

线段树练习

人员

李锦澍、陈欣妙、刘奕辰、杨俊彦、袁晨峻 到课

上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/1687>

The screenshot shows a competition results page with the following header navigation: 比赛概况, 题目列表, 选择题列表, 提交记录, 实时榜单 (Real-time Ranking), and 选择题排行榜 (Choice Question Ranking). The main title of the contest is "王向东老师周日八点半C++线段树区间修改". Below the title is a refresh button. The table displays the following data:

#	用户名	姓名	编程分	时间	A	B	C	D
1	yuanchenjun	袁晨峻	400	2244	100	100	100	100
2	chenxinmiao	陈欣妙	400	2830	100	100	100	100
3	liuyichen	刘奕辰	200	1903	100		100	
4	yangjuniyan	杨俊彦	200	4198	100	100		
5	lijinshu	李锦澍	200	4307	100	100		
6	zhouzhirun	周治润	100	2297	100			

本周作业

<https://cppoj.kids123code.com/contest/1787> (课上讲了 A ~ C 题, 课后作业是 C 题)

课堂表现

今天讲了几道线段树比较复杂的题目, 同学们课上做起来整体会比较吃力一些, 会遇到各种各样的错误

连上之前的课, 最近 3 节课一共讲了 10 道线段树的题目, 同学们课下要多复习这 10 道题, 确保能把这 10 道题写熟, 这样以后遇到线段树的题, 才能比较快的做对。

课堂内容

[TJOI2009] 开关 (上周作业)

tr[u].sum: u 这一段中 1 的数量

tr[u].len: u 这一段的长度

tr[u].flag: u 这一段被修改了多少次

每修改一次, 相当于做一次翻转, sum 需要改成 len-sum

```
#include <bits/stdc++.h>
using namespace std;
```

```
const int maxn = 1e5 + 5;
struct node {
    int l, r, len, sum;
    int flag;
} tr[maxn*4];

void pushup(int u) { tr[u].sum = tr[u*2].sum + tr[u*2+1].sum; }

void pushdown(int u) {
    if (tr[u].flag & 1) {
        tr[u*2].flag += 1, tr[u*2+1].flag += 1;
        tr[u*2].sum = tr[u*2].len - tr[u*2].sum;
        tr[u*2+1].sum = tr[u*2+1].len - tr[u*2+1].sum;
    }
    tr[u].flag = 0;
}

void build(int u, int l, int r) {
    tr[u] = {l, r, r-l+1};
    if (l == r) return;

    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
}

void modify(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) {
        tr[u].flag++; tr[u].sum = tr[u].len-tr[u].sum; return;
    }

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r);
    if (r > mid) modify(u*2+1, l, r);
    pushup(u);
}

int query(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) return tr[u].sum;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    int sum = 0;
    if (l <= mid) sum += query(u*2, l, r);
    if (r > mid) sum += query(u*2+1, l, r);
    return sum;
}

int main()
{
    int n, m; cin >> n >> m;
    build(1, 1, n);
    while (m -- ) {
        int op, l, r; cin >> op >> l >> r;
        if (op == 1) modify(1, l, r);
        else cout << query(1, l, r) << endl;
    }
}
```

```

    if (op == 0) modify(1, l, r);
    else cout << query(1, l, r) << endl;
}
return 0;
}

```

扶苏的问题

同时维护 flag 和 add 两个懒标记, 代表是否变为同一值 和 加了多少

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 1e6 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct node {
    int l, r, maxx, flag, add;
} tr[maxn*4];
int w[maxn];

void pushup(int u) { tr[u].maxx = max(tr[u*2].maxx, tr[u*2+1].maxx); }
void pushdown(int u) {
    if (tr[u].flag != inf) {
        int v = tr[u].flag + tr[u].add;
        tr[u*2].flag = v, tr[u*2].add = 0, tr[u*2].maxx = v;
        tr[u*2+1].flag = v, tr[u*2+1].add = 0, tr[u*2+1].maxx = v;
        tr[u].flag = inf, tr[u].add = 0;
    } else if (tr[u].add) {
        tr[u*2].add += tr[u].add, tr[u*2].maxx += tr[u].add;
        tr[u*2+1].add += tr[u].add, tr[u*2+1].maxx += tr[u].add;
        tr[u].add = 0;
    }
}
void build(int u, int l, int r) {
    tr[u] = {l, r, 0, inf, 0};
    if (l == r) { tr[u].maxx = w[l]; return; }

    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
    pushup(u);
}
void modify(int u, int l, int r, bool is_flag, int k) {
    if (tr[u].l>=l && tr[u].r<=r) {
        if (is_flag) tr[u].flag = k, tr[u].add = 0, tr[u].maxx = k;
        else tr[u].add += k, tr[u].maxx += k;
        return;
    }
    pushdown(u);
}

```

```

int mid = (tr[u].l + tr[u].r) / 2;
if (l <= mid) modify(u*2, l, r, is_flag, k);
if (r > mid) modify(u*2+1, l, r, is_flag, k);
pushup(u);
}
int query(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) return tr[u].maxx;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2, res = -inf;
    if (l <= mid) res = max(res, query(u*2, l, r));
    if (r > mid) res = max(res, query(u*2+1, l, r));
    return res;
}

signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    build(1, 1, n);

    while (m -- ) {
        int op, l, r; cin >> op >> l >> r;
        if (op == 1) {
            int k; cin >> k; modify(1, l, r, true, k);
        } else if (op == 2) {
            int k; cin >> k; modify(1, l, r, false, k);
        } else cout << query(1, l, r) << "\n";
    }
    return 0;
}

```

小白逛公园

维护 sum, lmax, rmax, vmax 四个信息

分别代表 区间和, 前缀最大子段, 后缀最大子段, 内部最大子段 4 个信息

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 5e5 + 5;
const int inf = 0x3f3f3f3f;
struct node {
    int l, r;
    int sum, lmax, rmax, vmax;
} tr[maxn*4];

```

```
int w[maxn];

void pushup(int u) {
    node &uu = tr[u], &ll = tr[u*2], &rr = tr[u*2+1];
    uu.sum = ll.sum + rr.sum;
    uu.lmax = max(ll.lmax, ll.sum + rr.lmax);
    uu.rmax = max(rr.rmax, rr.sum + ll.rmax);
    uu.vmax = max({ll.vmax, rr.vmax, ll.rmax+rr.lmax});
}

void build(int u, int l, int r) {
    tr[u] = {l, r, 0, 0, 0};
    if (l == r) { tr[u].sum = tr[u].lmax = tr[u].rmax = tr[u].vmax = w[l]; return; }

    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
    pushup(u);
}

void modify(int u, int pos, int k) {
    if (tr[u].l == tr[u].r) {
        tr[u].sum = tr[u].lmax = tr[u].rmax = tr[u].vmax = k; return;
    }

    int mid = (tr[u].l + tr[u].r) / 2;
    if (pos <= mid) modify(u*2, pos, k);
    else modify(u*2+1, pos, k);
    pushup(u);
}

node query(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) return tr[u];

    int mid = (tr[u].l + tr[u].r) / 2;
    if (r <= mid) return query(u*2, l, r);
    if (l > mid) return query(u*2+1, l, r);

    node a = query(u*2, l, r), b = query(u*2+1, l, r);
    node c = {a.l, b.r, a.sum+b.sum, max(a.lmax,a.sum+b.lmax), \
              max(b.rmax,b.sum+a.rmax), max({a.vmax,b.vmax,a.rmax+b.lmax})};
    return c;
}

int main()
{
    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    build(1, 1, n);

    while (m -- ) {
        int op; cin >> op;
        if (op == 1) {
            int l, r; cin >> l >> r;
            if (l > r) swap(l, r);
            cout << query(1, l, r).vmax << endl;
        } else {
            int pos, k; cin >> pos >> k;
        }
    }
}
```

```

        modify(1, pos, k);
    }
}
return 0;
}

```

无聊的数列

维护 2 个懒标记 K 和 D, 分别代表对于 u 这一段, 对首项加了多少, 以及后面每一项的公差多多少

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 1e5 + 5;
struct node {
    int l, r, K, D;
} tr[maxn*4];
int w[maxn];

void pushdown(int u) {
    node &uu = tr[u], &l1 = tr[u*2], &r1 = tr[u*2+1];
    l1.K += uu.K, l1.D += uu.D;
    r1.K += uu.K + uu.D*(r1.l-uu.l), r1.D += uu.D;
    uu.K = 0, uu.D = 0;
}
void build(int u, int l, int r) {
    tr[u] = {l, r, 0, 0};
    if (l == r) { tr[u].K = w[l]; return; }

    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
}
void modify(int u, int l, int r, int k, int d) {
    if (tr[u].l>=l && tr[u].r<=r) {
        tr[u].K += k + d*(tr[u].l-l), tr[u].D += d;
        return;
    }

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r, k, d);
    if (r > mid) modify(u*2+1, l, r, k, d);
}
int query(int u, int pos) {
    if (tr[u].l == tr[u].r) return tr[u].K;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (pos <= mid) return query(u*2, pos);
}

```

```
    return query(u*2+1, pos);
}

signed main()
{
    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    build(1, 1, n);

    while (m -- ) {
        int op; cin >> op;
        if (op == 1) {
            int l, r, k, d; cin >> l >> r >> k >> d;
            modify(1, l, r, k, d);
        } else {
            int pos; cin >> pos;
            cout << query(1, pos) << endl;
        }
    }
    return 0;
}
```