补充

主席树求静态区间第 K 大

```
#include<bits/stdc++.h>
using namespace std;
const int N = 2e5 + 10;
int l[N << 5], r[N << 5], cnt, sum[N << 5], a[N], b[N], c[N], n, q, m;
inline int Build(int t, int w) {
    int now = ++cnt, mid = (t + w) >> 1;
    sum[now] = 0;
    if (t < w) { l[now] = Build(t, mid); r[now] = Build(mid + 1, w); }
    return now;
}
inline int add(int k, int t, int w, int x) {
    int now = ++cnt, mid = (t + w) >> 1;
    l[now] = l[k], r[now] = r[k], sum[now] = sum[k] + 1;
    if (t < w && x <= mid) l[now] = add(l[k], t, mid, x);
    else if (t < w && x > mid) r[now] = add(r[k], mid + 1, w, x);
    return now;
}
inline int ask(int L, int R, int t, int w, int x) {
    if (t >= w) return t;
    int mid = (t + w) >> 1, k = sum[l[R]] - sum[l[L]];
    if (k >= x) return ask(l[L], l[R], t, mid, x); else  return ask(r[L], r[R], mid +
1, w, x - k);
}
int main() {
    n = read(), q = read();
    for (int i = 1; i <= n; i++) b[i] = a[i] = read();
    std::sort(b + 1, b + 1 + n);
    m = unique(b + 1, b + 1 + n) - b - 1;
    c[0] = Build(1, m);
    for (int i = 1; i <= n; i++) {
        int now = lower_bound(b + 1, b  + 1 + m, a[i]) - b;
        c[i] = add(c[i - 1], 1, m, now);
    }
    for (int i = 1; i <= q; i++) {
        int x = read(), y = read(), z = read();
        printf("%d\n", b[ask(c[x - 1], c[y], 1, m, z)]);
    }
    return 0;
}
```

主席树求区间前 k 大和

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

```
const int N = 2e5 + 10;
int l[N << 5], r[N << 5], cnt, a[N], b[N], c[N], n, q, m, L, k, d[N];
long long Max, s[N << 5], sum[N << 5], val[N << 5], pre[N << 5];
inline int read() {
    int x = 0, k = 1; char ch = getchar();
    for (; ch < 48 || ch > 57; ch = getchar()) k ^= (ch == '-');
    for (; ch >= 48 && ch <= 57; ch = getchar()) x = x * 10 + (ch ^ 48);
    return k ? x : -x;
}
inline int Build(int t, int w) {
    int now = ++cnt, mid = (t + w) >> 1;
    sum[now] = 0, val[now] = 0, pre[now] = 0;
    if (t == w) {
        return now;
    }
    l[now] = Build(t, mid); r[now] = Build(mid + 1, w);
    return now;
}
inline int add(int k, int t, int w, int x, int p) {
    int now = ++cnt, mid = (t + w) >> 1;
    l[now] = l[k], r[now] = r[k], sum[now] = sum[k] + 1, pre[now] = pre[k] + p;
    if (t == w) {
        val[now] = p;
        return now;
    }
    if (x <= mid) l[now] = add(l[k], t, mid, x, p);
    else if (x > mid) r[now] = add(r[k], mid + 1, w, x, p);
    return now;
}
inline int ask(int L, int R, int t, int w, int x) {
    if (t == w) return val[R] * x;
    int mid = (t + w) >> 1, k = sum[r[R]] - sum[r[L]];
    if (k < x) return ask(l[L], l[R], t, mid, x - k) + pre[r[R]] - pre[r[L]]; else
 return ask(r[L], r[R], mid + 1, w, x);
}
signed main() {
    n = read();
    for (int i = 1; i <= n; i++)
        d[i] = read(), s[i] = s[i - 1] + d[i];
    for (int i = 1; i <= n; i++) b[i] = a[i] = read();
    for (int i = 1; i <= n; i++) b[n + i] = a[n + i] = a[i], s[n + i] = s[n + i - 1] +
d[i];
    n = n * 2;
    std::sort(b + 1, b + 1 + n);
    m = unique(b + 1, b + 1 + n) - b - 1;
    c[0] = Build(1, m);
    for (int i = 1; i <= n; i++) {
        int now = lower_bound(b + 1, b  + 1 + m, a[i]) - b;
        c[i] = add(c[i - 1], 1, m, now, a[i]);
    }
    k = read(), L = read();
    // for (int i = 1; i <= n; i++)
    //     printf("%lld ", s[i]);
    for (int i = n / 2 - k + 1; i <= n / 2 + 1; i++) {
        // printf("%d %d\n", i + k - 1, i - 1);
        // printf("%lld %lld\n", s[i + k - 1] - s[i - 1], ask(c[i - 1], c[i + k - 1],
1, m, L));
        Max = max(Max, s[i + k - 1] - s[i - 1] + ask(c[i - 1], c[i + k - 1], 1, m,
L));
        //ask(c[i - 1], c[i + k - 1], 1, m, L) 求区间前 k 大和
    }
    printf("%lld\n", Max);
    return 0;
}
```

数位DP

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int Mo = 1e9 + 7;
int l1, l2, l3, r1, r2, r3, a[10], f[40][40];
inline int read() {
    int x = 0, k = 1; char ch = getchar();
    for (; ch < 48 || ch > 57; ch = getchar()) k ^= (ch == '-');
    for (; ch >= 48 && ch <= 57; ch = getchar()) x = x * 10 + (ch ^ 48);
    return k ? x : -x;
}
inline int ksm(int a, int b) {
    int ans = 1;
    for (; b; b >>= 1, a = 1LL * a * a % Mo) if (b & 1) ans = 1LL * ans * a % Mo;
    return ans;
}
inline int w1(int x, int i) {
    return ((x >> i) & 1);
}
int dfs(int x, int y, int z) {
    memset(f, 0, sizeof(f));
    f[0][0] = 1;
    int cnt = 0;
    for (int i = 0; i <= 31; i++) {
        cnt = w1(x, i) * 4 + w1(y, i) * 2 + w1(z, i);
        for (int j = 1; j <= 4; j++) {
            for (int k = 0; k < 8; k++) {
                int kk = 0;
                for (int t = 0; t <= 2; t++)
                    if (w1(a[j], t) == 1 && w1(cnt, t) == 0)
                        kk += (1 << t);
                    else if (w1(a[j], t) == w1(cnt, t) && w1(k, t) == 1)
                        kk += (1 << t);
                f[i + 1][kk] = (f[i + 1][kk] + f[i][k]) % Mo;
            }
        }
    }
    // for (int i = 0; i <= 32; i++) {
    //     for (int k = 0; k < 8; k++)
    //         printf("%d ", f[i][k]);
    //     printf("\n");
    // }
    // printf("\n");
    // printf("%lld\n", f[32][0]);
    return f[32][0];
}
inline int solve() {
    return ((dfs(r1, r2, r3) - dfs(r1, r2, l3 - 1) - dfs(r1, l2 - 1, r3) - dfs(l1 - 1,
r2, r3) + dfs(l1 - 1, l2 - 1, r3) + dfs(l1 - 1, r2, l3 - 1) + dfs(r1, l2 - 1, l3 - 1)
- dfs(l1 - 1, l2 - 1, l3 - 1)) % Mo + Mo) % Mo;
}
```

```
signed main() {
    a[1] = 0, a[2] = 3, a[3] = 5, a[4] = 6;
    // for (int T = read(); T--; ) {
    l1 = read(), r1 = read(), l2 = read(), r2 = read(), l3 = read(), r3 = read();
    printf("%lld\n", (Mo + 1 - solve() * ksm((r1 - l1 + 1) * (r2 - l2 + 1) % Mo * (r3
- l3 + 1) % Mo, Mo - 2) % Mo) % Mo);
    // }
}
```

Dinic

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N = 1e5 + 10, INF = 1e18;
int cnt, dis[N], head[N], head1[N], w, s, t, n, m, u, v, ans;
bool tf[N];
struct Node {
    int to, nxt, w;
} E[N];
inline int read() {
    int x = 0, k = 1; char ch = getchar();
    for (; ch < 48 || ch > 57; ch = getchar()) k ^= (ch == '-');
    for (; ch >= 48 && ch <= 57; ch = getchar()) x = x * 10 + (ch ^ 48);
    return k ? x : -x;
}
inline void Build(int x, int y, int z) {
    E[++cnt].to = y, E[cnt].nxt = head[x], head[x] = cnt, E[cnt].w = z;
}
inline bool BFS() {
    memset(dis, 0, sizeof(dis));
    // memset(tf, false, sizeof(tf));
    dis[s] = 1;
    queue<int> q;
    q.push(s);
    for (; !q.empty(); ) {
        int x = q.front();
        q.pop();
        // tf[x] = true;
        head1[x] = head[x];
        for (int i = head[x]; i; i = E[i].nxt) {
            int now = E[i].to;
            if (dis[now] == 0 && E[i].w) {
                dis[now] = dis[x] + 1;
                q.push(now);
                if (now == t)
                return 1;
            }
        }
    }
    return 0;
}
inline int DFS(int x, int flow) {
    if (x == t)
        return flow;
    int fl = flow;
    for (int i = head1[x]; i; i = E[i].nxt) {
        int now = E[i].to;
        head1[x] = i;
        if (dis[now] == dis[x] + 1 && E[i].w) {
            int lo = DFS(now, min(fl, E[i].w));
            fl -= lo;
            E[i].w -= lo, E[i ^ 1].w += lo;
            if (!fl)
                break;
        }
    }
    return flow - fl;
}
signed main() {
    cnt = 1;
    n = read(), m = read(), s = read(), t = read();
    for (int i = 1; i <= m; i++) {
        u = read(), v = read(), w = read();
        Build(u, v, w);
        Build(v, u, 0);
    }
    for (; BFS(); ) {
        memcpy(head1, head, sizeof(head1));
        ans += DFS(s, INF);
    }
    printf("%lld\n", ans);
    return 0;
}
```

线段树

单点修改、区间修改、区间查询（区间求和，求区间最大值，求区间最小值）等操作

区间 $+k$ 加区间 $*k$ 加区间求和

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

const int N = 1e5 + 10;
int n, Mo, m, val[N], ans;

struct Node {
    int plus, mul, sum;
} tree[N << 2];

inline int read() {
    int x = 0, k = 1; char c = getchar();
    for (; c < 48 || c > 57; c = getchar()) k ^= (c == '-');
    for (; c >= 48 && c <= 57; c = getchar()) x = x * 10 + (c ^ 48);
    return k ? x : -x;
}

inline void pushup(int p) {
    tree[p].sum = (tree[p << 1].sum + tree[p << 1 | 1].sum) % Mo;
}

inline void pushdown(int p, int k) {
    tree[p << 1].sum = (tree[p << 1].sum * tree[p].mul % Mo + tree[p].plus * (k - (k
>> 1)) % Mo) % Mo;
    tree[p << 1 | 1].sum = (tree[p << 1 | 1].sum * tree[p].mul % Mo + tree[p].plus *
(k >> 1) % Mo) % Mo;
    tree[p << 1].mul = tree[p << 1].mul * tree[p].mul % Mo;
    tree[p << 1 | 1].mul = tree[p << 1 | 1].mul * tree[p].mul % Mo;
    tree[p << 1].plus = (tree[p << 1].plus * tree[p].mul % Mo + tree[p].plus) % Mo;
    tree[p << 1 | 1].plus = (tree[p << 1 | 1].plus * tree[p].mul % Mo + tree[p].plus)
% Mo;
    tree[p].plus = 0, tree[p].mul = 1;
}

void Build(int p, int l, int r) {
    if (l == r) {
        tree[p].sum = val[l] % Mo;
        return;
    }
    Build(p << 1, l, l + r >> 1);
    Build(p << 1 | 1, (l + r >> 1) + 1, r);
    pushup(p);
}

void Update_plus(int p, int l, int r, int L, int R, int Val) {
    if (L <= l && R >= r) {
        tree[p].sum = (tree[p].sum + (r- l + 1) * Val % Mo) % Mo;
        tree[p].plus = (tree[p].plus + Val) % Mo;
        return;
    }
    pushdown(p, r - l + 1);
    if (l + r >> 1 >= L)
        Update_plus(p << 1, l, l + r >> 1, L, R, Val);
    if (l + r >> 1 < R)
        Update_plus(p << 1 | 1, (l + r >> 1) + 1, r, L, R, Val);
    pushup(p);
}
void Update_Mul(int p, int l, int r, int L, int R, int Val) {
    if (L <= l && R >= r) {
        tree[p].plus = tree[p].plus * Val % Mo;
        tree[p].mul = tree[p].mul * Val % Mo;
        tree[p].sum = tree[p].sum * Val % Mo;
        return;
    }
    pushdown(p, r - l + 1);
    if (l + r >> 1 >= L)
        Update_Mul(p << 1, l, l + r >> 1, L, R, Val);
    if (l + r >> 1 < R)
        Update_Mul(p << 1 | 1, (l + r >> 1) + 1, r, L, R, Val);
    pushup(p);
}

int query(int p, int l, int r, int L, int R) {
    if (L <= l && R >= r) {
        return tree[p].sum;
    }
    pushdown(p, r - l + 1);
    int ans = 0;
    if (l + r >> 1 >= L)
        ans += query(p << 1, l, l + r >> 1, L, R);
    ans %= Mo;
    if (l + r >> 1 < R)
        ans += query(p << 1 | 1, (l + r >> 1) + 1, r, L, R);
    ans %= Mo;
    pushup(p);
    return ans;
}

signed main() {
    n = read(), m = read(), Mo = read();
    for (int i = 1; i <= n; i++)
        val[i] = read() % Mo;
    Build(1, 1, n);
    for (int i = 1; i <= (N * 4); i++)
        tree[i].mul = 1;
    for (int i = 1; i <= m; i++) {
        int opt = read();
        if (opt == 1) {
            int x = read(), y = read(), k = read() % Mo;
            Update_Mul(1, 1, n, x, y, k);
        }
        else if (opt == 2) {
            int x = read(), y = read(), k = read() % Mo;
            Update_plus(1, 1, n, x, y, k);
        }
        else if (opt == 3) {
            int x = read(), y = read();
            printf("%lld\n", query(1, 1, n, x, y));
        }
    }
    return 0;
}
```

$A[x]$ 改成 y

求 $[x, y]$ 这个区间的最大连续和

```
#include <iostream>
#include <cstring>
#include <algorithm>
#define lch (k<<1)
#define rch (k<<1|1)
#define mid (l+r>>1)
using namespace std;
const int N=1e5+7;
int n,m,a[N];
struct node{
    int lm,rm,sm,mx;
}tr[4*N];
node merge(node L,node R){
    node M;
    M.lm=max(L.lm,L.sm+R.lm);
    M.rm=max(R.rm,R.sm+L.rm);
    M.mx=max({L.mx,R.mx,M.lm,M.rm,L.rm+R.lm});
    M.sm=L.sm+R.sm;
    return M;
}
void pushup(int k){
    tr[k]=merge(tr[lch],tr[rch]);
}
void build(int k,int l,int r){
    if(l==r){
        tr[k].lm=tr[k].rm=tr[k].sm=tr[k].mx=a[l];
        return;
    }
    build(lch,l,mid);
    build(rch,mid+1,r);
    pushup(k);
}
node query(int k,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr){
        return tr[k];
    }
    if(qr<=mid) return query(lch,l,mid,ql,qr);
    else if(ql>mid) return query(rch,mid+1,r,ql,qr);
    else return merge(query(lch,l,mid,ql,qr),query(rch,mid+1,r,ql,qr));
}

void update(int k,int l,int r,int p,int v){
    if(l==r){
        tr[k].lm=tr[k].rm=tr[k].sm=tr[k].mx=v;
        return;
    }
    if(p<=mid){
        update(lch,l,mid,p,v);
    }else{
        update(rch,mid+1,r,p,v);
    }
    pushup(k);
}

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
    }
    build(1,1,n);
    scanf("%d",&m);
    while(m--){
        int op,l,r;
        scanf("%d %d %d",&op,&l,&r);
        if(op==0){
            update(1,1,n,l,r);
        }else{
            printf("%d\n",query(1,1,n,l,r).mx);
        }
    }
}
```

区间修改

维护区间最长连续的 1

```
#include <iostream>
#define lch (k<<1)
#define rch (k<<1|1)
#define mid (l+r>>1)
using namespace std;
const int N=1e5+7;
int n,m;
int len[4*N],sum[4*N],lmx[4*N],rmx[4*N],lz[4*N];
//sum区间最长连续空房的长度
```

```
//lmx从l端点开始最长连续空房的长度
//rmx从r端点开始最长连续空房的长度
//lazy为1表示退房，为2表示开房
//区间长度，记录后方便计算
void init(int k,int l,int r){
    sum[k]=len[k]=lmx[k]=rmx[k]=r-l+1;
    lz[k]=-1;
    if(l==r){
        return;
    }
    init(lch,l,mid);
    init(rch,mid+1,r);
}
void pushup(int k){
    lmx[k]=(lmx[lch]==len[lch])? lmx[lch]+lmx[rch]:lmx[lch];
    //若左儿子区间全空那么lmax可以横跨左右儿子，否则不能
    rmx[k]=(rmx[rch]==len[rch])? rmx[rch]+rmx[lch]:rmx[rch];
    //若右儿子区间全空那么rmax可以横跨左右儿子，否则不能
    sum[k]=max(rmx[lch]+lmx[rch],max(sum[lch],sum[rch]));
    //有三种情况，sum全在左儿子，全在右儿子，横跨左右儿子
}

void pushdown(int k){
    if(lz[k]!=-1){
        //下传lazy标记
        lz[lch]=lz[rch]=lz[k];
        sum[lch]=lmx[lch]=rmx[lch]=len[lch]*lz[k];
        sum[rch]=lmx[rch]=rmx[rch]=len[rch]*lz[k];
        lz[k]=-1;
    }
}
//区间赋值0/1
void update(int k,int l,int r,int ql,int qr,int val){
    if(ql<=l&&r<=qr){

        lmx[k]=rmx[k]=sum[k]=len[k]*val;
        lz[k]=val;
        return;
    }
    pushdown(k);
    if(ql<=mid)   update(lch,l,mid,ql,qr,val);
    if(mid+1<=qr) update(rch,mid+1,r,ql,qr,val);
    pushup(k);
}
//寻找最左边大于等于x的线段
int query(int k,int l,int r,int x){
    if(l==r) return l;
    pushdown(k);
    if(sum[lch]>=x) return query(lch,l,mid,x);
    //递归到左儿子
    else if(rmx[lch]+lmx[rch]>=x) return mid-rmx[lch]+1;
    //左右儿子合并后满足就用中间
    else return query(rch,mid+1,r,x);
    //递归到右儿子
}
int ask(int k,int l,int r,int p){
    if(l==r) return sum[k];
    pushdown(k);
    if(p<=mid) return ask(lch,l,mid,p);
    else return ask(rch,mid+1,r,p);
}
```

```
int main(){
    cin>>n>>m;
    init(1,1,n);
    while(m--){
        int op,x,y;
        cin>>op;
        if(op==1){
            cin>>x;
            if(sum[1]>=x){
                int pos=query(1,1,n,x);
                cout<<pos<<"\n";
                update(1,1,n,pos,pos+x-1,0);
            }else{
                cout<<"0\n";
            }
        }else{
            cin>>x>>y;
            update(1,1,n,x,x+y-1,1);
        }
    }
}
```

单点修改 查询区间单调连续序列

```
#include <iostream>
#define lch (k<<1)
#define rch (k<<1|1)
#define mid (l+r>>1)
using namespace std;
typedef long long ll;
const int N=2e5+10;
int n,m,a[N];
struct node{
    int lx,rx,lv,rv,len;
    ll s;
}tr[4*N];

node merge(node LC,node RC){
    node RT;
    if(LC.rv<=RC.lv){
        RT.s=LC.s+RC.s+1ll*LC.rx*RC.lx;
        if(LC.lx==LC.len)
            RT.lx=LC.lx+RC.lx;
        else RT.lx=LC.lx;
        if(RC.rx==RC.len) RT.rx=RC.rx+LC.rx;
        else RT.rx=RC.rx;
    }
    else{
        RT.s=LC.s+RC.s;
        RT.lx=LC.lx;
        RT.rx=RC.rx;
    }
    RT.lv=LC.lv,RT.rv=RC.rv;
    RT.len=LC.len+RC.len;
    return RT;
}
void pushup(int k){
    tr[k]=merge(tr[lch],tr[rch]);
}
void init(int k,int l,int r){
    tr[k].len=r-l+1;
    if(l==r){
        tr[k].lv=tr[k].rv=a[l];
        tr[k].lx=tr[k].rx=1;
        tr[k].s=1;
        return;
    }
    init(lch,l,mid);
    init(rch,mid+1,r);
    pushup(k);
}
void update(int k,int l,int r,int p,int v){
    if(l==r){
        tr[k].lv=tr[k].rv=v;
        tr[k].lx=tr[k].rx=1;
        tr[k].s=1;
        return;
    }
    if(p<=mid) update(lch,l,mid,p,v);
    else update(rch,mid+1,r,p,v);
    pushup(k);
}
node query(int k,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr){
        return tr[k];
    }
    if(qr<=mid) return query(lch,l,mid,ql,qr);
    else if(ql>mid) return query(rch,mid+1,r,ql,qr);
    else return merge(query(lch,l,mid,ql,qr),query(rch,mid+1,r,ql,qr));
}

int main(){
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    init(1,1,n);
    while(m--){
        int op,x,y;
        scanf("%d %d %d",&op,&x,&y);
        if(op==1){
            update(1,1,n,x,y);
        }else{
            printf("%lld\n",query(1,1,n,x,y).s);
        }
    }
}
```

区间最大最小值

```
//问最小值
//Q a b 询问(a,b)中最小值
//C a b 将a点值改为b
#include <bits/stdc++.h>
using namespace std;
#define maxn 200005

int min(int a, int b)
{
    return a>b ? b : a;
}
int tree[4 * maxn];
```

```c
void pushup(int i)
{
    tree[i] = min(tree[i << 1], tree[i << 1 | 1]);
}

void build(int i, int l, int r)
{
    if (l == r)
    {
        scanf("%lld", &tree[i]);
        return;
    }
    int mid = (l + r) / 2;
    build(i << 1, l, mid);
    build(i << 1 | 1, mid + 1, r);
    pushup(i);
}

void update(int i, int l, int r, int x, int val)
{
    if (l == r)///l==x²»±ØÒª
    {
        tree[i] = val;
        return;
    }
    int mid = (l + r) / 2;
    if (x <= mid) update(i << 1, l, mid, x, val);
    else update(i << 1 | 1, mid + 1, r, x, val);
    pushup(i);
}

int query(int i, int l, int r, int x, int y)
{
    if (x <= l && r <= y)
        return tree[i];
    int minn = 9999999;
    int mid = (l + r) / 2;
    if (x <= mid) minn = min(minn, query(i << 1, l, mid, x, y));
    if (y>mid) minn = min(minn, query(i << 1 | 1, mid + 1, r, x, y));
    return minn;
}

int main()
{
    int n, m;
    int b, c;
    char a;
    while (scanf("%d%d", &n, &m) != -1)
    {
        build(1, 1, n);
        while (m--)
        {
            scanf(" %c%d%d", &a, &b, &c);
            if (a == 'Q')
                printf("%d\n", query(1, 1, n, b, c));
            else
                update(1, 1, n, b, c);
        }
    }
    return 0;
}
```

区间最值操作

长度为 n 的序列，支持区间加 /区间对 x 取 min / 区间对 x 取 max /求区间和/求区间最大值/求区间最小值。

```
#include <cstdio>
#include <iostream>
using namespace std;

int inline rd() {
  register char act = 0;
  register int f = 1, x = 0;
  while (act = getchar(), act < '0' && act != '-')
    ;
  if (act == '-') f = -1, act = getchar();
  x = act - '0';
  while (act = getchar(), act >= '0') x = x * 10 + act - '0';
  return x * f;
}

const int N = 5e5 + 5, SZ = N << 2, INF = 0x7fffffff;

int n, m;
int a[N];

struct data {
  int mx, mx2, mn, mn2, cmx, cmn, tmx, tmn, tad;
  long long sum;
} t[SZ];

void pushup(int u) {
  const int lu = u << 1, ru = u << 1 | 1;
  t[u].sum = t[lu].sum + t[ru].sum;
  if (t[lu].mx == t[ru].mx) {
    t[u].mx = t[lu].mx, t[u].cmx = t[lu].cmx + t[ru].cmx;
    t[u].mx2 = max(t[lu].mx2, t[ru].mx2);
  } else if (t[lu].mx > t[ru].mx) {
    t[u].mx = t[lu].mx, t[u].cmx = t[lu].cmx;
    t[u].mx2 = max(t[lu].mx2, t[ru].mx);
  } else {
    t[u].mx = t[ru].mx, t[u].cmx = t[ru].cmx;
    t[u].mx2 = max(t[lu].mx, t[ru].mx2);
  }
  if (t[lu].mn == t[ru].mn) {
    t[u].mn = t[lu].mn, t[u].cmn = t[lu].cmn + t[ru].cmn;
    t[u].mn2 = min(t[lu].mn2, t[ru].mn2);
  } else if (t[lu].mn < t[ru].mn) {
    t[u].mn = t[lu].mn, t[u].cmn = t[lu].cmn;
    t[u].mn2 = min(t[lu].mn2, t[ru].mn);
  } else {
    t[u].mn = t[ru].mn, t[u].cmn = t[ru].cmn;
    t[u].mn2 = min(t[lu].mn, t[ru].mn2);
  }
}
```

```cpp
void push_add(int u, int l, int r, int v) {
  // 更新加法标记的同时，更新 $\min$ 和 $\max$ 标记
  t[u].sum += (r - l + 1ll) * v;
  t[u].mx += v, t[u].mn += v;
  if (t[u].mx2 != -INF) t[u].mx2 += v;
  if (t[u].mn2 != INF) t[u].mn2 += v;
  if (t[u].tmx != -INF) t[u].tmx += v;
  if (t[u].tmn != INF) t[u].tmn += v;
  t[u].tad += v;
}

void push_min(int u, int tg) {
  // 注意比较 $\max$ 标记
  if (t[u].mx <= tg) return;
  t[u].sum += (tg * 1ll - t[u].mx) * t[u].cmx;
  if (t[u].mn2 == t[u].mx) t[u].mn2 = tg;  // !!!
  if (t[u].mn == t[u].mx) t[u].mn = tg;    // !!!!!
  if (t[u].tmx > tg) t[u].tmx = tg;        // 更新取 $\max$ 标记
  t[u].mx = tg, t[u].tmn = tg;
}

void push_max(int u, int tg) {
  if (t[u].mn > tg) return;
  t[u].sum += (tg * 1ll - t[u].mn) * t[u].cmn;
  if (t[u].mx2 == t[u].mn) t[u].mx2 = tg;
  if (t[u].mx == t[u].mn) t[u].mx = tg;
  if (t[u].tmn < tg) t[u].tmn = tg;
  t[u].mn = tg, t[u].tmx = tg;
}

void pushdown(int u, int l, int r) {
  const int lu = u << 1, ru = u << 1 | 1, mid = (l + r) >> 1;
  if (t[u].tad)
    push_add(lu, l, mid, t[u].tad), push_add(ru, mid + 1, r, t[u].tad);
  if (t[u].tmx != -INF) push_max(lu, t[u].tmx), push_max(ru, t[u].tmx);
  if (t[u].tmn != INF) push_min(lu, t[u].tmn), push_min(ru, t[u].tmn);
  t[u].tad = 0, t[u].tmx = -INF, t[u].tmn = INF;
}

void build(int u = 1, int l = 1, int r = n) {
  t[u].tmn = INF, t[u].tmx = -INF;  // 取极限
  if (l == r) {
    t[u].sum = t[u].mx = t[u].mn = a[l];
    t[u].mx2 = -INF, t[u].mn2 = INF;
    t[u].cmx = t[u].cmn = 1;
    return;
  }
  int mid = (l + r) >> 1;
  build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
  pushup(u);
}

void add(int L, int R, int v, int u = 1, int l = 1, int r = n) {
  if (R < l || r < L) return;
  if (L <= l && r <= R) return push_add(u, l, r, v);  // !!! 忘 return
  int mid = (l + r) >> 1;
  pushdown(u, l, r);
  add(L, R, v, u << 1, l, mid), add(L, R, v, u << 1 | 1, mid + 1, r);
  pushup(u);
}
```

```cpp
void tomin(int L, int R, int v, int u = 1, int l = 1, int r = n) {
  if (R < l || r < L || t[u].mx <= v) return;
  if (L <= l && r <= R && t[u].mx2 < v) return push_min(u, v);  // BUG: 忘了返回
  int mid = (l + r) >> 1;
  pushdown(u, l, r);
  tomin(L, R, v, u << 1, l, mid), tomin(L, R, v, u << 1 | 1, mid + 1, r);
  pushup(u);
}

void tomax(int L, int R, int v, int u = 1, int l = 1, int r = n) {
  if (R < l || r < L || t[u].mn >= v) return;
  if (L <= l && r <= R && t[u].mn2 > v) return push_max(u, v);
  int mid = (l + r) >> 1;
  pushdown(u, l, r);
  tomax(L, R, v, u << 1, l, mid), tomax(L, R, v, u << 1 | 1, mid + 1, r);
  pushup(u);
}

long long qsum(int L, int R, int u = 1, int l = 1, int r = n) {
  if (R < l || r < L) return 0;
  if (L <= l && r <= R) return t[u].sum;
  int mid = (l + r) >> 1;
  pushdown(u, l, r);
  return qsum(L, R, u << 1, l, mid) + qsum(L, R, u << 1 | 1, mid + 1, r);
}

long long qmax(int L, int R, int u = 1, int l = 1, int r = n) {
  if (R < l || r < L) return -INF;
  if (L <= l && r <= R) return t[u].mx;
  int mid = (l + r) >> 1;
  pushdown(u, l, r);
  return max(qmax(L, R, u << 1, l, mid), qmax(L, R, u << 1 | 1, mid + 1, r));
}

long long qmin(int L, int R, int u = 1, int l = 1, int r = n) {
  if (R < l || r < L) return INF;
  if (L <= l && r <= R) return t[u].mn;
  int mid = (l + r) >> 1;
  pushdown(u, l, r);
  return min(qmin(L, R, u << 1, l, mid), qmin(L, R, u << 1 | 1, mid + 1, r));
}

int main() {
  n = rd();
  for (int i = 1; i <= n; i++) a[i] = rd();
  build();
  m = rd();
  for (int i = 1; i <= m; i++) {
    int op, l, r, x;
    op = rd(), l = rd(), r = rd();
    if (op <= 3) x = rd();  // scanf("%d",&x);
    if (op == 1)
      add(l, r, x);
    else if (op == 2)
      tomax(l, r, x);
    else if (op == 3)
      tomin(l, r, x);
    else if (op == 4)
      printf("%lld\n", qsum(l, r));
    else if (op == 5)
      printf("%lld\n", qmax(l, r));
    else
      printf("%lld\n", qmin(l, r));
  }
  return 0;
}
```

平衡树

1. 插入 x 数
2. 删除 x 数(若有多个相同的数，因只删除一个)
3. 查询 x 数的排名(排名定义为比当前数小的数的个数 $+1$)
4. 查询排名为 x 的数
5. 求 x 的前驱(前驱定义为小于 x，且最大的数)
6. 求 x 的后继(后继定义为大于 x，且最小的数)

```
// luogu-judger-enable-o2
// fhq treap


// luogu-judger-enable-o2
#include<bits/stdc++.h>
using namespace  std;
const int N = 1e5 + 10;
int root, n, a, x, y, z, op, Size, siz[N], val[N], rd[N], ch[N][3];

inline int read() {
    char ch; bool f = false; int res = 0;
    while (((ch = getchar()) < '0' || ch > '9') && ch != '-');
    if (ch == '-') f = true; else res = ch - '0';
    while ((ch = getchar()) >= '0' && ch <= '9') res = res * 10 + (ch ^ 48);
    return f? ~res + 1 : res;
}

inline void pushup (int x) {
    siz[x] = siz[ch[x][0]] + siz[ch[x][1]] + 1;
}

inline int New (int x) {
    siz[++Size] = 1, val[Size] = x, rd[Size] = rand();
    return Size;
}

void split (int p, int &x, int &y, int a) {
    if (p == 0) x = y = 0;
    else {
        if (val[p] <= a) {
            x = p;
            split(ch[p][1], ch[p][1], y, a);
        }
        else {
            y = p;
            split(ch[p][0], x, ch[p][0], a);
        }
        pushup(p);
    }
}
```

```cpp
int merge (int a, int b) {
    if (a == 0 || b == 0)
        return a + b;
    if (rd[a] < rd[b]) {
        ch[a][1] = merge(ch[a][1], b);
        pushup(a);
        return a;
    }
    else  {
        ch[b][0] = merge(a, ch[b][0]);
        pushup(b);
        return b;
    }
}

inline int kth (int x, int a) {
    while (true) {
        if (a <= siz[ch[x][0]])
            x = ch[x][0];
        else if (a == siz[ch[x][0]] + 1)
            return x;
        else a -= siz[ch[x][0]] + 1, x = ch[x][1];
    }
}

int main() {
    srand(time(0));
    n = read();
    for (int i = 1; i <= n; i++) {
        op = read(), a = read();
        if (op == 1) { // insert
            split(root, x, y, a);
            root = merge(merge(x, New(a)), y);
        }
        else if (op == 2) { // delete
            split(root, x, z, a);
            split(x, x, y, a - 1);
            y = merge(ch[y][0], ch[y][1]);
            root = merge(merge(x, y), z);
        }
        else if (op == 3) { // rank
            split(root, x, y, a - 1);
            printf("%d\n", siz[x] + 1);
            root = merge(x, y);
        }
        else if (op == 4) { // xth
            printf("%d\n", val[kth(root, a)]);
        }
        else if (op == 5) { // pred
            split(root, x, y, a - 1);
            printf("%d\n", val[kth(x, siz[x])]);
            root = merge(x, y);
        }
        else if (op == 6) { // succ
            split(root, x, y, a);
            printf("%d\n", val[kth(y, 1)]);
            root = merge(x, y);
        }
    }
    return 0;
```

```
  }
```