比赛链接：AtCoder Beginner Contest 127

E - Cell Distance

题意

给一个 $n\times m$ 的网格和一个整数 k 每次取 k 个格子 (x_i,y_i) 得到 $v = \sum_{i=1}^{k-1}\sum_{j=i+1}^{k}(|x_i-x_j|+|y_i-y_j|)$求所有不同取法的 $\sum v$

数据范围

$2 \le n\times m \le 2\times 10^5$

$2 \le k \le n\times m$

题解

考虑两个格子 (x_i,y_i) 和 (x_j,y_j) 对答案的贡献为 $(|x_i-x_j|+|y_i-y_j|){k-2 \choose nm-2}$所以只要求一遍两两 $|x_i-x_j|+|y_i-y_j|$ 的值即可。

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
const ll M = 1e9+7;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
ll qpow(ll a,ll b) {ll
s=1;while(b){if(b&1)s=(s*a)%M;a=(a*a)%M;b>>=1;}return s; }
ll fac[maxn],inv[maxn];

void init()
{
    int n = 2e5;
    fac[0] = 1;
    rep(i,1,n) fac[i] = fac[i-1]*i%M;
    inv[n] = qpow(fac[n],M-2);
```

```
    per(i,n-1,0) inv[i] = inv[i+1]*(i+1)%M;
}
ll C(ll a,ll b) {return fac[a]*inv[a-b]%M*inv[b]%M;}
int main()
{
    fastio(); init();
    int n,m,k;
    cin>>n>>m>>k;
    ll b = C(n*m-2,k-2);
    ll s = 0,inv2 = qpow(2,M-2);
    for(ll i=1;i<=n;i++){
        for(ll j=1;j<=m;j++){
            ll a = n * ((j*(j-1)/2 + (1+m-j)*(m-j)/2) % M) % M;
            ll b = m * ((i*(i-1)/2 + (1+n-i)*(n-i)/2) % M) % M;
            s = (s + a + b) % M;
        }
    }
    cout<<b*s%M*inv2%M<<endl;
    return 0;
}
</hidden>
```

F - Absolute Minima

题意

Q 次两种操作维护一个函数。一开始的函数为 $f(x) = 0$每次操作给两个整数 a,b 使得 $g(x)=f(x)+|x-a|+b$ 然后用 $g(x)$ 更新 $f(x)$ 或者查询使得 $f(x)$ 最小的 x 和这个最小的 $f(x)$ 如果有多个 x 输出最小的。

数据范围

$1\le Q \le 2\times 10^5$

$-10^9 \le a,b \le 10^9$

题解

最后的函数都是形如 $f(x) = \sum |x-a_i| + B$ 的，于是最小值的 x 在这些a_i的中间取得，即对 a_i 排序后，如果有奇数个则为 $a_{\frac {n+1}2}$,否则为 $a_{\frac n2}$

比赛的时候做法比较复杂，离线离散化 a_i 后维护两棵线段树，一棵是权值线段树，另一棵是区间求和的线段树。每次查询时，通过权值线段树得到 $x = argmin \; f(x)$,然后通过区间求和得到 $\sum_{a_i \le x} a_i$ 和 $\sum_{a_i>x} a_i$再结合 $B = \sum b_i$ 就可以得到 $f_{min}(x)$

```
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

vector<pii_> op;
vector<int> g;
int tr[maxn<<2];
ll sum[maxn<<2];
void update(int id,int stl,int str,int pos)
{
    if(stl==str){
        tr[id]++;return ;
    }
    int mid = (stl+str)>>1;
    if(pos<=mid) update(id<<1,stl,mid,pos);
    else update(id<<1|1,mid+1,str,pos);
    tr[id] = tr[id<<1] + tr[id<<1|1];
}
int query(int id,int l,int r,int k)
{
    if(l==r){
        return l;
    }
    int mid = (l+r)>>1;
    if(tr[id<<1] >= k) return query(id<<1,l,mid,k);
    else return query(id<<1|1,mid+1,r,k-tr[id<<1]);
}
void change(int id,int l,int r,int pos,int k)
{
    if(l==r){
        sum[id] += k;
        return ;
    }
    int mid = (l+r)>>1;
    if(pos<=mid) change(id<<1,l,mid,pos,k);
    else change(id<<1|1,mid+1,r,pos,k);
```

```
    sum[id] = sum[id<<1] + sum[id<<1|1];
}
ll ask(int id,int stl,int str,int l,int r)
{
    if(stl==l && str==r) return sum[id];
    int mid = (stl+str)>>1;
    if(r<=mid) return ask(id<<1,stl,mid,l,r);
    else if(l>mid) return ask(id<<1|1,mid+1,str,l,r);
    else return ask(id<<1,stl,mid,l,mid) + ask(id<<1|1,mid+1,str,mid+1,r);
}
int main()
{
    fastio();
    int q,o,a,b; cin>>q;
    ll B = 0;
    while(q--){
        cin>>o;
        if(o==1){
            cin>>a>>b;
            op.pb(mp_(a,b));
            g.pb(a);
        }else{
            op.pb(mp_(-inf,-inf));
        }
    }
    sort(ALL(g));
    g.erase(unique(ALL(g)),g.end());
    int n = g.size();
    int cnt = 0;
    for(auto x:op){
        if(x.fi==-inf && x.se==-inf){
            int id = query(1,1,n,(cnt+1)/2) - 1;
            ll ans1 = g[id];
            ll k = (cnt+1)/2;
            ll ans2;
            {
                int L = 1,R = cnt,res=-1;
                while(L<=R){
                    int mid = (L+R)>>1;
                    if(query(1,1,n,mid)>id+1) R=mid-1,res=mid;
                    else L = mid+1;
                }
                if(res==-1) ans2 = ans1 * cnt - ask(1,1,n,1,id+1);
                else{
                    ll tmp = ask(1,1,n,1,id+1);
                    ans2 = ans1 * (res-1) - tmp + sum[1] - tmp - ans1 *
(cnt-res+1);
                }
            }
            cout<<ans1<<" "<<ans2+B<<endl;
```

```
        }else{
            int pos = lower_bound(ALL(g),x.fi)-g.begin()+1;
            update(1,1,n,pos);
            change(1,1,n,pos,x.fi);
            cnt++;
            B += x.se;
        }
    }
    return 0;
}
</hidden>
```

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai_milk:atcoder_beginner_contest_127_vp&rev=1595301226

Last update: **2020/07/21 11:13**