题库


acm International Collegiate Programming Contest
IBM
event sponsor

目录

动态规划

Dij 更新 DP

```
/*
* 题意：一个公园里面有n(<=50)个地区，m条边相连，这个公园中有k(<=8)个景区，每个景区需要排队，如果没有
*       通行证那么将会花费FTi,否则花费Fi的时间，需要求出从节点1出发经过所有景区再回到节点1的最小时间
*       花费。会告诉你每个景区在哪个地区，同时能在哪些地区获得这个景区的通行证。
*
* 解法：其实这个比较简单，以(当前节点，当前经过那些景区，当前持有那些景区的票)为状态，然后用dijkstra
*       去更新就行了，但是这里比较坑的地方就是一个地区可能有很多个状态，那么这里就必须要先处理一下。
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 10
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
// bool vis[1 << 8][55][1 << 8];
int dp[1 << 8][55][1 << 8];
int n, m, k;
vector<PII > G[55];
vector<int> isinter[55];
int tic[55];
struct node
{
    int s1, s2, u;
    int dist;
    node(){}
    node(int _s1, int _u, int _s2, int d){
        s1 = _s1, s2 = _s2, u = _u, dist = d;
    }
```

```
    bool operator < (const node& o) const{
        return dist > o.dist;
    }
};
int inter[N], oriT[N], curT[N];

int dijsktra()
{
    // memset(vis, 0, sizeof(vis));
    memset(dp, 0x3f, sizeof(dp));
    priority_queue<node> que;
    int s1 = 0, s2 = 0;
    int d = 0;
    que.push(node(s1, 1, s2, d));
    dp[s1][1][s2] = d;
    while(!que.empty())
    {
        node nod = que.top();que.pop();
        s1 = nod.s1, s2 = nod.s2;
        int u = nod.u, dist = nod.dist;
        if(dp[s1][u][s2] < dist) continue;
        // if(vis[s1][u][s2]) continue;
        // vis[s1][u][s2] = 1;
        if((s1 == ((1 << k) - 1)) && u == 1) return dist;
        for(int i = 0 ; i < G[u].size();i ++)
        {
            int v = G[u][i].first;
            int w = G[u][i].second;
            int ns2 = s2 | tic[v];
            int nd = dist + w;
            if(nd < dp[s1][v][ns2])
                que.push(node(s1, v, ns2, nd)), dp[s1][v][ns2] = nd;
            for(int j = 0; j < isinter[v].size(); j++)
            {
                int nd1 = nd;
                int idx = isinter[v][j];
                int ns1 = s1;
                if((ns1 & (1 << idx)) == 0)
                {
                    if(ns2 & (1 << idx)){
                        nd1 += curT[idx];
                    }else {
                        nd1 += oriT[idx];
                    }
                }
                ns1 |= (1 << idx);
                if(nd1 < dp[ns1][v][ns2])
                que.push(node(ns1, v, ns2, nd1)), dp[ns1][v][ns2] =
nd1;
            }
        }
    }
    return -1;
}
int main()
{
//    Open();
    int T;int cas = 1;
    scanf("%d", &T);
    while(T--)
    {
```

```
        memset(tic, 0, sizeof(tic));
        scanf("%d%d%d", &n, &m, &k);
        for(int i = 0 ;i <= n; i++) G[i].clear(),
G[i].push_back(PII(i, 0)), isinter[i].clear();
        for(int i = 0; i < m; i++)
        {
            int u, v, w;scanf("%d%d%d", &u, &v, &w);
            G[u].push_back(PII(v, w));
            G[v].push_back(PII(u, w));
        }
        for(int i = 0; i < k; i++)
        {
            scanf("%d%d%d", &inter[i], &oriT[i], &curT[i]);
            isinter[inter[i]].push_back(i);
//            if(isinter[inter[i]] != -1) while(1);
//            isinter[inter[i]] = i;
            int ni;
            scanf("%d", &ni);
            while(ni--)
            {
                int x;
                scanf("%d", &x);
                tic[x] |= (1<<i);
            }
        }
        printf("Case #%d: %d\n", cas++, dijsktra());
    }
    return 0;
}
```

双维属性 DP CodeForces_571B_贪心+DP

```
/*
* 题意：给出n个数，以及k，可以重排序列顺序，要求使得sigma(abs(a[i] -
a[k+i]))(1 <= i <= n - k)最小
*
* 做法：首先分析题目，这个题相当于有k条链子，如果想象吧同一条链子上面的值放在一
起，并且从小到大排序
*       ，那么会发现实际上每条链子上的贡献就是这条链子的最大值-最小值。那么如果把
整个a数组排序，
*       相同链子中的值一定是连续的，这样才会使得答案最小，那么做法就是首先将整个数
组排序，然后将这n
*       个数分成k个部分即可，但是当n不整除k的时候，各条链子中的值不一样，会有一部
分(第一类)比另一部分
*       (第二类)多1；所以这里就需要一个dp，dp[i][j]表示当前处理好了i条第一类链
子，j条第二类链子的最小值
*       那么答案就是dp[n%k][k - (n%k)];n%k为第一类链子的数量
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 220
```

```
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<long long,long long> PII;
const long long INF=0x3f3f3f3f;
const double eps = 1e-10;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
long long dp[5050][5050];
long long a[303333];
long long n, k;
int main()
{
    Open();
    scanf("%I64d%I64d", &n, &k);
    for(long long i = 0; i < n; i++){
        scanf("%I64d", &a[i]);
    }
    sort(a, a+n);
    dp[0][0] = 0;
    long long len = n / k;
    long long limit = n % k;
    memset(dp, 0x3f, sizeof(dp));
    dp[0][0] = 0;
    for(long long i=0;i<=limit;i++){
        for(long long j=0;j<=k-limit;j++){
            long long pos = (len+1)*(i) + len*j;
            dp[i+1][j] = min(dp[i+1][j] , dp[i][j] + a[pos+len+1-1] -
a[pos] );
            dp[i][j+1] = min(dp[i][j+1] , dp[i][j] + a[pos+len-1] -
a[pos] );
        }
    }
    printf("%I64d\n", dp[limit][k - limit]);
    return 0;
}
```

双维属性 DP UESTC_1225_具有相似属性的 DP 状态为三维

```
/*
* 题意：这里有一栋计划修建的豪华大楼，总共有n层楼，每一层都会修建一个乒乓球馆或者
游泳馆，
*       同时每一层都有Ti个人打乒乓球，Pi个人游泳，但是每层只会修建一种场馆，这就
导致有一
*       部分人会跑到距离这一层最近的有相应设施的楼层去玩。那么对于每个人来说都会有
一个花
*       费(花费即为这个人的起始层与目标层的层数差)，要用一种修建方案使得总花费最
小。输出
*       最小花费
*
```

```
* dp[i][j][k],表示第i个位置放置的设施为k(0/1)，前面有连续j个!k设施的最小花费。
* 所以dp[i][j][k] = min(dp[i-j][l][!k]) + sum(i-j, i);这里l不需要枚举，只需要
* 计算过程中记一下这一维的最小值即可。sum(i-j, i)预处理可得。
* 这里处理sum(i, j)之间的答案也有一些技巧，首先不考虑i==1/j==n的情况，那么[i, j]这个区间
* 的花费为v[i]*1 + v[i+1]*2 + v[i+2]*3 + ... + v[j-2]*3 + v[j-1]*2 + v[j]*1,也就是可以看
* 做是两个等差序列的和，形如sigma(v[i+k]*k)(1<=k<=j)这种形式的区间和，我们可以先处理出
* v[i]*i和v[i]的前缀和prei[i]和pre[i]，那么sigma(v[i+k]*k)(1<=k<=j) = prei[j+k] - prei[i]
*  - (pre[j+k] - pre[i])*i;
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <ctime>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 4050
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;

typedef long long LL;
typedef pair<LL, LL> PII;
const LL INF=0x3f3f3f3f;
const double eps = 1e-10;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("F:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
LL v[N][2];
LL prei[N][2];
LL sufi[N][2];
LL pre[N][2];
LL n;
//LL dp[N][N][2];
LL res[N][2];
LL getsum(LL l, LL r, LL k)
{
    if(l == 1) return sufi[1][k] - sufi[r+1][k] - pre[r][k]*(n-r);
    if(r == n) return prei[n][k] - prei[l-1][k] - (pre[n][k] - pre[l-1][k])*(l-1);
    if(l == r) return v[l][k];
    if(l == r - 1) return v[l][k] + v[r][k];
    LL mid = l + r >> 1;
    LL res = prei[mid][k] - prei[l-1][k] - (pre[mid][k] - pre[l-
```

```
1][k])*(l-1);
    l = mid+1;
    res += (sufi[l][k] - sufi[r+1][k]) - (pre[r][k] - pre[l-
1][k])*(n-r);
    return res;
}
int main()
{
//    Open();
    LL T;scanf("%lld",&T);
    for(LL Cas = 1; Cas <= T;Cas++){
        printf("Case #%lld: ",Cas);
        scanf("%lld", &n);
        memset(res, 0x3f, sizeof(res));
        pre[0][0] = pre[0][1] = 0;
        prei[0][0] = prei[0][1] = 0;
        for(LL i = 1; i <= n; i++)
        {
            scanf("%lld%lld" ,&v[i][0], &v[i][1]);
            pre[i][0] = pre[i-1][0] + v[i][0];
            pre[i][1] = pre[i-1][1] + v[i][1];
            prei[i][0] = prei[i-1][0] + v[i][0] * i;
            prei[i][1] = prei[i-1][1] + v[i][1] * i;
        }
        sufi[n+1][0] = sufi[n+1][1] = 0;
        for(LL i = n, cnt = 1; i >= 1; i--, cnt++)
            for(LL k = 0; k < 2; k++)
                sufi[i][k] = sufi[i+1][k] + v[i][k]*cnt;
        res[0][0] = res[0][1] = 0;
        for(LL i = 1; i <= n; i++)
            for(LL j = 1; j <= i; j++)
                for(LL k = 0; k < 2; k++){
                    LL tmpans = res[i-j][!k] + getsum(i-j+1, i, !k);
                    if(j != n)res[i][k] = min(res[i][k], tmpans);
                }
        LL ans = min(res[n][0], res[n][1]);
        printf("%lld\n", ans);
    }
    return 0;
}
```

组合计数 DP

```
/*
* http://acm.hdu.edu.cn/showproblem.php?pid=5136
* HDU 5136 Yue Fei's Battle
* 题意：你需要求出一棵树，这棵树的最长的路径上面的点数需要恰好为K，每个点最多只能有三个相邻的点。问有多少种构造方式，同构的树算一种(翻转某节点的子树之后树的形态相同)。
*
* 这个组合计数题对我来说还是比较困难，想了个大概，但是最主要的处理同构的地方完全没有想到。
* 首先这样考虑，因为这棵树的最长的路径上面的点数必须恰好为K，也就是说这棵树的直径为K，那么可以从这上面当做切入点。
* 设f[i]为层数为i的树，且每个节点的度数最多为3，根节点的度数最多为2的树的构造方案数，g[i] = sigma(f[j])(0<=j<=i);
```

```
* 假设从左到右构造；
* 直径上面的第i个点(端点为第0个点)我们都可以接上一棵层数为1~i的树，这样能保证这棵树的直径不超过K，假设当前0~(i-1)个点的方案数已经计算好，为tmp,(i<k/2)
* 那此时在第i个点上我们可以接上一棵层数为1~i的树，但是当接上一棵层数为i的树之后，主链势必会发生变化，也就是说当前接上去的子树会和之前已经构造的
* 0~(i-1)个点形成的树发生重构的现象，此时结果一定是计算重复了，那我们可以先将层数为i-2的树接上去，此时方案为tmp * g[i-1]；对于接上一棵层数为i的
* 树的方案数来说，因为此时主链有两条，能形成一条主链的方案数为f[i]，由于需要避免同构，那么我们可以在f[i]中选择两种出来构成这两条主链，那么总方案
* 数即为 tmp * g[i-1] + f[i]*(f[i-1])/2+f[i](因为两条链的构成相同也是一种方案)；那么一直递推到直径中间即可。
*
* 接下来还需要分类讨论，假设k为偶数，且前k/2个点的树的方案数为tmp，那么总的方案数即为tmp*(tmp-1)/2+tmp(加tmp的原因同上)。
* 如果k为奇数，那么还需要在中间的那个点上面接上树，此时可以接上层数为0~(k/2+1)，和上面一样，这里在接上层数为(k/2+1)的树时会出现三条主链，那么有以下几种情况：
* 1．三条主链都不同：此时方案为C(f[k/2+1]，3)；
* 2．两条主链相同，一条不同：此时方案为C(f[k/2+1]，2) * 2 (如果我选择的两种方案为A，B 那么组合方式可为{A,A,B},{A,B,B}两种)；
* 3．三条主链都相同：此时方案为C(f[k/2+1]，1)；
* 于是此时的总方案为tmp*g[k/2]+C(f[k/2+1]，3)+C(f[k/2+1]，
2)*2+C(f[k/2+1]，1)；
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <map>
#define LL long long
using namespace std;
#define N 300
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
const LL maxn=100000;
const LL mod=1e9+7.5;
LL f[maxn+5];
LL g[maxn+5];
LL gcd(LL a,LL b,LL &d,LL &x,LL &y)
{
    if(!b){d=a;x=1;y=0;}
    else {gcd(b,a%b,d,y,x);y-=x*(a/b);}
}
LL inv(LL a,LL n)
{
    LL d,x,y;
    gcd(a,n,d,x,y);
    return d==1?(x+n)%n:-1;
}
```

```
int main()
{
//    Open();
    g[0]=1;
    f[0]=1;
    for(LL i=1;i<=maxn;i++)
    {
        f[i]=(f[i-1]*(g[i-2]+1)%mod+f[i-1]*(f[i-1]-
1+mod)%mod*inv(2,mod))%mod;
        g[i]=(f[i]+g[i-1])%mod;
    }
    LL k;
    while(scanf("%I64d",&k)==1&&k)
    {
        if(k==1||k==2){printf("1\n");continue;}
        k-=2;
        LL ans=1LL;
        for(LL i=1;i<=k/2;i++){
            ans=(ans*(g[i-1]))%mod;
            ans = (ans + f[i] * (f[i] - 1)%mod * inv(2, mod) %mod)%mod;
            ans = (ans + f[i])%mod;
        }
        LL tmp = ans;
        ans = tmp*(tmp-1)%mod*inv(2, mod)%mod;
        ans = (ans + tmp)%mod;
        if(k%2==1){
            ans = ans * g[k/2]%mod;
            ans = (ans + tmp * (tmp - 1)%mod * (tmp - 2) %mod * inv(6,
mod)%mod)%mod;
            ans = (ans + tmp * (tmp - 1)%mod)%mod;
            ans = (ans + tmp)%mod;
        }
        printf("%I64d\n",(ans+mod)%mod);
    }
    return 0;
}
```

区间 DP

```
//http://codeforces.com/gym/100543/attachments
/*
*   The aliens from outer space have (finally!) invaded Earth. Defend
yourself, or be disintegrated!
*   Or assimilated. Or eaten. We are not yet sure.
*   The aliens follow a known attack pattern. There are n attackers,
the i-th one appears at time
*   ai, at distance di from you. He must be destroyed no later than at
time bi, or else he will fire his
*   weapon, which will definitely end the fight.
*   Your weapon is an area-blaster, which can be set to any given
power. If fired with power R,
*   it momentarily destroys all aliens at distance R or smaller. It
also consumes R fuel cells.
*   Determine the minimal cost (measured in fuel cells) of destroying
all the aliens, without being
*   killed.

*   Input
```

```
*   The first line of input contains the number of test cases T. The
descriptions of the test cases
*   follow:
*   Each test case starts with a line containing the number of aliens
n (1 <= n <= 300). Of the next
*   n lines, the i-th one contains three integers ai, bi, di, (1 <= ai <
bi <= 10 000; 1 <= di <= 10 000).
*   The i-th alien appears at time ai, is idle until bi, and his
distance from you is di.
*
*   Output
*   For each test case, output one line containing the minimum number
of cells needed to destroy
*   all the aliens.
*
*   题意：外星人正在计划入侵地球！你得知了会有n个外星人将会降临到地球上，每个外星
人会在ai时刻降临在距离
*         di的地方，而且必须在bi时刻前消灭它，不然就会外星人将会攻击你，你每一次
的攻击可以选定一个值R，
*         与此同时，与你距离小于等于R的所有外星人将会被你消灭，同时你会消耗R点能
量，需要求出最小的能量消耗
*
*   做法：区间dp，首先将所有时刻点离散化出来，dp[i][j]表示消灭出现在
(p[i],p[j])这个开区间内的所有敌人消灭的
*         最小花费，其中p数组为离散化的数组，这里更新比较不一样，一般的dp[i][j]
都是枚举i~j中的全部中间点，
*         去更新当前dp值，但是这里要使得(p[i],p[j])这个区间中的所有外星人都消灭
的话，首先R必须选择这个区间
*         内的最大距离，记这个外星人的下标为i，那么分裂点只能在a[i]~b[i]之间，才
能保证这个距离最远的外星人
*         会被消灭。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 610
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int dp[N][N];
vector<int> p;
```

```
int a[N], b[N], d[N];
int n;
int main()
{
    Open();
    int T;scanf("%d", &T);
    while(T--)
    {
        p.clear();
        scanf("%d", &n);
        for(int i=0;i<n;i++)
        {
            scanf("%d%d%d", &a[i], &b[i], &d[i]);
            p.push_back(a[i]), p.push_back(b[i]);
        }
        p.push_back(-INF), p.push_back(INF);
        sort(p.begin(), p.end());
        p.erase(unique(p.begin(), p.end()), p.end());
        for(int len = 0; len < p.size(); len++)
        {
            for(int i = 0; i < p.size();i++)
            {
                int j = i + len, hst = -1;
                for(int l = 0; l < n; l++)
                {
                    if(a[l] > p[i] && b[l] < p[j] && (hst == -1 ||
d[hst] < d[l])) hst = l;
                }
                if(hst == -1) dp[i][j] = 0;
                else {
                    dp[i][j] = INF;
                    int l = lower_bound(p.begin(), p.end(), a[hst]) -
p.begin();
                    int r = lower_bound(p.begin(), p.end(), b[hst]) -
p.begin();
                    for(int k = l; k <= r; k++)
                    {
                        dp[i][j] = min(dp[i][j], d[hst] + dp[i][k] +
dp[k][j]);
                    }
                }
            }
        }
        printf("%d\n", dp[0][p.size() - 1]);
    }
    return 0;
}
```

树形 DP

```
/*
* In The City of Eternal Festivities, there are ?? street junctions
and ??? 1 bidirectional streets,
* each street connecting two of the junctions. Between every two
junctions, there is exactly one
* (direct or indirect) path connecting them. No junction is an
endpoint for more than 10 streets.
* Every 13th of September (the 256th day of the year), there are many
festivities going on in
```

```
* The City. In particular, the citizens want to organize ?? parades.
The parade number ?? starts at
* junction ???? and ends at ????, following the unique path between
the endpoints.
* As the mayor of The City, you are responsible for citizens' safety.
Therefore you decreed that
* no two parades are ever allowed to use the same street, though they
can have common junctions,
* or even common endpoints.
* To appease your citizens, try to organize as many parades as
possible, without breaking the
* safety regulations.
*
* Input
* The first line of input contains the number of test cases ??. The
descriptions of the test cases
* follow:
* The first line of each test case contains a single integer: the
number of junctions ?? (2 ? ?? ?
* 1000). Each of the next ?? ? 1 lines contains two integers ??, ??
(1 ? ?? ?= ?? ? ??), denoting that
* junctions ?? and ?? are connected by a street. Each junction has at
most 10 streets leaving it.
* The next line contains a single integer: the number of planned
parades ??, 0 ? ?? ? (??? 2)?.
* Each of the next ?? lines contains two integers ????, ????
(1 ? ???? ?= ???? ? ??), meaning that a parade is
* planned to start at junction ????, and finish at junction ????. No
two parades share both endpoints.
*
* Output
* For each test case, output one line containing the largest number
of parades that can be
* organized with no street used by more than one parade.

* 题意：一个城市中有n个点，n-1条边，形成一棵树，节日的时候，有很多队伍想要在(ui,
vi)之间的街上游行
*       市长为了保证每个队伍的安全，需要保证没有一条街同时在两个队伍的游行路线上，
所以需要选择一种
*       方案，使得能够游行的队伍尽可能多，输出这个数量。
*
* 做法：树形DP，内部状压...大概能这么说，每个dfs(u)返回u这个节点子树的最大游行队
伍数量，同时维护好
*       那些节点还能够往上传。具体解释在代码注释中。
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 2010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
```

```
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int n, m;
vector<int> G[N];
bool pa[N][N];
int use[N][N];
int tail[N];

/*
* 这里dfs返回的是以u为根节点的子树上的最大的答案。然后每次dfs计算出以u为LCA的点对的最大数量
* 之所以能这样做是因为对每个节点的某个儿子来说，只可能有一个节点传上来(答案只可能增加1)。
* use[u]中存储的是以u为根节点的子树中有哪些点可以走到当前节点。
*/
int dfs(int u, int fa)
{
    int ans = 0;
    int siz = G[u].size();
    siz--;
    for(int i = 0; i < siz; i ++)
    {
        if(G[u][i] == fa) {
            swap(G[u][i], G[u][siz]);
        }
        int v = G[u][i];
        ans += dfs(v, u);
    }

    //首先处理子树中的点直接到根节点的
    for(int i = 0; i < siz; i++)
    {
        int v = G[u][i];
        for(int j = 0; j < tail[v]; j++)
        {
            if(pa[ use[v][j] ][u] ){
                ans ++, tail[v] = 0;
            }
        }
    }

    //这里处理的是哪些子树可以连接
    bool link[11][11];
    memset(link, 0, sizeof(link));
    for(int i = 0; i < siz; i ++)
        for(int j = i + 1 ; j < siz; j++)
        {
            int v1 = G[u][i], v2 = G[u][j];
            bool flag = true;
            for(int n1 = 0; n1 < tail[v1] && flag; n1++)
                for(int n2 = 0; n2 < tail[v2] && flag; n2 ++)
```

```
                if(pa[ use[v1][n1] ][ use[v2][n2] ])
                {
                    link[i][j] = link[j][i] = true;
                    flag = false;
                }
        }

    //状压DP计算当前状态的子树被选择能得到的最大的点对数
    int limit = 1 << siz;
    int dp[1111];
    dp[0] = 0;
    for(int s = 1; s < limit; s++)
    {
        int a = s & -s;
        int idx = __builtin_ctz(a);
        int tmps = s - a;
        dp[s] = dp[tmps];
        for(int i = 0; i < siz; i++)
        {
            if((tmps & (1 << i)) && link[idx][i])
            {
                dp[s] = max(dp[s], dp[tmps ^ (1 << i)] + 1);
            }
        }
    }
    ans += dp[limit - 1];//更新答案

    //维护当前节点的子树有哪些点可以往上传
    use[u][tail[u] ++] = u;
    for(int i = 0; i < siz; i++)
    {
        if(dp[(limit - 1) ^ (1 << i)] == dp[limit - 1])//表示这个点无所谓
不影响答案
        {
            for(int j = 0; j < tail[G[u][i]]; j++)
                use[u][tail[u]++] = use[G[u][i]][j];
        }
    }
    return ans;
}
int main()
{
    Open();
    int T;
    scanf("%d", &T);
    while(T--)
    {
        memset(tail, 0, sizeof(tail));
        memset(pa, 0, sizeof(pa));
        scanf("%d", &n);
        for(int i = 0; i <= n; i++) G[i].clear();
        for(int i = 1; i < n; i++)
        {
            int u, v;
            scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        scanf("%d", &m);
        for(int i = 0 ;i < m; i++)
```

```
        {
            int u, v;
            scanf("%d%d", &u, &v);
            pa[u][v] = pa[v][u] = 1;
        }
        G[1].push_back(-1);
        int ans = dfs(1, -1);
        printf("%d\n", ans);
    }
    return 0;
}
```

搜索 DP

```
/*
* Some computer games are extremely fun and this problem may be about
one of these.
* You are given a sequence of one dimensional blocks, each of length
that is a power of two.
* The goal of the game is to merge all the blocks into one big block.
The blocks are presented
* one by one, and for each one you have to decide whether to stick it
immediately to the left or
* immediately to the right of the previous blocks.
* Every time two blocks of the same size are adjacent, they merge
into one block that is twice
* as long as each of them. Note that, as long as possible, the
resulting blocks immediately merge
* with adjacent ones. For example, if the current sequence of blocks
is 2, 4, 16, then sticking 2 on
* the left side leads to 8, 16, while sticking it on the right side
gives 2, 4, 16, 2. Note that at any
* moment there is at most one mergeable pair of blocks.
* You have lost the game (again!) and you are wondering whether there
was any way to win
* at all. Analyze the sequence to find out.
*
* Input
* The first line of input contains the number of test cases T . The
descriptions of the test cases
* follow:
* Each test case consists of two lines. The first one contains a
positive integer n ¬ 1 000 – the
* length of the sequence. The next line contains a sequence of n
block lengths, each a power of
* two. The sum of all the lengths does not exceed 213.
*
* Output
* For each test case, output one line containing the word no if
winning the game is not possible.
* Otherwise, output a word consisting of n letters l and r, denoting
whether the corresponding
* block should be stuck to the left or to the right. Note that for
the first block it does not matter.
*
* 题意：这是一个类似2048的游戏，不过这个游戏只会在原有图形的左边或者右边会出现新
的方块，也仍然是相同
```

```
*       的会合并，只会出现二次幂的数。那么给出第i次出现的方块的值，并不知道出现在左边还是右边，问给出
*       的序列能不能通过某种(每块方块放左/右)方案使得最后只剩下一个数字，有一个限制为所有值的和不超过
*       1<<13
*
* 做法：这里既然限制了所有数的加和，且给出的数都是二进制数，那么很明显可以在这上面做文章(切入点)。
*       抽象的想一下，任何一个合法状态都只能是这样的：中间的值最大，两边递减，类似梯形。对于每个数来说
*       不是放左边就是放右边，每个方块有两种方式，还是很像一个DP，于是首先以dp[i][sl][sr]做状态，表示
*       处理到第i个方块，左边的状态为sl，右边的状态为sr，这一状态可不可行，这里的左边右边是相对于最大值
*       的，由于最大值也只可能一个，规定最大值一定在左边。现在考虑状态如何表示，分析可知，左边的值由于
*       是严格递增的，也就是说每一个数不可能出现大于1次，这意味着状态可用一个二进制数表示，同时还有一个
*       性质：表示sl这个状态的整数即为左边的方块的值的加和，这一个性质可以大大降低编程复杂度，右边也一样。
*       但是这样复杂度很明显为n*(1<<13)*(1<<13)一定是不行的，然后突然又发现，对于一个特定的i来说，确定
*       sl的话，意味着sr就确定了，因为sl+sr一定为a[1]+a[2]+...+a[i],也就是说这里复杂度降低一维！
*       于是，此题可解！
*       当然还是有许多细节处理，由于规定了最大值一定在左边，那么每次的更新都要维护这一性质，检测右边是否
*       大于左边。同时更新其间也有很多位运算。
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 1010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int dp[N][1<<14];
PII p[N][1<<14];
int cas = 1;
```

```
int n, sum[N], sta[N];
int f[N];
int main()
{
    Open();
    int T;scanf("%d", &T);
    while(T--)
    {
        cas ++;
        scanf("%d", &n);
        for(int i = 0; i < n; i++){
            scanf("%d", &f[i]);
            sum[i] = f[i];
            if(i) sum[i] += sum[i-1];
        }
        dp[0][f[0]] = cas;
        p[0][f[0]] = PII(0, 0);
        dp[0][0] = cas;
        p[0][0] = PII(0, 1);
        for(int i = 1;i < n; i++)
        {
            int cur = f[i];
            for(int j = 0; j <= sum[i-1]; j++)
            {
                if(dp[i-1][j] != cas) continue;
                int rit = sum[i-1] - j;
                if(j == 0 || (cur <= (j & (-j)))){
                    int rcnt = 31 - __builtin_clz(rit);
                    int lcnt = 31 - __builtin_clz(j + cur);
                    if(rcnt == lcnt)
                        dp[i][j + cur + (1<<lcnt)] = cas, p[i][j + cur +
(1<<lcnt)] = PII(j, 0);
                    else
                        dp[i][j + cur] = cas, p[i][j + cur] = PII(j, 0);
                }
                if(rit == 0 || cur <= (rit & (-rit))){
                    int rcnt = 31 - __builtin_clz(cur + rit);
                    int lcnt = 31 - __builtin_clz(j);
                    if(rcnt == lcnt)
                        dp[i][j + (1 << lcnt)] = cas, p[i][j + (1 <<
lcnt)] = PII(j, 1);
                    else
                        dp[i][j] = cas, p[i][j] = PII(j, 1);
                }
                if(__builtin_popcount(j) == 1 && j > rit && cur > j){
                    dp[i][cur] = cas, p[i][cur] = PII(j, 0);
                }
                if(__builtin_popcount(rit) == 1 && rit > j && cur >
rit){
                    dp[i][j + rit] = cas, p[i][j + rit] = PII(j, 1);
                }
            }
        }
        int tail = 0;
        if(__builtin_popcount(sum[n-1]) != 1 || (dp[n-1][sum[n-1]] !=
cas && dp[n-1][0] != cas)){
            printf("no\n");
            continue;
        }

        int pre, cnt;
```

```
        if(dp[n-1][sum[n-1]] == cas)
            pre = sum[n-1], cnt = n-1;
        else
            pre = 0, cnt = n-1;

        while(cnt >= 0){
            PII cur = p[cnt][pre];
            sta[tail++] = cur.second;
            pre = cur.first;
            cnt--;
        }
        for(int i = tail - 1; i >= 0; i--)
        {
            printf("%c", "lr"[sta[i]]);
        }
        printf("\n");
    }
    return 0;
}
```

分数规划+单调队列

```
//http://acm.hust.edu.cn/vjudge/contest/view.action?cid=87817#problem
/F
/*
* Annie Sweety is studying the problem to select rectangular regions
in a given image. She views an
* image being just a rectangular matrix with pixels as miniature
rectangles sandwiched together. The
* rectangular regions Annie is interested are those rectangular
blocks within the pixels matrix that are
* large enough and also with brightest possible averaged value
pixels.
* Formally, given an m × n integer matrix, you are asked to write a
program to find the submatrix,
* so that the average is maximum on the condition that the total
number of elements in the submatrix
* need to be no less than a given number k. The submatrix in request
consists of continuous rows and
* columns of the given matrix. That is, it is a continuous rectangle
region of the matrix in question. Here
* we define the average in a submatrix as the sum between all the
elements on the submatrix divided by
* the number of elements evaluating. For example, given constraint k
= 4 and the following 3×5 matrix:
* 9 2 6 3 9
* 7 7 5 7 8
* 4 7 8 7 5
* The constrained maximum average submatrix is the following 2 3
submatrix
* 7 5 7
* 7 8 7
* with size 6, while its average is (7 + 5 + 7 + 7 + 8 + 7)/6 = 41/6
= 6.833 .
* Technical Specification
```

```
* • The row, m, of each matrix can be as large as 600, and column, n,
can be as large as 10,000.
* • The value of each element of the matrix is a nonnegative integer
at most 5,000.
* • k ≤ mn ≤ 1, 000, 000.
*
* Input
* The input consists of several instances of matrices with
corresponding constrained sizes. The inputs are
* just a list of integers. The first input integer indicates the
total number of input instances. For each
* instance of matrix and constraint, the first two integers are the
row number, m, and column number
* n of the given matrix, representing a following m × n integer
matrix. Following m, n, the third integer
* represents the constraint size k.
* After (m, n, k), there will be m lines representing the m rows of
the matrix; each line (row) contains
* exactly n integers. Thus, there is totally m · n integers for the
particular matrix.
*
* Output
* For each matrix of the input, calculate its constrained maximum
average submatrix and output the
* average value of the submatrix rounded to the nearest thousandth.
*
* 题意：给出一个n*m的矩阵，以及一个k，求一个元素不小于k的子矩阵的平均值的最大
*
* 做法：很明显的分数规划问题，需求就是max(sigma(a[i])/n)的最大值，设答案为
ans，代入式子中，再变一下形：
*       sigma(a[i]) - n*ans = 0,那么这里可以二分枚举ansp，这有下面不等式成
立：
*       max(sigma(a[i]) - n*ansp) < 0 => ansp > ans
*       max(sigma(a[i]) - n*ansp) > 0 => ansp < ans
*       按照这个规则二分ans即可，求max这一部分可以用单调队列解决，需要先处理出每
一列的前缀和，
*       在二份函数中变为一个一维数组的单调队列，如此即可高效算出max(sigma(a[i])
- n*ansp);
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 10100
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
const double eps = 1e-5;
void Open()
{
    #ifndef ONLINE_JUDGE
```

```
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
double f[610][10010];
double colsum[10010];
int n, m, K;
double sta[N];
bool check(double g)
{
    for(int i = 1; i <= n; i++)
        for(int j = i; j <= n; j++)
        {
            int rowlen = j - i + 1;
            int collen = (K - 1) / rowlen + 1;
            collen = max(1, collen);
            int tail = 0;
//            sta[tail++] = 0;
            colsum[0] = 0;
            for(int k = 1; k <= m; k++)
            {
                colsum[k] = f[j][k] - f[i-1][k] - g * rowlen +
colsum[k-1];
                while(k >= collen && tail > 0 && sta[tail - 1] >
colsum[k - collen]) tail--;
                if(k >= collen) sta[tail++] = colsum[k - collen];
                if(tail > 0 && colsum[k] - sta[0] > -eps) return true;
            }
        }
    return false;
}
int main()
{
    Open();
    int T;scanf("%d", &T);
    while(T--)
    {
        scanf("%d%d%d", &n, &m, &K);
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= m; j++)
                scanf("%lf", &f[i][j]);
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= m; j++)
                f[i][j] += f[i-1][j];
        double lb = -1, ub = 5001;
        while(lb + eps < ub){
            double mid = (lb + ub) / 2;
            if(check(mid)) lb = mid;
            else ub = mid;
        }
        printf("%.3f\n", lb);
    }
    return 0;
}
```

高级数据结构

CDQ

CDQ+可回滚并查集

```
/*
* Problem Description
* A connected, undirected graph of N vertices and M edges is given to
CRB.
* A pair of vertices (u, v) (u < v) is called critical for edge e if
and only
* if u and v become disconnected by removing e.
* CRB's task is to find a critical pair for each of M edges. Help
him!
*
* Input
* There are multiple test cases. The first line of input contains an
integer T, indicating
* the number of test cases. For each test case:
* The first line contains two integers N, M denoting the number of
vertices and the number of edges.
* Each of the next M lines contains a pair of integers a and b,
denoting an undirected edge
* between a and b.
* 1 ≤ T ≤ 12
* 1 ≤ N, M ≤ 105
* 1 ≤ a, b ≤ N
* All given graphs are connected.
* There are neither multiple edges nor self loops, i.e. the graph is
simple.
*
* Output
* For each test case, output M lines, i-th of them should contain two
integers u and v,
* denoting a critical pair (u, v) for the i-th edge in the input.
* If no critical pair exists, output "0 0" for that edge.
* If multiple critical pairs exist, output the pair with largest u.
If still ambiguous,
* output the pair with smallest v.
*
* 题意：给出一个简单图(无重边，无自环)，要求输出每条边e的"至关重要对"，也就是如果
删除e这条边，
*       这两个点就不连通。
*
* 做法：这里用可回滚并查集(就是暴力回滚而已)+CDQ，并查集合并方式必须是启发式合
并，而不能有路径压缩，
*       CDQ分治边表，到CDQ的每一层[l,r]时，并查集中的状态都是[1,l-1]与[r+1,m]
中的全部边合并之后的状态
*       那么当分治到长度为1的时候，此时并查集的状态即为整个图中不包含这条边的状
态。函数返回上一层的时
*       候，回滚即可。回滚这一步的复杂度与合并是一样的。
*/

#include <iostream>
#include <cstdio>
```

```cpp
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int rk[N], pa[N], ma[N];
struct node{
    int u, pa, rk, ma;
}sta[N*2];
struct edge{
    int u, v;
}e[N];
int tail  = 0;
int n, m;
PII ans[N];
int find(int x)
{
    while(x != pa[x]) x = pa[x];
    return x;
}
void unite(int u, int v){
    u = find(u), v = find(v);
    if(u == v) return ;
    sta[tail++] = (node){u, u, rk[u], ma[u]};
    sta[tail++] = (node){v, v, rk[v], ma[v]};
    if(rk[u] > rk[v]){
        swap(u, v);
    }
    pa[u] = v;
    if(rk[u] == rk[v]) rk[v]++;
    ma[v] = max(ma[v], ma[u]);
}
void rollback(int tmp)
{
    while(tail > tmp){
        tail--;
        int u = sta[tail].u;
        pa[u] = sta[tail].pa;
        rk[u] = sta[tail].rk;
        ma[u] = sta[tail].ma;
    }
}
void UnionAll(int l, int r)
{
```

```
    for(int i = l; i <= r; i++)
    {
        unite(e[i].u, e[i].v);
    }
}
void divide(int l, int r)
{
    if(l == r){
        int u = find(e[l].u), v = find(e[l].v);
        if(u == v) ans[l] = PII(0, 0);
        else{
            int tmp = min(ma[u], ma[v]);
            ans[l] = PII(tmp, tmp+1);
        }
        return ;
    }
    int mid = l + r >> 1;
    int tmp = tail;
    UnionAll(mid+1, r);
    divide(l, mid);
    rollback(tmp);

    UnionAll(l, mid);
    divide(mid+1, r);
    rollback(tmp);
}
char *ch, buf[40*1024000+5];

template <class T>
void read(T &x) {
    for (++ch; *ch <= 32; ++ch);
    for (x = 0; '0' <= *ch; ch++)   x = x * 10 + *ch - '0';
}

int main()
{
    Open();
    ch = buf - 1;
    fread(buf, 1, 1000*35*1024, stdin);
    int T;
    read(T);
//    scanf("%d",&T);
    while(T--)
    {
//        scanf("%d%d", &n, &m);
        read(n);read(m);
        tail = 0;
        for(int i = 1; i <= n; i++)
            pa[i] = i, rk[i] = 1, ma[i] = i;
        for(int i = 1; i <= m; i++)
        {
            read(e[i].u);read(e[i].v);
//            scanf("%d%d", &e[i].u, &e[i].v);
        }
        divide(1, m);
        for(int i = 1; i <= m; i++)
        {
            printf("%d %d\n", ans[i].first, ans[i].second);
        }
    }
    return 0;
```

```
}
```

CDQ+树状数组+归并

```
/*
* Problem Description
* One day Lillian gets some segments from her fans Lawson with
lengths of 1,2,3... and she intends to display them by adding them to
a number line.At the i-th add operation,she will put the segment with
length of i on the number line.Every time she put the segment on the
line,she will count how many entire segments on that segment.During
the operation ,she may delete some segments on the line.(Segments are
mutually independent)
*
* Input
* There are multiple test cases.
* The first line of each case contains a integer n — the number of
operations(1<=n<=2*105,∑n<=7*105)
* Next n lines contain the descriptions of the operatons,one
operation per line.Each operation contains two integers a , b.
* if a is 0,it means add operation that Lilian put a segment on the
position b(|b|<109) of the line.
* (For the i-th add operation,she will put the segment on [b,b+i] of
the line, with length of i.)
* if a is 1,it means delete operation that Lilian will delete the
segment which was added at the b-th add operation.
*
* Output
* For i-th case,the first line output the test case number.
* Then for each add operation,ouput how many entire segments on the
segment which Lillian newly adds.
*
* 题意：有n条线段，每条线段的长度分别是1,2,3,...,n，现在要按顺序讲这些线段放在数
轴上，两种操作：
*       第一种：将待放进去的线段放到数轴的b位置(起始位置为b),要求输出这条线段覆盖
了多少条完整的
*       线段(还在数轴上的)，
*       第二种：将第b次操作放入的线段删除。
* 做法：这个题有两种做法，一种是CDQ，这种方法很暴力啦，不过要注意，这里不能再CDQ
里面直接排序，而是需要用
*       下层排序之后的结果进行归并排序，降低复杂度，才能卡着过，至于在cdq里面的
话，就是先保证左端点在当前
*       线段左端点右边，再去树状数组中查找有多少个右端点在当前线段的右端点的左边。
*       另一种是树状数组，上面CDQ没有利用到题目中的线段长度递增这一性质。只需要用
两个树状数组，一个(lc)记录
*       左端点，一个(rc)记录右端点即可，对于一个区间[l,r]，ans = getsum(rc,
r) - getsum(lc, l-1)。如果题目
*       没有规定长度递增的话，这样是有问题，因为在减左端点的时候，会将lp<l&&rp>r
的那种区间减去，但是
*       这里规定了区间长度，所以这里也并没有那样的区间这样的话，代码也好写一些。
*
* 扩展：如果这里没有时效性的操作，可以将所有区间按照左端点排序，从左到右做一遍，维
护树状数组的信息，这样
*       也可以求出所有区间内含的所有完整区间的个数。
```

```
*/

#include <iostream>
#include <cstdio>
#include <algorithm>
#include <queue>
#include <cstring>
#include <string>
#include <cmath>
#include <set>
#include <map>
#include <vector>
#include <climits>
using namespace std;
#define pb push_back
#define ALL(x) x.begin(),x.end()
#define VINT vector<int>
#define PII pair<int,int>
#define MP(x,y) make_pair((x),(y))
#define ll long long
#define ull unsigned ll
#define MEM0(x)  memset(x,0,sizeof(x))
#define MEM(x,val) memset((x),val,sizeof(x))
#define scan(x) scanf("%d",&(x))
#define scan2(x,y) scanf("%d%d",&(x),&(y))
#define scan3(x,y,z) scanf("%d%d%d",&(x),&(y),&(z))
#define scan4(x,y,z,k) scanf("%d%d%d%d",&(x),&(y),&(z),&(k))
#define Max(a,b) a=max(a,b)
#define Min(a,b) a=min(a,b)
using namespace std;
const int N=400045;

int mp[N];
struct node
{
    int ty, l, r;
    node(){}
    node(int ty, int l, int r):ty(ty), l(l), r(r){}
}q[N];
int n;
int idx[N];
int tmpidx[N];
int cnt = 0;//
int rc[N];//
int lc[N];//
int ans[N];
void add(int c[], int x, int val)
{
    if( x == 0 ) return ;
    for(int i=x;i <= cnt+10;i += i & (-i)) c[i] += val;
}

void bitclear(int c[], int x)
{
    if( x == 0 ) return ;
    for(int i=x;i <= cnt+10;i += i & (-i)) c[i] = 0;
}

int getsum(int c[], int x)
{
    int rnt = 0;
```

```
    for(int i = x;i>0;i -= i & (-i)) rnt += c[i];
    return rnt;
}
bool cmp(int a, int b)
{
    return q[a].l > q[b].l;
}
bool bigcmp(int a, int b)
{
    return a > b;
}
void divide(int l, int r)
{
    if(l >= r)
    {
        return ;
    }
    int mid = (l+r)/2;
    divide(l, mid);
    divide(mid+1, r);
    int midmid = (l + mid)/2;
    std::merge(tmpidx+l, tmpidx+midmid+1, tmpidx+midmid+1,
tmpidx+mid+1, idx+l, cmp);
    midmid = (mid+1+r)/2;
    std::merge(tmpidx+mid+1, tmpidx+midmid+1, tmpidx+midmid+1,
tmpidx+r+1, idx+mid+1, cmp);
    int tail = l;
    for(int i = mid+1; i<=r; i++)
    {
        if(q[idx[i]].ty == 1) continue;
        while(tail <= mid && q[idx[tail]].l >= q[idx[i]].l){
            if(q[idx[tail]].ty == 0) add(rc, q[idx[tail]].r, 1);
            else if(q[idx[tail]].ty == 1) add(rc, q[idx[tail]].r, -1);
            tail++;
        }

        if(q[idx[i]].ty == 0) ans[idx[i]] += getsum(rc, q[idx[i]].r);
    }
    for(int i = l; i <= mid ; i++)
        bitclear(rc, q[i].r);
    for(int i = l; i <= r; i++)
        tmpidx[i] = idx[i];
}
template<class T>
inline bool read(T &n){
    T x = 0, tmp = 1; char c = getchar();
    while ((c < '0' || c > '9') && c != '-' && c != EOF) c =
getchar();
    if (c == EOF) return false;
    if (c == '-') c = getchar(), tmp = -1;
    while (c >= '0' && c <= '9') x *= 10, x += (c - '0'), c =
getchar();
    n = x*tmp;
    return true;
}

template <class T>
inline void write(T n) {
    if (n < 0) {
        putchar('-');
        n = -n;
```

```
    }
    int len = 0, data[20];
    while (n) {
        data[len++] = n % 10;
        n /= 10;
    }
    if (!len) data[len++] = 0;
    while (len--) putchar(data[len] + 48);
}
int main()
{
#ifndef ONLINE_JUDGE
    freopen("D:/in.txt","r",stdin);
#endif
    int Cas = 1;
    while(~scanf("%d", &n))
    {
        printf("Case #%d:\n",Cas++);
        memset(rc, 0, sizeof(rc));
        memset(ans, 0, sizeof(ans));
        cnt = 0;
        int len = 1;
        for(int i=1;i<=n;i++)
        {
            int a, b;
            scanf("%d%d", &a, &b);
//            read(a);read(b);
            if(a == 0){
                q[i] = node(a, b, b+len);
                mp[cnt++] = b, mp[cnt++] = b+len;
                idx[len] = i;
                len++;
            }else{
                q[i] = node(a, q[idx[b]].l, q[idx[b]].r);
            }
        }
        sort(mp, mp+cnt);
        cnt = unique(mp, mp+cnt) - mp;
        for(int i=1;i<=n;i++)
        {
            q[i].l = lower_bound(mp, mp+cnt, q[i].l) - mp + 1;
            q[i].r = lower_bound(mp, mp+cnt, q[i].r) - mp + 1;
        }
        for(int i=1;i<=n;i++)
            tmpidx[i] = i;
        divide(1, n);
        for(int i=1;i<=n;i++)
        {
            if(q[i].ty == 0){
//                printf("%d\n", ans[i]);
                write(ans[i]);
                putchar('\n');
            }
        }
    }
    return 0;
}
```

CDQ+可回滚并查集二分图

```
/*
* Problem Description
* One day Lillian gets some segments from her fans Lawson with lengths of 1,2,3... and she intends to display them by adding them to a number line.At the i-th add operation,she will put the segment with length of i on the number line.Every time she put the segment on the line,she will count how many entire segments on that segment.During the operation ,she may delete some segments on the line.(Segments are mutually independent)
*
* Input
* There are multiple test cases.
* The first line of each case contains a integer n — the number of operations(1<=n<=2*105,∑n<=7*105)
* Next n lines contain the descriptions of the operatons,one operation per line.Each operation contains two integers a , b.
* if a is 0,it means add operation that Lilian put a segment on the position b(|b|<109) of the line.
* (For the i-th add operation,she will put the segment on [b,b+i] of the line, with length of i.)
* if a is 1,it means delete operation that Lilian will delete the segment which was added at the b-th add operation.
*
* Output
* For i-th case,the first line output the test case number.
* Then for each add operation,ouput how many entire segments on the segment which Lillian newly adds.
*
* 题意：有n条线段，每条线段的长度分别是1,2,3,...,n，现在要按顺序讲这些线段放在数轴上，两种操作：
*       第一种：将待放进去的线段放到数轴的b位置(起始位置为b),要求输出这条线段覆盖了多少条完整的
*       线段(还在数轴上的)，
*       第二种：将第b次操作放入的线段删除。
* 做法：这个题有两种做法，一种是CDQ，这种方法很暴力啦，不过要注意，这里不能再CDQ里面直接排序，而是需要用
*       下层排序之后的结果进行归并排序，降低复杂度，才能卡着过，至于在cdq里面的话，就是先保证左端点在当前
*       线段左端点右边，再去树状数组中查找有多少个右端点在当前线段的右端点的左边。
*       另一种是树状数组，上面CDQ没有利用到题目中的线段长度递增这一性质。只需要用两个树状数组，一个(lc)记录
*       左端点，一个(rc)记录右端点即可，对于一个区间[l,r]，ans = getsum(rc, r) - getsum(lc, l-1)。如果题目
*       没有规定长度递增的话，这样是有问题，因为在减左端点的时候，会将lp<l&&rp>r的那种区间减去，但是
*       这里规定了区间长度，所以这里也并没有那样的区间这样的话，代码也好写一些。
*
* 扩展：如果这里没有时效性的操作，可以将所有区间按照左端点排序，从左到右做一遍，维护树状数组的信息，这样
*       也可以求出所有区间内含的所有完整区间的个数。
*/

#include <iostream>
#include <cstdio>
#include <algorithm>
```

```
#include <queue>
#include <cstring>
#include <string>
#include <cmath>
#include <set>
#include <map>
#include <vector>
#include <climits>
using namespace std;
#define pb push_back
#define ALL(x) x.begin(),x.end()
#define VINT vector<int>
#define PII pair<int,int>
#define MP(x,y) make_pair((x),(y))
#define ll long long
#define ull unsigned ll
#define MEM0(x)  memset(x,0,sizeof(x))
#define MEM(x,val) memset((x),val,sizeof(x))
#define scan(x) scanf("%d",&(x))
#define scan2(x,y) scanf("%d%d",&(x),&(y))
#define scan3(x,y,z) scanf("%d%d%d",&(x),&(y),&(z))
#define scan4(x,y,z,k) scanf("%d%d%d%d",&(x),&(y),&(z),&(k))
#define Max(a,b) a=max(a,b)
#define Min(a,b) a=min(a,b)
using namespace std;
const int N=400045;

int mp[N];
struct node
{
    int ty, l, r;
    node(){}
    node(int ty, int l, int r):ty(ty), l(l), r(r){}
}q[N];
int n;
int idx[N];
int tmpidx[N];
int cnt = 0;//
int rc[N];//
int lc[N];//
int ans[N];
void add(int c[], int x, int val)
{
    if( x == 0 ) return ;
    for(int i=x;i <= cnt+10;i += i & (-i)) c[i] += val;
}

void bitclear(int c[], int x)
{
    if( x == 0 ) return ;
    for(int i=x;i <= cnt+10;i += i & (-i)) c[i] = 0;
}

int getsum(int c[], int x)
{
    int rnt = 0;
    for(int i = x;i>0;i -= i & (-i)) rnt += c[i];
    return rnt;
}
bool cmp(int a, int b)
{
```

```
    return q[a].l > q[b].l;
}
bool bigcmp(int a, int b)
{
    return a > b;
}
void divide(int l, int r)
{
    if(l >= r)
    {
        return ;
    }
    int mid = (l+r)/2;
    divide(l, mid);
    divide(mid+1, r);
    int midmid = (l + mid)/2;
    std::merge(tmpidx+l, tmpidx+midmid+1, tmpidx+midmid+1,
tmpidx+mid+1, idx+l, cmp);
    midmid = (mid+1+r)/2;
    std::merge(tmpidx+mid+1, tmpidx+midmid+1, tmpidx+midmid+1,
tmpidx+r+1, idx+mid+1, cmp);
    int tail = l;
    for(int i = mid+1; i<=r; i++)
    {
        if(q[idx[i]].ty == 1) continue;
        while(tail <= mid && q[idx[tail]].l >= q[idx[i]].l){
            if(q[idx[tail]].ty == 0) add(rc, q[idx[tail]].r, 1);
            else if(q[idx[tail]].ty == 1) add(rc, q[idx[tail]].r, -1);
            tail++;
        }

        if(q[idx[i]].ty == 0) ans[idx[i]] += getsum(rc, q[idx[i]].r);
    }
    for(int i = l; i <= mid ; i++)
        bitclear(rc, q[i].r);
    for(int i = l; i <= r; i++)
        tmpidx[i] = idx[i];
}
template<class T>
inline bool read(T &n){
    T x = 0, tmp = 1; char c = getchar();
    while ((c < '0' || c > '9') && c != '-' && c != EOF) c =
getchar();
    if (c == EOF) return false;
    if (c == '-') c = getchar(), tmp = -1;
    while (c >= '0' && c <= '9') x *= 10, x += (c - '0'), c =
getchar();
    n = x*tmp;
    return true;
}

template <class T>
inline void write(T n) {
    if (n < 0) {
        putchar('-');
        n = -n;
    }
    int len = 0, data[20];
    while (n) {
        data[len++] = n % 10;
        n /= 10;
```

```
    }
    if (!len) data[len++] = 0;
    while (len--) putchar(data[len] + 48);
}
int main()
{
#ifndef ONLINE_JUDGE
    freopen("D:/in.txt","r",stdin);
#endif
    int Cas = 1;
    while(~scanf("%d", &n))
    {
        printf("Case #%d:\n",Cas++);
        memset(rc, 0, sizeof(rc));
        memset(ans, 0, sizeof(ans));
        cnt = 0;
        int len = 1;
        for(int i=1;i<=n;i++)
        {
            int a, b;
            scanf("%d%d", &a, &b);
//            read(a);read(b);
            if(a == 0){
                q[i] = node(a, b, b+len);
                mp[cnt++] = b, mp[cnt++] = b+len;
                idx[len] = i;
                len++;
            }else{
                q[i] = node(a, q[idx[b]].l, q[idx[b]].r);
            }
        }
        sort(mp, mp+cnt);
        cnt = unique(mp, mp+cnt) - mp;
        for(int i=1;i<=n;i++)
        {
            q[i].l = lower_bound(mp, mp+cnt, q[i].l) - mp + 1;
            q[i].r = lower_bound(mp, mp+cnt, q[i].r) - mp + 1;
        }
        for(int i=1;i<=n;i++)
            tmpidx[i] = i;
        divide(1, n);
        for(int i=1;i<=n;i++)
        {
            if(q[i].ty == 0){
//                printf("%d\n", ans[i]);
                write(ans[i]);
                putchar('\n');
            }
        }
    }
    return 0;
}
```

CDQ+动态维护凸包_暴力重做，删点

```
/*
* 题意：一个狗国家的狗国王有一个装糖的盒子，每颗糖有两个属性p,q，分别代表甜度和咸
```

度，每只狗
*　　对于甜度和咸度的偏爱度不一样，所以每条狗有两个参数x,y，每颗糖对于特定的狗的
*　　美味度等于p*x+q*y。现在有50000个操作，分为三种：
*　　将新的糖(p,q)放入盒子中
*　　将盒子中存在的糖(p,q)吃掉
*　　给出一条狗的参数(x,y)，询问当前存在的糖的最大美味度。
*
* 做法：这个题是2014年广州现场赛的A题，模拟赛做的时候虽然推出了公式，但是并没有发现最优解为凸包
* 的性质，于是一直到比赛结束都没有做出这个题，但是当时最诡异的是，有一个排名比较靠后的队伍在
* 16分钟就将此题一血拿走，整场比赛解出此题的不过10支左右队伍。当比赛结束的时候，看了一下别人的
* 代码，发现我能在hust上面看到的代码居然都是暴力水过去的！…我觉得好醉啊…，继续在网上买了搜索此
* 题题解，发现都是直接n2暴力来一遍的…，仔细想了一下，n2暴力的话，复杂度在最坏的情况下
* 为25000*25000，大概6*108左右，这种复杂度我是无论如何都不敢敲的…，但是如果数据是随机生成的
* 话，询问，插入，删除操作只占50000*1/3，那么这个复杂度以当时的30s的时限来说，还是可以跑出结
* 果的…于是这么一个现场10支队伍过的难题，就被这样水过去了。
*
* 此题虽然能够暴力水过去，但是无疑思考更一般的通解是很有必要的。
* 后来听说翁教他们当时是分块暴力重建凸包过去的，我就去请教了一下翁教，终于由之前的公式发现了
* 最优解在凸包上的这一重大性质。
* 此题正解应该是cdq分治+维护动态凸包，或者分块暴力重建凸包。而我的做法是CDQ维护凸包。具体是这样的：
*
* 首先得先分析出最优解一定在凸包上这一性质：
* 考虑这样的两颗糖果A(p1,q1),B(p2,q2)，假设A比B优，那么存在这样的表达式：p1*x+q1*y≥p2*x+q2*y,
* 变形一下式子之后得到这样的表达式：p2-p1q2-q1≤-yx(x>0且q2>q1)，也就是说，只要A，B满足上述式
* 子就一定存在A比B更优。我们可以将q看成横坐标，p看成纵坐标，将每一颗糖看成一个点。假设K=-yx，
* 考虑这样的三个点A(q1,p1),B(q2,p2),C(q3,p3)。 设AB的斜率为k1， BC的斜率为k2， 则A比B优
* 需要满足k1<K，B比C更优需要满足k2<K，那么B比A,C都优的话，就需要满足k2<K<k1，也就是说需要满
* 足k2<k1，这种情况只有在三个点构成的图为上凸壳的时候才满足，如果图形是下凸壳的话，中间的B点
* 一定不可能比AC都优。
* 于是这里可以知道的是对于特定的K来说，最优解一定在可选点构成的上凸包上面，由于之前限定
* 了(x>0且q2>q1)当这个条件改变的时候，最优解也可能在下凸包上。于是我们只需要维护一个动态的凸包，
* 对于每一个询问，在相应的上凸壳或下凸壳上二分斜率就能找到最优解。
*
* 至于这里如何维护凸包的话，如果做过维护动态凸包的类似题的话，此时肯定就迎刃而解。
*
* 我的方法是用CDQ分治做这个凸包。

```
* 对于一个区间[l,r]，我首先将存在于[l, mid]中的点集取出来做出一个凸包，对于[l, mid]之间的
* '-1'操作，如果删除的是[l, mid]之间的点，那么就正常的删点；如果删的点是l左边的点，那么这个删除操作
* 就忽略。然后利用构建好的这个凸包去更新[mid+1, r]中的所有询问操作。
*
* 看似好像没问题，但是实际上这是错的…
* 因为在区间[mid+1, r]中可能存在删除操作，删掉了之前构建好的凸包上面的点，导致更新后面的询问出现错误，
* 那么我们还需要做的就是在删除点之后，继续维护这个凸包。对于[mid+1, r]中的加点操作，我们是可以忽略的，
* 因为后面更新右区间的时候也会更新的。
* 删除点的维护操作是这样的，首先现在凸包上面找到这个点，如果没找到说明不会改变凸包，但是
* 也要将这个点标记为已删除的点。如果这个点在凸包上面，那么找到这个点的在凸包上的前驱和后继，
* 将该点删除，然后在最开始构筑凸包的那个数组中，重新构造前驱到后继之间的凸包，最后在加上后面的凸包
* 即可。
* 做完这一步的话，就可以正常的更新所有询问咯~
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <map>
//#include <unordered_map>
#define N 50010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const LL INF=0x3f3f3f3f3f3f3f3fLL;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    int x, y;
    int ty, opid;
    bool operator<(const Point& o)const{
        return x < o.x || (x == o.x && y < o.y);
    }
}can[N], p1[N], p2[N], pt[N], tmp[N];
int n;
int m1, m2;
map<PII, int> mp;
```

```
LL ans[N];
bool vis[N];
int cnt;

LL Cross(Point A, Point B){return (LL)A.x * B.y - (LL)A.y * B.x;}
Point operator-(Point A, Point B) {return (Point){A.x - B.x, A.y -
B.y};}
void getConvex(int n)
{
    m1 = m2 = 0;
    sort(pt, pt+n);
    for(int i = 0; i < n; i++)
    {
        while(m1 > 1 && Cross(pt[i] - p1[m1-2], p1[m1-1] - p1[m1-2])
<= 0) m1--;
        p1[m1++] = pt[i];
    }
    for(int i = n-1; i >= 0; i--)
    {
        while(m2 > 1 && Cross(pt[i] - p2[m2-2], p2[m2-1] - p2[m2-2])
<= 0) m2--;
        p2[m2++] = pt[i];
    }
}

void updateConvex(Point deltP)
{
    int idx = -1;
    for(int i = 0; i < m1; i++)
    {
        if(p1[i].x == deltP.x && p1[i].y == deltP.y){idx = i;break;}
    }
    if(idx != -1)
    {
        int l = idx - 1, r = idx + 1;
        int tail = 0;
        for(int i = r+1; i < m1; i++)
            tmp[tail++] = p1[i];
        int lpt, rpt;
        if(l == -1) lpt = -1;
        if(r == m1) rpt = cnt - 1;
        for(int i = 0; i < cnt; i++){
            if(l != -1 && pt[i].x == p1[l].x && pt[i].y == p1[l].y &&
vis[pt[i].opid]) lpt = i;
            if(r != m1 && pt[i].x == p1[r].x && pt[i].y == p1[r].y &&
vis[pt[i].opid]) rpt = i;
        }
        ///
        m1 = l+1;
        for(int i = lpt + 1; i <= rpt; i++)
        {
            if(vis[pt[i].opid] == 0) continue;
            while(m1 > 1 && Cross(pt[i] - p1[m1-2], p1[m1-1] - p1[m1-
2]) <= 0) m1--;
            p1[m1++] = pt[i];
        }
        for(int i = 0; i < tail; i++)
            p1[m1++] = tmp[i];
    }

    idx = -1;
```

```
    for(int i = 0; i < m2; i++)
    {
        if(p2[i].x == deltP.x && p2[i].y == deltP.y) {idx = i;break;}
    }
    if(idx != -1)
    {
        int l = idx - 1, r = idx + 1;
        int tail = 0;
        for(int i = r+1; i < m2; i++)
            tmp[tail++] = p2[i];
        int lpt, rpt;
        if(l == -1) lpt = cnt;
        if(r == m2) rpt = 0;
        for(int i = 0; i < cnt; i++){
            if(l != -1 && pt[i].x == p2[l].x && pt[i].y == p2[l].y &&
vis[pt[i].opid]) lpt = i;
            if(r != m2 && pt[i].x == p2[r].x && pt[i].y == p2[r].y &&
vis[pt[i].opid]) rpt = i;
        }
        ///
        m2 = l+1;
        for(int i = lpt - 1; i >= rpt; i--)
        {
            if(vis[pt[i].opid] == 0) continue;
            while(m2 > 1 && Cross(pt[i] - p2[m2-2], p2[m2-1] - p2[m2-
2]) <= 0) m2--;
            p2[m2++] = pt[i];
        }
        for(int i = 0; i < tail; i++)
            p2[m2++] = tmp[i];
    }
}

void updateans(int x, int y, int idx)
{
    Point o =(Point){x, -y};
    if(m1 <= 2){
        for(int i = 0; i < m1; i++)
            ans[idx] = max(ans[idx], (LL)p1[i].x * y + (LL)p1[i].y *
x);
    }else{
        int lb = -1, ub = m1-1;
        while(lb + 1 < ub)
        {
            int mid = lb + ub >> 1;
            if(Cross(p1[mid+1] - p1[mid], o) <= 0) lb = mid;
            else ub = mid;
        }
        lb = max(0, lb);
        ans[idx] = max(ans[idx], (LL)p1[lb].x * y + (LL)p1[lb].y * x);
        if(ub < m1) ans[idx] = max(ans[idx], (LL)p1[ub].x * y +
(LL)p1[ub].y * x);
    }
    if(m2 <= 2) {
        for(int i = 0; i < m2; i++)
            ans[idx] = max(ans[idx], (LL)p2[i].x * y + (LL)p2[i].y *
x);
    }else{
        int lb = -1, ub = m2 - 1;
        while(lb + 1 < ub){
            int mid = lb + ub >> 1;
```

```
            if(Cross(p2[mid+1] - p2[mid], o) <= 0) lb = mid;
            else ub = mid;
        }
        lb = max(0, lb);
        ans[idx] = max(ans[idx],(LL)p2[lb].x * y + (LL)p2[lb].y * x);
        if(ub < m2) ans[idx] = max(ans[idx], (LL)p2[ub].x * y +
(LL)p2[ub].y * x);
    }
}
void divide(int l, int r)
{
    if(l >= r) return ;
    int mid = l + r >> 1;

    for(int i = l; i <= mid; i++) vis[i] = 0;
    for(int i = l; i <= mid; i++)
    {
        if(can[i].ty == 1) vis[i] = 1;
        if(can[i].ty == -1 && can[i].opid >= l && can[i].opid <= mid)
vis[can[i].opid] = 0;
    }
    cnt = 0;
    for(int i = l; i <= mid; i++)
        if(vis[i]) pt[cnt++] = can[i];

    getConvex(cnt);

    for(int i = mid+1; i <= r; i++){
        if(can[i].ty == 0) updateans(can[i].x, can[i].y, i);
        if(can[i].ty == -1 && can[i].opid >= l && can[i].opid <= mid
&& vis[can[i].opid]){
            vis[can[i].opid] = 0;
            updateConvex(can[can[i].opid]);
        }
    }

    divide(l, mid);
    divide(mid+1, r);
}
int main()
{
    Open();

    while(true)
    {
//        read(n);
        scanf("%d", &n);
        if(n == 0) break;
        mp.clear();
        for(int i = 0; i < n; i++)
        {
            int ty, x, y;
//            read(ty);read(x);read(y);
            scanf("%d%d%d", &ty, &x, &y);
            can[i] = (Point){y, x, ty, i};
            if(ty == 0) swap(can[i].x, can[i].y), ans[i] = -INF;
            if(ty == 1) mp[PII(y, x)] = i;
            if(ty == -1) can[i].opid = mp[PII(y, x)];
        }
        divide(0, n-1);
        for(int i = 0; i < n; i++)
```

```
        {
            if(can[i].ty == 0)
                printf("%I64d\n", ans[i]);
        }
//        cout<<endl;
    }
    return 0;
}
```

CDQ+BIT 维护最值及其方案数

```
/*
*  这里求最长的三维LIS，只需要排序x这一维，接下来的y，z就和之前以前处理即可
*  只是BIT这里需要处理一下方案数，处理方法如下代码
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100020
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;

const LL INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("/home/qingping/in.txt","r",stdin);
        //freopen("F:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
const LL mod = 1<<30;
LL n;
struct node{
    LL x, y, z, ans, num;
    void read()
    {
        scanf("%I64d%I64d%I64d", &x, &y, &z);
    }
    bool operator<(const node& o)const
    {
        return x < o.x || (x == o.x && y < o.y) || (x == o.x && y ==
o.y && z < o.z);
    }
}info[N];
LL tmp[N];
LL stay[N], staz[N];
PII c[N];
LL cmp(LL a, LL b)
```

```
{
    return info[a].y < info[b].y;
}
void add(LL x, LL val, LL num)
{
    if(x == 0) return ;
    for(LL i = x; i < n + 10; i += i & -i){
        if(c[i].first == val){
            c[i].second = (c[i].second + num) % mod;
        }else{
            c[i] = max(c[i], PII(val, num));
        }
    }
}
void set0(LL x)
{
    if(x == 0) return;
    for(LL i = x; i < n + 10; i += i & -i) c[i] = PII(0, 0);
}
PII getsum(LL x)
{
    PII res = PII(0, 0);
    for(LL i = x; i; i -= i & -i)
    {
        if(res.first == c[i].first){
            res.second = (res.second + c[i].second)%mod;
        }else
            res = max(res, c[i]);
    }
    return res;
}
void divide(LL l, LL r)
{
    if(l + 1 >= r) return ;
    LL mid = l + r >> 1;
    divide(l, mid);
    for(LL i = l; i < r; i++) tmp[i] = i;
    sort(tmp+l, tmp+mid, cmp);
    sort(tmp+mid, tmp+r, cmp);
    LL lid = l;
    for(LL i = mid; i < r; i++)
    {
        while(lid < mid && info[tmp[lid]].y <= info[tmp[i]].y)
            add(info[tmp[lid]].z, info[tmp[lid]].ans,
info[tmp[lid]].num), lid++;
        PII pp = getsum(info[tmp[i]].z);
        if(info[tmp[i]].ans == pp.first+1 && pp.first){
            info[tmp[i]].num = (pp.second + info[tmp[i]].num) % mod;
        } else if(info[tmp[i]].ans < pp.first + 1){
            info[tmp[i]].ans = pp.first+1;
            info[tmp[i]].num = pp.second;
        }
    }
    for(LL i = l; i < mid; i++)
        set0(info[i].z);
    divide(mid, r);
}
int main()
{
//    Open();
    LL T;scanf("%I64d", &T);
```

```
    while(T--)
    {
        scanf("%I64d", &n);
        LL ty = 0, tz = 0;
        for(LL i = 0; i < n; i++)
        {
            info[i].read();
            stay[ty++] = info[i].y;
            staz[tz++] = info[i].z;
            info[i].ans = 1;
            info[i].num = 1;
        }
        sort(stay, stay+ty);
        sort(staz, staz+tz);
        ty = unique(stay, stay+ty) - stay;
        tz = unique(staz, staz+tz) - staz;
        for(LL i = 0; i < n; i++){
            info[i].y = lower_bound(stay, stay+ty, info[i].y) - stay +
1;
            info[i].z = lower_bound(staz, staz+tz, info[i].z) - staz +
1;
        }
        sort(info, info+n);
        divide(0, n);
        LL ans = 0, num = 0;
        for(LL i = 0; i < n; i++)
        {
            if(ans == info[i].ans)
                num = (num + info[i].num) %mod;
            else if(ans < info[i].ans){
                ans = info[i].ans;
                num = info[i].num;
            }
        }
        printf("%I64d %I64d\n", ans, num);
    }
    return 0;
}
```

CDQ+树状数组+切比雪夫距离曼哈顿距离互化

```
/*
* 首先得将曼哈顿距离转化为切比雪夫距离，(x, y) --> (x - y, x + y)，由于可能有
负坐标，所以得加20000修正；
* 转化之后呢，这个问题就变为一个非常常见的二维区间和的问题啦。二维树状数组水之~
* 然而数组大小40000*40000，这怎么能开下呢！摔！（据说现场赛能开下。。hhh），想
过一下离散化，但是发现离散化
* 非常难处理，感觉略复杂，于是果断上了CDQ！CDQ的话，这个题也很好做，每个大区间中
考虑左区间的全部修改操作，
* 去更新右区间的所有查询操作。这里更新的话只需要将所有操作按照x坐标排序，存下y坐标
* 即可(不好讲。。具体看代码就知道了)
*/

#include <iostream>
#include <cstdio>
#include <stack>
```

```cpp
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
//        freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int n, m;
struct query{
    int op, x, val, id;
    int y1, y2;
    query(){}
    query(int o, int xx, int yy1, int yy2, int v, int i){
        op = o, x = xx, y1 = yy1, y2 = yy2, val = v, id = i;
    }

    bool operator< (const query& o) const{
        return x < o.x || (x == o.x && op < o.op);
    }
}qlist[N], que[N];
int ans[N];
int c[N];
int tmp[N];
int changetoX(int x, int y)
{
    return x - y + 20010;
}
int changetoY(int x, int y)
{
    return x + y + 20010;
}
void add(int x, int val)
{
    for(int i = x; i <= N; i += (-i) & i) c[i] += val;
}
int getsum(int x)
{
    int res = 0;
    for(int i = x; i > 0 ; i -= (-i) & i) res += c[i];
    return res;
}
void divide(int l, int r){
    if(l >= r) return ;
    int mid = l + r >> 1;
    int tail = 0;
    for(int i = l; i <= mid; i++){
```

```
        if(qlist[i].op == 1){
            que[tail++] = query(1, qlist[i].x, qlist[i].y1,
qlist[i].y1, qlist[i].val, qlist[i].id);
        }
    }
    for(int i = mid + 1; i <= r; i++) {
        if(qlist[i].op != 1){
            int val = qlist[i].val;
            que[tail++] = query(2, qlist[i].x - val - 1, qlist[i].y1 -
val, qlist[i].y2 + val, qlist[i].val, qlist[i].id);
            que[tail++] = query(3, qlist[i].x + val, qlist[i].y1 - val,
qlist[i].y2 + val, qlist[i].val, qlist[i].id);
        }
    }
    sort(que, que + tail);
    for(int i = 0 ; i < tail ; i++)
    {
        if(que[i].op == 1){
            add(que[i].y1, que[i].val);
        } else if(que[i].op == 2){
            tmp[que[i].id] = getsum(que[i].y2) - getsum(que[i].y1 - 1);
        } else {
            ans[que[i].id] += getsum(que[i].y2) - getsum(que[i].y1 - 1)
- tmp[que[i].id];
        }
    }
    for(int i = 0; i < tail; i++)
        if(que[i].op == 1) add(que[i].y1, -que[i].val);
    divide(l, mid);
    divide(mid+1, r);
}
int main()
{
//    Open();
    while(scanf("%d",&n)==1 && n){
        scanf("%d", &m);
        int qtail = 1;
        for(int i = 1 ; i <= m; i++)
        {
            ans[i] = 0;
            int op, x, y, val;
            scanf("%d%d%d%d", &op, &x, &y, &val);
            int curx = changetoX(x, y);
            int cury = changetoY(x, y);
            qlist[i] = query(op, curx, cury, cury, val, i);
        }
        divide(1, m);
        for(int i = 1; i <= m; i++)
        {
            if(qlist[i].op == 2){
                printf("%d\n", ans[i]);
            }
        }
    }
    return 0;
}
```

LCT 动态树

LCT+BIT

```
/*
LCT:

每次操作相当于只把区间[L,R]之间的边连起来，求联通分量的个数。
思路： 把操作排序后，对于区间[L,R]的操作，先把R所有v<R的边（R->v）加入集合， 用动态树+并查集维护[1,R]的联通分量的个数cnt
        那么，答案就是[L,R]区间的联通分量的个数+N-R+L-1;
        求区间[L,R]的联通分量的个数，可以考虑把所有[1,L-1]的边删掉后新形成的联通分量。
        假设当前的连通分量x为一棵树，那么删掉一条边肯定会形成一个新的连通分量，所以删掉所有[1,L]的边形成的连通分量的个数即
        这棵树上所有(u->v且v<L)的边的数量。
        如果当前联通分量是一棵树+一条边的话， 那么，如果树中有|x|-1条包含在[L,R]内的边，则不会形成新的连通分量，
        考虑到删掉的边的单调性，即如果对于一棵树加入一条新的边（u->v），那么这条边可以代替所有（u'->v' 且v'<v）的边，
        也就是说v'这条边就算在之后的操作中会被删掉也不会对答案产生影响，因此删掉这条边，并且将新边加入此树。

        故区间[L,R]的连通分量的个数即cnt+当前森林中(u->v,v<L)的边的数量；
        前者可以用LCT+并查集维护，后者可以用树状数组维护。

        对于当前新加入的边(u->v)如果u和v不在同一个连通分量，则用并查集合并他们并且将他们的LCT合并。
        否则的话， 将u所在的LCT中u->v这条链的边里面两点最小值最小的那一条边删除，并且在树状数组中将对应点也删除。

*/
#pragma comment(linker, "/STACK:1024000000")
#include <stdio.h>
#include <string.h>
#include <map>
#include <algorithm>
using namespace std ;

typedef long long LL ;

#define clr( a , x ) memset ( a , x , sizeof a )
#define ls ( o << 1 )
#define rs ( o << 1 | 1 )
#define lson ls , l , m
#define rson rs , m + 1 , r
#define root 1 , 1 , n
#define mid ( ( l + r ) >> 1 )

const int MAXN = 100005 ;
const int MAXE = 200005 ;
const int INF = 0x3f3f3f3f ;

struct Edge {
    int v , idx , n ;
```

```
    Edge () {}
    Edge ( int v , int idx , int n ) : v ( v ) , idx ( idx ) , n
( n ) {}
} ;

struct Node* null ;

struct Node {
    Node* c[2] ;
    Node* f ;
    bool flip ;
    int minv , val ;
    int eidx , idx ;
    void newnode ( int v , int i ) {
        c[0] = c[1] = f = null ;
        minv = val = v ;
        eidx = idx = i ;
        flip = 0 ;
    }
    void rev () {
        if ( this == null ) return ;
        swap ( c[0] , c[1] ) ;
        flip ^= 1 ;
    }
    void up () {
        if ( this == null ) return ;
        if ( val <= c[0]->minv && val <= c[1]->minv ) {
            minv = val ;
            eidx = idx ;
        } else if ( c[0]->minv <= c[1]->minv && c[0]->minv <= val ) {
            minv = c[0]->minv ;
            eidx = c[0]->eidx ;
        } else {
            minv = c[1]->minv ;
            eidx = c[1]->eidx ;
        }
    }
    void down () {
        if ( this == null ) return ;
        if ( flip ) {
            c[0]->rev () ;
            c[1]->rev () ;
            flip = 0 ;
        }
    }
    bool is_root () {
        return f == null || f->c[0] != this && f->c[1] != this ;
    }
    void sign_down () {
        if ( !is_root () ) f->sign_down () ;
        down () ;
    }
    void setc ( Node* o , int d ) {
        c[d] = o ;
        o->f = this ;
    }
    void rot ( int d ) {
        Node* p = f ;
        Node* g = f->f ;
        p->setc ( c[d] , !d ) ;
        if ( !p->is_root () ) g->setc ( this , f == g->c[1] ) ;
```

```
        else f = g ;
        setc ( p , d ) ;
        p->up () ;
    }
    void splay () {
        sign_down () ;
        while ( !is_root () ) {
            if ( f->is_root () ) rot ( this == f->c[0] ) ;
            else {
                if ( f == f->f->c[0] ) {
                    if ( this == f->c[0] ) f->rot ( 1 ) , rot ( 1 ) ;
                    else rot ( 0 ) , rot ( 1 ) ;
                } else {
                    if ( this == f->c[1] ) f->rot ( 0 ) , rot ( 0 ) ;
                    else rot ( 1 ) , rot ( 0 ) ;
                }
            }
        }
        up () ;
    }
    void access () {
        Node* o = this ;
        for ( Node* x = null ; o != null ; x = o , o = o->f ) {
            o->splay () ;
            o->setc ( x , 1 ) ;
            o->up () ;
        }
        splay () ;
    }
    void make_root () {
        access () ;
        rev () ;
    }
    void link ( Node* o ) {
        make_root () ;
        f = o ;
    }
    void cut () {
        access () ;
        c[0] = c[0]->f = null ;
        up () ;
    }
    void cut ( Node* o ) {
        make_root () ;
        o->cut () ;
    }
    int get_min ( Node* o ) {
        make_root () ;
        o->access () ;
        return o->eidx ;
    }
} ;

Node pool[MAXN + MAXE] ;
Node* cur ;
Node* node[MAXN] ;
Node* edge[MAXE] ;

Edge E[MAXE + MAXN] ;
int H[MAXN] , Q[MAXN] , cntE ;
```

```
int U[MAXE] , V[MAXE] ;
int ans[MAXN] ;
int p[MAXN] ;
int c[MAXN] ;
int n , m , q ;

int find ( int x ) {
    return p[x] == x ? x : ( p[x] = find ( p[x] ) ) ;
}

void init ( int n ) {
    cntE = 0 ;
    cur = pool ;
    cur->newnode ( INF , -1 ) ;
    null = cur ++ ;
    for ( int i = 1 ; i <= n ; ++ i ) {
        p[i] = i ;
        H[i] = -1 ;
        Q[i] = -1 ;
        c[i] = 0 ;
        cur->newnode ( INF , -1 ) ;
        node[i] = cur ++ ;
    }
}

void addedge ( int u , int v , int idx , int H[] ) {
    E[cntE] = Edge ( v , idx , H[u] ) ;
    H[u] = cntE ++ ;
}

void add ( int x , int v ) {
    for ( ; x <= n ; x += x & -x ) c[x] += v ;
}

int sum ( int x , int ans = 0 ) {
    for ( ; x > 0 ; x -= x & -x ) ans += c[x] ;
    return ans ;
}

void scanf ( int& x , char c = 0 ) {
    while ( ( c = getchar () ) < '0' ) ;
    x = c - '0' ;
    while ( ( c = getchar () ) >= '0' ) x = x * 10 + c - '0' ;
}

void solve () {
    int cnt = 0 ;
    int u , v ;
    init ( n ) ;
    for ( int i = 1 ; i <= n ; ++ i ) add ( i , -1 ) ;
    for ( int i = 0 ; i < m ; ++ i ) {
        scanf ( "%d%d" , &u , &v ) ;
        if ( u == v ) continue ;
        if ( u < v ) swap ( u , v ) ;
        addedge ( u , v , i , H ) ;
        U[i] = u ;
        V[i] = v ;
        cur->newnode ( v , i ) ;
        edge[i] = cur ++ ;
    }
    for ( int i = 0 ; i < q ; ++ i ) {
```

```
        scanf ( "%d%d" , &u , &v ) ;
        addedge ( v , u , i , Q ) ;
    }
    for ( int i = 1 ; i <= n ; ++ i ) {
        ++ cnt ;
        for ( int j = H[i] ; ~j ; j = E[j].n ) {
            int v = E[j].v , idx = E[j].idx ;
            if ( v == i ) continue ;
            int x = find ( i ) ;
            int y = find ( v ) ;
            if ( x != y ) {
                -- cnt ;
                p[x] = y ;
                edge[idx]->link ( node[i] ) ;
                edge[idx]->link ( node[v] ) ;
                add ( v , 1 ) ;
            } else {
             //   printf("%d____%d\n",i,v);
                int eidx = node[i]->get_min ( node[v] ) ;
             //   printf("dele V[%d]=%d\n",eidx,V[eidx]);
                if ( V[eidx] >= v ) continue ;
                edge[eidx]->cut ( node[U[eidx]] ) ;
                edge[eidx]->cut ( node[V[eidx]] ) ;
                add ( V[eidx] , -1 ) ;
                edge[idx]->link ( node[i] ) ;
                edge[idx]->link ( node[v] ) ;
                add ( v , 1 ) ;
            }
        }
    //    printf("R:%d\n",i);
        for ( int j = Q[i] ; ~j ; j = E[j].n ) {
            int v = E[j].v , idx = E[j].idx ;
            ans[idx] = cnt + sum ( v - 1 ) + n - i + v - 1 ;
          //  printf("idx:%d(%d->%d)__%d___%d\n",idx,i,v,cnt,sum(v-1));
        }
    }
    for ( int i = 0 ; i < q ; ++ i ) {
        printf ( "%d\n" , ans[i] ) ;
    }
}

int main () {
    while ( ~scanf ( "%d%d%d" , &n , &m , &q ) ) solve () ;
    return 0 ;
}
```

动态维护生成树

```
/*
*Description
SC省MY市有着庞大的地下水管网络，嘟嘟是MY市的水管局长（就是管水管的啦），嘟嘟作为水管局长的工作就是：每天供水公司可能要将一定量的水从x处送往y处，嘟嘟需要为供水公司找到一条从A至B的水管的路径，接着通过信息化的控制中心通知路径上的水管进入准备送水状态，等到路径上每一条水管都准备好了，供水公司就可以开始送水了。嘟嘟一次只能处理一项送水任务，等到当前的送水任务完成了，才能处理下一项。
在处理每项送水任务之前，路径上的水管都要进行一系列的准备操作，如清洗、消毒等等。嘟嘟在控制中心一声令下，这些水管的准备操作同时开始，但由于各条管道的长度、内径不同，进行准备操作需要的时间可能不同。供水公司总是希望嘟嘟能找到这样一条送水路径，路径上
```

的所有管道全都准备就绪所需要的时间尽量短。嘟嘟希望你能帮助他完成这样的一个选择路径的系统，以满足供水公司的要求。另外，由于MY市的水管年代久远，一些水管会不时出现故障导致不能使用，你的程序必须考虑到这一点。
不妨将MY市的水管网络看作一幅简单无向图（即没有自环或重边）：水管是图中的边，水管的连接处为图中的结点。

Input
输入文件第一行为3个整数：N，M，Q分别表示管道连接处（结点）的数目、目前水管（无向边）的数目，以及你的程序需要处理的任务数目（包括寻找一条满足要求的路径和接受某条水管坏掉的事实）。
以下M行，每行3个整数x，y和t，描述一条对应的水管。x和y表示水管两端结点的编号，t表示准备送水所需要的时间。我们不妨为结点从1至N编号，这样所有的x和y都在范围[1，N]内。
以下Q行，每行描述一项任务。其中第一个整数为k：若k=1则后跟两个整数A和B，表示你需要为供水公司寻找一条满足要求的从A到B的水管路径；若k=2，则后跟两个整数x和y，表示直接连接x和y的水管宣布报废（保证合法，即在此之前直接连接x和y尚未报废的水管一定存在）。

Output
按顺序对应输入文件中每一项k=1的任务，你需要输出一个数字和一个回车/换行符。该数字表示：你寻找到的水管路径中所有管道全都完成准备工作所需要的时间（当然要求最短）。

* 做法： 一般道路损坏的题目都是反过来，变成增加边，这个也是反过来变成加边，然后用LCT暴力搞即可
*/
```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <map>
//#include <unordered_map>
#define N 1200010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct edge{
    int u, v, w, useless, id;
    edge(){}
    edge(int _u, int _v, int _w, int _id){
        u = _u, v = _v, w = _w, useless = 0, id = _id;
    }
    bool operator < (const edge& o) const{
```

```
        return useless < o.useless || (useless == o.useless && w <
o.w);
    }
}e[N];
struct LCT{
    int ch[N][2],pre[N], rev[N];//rev这个数组是不能去掉的
    int ma[N], maid[N];
    bool rt[N];
    void Update_Add(int r,int d)
    {
//        if(!r)return;
//        key[r] += d;
//        add[r] += d;
//        Max[r] += d;
    }
    void Update_Rev(int r)
    {
        if(!r)return;
        swap(ch[r][0],ch[r][1]);
        rev[r] ^= 1;
    }
    void push_down(int r)
    {
//        if(add[r])
//        {
//            Update_Add(ch[r][0],add[r]);
//            Update_Add(ch[r][1],add[r]);
//            add[r] = 0;
//        }
        if(rev[r])
        {
            Update_Rev(ch[r][0]);
            Update_Rev(ch[r][1]);
            rev[r] = 0;
        }
    }
    void push_up(int r)
    {
        maid[r] = r;
        if(ma[maid[r]] < ma[maid[ch[r][0]]]) maid[r] = maid[ch[r][0]];
        if(ma[maid[r]] < ma[maid[ch[r][1]]]) maid[r] = maid[ch[r][1]];
//        Max[r] = max(max(Max[ch[r][0]],Max[ch[r][1]]),key[r]);
    }
    void init(int n, int m)
    {
        for(int i = 1; i <= n; i++)
            rev[i] = pre[i] = ch[i][0] = ch[i][1] = 0, rt[i] = true,
ma[i] = 0, maid[i] = i;
        for(int i = 1; i <= m; i++)
        {
            int id = i + n;
            rev[id] = pre[id] = ch[id][0] = ch[id][1] = 0;
            rt[id] = true;
            ma[id] = e[i].w;
            maid[id] = id;
        }
    }
    void Rotate(int x)
    {
        int y = pre[x], kind = ch[y][1]==x;
        ch[y][kind] = ch[x][!kind];
```

```
    pre[ch[y][kind]] = y;
    pre[x] = pre[y];
    pre[y] = x;
    ch[x][!kind] = y;
    if(rt[y])
        rt[y] = false, rt[x] = true;
    else
        ch[pre[x]][ch[pre[x]][1]==y] = x;
    push_up(y);
}
//P函数先将根结点到r的路径上所有的结点的标记逐级下放
void P(int r)
{
    if(!rt[r])P(pre[r]);
    push_down(r);
}
void Splay(int r)
{
    P(r);
    while( !rt[r] )
    {
        int f = pre[r], ff = pre[f];
        if(rt[f])
            Rotate(r);
        else if( (ch[ff][1]==f)==(ch[f][1]==r) )
            Rotate(f), Rotate(r);
        else
            Rotate(r), Rotate(r);
    }
    push_up(r);
}
int Access(int x)
{
    int y = 0;
    for( ; x ; x = pre[y=x])
    {
        Splay(x);
        rt[ch[x][1]] = true, rt[ch[x][1]=y] = false;
        push_up(x);
    }
    return y;
}
//判断是否是同根(真实的树，非splay)
bool judge(int u,int v)
{
    while(pre[u]) u = pre[u];
    while(pre[v]) v = pre[v];
    return u == v;
}
//先Access(r),形成一条路径，再使r成为它所在的树的根
void mroot(int r)
{
    Access(r);
    Splay(r);
    Update_Rev(r);
}
//调用后u是原来u和v的lca,v和ch[u][1]分别存着lca的2个儿子
//(原来u和v所在的2颗子树)
void lca(int &u,int &v)
{
```

```
        Access(v), v = 0;
        while(u)
        {
            Splay(u);
            if(!pre[u])return;
            rt[ch[u][1]] = true;
            rt[ch[u][1]=v] = false;
            push_up(u);
            u = pre[v = u];
        }
    }

    void link(int u,int v)
    {
        if(judge(u,v))
        {
    //        puts("-1");
            return;
        }
        mroot(u);//这里的换根操作需要特别注意，有的题目不能换根
        pre[u] = v;
    }
    //先将u变为将v与他的父亲连边切断
    void cut(int x, int v)
    {
        mroot(x);//这里的换根操作需要特别注意，有的题目不能换根
        Splay(v);
        pre[ch[v][0]] = pre[v];
        pre[v] = 0;
        rt[ch[v][0]] = true;
        ch[v][0] = 0;
        push_up(v);
    }
    //------------End---------------

    int queryid(int x, int y)
    {
        mroot(x);
        Access(y);
        Splay(y);
        return maid[y];
    }
}lct;
int pa[100100];
int find(int x)
{
    return pa[x] == x ? x : pa[x] = find(pa[x]);
}
bool unite(int u, int v)
{
    u = find(u), v = find(v);
    if(u == v) return false;
    pa[u] = v;
    return true;
}
int n, m, q;
int getint()
{
    char ch = getchar();
    for ( ; ch > '9' || ch < '0'; ch = getchar());
    int tmp = 0;
```

```
    for ( ; '0' <= ch && ch <= '9'; ch = getchar())
        tmp = tmp * 10 + int(ch) - 48;
    return tmp;
}
struct QUE{
    int op, u, v, id;
    QUE(){}
    QUE(int _op, int _u, int _v, int _w){
        op = _op, u = _u, v = _v, id = _w;
    }
}que[100010];
bool cmp(const edge& a, const edge& b){
    return a.id < b.id;
}
int ans[N];
struct Item{
        LL key,val,nxt;
};
struct UMP{
        Item item[N];
        LL itnum;
        LL head[1999993];
        LL MOD;
        UMP(){
                MOD=1999993;
                clear();
        }
        void clear(){
                memset(head,-1,sizeof(head));
                itnum=0;
        }
        LL& operator[](LL x){
                LL idx=x%MOD;
                //cerr<<idx<<endl;
                for(LL i=head[idx];i!=-1;i=item[i].nxt){
                        if(item[i].key==x) return item[i].val;
                }

                item[itnum]=(Item){x,0,head[idx]};
                head[idx]=itnum;
                return item[itnum++].val;
        }
}mp;
int main()
{
//    Open();
    while(~scanf("%d%d%d", &n, &m, &q))
    {
        mp.clear();
        for(int i = 0; i <= n; i++) pa[i] = i;
        for(int i = 1; i <= m; i++)
        {
            int u = getint(), v = getint(), w = getint();
            if(v < u) swap(u, v);
            LL hs = (LL)u * 100001LL + v;
            mp[hs] = i;
            e[i] = edge(u, v, w, i);
        }
        for(int i = 0; i < q; i++)
        {
            int op = getint();
```

```
            int u = getint(), v = getint(), id = 0;
            if(op == 2){
                if(v < u) swap(u, v);
                LL hs = (LL)u * 100001LL + v;
                id = mp[hs];
                e[id].useless = 1;
            }
            que[i] = QUE(op, u, v, id);
        }
        lct.init(n, m);
        sort(e+1, e+m+1);
        for(int i = 0; i < m; i++)
        {
            if(e[i].useless == 1) break;
            if(unite(e[i].u, e[i].v)){
                lct.link(e[i].u, e[i].id + n);
                lct.link(e[i].v, e[i].id + n);
            }
        }
        sort(e+1, e+1+m, cmp);
        int tail = 0;
        for(int i = q-1; i >= 0; i--)
        {
            int u = que[i].u, v = que[i].v, id = que[i].id, op =
que[i].op;
            if(op == 1){
                int idx = lct.queryid(u, v);
                ans[tail++] = lct.ma[idx];
            }else{
                int idx = lct.queryid(u, v);
                if(e[id].w < lct.ma[idx])
                {
                    lct.cut(e[idx-n].u, idx);
                    lct.cut(e[idx-n].v, idx);
                    lct.link(u, id+n);
                    lct.link(v, id+n);
                }
            }
        }
        for(int i = tail-1 ; i >= 0; i--)
            printf("%d\n", ans[i]);
    }
    return 0;
}
```

LCT，树链两点之间的最大连续和

```
//BZOJ_3669_维护最小生成树
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 200010
//#define lson x<<1
//#define rson x<<1|1
```

```
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("F:/in.txt","r",stdin);
        //freopen("F:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
#define max8(a,b,c,d,e,f,g,h)
max(a,max(b,max(c,max(d,max(e,max(f,max(g,h)))))))
vector<int> G[N];
int n, m;
struct LCT{
    int ch[N][2],pre[N], rev[N], val[N], mark[N], sz[N];//rev这个数组是
不能去掉的
    int L[N], R[N], opt[N], sum[N];
    bool rt[N];
    void Treaval(int x) {
        if(x) {
            Treaval(ch[x][0]);
//            printf("结点%2d:左儿子 %2d 右儿子 %2d 父结点 %2d size
= %2d ,key = %2d \n",x, ch[x][0], ch[x][1], pre[x], sz[x], opt[x]);
            Treaval(ch[x][1]);
        }
    }
    void Update_Mark(int r,int d)
    {
        if(!r)return;
        val[r] = mark[r] = d;
        opt[r] = R[r] = L[r] = max(sz[r]*d, d);
        sum[r] = sz[r] * d;
    }
    void Update_Rev(int r)
    {
        if(!r)return;
        swap(ch[r][0],ch[r][1]);
        swap(L[r], R[r]);
        rev[r] ^= 1;
    }
    void push_down(int r)
    {
        if(mark[r] != INF)
        {
            Update_Mark(ch[r][0],mark[r]);
            Update_Mark(ch[r][1],mark[r]);
            mark[r] = INF;
        }
        if(rev[r])
        {
            Update_Rev(ch[r][0]);
            Update_Rev(ch[r][1]);
            rev[r] = 0;
        }
    }
    void push_up(int r)
```

```
    {
        if(!r) return;
        int lc = ch[r][0], rc = ch[r][1];
        sz[r] = sz[lc] + sz[rc] + 1;
        sum[r] = sum[lc] + sum[rc] + val[r];
        L[r] = max(L[lc], sum[lc] + val[r] + max(0, L[rc]));
        R[r] = max(R[rc], sum[rc] + val[r] + max(0, R[lc]));
        opt[r] = max8(opt[lc], opt[rc], L[r], R[r], val[r],
val[r]+L[rc]+R[lc], val[r]+L[rc], val[r]+R[lc]);
//       Max[r] = max(max(Max[ch[r][0]],Max[ch[r][1]]),key[r]);
    }
    void init(int n)
    {
        opt[0] = mark[0] = L[0] = R[0] = -INF;
        sum[0] = sz[0] = 0;
        for(int i = 1; i <= n; i++)
        {
            rev[i] = pre[i] = ch[i][0] = ch[i][1] = 0, rt[i] = true;
            L[i] = R[i] = opt[i] = -INF, mark[i] = INF;
            sz[i] = 1;
        }
    }
    void Rotate(int x)
    {
        int y = pre[x], kind = ch[y][1]==x;
        ch[y][kind] = ch[x][!kind];
        pre[ch[y][kind]] = y;
        pre[x] = pre[y];
        pre[y] = x;
        ch[x][!kind] = y;
        if(rt[y])
            rt[y] = false, rt[x] = true;
        else
            ch[pre[x]][ch[pre[x]][1]==y] = x;
        push_up(y);
    }
    //P函数先将根结点到r的路径上所有的结点的标记逐级下放
    void P(int r)
    {
        if(!rt[r])P(pre[r]);
        push_down(r);
    }
    void Splay(int r)
    {
        P(r);
        while( !rt[r] )
        {
            int f = pre[r], ff = pre[f];
            if(rt[f])
                Rotate(r);
            else if( (ch[ff][1]==f)==(ch[f][1]==r) )
                Rotate(f), Rotate(r);
            else
                Rotate(r), Rotate(r);
        }
        push_up(r);
    }
    int Access(int x)
    {
        int y = 0;
        for( ; x ; x = pre[y=x])
```

```
        {
            Splay(x);
            rt[ch[x][1]] = true, rt[ch[x][1]=y] = false;
            push_up(x);
        }
        return y;
    }
    //判断是否是同根(真实的树，非splay)
    bool judge(int u,int v)
    {
        while(pre[u]) u = pre[u];
        while(pre[v]) v = pre[v];
        return u == v;
    }
    //先Access(r),形成一条路径，再使r成为它所在的树的根
    void mroot(int r)
    {
        Access(r);
        Splay(r);
        Update_Rev(r);
    }
    //------------End---------------
    void update(int u, int v, int va)
    {
        mroot(u);
        Access(v);
        Splay(v);
        Update_Mark(v, va);
    }
    int query(int u, int v)
    {
        mroot(u);
        Access(v);
        Splay(v);
        return opt[v];
    }
}lct;
void dfs(int u, int pa)
{
    lct.pre[u] = pa;
    for(int i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i];
        if(v == pa) continue;
        dfs(v, u);
    }
}
int main2()
{
//    Open();
    while(~scanf("%d%d", &n, &m))
    {
        for(int i = 1; i <= n; i++)
            scanf("%d", &lct.val[i]), lct.sum[i] = lct.val[i],
G[i].clear();
        lct.init(n);
        for(int i = 1; i < n; i++)
        {
            int u, v;scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
```

```
        }
        dfs(1, 0);
        while(m--)
        {
            int op, a, b, c;
            scanf("%d%d%d%d", &op, &a, &b, &c);
            if(op == 1){
                lct.update(a, b, c);
            }else{
                printf("%d\n", lct.query(a, b));
            }
        }
    }
    exit(0);
}

extern int main2(void) __asm__ ("_main2");
int main()
{
    int size = 256 << 20;  // 256Mb
    char *p = (char *)malloc(size) + size;
    __asm__ __volatile__(
        "mov  %0, %%rsp\n"    //这里很多时候会报错"bad register name
'%rsp'"此时只需要将rsp换成esp就行了(原理就是两个不同的寄存器，在某些平台上名字不
同)
        "push $_exit\n"
        "jmp _main2\n"
        :: "r"(p));
    return 0;
}
```

SPLAY

HDU 4453

```
/*
* HDU 4453 Looploop
* 。。。这个题太恶心了，写得精疲力尽，但是无疑是对splay又更加理解了
* 六种操作：
* 1. add x: 找到树中的第k2+1个点，旋转为根，然后给左儿子打标记即可
* 2. reverse: 找到树中的第k1+1个点，旋转为根，然后给左儿子打标记即可
* 3. insert x: 啊。。我这里是比较丑的写法，直接找到最左边的儿子，
*             把新节点变成这个节点的父亲，然后暴力往上push_up。。但是
*             其实利用splay旋转的性质完全可以不这么做，比如说将最左边
*             儿子旋转到根，然后再插入，这样更新会方便一些吧。
* 4. delete: 就只是删除最左边一个点而已，旋转到根，然后删。
* 5. move x: 将第一个点放到最后去，也就是将第一个点删除，然后给尾巴添加
*           一个新节点。挺简单的一个操作。不过我这里写的太挫了。。
* 6. query: 这个不用说啦，找到最左一个点，然后输出即可。
*
* 需要注意的就是push_down, push_up需要随时考虑用不用
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
```

```
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 300010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int a[N];
int val[N];
int pre[N], ch[N][2], add[N], flip[N], sz[N];
int tot, root;
struct Splay
{
    void Treaval(int x) {
        if(x) {
            Treaval(ch[x][0]);
            printf("结点%2d:左儿子 %2d 右儿子 %2d 父结点 %2d size
= %2d ,key = %2d \n",x, ch[x][0], ch[x][1], pre[x], sz[x], val[x]);
            Treaval(ch[x][1]);
        }
    }
    void init()
    {
        tot = root = 0;
    }
    int newnode(int fa, int v)
    {
        int k = ++tot;
        ch[k][0] = ch[k][1] = 0;
        pre[k] = fa;
        val[k] = v;
        add[k] = 0, flip[k] = 0;
        sz[k] = 1;
        return k;
    }
    void push_down(int x)
    {
        if(add[x]){
            int lc = ch[x][0];
            int rc = ch[x][1];
            if(lc != 0) val[lc] += add[x], add[lc] += add[x];
            if(rc != 0) val[rc] += add[x], add[rc] += add[x];
            add[x] = 0;
        }
        if(flip[x]){
            flip[x] = 0;
            swap(ch[x][0], ch[x][1]);
```

```
        if(ch[x][0] != 0) flip[ch[x][0]] ^= 1;
        if(ch[x][1] != 0) flip[ch[x][1]] ^= 1;
    }
}
void push_up(int x)
{
    sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + 1;
}
void rotate(int x)
{
    int y = pre[x], d = (ch[y][1] == x);
    push_down(y);push_down(x);
    ch[y][d] = ch[x][!d];
    if(ch[x][!d]) pre[ch[x][!d]] = y;
    ch[x][!d] = y;
    pre[x] = pre[y];
    pre[y] = x;
    if(pre[x]) ch[pre[x]][ch[pre[x]][1] == y] = x;
    push_up(y);
    push_up(x);
}
void splay(int x, int goal)
{
    while(pre[x] != goal){
        int f = pre[x], ff = pre[f];
        if(ff == goal) rotate(x);
        else if((ch[ff][1] == f) == (ch[f][1] == x))
            rotate(f), rotate(x);
        else
            rotate(x), rotate(x);
    }
    push_up(x);
    if(goal == 0) root = x;
}
int kth(int k)
{
    int x = root;
    while(x)
    {
        push_down(x);
        if(sz[ch[x][0]] >= k) x = ch[x][0];
        else {
            k -= sz[ch[x][0]] + 1;
            if(k == 0) return x;
            x = ch[x][1];
        }
    }
    return x;
}
int build(int l, int r, int fa){
    if(l > r) return 0;
    int mid = l + r >> 1;
    int k = newnode(fa, a[mid]);
    ch[k][0] = build(l, mid-1, k);
    ch[k][1] = build(mid+1, r, k);
    push_up(k);
    return k;
}
void increase(int k, int x)
{
    int idx = kth(k + 1);
```

```
        if(idx == 0){
            val[root] += x;
            add[root] += x;
            return ;
        }
        splay(idx, 0);
        val[ch[idx][0]] += x;
        add[ch[idx][0]] += x;
    }
    void reverse(int k)
    {
        int idx = kth(k + 1);
        if(idx == 0){
            flip[root] ^= 1;
            return ;
        }
        splay(idx, 0);
//        Treaval(root);
        flip[ch[idx][0]] ^= 1;
    }
    void remove()
    {
        int x = root;
        push_down(x);
        while(ch[x][0]) x = ch[x][0], push_down(x);
        splay(x, 0);
        if(ch[x][0] == 0){
            root = ch[x][1];
            pre[ch[x][1]] = 0;
            return ;
        }
        int y = ch[x][0];
        push_down(y);
        while(ch[y][1]) y = ch[y][1], push_down(y);
        splay(y, x);
        ch[y][1] = ch[x][1];
        if(ch[x][1]) pre[ch[x][1]] = y;
        pre[y] = 0;
        root = y;
        push_up(y);
    }
    void insert(int v){
        if(ch[root][0] == 0) splay(ch[root][1], 0);
        int x = root;
        push_down(x);
        while(ch[x][0]) x = ch[x][0], push_down(x);
        int tmpk = newnode(pre[x], v);
        ch[tmpk][0] = x, ch[tmpk][1] = ch[x][1];
        ch[pre[x]][0] = tmpk;

        if(ch[x][1] != 0) pre[ch[x][1]] = tmpk;
        ch[x][1] = 0;
        pre[x] = tmpk;
        while(x) push_up(x), x = pre[x];
    }
    void move(int kind){
        if(kind == 1){
            int x = root;push_down(x);
            while(ch[x][1]) x = ch[x][1], push_down(x);
            splay(x, 0);
            int y = ch[x][0];push_down(y);
```

```
            while(ch[y][1]) y = ch[y][1], push_down(y);
            splay(y, x);
            pre[y] = 0;
            root = y;
            push_up(y);

            ch[x][1] = ch[x][0] = 0;
            y = root;push_down(y);
            while(ch[y][0]) y = ch[y][0], push_down(y);
            splay(y, 0);
            ch[y][0] = x; pre[x] = y;
            sz[x] = 1;
            push_up(y);
        }else{
            int x = root; push_down(x);
            while(ch[x][0]) x = ch[x][0], push_down(x);
            splay(x, 0);
            int y = ch[x][1]; push_down(y);
            while(ch[y][0]) y = ch[y][0], push_down(y);
            splay(y, x);
            pre[y] = 0; root = y;
            push_up(y);

            ch[x][1] = ch[x][0] = 0;
            y = root; push_down(y);
            while(ch[y][1]) y = ch[y][1], push_down(y);
            splay(y, 0);
            ch[y][1] = x, pre[x] = y;
            sz[x] = 1; push_up(y);
        }
    }
    int query()
    {
        int x = root;push_down(x);
        while(ch[x][0]) x = ch[x][0], push_down(x);
        return val[x];
    }
}spl;
char tmp[22];
int main()
{
    Open();
    int n, m, k1, k2;
    int cas = 1;
    while(scanf("%d%d%d%d", &n, &m, &k1, &k2), n||m||k1||k2)
    {
        for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
        spl.init();
        root = spl.build(1, n, 0);
//        spl.Treaval(root);
        printf("Case #%d:\n", cas++);
        while(m --)
        {
            scanf("%s", tmp);
            if(tmp[0] == 'm')
            {
                int kind;
                scanf("%d", &kind);
                spl.move(kind);
            }
            if(tmp[0] == 'q')
```

```
            {
                printf("%d\n", spl.query());
            }
            if(tmp[0] == 'i')
            {
                int v;scanf("%d", &v);
                spl.insert(v);
            }
            if(tmp[0] == 'r')
            {
                spl.reverse(k1);
            }
            if(tmp[0] == 'a')
            {
                int v;scanf("%d", &v);
                spl.increase(k2, v);
            }
            if(tmp[0] == 'd')
            {
                spl.remove();
            }
//            printf("-------------------------%s--------------------
---\n", tmp);
//            spl.Treaval(root);
            int asdfasd = 11212;
        }
    }
    return 0;
}
```

HDU 3436

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 400010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<char,int> PCI;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int a[N];
int n, m;
int st[N], ed[N];
```

```
//int mx[N];//存储以该节点为根的子树中的最大值
int sz[N];//存储以该结点为根的子树中的节点数
int pre[N], ch[N][2];//该节点的儿子和父亲
int val[N];
int tot, root;//tot总共有多少个节点，root根节点的编号
int key[N];
int rev[N];
struct Splay
{
    void init()
    {
        tot = root = 0;
    }
    int newnode(int fa, int idx, int preidx)
    {
        int k;
        if(!preidx) k = ++tot;
        else k = preidx;
        key[idx] = k;
        rev[k] = idx;
        ch[k][0] = ch[k][1] = 0;
        pre[k] = fa;
//        val[k] = v;
        val[k] = sz[k] = ed[idx] - st[idx] + 1;
        return k;
    }
    void push_up(int x)
    {
        sz[x] = sz[ch[x][1]] + sz[ch[x][0]] + val[x];
    }
    void rotate(int x)
    {
        int y = pre[x], d = (ch[y][1] == x);
        //push_down(y);push_down(x);
        ch[y][d] = ch[x][!d];
        if(ch[x][!d]) pre[ch[x][!d]] = y;
        ch[x][!d] = y;
        pre[x] = pre[y];
        pre[y] = x;
        if(pre[x]) ch[pre[x]][ch[pre[x]][1] == y] = x;
        push_up(y);
        push_up(x);
    }
    void splay(int x, int goal)//将x旋转到goal的下面
    {
        while(pre[x] != goal)
        {
            int f = pre[x], ff = pre[f];
            if(ff == goal) rotate(x);
            else if((ch[ff][1] == f) == (ch[f][1] == x))
                rotate(f), rotate(x);
            else
                rotate(x), rotate(x);
        }
        push_up(x);
        if(goal == 0) root = x;//根的父亲为null
    }
    int kth(int k)//寻找splay中的第k大的节点
    {
        int x = root;
```

```
        while(x)
        {
            if(sz[ch[x][0]] >= k) x = ch[x][0];
            else {
                k -= sz[ch[x][0]];
                if(k <= val[x]) return st[rev[x]] + k - 1;
                else k -= val[x];
//                if(k == 0) return st[rev[x]];
                x = ch[x][1];
            }
        }
        return st[rev[x]];
    }
    int build(int l, int r, int fa)
    {
        if(l > r) return 0;
        int mid = l + r >> 1;
        int k = newnode(fa, mid, 0);
        ch[k][0] = build(l, mid-1, k);
        ch[k][1] = build(mid+1, r, k);
        push_up(k);
        return k;
    }
    void remove(int x)
    {
        splay(x, 0);
        if(ch[x][0] == 0){
            root = ch[x][1];
            pre[ch[x][1]] = 0;
            return ;
        }
        int y = ch[x][0];
        while(ch[y][1]) y =ch[y][1];
        splay(y, x);
        ch[y][1] = ch[x][1];
        if(ch[x][1]) pre[ch[x][1]] = y;
        pre[y] = 0;
        root = y;
        return ;
    }
    void insert(int v, int preidx)//将节点插入树中，并让其变为树中最左端的节
点
    {
        int x = root;
        while(ch[x][0]) x = ch[x][0];
        int tmp = newnode(x, v, preidx);
        ch[x][0] = tmp;
        while(x) push_up(x), x = pre[x];
        splay(tmp, 0);
    }
}spl;
PCI query[N];
char op[11];
int sta[N];
int binaryS(int l, int r, int x)
{
    int lb = l - 1, ub = r + 1;
    int mid;
    while(lb + 1 < ub)
    {
        mid = lb + ub >> 1;
```

```
        if(st[mid] <= x && ed[mid] >= x) return mid;
        else if(st[mid] > x) ub = mid;
        else if(ed[mid] < x) lb = mid;
    }
    return mid;
}
int main()
{
    Open();
    int T;scanf("%d", &T);
    int cas = 1;
    while(T--)
    {
        scanf("%d%d", &n, &m);
        spl.init();
        int cnt = 0;
        sta[cnt++] = 1;
        for(int i = 0; i < m; i++)
        {
            int x;
            scanf("%s%d", op, &x);
            query[i] = PCI(op[0], x);
            if(op[0] != 'R')
                sta[cnt++] = x;
        }
        sta[cnt++] = n;
        sort(sta, sta+cnt);
        cnt = unique(sta, sta+cnt) - sta;
        int tail = 0;
        for(int i = 0; i < cnt; i++){
            if(i && sta[i] != sta[i-1] + 1){
                st[tail] = sta[i-1] + 1;
                ed[tail++] = sta[i] - 1;
            }
            st[tail] = ed[tail] = sta[i];
            tail++;
        }
        root = spl.build(0, tail - 1, 0);
        for(int i = 0; i < m; i++)
            if(query[i].first != 'R')
                query[i].second = binaryS(0, tail, query[i].second);
        printf("Case %d:\n", cas++);
        for(int i = 0; i < m; i++)
        {
            if(query[i].first == 'Q') {
                int x = query[i].second;
                spl.splay(key[x], 0);
                int res = sz[ch[root][0]] + 1;
                printf("%d\n", res);
            }else if(query[i].first == 'R') {
                printf("%d\n", spl.kth(query[i].second));
            }else if(query[i].first == 'T') {
                int x = query[i].second;
                spl.remove(key[x]);
                spl.insert(x, key[x]);
            }
        }
    }
    return 0;
}
```

HDU 1754

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 200010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int a[N];
int n, m;
struct Splay
{
    int mx[N];//存储以该节点为根的子树中的最大值
    int sz[N];//存储以该结点为根的子树中的节点数
    int pre[N], ch[N][2];//该节点的儿子和父亲
    int val[N];
    int tot, root;//tot总共有多少个节点，root根节点的编号

    void init()
    {
        tot = root = 0;
    }
    int newnode(int fa, int v)
    {
        int k = ++tot;
        ch[k][0] = ch[k][1] = 0;
        pre[k] = fa;
        mx[k] = v;
        val[k] = v;
        sz[k] = 1;
        return k;
    }
    void push_up(int x)
    {
        mx[x] = max(mx[ch[x][0]], mx[ch[x][1]]);
        mx[x] = max(mx[x], val[x]);
        sz[x] = sz[ch[x][1]] + sz[ch[x][0]] + 1;
    }
    void rotate(int x)
    {
        int y = pre[x], d = (ch[y][1] == x);
        //push_down(y);push_down(x);
```

```
        ch[y][d] = ch[x][!d];
        if(ch[x][!d]) pre[ch[x][!d]] = y;
        ch[x][!d] = y;
        pre[x] = pre[y];
        pre[y] = x;
        if(pre[x]) ch[pre[x]][ch[pre[x]][1] == y] = x;
        push_up(y);
        push_up(x);
    }
    void splay(int x, int goal)//将x旋转到goal的下面
    {
        while(pre[x] != goal)
        {
            int f = pre[x], ff = pre[f];
            if(ff == goal) rotate(x);
            else if((ch[ff][1] == f) == (ch[f][1] == x))
                rotate(f), rotate(x);
            else
                rotate(x), rotate(x);
        }
        push_up(x);
        if(goal == 0) root = x;//根的父亲为null
    }
    int kth(int k)//寻找splay中的第k大的节点
    {
        int x = root;
        while(x)
        {
            if(sz[ch[x][0]] >= k) x = ch[x][0];
            else {
                k -= sz[ch[x][0]] + 1;
                if(k == 0) return x;
                x = ch[x][1];
            }
        }
        return x;
    }
    int build(int l, int r, int fa)
    {
        if(l > r) return 0;
        int mid = l + r >> 1;
        int k = newnode(fa, a[mid]);
        ch[k][0] = build(l, mid-1, k);
        ch[k][1] = build(mid+1, r, k);
        push_up(k);
        return k;
    }
}spl;
int op[11];
int main()
{
    Open();
    while(~scanf("%d%d", &n, &m))
    {
        for(int i = 1; i <= n; i++)
            scanf("%d", &a[i]);
        a[0] = a[n+1] = -INF;
        spl.init();
        spl.root = spl.build(0, n+1, 0);
        while(m--)
        {
```

```
            scanf("%s", op);
            if(op[0] == 'Q'){
                int x, y;
                scanf("%d%d", &x, &y);
                spl.splay(spl.kth(x), 0);
                spl.splay(spl.kth(y+2), spl.root);
                int idx = spl.ch[spl.root][1];
                idx = spl.ch[idx][0];
                printf("%d\n", spl.mx[idx]);
            }else {
                int x, v;
                scanf("%d%d", &x, &v);
//                spl.splay(spl.kth(x), 0);
//                spl.splay(spl.kth(x+2), spl.root);
//                int y = spl.ch[spl.root][1];
//                spl.ch[y][0] = spl.newnode(y, v);
//                spl.push_up(y);
//                spl.push_up(spl.root);
                int idx = spl.kth(x+1);
                spl.splay(idx, 0);
                spl.val[idx] = v;
                spl.push_up(idx);
            }
        }
    }
    return 0;
}
```

可持久化字典树

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("F:/in.txt","r",stdin);
        //freopen("F:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}

struct node{
    int go[2];
```

```
    int cnt;
}pool[N*20];
int tot;//
vector<int> G[N];//
int pa[20][N];//
int w[N];//
int dep[N];//
int n,m;
int root[N];//

int insert(int pre, int val)
{
    int p = ++tot, ret = p;
    pool[p] = pool[pre];
    for(int i = 15; i >= 0; i--)
    {
        int tmp = (val>>i)&1;
        int cur = ++tot;
        pool[cur] = pool[pool[p].go[tmp]];
        pool[cur].cnt++;
        pool[p].go[tmp] = cur;
        p = cur;
    }
    return ret;
}
void dfs(int v, int p, int d)
{
    root[v] = insert(root[max(0, p)], w[v]);
    pa[0][v] = p;
    dep[v] = d;
    for(int i = 0; i < G[v].size(); i ++)
        if(G[v][i] != p) dfs(G[v][i], v, d+1);
}
void init()
{
    tot = 0;
    root[0] = 0;
    memset(pool, 0, sizeof(pool));
    dfs(1, -1, 0);
    for(int k = 0; k + 1 < 20; k++)
        for(int v = 1; v <= n; v++)
            if(pa[k][v] < 0) pa[k+1][v] = -1;
            else pa[k+1][v] = pa[k][pa[k][v]];
}
int lca(int u, int v)
{
    if(dep[u] > dep[v]) swap(u, v);
    for(int k = 0; k < 20; k++)
        if((dep[v] - dep[u]) >> k & 1)
            v = pa[k][v];
    if(u == v) return u;
    for(int k = 19; k >= 0; k--)
        if(pa[k][u] != pa[k][v])
            u = pa[k][u], v = pa[k][v];
    return pa[0][u];
}
int getans(int u, int v, int val)
{
    int LCA = lca(u, v);
    int pu = root[u], pv = root[v], pl = root[LCA];
    int ans = 0;
```

```
    for(int i = 15; i >= 0; i--)
    {
        int tmp = (val >> i)&1;
        int sum = pool[pool[pu].go[!tmp]].cnt +
pool[pool[pv].go[!tmp]].cnt - 2 * pool[pool[pl].go[!tmp]].cnt;
        if(sum > 0){
            pu = pool[pu].go[!tmp];
            pv = pool[pv].go[!tmp];
            pl = pool[pl].go[!tmp];
            ans += 1<<i;
        }else{
            pu = pool[pu].go[tmp];
            pv = pool[pv].go[tmp];
            pl = pool[pl].go[tmp];
        }
    }
    return max(val ^ w[LCA], ans);
}
int main()
{
//    Open();
    while(~scanf("%d%d", &n, &m))
    {
        for(int i = 1; i <= n; i++)
            scanf("%d", &w[i]), G[i].clear();
        for(int i = 1; i < n; i++)
        {
            int u, v;scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        init();
        while(m--)
        {
            int u, v, z;
            scanf("%d%d%d", &u, &v, &z);
            printf("%d\n", getans(u, v, z));
        }
    }
    return 0;
}
```

树状数组

HDU 5517 二维树状数组-三维偏序

```
/*
* 题意：给出n个(a, b), m个(c, d, e)，构造一个集合(a, c, d)当存在b=e的时候，
* 问构造出来的集合中有多少个TOP三元组(不存在三个元素都大于该三元组的三元组)
*
* 首先将m个三元组排序，第一关键字为ci，第二关键字为di，从大到小排序。
* 然后将ai和bi处理为一个vector，用二维树状数组搞一搞即可
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
```

```
#include <queue>
#include <algorithm>
#include <cmath>
#include <set>
//#include <unordered_map>
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("F:/in.txt","r",stdin);
        //freopen("F:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
#define N 100100
int man, mam;
int c[1110][1110];
void Update(int x, int y, int a)
{
    for(int i = x; i ; i -= i & -i)
        for(int j = y; j ; j -= j & -j)
            c[i][j] = max(c[i][j], a);
}
int getmax(int x, int y)
{
    int res = 0;
    for(int i = x; i < man + 10; i += i & -i)
        for(int j = y; j < mam + 10; j += j & -j)
            res = max(res, c[i][j]);
    return res;
}

template<class T>
inline bool read(T &n){
    T x = 0, tmp = 1; char c = getchar();
    while ((c < '0' || c > '9') && c != '-' && c != EOF) c =
getchar();
    if (c == EOF) return false;
    if (c == '-') c = getchar(), tmp = -1;
    while (c >= '0' && c <= '9') x *= 10, x += (c - '0'), c =
getchar();
    n = x*tmp;
    return true;
}
struct node
{
    int c, d, e;
    void input()
    {
        read(c);
        read(d);
        read(e);
//        scanf("%d%d%d", &c, &d, &e);
    }
```

```
    bool operator<(const node& o)const
    {
        return c > o.c || (c == o.c && d > o.d);
    }
    bool cmp(const node& o)const
    {
        return c == o.c && d == o.d;
    }
}E[N];
vector<int> B[N];
PII maB[N];
PII BB[N];
int main()
{
//    Open();
    int T;//scanf("%d", &T);
    read(T);
    int cas = 1;
    while(T--)
    {
        memset(c, 0, sizeof(c));
        memset(maB, 0, sizeof(maB));
        int n, m;
        read(n);read(m);
        int mab = 0;
//        scanf("%d%d", &n, &m);
        for(int i = 0; i < n; i++)
        {
            int a, b;
//            scanf("%d%d", &a, &b);
            read(a);read(b);
            mab = max(mab, b);
            BB[i] = PII(a, b);
            B[b].push_back(a);
        }
        for(int i = 1; i <= mab; i++)
        {
            if(B[i].size() == 0) continue;
            sort(B[i].begin(), B[i].end(), greater<int>());
            int j = 0;
            while(j < B[i].size() && B[i][j] == B[i][0]) j++;
            maB[i] = PII(B[i][0], j);
        }
        man = mam = 0;
        for(int i = 0; i < m; i++)
        {
            E[i].input();
            man = max(man, E[i].c);
            mam = max(mam, E[i].d);
        }
        sort(E, E+m);
        LL ans = 0;
        for(int i = 0; i < m; i++)
        {
            int ed = i; int ma = 0;
            while(ed < m && E[ed].cmp(E[i]))
            {
                int b = E[ed].e;
                ma = max(ma, maB[b].first);
                ed++;
            }
```

```
            LL sum = 0;
            if(ma != 0)
            {
                for(int j = i; j < ed; j++)
                {
                    int b = E[j].e;
                    if(ma == maB[b].first) sum += maB[b].second;
                }
                int exitma = getmax(E[i].c, E[i].d);
                if((exitma != 0 && exitma < ma) || exitma == 0)
                {
                    ans += sum;
                }
                Update(E[i].c, E[i].d, ma);
            }
            i = ed-1;
        }
        for(int i = 0; i < n; i++)
            B[BB[i].second].clear();
        printf("Case #%d: %I64d\n",cas++, ans);
    }
    return 0;
}
```

HDU 5372 树状数组

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <queue>
#include <cstring>
#include <string>
#include <cmath>
#include <set>
#include <map>
#include <vector>
#include <climits>
using namespace std;
#define pb push_back
#define ALL(x) x.begin(),x.end()
#define VINT vector<int>
#define PII pair<int,int>
#define MP(x,y) make_pair((x),(y))
#define ll long long
#define ull unsigned ll
#define MEM0(x)  memset(x,0,sizeof(x))
#define MEM(x,val) memset((x),val,sizeof(x))
#define scan(x) scanf("%d",&(x))
#define scan2(x,y) scanf("%d%d",&(x),&(y))
#define scan3(x,y,z) scanf("%d%d%d",&(x),&(y),&(z))
#define scan4(x,y,z,k) scanf("%d%d%d%d",&(x),&(y),&(z),&(k))
#define Max(a,b) a=max(a,b)
#define Min(a,b) a=min(a,b)
using namespace std;
const int N=400045;
int mp[N];
struct node
{
    int ty, l, r;
```

```
    node(){}
    node(int ty, int l, int r):ty(ty), l(l), r(r){}
}q[N];
int n;
int idx[N];
int cnt = 0;//
int rc[N];//
int lc[N];//
void add(int c[], int x, int val)
{
    if( x == 0 ) return ;
    for(int i=x;i <= cnt+10;i += i & (-i)) c[i] += val;
}
int getsum(int c[], int x)
{
    int rnt = 0;
    for(int i = x;i>0;i -= i & (-i)) rnt += c[i];
    return rnt;
}
template<class T>
inline bool read(T &n){
    T x = 0, tmp = 1; char c = getchar();
    while ((c < '0' || c > '9') && c != '-' && c != EOF) c =
getchar();
    if (c == EOF) return false;
    if (c == '-') c = getchar(), tmp = -1;
    while (c >= '0' && c <= '9') x *= 10, x += (c - '0'), c =
getchar();
    n = x*tmp;
    return true;
}
template <class T>
inline void write(T n) {
    if (n < 0) {
        putchar('-');
        n = -n;
    }
    int len = 0, data[20];
    while (n) {
        data[len++] = n % 10;
        n /= 10;
    }
    if (!len) data[len++] = 0;
    while (len--) putchar(data[len] + 48);
}
int main()
{
#ifndef ONLINE_JUDGE
    freopen("D:/in.txt","r",stdin);
#endif
    int Cas = 1;
    while(~scanf("%d", &n))
    {
        printf("Case #%d:\n",Cas++);
        memset(rc, 0, sizeof(rc));
        memset(lc, 0, sizeof(lc));
        cnt = 0;
        int len = 1;
        for(int i=1;i<=n;i++)
        {
            int a, b;
```

```
            scanf("%d%d", &a, &b);
//            read(a);read(b);
            if(a == 0){
                q[i] = node(a, b, b+len);
                mp[cnt++] = b, mp[cnt++] = b+len;
                idx[len] = i;
                len++;
            }else{
                q[i] = node(a, q[idx[b]].l, q[idx[b]].r);
            }
        }
        sort(mp, mp+cnt);
        cnt = unique(mp, mp+cnt) - mp;
        for(int i=1;i<=n;i++)
        {
            q[i].l = lower_bound(mp, mp+cnt, q[i].l) - mp + 1;
            q[i].r = lower_bound(mp, mp+cnt, q[i].r) - mp + 1;
        }
        for(int i=1;i<=n;i++)
        {
            if(q[i].ty == 0) {
                printf("%d\n", getsum(rc, q[i].r) - getsum(lc, q[i].l -
1));
                add(rc, q[i].r, 1), add(lc, q[i].l, 1);
            }else {
                add(rc, q[i].r, -1), add(lc, q[i].l, -1);
            }
        }
    }
    return 0;
}
```

HDU 4777 离线查找一个区间内有多少完整区间

```
/*
* 首先先预处理出L[i],R[i]数组，分别表示小于i的最靠后的一个与i不互质的下标，和大于i的最靠前的一个与i不互质的下标。
* 大体思路就是计算每个区间有多少个会打架的人，对于一个询问，我先减去这个区间(L<=i<=R)里面L[i]>=L的有多少个，
* R[i]<=R的有多少个，全加起来，但是这样会加重了L[i],R[i]都在区间里面的数，再减去这些就行了。
* 对于每一个i来说，有三个区间[L[i], i], [i, R[i]], [L[i], R[i]]，答案就是第一种区间数+第二种-第三种
* 先读入所有询问[L,R]，再将所有上述区间的右端点全部存入树状数组中(+1)，那么我只需要计算在[L, R]中有多少个1就行了，
* 但是这样会计算到左端点小于L的区间，于是我们只需要在扫完i之后，将以i为左端点的区间的右端点全部减去(-1)即可。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 200210
```

```
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
const int MAXN = 222222;
int last[MAXN];
int L[MAXN] , R[MAXN];
int f[MAXN];
int zhishu[666];
int tail;
int n,m;
int prm[888+11];
bool visP[888+11];
int pn;
void getPrm(){
    for(int i=2;i<=888;i++){
        if(!visP[i]){
            prm[pn++] = i;
            for(int j=i;j<=888;j+=i) visP[j]=true;
        }
    }
}

void fenjie(int x){
    tail=0;
    for(int i=0;i<pn && prm[i] * prm[i] <= x;i++){
        if(x % prm[i] == 0){
            zhishu[tail++] = prm[i];
            while(x % prm[i] == 0) x /= prm[i];
        }
    }
    if(x > 1) zhishu[tail ++ ] = x;
    int sldfkjadsf=1;
}
void getLR(){
    for(int i=1;i<=n;i++){
        fenjie(f[i]);
        int maxer = 0;
        for(int j=0;j<tail;j++){
            maxer = max(maxer , last[zhishu[j]] );
        }
        L[i] = maxer;
        for(int j=0;j<tail;j++){
            last[zhishu[j]] = i;
        }
    }
    fill(last+1 , last+MAXN , n+1 );
    for(int i=n;i>=1;i--){
        fenjie(f[i]);
        int miner = n+1;
```

```
        for(int j=0;j<tail;j++){
            miner = min(miner , last[zhishu[j]] );
        }
        R[i] = miner;
        for(int j=0;j<tail;j++){
            last[zhishu[j]] = i;
        }
    }
}
int c1[N], c2[N];
vector<int> vli[N];
vector<int> vlr[N];
vector<PII > vqu[N];
int ans[N];
void add(int c[], int x, int val)
{
    if(x == 0) return ;
    for(int i = x; i <= n+10; i += (-i) & i) c[i] += val;
}
int getsum(int c[], int x)
{
    int res = 0;
    for(int i = x; i > 0; i -= (-i) & i) res += c[i];
    return res;
}
int main()
{
    getPrm();
    while(scanf("%d%d",&n,&m)==2){
        memset(last , 0 , sizeof(last));
        memset(c1, 0, sizeof(c1));
        memset(c2, 0, sizeof(c2));
        for(int i=1;i<=n;i++){
            vli[i].clear();
            vlr[i].clear();
            vqu[i].clear();
            scanf("%d",&f[i]);
        }
        getLR();
        for(int i=1;i<=n;i++){
            if(L[i] != 0){
                vli[L[i]].push_back(i);
                vlr[L[i]].push_back(R[i]);
                add(c1, i, 1);
                add(c2, R[i], 1);
            }
            vli[i].push_back(R[i]);
            add(c1, R[i], 1);
        }
        for(int i = 1; i <= m; i++)
        {
            int l, r;
            scanf("%d%d", &l, &r);
            vqu[l].push_back(PII(r, i));
        }
        for(int i = 1; i <= n; i++)
        {
            for(int j = 0; j < vqu[i].size(); j ++)
            {
                int r = vqu[i][j].first, idx = vqu[i][j].second;
                ans[idx] = (r - i + 1) - (getsum(c1, r) - getsum(c1, i
```

```
- 1)) + (getsum(c2, r) - getsum(c2, i - 1));
            }
            for(int j = 0; j < vli[i].size(); j++){
                add(c1, vli[i][j], -1);
            }
            for(int j = 0; j < vlr[i].size(); j++){
                add(c2, vlr[i][j], -1);
            }
        }
        for(int i = 1;i <= m; i++)
            printf("%d\n", ans[i]);
    }
    return 0;
}
```

线段树

主席树-区间 K 大

```
/* ***********************************************
Author       :kuangbin
Created Time  :2013-9-4 20:13:20
File Name     :POJ2104.cpp
************************************************ */

#include <stdio.h>
#include <string.h>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <string>
#include <math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;

const int MAXN = 100010;
const int M = MAXN * 30;
int n,q,m,tot;
int a[MAXN], t[MAXN];
int T[M], lson[M], rson[M], c[M];

void Init_hash()
{
    for(int i = 1; i <= n;i++)
        t[i] = a[i];
    sort(t+1,t+1+n);
    m = unique(t+1,t+1+n)-t-1;
}
int build(int l,int r)
{
    int root = tot++;
    c[root] = 0;
    if(l != r)
```

```
    {
        int mid = (l+r)>>1;
        lson[root] = build(l,mid);
        rson[root] = build(mid+1,r);
    }
    return root;
}
int hash(int x)
{
    return lower_bound(t+1,t+1+m,x) - t;
}
int update(int root,int pos,int val)
{
    int newroot = tot++, tmp = newroot;
    c[newroot] = c[root] + val;
    int l = 1, r = m;
    while(l < r)
    {
        int mid = (l+r)>>1;
        if(pos <= mid)
        {
            lson[newroot] = tot++; rson[newroot] = rson[root];
            newroot = lson[newroot]; root = lson[root];
            r = mid;
        }
        else
        {
            rson[newroot] = tot++; lson[newroot] = lson[root];
            newroot = rson[newroot]; root = rson[root];
            l = mid+1;
        }
        c[newroot] = c[root] + val;
    }
    return tmp;
}
int query(int left_root,int right_root,int k)
{
    int l = 1, r = m;
    while( l < r)
    {
        int mid = (l+r)>>1;
        if(c[lson[left_root]]-c[lson[right_root]] >= k )
        {
            r = mid;
            left_root = lson[left_root];
            right_root = lson[right_root];
        }
        else
        {
            l = mid + 1;
            k -= c[lson[left_root]] - c[lson[right_root]];
            left_root = rson[left_root];
            right_root = rson[right_root];
        }
    }
    return l;
}
int main()
{
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
```

```
    while(scanf("%d%d",&n,&q) == 2)
    {
        tot = 0;
        for(int i = 1;i <= n;i++)
            scanf("%d",&a[i]);
        Init_hash();
        T[n+1] = build(1,m);
        for(int i = n;i ;i--)
        {
            int pos = hash(a[i]);
            T[i] = update(T[i+1],pos,1);
        }
        while(q--)
        {
            int l,r,k;
            scanf("%d%d%d",&l,&r,&k);
            printf("%d\n",t[query(T[l],T[r+1],k)]);
        }
    }
    return 0;
}
```

主席树-区间多少个不同数

```
map<int,int>mp;
int a[N],tot,n,q;
int T[M],lson[M],rson[M],val[M];
int bulid(int l,int r){
    int root=tot++;
    val[root]=0;
    int m=(l+r)>>1;
    if(l!=r){
        lson[root]=bulid(l,m);
        rson[root]=bulid(m+1,r);
    }
    return root;
}
int update(int root,int pos,int v){
    int newroot=tot++,tmp=newroot;
    int l=1,r=n;
    val[newroot]=val[root]+v;
    while(l<r){
        int m=(l+r)>>1;
        //更新的时候需要充分利用历史信息
        //更新原来的左子树，右子树不变
        if(pos<=m){
            lson[newroot]=tot++;rson[newroot]=rson[root];
            newroot=lson[newroot];root=lson[root];
            r=m;
        }
        //更新右子树
        else{
            rson[newroot]=tot++;lson[newroot]=lson[root];
            newroot=rson[newroot];root=rson[root];
            l=m+1;
        }
        val[newroot]=val[root]+v;
    }
    return tmp;
```

```
}
int query(int root,int pos){
    int ret=0;
    int l=1,r=n;
    while(pos<r){
        int m=(l+r)>>1;
        if(pos<=m){
            r=m;
            root=lson[root];
        }
        else{
            ret+=val[lson[root]];
            root=rson[root];
            l=m+1;
        }
    }
    return ret+val[root];
}
int main(){
    while(scanf("%d",&n)!=EOF){
        tot=0;   //结点数
        for(int i=1;i<=n;i++)
            scanf("%d",&a[i]);
        T[n+1]=bulid(1,n);
        for(int i=n;i;i--){
            int nxt;
            map<int,int>::iterator it=mp.find(a[i]);
            if(it==mp.end()) nxt=n+1;
            else nxt=it->second;
            //如果这是第一次出现，也就是最后一个位置上，则直接更新
            if(nxt>n)
                T[i]=update(T[i+1],i,1);
            //在原来的位置上擦掉，在当前位置更新
            else{
                int t=update(T[i+1],nxt,-1);
                T[i]=update(t,i,1);
            }
            mp[a[i]]=i;
        }
        scanf("%d",&q);
        while(q--){
            int l,r;
            scanf("%d%d",&l,&r);
            printf("%d\n",query(T[l],r));
        }
    }
    return 0;
}
```

区间乘，区间除-扩展 GCD 求(互质)逆元

```
/*
* 题意：先给出n个数，然后有区间乘和区间除的操作。要求对于每个询问操作输出当前区间的乘积mod M
*
* 做法：这里由于M是给出来的数，并不一定是质数，所以不能直接求逆，但是对于与M互质的a来说，a却是
*       有逆元存在的，所以这里线段树每个节点维护M的所有质因子表(存储指数数目)，每
```

次修改先将操作数分解
*　　质因数，对于在M的质因子表中的质因子，则直接在指数上进行加减，而其余的就直接对M求逆即可。
*/

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <map>
#include <assert.h>
#include <cmath>
#include <time.h>
#define N 10010
#define lson x<<1
#define rson x<<1|1
#define mid ((lt[x].l+lt[x].r)/2)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
void gcd(int a, int b, int& d, int& x, int& y){
    if(!b){d = a; x = 1; y = 0;}
    else {gcd(b, a%b, d, y, x); y -= x * (a / b);}
}
int inv(int a, int n)
{
    int d, x, y;
    gcd(a, n, d, x, y);
    return d == 1 ? (x + n) % n : -1;
}
int pow_mod(int x, int k, int mod)
{
    int res = 1;
    while(k)
    {
        if(k & 1) res = (LL)res * x % mod;
        x = (LL)x * x % mod;
        k >>= 1;
    }
    return res;
}
int n, M;//
int Mplist[111];//
int b[N];//
int apcnt[N][20];//
int addpro;//
vector<PII> addp;//
map<int, int> pid;//
bool pvis[100010];//
int pri[100010];//
int prin;//
int mcnt = 0;//
void getpri(int UP){
    prin = 0;
    memset(pvis, 0, sizeof(pvis));
    for(int i = 2; i < UP; i++) {
        if(pvis[i] == 0) pri[prin++] = i;
        for(int j = 0; j < prin && (LL)pri[j]*i < UP; j++) {
            pvis[pri[j]*i] = 1;
```

```
            if(i % pri[j] == 0) break;
        }
    }
}
struct node{
    int l, r, pro;
    int pcnt[20];
    int lazy[20];
    int lazypro;
    bool la;
    bool exist0;
    node(){}
    node(int _l, int _r)
    {
        l = _l, r = _r;
        pro = 1;
        la = false;
        lazypro = 1;
        exist0 = false;
        memset(pcnt, 0, sizeof(pcnt));
        memset(lazy, 0, sizeof(lazy));
    }
    void set0()
    {
        pro = lazypro = la = 0;
        exist0 = 1;
    }
    int len()
    {
        return r - l + 1;
    }
}lt[N * 6];
void push_up(int x)
{
    for(int i = 0; i < addp.size(); i++){
        int id = addp[i].first;
        lt[x].pcnt[id] = lt[lson].pcnt[id] + lt[rson].pcnt[id];
    }
    lt[x].pro = (LL)lt[lson].pro * lt[rson].pro % M;
    lt[x].exist0 = lt[lson].exist0 || lt[rson].exist0;
    if(lt[lson].lazypro == 0 && lt[rson].lazypro == 0) lt[x].lazypro
= 0;
}
void push_down(int x)
{
    if(lt[x].lazypro == 0){
        lt[lson].set0();
        lt[rson].set0();
        return ;
    }
    if(lt[x].la){
        lt[x].la = false;
        for(int i = 0; i < mcnt; i++) {
            if(lt[x].lazy[i] == 0) continue;
            lt[lson].pcnt[i] += lt[x].lazy[i] * lt[lson].len();
            lt[rson].pcnt[i] += lt[x].lazy[i] * lt[rson].len();
            lt[lson].lazy[i] += lt[x].lazy[i];
            lt[rson].lazy[i] += lt[x].lazy[i];
            lt[x].lazy[i] = 0;
        }
        lt[lson].la = true;
```

```
        lt[rson].la = true;
    }
    if(lt[x].lazypro > 1){
        lt[lson].pro = (LL)lt[lson].pro * pow_mod(lt[x].lazypro,
lt[lson].len(), M) % M;
        lt[rson].pro = (LL)lt[rson].pro * pow_mod(lt[x].lazypro,
lt[rson].len(), M) % M;
        lt[lson].lazypro = (LL)lt[lson].lazypro * lt[x].lazypro % M;
        lt[rson].lazypro = (LL)lt[rson].lazypro * lt[x].lazypro % M;
        lt[x].lazypro = 1;
    }
}
void build(int l, int r, int x)
{
    lt[x] = node(l, r);
    if(l == r)return ;
    build(l, mid, lson);
    build(mid+1, r, rson);
    push_up(x);
}
void update(int l, int r, int x)
{
    if(lt[x].lazypro == 0) return ;
    if(lt[x].l >= l && lt[x].r <= r){
        if(addpro == 0){
            lt[x].set0();
            return ;
        }
        lt[x].la = true;
        for(int i = 0; i < addp.size(); i++)
        {
            lt[x].lazy[addp[i].first] += addp[i].second;
            lt[x].pcnt[addp[i].first] += addp[i].second * lt[x].len();
        }
        lt[x].pro = (LL)lt[x].pro * pow_mod(addpro, lt[x].len(), M) %
M;
        lt[x].lazypro = (LL)lt[x].lazypro * addpro % M;
        return ;
    }
    push_down(x);
    if(r <= mid) update(l, r, lson);
    else if(l > mid) update(l, r, rson);
    else update(l, mid, lson), update(mid+1, r, rson);
    push_up(x);
}
int query(int l, int r, int x)
{
    if(lt[x].lazypro == 0) return 0;
    if(lt[x].l >= l && lt[x].r <= r)
    {
        if(lt[x].exist0) return 0;
        int res = lt[x].pro;
        for(int i = 0; i < mcnt; i++)
        {
            res = (LL)res * pow_mod(Mplist[i], lt[x].pcnt[i], M) % M;
        }
        return res % M;
    }
    push_down(x);
    if(r <= mid) return query(l, r, lson);
    else if(l > mid) return query(l, r, rson);
```

```
    else return (LL)query(l, mid, lson) * query(mid+1, r, rson) % M;
}
void getUpArg(int val, bool isMul)
{
    if(val == 0){
        addpro = 0;
        return ;
    }
    int res = 0;
    for(int i = 0; i < mcnt && Mplist[i] <= val; i++){
        while(val % Mplist[i] == 0){
            res++;
            val /= Mplist[i];
        }
        if(res){
            if(!isMul) res = -res;
            addp.push_back(PII(i, res));
            res = 0;
        }
    }
    addpro = val;
    if(!isMul) addpro = inv(addpro, M);
}
int main()
{
//    Open();
    getpri(100010);
    int T ;scanf("%d", &T);
    int cas = 1;
    while(T--)
    {
        pid.clear();
        memset(apcnt, 0, sizeof(apcnt));
        mcnt = 0;
        scanf("%d%d", &n, &M);
        int tmpm = M;
        for(int i = 0; i < prin && pri[i] <= tmpm; i++){
            assert(tmpm != 0);
            if(tmpm % pri[i] == 0) Mplist[mcnt] = pri[i], pid[pri[i]] =
mcnt++;
            while(tmpm % pri[i] == 0) tmpm /= pri[i];
        }
        if(tmpm > 1){
            Mplist[mcnt] = tmpm;
            pid[tmpm] = mcnt++;
        }
        build(1, n, 1);
        assert(mcnt < 20);
        for(int i = 1; i <= n; i++)
        {
            int tmp;scanf("%d", &tmp);
            getUpArg(tmp, 1);
            update(i, i, 1);
            addp.clear();
        }
        int q;scanf("%d" ,&q);
        printf("Case #%d:\n", cas++);
        while(q--)
        {
            char op[3];
            int L, R;
```

```
            scanf("%s%d%d", op, &L, &R);
            if(op[0] == 'M'){
                int x;scanf("%d", &x);
                getUpArg(x, 1);
                update(L, R, 1);
                addp.clear();
            }else if(op[0] == 'D') {
                int x ;scanf("%d", &x);
                getUpArg(x, 0);
                update(L, R, 1);
                addp.clear();
            }else{
                printf("%d\n", query(L, R, 1));
            }
        }
    }
    return 0;
}
```

线段树维护等差系数数组

```
/*
* Problem Description
She says that any Pavarotti among the nightingales will serenade his
mate while she sits on her eggs.
She says that any excited database can answer the queries
efficiently.

You are given the two dimensional database as a matrix A with n rows
and n columns. In the beginning, A[i][j]=0 for all 1≤i,j≤n.
Then q operations or queries will be given in turn.

You should maintain the database for two type of operations:
1 L R: for each element A[i][j] which satisfy L≤i+j≤R, increase the
value to A[i][j]+1, where 2≤L≤R≤2n.
2 L R: for each element A[i][j] which satisfy L≤i-j≤R, increase the
value to A[i][j]+1, where 1-n≤L≤R≤n-1.
Meanwhile, you should answer the queries:
3 x1 x2 y1 y2: count the value of elements A[i][j] which satisfy
x1≤i≤x2 and y1≤j≤y2, where 1≤x1<x2≤n and 1≤y1<y2≤n.


Input
The input contains several test cases. The first line of the input is
a single integer t which is the number of test cases. Then t test
cases follow.

Each test case contains several lines. The first line contains the
integer n and q.
The i-th line of the next q lines contains an operation ``1 L R" or
``2 L R", or a query ``3 x1 x2 y1 y2".

The sum of n for all test cases would not be larger than 200000 and
the sum of q would not be larger than 50000.
*
* 做法：这个题稍加分析就会发现，每一次改变值都是正副对角线方向上的所有数加值，那么
我们可以用两棵线段
*       树去分别维护正副对角线上的a[i]*i和a[i]的区间和，以此来计算所求矩形的和。
```

```
*
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 200010
#define lson x<<1
#define rson x<<1|1
#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct node
{
    LL l, r, sumi, sum, lazy;
    node(){}
    node(LL _l, LL _r, LL _sumi, LL _sum, LL _lazy)
    {
        l = _l, r = _r, sumi = _sumi, sum = _sum, lazy = _lazy;
    }
    LL len()
    {
        return r - l + 1;
    }
}lt1[N*6], lt2[N*6];
void build(LL l, LL r, LL x, node lt[])
{
    lt[x] = node(l, r, 0, 0, 0);
    if(l == r) return ;
    build(l, mid, lson, lt);
    build(mid+1, r, rson, lt);
}
void push_down(LL x, node lt[])
{
    if(lt[x].lazy)
    {
        lt[lson].sum += lt[x].lazy * lt[lson].len();
        lt[lson].sumi += lt[x].lazy * lt[lson].len() * (mid + lt[x].l)
/ 2;
        lt[lson].lazy += lt[x].lazy;
        lt[rson].sum += lt[x].lazy * lt[rson].len();
        lt[rson].sumi += lt[x].lazy * lt[rson].len() * (mid+1+lt[x].r)
/ 2;
        lt[rson].lazy += lt[x].lazy;
        lt[x].lazy = 0;
    }
```

```
}
void push_up(LL x, node lt[])
{
    lt[x].sumi = lt[lson].sumi + lt[rson].sumi;
    lt[x].sum = lt[lson].sum + lt[rson].sum;
}
void update(LL l, LL r, LL x, node lt[])
{
    if(lt[x].l >= l && lt[x].r <= r)
    {
        lt[x].sum += lt[x].len();
        lt[x].sumi += (r + l) * lt[x].len() / 2;
        lt[x].lazy ++;
        return ;
    }
    push_down(x, lt);
    if(r <= mid) update(l, r, lson, lt);
    else if(l > mid) update(l, r, rson, lt);
    else update(l, mid, lson, lt), update(mid+1, r, rson, lt);
    push_up(x, lt);
}
PII query(LL l, LL r, LL x, node lt[])
{
    if(lt[x].l >= l && lt[x].r <= r)
    {
        return PII(lt[x].sumi, lt[x].sum);
    }
    push_down(x, lt);
    if(r <= mid) return query(l, r, lson, lt);
    else if(l > mid)return query(l, r, rson, lt);
    else {
        PII p1 = query(l, mid, lson, lt);
        PII p2 = query(mid+1, r, rson, lt);
        return PII(p1.first + p2.first, p2.second + p1.second);
    }
}
LL getans(LL l, LL r, LL x, node lt[])
{
    PII res = query(l, r, x, lt);
    return res.first - (l - 1) * res.second;
}
int main()
{
    Open();
    LL T;scanf("%I64d", &T);
    LL cas = 1;
    while(T--)
    {
        LL n, q;
        scanf("%I64d%I64d", &n, &q);
        LL offset = n;
        printf("Case #%I64d:\n", cas++);
        build(1, 2*n, 1, lt1);
        build(1, 2*n, 1, lt2);
        while(q--)
        {
            LL op;
            scanf("%I64d", &op);
            if(op == 1) {
                LL l, r;
                scanf("%I64d%I64d", &l, &r);
```

```
            update(l, r, 1, lt1);
        }else if(op == 2){
            LL l, r;
            scanf("%I64d%I64d", &l, &r);
            update(l + offset, r + offset, 1, lt2);
        }else {
            LL x1, y1, x2, y2;
            scanf("%I64d%I64d%I64d%I64d", &x1, &x2, &y1, &y2);
            LL res = getans(x1+y1, x2+y2, 1, lt1);
            if(x1 + y2 + 1 <= x2 + y2)res -= getans(x1+y2+1, x2+y2,
1, lt1);
            if(x2 + y1 + 1 <= x2 + y2)res -= getans(x2+1+y1, x2+y2,
1, lt1);
            res += getans(x1 - y2 + offset, x2 - y1 + offset, 1,
lt2);
            if(x1 - (y1 - 1) <= x2 - y1) res -= getans(x1 - (y1 -
1) + offset, x2 - y1 + offset, 1, lt2);
            if(x2 + 1 - y2 <= x2 - y1) res -= getans(x2 + 1 - y2 +
offset, x2 - y1 + offset, 1, lt2);
            printf("%I64d\n", res);
        }
    }
}
return 0;
}
```

主席树-区间修改

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
#define M 2500010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}

int lson[M], rson[M], ma[M], tot, lazy[M], mi[M];//维护的信息
int Tn;
int dep[2][N];
int fa[2][N];
```

```
int st[N], ed[N];
vector<int> G[2][N];
int n, m;
int T[N];//存储每个节点的线段树的根节点编号。
void dfsseq(int u, vector<int> G[], int d)
{
    dep[1][u] = d;
    st[u] = ++Tn;
    for(int i = 0; i < G[u].size(); i ++)
        dfsseq(G[u][i], G, d + 1);
    ed[u] = Tn;
}
void dfsdep(int u, int d)
{
    dep[0][u] = d;
    for(int i = 0; i < G[0][u].size(); i ++)
        dfsdep(G[0][u][i], d+1);
}
int build(int l, int r)
{
    int root = tot++;
    ma[root] = 0, mi[root] = 0, lazy[root] = 0;
    if(l == r) return root;
    int mid = l + r >> 1;
    lson[root] = build(l, mid);
    rson[root] = build(mid+1, r);
    return root;
}
// 有的模板下放懒标记的时候，用一个数组标记了当前点的左右儿子是否是历史版本， 如果是的话再重新申请节点，如果不是就直接更新。
// 这么做的意义是为了节省空间，我个人测试之后发现这样做不得不再开一个与线段树大小相同的标记数组，其实大多数情况下内存是会更
// 大的...我一开始的做法就是，不管是不是历史版本， 只要懒标记有效，我就申请两个新节点当做儿子。
// 但是也会有恶心的数据会让这种不加标记的MLE的，所以注意取舍。
void push_down(int root){
    if(lazy[root]){
        lson[tot] = lson[lson[root]];
        rson[tot] = rson[lson[root]];
        ma[tot] = max(ma[lson[root]], lazy[root]);
        mi[tot] = max(mi[lson[root]], lazy[root]);
        lazy[tot] = max(lazy[lson[root]], lazy[root]);
        lson[root] = tot++;
        lson[tot] = lson[rson[root]];
        rson[tot] = rson[rson[root]];
        ma[tot] = max(ma[rson[root]], lazy[root]);
        mi[tot] = max(mi[rson[root]], lazy[root]);
        lazy[tot] = max(lazy[rson[root]], lazy[root]);
        rson[root] = tot++;
        lazy[root] = 0;
    }
}
void push_up(int root){
    ma[root] = max(ma[lson[root]], ma[lson[root]]);
    mi[root] = min(mi[lson[root]], mi[lson[root]]);
}
int update(int root, int L, int R, int l, int r, int val)// [l, r]修改区间，[L， R]当前区间
{
    if(mi[root] >= val) return root;
```

```
    int newroot = tot ++;
    if(L >= l && R <= r){
        lazy[newroot] = max(lazy[root], val);
        ma[newroot] = max(ma[root], val);
        mi[newroot] = max(mi[root], val);
        lson[newroot] = lson[root];
        rson[newroot] = rson[root];
        return newroot;
    }
    lazy[newroot] = ma[newroot] = 0, mi[newroot] = 0;
    push_down(root);
    int mid = L + R >> 1;
    if(r <= mid)
    {
        lson[newroot] = update(lson[root], L, mid, l, r, val);
        rson[newroot] = rson[root];
    }else if(l > mid){
        lson[newroot] = lson[root];
        rson[newroot] = update(rson[root], mid+1, R, l, r, val);
    }else{
        lson[newroot] = update(lson[root], L, mid, l, mid, val);
        rson[newroot] = update(rson[root], mid+1, R, mid+1, r, val);
    }
    push_up(newroot);
    return newroot;
}
int query(int root, int L, int R, int idx){
    if(L == R && L == idx) return ma[root];
    push_down(root);
    int mid = L + R >> 1;
    if(idx <= mid) return query(lson[root], L, mid, idx);
    else return query(rson[root], mid+1, R, idx);
}
void dfs(int u, vector<int> G[])
{
    T[u] = update(T[fa[0][u]], 1, Tn, st[u], ed[u], u);
    for(int i = 0; i < G[u].size(); i ++)
    {
        dfs(G[u][i], G);
    }
}

int main()
{
    Open();
    while(~scanf("%d%d", &n, &m)){
        Tn = 0;
        tot = 0;
        for(int i = 0; i <= n; i++) G[1][i].clear(), G[0][i].clear();
        for(int i = 2; i <= n; i++)
        {
            scanf("%d", &fa[0][i]);
            G[0][fa[0][i]].push_back(i);
        }
        for(int i = 2; i <= n; i++)
        {
            scanf("%d", &fa[1][i]);
            G[1][fa[1][i]].push_back(i);
        }
        dfsseq(1, G[1], 1);
        dfsdep(1, 1);
```

```
        fa[0][1] = n + 1;
        T[n+1] = build(1, Tn);
        dfs(1, G[0]);
        int preans = 0;
        while(m--)
        {
            int u, v;
            scanf("%d%d", &u, &v);
            u += preans; u %= n; u++;
            v += preans; v %= n; v++;
            preans = query(T[u], 1, Tn, st[v]);
            printf("%d %d %d\n", preans, dep[0][u] - dep[0][preans] +
1, dep[1][v] - dep[1][preans] + 1);
        }
    }
    return 0;
}
```

计算几何

凸包-极角排序-最小环

```
/*
* Problem Description
* Coach Zhang has an infinite ranch with N alpacas on it. The alpacas
are so lazy that they never move. The i-th (1≤i≤N) alpaca is located
at the point (Xi,Yi).
* Coach Zhang wants to monitor all the alpacas. He found M points
where cameras can be placed in his ranch. Coach Zhang only needs to
place one camera at a point if he thinks it necessary. An alpaca is
monitored when it is on one of the cameras, on the segment between
two cameras, or in the triangle formed by three cameras.
* Now Coach Zhang wants to know the minimum number of cameras
required to monitor all the alpacas.
*
* Input
* The first line of input contains an integer T, which represents the
number of test cases (T≤20).
* Each test case starts with a line containing N and M
(1≤N≤100000,1≤M≤500).
* Each of the next N+M lines contains two integers X and Y
(|X|,|Y|≤109). The first N lines represent the alpacas' positions and
the last M lines indicate the points that cameras can be placed.
*
* Output
* For each test case, output a single line consisting of "Case #X:
Y". X is the test case number starting from 1. Y is the minimum
number of cameras required to monitor all the alpacas. If it is
impossible to monitor all the alpacas, you should output -1 instead.
*
* 题意：有n头羊，m个监视器，给出监视器的位置，希望用最少的监视器监视到所有的羊，也
就是所有羊都在
*       监视器所组成的多边形内。
* 做法：先对n头羊做一个凸包，然后对监视器i，j，如果线段ij在凸包的右边且不与多边形
相交，那么建边
```

```
*       i->j,最后跑一边最小环即可。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
const double eps = 1e-8;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    LL x,y;
    Point(LL x = 0, LL y = 0):x(x),y(y){}
}pa[N],pb[N];
typedef Point Vector;
LL dcmp(double x){if(fabs(x) < eps) return 0;else return x<0?-1:1;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return
acos(Dot(A,B)/Length(A)/Length(B));}
LL Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
A.x/L);}//需要确保A不是0向量
bool OnSegment(Point p, Point a, Point b) { return dcmp(Length(p - a)
+ Length(p - b) - Length(a - b)) == 0; }//精度也很高的!
struct Line{
    Point p,v;
    double a,b,c,ang;//得到一般式的参数
    Line(Point p = Point(0,0), Vector v = Vector(0,0)):p(p),v(v){ang
= angle(v);}
```

```
    Point point(double t){return p + v*t;}//只能在点斜式中用
    bool operator < (const Line& L)const{
        return ang < L.ang;
    }
}L[N];
LL ConvexHull(Point* p, LL n, Point* ch)
{
    sort(p, p+n);
    LL m = 0;
    for(LL i = 0; i < n; i++)
    {
        while(m > 1 && Cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0)
m--;
        ch[m++] = p[i];
    }
    LL k = m;
    for(LL i = n - 2; i >= 0; i--)
    {
        while(m > k && Cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0)
m--;
        ch[m++] = p[i];
    }
    if(n > 1) m--;
    return m;
}
bool mp[505][505];
LL d[505];
LL n, m;
LL dij(LL s)
{
    memset(d, 0x3f, sizeof(d));
    priority_queue<PII, vector<PII >, greater<PII> > que;
    que.push(PII(0, s));
    d[s] = 0;
    while(!que.empty())
    {
        LL dis = que.top().first, u = que.top().second;
        que.pop();
        if(mp[u][s]) return dis + 1;
        if(dis > d[u]) continue;
        for(LL v = 0; v < m; v++){
            if(mp[u][v] && dis + 1 < d[v]){
                d[v] = dis + 1;
                que.push(PII(d[v], v));
            }
        }
    }
    return INF;
}
int main()
{
//    Open();
    LL T;scanf("%I64d", &T);
    LL cas = 1;
    while(T--)
    {
        scanf("%I64d%I64d", &n, &m);
        memset(mp, 0, sizeof(mp));
        for(LL i = 0 ; i < n; i++)
        {
            LL x, y;
```

```
            scanf("%I64d%I64d", &x, &y);
            pa[i] = Point(x, y);
        }
        LL convn = ConvexHull(pa, n, pb);
        for(LL i = 0; i < convn; i++){
            L[i] = Line(pb[i], pb[i+1]-pb[i]);
        }
        sort(L, L + convn);
        bool flag = false;
        for(LL i = 0 ; i < m; i++)
        {
            LL x, y;
            scanf("%I64d%I64d", &x, &y);
            pa[i] = Point(x, y);
            if(flag == false && pa[i] == pb[0]) flag = true;
        }
        printf("Case #%I64d: ", cas++);
        if(convn == 1 && flag){
            printf("1\n");
            continue;
        }
        flag = false;
        for(LL i = 0; i < m; i++)
            for(LL j = 0; j < m; j++)
            {
                if(i == j) continue;
                if(n == 1 && flag == false && OnSegment(pb[0], pa[i],
pa[j])) flag = true;
                if(n == 2 && flag == false
                && (OnSegment(pb[0], pa[i], pa[j]) || pb[0] == pa[i] ||
pb[0] == pa[j])
                && (OnSegment(pb[1], pa[i], pa[j]) || pb[1] == pa[i] ||
pb[1] == pa[j]))
                    flag = true;
                Vector v = pa[j] - pa[i];
                LL idx = upper_bound(L, L+convn, Line(Point(0, 0), v))
- L - 1;
                idx = (idx + convn) % convn;
                if(Cross(v, L[idx].p - pa[i]) >= 0 && Cross(v, L[idx].p
+ L[idx].v - pa[i]) >= 0){
                    if(n <= 2 && v == L[idx].v && (OnSegment(L[idx].p,
pa[i], pa[j]) || L[idx].p == pa[i] || L[idx].p == pa[j]));
                    else
                        mp[i][j] = 1;
                }
            }
        if(flag){
            printf("2\n");
            continue;
        }
        LL ans = INF;
        for(LL i = 0; i < m; i++)
        {
            ans = min(ans, dij(i));
        }
        if(ans == INF) ans = -1;
        printf("%I64d\n", ans);
    }
    return 0;
}
```

凸包上选择 K 个点使得面积最大_枚举起点

```
/*
* HDU_5473 给出n个点，选择k个点使得形成凸包的面积最大
* 设m为n个点凸包上点的个数。
* 1. 当k >= m时，答案毫无疑问就是凸包面积
* 2. 当k < m 时，选择出来的k个点一定是凸包上的点。那么我们枚举n个点，使得凸包在这里断开形成链；
*    dp[i][j]表示前i个点中选择了j个，且i+1 ~ m中所有点都被选中的最大面积。转移只需要枚举前一个选中的点i'即可。
*    这一步复杂度为O(n^3),再加上一开始枚举的起点，那么复杂度为O(n^4).但是注意到枚举的起点只要在最优解点集中的话，
*    就能算出正确答案，这里并不需要枚举所有点。而只需要随机枚举c*(n/k)个点，这样答案错误的概率为(1-k/n)^(c*(n/k))
*    大概是e^(-c)，这个概率是非常小的。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <time.h>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    LL x,y;
    Point(LL x = 0, LL y = 0):x(x),y(y){}
}pa[N],pb[N];

typedef Point Vector;
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return (a.x-b.x) == 0
&& (a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}
```

```
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return
acos(Dot(A,B)/Length(A)/Length(B));}
LL Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
A.x/L);}//需要确保A不是0向量
LL ConvexHull(Point* p, LL n, Point* ch)
{
    sort(p, p+n);
    LL m = 0;
    for(LL i=0;i<n;i++){
        while(m > 1 && Cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0)
m--;
        ch[m++] = p[i];
    }
    LL k=m;
    for(LL i=n-2;i>=0;i--){
        while(m>k && Cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m-
-;
        ch[m++] = p[i];
    }
    if(n>1) m--;
    return m;
}
LL PolygonArea(Point* p, LL n)
{
    LL area = 0;
    for(LL i=1;i<n-1;i++) area += Cross(p[i]-p[0], p[i+1]-p[0]);
    return abs(area);
}
bool vis[111];
LL dp[111][111];
int main()
{
//   Open();
    srand((int)time(NULL));
    LL T;scanf("%I64d", &T);
    LL cas = 1;
    while(T--)
    {
        memset(vis, 0, sizeof(vis));
        LL n, K;
        scanf("%I64d%I64d", &n, &K);
        for(LL i = 0; i < n; i++)
        {
            LL x, y;
            scanf("%I64d%I64d", &x, &y);
            pa[i] = Point(x, y);
        }
        LL m = ConvexHull(pa, n, pb);
        LL res = 0;
        if(K >= m) res = PolygonArea(pb, m);
        else{
            LL TIMES = min(m, 10LL);
            LL allarea = PolygonArea(pb, m);
            while(TIMES--){
                LL st = rand() % m;
                while(vis[st]) st = rand()%m;
```

```
                vis[st] = 1;
                memset(dp, 0, sizeof(dp));
                dp[0][0] = allarea;

                for(LL i = 1; i <= m; i++)
                {
                    LL curidx = st + i;
                    if(curidx >= m) curidx -= m;
                    LL tmparea = 0;
                    for(LL k = i-1; k >= 0; k--)
                    {
                        LL nxtidx = st + k;
                        if(nxtidx >= m) nxtidx -= m;
                        LL preidx = nxtidx + 1;
                        if(preidx >= m) preidx -= m;
                        tmparea += abs(Cross(pb[preidx] - pb[curidx],
pb[nxtidx] - pb[curidx]));
                        for(LL j = K; j > 0; j--)
                        {
                            dp[i][j] = max(dp[i][j], dp[k][j-1] -
tmparea);
                        }
                    }
                }
                res = max(res, dp[m][K]);
            }
        }
        printf("Case #%I64d: %I64d\n", cas++, res);
    }
    return 0;
}
```

半平面交

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 111
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
const double eps = 1e-8;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
```

```
}
struct Point
{
    double x, y;
    Point(){}
    Point(double x, double y):x(x), y(y){}
}pn[N*N];
typedef Point Vector;
LL dcmp(double x) {
    if(x < -eps) return -1;
    return x > eps;
}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}

double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
A.x/L);}//需要确保A不是0向量
struct line
{
    Point P;
    Vector v;
    double ang;
    line(){}
    line(Point P, Vector v):P(P), v(v){ang = atan2(v.y, v.x);}
    line(double a, double b, double c)
    {
        v = Vector(b, -a);
        if(b != 0) P = Point(0, -c/b);
        else P = Point(-c/a, 0);
        Vector nor = Normal(v);
        Point tmp = nor + P;
        if(dcmp(a * tmp.x + b * tmp.y + c) > 0){
            v = v*-1;
        }
        ang = atan2(v.y, v.x);
    }
    bool operator<(const line &o)const{
        return ang < o.ang;
    }
}L[111];
bool OnLeft(line L, Point P)
{
    return Cross(L.v, P - L.P) > 0;
}
Point GetIntersection(line a, line b)
{
    Vector u = a.P - b.P;
    double t = Cross(b.v, u) / Cross(a.v, b.v);
    return a.P + a.v*t;
}
LL HalfPlaneIntersection(line* L, LL n, Point* poly)
{
```

```
    sort(L, L+n);
    LL first, last;
    Point *p = new Point[n];
    line *q = new line[n];
    q[first = last = 0] = L[0];
    for(LL i = 1; i < n ;i++)
    {
        while(first < last && !OnLeft(L[i], p[last - 1])) last--;
        while(first < last && !OnLeft(L[i], p[first])) first++;
        q[++last] = L[i];
        if(fabs(Cross(q[last].v, q[last-1].v)) < eps){
            last--;
            if(OnLeft(q[last], L[i].P)) q[last] = L[i];
        }
        if(first < last) p[last - 1] = GetIntersection(q[last - 1],
q[last]);
    }
    while(first < last && !OnLeft(q[first], p[last - 1])) last -- ;
    if(last - first <= 1) return 0;
    p[last] = GetIntersection(q[last], q[first]);
    LL m = 0;
    for(LL i = first; i <= last; i++) poly[m++] = p[i];
    return m;
}
double PolygonArea(Point* p, LL n)
{
    double res = 0;
    for(LL i = 1; i < n-1; i++)
        res += Cross(p[i] - p[0], p[i+1] - p[0]);
    return res/2;
}
LL X[111], Y[111];
LL X2[111], Y2[111];
int main()
{
    Open();
    LL T;scanf("%I64d", &T);
    LL cas = 1;
    while(T--)
    {
        LL n, m;
        scanf("%I64d%I64d", &n, &m);
        memset(X, 0, sizeof(X));
        memset(Y, 0, sizeof(Y));
        memset(X2, 0, sizeof(X2));
        memset(Y2, 0, sizeof(Y2));
        for(LL i = 1; i <= n; i++){
            for(LL j = 1 ;j <= m; j++){
                LL x, y;
                scanf("%I64d%I64d", &x, &y);
                X2[i] += x * x;
                Y2[i] += y * y;
                X[i] += x;
                Y[i] += y;
            }
        }
        printf("Case #%I64d:", cas++);
        for(LL i = 1; i <= n; i++){
            L[0] = line(Point(0, 0), Vector(4095, 0));
            L[1] = line(Point(0, 0), Vector(0, -4095));
            L[2] = line(Point(4095, 0), Vector(0, 4095));
```

```
            L[3] = line(Point(0, 4095), Vector(-4095, 0));
            LL tail = 4;
            for(LL j = 1; j <= n; j++){
                if(i == j) continue;
                LL a = 2 * (X[j] - X[i]);
                LL b = 2 * (Y[j] - Y[i]);
                LL c = X2[i] - X2[j] + Y2[i] - Y2[j];
                L[tail++] = line(a, b, c);
            }
            LL cnt = HalfPlaneIntersection(L, tail, pn);
            printf(" %.0f", PolygonArea(pn, cnt) + eps);
        }
        cout<<endl;
    }
    return 0;
}
```

求凸包的面积期望

```
/*
* 凸包面积为多边形中各个矢量三角形的面积之和
* 而面积之和可转化为三角形的两向量的叉积，也就是说凸包上的每一条边对总面积都有贡献，那么只需要枚举点集中的所有边，
* 再计算出这条边在凸包中的概率，即可得出该边对答案的贡献。
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF = 0x3f3f3f3f;
const LL mod = 1000000007;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
LL x[N], y[N];
int main()
{
//    Open();
    LL T;scanf("%I64d", &T);
    while(T--)
```

```
    {
        LL n;
        scanf("%I64d", &n);
        for(LL i = 0; i < n; i++) scanf("%I64d%I64d", &x[i], &y[i]);
        LL px = 0, py = 0, mul = 1, pxx = 0, pyy = 0;
        for(LL i = n - 2; i >= 1; i--){
            mul = mul * 2 % mod;
            px = (px + x[i]) % mod;
            py = (py + y[i]) % mod;
            pxx = (pxx + x[i] * mul % mod) % mod;
            pyy = (pyy + y[i] * mul % mod) % mod;
        }

        LL tail = n-1;
        LL head = 1;
        LL ans = 0;
        for(LL i = 0; i < n; i++)
        {
            ans = (ans + x[i] * (pyy - py) % mod) %mod;
            ans = (ans - y[i] * (pxx - px) % mod) %mod;
            pyy = (pyy - y[head] * mul % mod) %mod;
            py = (py - y[head]) % mod;
            pyy = pyy * 2 %mod + y[tail]*2;
            py = (py + y[tail]);

            pxx = (pxx - x[head] * mul % mod) %mod;
            px = (px - x[head]) % mod;
            pxx = pxx * 2 %mod + x[tail]*2;
            px = (px + x[tail]);
            head++, tail++;
            head %= n, tail %= n;
        }
        printf("%I64d\n", (ans + mod)%mod);
    }
    return 0;
}
```

智商题-极角排序

```
/*
* 首先给出一个n个点的凸包，再给出K个在凸包内的点，要求在凸包上选择m个点，使得这个m个点构成的凸包包含这K个点，要满足如下条件：
* 1. m<=2*K
* 2. m个点的凸包的周长小于原凸包
*
* 分析：这个题也就是个智商题，首先我们可以发现的是，其实第二个条件是没卵用的，在原凸包上选择有限个点构成的图形周长会更长？
* 反正我是不相信的。所以也只是需要满足第一个条件。于是思路走向应该是这样：
* （恩...2*K？好诡异，是不是说明解法和给出的点或者线段有关系？）这算是一个切入点吧。当然给出的K个点先求一个凸包。只要包含凸包
* 那么一定是满足条件的。假设K个点的图形构成的凸包有p个点，那么p <= K，如果也就是说，只要m <= 2*p <= 2*K即可！于是感觉像是对于
* 每个点在原凸包上面找两个点使得这个点一定在结果图形内？的确是这样！在p凸包内任意找一个点O，假设pi为p凸包上的点，Ai为Opi射线
* 与原凸包上线段的交点。那么只需要将Bi的线段上的两个点包含在结果中，结果一定是符合条件的！于是，问题迎刃而解。
*
```

```
* 那么这里已经将问题转化为：有一组同源(O)射线，判断一个凸包上的线段与同源射线中的线有无交点即可。
* 这个问题可以使用极角排序解决。(极角排序应该是将平面上的属性转移为线性的有效工具！)
*
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <assert.h>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 200010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    double x,y;
    int id;
    Point(double x = 0, double y = 0):x(x),y(y){}
}pa[N],pk[N], ch[N];
const double eps = 1e-10;
const double PI = acos(-1.0);
typedef Point Vector;
int dcmp(double x){if(fabs(x) < eps) return 0;else return x<0?-1:1;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}//返回A向量的极角
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return
acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}
double torad(double ang){return ang / 180 * PI;}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
```

```
A.x/L);}//需要确保A不是0向量
//非规范相交，端点上视为在线段上
bool OnSegment(Point p, Point a, Point b) { return dcmp(Length(p - a)
+ Length(p - b) - Length(a - b)) == 0; }//精度也很高的!

int Convex(Point *p, int n, Point *ch)
{
    sort(p, p+n);
    int m = 0;
    for(int i = 0; i < n; i++) {
        while(m > 1 && Cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0)
m--;
        ch[m++] = p[i];
    }
    int k = m;
    for(int i = n-1; i >= 0; i --) {
        while(m > k && Cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0)
m--;
        ch[m++] = p[i];
    }
    if(n > 1) m--;
    return m;
}
bool isPointInConvexPolygon(Point p, Point* poly, int n)
{
    for(int i=0;i<n;i++)
        if(dcmp(Cross(poly[(i+1)%n] - poly[i], p - poly[i])) <= 0)
return 0;
    return 1;
}
double sta[N];
int ans[N*2];
int main()
{
//    Open();
    int T;scanf("%d", &T);
    while(T--)
    {
        int n;
        scanf("%d", &n);
        for(int i = 0; i < n; i++)
        {
            double x, y;scanf("%lf%lf", &x, &y); pa[i] = Point(x, y);
            pa[i].id = i;
        }
        int m;
        scanf("%d", &m);
        for(int i = 0; i < m; i++)
        {
            double x, y;scanf("%lf%lf", &x, &y); pk[i] = Point(x, y);
        }
        m = Convex(pk, m, ch);
        double ang = Angle(ch[2] - ch[1], ch[0] - ch[1]) / 2;
        Point O = ch[1] + Rotata(ch[2] - ch[1], ang) * 0.5;
//        assert(isPointInConvexPolygon(O, ch, m));
        int tail = 0;
        for(int i = 0; i < m; i++)
        {
            sta[tail++] = angle(ch[i] - O);
        }
        int anst = 0;
```

```
        sort(sta, sta+tail);
        for(int i = 0; i < n; i++)
        {
            int nxt = (i+1)%n;
            double ast = angle(pa[i] - O), aed = angle(pa[nxt] - O);
            if(dcmp(aed - ast) > 0)
            {
                int idx = lower_bound(sta, sta+tail, ast-eps) - sta;
                if(dcmp(aed - sta[idx]) >= 0){
                    ans[anst++] = pa[i].id;
                    ans[anst++] = pa[nxt].id;
                }
            }else{
                int idx1 = lower_bound(sta, sta+tail, ast-eps) - sta;
                int idx2 = upper_bound(sta, sta+tail, aed+eps) - sta;
                if(idx1 != tail || idx2 > 0){
                    ans[anst++] = pa[i].id;
                    ans[anst++] = pa[nxt].id;
                }
            }
        }
        sort(ans, ans+anst);
        anst = unique(ans, ans+anst) - ans;
        if(anst < 3){
            printf("No\n");
        }else{
            printf("Yes\n%d\n", anst);
            for(int i = 0; i < anst; i++)
                printf("%d%c", ans[i]+1, " \n"[i == anst-1]);
        }
    }
    return 0;
}
```

多边形转化为最短路-三角形化点

```
/*
* 给出n棵树，每棵树有一个种类Ai，再给出m个点，需要在这m个点中选出若干个点，
* 使得这个多边形包含所有类型的树
*
* 所以还是智商题，Orz。。。
* 首先这个多边形一定是凸的，如果是凹的，一定有对应的凸多边形更优。
* 那么既然是凸多边形，这个多边形一定能分解为若干三角形链！。。于是就类似路径，所以
建边，
* 然后跑一边最短路(优先队列BFS)差不多就行了
*
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 333
//#define lson x<<1
```

```
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<double,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int id[41][41][41];//
int s[41*41*41];//
double idw[41*41*41];//
struct node{
    int v, s;
    double w;
    node(){}
    node(int _v, int _s, double _w){
        v = _v, s = _s, w = _w;
    }
    bool operator<(const node& o)const{
        return w > o.w;
    }
};
vector<node> G[41*41*41];//
int kind[N];//
int n, m, k;//
struct Point
{
    double x, y;
    Point(double x = 0, double y = 0):x(x), y(y){}
}pa[N], pb[N];//
const double eps = 1e-10;
typedef Point Vector;
int dcmp(double x){if(x < -eps) return -1;return x > eps;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}//返回A向量的极角
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
bool isPointInTriangle(Point P, Point A, Point B, Point C)
{
    if(dcmp(Cross(B - A, P - A)) == dcmp(Cross(C - B, P - B))
       && dcmp(Cross(C - B, P - B)) == dcmp(Cross(A - C, P - C)))
return true;
    return false;
}
int getid(int a, int b, int c)
{
```

```
    if(a > b) swap(a, b);
    if(a > c) swap(a, c);
    return id[a][min(b, c)][max(b, c)];
}
int limit;//
double dp[41*41*41][(1 << 6)+10];//
double BFS(int s)
{
    priority_queue<node> que;
    que.push(node(0, 0, 0));
    dp[0][0] = 0;
    while(!que.empty())
    {
        node cur = que.top();que.pop();
        int u = cur.v, s = cur.s;
        double w = cur.w;
        if(s == limit-1) return w;
        for(int i = 0; i < G[u].size(); i++)
        {
            int nxts = G[u][i].s | s;
            int v = G[u][i].v;
            double nxtw = G[u][i].w;
            if(dp[v][nxts] > nxtw + w + eps){
                dp[v][nxts] = nxtw + w;
                que.push(node(v, nxts, dp[v][nxts]));
            }
        }
    }
    return -1;
}
double getlen(Point A, Point B, Point C){
    return Length(A-B)+Length(A-C)+Length(B-C);
}
int main()
{
//    Open();
    while(~scanf("%d%d%d", &n, &m, &k)){
        for(int i = 0; i < n; i++)
        {
            double x, y;scanf("%lf%lf", &x, &y);pa[i] = Point(x, y);
        }
        for(int i = 0; i < n; i++){
            scanf("%d", &kind[i]);kind[i]--;
        }
        for(int i = 0; i < m; i++)
        {
            double x, y;scanf("%lf%lf", &x, &y);pb[i] = Point(x, y);
        }
        limit = 1 << k;
        int tot = 0;
        for(int A = 0; A < m; A++)
            for(int B = A+1; B < m; B++)
                for(int C = B+1; C < m; C++){
                    id[A][B][C] = ++tot;
                    s[tot] = 0;
                    for(int i = 0; i < n; i++)
                    {
                        if(isPointInTriangle(pa[i], pb[A], pb[B],
pb[C])){
                            s[tot] |= (1 << kind[i]);
                        }
```

```
                }
                idw[tot] = getlen(pb[A], pb[B], pb[C]);
            }
        for(int i = 0; i <= tot; i++) {G[i].clear(); if(i)
G[0].push_back(node(i, s[i], idw[i]));}
        for(int A = 0; A < m; A++)
            for(int B = A+1; B < m; B++)
                for(int C = B+1; C < m; C++)
                    for(int i = 0; i < m; i++)
                    {
                        if(i == A || i == B || i == C) continue;
                        int curid = id[A][B][C];
                        double AB = Length(pb[A] - pb[B]);
                        double BC = Length(pb[C] - pb[B]);
                        double AC = Length(pb[A] - pb[C]);
                        double Ai = Length(pb[A] - pb[i]);
                        double Bi = Length(pb[B] - pb[i]);
                        double Ci = Length(pb[C] - pb[i]);
                        double oriw = AB+BC+AC, neww;
                        int t1 = dcmp(Cross(pb[A] - pb[C], pb[i] -
pb[C]));
                        int t2 = dcmp(Cross(pb[A] - pb[C], pb[B] -
pb[C]));
                        if(t1 * t2 < 0)
                        {
                            int v = getid(i, A, C);
                            G[curid].push_back(node(v, s[v], Ai + Ci -
AC));
                        }
                        t1 = dcmp(Cross(pb[B] - pb[C], pb[i] - pb[C]));
                        t2 = dcmp(Cross(pb[B] - pb[C], pb[A] - pb[C]));
                        if(t1 * t2 < 0)
                        {
                            int v = getid(i, B, C);
                            G[curid].push_back(node(v, s[v], Bi + Ci -
BC));
                        }
                        t1 = dcmp(Cross(pb[A] - pb[B], pb[i] - pb[B]));
                        t2 = dcmp(Cross(pb[A] - pb[B], pb[C] - pb[B]));
                        if(t1 * t2 < 0)
                        {
                            int v = getid(i, A, B);
                            G[curid].push_back(node(v, s[v], Ai + Bi -
AB));
                        }
                    }
        for(int i = 0; i <= tot; i++)
            for(int j = 0; j < limit; j++) dp[i][j] = INF;
        double ans = BFS(0);
        if(dcmp(ans) < 0) printf("Impossible\n");
        else printf("%.12f\n", ans);
    }
    return 0;
}
```

旋转多边形转化为旋转单点重心

```
#include <iostream>
#include <cstdio>
```

```
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <assert.h>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
typedef pair<double, double> PDD;
const LL INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("F:/in.txt","r",stdin);
//        freopen("F:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    double x,y;
    Point(double x = 0, double y = 0):x(x),y(y){}
    void read()
    {
        scanf("%lf%lf", &x, &y);
    }
}pa[N],pb[N];
const double eps = 1e-10;
const double PI = acos(-1.0);
typedef Point Vector;
int dcmp(double x){if(fabs(x) < eps) return 0;else return x<0?-1:1;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}//返回A向量的极角
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B)
{
    double d=Dot(A,B)/Length(A)/Length(B);
    if(dcmp(d-1)==0) return 0;
    if(dcmp(d+1)==0) return PI;
    return acos(d);
}//A到B的逆时针转的角
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}//A逆时针转ang弧度
double torad(double ang){return ang / 180 * PI;}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
```

```
A.x/L);}//需要确保A不是0向量，左转90度
//非规范相交，端点上视为在线段上
bool OnSegment(Point p, Point a, Point b) { return dcmp(Length(p - a)
+ Length(p - b) - Length(a - b)) == 0; }//精度也很高的!

Point MassCenter(Point a[] , int n){
    Point ans = Point(0,0);
    double area = 0;
    a[n] = a[0];
    for(int i=0;i<n;i++) ans = ans+(a[i] + a[i+1])*Cross(a[i] ,
a[i+1]), area += Cross(a[i], a[i+1]);
    if(dcmp(area) == 0) return Point(0, 0);
    return ans / (area * 3.0);
}
Point O, S;
double solve(Point st, double len, double UP)
{
    double oldl = Length(S - st);
    double lb = 0, ub = UP, ans = 0;
    int cnt = 50;
    while(cnt--)
    {
        double mid = (lb + ub) * 0.5;
        double newl = Length(S - (Rotata(st - O, -mid) + O));
        if(dcmp(newl - oldl - len) >= 0) ub = mid;
        else lb = mid;
    }
    return (ub+lb)*0.5;
}
double solve2(double a, double b, double c)
{
    double delta = b*b - 4.0 * a * c;
    return (-b + sqrt(delta)) / (2.0 * a);
}
Point getP(Point p, Vector v, Point C, double r)
{
    double a = v.x, b = p.x - C.x, c = v.y, d = p.y - C.y;
    double e = a*a + c*c, f = 2*(a*b+c*d), g = b*b+d*d-r*r;
    double delta = f*f - 4*e*g;
    return p + v * (-f + sqrt(delta)) / (2*e);
}
struct info{
    double len, ang;
    Point cur;
}preinfo[N];
int getid(int UP, double l)
{
    int lb = -1, ub = UP;
    while(lb + 1 < ub)
    {
        int mid = (ub + lb) >> 1;
        if(dcmp(l - preinfo[mid].len) == 0) return mid;
        if(mid+1 < UP &&dcmp(l - preinfo[mid].len)*dcmp(l -
preinfo[mid+1].len) < 0) return mid;
        if(dcmp(l - preinfo[mid].len) > 0) lb = mid;
        else ub = mid;
    }
    return ub + lb >> 1;
}
int main()
{
```

```
    Open();
//    cout<<cos(111)<<endl;
    int T;scanf("%d", &T);
    int cas = 1;
    while(T--)
    {
        printf("Case #%d:\n", cas++);
        int n;
        scanf("%d", &n);
        for(int i = 0; i < n; i++)
            pa[i].read();
        pa[n] = pa[0];
        double sx, sy;
        scanf("%lf%lf", &sx, &sy);
        S = Point(sx, sy);
        O = MassCenter(pa, n);
        double a = Length(S - O);
        double alllen = 0;
        for(int i = 0; i < n; i++)
            alllen += Length(pa[i] - pa[(i+1)%n]);
        int st;
        double ang = -INF;
        for(int i = 0; i < n; i++)
        {
            if(dcmp(angle(pa[i] - S) - ang) > 0){
                ang = angle(pa[i] - S);
                st = i;
            }
        }
        Point las = S, cur;
        preinfo[0].cur = S;
        for(int i = st, k = 0; k < n; k++)
        {
            int id = (i + k) % n;
            cur = getP(pa[id], pa[id+1] - pa[id], O, a);
            preinfo[k+1].len = preinfo[k].len + Length(cur - pa[id]) -
Length(las - pa[id]);
            preinfo[k+1].ang = preinfo[k].ang + Angle(las - O, cur -
O);
            preinfo[k+1].cur = cur;
            las = cur;
        }
        int m;
        scanf("%d", &m);
        while(m--)
        {
            double l;
            double ans = 0;
            scanf("%lf", &l);
            LL num = floor(l / alllen) + eps;
            l -= num * alllen;
            ans += 360LL * num;
            double tmpans = 0;
            if(dcmp(l)>0)
            {
                int idx = getid(n+1, l);
//                int idx = lower_bound(preinfo, preinfo+tail, PDD(l, -
INF)) - preinfo - 1;
                tmpans = preinfo[idx].ang;
                cur = preinfo[idx].cur;
                l -= preinfo[idx].len;
```

```
                if(dcmp(l) > 0){
                    idx = (st+idx)%n;
                    double OP = Length(O - pa[idx]);
                    l += Length(pa[idx]-cur);
                    ang = (OP*OP + a*a - l*l)/(2.0*OP*a);
                    ang = max(-1.0, min(ang, 1.0));
                    ang = acos(ang);
                    ang -= abs(Angle(pa[idx] - O, cur - O));
                    tmpans += ang;
                }
            }
            printf("%.3f\n", ans + (tmpans) / PI * 180);
        }
    }
    return 0;
}
```

圆圆，圆多边形的交界周长

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <assert.h>
//#include <unordered_map>
#define N 3030
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    double x,y;
    Point(double x = 0, double y = 0):x(x),y(y){}
}pa[N],pb[N];

typedef Point Vector;
const double eps = 1e-8;
const double PI = acos(-1.0);
int dcmp(double x){if(fabs(x) < eps) return 0;else return x<0?-1:1;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
```

```
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return
acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}
double torad(double ang){return ang / 180 * PI;}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
A.x/L);}//需要确保A不是0向量
bool OnSegment(Point p, Point a, Point b) { return dcmp(Length(p - a)
+ Length(p - b) - Length(a - b)) == 0; }//精度也很高的!

struct Poly
{
    Point p[N];
    int num;
    Poly(){}
    Poly(int _n){num = _n;}
    double Len()
    {
        double res = 0;
        for(int i = 0; i < num; i++)
        {
            int nxt = i + 1;
            if(nxt == num)nxt = 0;
            res += Length(p[nxt] - p[i]);
        }
        return res;
    }
};
vector<Poly> allP;
struct Circle
{
    Point c;
    double r;
    Circle(){}
    Circle(Point c, double r):c(c),r(r){}
    Point point(double a)
    {
        return Point(c.x + cos(a)*r, c.y+sin(a)*r);
    }
    double Len()
    {
        return 2 * PI * r;
    }
}Cir[N];
struct Line
{
    Point p;
    Vector v;
    double ang;
    Line(){}
    Line(Point p, Vector v):p(p), v(v){ang = atan2(v.y, v.x);}
```

```
    Line(double a, double b, double c)
    {
        v = Vector(b, -a);
        if(b != 0) p = Point(0, -c/b);
        else p = Point(-c/a, 0);
        Vector nor = Normal(v);
        Point tmp = nor + p;
        if(dcmp(a * tmp.x + b * tmp.y + c) > 0){
            v = v*-1;
        }
        ang = atan2(v.y, v.x);
    }
    Point point(double t){return p + v*t;}//只能在点斜式中用
    bool operator<(const Line &o)const{
        return ang < o.ang;
    }
}L[N];
double DistanceToLine(Point P, Point A, Point B)
{
    Vector v1 = B-A,v2 = P-A;
    return fabs(Cross(v1,v2)) / Length(v1);
}
//直线与圆相交,直线必须是点斜式
int getLineCircleIntersection(Line L, Circle C, vector<Point >& sol)
{
    double a = L.v.x, b = L.p.x - C.c.x, c = L.v.y, d = L.p.y -
C.c.y;
    double e = a*a + c*c, f = 2*(a * b + c * d), g = b*b+d*d-C.r*C.r;
    double delta = f*f - 4*e*g;
    double dist = DistanceToLine(C.c, L.p, L.p + L.v);
    double t1,t2;
    if(dcmp(dist - C.r) > 0) return 0;
    if(dcmp(dist - C.r) == 0) {
        t1 = t2 = -f / (2*e);
        if(dcmp(t1)>= 0 && dcmp(1-t1)>=0)sol.push_back(L.point(t1));
        return 1;
    }
    t1 = (-f - sqrt(delta)) / (2*e);
    if(dcmp(t1)>= 0 && dcmp(1-t1)>=0) sol.push_back(L.point(t1));
    t2 = (-f + sqrt(delta)) / (2*e);
    if(dcmp(t2)>= 0 && dcmp(1-t2)>=0) sol.push_back(L.point(t2));
    return 2;
}
#define D(x) ((x)*(x))
struct CPIArea
{
    Circle cir;
    double Scir;
    Point p[N];
    int tail;
    CPIArea() { tail=0; }
    CPIArea(Circle cir):cir(cir) { Scir = PI*cir.r*cir.r; tail=0; }
    //tp[]是多边形的点集，n是点的个数。tp[]必须满足点是按顺时针或者逆时针排序的
    double solvePoly(Point tp[],int n)
    {
        tail = 0;
        for(int i=0; i<n; i++)
        {
            p[tail++]=tp[i];//p[]是囊括了圆和多边形交点的点集，也是按顺时针或
者逆时针排序的
```

```
            Line line = Line(tp[i],tp[(i+1)%n] - tp[i]);
            vector<Point > sol;
            sol.clear();
            getLineCircleIntersection(line, cir, sol);

            for(int j=0; j<sol.size(); j++)
            {
                p[tail++]=sol[j];
            }
        }
        double polyres = 0;
        for(int i=0; i<tail; i++)
        {
            Point O = cir.c;
            Point mid = (p[i]+p[(i+1)%tail])/2;
            if(dcmp(Length(mid - O) - cir.r) < 0){
                polyres += Length(p[i] - p[(i+1)%tail]);
            }
        }
        return polyres;
    }
    double solveCircle(Circle tc)
    {
        double d = Length(tc.c - cir.c), r1 = tc.r, r2 = cir.r;
        if(dcmp(d - r1 - r2) >= 0) return 0;
        if(tc.c == cir.c){
            if(dcmp(r2 - r1) >= 0) return 2 * PI * tc.r;
            return 0;
        }
        if(dcmp(r2 - d - r1) >= 0) return 2 * PI * r1;
        if(dcmp(r1 - d - r1) >= 0) return 0;
        double w = (D(r1)+D(d)-D(r2)) / (2.0 * r1 * d);
        w = min(w, 1.0);
        double ang = acos(w);
        return 2 * ang * r1;
    }
};
int ctail;
double cx, cy;
double calc(double r)
{
    CPIArea fuck(Circle(Point(cx, cy), r));
    double res = 0;
    for(int i = 0; i < ctail; i++)
    {
        res += fuck.solveCircle(Cir[i]);
    }
    for(int i = 0; i < allP.size(); i++)
    {
        res += fuck.solvePoly(allP[i].p, allP[i].num);
    }
    return res;
}
int main()
{
    int T;scanf("%d", &T);
    int cas = 1;
    while(T--)
    {
        int n;
        double len;
```

```
        scanf("%d%lf", &n, &len);
        allP.clear();
        double alllen = 0;
        char ty[3];
        ctail = 0;
        for(int i = 0; i < n; i++){
            scanf("%s", ty);
            if(ty[0] == 'C'){
                double x, y, r;
                scanf("%lf%lf%lf", &x, &y, &r);
                Cir[ctail++] = Circle(Point(x, y), r);
                alllen += Cir[ctail-1].Len();
            }else{
                int np;
                scanf("%d", &np);
                int id = allP.size();
                allP.push_back(Poly(np));
                for(int j = 0; j < np; j++)
                {
                    double x, y;
                    scanf("%lf%lf", &x, &y);
                    allP[id].p[j] = Point(x, y);
                }
                alllen += allP[id].Len();
            }
        }
        printf("Case #%d: ", cas++);
        scanf("%lf%lf", &cx, &cy);
        if(dcmp(len - alllen) == 0) {
            printf("inestimable\n");
            continue;
        }
        double lb = 0, ub = 100000;
        double ans = -1;
        int cnt = 100;//浮点二分一定要用次数控制！100次，
        while(cnt--)
        {
            double mid = (lb + ub) / 2;
            double res = calc( mid);
            if(dcmp(res - len) > 0) ub = mid;
            else if(dcmp(res - len) == 0) lb = ans = mid;//这里答案如果为
一个区间的话，这里需要特别注意
            else lb = mid;
        }
        if(ans == -1){
            printf("impossible\n");
            continue;
        }
        printf("%.2f\n", ans);
    }
    return 0;
}
```

线段树_扫描线_求面积

```
#include <iostream>
#include <cstdio>
#include <stack>
```

```cpp
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 10100
#define lson x<<1
#define rson x<<1|1
#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
double sty[N];
struct Node{
    LL l, r;
    LL st, ed, sum;
    LL lazy;
    Node(){}
    Node(LL _l, LL _r, LL _s ,LL _e){
        l = _l, r = _r, st = _s, ed = _e, lazy = sum = 0;
    }
    LL len()
    {
        return ed - st;
    }
}lt[N*6];
void push_up(LL x)
{
    if(lt[x].lazy > 0) lt[x].sum = lt[x].len();
    else if(lt[x].l == lt[x].r) lt[x].sum = 0;
    else lt[x].sum = lt[lson].sum + lt[rson].sum;
}
void build(LL l, LL r, LL x)
{
    lt[x] = Node(l, r, sty[l], sty[r]);
    if(l == r) return ;
    if(mid != r)build(l, mid, lson);
    if(mid != l)build(mid, r, rson);
}
void update(LL l, LL r, LL x, LL val)
{
    if(lt[x].l >= l && lt[x].r <= r)
    {
        lt[x].lazy += val;
        push_up(x);
        return;
    }
    if(r <= mid) update(l, r, lson, val);
    else if(l >= mid) update(l, r, rson, val);
    else update(l, mid, lson, val), update(mid, r, rson, val);
    push_up(x);
```

```
}
struct Rec{
    LL x, y1, y2;
    LL ly1, ly2, val;
    Rec(){}
    Rec(LL _x, LL _y1, LL _y2, LL v){
        x = _x, y1 = _y1, y2 = _y2, val = v;
    }
    bool operator<(const Rec& o)const{
        return x < o.x;
    }
}r[N];
int main()
{
//    Open();
    LL cas = 1;
    LL x1, y1, x2, y2;
    while(~scanf("%I64d%I64d%I64d%I64d", &x1, &y1, &x2, &y2), x1 != -
1)
    {
        LL n = 0;
        LL tx = 0, ty = 0;
        r[n*2] = Rec(x1, y1, y2, 1);
        r[n*2+1] = Rec(x2, y1, y2, -1);
        sty[ty++] = y1, sty[ty++] = y2;
        n++;
        while(~scanf("%I64d%I64d%I64d%I64d", &x1, &y1, &x2, &y2),
x1 != -1)
        {
            r[n*2] = Rec(x1, y1, y2, 1);
            r[n*2+1] = Rec(x2, y1, y2, -1);
            sty[ty++] = y1, sty[ty++] = y2;
            n++;
        }
        sort(r, r+2*n);
        sort(sty, sty+ty);
        build(0, ty-1, 1);
        LL ans = 0;
        for(LL i = 0; i < 2*n; i++)
        {
            r[i].ly1 = lower_bound(sty, sty+ty, r[i].y1) - sty;
            r[i].ly2 = lower_bound(sty, sty+ty, r[i].y2) - sty;
        }
        update(r[0].ly1, r[0].ly2, 1, r[0].val);
        for(LL i = 1; i < 2 * n; i++)
        {
            ans += lt[1].sum * (r[i].x - r[i-1].x);
            update(r[i].ly1, r[i].ly2, 1, r[i].val);
        }
        printf("%I64d\n", ans);
    }
    return 0;
}
```

扫描线_线段树_求周长

```
#include <iostream>
#include <cstdio>
#include <stack>
```

```cpp
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 20000
#define lson x<<1
#define rson x<<1|1
#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int sty[N], stx[N];
struct Node{
    int l, r;
    int tl, tr, sum;//tl,tr表示是否顶满左(右)区间
    int lazy;
    Node(){}
    Node(int _l, int _r){
        l = _l, r = _r, tl = tr = 0, lazy = sum = 0;
    }
}lt[N*6];
void push_up(int x)
{
    if(lt[x].lazy > 0) lt[x].sum = 2, lt[x].tl = lt[x].tr = 1;
    else if(lt[x].l == lt[x].r) lt[x].sum = 0;
    else {
        lt[x].sum = lt[lson].sum + lt[rson].sum;
        lt[x].tl = lt[lson].tl, lt[x].tr = lt[rson].tr;
        if(lt[lson].tr && lt[rson].tl) lt[x].sum -= 2;
    }
}
void build(int l, int r, int x)
{
    lt[x] = Node(l, r);
    if(l == r) return ;
    if(mid != r)build(l, mid, lson);
    if(mid != l)build(mid, r, rson);
}
void update(int l, int r, int x, int val)
{
    if(lt[x].l >= l && lt[x].r <= r)
    {
        lt[x].lazy += val;
        push_up(x);
        return;
    }
    if(r <= mid) update(l, r, lson, val);
    else if(l >= mid) update(l, r, rson, val);
    else update(l, mid, lson, val), update(mid, r, rson, val);
    push_up(x);
```

```
}
struct Rec{
    int x, y1, y2;
    int ly1, ly2, val;
    Rec(){}
    Rec(int _x, int _y1, int _y2, int v){
        x = _x, y1 = _y1, y2 = _y2, val = v;
    }
    bool operator<(const Rec& o)const{
        return x < o.x;
    }
}rx[N], ry[N];
int main()
{
//    Open();
    int cas = 1;
    int n;
    while(~scanf("%d", &n))
    {
        int tx = 0, ty = 0;
        for(int i = 0; i < n; i++)
        {
            int x1, y1, x2, y2;
            scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
            rx[i*2] = Rec(x1, y1, y2, 1);
            rx[i*2+1] = Rec(x2, y1, y2, -1);
            ry[i*2] = Rec(y1, x1, x2, 1);
            ry[i*2+1] = Rec(y2, x1, x2, -1);
            sty[ty++] = y1, sty[ty++] = y2;
            stx[tx++] = x1, stx[tx++] = x2;
        }
        sort(rx, rx+2*n);
        sort(ry, ry+2*n);
        sort(sty, sty+ty);
        sort(stx, stx+tx);
        build(0, ty-1, 1);
        int ans = 0;
        for(int i = 0; i < 2*n; i++)
        {
            rx[i].ly1 = lower_bound(sty, sty+ty, rx[i].y1) - sty;
            rx[i].ly2 = lower_bound(sty, sty+ty, rx[i].y2) - sty;
            ry[i].ly1 = lower_bound(stx, stx+tx, ry[i].y1) - stx;
            ry[i].ly2 = lower_bound(stx, stx+tx, ry[i].y2) - stx;
        }
        update(rx[0].ly1, rx[0].ly2, 1, rx[0].val);
        for(int i = 1; i < 2 * n; i++)
        {
            ans += lt[1].sum * (rx[i].x - rx[i-1].x);
            update(rx[i].ly1, rx[i].ly2, 1, rx[i].val);
        }
        build(0, tx-1, 1);
        update(ry[0].ly1, ry[0].ly2, 1, ry[0].val);
        for(int i = 1; i < 2 * n; i++){
            ans += lt[1].sum * (ry[i].x - ry[i-1].x);
            update(ry[i].ly1, ry[i].ly2, 1, ry[i].val);
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

扫描线_线段树_求面积

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 1010
#define lson x<<1
#define rson x<<1|1
#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
double sty[N];
struct Node{
    int l, r;
    double st, ed, sum;
    int lazy;
    Node(){}
    Node(int _l, int _r, double _s ,double _e){
        l = _l, r = _r, st = _s, ed = _e, lazy = sum = 0;
    }
    double len()
    {
        return ed - st;
    }
}lt[N*6];
void push_up(int x)
{
    if(lt[x].lazy > 0) lt[x].sum = lt[x].len();
    else if(lt[x].l == lt[x].r) lt[x].sum = 0;
    else lt[x].sum = lt[lson].sum + lt[rson].sum;
}
void build(int l, int r, int x)
{
    lt[x] = Node(l, r, sty[l], sty[r]);
    if(l == r) return ;
    if(mid != r)build(l, mid, lson);//线段树中存储线段，于是这里l,mid表示开
始和结束，右区间不能是mid+1，不然会丢失一段区间
    if(mid != l)build(mid, r, rson);//
}
void update(int l, int r, int x, int val)
{
    if(lt[x].l >= l && lt[x].r <= r)
    {
        lt[x].lazy += val;
```

```
        push_up(x);
        return;
    }
    if(r <= mid) update(l, r, lson, val);
    else if(l >= mid) update(l, r, rson, val);
    else update(l, mid, lson, val), update(mid, r, rson, val);
    push_up(x);
}
struct Rec{
    double x, y1, y2;
    int ly1, ly2, val;
    Rec(){}
    Rec(double _x, double _y1, double _y2, int v){
        x = _x, y1 = _y1, y2 = _y2, val = v;
    }
    bool operator<(const Rec& o)const{
        return x < o.x;
    }
}r[N];
int main()
{
//    Open();
    int cas = 1;
    int n;
    while(~scanf("%d", &n), n)
    {
        int tx = 0, ty = 0;
        for(int i = 0; i < n; i++)
        {
            double x1, y1, x2, y2;
            scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
            r[i*2] = Rec(x1, y1, y2, 1);
            r[i*2+1] = Rec(x2, y1, y2, -1);
            sty[ty++] = y1, sty[ty++] = y2;
        }
        sort(r, r+2*n);
        sort(sty, sty+ty);
        build(0, ty-1, 1);
        double ans = 0;
        for(int i = 0; i < 2*n; i++)
        {
            r[i].ly1 = lower_bound(sty, sty+ty, r[i].y1) - sty;
            r[i].ly2 = lower_bound(sty, sty+ty, r[i].y2) - sty;
        }
        update(r[0].ly1, r[0].ly2, 1, r[0].val);
        for(int i = 1; i < 2 * n; i++)
        {
            ans += lt[1].sum * (r[i].x - r[i-1].x);
            update(r[i].ly1, r[i].ly2, 1, r[i].val);
        }
        printf("Test case #%d\n", cas++);
        printf("Total explored area: %.2f\n\n", ans);
    }
    return 0;
}
```

圆交扫描线

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <map>
//#include <unordered_map>
#define N 200020
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
#define ID(x, y) ((x)*m+(y))
#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f3f3f3f3fLL;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("/home/qingping/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}

#define D(x) ((x)*(x))
const double eps = 1e-8;
LL n;
struct Cir{
    LL x, y, r, ty, id;
    void read()
    {
        scanf("%lld%lld%lld", &x, &y, &r);
    }
    Cir(){}
    Cir(LL _x, LL _y, LL _r, LL _id, LL _ty = 0){
        x = _x, y = _y, r = _r, id = _id, ty = _ty;
    }
    bool operator<(const Cir& o)const{
        return x < o.x;
    }
}c[N], ac[N];
LL curx;
double delty, deltoy, cy, oy;
struct info{
    LL id, ty;
    info(){}
    info(LL _id, LL _ty){
        id = _id, ty = _ty;
    }
    double y()const{
        if(id == -1) return (ty == 1 ? INF : -INF);
        delty = sqrt((double)D(c[id].r) - D(curx - c[id].x));
        if(ty == 1) return c[id].y + delty;
        return c[id].y - delty;
    }
```

```
    bool operator<(const info& o)const{
        cy = y(), oy = o.y();
        if(abs(cy - oy) < eps) return ty < o.ty;
        return y() < o.y();
    }
};
LL ans[N];
multimap<info, LL> mp;
multimap<info, LL>::iterator it, preit, nxtit, itr[N][2];
LL nxtid, preid;
bool CinsC(LL a, LL b)
{
    double d = sqrt((double)D(c[a].x - c[b].x) + D(c[a].y - c[b].y));
    if(c[a].r + c[b].r > d + eps) return true;
    else return false;
}
void ins(LL y, LL id){
    it = mp.insert(make_pair(info(id, 1), 0));
    //it = mp.find(info(id, 1));
    nxtit = preit = it;
    LL res = 0;
    nxtit++, preit--;
    nxtid = (nxtit -> first).id;
    if(nxtid == -1 || CinsC(id, nxtid)) res = max(res, (nxtit ->
second)+1);
    else res = max(res, nxtit -> second);
    preid = (preit -> first).id;
    if(preid == -1 || CinsC(id, preid)) res = max(res, (preit ->
second)+1);
    else res = max(res, preit -> second);
    it -> second = res;
    itr[id][0] = it;
    itr[id][1] = mp.insert(make_pair(info(id, -1), res));
    ans[res]++;
}
int main()
{
//    Open();
    LL T;scanf("%lld", &T);
    while(T--)
    {
        scanf("%lld", &n);
        for(LL i = 0; i < n; i++)
        {
            c[i].read();
            ac[i*2] = Cir(c[i].x - c[i].r, c[i].y, c[i].r, i, 1);
            ac[i*2 + 1] = Cir(c[i].x + c[i].r, c[i].y, c[i].r, i, -1);
        }
        memset(ans, 0, sizeof(ans));
        sort(ac, ac+2*n);
        mp.clear();
        mp.insert(make_pair(info(-1, 1), 0));
        mp.insert(make_pair(info(-1, -1), 0));
        for(LL i = 0; i < 2 * n; i++)
        {
            curx = ac[i].x;
            if(ac[i].ty == 1){
                ins(ac[i].y, ac[i].id);
            }else{
                mp.erase(itr[ac[i].id][0]);
                mp.erase(itr[ac[i].id][1]);
```

```
            }
        }
        for(LL i = 1; ans[i]; i++)
        {
            printf("%lld\n", ans[i]);
        }
    }
    return 0;
}
```

扫描线_圆交

```
/*
* 题意：给出n个不相交，不接触的圆，求的是恰好包含i个圆的圆有多少个。
*
* 做法：大概都是以x为扫描线，比较y的值，这里y的值是动态比较的，虽然是变化的，但是map中的相对顺序却
*       并不会改变。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <map>
//#include <unordered_map>
#define N 200020
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
#define ID(x, y) ((x)*m+(y))
#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f3f3f3f3fLL;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("/home/qingping/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}

#define D(x) ((x)*(x))
const double eps = 1e-8;
LL n;
struct Cir{
    LL x, y, r, ty, id;
    void read()
    {
        scanf("%lld%lld%lld", &x, &y, &r);
    }
    Cir(){}
    Cir(LL _x, LL _y, LL _r, LL _id, LL _ty = 0){
```

```
        x = _x, y = _y, r = _r, id = _id, ty = _ty;
    }
    bool operator<(const Cir& o)const{
        return x < o.x;
    }
}c[N], ac[N];
LL curx;
double delty, deltoy, cy, oy;
struct info{
    LL id, ty;
    info(){}
    info(LL _id, LL _ty){
        id = _id, ty = _ty;
    }
    double y()const{
        if(id == -1) return (ty == 1 ? INF : -INF);
        delty = sqrt((double)D(c[id].r) - D(curx - c[id].x));
        if(ty == 1) return c[id].y + delty;
        return c[id].y - delty;
    }
    bool operator<(const info& o)const{
        cy = y(), oy = o.y();
        if(abs(cy - oy) < eps) return ty < o.ty;
        return y() < o.y();
    }
};
LL ans[N];
multimap<info, LL> mp;
multimap<info, LL>::iterator it, preit, nxtit, itr[N][2];
LL nxtid, preid;
bool CinsC(LL a, LL b)
{
    double d = sqrt((double)D(c[a].x - c[b].x) + D(c[a].y - c[b].y));
    if(c[a].r + c[b].r > d + eps) return true;
    else return false;
}
void ins(LL y, LL id){
    it = mp.insert(make_pair(info(id, 1), 0));
    //it = mp.find(info(id, 1));
    nxtit = preit = it;
    LL res = 0;
    nxtit++, preit--;
    nxtid = (nxtit -> first).id;
    if(nxtid == -1 || CinsC(id, nxtid)) res = max(res, (nxtit ->
second)+1);
    else res = max(res, nxtit -> second);
    preid = (preit -> first).id;
    if(preid == -1 || CinsC(id, preid)) res = max(res, (preit ->
second)+1);
    else res = max(res, preit -> second);
    it -> second = res;
    itr[id][0] = it;
    itr[id][1] = mp.insert(make_pair(info(id, -1), res));
    ans[res]++;
}
int main()
{
//    Open();
    LL T;scanf("%lld", &T);
    while(T--)
    {
```

```
        scanf("%lld", &n);
        for(LL i = 0; i < n; i++)
        {
            c[i].read();
            ac[i*2] = Cir(c[i].x - c[i].r, c[i].y, c[i].r, i, 1);
            ac[i*2 + 1] = Cir(c[i].x + c[i].r, c[i].y, c[i].r, i, -1);
        }
        memset(ans, 0, sizeof(ans));
        sort(ac, ac+2*n);
        mp.clear();
        mp.insert(make_pair(info(-1, 1), 0));
        mp.insert(make_pair(info(-1, -1), 0));
        for(LL i = 0; i < 2 * n; i++)
        {
            curx = ac[i].x;
            if(ac[i].ty == 1){
                ins(ac[i].y, ac[i].id);
            }else{
                mp.erase(itr[ac[i].id][0]);
                mp.erase(itr[ac[i].id][1]);
            }
        }
        for(LL i = 1; ans[i]; i++)
        {
            printf("%lld\n", ans[i]);
        }
    }
    return 0;
}
```

极角离散化_求区间内的完整区间个数，求区间不相交的个数

```
/*
*   给出n个点，求pair(x, y)满足第x个点和第y个点的直线与原点的坐标小于d的pair数。
*
*   首先对于圆内的点来说，和其他所有点组成的点对都是满足条件的，那么考虑在圆外的点。
*   对于圆外一个特定点，做圆的两条切线，会得到一个圆弧区间。那么可以分析得出的是：
*   两点直线经过圆的重要条件是，这两点对应的圆弧区间要么包含，要么相离，那么对于一个点来说。我们只需要分别统计
*   与这个区间相离，这个区间包含完整区间的个数，最后加起来即可。这里需要小心的是相离是双向关系，单独统计最后除2即可。
*   于是，现在问题转化为上述形式，就可以用树状数组搞一下啦！
*   当然面临的问题还有，需要将圆周断开成为一条链，断开的地方不能有区间覆盖。这里我们选择极角作为这条链，那么对于
*   那种跨过端点的区间，我们只需要将该区间取反即可，这样是不影响最终结果的，这一点简单画图可知。
*
*   对了，值得注意的是，这里要求的是距离小于d的，等于的也是不行的！
*   另外，为了防止精度问题，这里手写了离散化二分查找...(对于整型来说都是sort->unique->lower_bound,但是这里不敢用)
*/

#include <iostream>
```

```cpp
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <map>
//#include <unordered_map>
#define N 600010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    double x,y;
    Point(double x = 0, double y = 0):x(x),y(y){}
}pa[N],pb[N];

struct Circle
{
    Point c;
    double r;
    Circle(){}
    Circle(Point c, double r):c(c),r(r){}
    Point point(double a)
    {
        return Point(c.x + cos(a)*r, c.y+sin(a)*r);
    }
}Cir;
const double eps = 1e-9;
const double PI = acos(-1.0);
typedef Point Vector;
int dcmp(double x){if(fabs(x) < eps) return 0;else return x<0?-1:1;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return
acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
```

```
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}
double torad(double ang){return ang / 180 * PI;}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
A.x/L);}//需要确保A不是0向量
bool OnSegment(Point p, Point a, Point b) { return dcmp(Length(p - a)
+ Length(p - b) - Length(a - b)) == 0; }//精度也很高的!
struct Line{
    Point p,v;
    double a,b,c;//得到一般式的参数
    Line(){}
    Line(Point p = Point(0,0), Vector v = Vector(0,0)):p(p),v(v){a =
v.y - p.y; b = p.x - v.x; c = p.y*v.x - v.y*p.x;}
    Point point(double t){return p + v*t;}//只能在点斜式中用
    bool operator < (const Line& L)const{
        return angle(v) < angle(L.v);
    }
};
//过点P的圆C的切线
double getTangents(Point P, Circle C)
{
    Vector u = C.c - P;
    double dist = Length(u);
    if(dcmp(dist - C.r) < 0) return -1;
    if(dcmp(dist - C.r) == 0) {
        return 0;
    }
    double ang = acos(C.r/dist);
//    v[0] = Rotata(u, -ang);
//    v[1] = Rotata(u, ang);
    return ang;
}
double NormalAng(double x)
{
    if(x > 0){
        while(x > PI) x -= 2.0 * PI;
    }else{
        while(x < -PI) x += 2.0 * PI;
    }
    return x;
}
int n, d;
struct S{
    double tst, ted;
    int st, ed;
    S(){}
    void upd()
    {
        if(st > ed) swap(st, ed);
    }
    bool operator<(const S& o)const{
        return ed < o.ed;
    }
}seg[N];
double tmp[N];
double sta[N];
int binaryS(double x, int tail)
{
    int res = tail;
    int lb = -1, ub = tail;
    while(lb + 1 < ub)
```

```
    {
        int mid = ub+lb >> 1;
        if(dcmp(sta[mid] - x) >= 0) ub = res = mid;
        else lb = mid;
    }
    return res;
}
int L[N], R[N];
int limit;
void add(int c[], int x, int val)
{
    if(x == 0) return ;
    for(int i = x; i < limit + 10; i += i & (-i)) c[i] += val;
}
int getsum(int c[], int x)
{
    int res = 0;
    for(int i = x; i ; i -= i & (-i)) res += c[i];
    return res;
}
int main()
{
//    Open();
    while(~scanf("%d%d", &n, &d))
    {
        for(int i = 0; i < n; i++)
        {
            int x, y;
            scanf("%d%d", &x, &y);
            pa[i] = Point(x, y);
//            scanf("%lf%lf", &pa[i].x, &pa[i].y);
        }
        Cir = Circle(Point(0, 0), d);
        int num0 = 0;
        int tailn = 0, tailtmp = 0;
        for(int i = 0 ; i < n; i++)
        {
            double ang = getTangents(pa[i], Cir);
            if(dcmp(ang) == -1) num0 ++;
            else{
                seg[tailn].tst = NormalAng(angle(pa[i]) - ang);
                seg[tailn++].ted = NormalAng(angle(pa[i]) + ang);
                tmp[tailtmp++] = seg[tailn-1].tst;
                tmp[tailtmp++] = seg[tailn-1].ted;
            }
        }
        sort(tmp, tmp+tailtmp);
        int tailsta = 0;
        for(int i = 0; i < tailtmp; i++)
        {
            if(i == tailtmp-1 || dcmp(tmp[i+1] - tmp[i]) != 0){
                sta[tailsta++] = tmp[i];
            }
        }
        for(int i = 0; i < tailn; i++)
        {
            seg[i].st = binaryS(seg[i].tst, tailsta) + 1;
            seg[i].ed = binaryS(seg[i].ted, tailsta) + 1;
            seg[i].upd();
        }
        sort(seg, seg+tailn);
```

```
        limit = seg[tailn-1].ed;
        if(limit > N) while(1);
        for(int i = 0; i < limit + 10; i++) L[i] = R[i] = 0;
        LL ans2 = 0, ans = 0;
        for(int i = 0; i < tailn; i++){
            ans2 += getsum(R, seg[i].st-1);
            ans += getsum(L, seg[i].ed - 1) - getsum(L, seg[i].st);
            add(R, seg[i].ed, 1);
            add(L, seg[i].st, 1);
        }
        for(int i = 0; i < limit + 10; i++) L[i] = 0;
        for(int i = tailn-1; i >= 0;i --)
        {
            ans2 += getsum(L, limit) - getsum(L, seg[i].ed);
            add(L, seg[i].st, 1);
        }
        ans += (LL)num0 * (n - num0);
        ans2 += (LL)num0 * (num0 - 1);
        if(ans2 < 0 || ans < 0) while(1);
        printf("%lld\n", ans2/2 + ans);
    }
    return 0;
}
```

简单多边形与圆交周长_面积交模板

```
/*
* 多边形与圆交面积模板中，每次加上去的都是有向面积，这样的话，多的面积会被容斥掉。对于此题来说，要求的是周长
* 由于遍历顺序为多边形的点序，那么多边形上的周长是不可能容斥出去的，而是全部都会被当做正值加到最终答案中。
* 而对于一段圆弧来说，根据遍历顺序是能够将多余的部分减去的，所以这里仍然采用之前的方法--加上有向周长。
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
```

```
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Point
{
    double x,y;
    Point(double x = 0, double y = 0):x(x),y(y){}
}pa[N],pb[N];

typedef Point Vector;
const double eps = 1e-10;
const double PI = acos(-1.0);
int dcmp(double x){if(fabs(x) < eps) return 0;else return x<0?-1:1;}
Vector operator+(Vector A,Vector B){return Vector(A.x+B.x, A.y+B.y);}
Vector operator-(Point A,Point B){return Vector(A.x-B.x, A.y-B.y);}
Vector operator*(Vector A, double p){return Vector(A.x*p, A.y*p);}
Vector operator/(Vector A, double p){return Vector(A.x/p, A.y/p);}
bool operator<(const Point& a, const Point& b){return a.x<b.x || (a.x
== b.x && a.y < b.y);}
bool operator==(const Point& a, const Point& b){return dcmp(a.x-b.x)
== 0 && dcmp(a.y-b.y) == 0;}
double angle(Vector A){return atan2(A.y,A.x);}
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return
acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A, Vector B){return A.x*B.y-A.y*B.x;}
Vector Rotata(Vector A,double rad){return Vector(A.x*cos(rad)-
A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));}
double torad(double ang){return ang / 180 * PI;}
Vector Normal(Vector A){double L = Length(A); return Vector(-A.y/L,
A.x/L);}//需要确保A不是0向量
bool OnSegment(Point p, Point a, Point b) { return dcmp(Length(p - a)
+ Length(p - b) - Length(a - b)) == 0; }//精度也很高的!
struct Circle
{
    Point c;
    double r;
    Circle(){}
    Circle(Point c, double r):c(c),r(r){}
    Point point(double a)
    {
        return Point(c.x + cos(a)*r, c.y+sin(a)*r);
    }
};
struct Line{
    Point p,v;
    double a,b,c;//得到一般式的参数
    Line(Point p = Point(0,0), Vector v = Vector(0,0)):p(p),v(v){a =
v.y - p.y; b = p.x - v.x; c = p.y*v.x - v.y*p.x;}
    Point point(double t){return p + v*t;}//只能在点斜式中用
    bool operator < (const Line& L)const{
        return angle(v) < angle(L.v);
    }
}L[N];
double DistanceToLine(Point P, Point A, Point B)
{
    Vector v1 = B-A,v2 = P-A;
    return fabs(Cross(v1,v2)) / Length(v1);
}
```

```
//直线与圆相交,直线必须是点斜式
int getLineCircleIntersection(Line L, Circle C, vector<Point >& sol)
{
    double a = L.v.x, b = L.p.x - C.c.x, c = L.v.y, d = L.p.y -
C.c.y;
    double e = a*a + c*c, f = 2*(a * b + c * d), g = b*b+d*d-C.r*C.r;
    double delta = f*f - 4*e*g;
    double dist = DistanceToLine(C.c, L.p, L.p + L.v);
    double t1,t2;
    if(dcmp(dist - C.r) > 0) return 0;
    if(dcmp(dist - C.r) == 0) {
        t1 = t2 = -f / (2*e);
        if(dcmp(t1)>= 0 && dcmp(1-t1)>=0)sol.push_back(L.point(t1));
        return 1;
    }
    t1 = (-f - sqrt(delta)) / (2*e);
    if(dcmp(t1)>= 0 && dcmp(1-t1)>=0) sol.push_back(L.point(t1));
    t2 = (-f + sqrt(delta)) / (2*e);
    if(dcmp(t2)>= 0 && dcmp(1-t2)>=0) sol.push_back(L.point(t2));
    return 2;
}
struct CPIArea
{
    Circle cir;
    double Scir;

    Point p[N];
    int tail;
    CPIArea() { tail=0; }
    CPIArea(Circle cir):cir(cir) { Scir = PI*cir.r*cir.r; tail=0; }
    //tp[]是多边形的点集，n是点的个数。tp[]必须满足点是按顺时针或者逆时针排序的
    double solve(Point tp[],int n)
    {
        tail = 0;
        for(int i=0; i<n; i++)
        {
            p[tail++]=tp[i];//p[]是囊括了圆和多边形交点的点集，也是按顺时针或
者逆时针排序的
            Line line = Line(tp[i],tp[(i+1)%n] - tp[i]);
            vector<Point > sol;
            sol.clear();
            getLineCircleIntersection(line, cir, sol);

            for(int j=0; j<sol.size(); j++)
            {
                p[tail++]=sol[j];
            }
        }
        double Cirres = 0;
        double polyres = 0;
        for(int i=0; i<tail; i++)
        {
            Point O = cir.c;
            if(dcmp(Cross( p[i]-O , p[(i+1)%tail]-O)) == 0)
            {
                Point mid = (p[i]+p[(i+1)%tail])/2;
                if(dcmp(Length(mid - O) - cir.r) < 0){
                    polyres += Length(p[i] - p[(i+1)%tail]);
                }
                continue;
```

```
            }
            double ang = Angle(p[(i+1)%tail]-O , p[i]-O);
            if( dcmp(Cross( p[i]-O , p[(i+1)%tail]-O)) > 0 ) ang*=1;
            else ang*=-1;
            double Sshan = ang/(2*PI)*Scir;
            double Strian = Cross(p[i] - O, p[(i+1)%tail] - O) * 0.5;
            //Area(O , p[i] ,p[(i+1)%tail] );
            if(dcmp( abs(Sshan) - abs(Strian)) <=0  )
            {
                Cirres += ang * cir.r;
            }
            else polyres += abs(Length(p[i] - p[(i+1)%tail]));
        }
        return abs(Cirres) + polyres;
    }
};
int main()
{
    //Open();
    int n;
    while(scanf("%d", &n), n){
        for(int i = 0 ; i < n; i++)
        {
            scanf("%lf%lf", &pa[i].x, &pa[i].y);
        }
        double rx, ry, r;
        scanf("%lf%lf%lf", &rx, &ry, &r);
        Circle c(Point(rx, ry), r);
        CPIArea solver(c);
        double ans = solver.solve(pa, n);
        printf("%.0f\n", ans);
    }
    return 0;
}

/*
5
0 0
1 1
3 1
4 0
2 -2
2 0 1


4
0 0
2 2
4 0
2 -2
2 0 3

4
0 0
2 2
4 0
2 -2
2 0 1

3
1 1
```

```
1 -1
0 0
2 0 3

3
1 1
1 -1
0 0
2 0 1


5
0 1
1 0
2 1
2 -1
0 -1
1 0 1

4
-10 -10
-10 10
10 10
10 -10
-10 -10 20
4
-10 -10
-10 10
10 10
10 -10
-10 -10 4
4
-10 -10
-10 10
10 10
10 -10
0 0 10
4
-40 -40
-40 40
40 40
40 -40
0 0 50

4
-10 -10
-10 10
10 10
10 -10
-10 -10 20
4
-10 -10
-10 10
10 10
10 -10
-10 -10 4
4
-10 -10
-10 10
10 10
10 -10
```

```
0 0 10
4
-40 -40
-40 40
40 40
40 -40
0 0 50
0

6.2831853072
11.3137084990
6.2831853072
4.8284271247
0.0000000000
6.7123889804
71.4159265359
14.2831853072
62.8318530718
296.7588218417
71.4159265359
14.2831853072
62.8318530718
296.7588218417

*/
```

乱搞

子数组与和和异或和相同的个数_固定起点

```
/*
* 题意：题意如上。。
*
* 做法：对于数组的与和来说，固定起点的话，不同的值只可能有log个，那么只需要维护固定起点之后，这个
*       数组有多少个j满足i-j区间的异或和等于一个定值即可。由于异或的特殊性质，可以先重做数组为了求
*       异或和，具体代码中有
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <set>
#include <map>
#include <ctime>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
```

```
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
const LL INF=0x3f3f3f3f;
void Open()
{
//    freopen("D:/in.txt","r",stdin);
    freopen("hack.in","r",stdin);
    freopen("hack.out","w",stdout);
}

LL sta[N];
LL a[N];
LL f[N];
LL pre[44];
vector<LL> id[N];
set<PII>::iterator sit;
LL n;
int main()
{
    Open();
    while(~scanf("%I64d", &n))
    {
        LL tail = 0;
        sta[tail++] = f[n-1] = 0;
        for(LL i = 0; i < n; i ++) scanf("%I64d", &a[i]);
        for(LL i = n - 2; i >= 0; i--) f[i] = f[i+1] ^ a[i + 1],
sta[tail++] = f[i];
        sort(sta, sta+tail);
        tail = unique(sta, sta+tail) - sta;
        for(LL i = 0; i <= tail; i ++) id[i].clear();
        for(LL i = 0; i < n; i++)
        {
            f[i] = lower_bound(sta, sta + tail, f[i]) - sta;
            id[f[i]].push_back(i);
        }
        for(LL i = 0; i < 31; i ++) pre[i] = n;
        LL ans = 0;
        LL X = 0;
        for(LL i = n - 1; i >= 0; i--)
        {
            X ^= a[i];
            set<PII > S;
            for(LL j = 0; j < 31; j++)
            {
                if(!(a[i] & (1 << j)))
                    pre[j] = min(i, pre[j]);
                if(pre[j] < n) S.insert(PII(pre[j], j));
            }
            S.insert(PII(n, 0));
            LL val = (1LL << 31) - 1;
            LL preidx = i;
            for(sit = S.begin(); sit != S.end(); sit++)
            {
                LL curidx = (*sit).first, dig = (*sit).second;
                if(curidx != preidx)
                {
                    LL findidx = lower_bound(sta, sta+tail, val ^ X) -
sta;
                    if(sta[findidx] == (val ^ X))
                    {
```

```
                    LL r = upper_bound(id[findidx].begin(),
id[findidx].end(), curidx - 1) - id[findidx].begin() - 1;
                    LL l = lower_bound(id[findidx].begin(),
id[findidx].end(), preidx) - id[findidx].begin();
                    ans += max(0LL, (r - l + 1));
                }
            }
            preidx = curidx;
            val ^= (1LL << dig);
        }
    }
    printf("%I64d\n", ans);
  }
  return 0;
}
```

树

动态 MST

版本一

```
/*
这份代码计算
一个图中有多少条边必在MST上
也等价于，将图中某一条边更改后，用O（1）的复杂度在线回答新的图的MST数值。查询之前
需要O（n^2）（n为点数）的时间来预处理。
*/
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<vector>
using namespace std;
#define scan(x) scanf("%d",&(x))
#define scan2(x,y) scanf("%d%d",&(x),&(y))
#define scan3(x,y,z) scanf("%d%d%d",&(x),&(y),&(z))
#define scan4(x,y,z,k) scanf("%d%d%d%d",&(x),&(y),&(z),&(k))
const int maxn = 3003;
const int maxm = maxn * maxn;
const int inf = 1000000000;

int n, m;
__int64 mst;
int map[maxn][maxn];
int dp[maxn][maxn], best[maxn][maxn];
int dis[maxn], pre[maxn];
bool vis[maxn];
vector<int> edge[maxn];
#include<iostream>
int minz(int a, int b)
{
  return a < b ? a : b;
}
```

```
void init()
{
    int i, j;
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
            map[i][j] = dp[i][j] = inf;
        edge[i].clear();
        vis[i] = 0;
        pre[i] = -1;
        dis[i] = inf;
    }
}
void input()
{
//cerr<<"st"<<endl;
//    int x, y, z;
//    while(m--)
//    {
//        scanf("%d%d%d", &x, &y, &z);
//        cerr<<x<<" "<<y<<" "<<z<<endl;
//        map[x][y] = map[y][x] = z;
//    }

        for(int i=0;i<n;i++){
                for(int j=i+1;j<n;j++){
                        int w;scan(w);
                        map[i][j] = map[j][i] = w;
                        //cerr<<i<<" "<<j<<" "<<w<<endl;
                }
        }
}

void prim()
{
    int i, j, k;
    for(i = 1; i < n; i++)
    {
        dis[i] = map[0][i];
        pre[i] = 0;
    }
    dis[0] = inf;
    vis[0] = 1;
    pre[0] = -1;
    mst = 0;

    for(i = 0; i < n-1; i++)
    {
        k = 0;
        for(j = 1; j < n; j++)
            if(!vis[j] && dis[k] > dis[j])
                k = j;

        vis[k] = 1;
        mst += dis[k];
       //建最小生成树
        if(pre[k] != -1)
            edge[k].push_back(pre[k]),
            edge[pre[k]].push_back(k);

        for(j = 1; j < n; j++)
```

```
            if(!vis[j] && dis[j] > map[k][j] )
                dis[j] = map[k][j], pre[j] = k;
    }
}

int dfs1(int u, int fa, int rt) // 求 点rt 到 以u为根的数及其子树的最小距离
{
    int i;
    for(i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if(v == fa) continue;
        dp[rt][u] = minz(dp[rt][u], dfs1(v, u, rt));
    }
    if(fa != rt) dp[rt][u] = minz(dp[rt][u], map[rt][u]);
    return dp[rt][u];
}

int dfs2(int u, int fa, int rt) // 求 以rt为根的数及其子树 到 以u为根的数及
其子树的最小距离
{
    int i;
    int ans = dp[u][rt];
    for(i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if(v == fa) continue;
        ans = minz(ans, dfs2(v, u, rt));
    }
    return ans;
}

void solve()
{
    int i,j;
    for(i = 0; i < n; i++)
        dfs1(i, -1, i);
    for(i = 0; i < n; i++)
        for(j = 0; j < edge[i].size(); j++)
        {
            int v = edge[i][j];
            best[i][v] = best[v][i] = dfs2(v, i, i);
        }
}

void query()
{
//    int x, y, z;
//    double sum = 0;
//    scanf("%d", &m);
//    x=1,y=3,z=10;
//        if(pre[x] != y && pre[y] != x)
//            sum += mst * 1.0;
//        else
//            sum += mst * 1.0 - map[x][y] + minz(best[x][y], z);
//    printf("%.4f\n", sum/1);

    int ans =0 ;
    for(int i=0;i<n;i++){
        int x=i;
        for(int j=0;j<edge[i].size() ; j++){
```

```
                int y=edge[i][j];
                if(x<y) continue;
                int sum = mst - map[x][y] + best[x][y];
                //cerr<<"best:"<<best[x][y]<<endl;
                if(sum != mst){
                        ans++;
                }
        }
    }
    printf("%d\n",ans);
}

//int main()
//{
//freopen("C:/OJ/in.txt","r",stdin);
//    while( ~scanf("%d%d", &n, &m) && n + m)
//    {
//        init();
//        input();
//        prim();
//        solve();
//        query();
//    }
//    return 0;
//}

int main(){
    #ifndef ONLINE_JUDGE
        freopen("C:/OJ/in.txt","r",stdin);
    #endif
    int T;scan(T);
    while(T--){
                scan(n);
                m=n-1;
                init();
                input();
               // cerr<<"start:"<<map[0][1]<<endl;
                prim();
                //std::cerr<<mst<<std::endl;
                solve();
                query();
               // cerr<<"end:"<<map[0][1]<<endl;
    }
}
```

版本二

```
#include<iostream>
#include<stdio.h>
#include<string.h>
#include<cmath>
#include<algorithm>
#include<vector>
using namespace std;
#define scan(x) scanf("%d",&(x))
#define scan2(x,y) scanf("%d%d",&(x),&(y))
#define scan3(x,y,z) scanf("%d%d%d",&(x),&(y),&(z))
#define scan4(x,y,z,k) scanf("%d%d%d%d",&(x),&(y),&(z),&(k))
```

```
const int maxn = 1010;
const int maxm = maxn * maxn;
const double inf = 0x3f3f3f3f3f3f3f3fLL;
typedef pair<double, double> PDD;
const double eps = 1e-8;

int n, k;
double mst;
double mp[maxn][maxn], dp[maxn][maxn], best[maxn][maxn], dis[maxn];
int pre[maxn];//
bool vis[maxn];//
vector<int> edge[maxn];//
PDD p[maxn];//
void init()
{
    int i, j;
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
            dp[i][j] = inf;
        edge[i].clear();
        vis[i] = 0;
        pre[i] = -1;
        dis[i] = inf;
    }
}
#define D(x) (x)*(x)
double dist(int i, int j)
{
    return sqrt((double)D(p[i].first - p[j].first) + D(p[i].second -
p[j].second));
}
void input()
{
    for(int i = 0; i < n; i++)
        scanf("%lf%lf", &p[i].first, &p[i].second);
    for(int i=0; i<n; i++)
        for(int j=i; j<n; j++)
            mp[i][j] = mp[j][i] = dist(i, j);
}
void prim()
{
    int i, j, k;
    for(i = 1; i < n; i++)
    {
        dis[i] = mp[0][i];
        pre[i] = 0;
    }
    dis[0] = inf;
    vis[0] = 1;
    pre[0] = -1;
    mst = 0;

    for(i = 0; i < n-1; i++)
    {
        k = 0;
        for(j = 1; j < n; j++)
            if(!vis[j] && dis[k] > dis[j] + eps)
                k = j;

        vis[k] = 1;
```

```
        mst += dis[k];
        //建最小生成树
        if(pre[k] != -1)
            edge[k].push_back(pre[k]),
                edge[pre[k]].push_back(k);

        for(j = 1; j < n; j++)
            if(!vis[j] && dis[j] > mp[k][j] + eps)
                dis[j] = mp[k][j], pre[j] = k;
    }
}

double dfs1(int u, int fa, int rt) // 求 点rt 到 以u为根的数及其子树的最小距离
{
    int i;
    for(i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if(v == fa) continue;
        dp[rt][u] = min(dp[rt][u], dfs1(v, u, rt));
    }
    if(fa != rt) dp[rt][u] = min(dp[rt][u], mp[rt][u]);
    return dp[rt][u];
}
double dfs2(int u, int fa, int rt) // 求 以rt为根的数及其子树 到 以u为根的数及其子树的最小距离
{
    int i;
    double ans = dp[u][rt];
    for(i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if(v == fa) continue;
        ans = min(ans, dfs2(v, u, rt));
    }
    return ans;
}
void solve()
{
    int i,j;
    for(i = 0; i < n; i++)
        dfs1(i, -1, i);
    for(i = 0; i < n; i++)
        for(j = 0; j < edge[i].size(); j++)
        {
            int v = edge[i][j];
            best[i][v] = best[v][i] = dfs2(v, i, i);
        }
}
void query()
{
    double ans = mst;
    for(int i=1; i<n; i++)
    {
        int x=i;
        for(int j=0; j<edge[i].size() ; j++)
        {
            int y=edge[i][j];
            if(y == 0) continue;
```

```
//            if(x<y) continue;
            double sum = mst - mp[x][y] + best[x][y];//O(1)回答去除每条边之后重新生成最小生成树的权值和
            ans = max(sum, ans);
        }
    }
    printf("%.2f\n",ans*k);
}

int main()
{
#ifndef ONLINE_JUDGE
    //freopen("F:/in.txt","r",stdin);
#endif
    int T;
    scan(T);
    while(T--)
    {
        scanf("%d%d", &n, &k);
        init();
        input();
        prim();
        solve();
        query();
    }
}
```

Dfs 序处理树

CF 570D dfs 序 压位

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
#include <set>
//#include <unordered_map>
#define N 500010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
vector<int> G[N];
vector<vector<int > > deplist;
```

```
vector<vector<int > > depch;
int st[N];
int ed[N];
char num[N];
int Tn = 0;
int mad;
int n, m;
char str[N];
void dfs(int u, int d)
{
    st[u] = ++Tn;
    num[Tn] = str[u];
    if(d == deplist.size())
        deplist.push_back(vector<int>());
    deplist[d].push_back(st[u]);
    mad = max(mad, d);
    for(int i = 0;i<G[u].size();i++)
    {
        int v = G[u][i];
        dfs(v, d+1);
    }
    ed[u] = Tn;
}
int main()
{
    Open();
    scanf("%d%d", &n, &m);
    for(int i=2;i<=n;i++){
        int x;
        scanf("%d", &x), G[x].push_back(i);
    }
    scanf("%s", str+1);
    mad = -1;
    dfs(1, 0);
    for(int i = 0; i <= mad; i++)
    {
        for(int j = 0; j<deplist[i].size(); j++)
        {
            int u = num[deplist[i][j]] - 'a';
            if(j == 0) depch.push_back(vector<int>());

            depch[i].push_back((1<<u));
        }
        for(int j=1;j<depch[i].size();j++)
            depch[i][j] = depch[i][j] ^ depch[i][j-1];
    }
    for(int i=0;i<m;i++)
    {
        int v, h;
        scanf("%d%d", &v, &h);
        h--;
        if(h > mad) {
            printf("Yes\n");
            continue;
        }
        int l = lower_bound(deplist[h].begin(), deplist[h].end(),
st[v]) - deplist[h].begin();
        int r = upper_bound(deplist[h].begin(), deplist[h].end(),
ed[v]) - deplist[h].begin();
        r--;
        if(l == deplist[h].size() || r == deplist[h].size() || l > r)
```

```
        {
            printf("Yes\n");
            continue;
        }
        int cnt = 0;
        if(l == 0) cnt = depch[h][r];
        else cnt = depch[h][r] ^ depch[h][l-1];

        if(__builtin_popcount(cnt) > 1){
            printf("No\n");
        }else {
            printf("Yes\n");
        }
    }
    return 0;
}
```

CF_208E_dfs 序区间中某个值的数目

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
int deep[N], pa[20][N];
int n, m;
vector<int > G[N];
vector<int > depid[N];
int Tn = 1;
int ans[N];
int st[N], ed[N];
int dfs_init(int u, int fa, int dep)
{
    deep[u] = dep; pa[0][u] = fa;
    st[u] = Tn++;
    depid[dep].push_back(st[u]);
    for(int i = 0; i < G[u].size(); i++)
        dfs_init(G[u][i], u, dep+1);
    ed[u] = Tn;
```

```
}
void lca_init()
{
    int root = 0;
    dfs_init(root, -1, 0);
    for(int k = 0; k + 1 < 20; k ++)
        for(int v = 0; v <= n; v++)
            if(pa[k][v] < 0) pa[k+1][v] = -1;
            else pa[k+1][v] = pa[k][pa[k][v]];
}
int find_pfa(int u, int p)
{
    for(int i = 19; i >= 0; i--)
        if(p & (1 << i)) u = pa[i][u];
    return u;
}
int find_ans(int u, int dep){
    int l = st[u], r = ed[u];
    int lid = lower_bound(depid[dep].begin(), depid[dep].end(), l) -
depid[dep].begin();
    int rid = upper_bound(depid[dep].begin(), depid[dep].end(), r) -
depid[dep].begin();
    return max(0, rid - lid - 1);
}
int main()
{
    Open();
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
        int x;
        scanf("%d", &x);
        G[x].push_back(i);
    }
    scanf("%d", &m);
    lca_init();
    for(int i = 0; i < m; i++)
    {
        int v, p;
        scanf("%d%d", &v, &p);
        int pfa = find_pfa(v, p);
        if(pfa > 0) printf("%d", find_ans(pfa, deep[v]));
        else printf("0");
        if(i == m - 1) putchar('\n');
        else putchar(' ');
    }
    return 0;
}
```

分治

链分治-重链线段树

```
/*
* SPOJ Query on a tree IV
*
* 据说这是query on a tree 1~5 中最繁琐的一道题，的确是这样！
*
```

```
* 解法来自：分治算法在树的路径问题中的应用--漆子超
* 我将论文中我觉得没说清楚的地方说一下。
* 此题首先需要将原树轻重链剖分，然后为每一条重链都维护一棵线段树，两条重链间由一条轻边链接。
* 每条重链上需要维护这样的信息：maxl, maxr, opt; opt 就是在这条链上的答案。
* 首先对每一个点处理出这样一个量D[i], D2[i],表示这个节点向下至某个白色结点的路径中长度的最大值和次大值(路
* 径中不包含同重链的点，如果没有白色节点不存在，长度记为-INF)
* 那么线段树的某个区间[l, r]的opt的含义就是d[i]+dist[i,j]+d[j],i,j都在区间中，dist[i,j]表示第i个节点到第j
* 个节点的距离；而maxl，maxr是用来维护opt的量，maxl表示一条左端点在该区间内且还可以向左延伸，右端点为一个
* 白色节点的路径的最大长度；maxr相反。那么由这两个量父亲节点的opt可以得到维护(opt[fa] = maxr[lc]+maxl[rc]+dist[mid, mid+1])
* 于是，整棵树就被若干棵线段树得以维护。最终答案只需要用一个堆存储所有线段树的opt值即可。
* 但是其中还有一个特别重要的d[]和d2[]数组，这两个数组的维护在论文里面写的比较清楚，这里就不说了。大概就是
* 每个节点开一个堆，用来记录所有可能的d值，然后就能o(1)取出来了。
* 对于修改操作，由于一个节点到根的路径上不超过log条轻边，那么需要修改的线段树也不超过log个，修改线段树log
*
* 复杂度为O(Qlog(n)^2)
*
* 当然这其中包含众多下标处理...首先每条重链都有一个线段树，不可能每颗线段树都开一个数组，所以开了一个节点池
* 每个区间记录儿子所在的下标，总的节点数不超过6*N个。这里需要知道每个节点对应那颗线段树，反过来也需要知道
* 每颗线段树上的节点对应原树上的那个节点。这里的处理方法是利用重儿子将原树节点重新排列，记录每个线段树根节点
* 的下标，并记录线段树起始节点的偏移量，那么根据这些信息即可查找与反找。每个数组的意义代码中有详细注释。上面
* 的dist值也可以利用这个来维护。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
```

```
    #endif // ONLINE_JUDGE
}
const int MAXN = 100010;
int top[MAXN];//top[v]表示v所在的重链的顶端节点
int fa[MAXN]; //父亲节点
int deep[MAXN];//深度
int sz[MAXN];//sz[v]表示以v为根的子树的节点数
int p[MAXN];//p[v]表示v与其父亲节点的连边在线段树中的位置
int fp[MAXN];//和p数组相反
int son[MAXN];//重儿子
int col[MAXN];//颜色
int val[MAXN];
int pos;
priority_queue<PII > que[N], ansque;//que:存储D(i)
struct Edge
{
    int to, w, nxt;
    Edge(){}
    Edge(int _t, int _w, int _next){
        to = _t, w = _w, nxt = _next;
    }
}e[MAXN*2];
int head[MAXN],tot;
void addedge(int u,int v, int w)
{
    e[tot] = Edge(v, w, head[u]);
    head[u] = tot++;
}

//-------segment Tree---------//
struct node{
    int l, r, maxl, maxr, opt;
    int ch[2];
    node(){}
    node(int _l, int _r, int _ml, int _mr, int _op){
        l = _l, r = _r, maxl = _ml, maxr = _mr, opt = _op;
        ch[0] = ch[1] = 0;
    }
}lt[N*8];
int nodenum = 0;
int root[N];//root[i]表示节点i(head)所在的重链表示的线段树在pool中的位置
int pn[N];//类似DFS序，相当于将节点重排，将同一条重链上的节点放到一起
int idx[N];//idx[i]表示节点i在他所在的重链线段树上的rank
int num[N];//num[i]表示节点i(head)所在的重链有多少个节点
int dis[N];//dis[i]表示节点i到对应线段树最左节点的距离
int sonw[N];

void dfs1(int u,int pre,int d) //第一遍dfs求出fa,deep,sz,son,val
{
    deep[u] = d;
    fa[u] = pre;
    sz[u] = 1;
    for(int i = head[u];i != -1; i = e[i].nxt)
    {
        int v = e[i].to;
        if(v != pre)
        {
            val[v] = e[i].w;
            dfs1(v,u,d+1);
```

```
            sz[u] += sz[v];
            if(son[u] == -1 || sz[v] > sz[son[u]])
                son[u] = v, sonw[u] = val[v];
        }
    }
}
int Tn;
int fpn[N];
void getpos(int u,int sp, int dist) //第二遍dfs求出top和p,num;
{
    pn[++Tn] = u;
    fpn[u] = Tn;
    dis[Tn] = dist;
    top[u] = sp;
    num[sp]++;
    if(son[u] != -1)
    {
        p[u] = pos++;
        fp[p[u]] = u;
        getpos(son[u],sp, dist + sonw[u]);
    }
    else
    {
        p[u] = pos++;
        fp[p[u]] = u;
        return;
    }
    for(int i = head[u] ; i != -1; i = e[i].nxt)
    {
        int v = e[i].to;
        if(v != son[u] && v != fa[u])
            getpos(v,v, 0);
    }
}

int add(int a, int b)
{
    if(a == -INF || b == -INF) return -INF;
    return a + b;
}
void push_up(int off, int rt)
{
    int lc = lt[rt].ch[0], rc = lt[rt].ch[1];
    int mid = lt[rt].l + lt[rt].r >> 1;
    lt[rt].maxl = max(lt[lc].maxl, add(lt[rc].maxl,  dis[off + mid +
1] - dis[off + lt[rt].l]));
    lt[rt].maxr = max(lt[rc].maxr, add(lt[lc].maxr,  dis[off +
lt[rt].r] - dis[off + mid]));
    lt[rt].opt = max(lt[lc].opt, max(lt[rc].opt, add(add(lt[lc].maxr,
lt[rc].maxl), dis[off + mid + 1] - dis[off + mid])));
}
int build(int l, int r){
    int rt = nodenum++;
    lt[rt] = node(l, r, -INF, -INF, -INF);
    if(l == r) return rt;
    int mid = l + r >> 1;
    lt[rt].ch[0] = build(l, mid);
    lt[rt].ch[1] = build(mid+1, r);
    return rt;
}
void update(int id, int x, int off, bool isrt)
```

```
{
    if(lt[x].l == id && lt[x].r == id)
    {
        int u = pn[off + id];
        int d = -INF, sond = -1, d2 = -INF;
        while(!que[u].empty())
        {
            PII pp = que[u].top();que[u].pop();
            int v = pp.second, w = pp.first;
            if(lt[root[v]].maxl + val[v] != w) continue;
            d = w;sond = v;
            break;
        }
        while(!que[u].empty()){
            PII pp = que[u].top();
            int v = pp.second, w = pp.first;
            if(v == sond || lt[root[v]].maxl + val[v] != w)
{que[u].pop();continue;}
            d2 = w;
            break;
        }
        if(sond != -1) que[u].push(PII(d, sond));
        if(!col[u]) d = max(d, 0);
        if(col[u]){
            lt[x].maxl = lt[x].maxr = d;
            lt[x].opt = add(d, d2);
        }else{
            d = max(d, 0);
            lt[x].maxl = lt[x].maxr = d;
            lt[x].opt = max(d, add(d, d2));
        }
        if(isrt && lt[x].opt != -INF) ansque.push(PII(lt[x].opt, x));
        return ;
    }
    int mid = lt[x].l + lt[x].r >> 1;
    if(id <= mid) update(id, lt[x].ch[0], off, false);
    else update(id, lt[x].ch[1], off, false);
    push_up(off, x);
    if(isrt && lt[x].opt != -INF) ansque.push(PII(lt[x].opt, x));
}
int dfsFire(int u, int p, int id)
{
    idx[u] = id;
    if(id == 1) root[u] = build(1, num[u]);
    int w;
    for(int i = head[u]; ~i; i = e[i].nxt)
    {
        int v = e[i].to;
        if(v == p) continue;
        if(v == son[u]) {w = e[i].w;continue;}
        dfsFire(v, u, 1);
        que[u].push(PII(lt[root[v]].maxl+val[v], v));
    }
    update(idx[u], root[top[u]], fpn[top[u]]-1, true);
    if(son[u] != -1) dfsFire(son[u], u, id+1);
}

void init(int n)
{
    tot = 0;
    memset(head,-1,sizeof(head));
```

```
    pos = 0;
    Tn = 0;
    nodenum = 0;
    memset(son,-1,sizeof(son));
    memset(num, 0,sizeof(num));
    memset(col, 0, sizeof(col));
    while(!ansque.empty()) ansque.pop();
    for(int i = 0; i <= n; i++)
        while(!que[i].empty()) que[i].pop();
}

int main()
{
//    Open();
    int n;
    while(~scanf("%d", &n))
    {
        init(n);
        for(int i = 1; i < n; i++)
        {
            int u, v, w;scanf("%d%d%d", &u, &v, &w);
            addedge(u, v, w);
            addedge(v, u, w);
        }
        int res = n;
        dfs1(1, 0, 0);
        getpos(1, 1, 0);
        dfsFire(1, 0, 1);
        int q;scanf("%d", &q);
        while(q--)
        {
            char op[33];
            scanf("%s", op);
            if(op[0] == 'A'){
                if(res == 0){
                    puts("They have disappeared.");
                    while(!ansque.empty()) ansque.pop();
                }else if(res == 1){
                    puts("0");
                    while(!ansque.empty()) ansque.pop();
                }else{
                    bool flag = true;
                    while(!ansque.empty()){
                        PII pp = ansque.top();
                        int x = pp.second, w = pp.first;
                        if(lt[x].opt != w){
                            ansque.pop();
                            continue;
                        }
                        flag = false;
                        printf("%d\n", w);
                        break;
                    }
                    if(flag){
                        puts("They have disappeared.");
                    }
                }
            }else{
                int u;scanf("%d", &u);
                col[u] ^= 1;
                if(col[u]) res--;
```

```
                else res++;
                while(u){
                    int tp = top[u];
                    update(idx[u], root[tp], fpn[tp]-1, true);
                    if(lt[root[tp]].maxl != -INF)
                        que[fa[tp]].push(PII(lt[root[tp]].maxl + val[tp],
tp));
                    u = fa[tp];
                }
            }
        }
    }
    return 0;
}
```

QTREE5 树分治+优先队列

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 200010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int > PII;
const int INF=0x3f3f3f3f;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct Reader{
    static const int MSIZE = 1000 * 8 * 1024;
    char buf[MSIZE], *pt = buf, *o = buf;
    void init(){
        fread(buf, 1, MSIZE, stdin);
    }
    char getch()
    {
        char ch;
        while((*pt < 'A' || *pt > 'Z') && (*pt < 'a' || *pt > 'z'))
pt++;
        ch = *pt;pt++;
        return ch;
    }
    int getint()
    {
        int f = 1, x = 0;
```

```
        while(*pt != '-' && !isdigit(*pt)) pt++;
        if(*pt == '-') f = -1, pt++;
        else x = *pt++ - 48;
        while(isdigit(*pt)) x = x * 10 + *pt++ - 48;
        return x * f;
    }
}frd;
struct edge{
    int to, nxt;
    edge(){}
    edge(int _t, int _n)
    {
        to = _t, nxt = _n;
    }
}e[N*2];
int head[N], eN = 0;
void addedge(int u, int v)
{
    e[eN] = edge(v, head[u]);
    head[u] = eN++;
}
priority_queue<PII, vector<PII>, greater<PII > > que[N];
vector<PII > wV[N];
int root, s[N], f[N];
int col[N];
bool vis[N];
int n;
int maxn;
int getroot(int now, int fa, int sz)
{
    int cnt=1;
    int mx=0;
    for(int i=head[now]; i!=-1; i=e[i].nxt)
    {
        int to=e[i].to;
        if(to==fa || vis[to]) continue;
        f[to]=getroot(to,now,sz);
        mx = max(mx,f[to]);
        cnt+=f[to];
    }
    mx = max(mx,sz-cnt);
    if(mx<maxn)
    {
        maxn=mx, root=now;
        for(int i = head[now]; i != -1; i = e[i].nxt)
        {
            int to = e[i].to;
            if(vis[to]) continue;
            if(to == fa)
            {
                s[to] = sz - cnt;
                continue;
            }
            s[to] = f[to];
        }
    }
    return cnt;
}
void dfsgao(int u, int fa, int rt, int dis)
{
    wV[u].push_back(PII(dis, rt));
```

```
    for(int i = head[u]; ~i ; i = e[i].nxt)
    {
        int v = e[i].to;
        if(vis[v] || v == fa) continue;
        dfsgao(v, u, rt, dis+1);
    }
}
void dfs(int u)
{
    maxn = INF;
    getroot(u, 0, s[u]);
    int trt = root;
    vis[trt] = 1;
    dfsgao(trt, 0, trt, 0);
    for(int i = head[trt]; ~i ; i = e[i].nxt)
    {
        int v = e[i].to;
        if(vis[v]) continue;
        dfs(v);
    }
}
bool pvvis[N];
int main()
{
//    Open();
    frd.init();
    memset(head, -1, sizeof(head));
    n = frd.getint();
//    scanf("%d", &n);
    for(int i = 1; i < n; i++){
        int x, y;
        x = frd.getint();
        y = frd.getint();
//        scanf("%d%d", &x, &y);
        addedge(x, y);
        addedge(y, x);
    }
    s[1] = n;
    dfs(1);
    int q;
    q = frd.getint();
    int ndnum = 0;
//    scanf("%d", &q);
    while(q--){
        int op, u;
        op = frd.getint();
        u  = frd.getint();
//        scanf("%d%d", &op, &u);
        if(op == 0){
            col[u] ^= 1;
            if(col[u]) ndnum++;
            else ndnum--;
            if(col[u]){
                //pvvis[u] = 1;
                for(int i = 0; i < wV[u].size(); i ++)
                {
                    int v = wV[u][i].second;
                    int dis = wV[u][i].first;
                    que[v].push(PII(dis, u));
                }
            }
```

```
        }else{
            if(ndnum == 0){
                puts("-1");
                continue;
            }
            int ans = INF;
            for(int i = 0; i < wV[u].size(); i ++){
                int v = wV[u][i].second;
                int dis = wV[u][i].first;
                while(!que[v].empty()){
                    PII pp = que[v].top();
                    int curv = pp.second, dist = pp.first;
                    if(!col[curv]) {que[v].pop(); continue;}
                    ans = min(ans, dist+dis);
                    break;
                }
            }
            if(ans == INF) ans = -1;
            if(ans == -1) while(1);
            printf("%d\n", ans);
        }
    }
    return 0;
}
```

CF Groups 树分治的查询

```
/*
http://codeforces.com/group/qcIqFPYhVr/contest/203881/problem/X
一棵n（n<=1e5）个点的有边权的树，q(q<=1e5)个查询（v，L），求v点周围距离他不超过L的点的个数。
利用树分治， 显然树上的任何一个点，可以达到它的重心不会超过logn个。
对于点v周围的某个距离他L远的u点而言，u的重心要么是v（距离为L），要么是包括v的某个重心x（距离为L-dist(v,x)）。
那么对于每个点v，预处理出以他为中心的子树上的所有的点到他的距离，用vector存下来并排序，同时处理出每个点被包括的所有中心。
空间复杂度为O(nlogn)，时间复杂度为O(nlognlong)。
现在考虑去重，一个点u如果在v的某个重心x上计算一次后，必然还会被包含x的重心再计算一次，所以对于每个点，还要处理出以他为重心时的每一棵子
树对他的贡献，考虑到每个点最多作为一个点的直接子树，所以空间复杂度会再多一倍。
*/
#include<bits/stdc++.h>
#define MAXN 110000
typedef long long LL;
struct edge{
    int to,next,cost;
}E[MAXN*2];
struct Node{
    int root,subtree;
    LL dis;
    Node(int _root = 0, int _subtree = 0,LL _dis = 0){
        root = _root;
        subtree = _subtree;
        dis = _dis;
    }
};
std::vector<Node>vec[MAXN];
```

```
std::vector<LL> vs[MAXN<<1];
bool vis[MAXN];
int fa[MAXN],sz[MAXN],que[MAXN],head[MAXN],si,cnt=0,id[MAXN];
LL dist[MAXN];
void add_edge(int u,int v,int w)
{
    E[si].to=v;
    E[si].next=head[u];
    E[si].cost=w;
    head[u]=si++;
}
int getroot(int root)
{
    int l=0,r=0;fa[root]=0;
    for(que[r++]=root;l<r;l++){
        int v=que[l];
        for(int i=head[v];~i;i=E[i].next){
            int u=E[i].to;
            if(u==fa[v]||vis[u])continue;
            fa[u]=v;
            que[r++]=u;
        }
    }
    int res=MAXN;
    for(;r;r--){
        int v=que[r-1];
        sz[v]=1;
        int m=0;
        for(int i=head[v];~i;i=E[i].next){
            int u=E[i].to;
            if(u==fa[v]||vis[u])continue;
            sz[v]+=sz[u];
            m=std::max(sz[u],m);
        }
        m=std::max(m,l-sz[v]);
        if(m<res){
            res=m;
            root=v;
        }
    }
    return root;
}
void go(int v,int root,int w,int subtree)
{
    int l=0,r=0;fa[v]=root;dist[v]=w;
    vs[subtree].clear();
    for(que[r++]=v;l<r;l++){
        int u=que[l];
        vs[subtree].push_back(dist[u]);
        vs[id[root]].push_back(dist[u]);
        vec[u].push_back(Node(id[root],subtree,dist[u]));
        for(int i=head[u];~i;i=E[i].next){
            int to=E[i].to;
            if(to==fa[u]||vis[to])continue;
            dist[to]=dist[u]+E[i].cost;
            fa[to]=u;
            que[r++]=to;
        }
    }
}
void solve(int root)
```

```
{
    root=getroot(root);
    id[root]=++cnt;
    dist[root]=0;fa[root]=0;
    vis[root]=true;
    vec[root].push_back(Node(id[root],-1,0));
    vs[id[root]].clear();
    vs[id[root]].push_back(0);
    for(int i=head[root];~i;i=E[i].next){
        int u=E[i].to;
        if(vis[u])continue;
        go(u,root,E[i].cost,++cnt);
    }
    for(int i=head[root];~i;i=E[i].next){
        int u=E[i].to;
        if(vis[u])continue;
        solve(u);
    }
}
int query(int v,LL x)
{
    if(x<0)return 0;
    return std::upper_bound(vs[v].begin(),vs[v].end(),x)-
vs[v].begin();
}
int main()
{
    int n,q;
    scanf("%d%d",&n,&q);
    memset(head,-1,sizeof head);si=cnt=0;
    for(int i=1,u,v,w;i<n;i++){
        scanf("%d%d%d",&u,&v,&w);
        add_edge(u,v,w);
        add_edge(v,u,w);
    }
    solve(getroot(1));
    for(int i=1;i<=cnt;i++){
        std::sort(vs[i].begin(),vs[i].end());
    }
    LL L;
    for(int i=0,v;i<q;i++){
        scanf("%d%I64d",&v,&L);
        int ans=0;
        for(int j=0;j<vec[v].size();j++){
            Node tmp=vec[v][j];
            ans+=query(tmp.root,L-tmp.dis);
            if(~tmp.subtree)ans-=query(tmp.subtree,L-tmp.dis);
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

数论

CF#259D fwt 快速沃尔什变换

```
//详细解释在Onenote上有
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long LL;
const int N = (1 << 20) + 10;
int m, n;
LL t, mod;
int b[22];
LL A[N], B[N];
LL aa[N];
int count(int s) {
    int ret = 0;
    while (s) s -= s & -s, ++ret;
    return ret;
}
LL mul(LL a, LL b, LL p) {
    return (a * b - (LL)((long double)a / p * b + 1e-3) * p + p) % p;
}
void arrMul(LL *c, LL *a, LL *b) {
    for (int i = 0; i < n; ++i)
        c[i] = mul(a[i], b[i], mod);
}
void FWT(LL *a, int n) {
    if (n == 1) return;
    int m = n >> 1;
    FWT(a, m);
    FWT(a + m, m);
    for (int i = 0; i < m; ++i) {
        LL u = a[i], v = a[i + m];
        a[i] = (u + v) % mod;
        a[i + m] = (u - v + mod) % mod;
    }
}
void dFWT(LL *a, int n) {
    if (n == 1) return;
    int m = n >> 1;
    for (int i = 0; i < m; ++i) {
        LL u = a[i], v = a[i + m];
        a[i] = (u + v) % mod;
        a[i + m] = (u - v + mod) % mod;
    }
    dFWT(a, m);
    dFWT(a + m, m);
}
void Conv(LL *A, LL *B, int n, int t)//t -> 运算次数，n -> 运算长度，B
-> 为转换数组，A -> 结果/初值数组。
{
    FWT(A, n);
    FWT(B, n);
    for (; t; t >>= 1, arrMul(B, B, B)) //类似快速幂
        if (t & 1) arrMul(A, A, B);
    dFWT(A, n);
```

```
}
int main() {
    cin >> m >> t >> mod;
    n = 1 << m;
    mod *= n;
    for (int i = 0; i < n; ++i) {
        scanf("%I64d", A + i);
        A[i] %= mod;
    }
    for (int i = 0; i <= m; ++i) {
        scanf("%d", b + i);
        b[i] %= mod;
    }
    for (int i = 0; i < n; ++i) {
        B[i] = b[count(i)];
    }

    Conv(A, B, n, t);
    for (int i = 0; i < n; ++i) {
        printf("%I64d\n", A[i] >> m);
    }
}
```

Hiho1230-FWT 快速沃尔什变换

```
/*
*       dp[i][j]=sigma(dp[i-1][k]*b[f(k^j)]
*
*   这个吊模板就是用来解决上述式子的，更为标准的是CF#259div1D那个题中的表达式。
*   可以很快速的求解dp[n]的每一项。复杂度为m*log(n)，m为第二维大小
*
*   这个题，我的想法是这样的：首先枚举X，那么也就是需要在[x, m+x]中找出n个数，使得这n个数异或和为0，
*   然后方案数加起来即可。那么对于每个枚举的X来说，我们记dp[i][j]表示前i个数，异或和为j的方案数，
*   那么dp[i][j]=sigma(dp[i-1][k]*b[k^j])。其中b[i]=1当且仅当x <= i <=x+m，否则为0；那么上述式子其实
*   和上一题CF的题就没多大区别了。然后就直接套用fwt，当然这里需要注意的是模数必须乘上做fwt的那个数组
*   的大小len(这里的len也和NTT,FFT一样必须是大于等于N的最小二次幂数)。最后答案也需要除以这个len。
*   下面的解法中，直接用a[i]数组pow_mod,我认为是因为dp[0][j]的初始值都与b[i]数组相同；
*   验证发现的确两种方法都可以AC。
*/
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
```

```
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
const LL basemod = 1000000007;
LL mod = basemod;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
//防爆long long mul_mod
LL mul(LL a, LL b, LL p) {
    return (a * b - (LL)((long double)a / p * b + 1e-3) * p + p) % p;
}
void arrMul(LL n, LL *c, LL *a, LL *b) {
    for (int i = 0; i < n; ++i)
        c[i] = mul(a[i], b[i], mod);
}
LL pow_mod(LL x, LL k, LL mod)
{
    LL res = 1;
    while(k)
    {
        if( k & 1) res = mul(res, x, mod);
        x = mul(x, x, mod);
        k >>= 1;
    }
    return res;
}
void FWT(LL *a, int n) {
    if (n == 1) return;
    int m = n >> 1;
    FWT(a, m);
    FWT(a + m, m);
    for (int i = 0; i < m; ++i) {
        LL u = a[i], v = a[i + m];
        a[i] = (u + v) % mod;
        a[i + m] = (u - v + mod) % mod;
    }
}
void dFWT(LL *a, int n) {
    if (n == 1) return;
    int m = n >> 1;
    for (int i = 0; i < m; ++i) {
        LL u = a[i], v = a[i + m];
        a[i] = (u + v) % mod;
        a[i + m] = (u - v + mod) % mod;
    }
    dFWT(a, m);
    dFWT(a + m, m);
}
void Conv(LL *A, LL *B, int n, int t)//t -> 运算次数，n -> 运算长度，B
-> 为转换数组，A -> 结果/初值数组。
{
    FWT(A, n);
    FWT(B, n);
```

```
    for (; t; t >>= 1, arrMul(n, B, B, B)) //类似快速幂
        if (t & 1) arrMul(n, A, A, B);
    dFWT(A, n);
}
LL a[1<<12], b[1<<12];
LL solve(int n, int m, int L, int R)
{
    LL len = 1;
    while(len <= R) len *= 2;
    mod = basemod * len;
    for(int i = 0; i < len ;i++)
        a[i] = b[i] = (i >= L && i <= R);
//    FWT(a, len);
//    for(int i = 0; i < len; i++)
//        a[i] = pow_mod(a[i], 2 * n + 1, mod);
//    dFWT(a, len);
    Conv(a, b, len, 2*n);
    return (a[0] + mod)%mod/len;
}
int main()
{
    Open();
    int n, m, L, R;
    while(~scanf("%d%d%d%d", &n, &m, &L, &R))
    {
        LL ans = 0;
        for(int i = L; i <= R; i++)
            ans = (ans + solve(n, m, i, i+m))%basemod;
        printf("%lld\n", ans);
    }
    return 0;
}
```

CRT 求 bell 数

```
/*
* 题意：求bell[n]%m，n < 1<<31;
* 做法：由于给出的m是一个可分解为幂次全为1的质因数相乘的模数，所以可以用CRT去弄，再加上这个公式：
*       bell[n+p]%p = (bell[n+1]+bell[n])%p;这样就可以用矩阵快速幂求出结果，再让CRT去合并即可。
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 55
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
```

```
typedef long long LL;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
void Open()
{
#ifndef ONLINE_JUDGE
    freopen("F:/in.txt","r",stdin);
    //freopen("F:/my.txt","w",stdout);
#endif // ONLINE_JUDGE
}
const LL mod[5] = {31, 37, 41, 43, 47};
void mul(LL A[N][N], LL B[N][N], LL t[N][N], LL n, LL m, LL l, LL id)
{
    LL tmp[N][N];
    for(LL i = 0; i < n; i++)
        for(LL j = 0; j < l; j++)
        {
            tmp[i][j] = 0;
            for(LL k = 0; k < m; k++)
                tmp[i][j] = (tmp[i][j] + A[i][k] * B[k][j])%mod[id];
        }
    for(LL i = 0; i < n; i++)
        for(LL j = 0; j < l; j++)
            t[i][j] = tmp[i][j];
}
void expo(LL p[N][N], LL e[N][N], LL k, LL n, LL id)
{
    for(LL i = 0; i < n; i++)
        for(LL j = 0; j < n; j++)
            e[i][j] = (i == j);
    while(k)
    {
        if(k & 1) mul(e, p, e, n, n, n, id);
        mul(p, p, p, n, n, n, id);
        k >>= 1;
    }
}
LL a[N][N], b[N][N], c[N][N], res[N];
LL s[N][N];
LL init(LL id, LL n)
{
    LL p = mod[id];
    for (int i = 0; i < p; ++i)
    {
        s[i][0] = s[i][i] = 1;
        for (int j = 1; j < i; ++j)
            s[i][j] = (s[i-1][j] + s[i - 1][j - 1])%p;
    }
    memset(a, 0, sizeof(a));
    a[0][0] = 1;
    a[0][1] = 1;
    for (int i = 2; i < p; ++i)
    {
        a[0][i] = 0;
        for (int j = 0; j < i; ++j)
            a[0][i] = (a[0][i] + s[i - 1][j] * a[0][j])%p;
    }
    if(n < p) return a[0][n];
    memset(b, 0, sizeof(b));
    memset(c, 0, sizeof(c));
    for(LL i = 0; i < p-1; i++)
```

```
        b[i][i] = b[i+1][i] = 1;
    b[p-1][p-1] = b[0][p-1] = b[1][p-1] = 1;
    expo(b, c, n/p, p, id);
    mul(a, c, a, 1, p, p, id);
    return a[0][n%p]%p;
}
void gcd(LL a, LL b, LL& d, LL& x, LL& y)
{
    if(!b)
    {
        d = a;
        x = 1;
        y = 0;
    }
    else
    {
        gcd(b, a%b, d, y, x);
        y -= x * (a / b);
    }
}
LL china(LL n, LL *a, const LL *m)
{
    LL M = 1, d, y, x = 0;
    for(LL i = 0; i < n; i++) M *= m[i];
    for(LL i = 0; i < n; i++)
    {
        LL w = M / m[i];
        gcd(m[i], w, d, d, y);
        x = (x + y*w%M*a[i]%M)%M;
    }
    return (x + M) % M;
}
int main()
{
//    Open();
    LL T;
    scanf("%I64d", &T);
    while(T--)
    {
        LL n;
        scanf("%I64d", &n);
        for(LL i = 0; i < 5; i++)
            res[i] = init(i, n);
        LL ans = china(5, res, mod);
        printf("%I64d\n", ans);
    }
    return 0;
}
```

二维递推式，化无规律递推为等差+等比

```
/*
*   Problem Description
As we know, sequence in the form of an=a1+(n?1)d is called arithmetic
progression and sequence in the form of bn=b1qn?1(q>1,b1≠0) is called
geometric progression. Huazheng wants to use these two simple
sequences to generate a simple matrix. Here is what he decides to do:
Use the geometric progression as the first row of the simple matrix:
c[0,n]=bn
```

Use the arithmetic progression as the first column of the simple matrix: c[n,0]=an
Calculate the item at n-th row, m-th column of the simple matrix as c[n,m]=c[n?1,m]+c[n,m?1], where n≥1 and m≥1.
Given the two sequences, Huazheng wants to know the value of cn,m, but he is too busy with his research to figure it out. Please help him to work it out. By the way, you can assume that c0,0=0.

Input
The first line of input contains a number T indicating the number of test cases (T≤200).
For each test case, there is only one line containing six non-negative integers b1,q,a1,d,n,m. (0≤n≤10000). All integers are less than 231.

Output
For each test case, output a single line consisting of "Case #X: Y". X is the test case number starting from 1. Y is cn,m module 1000000007.

```
*
*   这个题极其恶心...简易题解链接：http://talk.icpc-camp.org/d/84-2015-g-simple-matrix/2
*   由于n小于10000,对于等差数列那一项我们可以利用链接中的公式直接暴力求和，但是m的范围及其大，等比数列那边不能
*   (sum = (1<=i<=n)C(n+m-i-1, n-i)*a[i] + (1<=i<=m)C(n+m-i-1, m-i)*b[i])
*   我们考虑简化等比那边的计算方法，第0行为等比数列，那么第一行为等比数列前n项和，通项公式化开之后发现其实新的
*   数列格式为一个等比+常数。那么我们可以将每一行的常数丢给a[i]去处理，这样只剩下一行等比数列，继续往下递归。
*   这里蛋疼的地方在于每次扔出来的这个常数并不是整数。。可能是分数，但是最终结果一定是整数，而且得取模，炒鸡蛋疼
*   我这里中途一直维护了分子和分母，直到最终一行的等比数列算出来之后再加上去，此时结果一定为整数，在对分母取逆元
*   乘上去即可。
*   常数的格式为：            -b1*q^(i-1)/(q-1)^i
*   每一行等比数列首项为：     b1*q^(i)  /(q-1)^i
*/

#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 100010
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef long long LL;
typedef pair<LL,LL> PII;
```

```
const LL INF=0x3f3f3f3f;
const LL mod = 1000000007;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
LL a[N];
LL pow_mod(LL x, LL k, LL mod)
{
    LL res = 1;
    while(k)
    {
        if(k & 1) res = res * x % mod;
        x = x * x % mod;
        k >>= 1;
    }
    return res;
}
int main()
{
//    Open();
    LL T;scanf("%I64d", &T);LL cas = 1;
    while(T--)
    {
        LL b1, q, a1, d, n, m;
        scanf("%I64d%I64d%I64d%I64d%I64d%I64d", &b1, &q, &a1, &d, &n,
&m);
        a[1] = a1;
        for(LL i = 2; i <= n; i++){
            a[i] = (a[i-1] + d)%mod;
        }
        LL ans = a[n];
        LL C = 1;
        for(LL i = n-1; i >= 1; i--)
        {
            C = C * (n + m - i - 1) % mod * pow_mod(n - i, mod - 2,
mod)%mod;
            ans = (ans + a[i] * C)%mod;
        }

        LL zi = 0;
        LL mu = 1;
        LL qpow = 1;
        for(LL i = 1; i < n; i++){
            mu = mu * (q-1) % mod;
            zi = (zi * (q-1) % mod - C * qpow % mod * b1 % mod) % mod;
            qpow = qpow * q % mod;
            C = C * (n - i) %mod * pow_mod(n+m-i-1, mod-2, mod)%mod;
        }
        mu = mu * (q-1)%mod;
        zi = zi * (q-1)%mod;//n次
        zi = (zi + b1 * pow_mod(q, n-1, mod) %mod * (pow_mod(q, m,
mod) - 1) % mod) %mod;
        ans = (ans + zi * pow_mod(mu, mod-2, mod)%mod)%mod;
        if(ans < 0) ans += mod;
        printf("Case #%I64d: %I64d\n", cas++, ans%mod);
    }
    return 0;
```

```
}
```

图

最小费用最大流

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 2020
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<long long,long long> PII;
const int INF=0x3f3f3f3f;
const long int mod = 1000000007;
void Open()
{
    #ifndef ONLINE_JUDGE
        freopen("D:/in.txt","r",stdin);
        //freopen("D:/my.txt","w",stdout);
    #endif // ONLINE_JUDGE
}
struct node
{
    int h, d;
    bool operator<(const node& o)const{
        return h > o.h || (h == o.h && d < o.d);
    }
}app[N];

struct Edge{
    int u, v, cap, flow, cost;
};
int cas = 1;
struct MCMF{
    int n, m, s, t;
    vector<Edge> edges;
    vector<int> G[N];
    int inq[N];
    int d[N];
    int p[N];
    int a[N];

    void init(int n, int s, int t) {
        this -> n = n;
        this -> s = s;
        this -> t = t;
        for(int i = 0; i<n; i++) G[i].clear();
```

```
        edges.clear();
    }
    void addedge(int u, int v, int cap, int cost){
        edges.push_back((Edge){u, v, cap, 0, cost});
        edges.push_back((Edge){v, u, 0, 0, -cost});
        m = edges.size();
        G[u].push_back(m-2);
        G[v].push_back(m-1);
//        cerr<<u<<" -> "<<v<<" : "<<cost<<endl;
    }
    bool SPFA(int& flow, int& cost){
        for(int i=0;i<n;i++) d[i] = INF, inq[i] = 0;
        d[s] = 0; inq[s] = 1; p[s] = 0; a[s] = INF;
        queue<int> Q;
        Q.push(s);
        while(!Q.empty()){
            int u = Q.front(); Q.pop();
            inq[u] = 0;
            for(int i = 0; i < G[u].size(); i++) {
                Edge& e = edges[G[u][i]];
                if(e.cap > e.flow && d[e.v] > d[u] + e.cost){
                    d[e.v] = d[u] + e.cost;
                    p[e.v] = G[u][i];
                    a[e.v] = min(a[u], e.cap - e.flow);
                    if(!inq[e.v]) {Q.push(e.v); inq[e.v] = 1;}
                }
            }
        }
        if(d[t] >= 0) return false;
        flow += a[t];
        cost += d[t];
        int u = t;
        while(u != s){
            edges[p[u]].flow += a[t];
            edges[p[u]^1].flow -= a[t];
            u = edges[p[u]].u;
        }
        return true;
    }
    int Mincost() {
        int flow = 0, cost = 0;
        while(SPFA(flow, cost));
        return cost;
    }
}mcmf;

template <class T>
void read(T &x) {
    for (++ch; *ch <= 32; ++ch);
    for (x = 0; '0' <= *ch; ch++)   x = x * 10 + *ch - '0';
}

int main()
{
    Open();
    ch = buf - 1;
    fread(buf, 1, 1000 * 35 * 1024, stdin);

    long long T;
    read(T);
//    scanf("%I64d", &T);
```

```
    while(T--)
    {
        cas++;
        int n;
        read(n);
//        scanf("%d", &n);
        for(int i = 0; i < n; i++){
            read(app[i].h); read(app[i].d);
//            scanf("%d%d", &app[i].h, &app[i].d);
        }
        sort(app, app+n);
        int s = 2*n, t = 2*n+1, ss = 2*n+2;
        mcmf.init(2*n+3, s, t);
        for(int i = 0; i < n; i++) mcmf.addedge(i, n+i, 1, -1),
mcmf.addedge(i, n+i, 1, 0);//由于下面的剪枝，这里必须加一条边使得该流不花费费
用(不吃这个苹果)经过这个点
        for(int i = 0; i < n; i++){
            int pre = INF;
            for(int j = i + 1; j < n; j++)
            {
                if(app[i].d <= app[j].d && app[j].d < pre)
                    mcmf.addedge(i+n, j, 2, 0), pre = app[j].d;
            }
        }
        mcmf.addedge(s, ss, 2, 0);
        for(int i = 0; i < n; i++){
            mcmf.addedge(ss, i, 2, 0);
            mcmf.addedge(i+n, t, 2, 0);
        }
        write(-mcmf.Mincost());
        puts("");
//        printf("%d\n", -mcmf.Mincost());
    }
    return 0;
}
```

Xian'C 最大密度子图，二分+最小割

```
#include <iostream>
#include <cstdio>
#include <stack>
#include <cstring>
#include <queue>
#include <algorithm>
#include <cmath>
//#include <unordered_map>
#define N 222
//#define lson x<<1
//#define rson x<<1|1
//#define mid ((lt[x].l+lt[x].r)/2)
//#define ID(x, y) ((x)*m+(y))
//#define CHECK(x, y) ((x)>=0 && (x)<n && (y)>=0 && (y)<m)
using namespace std;
typedef pair<int,int> PII;
const int INF=0x3f3f3f3f;
const double eps = 1e-9;
void Open()
{
#ifndef ONLINE_JUDGE
```

```
    freopen("D:/in.txt","r",stdin);
    //freopen("D:/my.txt","w",stdout);
#endif // ONLINE_JUDGE
}
int dcmp(double x)
{
    if(fabs(x) < eps) return 0;
    return x > eps;
}
struct Edge
{
    int from, to;
    double cap, flow;
};
struct ISAP
{
    int n, m, s, t;
    vector<Edge> edges;
    vector<int> G[N];   // 邻接表，G[i][j]表示结点i的第j条边在e数组中的序号
    bool vis[N];        // BFS使用
    int d[N];           // 从起点到i的距离
    int cur[N];         // 当前弧指针
    int p[N];           // 可增广路上的上一条弧
    int num[N];         // 距离标号计数

    void AddEdge(int from, int to, double cap)
    {
        edges.push_back((Edge)
        {
            from, to, cap, 0
        });
        edges.push_back((Edge)
        {
            to, from, 0, 0
        });
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool BFS()
    {
        memset(vis, 0, sizeof(vis));
        queue<int> Q;
        Q.push(t);
        vis[t] = 1;
        d[t] = 0;
        while(!Q.empty())
        {
            int x = Q.front();
            Q.pop();
            for(int i = 0; i < G[x].size(); i++)
            {
                Edge& e = edges[G[x][i]^1];
                if(!vis[e.from] && dcmp(e.cap - e.flow) > 0)
                {
                    vis[e.from] = 1;
                    d[e.from] = d[x] + 1;
                    Q.push(e.from);
                }
```

```
            }
        }
        return vis[s];
    }

    void ClearAll(int n)
    {
        this->n = n;
        for(int i = 0; i < n; i++) G[i].clear();
        edges.clear();
    }

    void ClearFlow()
    {
        for(int i = 0; i < edges.size(); i++) edges[i].flow = 0;
    }

    double Augment()
    {
        int x = t;
        double a = INF;
        while(x != s)
        {
            Edge& e = edges[p[x]];
            a = min(a, e.cap-e.flow);
            x = edges[p[x]].from;
        }
        x = t;
        while(x != s)
        {
            edges[p[x]].flow += a;
            edges[p[x]^1].flow -= a;
            x = edges[p[x]].from;
        }
        return a;
    }

    double Maxflow(int s, int t)
    {
        this->s = s;
        this->t = t;
        double flow = 0;
        BFS();
        memset(num, 0, sizeof(num));
        for(int i = 0; i < n; i++) num[d[i]]++;
        int x = s;
        memset(cur, 0, sizeof(cur));
        while(d[s] < n)
        {
            if(x == t)
            {
                flow += Augment();
                x = s;
            }
            int ok = 0;
            for(int i = cur[x]; i < G[x].size(); i++)
            {
                Edge& e = edges[G[x][i]];
                if(dcmp(e.cap - e.flow) > 0 && d[x] == d[e.to] + 1)   //
Advance
                {
```

```
                ok = 1;
                p[e.to] = G[x][i];
                cur[x] = i; // 注意
                x = e.to;
                break;
            }
        }
        if(!ok)   // Retreat
        {
            int m = n-1; // 初值注意
            for(int i = 0; i < G[x].size(); i++)
            {
                Edge& e = edges[G[x][i]];
                if(dcmp(e.cap - e.flow) > 0) m = min(m, d[e.to]);
            }
            if(--num[d[x]] == 0) break;
            num[d[x] = m+1]++;
            cur[x] = 0; // 注意
            if(x != s) x = edges[p[x]].from;
        }
    }
    return flow;
}

vector<int> Mincut()   // call this after maxflow
{
    BFS();
    vector<int> ans;
    for(int i = 0; i < edges.size(); i++)
    {
        Edge& e = edges[i];
        if(!vis[e.from] && vis[e.to] && e.cap > 0)
ans.push_back(i);
    }
    return ans;
}

void Reduce()
{
    for(int i = 0; i < edges.size(); i++) edges[i].cap -=
edges[i].flow;
}

void print()
{
    printf("Graph:\n");
    for(int i = 0; i < edges.size(); i++)
        printf("%d->%d, %d, %d\n", edges[i].from, edges[i].to ,
edges[i].cap, edges[i].flow);
}
};


ISAP mf;

int U;
int n, a[N];
int deg[N];
int source, sink;
int check(double g)
{
```

```
    mf.ClearAll(n+2);
    memset(deg, 0, sizeof(deg));
    for(int i=1; i<=n; i++)
        for(int j = i+1; j <= n; j++)
            if(a[j] < a[i]) mf.AddEdge(i, j, 1), mf.AddEdge(j, i, 1),
deg[i]++, deg[j]++;
    for(int i=1; i<=n; i++)
        mf.AddEdge(source, i, U);
    for(int i=1; i<=n; i++)
        mf.AddEdge(i, sink, g*2.0 + 1.0*U - 1.0*deg[i]);
    double curf = mf.Maxflow(source, sink);
    return dcmp((U*n - curf)/2.0);
}
int main()
{
    Open();
    int T;
    scanf("%d", &T);
    int cas = 1;
    while(T--)
    {
        scanf("%d", &n);
        for(int i=1; i<=n; i++)
            scanf("%d", &a[i]);
        sink = n+1, source = 0;
        U = n+3;
        double lb = 0, ub = n*n;
//        int tmp = check(6.25);
        while(lb + eps < ub)
        {
            double mid = (lb + ub) / 2;
            int tmp = check(mid);
            if(tmp > 0) lb = mid;
            else ub = mid;
        }
        printf("Case #%d: %.12f\n", cas++, lb+eps);
    }
    return 0;
}
```