NOIPmoni1题解

江南游

$dp[cnt,u,k]$表示经过cnt条水道，目前在u城镇，经过x城镇次数奇偶性与k相同,k = 0/1 对于一条水道(u,v),

1. v $\neq$ k $dp[cnt+1,v,k] = (dp[cnt,u,k] + dp[cnt+1,v,k])\%mod$
2. v $=$ k $dp[cnt+1,v,!k] = (dp[cnt,u,k] + dp[i+1,v,!k])\%mod$ 初始条件: $dp[0,S,0] = 1$

```
struct Edge {
 int to;
 int next;
}e[M];
```

int head[N], idx;

void add(int a, int b) { idx++; e[idx].to = b; e[idx].next = head[a]; head[a] = idx; }

int f[N][N][2];

int n, m, k, s, t, x;

int main() { ios::sync_with_stdio(false); cin.tie(nullptr);

```
    cin >> n >> m >> k >> s >> t >> x;

    for (int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }

    f[0][s][0] = 1;

    for (int i = 1; i <= k; i++) {
        for (int j = 1; j <= n; j++) {
            for (int l = head[j]; l; l = e[l].next) {
                int to = e[l].to;
                for (int r = 0; r <= 1; r++) {
                    if (j == x) f[i][j][r] = (f[i][j][r] + f[i - 1][to][1 - r]) % mod;
                    else f[i][j][r] = (f[i][j][r] + f[i - 1][to][r]) % mod;
                }
            }
        }
    }
    cout << f[k][t][0] << '\n';
    return 0;
```

}

狐狸与葫芦
葫芦是个栈，然后炼化是弹出栈顶，然后这个栈的进栈这个操作类似单调栈。
一只妖要么被一个“元妖”炼化出去，要么留到最后也没被炼化。我们记录$nxt[i]$为炼化妖i时正在进入的那只妖。对于区间$[L,R]$,第一只“元妖”是编号为L,下一只是$nxt[L]$,下下只是$nxt[nxt[L]]$,……,,最终跳到区间外或区间末尾，此时查询为$O(n)$,需要优化。
考虑预处理出nxt后倍增，设$nxt[i][j]$代表从j跳了2^i到的地方，预处理$O(nlogn)$,查询$O(logn)$
```cpp
const int N = 500005;
struct P{ int a, b; } f[N];

int n,q;
int nxt[21][N];

int main(){
    read(n); read(q);
    for (int i = 1; i <= n; ++i) read(f[i].a);
    for (int i = 1; i <= n; ++i) read(f[i].b);

    static pair<P, int> s[500005]; int p = 0;
    for (int i = 1; i <= n; ++i) {
        while(p && (s[p].first.a == f[i].a || s[p].first.b <= f[i].b)) {
            nxt[0][s[p].second] = i;
            --p;
        }
        s[++p] = {f[i], i};
    }

    for (int i = 1; i <= 20; ++i) {
        for (int j = 1; j <= n; ++j)
            nxt[i][j] = nxt[i - 1][nxt[i - 1][j]]; // 处理倍增
    }

    while (q--){
        int cnt = 0;
        int l, r;
        read(l); read(r);
        for (int i = 20; i >= 0; --i) {
            if (nxt[i][l] && nxt[i][l] <= r) { // 向右跳，但不越过边界
                cnt += 1 << i; // 跳了 2^i 次
                l = nxt[i][l];
            }
        }
        printf("%d\n", cnt + 1); // 要算上 L 本身
    }
    return 0;
}
```

单调栈预处理pre_i,表示妖i在$[pre_{i+1},i]$上都是“元妖”,然后求出$[l,r]$有多少pre_i小于l,(求区间小于l的数量)使用莫队维护，需要卡一下常数

```
int n, Q;
int a[500005], b[500005], p[500005];
int s[500005], top = 0;
int bl[500005];
struct qry{
    int l, r, id;
    friend bool operator < (const qry &x, const qry &y) {
        if (bl[x.l] != bl[y.l])
            return bl[x.l] < bl[y.l];
        if (bl[x.l] % 2 == 0)
            return x.r > y.r;
        return x.r < y.r;
    }
}q[500005];
int ans[500005], c[500005];
int head = 1, tail = 0;

int l[1005], r[1005], sum[1005], pos[500005];

void buildblock(){
    int k = sqrt(n);
    for (int i = 1; i <= k; i++)
        l[i] = r[i - 1] + 1, r[i] = r[i - 1] + k;
    if (r[k] < n)
        l[k + 1] = r[k] + 1, r[k + 1] = n, k++;
    for (int i = 1; i <= k; i++)
        for (int j = l[i]; j <= r[i]; j++)
            pos[j] = i;
}

int query(int p){
    int P = 0, ans = 0;
    for (int i = 1; i < pos[p]; i++)
        ans += sum[i];
    for (int i = l[pos[p]]; i <= p; i++)
        ans += c[i];
    return ans;
}

void mdf(int x, int v){
    c[p[x]] += v;
    sum[pos[p[x]]] += v;
}

signed main() {
    n = read(), Q = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    for (int i = 1; i <= n; i++)
        b[i] = read();
```

```
    for (int i = 1; i <= n; i++){
        while (top > 0 && !(a[i] != a[s[top]] && b[i] < b[s[top]]))
            top--;
        p[i] = s[top] + 1;
        s[++top] = i;
    }
    buildblock();
    int t = sqrt(n);
    for (int i = 1; i <= n; i++)
        bl[i] = (i - 1) / t + 1;
    for (int i = 1; i <= Q; i++)
        q[i].l = read(), q[i].r = read(), q[i].id = i;
    sort(q + 1, q + Q + 1);
    for (int i = 1; i <= Q; i++){
        while (tail < q[i].r)
            mdf(++tail, 1);
        while (tail > q[i].r)
            mdf(tail--, -1);
        while (head < q[i].l)
            mdf(head++, -1);
        while (head > q[i].l)
            mdf(--head, 1);
        ans[q[i].id] = query(q[i].l);
    }
    for (int i = 1; i <= Q; i++)
        printf("%d\n", ans[i]);
    return 0;
}
```

求区间小于l的数量还可以离线下来使用树状数组

```
void slove() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++)cin >> p[i].a;
    for (int i = 1; i <= n; i++)cin >> p[i].b;
    for (int i = 1; i <= n; i++)p[i].id = i;
    stack<node>stk;
    for (int i = 1; i <= n; i++) {
        while (stk.size() && (stk.top().a == p[i].a || stk.top().b <= p[i].b)) {
            stk.pop();
        }
        s[i] = stk.size() ? stk.top().id : 0;
        stk.push(p[i]);
    }
    for (int i = 1; i <= n; i++)s[i] += 1;//注意这个加1
    for (int i = 1; i <= m; i++) {
        int L, R; cin >> L >> R;
        g[L - 1].push_back({ -1,s[L],i });//把询问拆分
        g[R].push_back({ 1,s[L],i });
    }
    for (int i = 1; i <= n; i++) {
        update(s[i], 1);//update为树状数组的单点加
        for (auto v : g[i]) {
            ans[v.id] += v.k * query(v.x);//query为树状数组的前缀和
        }
    }
    for (int i = 1; i <= m; i++)cout << ans[i] << endl;
}
```

还可以使用归并树(自己去了解)

```
void Build(int de,int l,int r){
    if(l==r){
        f[de][l]=c[l];
        return;
    }
    int mid=(l+r)>>1;
    Build(de+1,l,mid);
    Build(de+1,mid+1,r);
    for(int i=l,j=mid+1,k=l;i<=mid||j<=r;){
        if(j>r) f[de][k++]=f[de+1][i++];
        else if(i>mid||f[de+1][i]>f[de+1][j]) f[de][k++]=f[de+1][j++];
        else f[de][k++]=f[de+1][i++];
    }
}//建树
int calc(int de,int L,int R,int l,int r,int x){
    if(L>=l&&R<=r) return lower_bound(f[de]+L,f[de]+R+1,x)-f[de]-L;
    int mid=(L+R)>>1,ans=0;
    if(mid>=l) ans+=calc(de+1,L,mid,l,r,x);
    if(mid<r) ans+=calc(de+1,mid+1,R,l,r,x);
    return ans;
}//模板部分
signed main(){
    //freopen("stack.in","r",stdin);
    //freopen("stack.out","w",stdout);
    int n,q;
    cin>>n>>q;
    for(int i=1;i<=n;i++) cin>>a[i].fi;
    for(int i=1;i<=n;i++) cin>>a[i].se;
    stack<pair<int,int> > s,s1;
    s.push({0,0});
    s1.push({0,0});
    for(int i=1;i<=n;i++){
        be:
        c[i]=s.top().fi;
        if(s.size()>1&&!(s.top().se!=a[i].fi&&s1.top().se>a[i].se)){
            s.pop();
            s1.pop();
            goto be;
        }
        s.push({i,a[i].fi});
        s1.push({i,a[i].se});
    }//模拟
    Build(1,1,n);
    while(q--){
        int x,y;
        cin>>x>>y;
        cout<<calc(1,1,n,x,y,x)<<endl;//求出 x-y 区间内数字小于 x 的数量
    }
}
```

还可以使用主席树(自己了解)

```cpp
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
int read()
{
    char c=getchar();int x=0;bool f=0;
    for(;!isdigit(c);c=getchar())f^=!(c^45);
    for(;isdigit(c);c=getchar())x=(x<<1)+(x<<3)+(c^48);
    if(f)x=-x;return x;
}
int n,m,p,t,a[500001],b[500001],c[500001],rt[500001];
struct tree
{
    int l,r,s;
}T[30000001];
void pushup(int x)
{
    T[x].s=T[T[x].l].s+T[T[x].r].s;
}
void build(int &x,int l,int r)
{
    x=++p;
    if(l==r) return;
    int z=l+r>>1;
    build(T[x].l,l,z);
    build(T[x].r,z+1,r);
}
void modify(int &x,int l,int r,int q)
{
    T[++p]=T[x];
    ++T[p].s;
    x=p;
    if(l==r) return;
    int z=l+r>>1;
    if(q<=z) modify(T[x].l,l,z,q);
    else modify(T[x].r,z+1,r,q);
}
int num(int x,int l,int r,int k)
{
    if(l==r) return 0;
    int z=l+r>>1;
    if(k==z) return T[T[x].l].s;
    if(k<z) return num(T[x].l,l,z,k);
    return T[T[x].l].s+num(T[x].r,z+1,r,k);
}
int main()
{
    freopen("stack.in","r",stdin);
    freopen("stack.out","w",stdout);
```

```
    n=read(),m=read();
    for(int i=1;i<=n;++i) a[i]=read();
    for(int i=1;i<=n;++i) b[i]=read();
    a[0]=0;
    b[0]=1e9;
    c[0]=0;
    build(rt[0],0,n);
    for(int i=1;i<=n;++i)
    {
        int l=0,r=t,z;
        while(l<r)
        {
            z=(l+r+1)>>1;
            if(b[i]<b[c[z]]) l=z;
            else r=z-1;
        }
        while(a[c[l]]==a[i]) --l;
        t=l+1;
        c[t]=i;
        rt[i]=rt[i-1];
        modify(rt[i],0,n,c[l]);
    }
    for(int i=1;i<=m;++i)
    {
        int l=read(),r=read();
        printf("%d\n",num(rt[r],0,n,l-1)-num(rt[l-1],0,n,l-1));
    }
    return 0;
}
```

五彩路

正常情况下路径长求法为$dis_u + dis_v - 2 \times dis_{LCA(u,v)}$ 考虑变化颜色带来的影响，只需记录一个点到根每种颜色出现次数cnt_i和每种颜色的边的长度和l_i， 那么变颜色x为y长度后的结果为$dis'_i = dis_i - l_i + cnt_i \times y$ 如何维护？将边权和边的颜色转到点上，树链剖分，对于每种颜色建立一棵动态开点线段树，储存所有该颜色的边的信息。

```cpp
#include<bits/stdc++.h>
using namespace std;
#define mid ((l+r)>>1)
int n,m,val[100100],col[100100],root[100100],tsz,dis[100100];
int
dfn[100100],rev[100100],fa[100100],dep[100100],son[100100],top[100100],sz[100100],head
[100100],cnt,tot;
struct node{
    int to,next,val,col;
}edge[200100];
void ae(int u,int v,int c,int w){

edge[cnt].next=head[u],edge[cnt].to=v,edge[cnt].val=w,edge[cnt].col=c,head[u]=cnt++;

edge[cnt].next=head[v],edge[cnt].to=u,edge[cnt].val=w,edge[cnt].col=c,head[v]=cnt++;
}
void dfs1(int x,int Fa){
    fa[x]=Fa,dep[x]=dep[Fa]+1,sz[x]=1;
    for(int i=head[x],y;i!=-1;i=edge[i].next){
        if((y=edge[i].to)==fa[x])continue;

dis[y]=dis[x]+edge[i].val,dfs1(y,x),sz[x]+=sz[y],val[y]=edge[i].val,col[y]=edge[i].col
;
        if(sz[son[x]]<sz[y])son[x]=y;
    }
}
void dfs2(int x){
    if(son[x])top[son[x]]=top[x],dfn[son[x]]=++tot,rev[tot]=son[x],dfs2(son[x]);
    for(int i=head[x],y;i!=-1;i=edge[i].next){
        y=edge[i].to;
        if(y==fa[x]||y==son[x])continue;
        top[y]=y,dfn[y]=++tot,rev[tot]=y,dfs2(y);
    }
}
struct SegTree{
    int lson,rson,sum,sz;
    SegTree(){lson=rson=sum=sz=0;}
    friend SegTree operator +(const SegTree &x,const SegTree &y){
        SegTree z;
        z.sum=x.sum+y.sum;
        z.sz=x.sz+y.sz;
        return z;
    }
}seg[20001000];
void modify(int &x,int l,int r,int P,int val){
    if(!x)x=++tsz;
    seg[x].sum+=val,seg[x].sz++;
    if(l==r)return;
    if(mid>=P)modify(seg[x].lson,l,mid,P,val);
    else modify(seg[x].rson,mid+1,r,P,val);
```

```
}
SegTree query(int x,int l,int r,int L,int R){
    if(!x)return SegTree();
    if(l>R||r<L)return SegTree();
    if(L<=l&&r<=R)return seg[x];
    return query(seg[x].lson,l,mid,L,R)+query(seg[x].rson,mid+1,r,L,R);
}
int ask(int x,int y,int c,int w){
    int X=x,Y=y;
    SegTree res;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        res=res+query(root[c],1,n,dfn[top[x]],dfn[x]),x=fa[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    if(x!=y)res=res+query(root[c],1,n,dfn[x]+1,dfn[y]);
    return dis[X]+dis[Y]-2*dis[x]+w*res.sz-res.sum;
}
int main(){
    scanf("%d%d",&n,&m),memset(head,-1,sizeof(head));
    for(int i=1,a,b,c,d;i<n;i++)scanf("%d%d%d%d",&a,&b,&c,&d),ae(a,b,c,d);
    dfs1(1,0),dfn[1]=rev[1]=tot=top[1]=1,dfs2(1);
    for(int i=2;i<=n;i++)modify(root[col[i]],1,n,dfn[i],val[i]);
    for(int
i=1,a,b,c,d;i<=m;i++)scanf("%d%d%d%d",&a,&b,&c,&d),printf("%d\n",ask(c,d,a,b));
    return 0;
}
```

还有多种做法，这种比较简单

未来轨迹的公共部分

https://www.cnblogs.com/shrshrshr/p/16774096.html Subtask1:对于a=b=1,c=n总有一个解。p = (1,2,...,n) q = r = (n,n-1,...,1) 其他的自己看上面这一篇和下面的题解 官方代码:

```
vector<int> compress(vector<int> a)
{
    set<ll> vals;
    for (auto it: a) vals.insert(it);

    int cur = 1; map<ll, int> mapka;

    for (auto it: vals)
    {
        mapka[it] = cur; cur++;
    }

    vector<int> ans;
    for (auto it: a) ans.push_back(mapka[it]);
    return ans;
}

vector<int> compress(vector<ll> a)
{
    set<ll> vals;
    for (auto it: a) vals.insert(it);

    int cur = 1; map<ll, int> mapka;

    for (auto it: vals)
    {
        mapka[it] = cur; cur++;
    }

    vector<int> ans;
    for (auto it: a) ans.push_back(mapka[it]);
    return ans;
}

vector<int> gen_perm(int n, int lis, int lds)
{
    vector<ll> guys;

    int rest = n - lis - lds + 1;

    set<ll> ins;
    for (int i = lds-1; i>=0; i--) ins.insert(i);
    for (int i = 0; i<lis; i++) ins.insert(1ll*i*lds);

    for (int i = 0; i<n; i++)
    {
        int val = lds*(i/lds) + (lds - 1 - i%lds);
        if (ins.count(val)) ins.erase(val);
        else rest--;
        guys.push_back(val);
```

```
        if (rest == 0 && i>=lds-1) break;
    }

    for (auto it: ins) guys.push_back(it);

    return compress(guys);
}

vector<vector<int>> solve(int n, ll a, ll b, ll c)
{
    vector<int> p, q, r;
    if (c == n)
    {
        for (int i = 1; i<=n; i++)
        {
            q.push_back(i); r.push_back(i);
        }

        for (int i = n; i>=a+1; i--) p.push_back(i);
        for (int i = 1; i<=a; i++) p.push_back(i);

        return {p, q, r};
    }

    int step = 0;
    while (a>=2 && (a-1)*(b-1)*(c-1)>=(n-1))
    {
        step++; a--; b--; c--; n--;
    }

    if (step > 0)
    {
        auto res = solve(n, a, b, c);
        for (int vec = 0; vec<3; vec++)
        {
            for (int i = n+1; i<=n+step; i++) res[vec].push_back(i);
        }
        return res;
    }

    if (a == 1)
    {
        for (int i = 1; i<=n; i++) p.push_back(i);
        for (int i = n; i>=1; i--) q.push_back(i);
        r = gen_perm(n, b, c);
        return {p, q, r};
    }

    if (a+b+c-2<=n)
    {
```

```
        vector<int> p1, q1, r1;
        for (int i = 1; i<=a*b*c; i++) p1.push_back(i);
        for (int i = b*c-1; i>=0; i--)
        {
            for (int j = 1; j<=a; j++) q1.push_back(i*a + j);
        }
        for (int i = 0; i<b; i++)
        {
            for (int j = a*c; j>=1; j--) r1.push_back(j + i*a*c);
        }

        /*print(p1);
        print(q1);
        print(r1);*/

        set<int> vals;
        for (int i = 1; i<=a; i++) vals.insert(i);
        for (int i = 0; i<b; i++) vals.insert(i*a*c + 1);
        for (int i = 0; i<c; i++) vals.insert(a*i + 1);

        for (int i = 1; i<=a*b*c; i++)
        {
            if (vals.size() == n) break;
            else vals.insert(i);
        }

        for (auto it: p1) if (vals.count(it)) p.push_back(it);
        for (auto it: q1) if (vals.count(it)) q.push_back(it);
        for (auto it: r1) if (vals.count(it)) r.push_back(it);

        p = compress(p); q = compress(q); r = compress(r);

        return {p, q, r};
    }

    if (a == 2 && b == 3 && c == 3 && n == 5)
    {
        p = {1, 2, 3, 4, 5};
        q = {3, 2, 1, 5, 4};
        r = {2, 1, 4, 3, 5};
        return {p, q, r};
    }

    for (int i = 1; i<=n; i++) p.push_back(i);
    for (int i = n; i>=3; i--) q.push_back(i); q.push_back(1); q.push_back(2);
    for (int i = n; i>=4; i--) r.push_back(i); r.push_back(1); r.push_back(3);
r.push_back(2);

    return {p, q, r};
}
```

```
void solve()
{
    ll n, a, b, c, output; cin>>n>>a>>b>>c>>output;

    if (b+c>a+n || a*b*c<n) {cout<<"NO"<<endl; return;}

    else cout<<"YES"<<endl;
    if (output)
    {
        auto res = solve(n, a, b, c);
        for (int i = 0; i<3; i++) print(res[i]);
    }
}

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(nullptr);

    int t; cin>>t;

    while (t--) solve();
}

/*
1
5 2 2 2 1
 */
```

原题题解是英文的，自己看

原题

- T1:[ABC244E] King Bombee
- T2: [NOI Online 2022 提高组] 丹钓战
- T3:[ABC133F] Colorful Tree
- T4: [eJOI202] LCS of Permutations 题解代码来自于洛谷题解和eJOI官方标程