字符串哈希

```cpp
#define ll long long
const int N = 2e5 + 1000;
int n, m;
int id[N], tot;
char s[N];
#include<array>
struct Shash{
    const ll base[2]={29,31};
    const ll hashmod[2]={(ll)1e9,998244353};

    array<vector<ll>,2>hsh,pwMod;
    void init(string &s){
        int n=s.size();s=' '+s;
        hsh[0].resize(n+1),hsh[1].resize(n+1);
        pwMod[0].resize(n+1),pwMod[1].resize(n+1);
        for(int i=0;i<2;i++){
            pwMod[i][0]=1;
            for(int j=1;j<=n;j++){
                pwMod[i][j]=pwMod[i][j-1]*base[i]%hashmod[i];
                hsh[i][j]=(hsh[i][j-1]*base[i]+s[j])%hashmod[i];
            }
        }
    }
    pair<ll,ll>get(int l,int r){
        pair<ll,ll>ans;
        ans.fi=(hsh[0][r]-hsh[0][l-1]*pwMod[0][r-l+1])%hashmod[0];
        ans.se=(hsh[1][r]-hsh[1][l-1]*pwMod[1][r-l+1])%hashmod[1];
        ans.fi=(ans.fi+hashmod[0])%hashmod[0];
        ans.se=(ans.se+hashmod[1])%hashmod[1];
        return ans;
    }
    bool same(int la,int ra,int lb,int rb){
        return get(la,ra)==get(lb,rb);
    }
};
```

splay

P3391 【模板】文艺平衡树

(对于这题，是不需要增加数据的，只要找左端点和右端点并进行区间翻转)

您需要写一种数据结构（可参考题目标题），来维护一个有序数列。

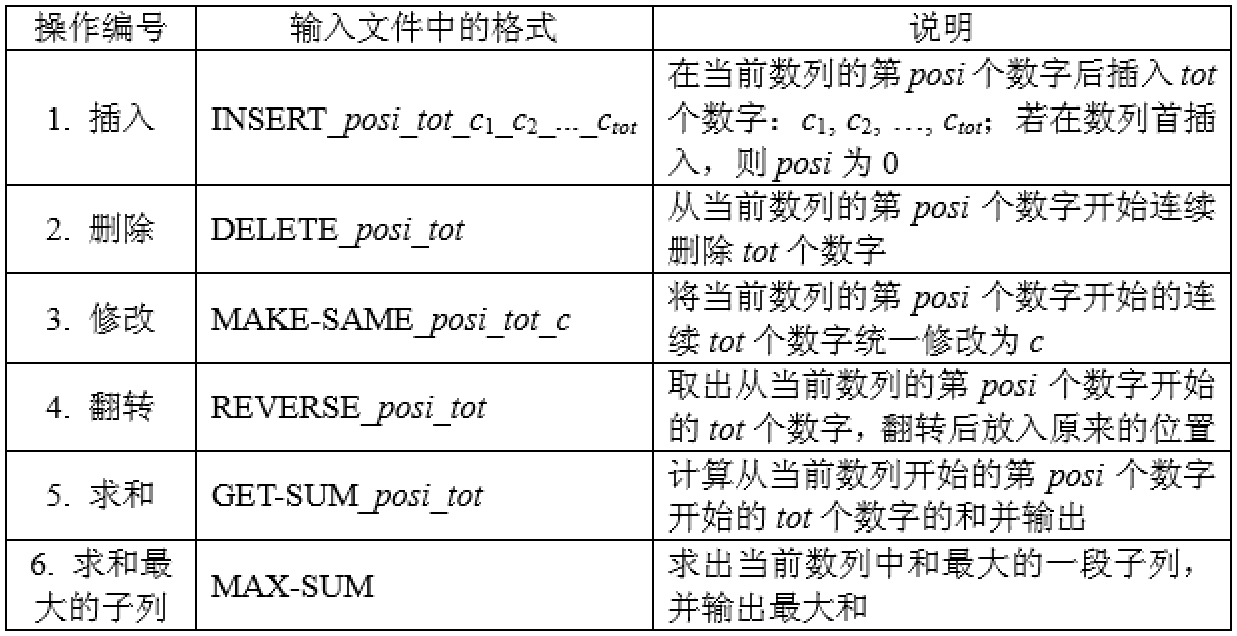
其中需要提供以下操作：翻转一个区间，例如原有序序列是 5 4 3 2 15 4 3 2 1，翻转区间是 [2,4][2,4] 的话，结果是 5 2 3 4 15 2 3 4 1。

输入：第一行两个正整数 *n*,*m*，表示序列长度与操作个数。序列中第 *i* 项初始为 *i*。
接下来 *m* 行，每行两个正整数 *l*,*r*，表示翻转的区间。

输出：输出一行 *n* 个正整数，表示原始序列经过 *m* 次变换后的结果。

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 2e5 + 1000;
int n,m;
int id[N], tot;//id主要是更改
namespace Splay {
    struct node {
        int s[2], fa, siz;
        bool rev;
    }t[N];
    #define ls(x) (t[x].s[0])
    #define rs(x) (t[x].s[1])
    #define fa(x) (t[x].fa)
    int root;
    void pushup(int x) { t[x].siz = t[ls(x)].siz + t[rs(x)].siz + 1; }
    void pushdown(int x) { if (t[x].rev) t[ls(x)].rev ^= 1, t[rs(x)].rev ^= 1,
swap(ls(x), rs(x)), t[x].rev = 0; }
    bool get(int x) { return rs(fa(x)) == x; }
    void rotate(int x) {
        int y = fa(x), z = fa(y); bool o = get(x);
        if (t[x].s[o ^ 1]) fa(t[x].s[o ^ 1]) = y;
        t[y].s[o] = t[x].s[o ^ 1], t[x].s[o ^ 1] = y;
        if (z) t[z].s[get(y)] = x; fa(x) = z, fa(y) = x;
        pushup(y), pushup(x);
    }
    void splay(int x, int y = 0) {
        for (int f; (f = fa(x)) != y; rotate(x))
            if (fa(f) != y) rotate(get(x) == get(f) ? f : x);
        if (!y) root = x;
    }
    int build(int l, int r) {
        int mid = (l + r) >> 1; t[mid] = {{0, 0}, 0, 1, 0};
        if (l < mid) ls(mid) = build(l, mid - 1), fa(ls(mid)) = mid;
        if (mid < r) rs(mid) = build(mid + 1, r), fa(rs(mid)) = mid;
        pushup(mid); return mid;
    }
    int find(int k) {
        int now = root;
        while (now && k) {pushdown(now);
            int sz = t[ls(now)].siz + 1;
            if (sz > k) now = ls(now);
            else if (sz == k) return now;
            else k -= sz, now = rs(now);
        } return 1;
    }
    void reverse(int l, int r) { ++l, ++r;
        int x = find(l - 1); splay(x, 0);
        int y = find(r + 1); splay(y, x);
        t[ls(y)].rev ^= 1;
    }
    void out(int x) {
        if (!x) return;
        pushdown(x);
        out(ls(x));
        if (x > 1 && x < n + 2) id[++tot] = x - 1;
        out(rs(x));
    }
}
signed main(){
    cin>>n>>m;
    Splay::root = Splay::build(1,n+2);
    while(m--){
        int u,v;cin>>u>>v;
        Splay::reverse(u, v);
    }
    Splay::out(Splay::root);
    for(int i=1;i<=n;i++)cout<<id[i]<<" ";
}
```

关于splay的其他操作



操作编号	输入文件中的格式	说明
1. 插入	INSERT_$posi$_tot_c_1_c_2_..._c_{tot}	在当前数列的第 $posi$ 个数字后插入 tot 个数字：c_1, c_2, …, c_{tot}；若在数列首插入，则 $posi$ 为 0
2. 删除	DELETE_$posi$_tot	从当前数列的第 $posi$ 个数字开始连续删除 tot 个数字
3. 修改	MAKE-SAME_$posi$_tot_c	将当前数列的第 $posi$ 个数字开始的连续 tot 个数字统一修改为 c
4. 翻转	REVERSE_$posi$_tot	取出从当前数列的第 $posi$ 个数字开始的 tot 个数字，翻转后放入原来的位置
5. 求和	GET-SUM_$posi$_tot	计算从当前数列开始的第 $posi$ 个数字开始的 tot 个数字的和并输出
6. 求和最大的子列	MAX-SUM	求出当前数列中和最大的一段子列，并输出最大和

代码

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
const int N = 500010, INF = 1e9;
int n, m;
```

```
struct Node {
    int s[2], p, v;
    int rev, same;
    int size, sum, ms, ls, rs;

    void init(int _v, int _p)
    {
        s[0] = s[1] = 0, p = _p, v = _v;
        rev = same = 0;
        size = 1, sum = ms = v;
        ls = rs = max(v, 0);
    }
} tr[N];
int root, nodes[N], tt;
int w[N];
void pushup(int x)
{
    auto &u = tr[x], &l = tr[u.s[0]], &r = tr[u.s[1]];
    u.size = l.size + r.size + 1;
    u.sum = l.sum + r.sum + u.v;
    u.ls = max(l.ls, l.sum + u.v + r.ls);
    u.rs = max(r.rs, r.sum + u.v + l.rs);
    u.ms = max(max(l.ms, r.ms), l.rs + u.v + r.ls);
}
void pushdown(int x)
{
    auto &u = tr[x], &l = tr[u.s[0]], &r = tr[u.s[1]];
    if (u.same) {
        u.same = u.rev = 0;
        if (u.s[0])
            l.same = 1, l.v = u.v, l.sum = l.v * l.size;
        if (u.s[1])
            r.same = 1, r.v = u.v, r.sum = r.v * r.size;
        if (u.v > 0) {
            if (u.s[0])
                l.ms = l.ls = l.rs = l.sum;
            if (u.s[1])
                r.ms = r.ls = r.rs = r.sum;
        } else {
            if (u.s[0])
                l.ms = l.v, l.ls = l.rs = 0;
            if (u.s[1])
                r.ms = r.v, r.ls = r.rs = 0;
        }
    } else if (u.rev) {
        u.rev = 0, l.rev ^= 1, r.rev ^= 1;
        swap(l.ls, l.rs), swap(r.ls, r.rs);
        swap(l.s[0], l.s[1]), swap(r.s[0], r.s[1]);
    }
}
void rotate(int x)
{
    int y = tr[x].p, z = tr[y].p;
    int k = tr[y].s[1] == x;
    tr[z].s[tr[z].s[1] == y] = x, tr[x].p = z;
    tr[y].s[k] = tr[x].s[k ^ 1], tr[tr[x].s[k ^ 1]].p = y;
    tr[x].s[k ^ 1] = y, tr[y].p = x;
    pushup(y), pushup(x);
}
void splay(int x, int k)
{
    while (tr[x].p != k) {
        int y = tr[x].p, z = tr[y].p;
        if (z != k)
            if ((tr[y].s[1] == x) ^ (tr[z].s[1] == y))
                rotate(x);
            else
                rotate(y);
        rotate(x);
    }
    if (!k)
        root = x;
}
int get_k(int k)
{
    int u = root;
    while (u) {
        pushdown(u);
        if (tr[tr[u].s[0]].size >= k)
            u = tr[u].s[0];
        else if (tr[tr[u].s[0]].size + 1 == k)
            return u;
        else
            k -= tr[tr[u].s[0]].size + 1, u = tr[u].s[1];
    }
}

int build(int l, int r, int p)
{
    int mid = l + r >> 1;
    int u = nodes[tt--];
    tr[u].init(w[mid], p);
    if (l < mid)
        tr[u].s[0] = build(l, mid - 1, u);
    if (mid < r)
        tr[u].s[1] = build(mid + 1, r, u);
    pushup(u);
    return u;
}
void dfs(int u)
{
    if (tr[u].s[0])
        dfs(tr[u].s[0]);
    if (tr[u].s[1])
```

```
        dfs(tr[u].s[1]);
    nodes[++tt] = u;
}
int main()
{
    for (int i = 1; i < N; i++)
        nodes[++tt] = i;
    scanf("%d%d", &n, &m);
    tr[0].ms = w[0] = w[n + 1] = -INF;
    for (int i = 1; i <= n; i++)
        scanf("%d", &w[i]);
    root = build(0, n + 1, 0);
    char op[20];
    while (m--) {
        scanf("%s", op);
        if (!strcmp(op, "INSERT")) {
            int posi, tot;
            scanf("%d%d", &posi, &tot);
            for (int i = 0; i < tot; i++)
                scanf("%d", &w[i]);
            int l = get_k(posi + 1), r = get_k(posi + 2);
            splay(l, 0), splay(r, l);
            int u = build(0, tot - 1, r);
            tr[r].s[0] = u;
            pushup(r), pushup(l);
        } else if (!strcmp(op, "DELETE")) {
            int posi, tot;
            scanf("%d%d", &posi, &tot);
            int l = get_k(posi), r = get_k(posi + tot + 1);
            splay(l, 0), splay(r, l);
            dfs(tr[r].s[0]);
            tr[r].s[0] = 0;
            pushup(r), pushup(l);
        } else if (!strcmp(op, "MAKE-SAME")) {
            int posi, tot, c;
            scanf("%d%d%d", &posi, &tot, &c);
            int l = get_k(posi), r = get_k(posi + tot + 1);
            splay(l, 0), splay(r, l);
            auto& son = tr[tr[r].s[0]];
            son.same = 1, son.v = c, son.sum = c * son.size;
            if (c > 0)
                son.ms = son.ls = son.rs = son.sum;
            else
                son.ms = c, son.ls = son.rs = 0;
            pushup(r), pushup(l);
        } else if (!strcmp(op, "REVERSE")) {
            int posi, tot;
            scanf("%d%d", &posi, &tot);
            int l = get_k(posi), r = get_k(posi + tot + 1);
            splay(l, 0), splay(r, l);
            auto& son = tr[tr[r].s[0]];
            son.rev ^= 1;
            swap(son.ls, son.rs);
            swap(son.s[0], son.s[1]);
            pushup(r), pushup(l);
        } else if (!strcmp(op, "GET-SUM")) {
            int posi, tot;
            scanf("%d%d", &posi, &tot);
            int l = get_k(posi), r = get_k(posi + tot + 1);
            splay(l, 0), splay(r, l);
            printf("%d\n", tr[tr[r].s[0]].sum);
        } else
            printf("%d\n", tr[root].ms);
    }

    return 0;
}

```