差分+线段树

[P1438 无聊的数列]

时间限制：1.00s 内存限制：125.00MB

题意

维护一个数列{a[i]}，支持两种操作：

1. **1 L R K D：** 给出一个长度等于R-L+1的等差数列，首项为K，公差为D，并将它对应加到a[L]~a[R]的每一个数上。即：令a[L]=a[L]+K，a[L+1]=a[L+1]+K+D， a[L+2]=a[L+2]+K+2D……a[R]=a[R]+K+(R-L)D。
2. **2 P：** 询问序列的第P个数的值a[P]。

思路

运用差分对a[l]到a[r]加上等差数列：

1. 对于L:sum[L]=sum[L]+K,
2. 对于区间(L,R]:sum[i]=sum[i]+D,i∈(L,R],
3. 对于R+1:sum[R+1]=sum[R+1]-(K+((R-L)*D). 而对于每次查询P的值,只要输出 s[P]+sum[1]+...+sum[P] 的值即可。

例： 对1~5加上K=1，D=2的等差数列

序号	1	2	3	4	5	6
原数组	1	3	2	4	5	6
等差数列	+1	+3	+5	+7	+9	+0
差分	+1	+2	+2	+2	+2	-9
答案	2	6	7	11	14	6

注意避免l=r和r=n的情况出错

代码

用时：128ms 内存：9.16mb

```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e6 + 10;
int n, m, rt;
int a[N];
class tree {
public:
        int sum, lazy;
        int len;
} t[N << 2];
```

```
#define lson rt << 1
#define rson rt << 1 | 1

template<class T>inline void read(T & x) {
        x = 0; int f = 0; char ch = getchar();
        while (!isdigit(ch)) f |= (ch == '-'), ch = getchar();
        while (isdigit(ch)) x = x * 10 + ch - '0', ch = getchar();
        x = f ? -x : x;
        return;
}

void pushup(int rt) {
        t[rt].sum = t[lson].sum + t[rson].sum;
}

void build(int l, int r, int rt) {
        t[rt].len = r - l + 1;
        if (l == r) return;
        int m = (l + r) >> 1;
        build(l, m, lson);
        build(m + 1, r, rson);
}

inline void pushdown(int rt) {
        if (t[rt].lazy) {
                t[lson].lazy += t[rt].lazy;
                t[rson].lazy += t[rt].lazy;
                t[lson].sum += t[lson].len * t[rt].lazy;
                t[rson].sum += t[rson].len * t[rt].lazy;
                t[rt].lazy = 0;
        }
}

void update(int L, int R, int c, int l, int r, int rt) {
        if (L <= l && r <= R) {
                t[rt].sum += c * t[rt].len;
                t[rt].lazy += c;
                return;
        }
        pushdown(rt);
        int m = (l + r) >> 1;
        if (L <= m) update(L, R, c, l, m, lson);
        if (R > m) update(L, R, c, m + 1, r, rson);
        pushup(rt);
}


int query(int L, int R, int l, int r, int rt) {
        if (L <= l && r <= R) return t[rt].sum;
        pushdown(rt);
        int m = (l + r) >> 1, ans = 0;
        if (L <= m) ans += query(L, R, l, m, lson);
        if (R > m) ans += query(L, R, m + 1, r, rson);
```

```
        return ans;
}

main() {
        read(n), read(m);
        for (int i = 1; i <= n; ++i) read(a[i]);
        build(1, n, 1);
        for (int i = 1, opt, l, r, k, d; i <= m; ++i) {
                read(opt);
                if (opt == 1) {
                        read(l), read(r), read(k), read(d);
                        update(l, l, k, 1, n, 1);
                        if (r > l)
                                update(l + 1, r, d, 1, n, 1);
                        if (r != n)
                                update(r + 1, r + 1, -(k + (r - l) * d), 1, n, 1);
                }
                else {
                        read(k);
                        printf("%d\n", a[k] + query(1, k, 1, n, 1));
                }
        }
        return 0;
}
```