线段树

单点修改，区间查询（pushup）

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

const int N = 5e5 + 10;
int n, m;
int a[N];
struct Node{
    int l, r, sum;
}tr[4 * N];

void pushup(Node& u, Node& l, Node& r){
    u.sum = l.sum + r.sum;
}

void pushup(int u){
    pushup(tr[u], tr[u << 1], tr[u << 1 | 1]);
}

void build(int u, int l, int r){
    tr[u].l = l, tr[u].r = r;
    if(l == r){
        tr[u].sum = a[l];
        return;
    }
    int mid = l + r >> 1;
    build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
    pushup(u);
}

void modify(int u, int x, int k){
    if(tr[u].l == tr[u].r){
        tr[u].sum += k;
        return;
    }
    int mid = tr[u].l + tr[u].r >> 1;
    if(x <= mid) modify(u << 1, x, k);
    else modify(u << 1 | 1, x, 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 >> 1;
    if(mid >= r) return query(u << 1, l, r);
    if(l > mid) return query(u << 1 | 1, l, r);
    Node left = query(u << 1, l, r), right = query(u << 1 | 1, l, r);
    Node res;
    pushup(res, left, right);
```

```
    return res;
}

int main(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    build(1, 1, n);
    for(int i = 1; i <= m; i++){
        int op, x, y, k;
        cin >> op;
        if(op == 1){
            cin >> x >> k;
            modify(1, x, k);
        }
        else{
            cin >> x >> y;
            cout << query(1, x, y).sum << '\n';
        }
    }
    return 0;
}
```

区间修改，区间查询（pushdown）

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

typedef long long ll;
const int N = 5e5 + 10;
int n, m;
ll a[N];
struct Node{
    int l, r;
    ll sum, add;
}tr[4 * N];

void pushdown(int u){
    Node &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    left.sum += (ll)(left.r - left.l + 1) * root.add, left.add += root.add;
    right.sum += (ll)(right.r - right.l + 1) * root.add, right.add += root.add;
    root.add = 0;
}

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

void build(int u, int l, int r){
    tr[u].l = l, tr[u].r = r;
    if(l == r){
        tr[u].sum = a[l];
        return;
    }
```

```
    int mid = l + r >> 1;
    build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
    pushup(u);
}

void modify(int u, int l, int r, ll k){
    if(tr[u].l >= l && tr[u].r <= r){
        tr[u].sum += (ll)(tr[u].r - tr[u].l + 1) * k;
        tr[u].add += k;
        return;
    }
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    if(mid >= l) modify(u << 1, l, r, k);
    if(mid < r) modify(u << 1 | 1, l, r, k);
    pushup(u);
}

ll 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 >> 1;
    ll sum = 0;
    if(mid >= l) sum += query(u << 1, l, r);
    if(mid < r) sum += query(u << 1 | 1, l, r);
    return sum;
}

int main(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    build(1, 1, n);
    for(int i = 1; i <= m; i++){
        int op, x, y;
        ll k;
        cin >> op;
        if(op == 1){
            cin >> x >> y >> k;
            modify(1, x, y, k);
        }
        else{
            cin >> x >> y;
            cout << query(1, x, y) << '\n';
        }
    }
    return 0;
}
```