

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [更多](#) ▼
[您还未登录！](#) [登录](#) [注册](#)

The time is passing

- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

poj 3468 树状数组解法

博客分类：

- [algorithms](#)

一 算法

树状数组天生用来动态维护数组前缀和，其特点是每次更新一个元素的值，查询只能查数组的前缀和，但这个题目求的是某一区间的数组和，而且要支持批量更新某一区间内元素的值，怎么办呢？实际上，还是可以把问题转化为求数组的前缀和。

首先，看更新操作 $\text{update}(s, t, d)$ 把区间 $A[s] \dots A[t]$ 都增加 d ，我们引入一个数组 $\text{delta}[i]$ ，表示 $A[i] \dots A[n]$ 的共同增量， n 是数组的大小。那么 update 操作可以转化为：

1) 令 $\text{delta}[s] = \text{delta}[s] + d$ ，表示将 $A[s] \dots A[n]$ 同时增加 d ，但这样 $A[t+1] \dots A[n]$ 就多加了 d ，所以

2) 再令 $\text{delta}[t+1] = \text{delta}[t+1] - d$ ，表示将 $A[t+1] \dots A[n]$ 同时减 d

然后来看查询操作 $\text{query}(s, t)$ ，求 $A[s] \dots A[t]$ 的区间和，转化为求前缀和，设 $\text{sum}[i] = A[1] + \dots + A[i]$ ，则

$$A[s] + \dots + A[t] = \text{sum}[t] - \text{sum}[s-1],$$

那么前缀和 $\text{sum}[x]$ 又如何求呢？它由两部分组成，一是数组的原始和，二是该区间内的累计增量和，把数组 A 的原始


值保存在数组 org 中，并且 $\text{delta}[i]$ 对 $\text{sum}[x]$ 的贡献值为 $\text{delta}[i] * (x+1-i)$ ，那么

$$\begin{aligned}\text{sum}[x] &= \text{org}[1] + \dots + \text{org}[x] + \text{delta}[1] * x + \text{delta}[2] * (x-1) + \text{delta}[3] * (x-2) + \dots + \text{delta}[x] * 1 \\ &= \text{org}[1] + \dots + \text{org}[x] + \text{segma}(\text{delta}[i] * (x+1-i)) \\ &= \text{segma}(\text{org}[i]) + (x+1) * \text{segma}(\text{delta}[i]) - \text{segma}(\text{delta}[i] * i), 1 \leq i \leq x\end{aligned}$$

这其实就是三个数组 $\text{org}[i]$ ， $\text{delta}[i]$ 和 $\text{delta}[i] * i$ 的前缀和， $\text{org}[i]$ 的前缀和保持不变，事先就可以求出来， $\text{delta}[i]$ 和 $\text{delta}[i] * i$ 的前缀和是不断变化的，可以用两个树状数组来维护。

树状数组的解法比朴素线段树快很多，如果把`long long`变量改成`__int64`，然后用C提交的话，可以达到1047ms，排在22名，但很奇怪，如果用`long long`变量，用gcc提交的话就要慢很多。

二 代码

C代码 

```
1. #include <stdio.h>
2.
3. #define DEBUG
4.
5. #ifndef DEBUG
6. #define debug(...) printf( __VA_ARGS__ )
7. #else
8. #define debug(...)
9. #endif
10.
11. #define N 100002
12.
13. #define lowbit(i) ( i & (-i) )
14.
15. /* 设delta[i]表示[i,n]的公共增量 */
16. long long c1[N];    /* 维护delta[i]的前缀和 */
17. long long c2[N];    /* 维护delta[i]*i的前缀和 */
18. long long sum[N];
19. int      A[N];
20. int n;
21.
22. long long query(long long *array, int i)
23. {
24.     long long tmp;
25.
26.     tmp = 0;
27.     while (i > 0) {
28.         tmp += array[i];
29.         i -= lowbit(i);
30.     }
31.     return tmp;
32. }
33.
34. void update(long long *array, int i, long long d)
35. {
36.     while (i <= n) {
37.         array[i] += d;
```

```

38.         i += lowbit(i);
39.     }
40. }
41.
42. int main()
43. {
44.     int      q, i, s, t, d;
45.     long long ans;
46.     char      action;
47.
48.     scanf("%d %d", &n, &q);
49.     for (i = 1; i <= n; i++) {
50.         scanf("%d", A+i);
51.     }
52.     for (i = 1; i <= n; i++) {
53.         sum[i] = sum[i-1] + A[i];
54.     }
55.
56.     while (q--) {
57.         getchar();
58.         scanf("%c %d %d", &action, &s, &t);
59.         if (action == 'Q') {
60.             ans = sum[t] - sum[s-1];
61.             ans += (t+1)*query(c1, t) - query(c2, t);
62.             ans -= (s*query(c1, s-1) - query(c2, s-1));
63.             printf("%lld\n", ans);
64.         }
65.         else {
66.             scanf("%d", &d);
67.             /* 把delta[i] (s<=i<=t)加d, 策略是
68.              *先把[s, n]内的增量加d, 再把[t+1, n]的增量减d
69.              */
70.             update(c1, s, d);
71.             update(c1, t+1, -d);
72.             update(c2, s, d*s);
73.             update(c2, t+1, -d*(t+1));
74.         }

```

```

75.     }
76.     return 0;
77. }

```

事实上, 还可以不通过求s和t的前缀和, 而是直接求出[s, t]的区间和, 这是因为:


$$\text{sum}[t] = \text{segma}(\text{org}[i]) + (t+1)*\text{segma}(\text{delta}[i]) - \text{segma}(\text{delta}[i]*i) \quad 1 \leq i \leq t$$

$$\text{sum}[s-1] = \text{segma}(\text{org}[i]) + s*\text{segma}(\text{delta}[i]) - \text{segma}(\text{delta}[i]*i) \quad 1 \leq i \leq s-1$$

[s, t]的区间和可以表示为:

$$\begin{aligned}
 \text{sum}[t]-\text{sum}[s-1] &= \text{org}[s] + \dots + \text{org}[t] + (t+1)*(\text{delta}[s] + \dots + \text{delta}[t]) + (t-s+1)*(\text{delta}[1] + \dots + \text{delta}[s-1]) \\
 &\quad - (\text{delta}[s]*s + \dots + \text{delta}[t]*t) \\
 &= \text{segma}(\text{org}[i]) + (t+1)* \text{segma}(\text{delta}[i]) - \text{segma}(\text{delta}[i]*i) , \quad s \leq i \leq t \\
 &\quad + (t-s+1)*\text{segma}(\text{delta}[i]), \quad 1 \leq i \leq s-1
 \end{aligned}$$

问题转化为求三个数组org, delta[i]和delta[i]*i的区间和, 而线段树可以直接求出区间和, 所以又得到了另外一种解法:

C代码 

```

1. #include <stdio.h>
2.
3. // #define DEBUG
4.
5. #ifdef DEBUG
6. #define debug(...) printf( __VA_ARGS__ )
7. #else
8. #define debug(...)

```

```
9. #endif
10.
11. #define N 100002
12.
13. /* 设delta[i]表示[i,n]的公共增量 */
14. long long tree1[262144]; /* 维护delta[i]的前缀和 */
15. long long tree2[262144]; /* 维护delta[i]*i的前缀和 */
16. long long sum[N];
17. int A[N];
18. int n, M;
19.
20. /* 查询[s,t]的区间和 */
21. long long query(long long *tree, int s, int t)
22. {
23.     long long tmp;
24.
25.     tmp = 0;
26.     for (s = s+M-1, t = t+M+1; (s^t) != 1; s >>= 1, t >>= 1) {
27.         if (~s&1) {
28.             tmp += tree[s^1];
29.         }
30.         if (t&1) {
31.             tmp += tree[t^1];
32.         }
33.     }
34.     return tmp;
35. }
36.
37. /* 修改元素i的值 */
38. void update(long long *tree, int i, long long d)
39. {
40.     for (i = (i+M); i > 0; i >>= 1) {
41.         tree[i] += d;
42.     }
43. }
44.
45. int main()
```



```

46. {
47.     int      q, i, s, t, d;
48.     long long ans;
49.     char      action;
50.
51.     scanf("%d %d", &n, &q);
52.     for (i = 1; i <= n; i++) {
53.         scanf("%d", A+i);
54.     }
55.     for (i = 1; i <= n; i++) {
56.         sum[i] = sum[i-1] + A[i];
57.     }
58.
59.     for (M = 1; M < (n+2); M <= 1);
60.
61.     while (q--) {
62.         getchar();
63.         scanf("%c %d %d", &action, &s, &t);
64.         if (action == 'Q') {
65.             ans = sum[t] - sum[s-1];
66.             ans += (t+1)*query(tree1, s, t)+(t-s+1)*query(tree1, 1, s-1);
67.             ans -= query(tree2, s, t);
68.             printf("%lld\n", ans);
69.         }
70.         else {
71.             scanf("%d", &d);
72.             /* 把delta[i] (s<=i<=t)加d, 策略是
73.              *先把[s,n]内的增量加d, 再把[t+1,n]的增量减d
74.              */
75.             update(tree1, s, d);
76.             update(tree2, s, d*s);
77.             if (t < n) {
78.                 update(tree1, t+1, -d);
79.                 update(tree2, t+1, -d*(t+1));
80.             }
81.         }
82.     }

```

```
83.     return 0;
84. }
```

两种解法本质上是一样的，其实zkw式线段树 == 树状数组，它们都可以支持查询某个区间的和，以及修改某个点的值，但不能直接修改某个区间的值，必须引入一个额外的数组，如这题的delta数组，把对区间的修改转化为对两个端点的修改。

分享到:  

[poi 2777 线段树](#) | [poi 3468 非递归的zkw式线段树解法](#)

- 2011-03-14 21:32
- 浏览 3459
- [评论 \(0\)](#)
- 分类: [编程语言](#)
- [相关推荐](#)

评论


发表评论



[您还没有登录, 请您登录后再发表评论](#)



kenby

- 浏览: 245549 次
- 性别: 

- 来自：北京



最近访客

[更多访客>>](#)



[huangrui](#)



[dylinshi126](#)



[jianzxx](#)



[chenying998179](#)

文章分类

- [全部博客 \(133\)](#)
- [Python \(3\)](#)
- [linux \(25\)](#)
- [c \(8\)](#)
- [web \(5\)](#)
- [algorithms \(38\)](#)
- [Network \(2\)](#)
- [erlang \(1\)](#)
- [java \(1\)](#)

社区版块

- [我的资讯 \(0\)](#)

- [我的论坛](#) (0)
- [我的问答](#) (0)

存档分类

- [2014-08](#) (1)
- [2013-11](#) (2)
- [2012-07](#) (2)
- [更多存档...](#)

最新评论

- [kenby](#): zmfyea 写道写的太赞了!!! 怎么没有下篇了? 好久没时间更 ...
[Yaf源码阅读之请求的处理\(二\)](#)
- [zmfyea](#): 写的太赞了!!! 怎么没有下篇了?
[Yaf源码阅读之请求的处理\(二\)](#)
- [frankfan915](#): 不错, 了解了
[Java静态内部类](#)
- [joeytang](#): 赞
[多线程与volatile变量](#)
- [u011076827](#): 读您这篇非常收益, 谢谢! 同时有个问题不太懂: (图2) IO1 ...
[Tornado源码分析之http服务器篇](#)

声明: ITeye文章版权属于作者, 受法律保护。没有作者书面许可不得转载。若作者同意转载, 必须以超链接形式标明文章原始出处和作者。

© 2003-2015 ITeye.com. All rights reserved. [京ICP证110151号 京公网安备110105010620]