

单调队列与单调栈

胡船长

初航我带你，远航靠自己

本章题目

- 1-校招. HZOJ-271: 滑动窗口
- 2-校招. HZOJ-270: 最大子序和
- 3-校招. Leetcode-Offer59: 队列的最大值
- 4-校招. Leetcode-1438: 绝对差不超过限制的最长连续子数组
- 5-校招. HZOJ-264: 最大矩形面积
- 6-竞赛. Leetcode-42: 接雨水
- 7-竞赛. Leetcode-862: 和至少为 K 的最短子数组
- 8-竞赛. HZOJ-372: 双生序列

本期内容

一. 单调队列

1. 单调队列的作用
2. 单调队列代码演示

二. 单调栈

1. 单调栈的作用
2. 单调栈代码演示

一. 单调队列

单 调 队 列

问题引入：

$\text{RMQ}(x, y)$ 就是询问数组 $[x, y]$ 区间内部的最小值

例如： $\text{RMQ}(0, 3) = 1$ ， $\text{RMQ}(3, 7) = 2$

现在，固定询问区间的尾部，例如： $\text{RMQ}(x, 7)$

请思考，如下序列中最少记录几个元素，就可以满足 $\text{RMQ}(x, 7)$ 的任何需求

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

单调队列

问题引入：

$\text{RMQ}(x, y)$ 就是询问数组 $[x, y]$ 区间内部的最小值

例如： $\text{RMQ}(0, 3) = 1$ ， $\text{RMQ}(3, 7) = 2$

最少记录如图4个蓝色元素，即可满足 $\text{RMQ}(x, 7)$ 的所有需求

注意：蓝色部分的元素，构成了一个单调递增的序列

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

单 调 队 列

结论：

维护一个以 j 位置为结尾的单调递增序列

就可以维护 $RMQ(i, j)$ 的答案，即以 j 为结尾的任意区间最小值

再加上区间长度限制，这就是【单调队列】了

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

单调队列-维护过程

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

0	1	2	3	4	5	6	7
1	4						

单调队列-维护过程

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

0	1	2	3	4	5	6	7
1	4	5					

单调队列-维护过程

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

0	1	2	3	4	5	6	7
	2						

单调队列-维护过程

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

0	1	2	3	4	5	6	7
	2	9					

单调队列-维护过程

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

0	1	2	3	4	5	6	7
	2	8					

单调队列-维护过程

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

0	1	2	3	4	5	6	7
		8	12				

单调队列

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

入队操作:

队尾入队, 会把之前破坏单调性的元素都从队尾移出 (维护单调性)

出队操作:

如果队首元素超出区间范围, 就将元素从队首出队

元素性质:

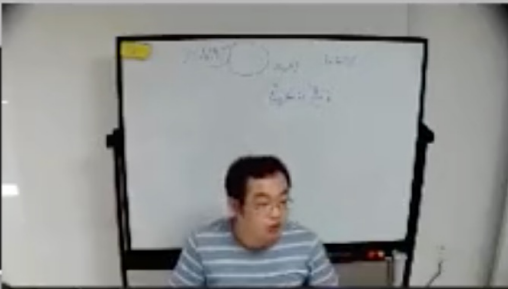
队首元素, 永远是当前维护区间的 (最大/最小) 值

序列中的每一个元素, 在依次入队的过程中, 每个元素都『黄』过

1. vim

vim %1 bash %2 bash %3

```
39 }
40
41 Node *insert_maintain(Node *root) {
42     if (!hasRedChild(root)) return root;
43     if (root->lchild->color == RED && root->rchild->color == RED, {
44         if (!hasRedChild(root->lchild) && !hasRedChild(root->rchild)) return root;
45         root->color = RED;
46         root->lchild->color = root->rchild->color = BLACK;
47         return root;
48     }
49     if (root->lchild->color == RED) {
50         if (!hasRedChild(root->lchild)) return root;
51
52
53     } else {
54         if (!hasRedChild(root->rchild)) return root;
55
56     }
57
58 }
```



单调队列：代码演示

59

60

```
61 Node *__insert(Node *root, int key) {
62     if (root == NIL) return getNewNode(key);
```

<-6班 资料 /X.现场撸代码 /15.RBT.cpp [FORMAT=unix] [TYPE=CPP] [POS=54,30][62%] 21/09/19 - 20:21

二. 单调栈

单调队列

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

入队操作:

队尾入队, 会把之前破坏单调性的元素都从队尾移出 (维护单调性)

出队操作:

如果队首元素超出区间范围, 就将元素从队首出队

元素性质:

队首元素, 永远是当前维护区间的 (最大/最小) 值

序列中的每一个元素, 在依次入队的过程中, 每个元素都『黄』过

单 调 栈

问题引入：

给一个序列，求序列中，每个元素左侧，第一个小于它的元素

观察单调队列的逻辑模型，每个黄色元素左侧第一个小于它的元素，是前一个黄色元素，根据入队列过程中，每一个元素都『黄』过，那么将所有元素依次入队，当前元素在队列中的前一个元素，即是问题所求。

这种不从头部出的结构，我们叫他【单调栈】

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

单 调 栈

所有被我打动了的，是什么？

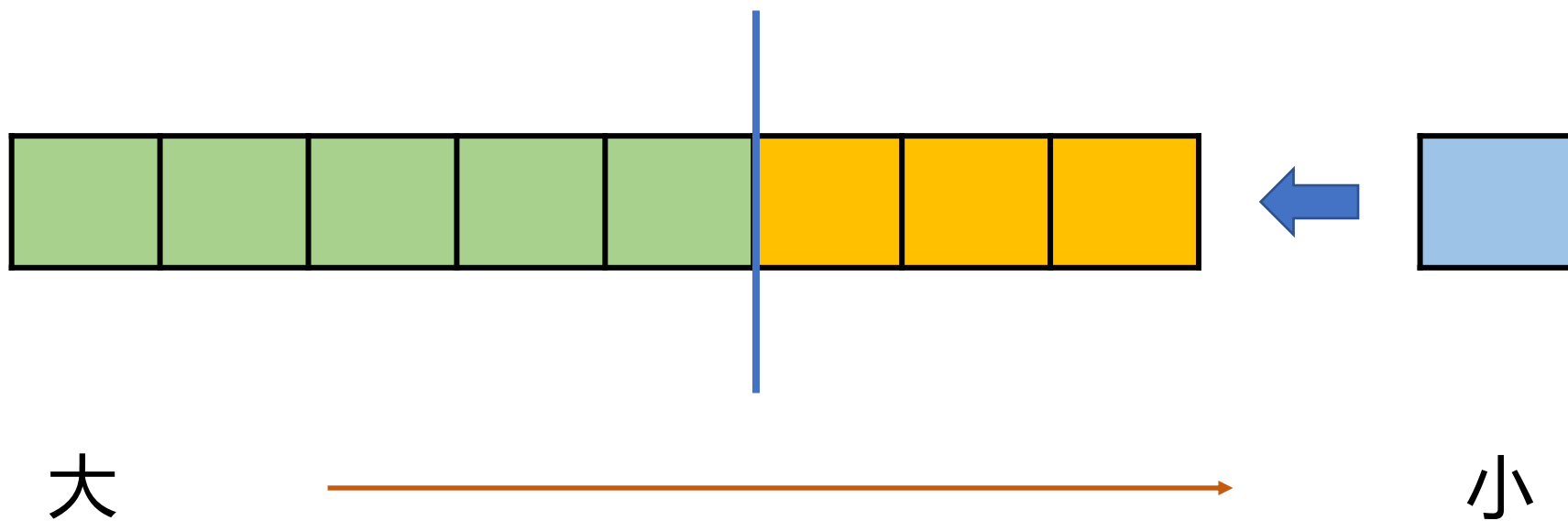
那个我打动不了的，是什么？

单 调 栈

所有被我打动了的，我是他们的男神！

那个我打动不了的，她是我生命中的女神！

单调栈



单 调 栈

单调递增：最近小于关系

单调递减：最近大于关系

总 结

单调队列： 擅长维护区间【最大/最小】值，最小值对应单调递增队列

单调栈： 擅长维护最近【大于/小于】关系

从左侧先入栈，就是维护左侧最近关系

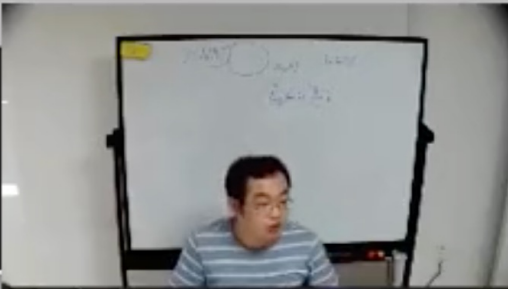
从右侧先入栈，就是维护右侧最近关系

0	1	2	3	4	5	6	7
3	1	4	5	2	9	8	12

1. vim

vim %1 bash %2 bash %3

```
39 }
40
41 Node *insert_maintain(Node *root) {
42     if (!hasRedChild(root)) return root;
43     if (root->lchild->color == RED && root->rchild->color == RED, {
44         if (!hasRedChild(root->lchild) && !hasRedChild(root->rchild)) return root;
45         root->color = RED;
46         root->lchild->color = root->rchild->color = BLACK;
47         return root;
48     }
49     if (root->lchild->color == RED) {
50         if (!hasRedChild(root->lchild)) return root;
51
52
53     } else {
54         if (!hasRedChild(root->rchild)) return root;
55
56     }
57
58 }
```

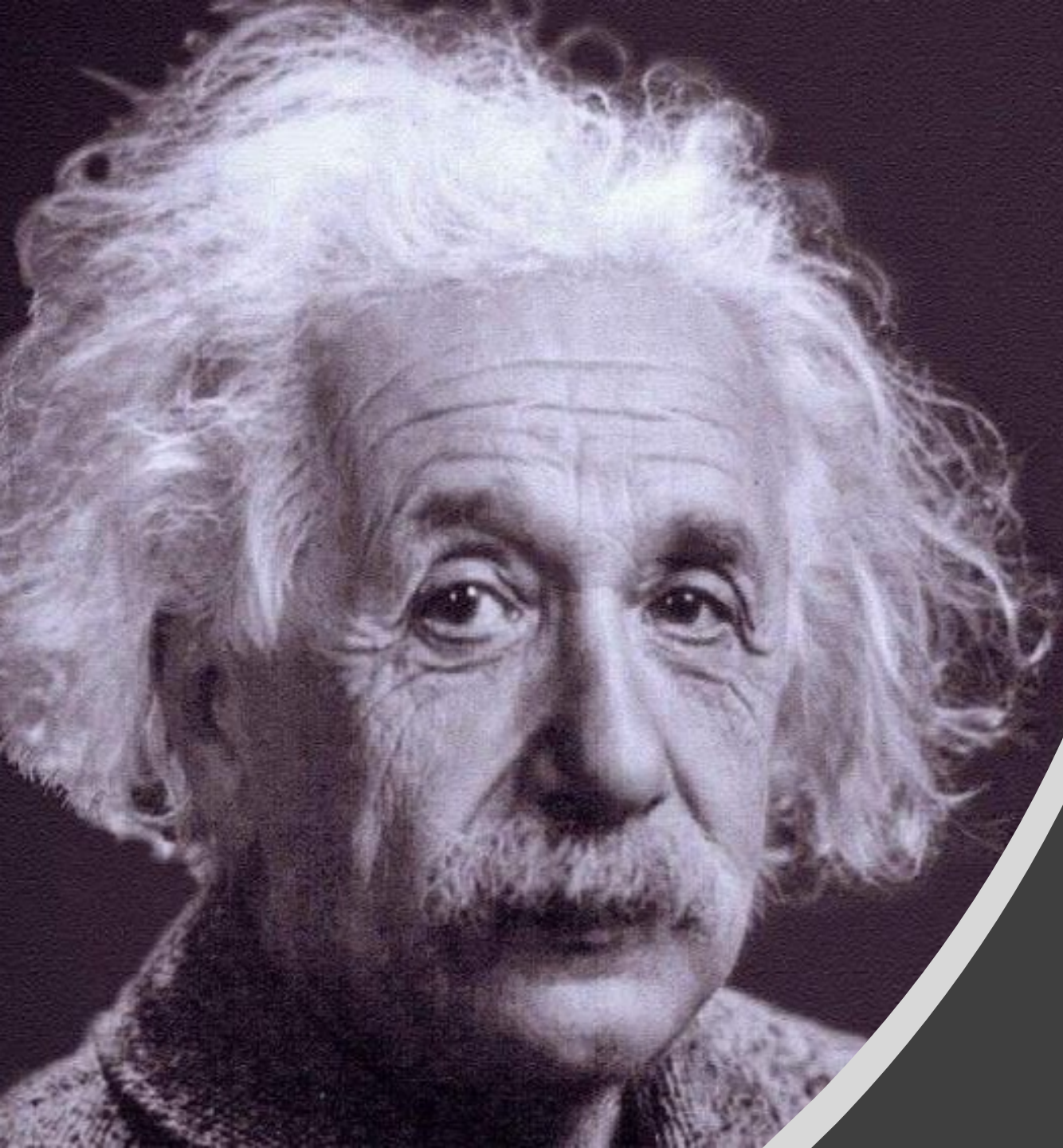


单调栈：代码演示

Node *__insert(Node *root, int key) {

if (root == NIL) return getNewNode(key);

<-6班 资料 /X.现场撸代码 /15.RBT.cpp [FORMAT=unix] [TYPE=CPP] [POS=54,30][62%] 21/09/19 - 20:21



为什么
会出一样的题目？