left = this.buildTree(start, mid, A);
        node.right = this.buildTree(mid + 1, end, A)
    }
    if (node.left != null && node.left.max > node.max)
        node.max = node.left.max;
    if (node.right != null && node.right.max > node.max)
        node.max = node.right.max;
    return node;
}
```

如果是range-sum线段树呢？

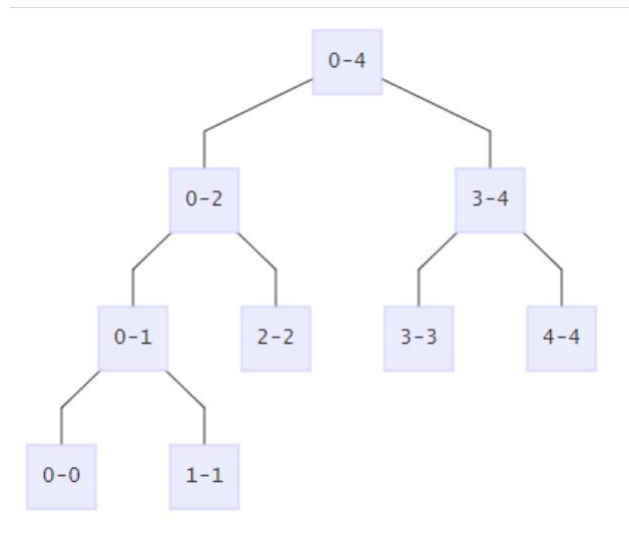
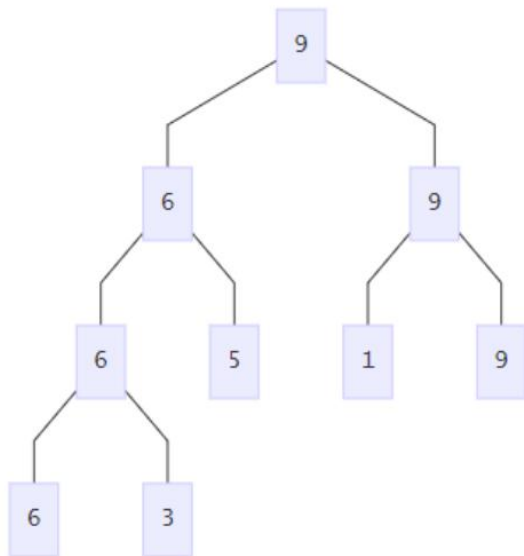
## 8、线段树的修改



## 8、线段树的修改

```
public void modify(SegmentTreeNode root, int index, int value) {  
    if(root.start == index && root.end == index) {  
        root.max = value;  
        return;  
    }  
  
    int mid = (root.start + root.end) / 2;  
    if(root.start <= index && index <= mid) {  
        modify(root.left, index, value);  
    }  
  
    if(mid < index && index <= root.end) {  
        modify(root.right, index, value);  
    }  
  
    root.max = Math.max(root.left.max, root.right.max);  
}
```

## 9、线段树的查询



## 9、线段树的查询



```
public int query(SegmentTreeNode root, int start, int end) {
    if(start == root.start && root.end == end) {
        return root.max;
    }

    int mid = (root.start + root.end)/2;
    int leftmax = Integer.MIN_VALUE, rightmax = Integer.MIN_VALUE;

    if(start <= mid) {
        if(mid < end) {
            leftmax = query(root.left, start, mid);
        } else {
            leftmax = query(root.left, start, end);
        }
    }

    if(mid < end) {
        if(start <= mid) {
            rightmax = query(root.right, mid+1, end);
        } else {
            rightmax = query(root.right, start, end);
        }
    }

    return Math.max(leftmax, rightmax);
}
```



扫描二维码关注微信/微博  
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏: <http://zhuanlan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: [www.jiuzhang.com](http://www.jiuzhang.com)