珂朵莉树

简介

珂朵莉树(Chthlly Tree)也叫老司机树，ODT(Old Driver Tree)。

只要是由区间赋值操作的数据结构都可以使用这种数据结构来骗分，在数据随机的情况下效率很高。对于add,assign和sum操作，用set实现的珂朵莉树复杂度位O(nloglogn)。

问题起源

[CF 896C]n个数m次操作：

①区间加一个数、

②区间赋值

③求区间第k小

④求区间幂次和

核心思想

把值相同的区间合并成一个节点保存在set里面。

操作

保存节点

用一个三元组(L,R,val)来表示，并用set维护。

```
struct Node{
    int l,r;
    mutable int v; //mutable关键字可以让我们在后面的操作中修改v
    Node(const int &il,const int &ir,const int &iv):l(il),r(ir),v(iv){}
    inline bool operator<(const Node &o)const{
        return l<o.l;
    }
}
set<Node>odt;
typedef set<Node>::iterator iter
```

split

将原本包含x的区间[l,r]分裂为[l,x)和[x,r]两个区间并返回指向后者的迭代器。

```
iter split(int x){
    if(x>n) return odt.end();
    auto it=--odt.upper_bound((Node){x,0,0});
    if(it->l==x) return it;
    int l=it->l,r=it->r,v=it->v;
    odt.erase(it);
    odt.insert(Node(l,x-1,v));
    return odt.insert(Node(x,r,v)).first;
}
```

那么对于所有[l,r]上的区间操作，都可以转化成set上[split(l),split(r+1)]的操作。

assign

用于对一段区间赋值，使ODT的大小下降。

```
void assign(int l,int r,int v){
    auto itr=split(r+1),itl=split(l);
    odt.erase(itl,itr);
    odt.insert(Node(l,r,v));
}
```

区间加

```
void add(int l,int r,ll val=1){
    split(l);
    auto itr=split(r+1),itl=split(l);
    for(;itl!=itr;++itl) itl->v+=val;
}
```

求第k小

```
ll rank(int l,int r,int k,bool reversed=0){
    if(reversed) k=r-l+2-k;
    split(l);
    auto itr=split(r+1),itl=split(l);
    vector<pair<int,int>>vp;
    for(;itl!=itr;++itl) vp.push_back({itl->v,itl->r-itl->l+1});
    sort(vp.begin(),vp.end());
    for(auto i:vp){
        k-=i.second;
        if(k<=0) return i.first;
    }
    return -1;
}
```

求幂次和

```
ll sum(int l,int r,int ex,int mod){
    split(l);
    auto itr=split(r+1),itl=split(l);
    ll res=0;
    for(;itl!=itr;++itl)
        res=(res+(ll)(itl->r-itl->l+1)*pow(itl->v,ex))%mod;
    return res;
}
```

其他操作

套如下模板

```
void performance(int l,int r){
    auto itr=split(r+1), itl=split(l);
    for(;itl!=itr;++itl){
         //操作主体
    }
}
```

珂朵莉树在进行求取区间左右端点操作时，必须先split右端点，再split左端点。否则返回的迭代器可能会再split右端点失效，导致RE。

例题

Codeforces [896C - Willem, Chtholly and Seniorious]

珂朵莉数的起源题，数据随机，支持以下操作：

- 区间加法
- 区间赋值
- 询问区间第K大
- 询问区间N次方和

```
#include<bits/stdc++.h>
#define IT set<node>::iterator
using namespace std;
typedef long long LL;
const int MOD7 = 1e9 + 7;
const int MOD9 = 1e9 + 9;
const int imax_n = 1e5 + 7;
struct node
{
    int l,r;
    mutable LL v;
    node(int L, int R=-1, LL V=0):l(L), r(R), v(V) {}
    bool operator<(const node& o) const
    {
        return l < o.l;
    }
};
LL pown(LL a, LL b, LL mod)
{
    LL res = 1;
    LL ans = a % mod;
    while (b)
    {
        if (b&1)
            res = res * ans % mod;
        ans = ans * ans % mod;
        b>>=1;
    }
    return res;
}
set<node> s;
IT split(int pos)
{
    IT it = s.lower_bound(node(pos));
    if (it != s.end() && it->l == pos)
        return it;
    --it;
    int L = it->l, R = it->r;
    LL V = it->v;
    s.erase(it);
    s.insert(node(L, pos-1, V));
    return s.insert(node(pos, R, V)).first;
}
void add(int l, int r, LL val)
{
    IT itr = split(r+1),itl = split(l);
    for (; itl != itr; ++itl)
        itl->v += val;
```

```
}
void assign_val(int l, int r, LL val)
{
    IT itr = split(r+1),itl = split(l);
    s.erase(itl, itr);
    s.insert(node(l, r, val));
}
LL ranks(int l, int r, int k)
{
    vector<pair<LL, int> > vp;
    IT itr = split(r+1),itl = split(l);
    vp.clear();
    for (; itl != itr; ++itl)
        vp.push_back(pair<LL,int>(itl->v, itl->r - itl->l + 1));
    sort(vp.begin(), vp.end());
    for (vector<pair<LL,int> >::iterator it=vp.begin();it!=vp.end();++it)
    {
        k -= it->second;
        if (k <= 0)
            return it->first;
    }
}
LL sum(int l, int r, int ex, int mod)
{
    IT itr = split(r+1),itl = split(l);
    LL res = 0;
    for (; itl != itr; ++itl)
        res = (res + (LL)(itl->r - itl->l + 1) * pown(itl->v, LL(ex), LL(mod)))
% mod;
    return res;
}
int n, m;
LL seed, vmax;
LL rd()
{
    LL ret = seed;
    seed = (seed * 7 + 13) % MOD7;
    return ret;
}
LL a[imax_n];
int main()
{
    cin>>n>>m>>seed>>vmax;
    for (int i=1; i<=n; ++i)
    {
        a[i] = (rd() % vmax) + 1;
        s.insert(node(i,i,a[i]));
    }
    s.insert(node(n+1, n+1, 0));
    int lines = 0;
    for (int i =1; i <= m; ++i)
    {
        int op = int(rd() % 4) + 1;
        int l = int(rd() % n) + 1;
        int r = int(rd() % n) + 1;
        if (l > r)
            swap(l,r);
        int x, y;
```

```
        if (op == 3)
            x = int(rd() % (r-l+1)) + 1;
        else
            x = int(rd() % vmax) +1;
        if (op == 4)
            y = int(rd() % vmax) + 1;
        if (op == 1)
            add(l, r, LL(x));
        else if (op == 2)
            assign_val(l, r, LL(x));
        else if (op == 3)
            cout<<ranks(l,r,x)<<endl;
        else
            cout<<sum(l,r,x,y)<<endl;
    }
    return 0;
}
```

[SCOI2010] 序列操作

给定一个01序列，支持以下操作：

- 0 l r ,把$[L,R]$区间内所有数全变成0
- 1 l r ,把$[L,R]$区间内所有数全变成1
- 2 l r ,把$[L,R]$区间内所有数全部取反
- 3 l r ,询问$[L,R]$区间内1的个数
- 4 l r ,询问$[L,R]$区间内最多有多少个连续的1

但是这道题不保证数据随机，加强之后就不能这样搞了。

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

struct nod{
    int l,r;
    mutable bool v;
    nod(int L,int R=-1,bool V=0):l(L),r(R),v(V){}
    bool operator<(const nod& o)const{
        return l<o.l;
    }
};
set<nod>s;
inline auto split(int pos){
    auto it=s.lower_bound(nod(pos));
    if(it!=s.end()&&it->l==pos) return it;
    --it;
    int L=it->l,R=it->r;
    bool V=it->v;
    s.erase(it);
    s.insert(nod(L,pos-1,V));
    return s.insert(nod(pos,R,V)).first;
}
inline void assign_val(int l,int r,bool val){
    auto itr=split(r+1),itl=split(l);
    s.erase(itl,itr);
    s.insert(nod(l,r,val));
```

```
}
inline void rev(int l,int r){
    auto itr=split(r+1),itl=split(l);
    for(;itl!=itr;++itl) (itl->v)^=1;
}
inline int sum(int l,int r){
    auto itr=split(r+1),itl=split(l);
    int res=0;
    for(;itl!=itr;++itl) res+=(itl->v)?(itl->r)-(itl->l)+1:0;
    return res;
}
inline int count(int l,int r) {
    int res=0,temp=0;
    auto itr=split(r+1),itl=split(l);
    for(;itl!=itr;++itl){
        if((itl->v)==false){
            res=max(res,temp);
            temp=0;
        }else temp+=(itl->r)-(itl->l)+1;
    }
    return max(res,temp);
}
signed main() {
    int n,m,op,L,R;
    scanf("%d%d",&n,&m);
    for(int i=0,x;i<n;++i) scanf("%d",&x),s.insert(nod(i,i,x));
    s.insert(nod(n,n,0));
    while(m--){
        scanf("%d%d%d",&op,&L,&R);
        if(op==0) assign_val(L,R,0);
        else if(op==1) assign_val(L,R,1);
        else if(op==2) rev(L,R);
        else if(op==3) printf("%d\n",sum(L,R));
        else printf("%d\n",count(L,R));
    }
    return 0;
}
```

参考资料

oi-wiki

https://www.bilibili.com/video/BV1QW411T7nz

https://www.cnblogs.com/yzhang-rp-inf/p/9443659.html