文艺平衡树

题目链接

P3391 【模板】文艺平衡树

对于一个数组，操作为对于l，r两区间内的数进行翻转操作

完整代码

```
#include <bits/stdc++.h>
#define endl '\n'

using namespace std;

typedef long long ll;
typedef vector<int> VI;
typedef pair<int, int> PII;

const int maxn = 1e5 + 5;
const ll mod = 1e9 + 7;

struct Node
{
    int val, key;
    int left, right;
    int size;
    bool lazy;
};

int root, cnt;
Node tree[maxn];

int newNode(int val)
{
    tree[++cnt].val = val;
    tree[cnt].key = rand();
    tree[cnt].left = tree[cnt].right = 0;
    tree[cnt].size = 1;
    tree[cnt].lazy = false;
    return cnt;
}

void update(int now)
{
    tree[now].size = tree[tree[now].left].size + tree[tree[now].right].size + 1;
}

void push_down(int now)
{
    swap(tree[now].left, tree[now].right);
    tree[tree[now].left].lazy ^= 1;
```

```
    tree[tree[now].right].lazy ^= 1;
    tree[now].lazy = false;
}

void split(int now, int size, int& ls, int& rs)
{
    if (!now)
    {
        ls = rs = 0;
        return;
    }
    if (tree[now].lazy)
        push_down(now);
    if (tree[tree[now].left].size < size)
    {
        ls = now;
        split(tree[now].right, size - tree[tree[now].left].size - 1,
tree[now].right, rs);
    }
    else
    {
        rs = now;
        split(tree[now].left, size, ls, tree[now].left);
    }
    update(now);
}

int merge(int ls, int rs)
{
    if (!ls || !rs)
        return ls + rs;
    if (tree[ls].key < tree[rs].key)
    {
        if (tree[ls].lazy)
            push_down(ls);
        tree[ls].right = merge(tree[ls].right, rs);
        update(ls);
        return ls;
    }
    else
    {
        if (tree[rs].lazy)
            push_down(rs);
        tree[rs].left = merge(ls, tree[rs].left);
        update(rs);
        return rs;
    }
}

void reverse(int l, int r)
{
    int ls, ms, rs;
    split(root, l - 1, ls, ms);
    split(ms, r - l + 1, ms, rs);
    tree[ms].lazy ^= 1;
```

```
    root = merge(merge(ls, ms), rs);
}

void mediumOrder(int now)//中序遍历
{
    if (!now)
        return;
    if (tree[now].lazy)
        push_down(now);
    mediumOrder(tree[now].left);
    cout << tree[now].val << " ";
    mediumOrder(tree[now].right);
}

void solve()
{
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i)
        root = merge(root, newNode(i));
    for (int i = 0; i < m; ++i)
    {
        int l, r;
        cin >> l >> r;
        reverse(l, r);
    }
    mediumOrder(root);
}

int main()
{
    int T = 1;
    while (T--)
        solve();
    return 0;
}
```