给出一个n*n（n为奇数）的蛇形矩阵，将其中的m个点的值改为各位数之后，给出p组询问，查询子矩阵的和。

将一个二维前缀和进行转化(内存存不下),若是 M(t1, x1, y1) 表示t1时刻修改[x, y] 处的值，Q(t2, x2, y2) 表示t2时刻询问[x, y]的前缀和，那么若是t1 ≤ t2, x1 ≤ x2, y1 ≤ y2，前者就会对后者有影响。

一开始已经按照时间存好了，只需要维护x 和 y。

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int N=1e6+5,M=800020;
int n,m,cnt,tot,ans[M];
struct BIT
{
    int p[N];
    inline int lowbit(int x){return x&(-x);}
    inline void insert(int pos,int val)
    {
        for (;pos<=n;pos+=lowbit(pos))
            p[pos] += val;
    }
    inline int query(int pos)
    {
        int res = 0;
        for (;pos;pos-=lowbit(pos))
            res += p[pos];
        return res;
    }
    inline void cl() {
        memset(p,0,sizeof(p));
    }
}tree;
struct order
{
    int flag,x,y,val;
    //当flag = 1时，表示的是修改，将[x,y]改成val; flag = 2,3为查询，查询到[x, y]的前缀和,val表示第几次询问
    bool operator < (order p)
    {
        if ( x ^ p.x ) return x < p.x;      //第一关键字为x, 第二关键字为y
        if ( y ^ p.y ) return y < p.y;
        return flag < p.flag;                //若是x y相同的话，修改会影响询问的，所以修改在前
    }
}a[M],cur[M];

int get(int x,int y)
{
    ll ans = 0,sum = 0;
    ll t = max(abs(x-(n+1)/2),abs(y-(n+1)/2));
    sum = 1LL*n*n - (2*t+1)*(2*t+1);
    if(x>=y) sum += abs((n+1)/2+t-x)+abs((n+1)/2+t-y)+1;
    else sum += 2*(2*t+1)-1+abs((n+1)/2-t-x)+abs((n+1)/2-t-y);
    while(sum)
```

```
        ans+=sum%10,sum/=10;
    return ans;
}


inline void cdq(int l,int r)
{
    if ( l == r ) return;
    int mid = l+r >> 1 , p = l , q = mid+1 , t = l-1;
    cdq(l,mid); cdq(mid+1,r);
    //前半部分的修改 对 后半部分的询问的影响
    while ( p <= mid && q <= r )
    {
        if ( a[p] < a[q] )
        {
            if ( a[p].flag == 1 ) tree.insert(a[p].y,a[p].val);  //若是前半部分的修
改
            cur[++t] = a[p++];
        }
        else
        {
            if ( a[q].flag == 2 ) ans[ a[q].val ] += tree.query(a[q].y);  //若是
后半部分的询问
            if ( a[q].flag == 3 ) ans[ a[q].val ] -= tree.query(a[q].y);
            cur[++t] = a[q++];
        }
    }
    while ( q <= r )              //看看后面的询问有没有完成，将后半部分补上
    {
        if ( a[q].flag == 2 ) ans[ a[q].val ] += tree.query(a[q].y);
        if ( a[q].flag == 3 ) ans[ a[q].val ] -= tree.query(a[q].y);
        cur[++t] = a[q++];
    }
    for (int i=l;i<p;i++)               //对树进行恢复
        if ( a[i].flag == 1 )
            tree.insert(a[i].y,-a[i].val);
    while ( p <= mid ) cur[++t] = a[p++];             //将前半部分补上
    for (int i=l;i<=r;i++) a[i] = cur[i];       //此时不仅对这个区间完成了计数，还进行
了排序
}

int T,p;
inline void input() {
    scanf("%d",&T);
    while(T--) {
        cnt = 0;
        tot = 0;
        tree.cl();
        memset(ans,0,sizeof(ans));
        scanf("%d%d%d",&n,&m,&p);
        for(int i=1;i<=m;i++) {
            int x,y;
            scanf("%d%d",&x,&y);
            int val = get(x,y);
            a[++cnt] = (order){1,x,y,val};
        }
        for(int i=1;i<=p;i++) {
            int x1,y1,x2,y2;
```

```
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            a[++cnt] = (order){2,x2,y2,++tot};
            a[++cnt] = (order){2,x1-1,y1-1,tot};
            a[++cnt] = (order){3,x1-1,y2,tot};
            a[++cnt] = (order){3,x2,y1-1,tot};
        }
        cdq(1,cnt);
        for (int i=1;i<=tot;i++)
            printf("%d\n",ans[i]);
    }
}

int main(void) {
    input();
    return 0;
}
```

有若干病毒，三维坐标为(x, y, z)，每一轮都有一些病毒被消灭，若是一个病毒不被消灭，那么存在另一个病毒，它的三维坐标均小于等于它(且至少有一维小于它)，问最少几轮可以消灭掉所有的病毒。

```
#include <bits/stdc++.h>

using namespace std;
typedef unsigned long long ll;
const int maxn = 1e5 + 10;
const double INF = 1e15;
const double eps = 1e-8;

int n, ans[maxn];
ll k1, k2;

struct node
{
    ll x, y, z;
    int id;
}arr[maxn];

bool cmpx(const node& m1, const node& m2)
{
    return m1.x < m2.x;
}

bool cmpy(const node& m1, const node& m2)
{
    return m1.y < m2.y;
}

struct treearray
{
    int tree[maxn], n;

    int lowerbit(int x)
    {
        return x&(-x);
    }
```

```
    int query(int i)                //查询[1 -i]最大值
    {
        int ans = 0;
        for(; i; i -= lowerbit(i))
            ans = max(ans, tree[i]);
        return ans;
    }

    void modify(int i,int k)              //将i处值修改为k，  维护最大值
    {
        for(;i <= n; i += lowerbit(i))
            tree[i] = max(k, tree[i]);
    }

    void Clear(int x)
    {
        for (int i = x; i <= n; i += lowerbit(i))
            tree[i] = 0;
    }
}t;

ll CoronavirusBeats()
{
    ll k3 = k1, k4 = k2;
    k1 = k4;
    k3 ^= k3 << 23;
    k2 = k3 ^ k4 ^ (k3 >> 17) ^ (k4 >> 26);
    return k2 + k4;
}

vector<ll> v;
void init()
{
    for (int i = 1; i <= n; i++)
    {
        arr[i].x = CoronavirusBeats();
        arr[i].y = CoronavirusBeats();
        arr[i].z = CoronavirusBeats();
        arr[i].id = i;
        v.push_back(arr[i].z);
    }
    sort(v.begin(), v.end());                                   //将z离散化，从1开始
    v.erase(unique(v.begin(), v.end()), v.end());       //去重
    for(int i = 1; i <= n; i++)
        arr[i].z = lower_bound(v.begin(), v.end(), arr[i].z) - v.begin() + 1;
    t.n = n;

}

void solve(int L, int R, int mid)
{
    int p1 = L, p2 = mid + 1;
    sort(arr + L, arr + 1 + mid, cmpy);
    sort(arr + 1 + mid, arr + R + 1, cmpy);           //之前未按照y进行排序

    while(p2 <= R)
    {
```

```
        while(p1 <= mid && arr[p1].y <= arr[p2].y)
        {
            t.modify(arr[p1].z, ans[arr[p1].id]);          //前半部分对后半部分进行保护
            p1++;
        }
        ans[arr[p2].id] = max(ans[arr[p2].id], t.query(arr[p2].z) + 1);
        p2++;
    }

    for(int i = L; i <= mid; i++)  t.Clear(arr[i].z);
    sort(arr + L, arr + R + 1, cmpx);        //先按X进行排序
}

void CDQ(int L, int R)
{
    if(L >= R) return;
    int mid = (L + R) / 2;
    CDQ(L, mid);
    solve(L, R, mid);
    CDQ(mid + 1, R);
}

int main()
{
    cin>>n>>k1>>k2;
    init();
    sort(arr + 1, arr + 1 + n, cmpx);

    for (int i = 1; i <= n; i++) ans[i] = 1;      //一开始初始化成1
    CDQ(1, n);

    int ret = 0;
    for(int i = 1; i <= n; i++)
        ret = max(ret, ans[i]);
    printf("%d\n", ret);
    for(int i = 1; i<= n; i++)
        printf("%d ", ans[i] - 1);
}
```