生日礼物(present,1s,512MB)

【算法分析】

算法一

考虑设$g_{x,y}$表示第x行第y个灯泡（以下记作灯泡(x,y)）的开关被按了几次，只有0,1两种取值。考虑灯泡(x_0,y_0)最终是亮的的条件，一个灯泡的亮暗不仅与其本身的开关有关，还与它周围八个方向（满足$|x-x_0|=1,|y-y_0|=2$或$|x-x_0|=1,|y-y_0|=2$）的灯泡开关有关，因此g_{x_0,y_0}和周围八个方向的灯泡的g的异或和为1。

对每个灯泡我们都列出这样一个异或方程，最后会得到一个含有$n\times m$个变量的异或方程组，用高斯消元求出该方程组的自由元个数t，则最后的答案就为2^t，恰好对应了每一组合法的解。

解异或方程组可以用 bitset 优化，时间复杂度$O(\frac{n^3m^3}{64})$，期望得分40分。

算法二

现在的主要问题是变量过多，导致解方程的复杂度难以承受。

考虑缩减变量的规模，如果我们确定了前两行和第一列的g的取值，那么下图中x所在位置g的取值已经确定，因为x的位置为(3,2)，而限制灯泡(1,1)是亮的的方程中的其它量都是已知的，我们可以反推出x所在位置g的取值。

*	*	*	*	*	*
*	*	*	*	*	*
*	x	.	.	.	.
*	.	.	.	.	.

同样地，我们按照行大小为第一关键字、列大小为第二关键字，依次遍历每个未确定g的取值的位置(i,j)，此时限制灯泡$(i-2,j-1)$是亮的的方程中的其它量都是已知的，我们可以反推出$g_{i,j}$。

因此，我们只需要设前两行和第一列的g为变量，变量个数只有$O(n+m)$的级别。

因为最后两行和最后一列的灯泡还没有被限制必须是亮的，我们同样对每个灯泡列出一个异或方程，恰好和变量的个数一样，同样用 bitset 优化高斯消元解异或方程组即可。

时间复杂度$O(\frac{(n+m)^3}{64})$，期望得分100分。

【参考程序】//陈贤

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

usingstd::bitset;
constint N = 605;
constint M = 1805;
constint dx[] = {1, 1, -1, -1, 2, 2, -2, -2},
      dy[] = {2, -2, 2, -2, 1, -1, 1, -1};
constint mod = 123456789;
bitset<M> f[M], a[N][N];
int n, m, G, nG, ans = 1, b[N][N];
bool vis[M];

template<class T>
inline T Max(T x, T y) {return x > y ? x : y;}
template<class T>
inline T Min(T x, T y) {return x < y ? x : y;}

inline void add(int&x, int y)
{
   x += y;
   x >= mod ? x -= mod : 0;
}

inline void solve(int x, int y)
{ //列出限制灯泡 (x,y) 是亮的的方程
   f[++nG] = a[x][y];
```

```
    f[nG][G + 1] = !f[nG][G + 1];
    for (int k = 0; k < 8; ++k)
    {
        inttx = x + dx[k],
            ty = y + dy[k];
        if (tx< 1 || tx> n || ty < 1 || ty > m)
            continue ;
        f[nG] ^= a[tx][ty];
    }
}

inline void Gauss()
{ //用 bitset 优化高斯消元解异或方程组
    for (inti = 1; i<= G; ++i)
    {
        int p = 0;
        for (int j = 1; j <= G; ++j)
            if (!vis[j] && f[j][i])
            {
                p = j;
                break ;
            }
        if (!p)
        {
            add(ans, ans);
            continue ;
        }
```

```
        vis[p] = true;
        for (int j = 1; j <= G; ++j)
            if (!vis[j] && f[j][i])
                f[j] ^= f[p];
    }
}

int main()
{
    freopen("present.in", "r", stdin);
    freopen("present.out", "w", stdout);

    scanf("%d%d", &n, &m);
    for (inti = 1, im = Min(n, 2); i<= im; ++i)
        for (int j = 1; j <= m; ++j)
            a[i][j][++G] = 1;
    for (inti = 3; i<= n; ++i)
        a[i][1][++G] = 1;
    for (inti = 3; i<= n; ++i)
        for (int j = 2; j <= m; ++j)
        { //用前两行和第一列的 g 表示其它位置的 g
            int x = i - 2, y = j - 1;
            a[i][j] = a[x][y];
            a[i][j][G + 1] = !a[i][j][G + 1];
            for (int k = 0; k < 8; ++k)
            {
                if (k == 4)
```

```
                    continue ;
                inttx = x + dx[k],
                    ty = y + dy[k];
                if (tx< 1 || tx> n || ty < 1 || ty > m)
                    continue ;
                a[i][j] ^= a[tx][ty];
            }
        }
    for (inti = Max(n - 1, 1); i<= n; ++i)
        for (int j = 1; j <= m; ++j)
            solve(i, j);
    for (inti = n - 2; i>= 1; --i)
        solve(i, m);
    Gauss();
    printf("%d\n", ans);

    fclose(stdin); fclose(stdout);
    return 0;
}
```

随机游走(average,1.5s,512MB)

【算法分析】

记$f[x][y]$表示$S = x, T = y$时的答案，边界$\forall 1 \le x \le n, f[x][y] = 0$。

设无向图的边集为E，点x的度数为deg_x，边(x, y)的长度为$g(x, y)$。考虑转移，分4种情况：

1. 两个人都停在原地：$f[x][y] += p_x \times p_y \times f[x][y]$
2. 小A停在原地，小B移动：$f[x][y] += \sum_{(v,y)\in E} p_x \times \frac{1-p_y}{deg_y} \times (f[x][v] + g(v, y))$
3. 小B停在原地，小A移动：$f[x][y] += \sum_{(u,x)\in E} p_y \times \frac{1-p_x}{deg_x} \times (f[u][y] + g(u, x))$
4. 两人同时移动：$f[x][y] += \sum_{(u,x)\in E} \sum_{(v,y)\in E} \frac{1-p_x}{deg_x} \times \frac{1-p_y}{deg_y} \times \left(f[u][v] + g(u, x) + g(v, y)\right)$

发现转移成环，考虑用高斯消元解决。发现无向图的边是不变的，但是边权会变。那么我们把边权看作方程中的未知数，枚举所有的x,y，用$O(n^2)$个$f[i][j]$和$O(n^2)$个$g(i,j)$表示每个$f[x][y]$。这样就得到了一个含$O(2n^2)$个未知数和$O(n^2)$个等式的方程组。

设系数矩阵A，$A[i][j]$表示第i个方程中第j个未知数的系数。我们令前$O(n^2)$个变量为f，后$O(n^2)$个变量为g。然后利用高斯消元，将A消成这样的形式：$\forall 1 \le i \le n^2, A[i][i] = 1$且$\forall 1 \le i,j \le n^2, j \ne i, A[i][j] = 0$。

这样我们就能用只含边权的式子表示每个$f[x][y]$了。那么对于每个询问，我们只要把给出的边权代入这个式子即可。

时间复杂度$O(n^6 + Tn^2)$。

【参考程序】//陈予菲

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

usingnamespace std;

#define ld longdouble

constint e = 25, o = 2 * e * e;
const ld eps = 1e-14;

int n, m, cnt[e][e], q, tot, L, R;
ld a[o][o], p[e], b[o], deg[e];
vector<int>g[e];
struct point
{
    int x, y;
    double z;
}c[o];

inlineint getid(int x, int y)
{
    if (x > y) swap(x, y);
    return n * n + cnt[x][y];
}

inlinevoid gauss(int n, int m) // 高斯消元
{
    int i, j, k;
    for(i = 1; i <= n; i++)
    {
        int x = 0;
        for (j = i; j <= n; j++)
            if (fabs(a[j][i]) > eps)
            {
                x = j;
                break;
            }
        if (x != i)
        {
            for(j = i; j <= m; j++)
                swap(a[x][j], a[i][j]);
            swap(b[x], b[i]);
```

```
        }
        for (j = 1; j <= n; j++)
            if (j != i)
            {
                ld v = a[j][i] / a[i][i];
                for (k = i; k <= m; k++)
                    a[j][k] -= a[i][k] * v;
                b[j] -= b[i] * v;
            }
    }
}
```

```
int main()
{
    freopen("average.in", "r", stdin);
    freopen("average.out", "w", stdout);
    scanf("%d%d%d", &n, &m, &q);
    int i, j, x, y, u, v;
    double z;
    for (i = 1; i <= m; i++)
    {
        scanf("%d%d", &x, &y);
        c[i].x = x; c[i].y = y;
        g[x].push_back(y);
        g[y].push_back(x);
        deg[x] += 1.0;
        deg[y] += 1.0;
    }
    for(i = 1; i < n; i++)
        for (j = i + 1; j <= n; j++)
            cnt[i][j] = ++cnt[0][0];
```

```
L = n * n + 1; R = L + cnt[0][0] - 1;
for (i = 1; i <= n; i++)
    scanf("%lf", &z), p[i] = z;
tot = n * n + n * (n - 1) / 2;
for (x = 1; x <= n; x++)
{
    int lenx = g[x].size();
    for(y = 1; y <= n; y++)
        if (x != y)
        {
            int leny = g[y].size(), pos = (x - 1) * n + y;
            a[pos][pos] = p[x] * p[y] - 1.0;//两个人都停在原地
            for (i = 0; i < lenx; i++)//小 B 停在原地，小 A 移动
            {
                int u = g[x][i];
                ld val = p[y] * (1.0 - p[x]) / deg[x];
                a[pos][(u - 1) * n + y] += val;
                a[pos][getid(u, x)] += val;
            }
            for (j = 0; j < leny; j++)//小 A 停在原地，小 B 移动
            {
                int v = g[y][j];
                ld val = p[x] * (1.0 - p[y]) / deg[y];
                a[pos][(x - 1) * n + v] += val;
                a[pos][getid(v, y)] += val;
            }
            for (i = 0; i < lenx; i++) //两个人同时移动
            {
              int u = g[x][i];
              for(j = 0; j < leny; j++)
              {
                int v = g[y][j];
                 ld val = (1.0-p[x])*(1.0-p[y])/deg[x]/deg[y];
                a[pos][(u - 1) * n + v] += val;
                a[pos][getid(x, u)] += val;
                a[pos][getid(y, v)] += val;
              }
            }
        }
}
for (i = 1; i <= n; i++)
{
    int pos = (i - 1) * n + i;
    a[pos][pos] = 1.0;
}
gauss(n * n, tot);
while (q--)
{
    for (i = 1; i <= m; i++)
        scanf("%lf", &c[i].z);
    scanf("%d%d", &x, &y);
    int pos = (x - 1) * n + y;
```

```
        ld ans = b[pos];
        for(i = 1; i <= m; i++)
            ans -= a[pos][getid(c[i].x, c[i].y)] * c[i].z;
        ans /= a[pos][pos];
        double res = ans;
        printf("%.2lf\n", res);
    }
    fclose(stdin); fclose(stdout);
    return0;
}
```

带权图（graph, 1s, 512MB）

【算法分析】

算法一：

一个首要的问题是图中的环可能有很多，我们没有办法全部枚举。但是我们的要求只是求出m条边的所有C权值，因此就考虑怎么恰当地利用条件三。

考虑原图的一棵生成树T。对于每条非树边(u,v)，找到u,v在生成树上的路径，那么 $u \to v \dashrightarrow u$（实线表示沿着树边走，虚线表示沿着非树边走）就是一个满足题目第三个条件要求的环，对于每条非树边都这么找路径，这样我们可以列出$m-n+1$个方程。因为正反边权和为 0，因此图中每个环的边权和都可以表示为这些环中的几个的和。

接着考虑对于每个点，用条件二列出n个方程。我们考虑这n个方程的和，我们发现每一条边正向和反向恰好都被加了一次，那么实际上这n个方程的和为0，也就是说其中一个方程肯定是没用的，实际上有用的只有$n-1$个方程，我们任选这n个中的$n-1$个即可。

那么这样我们就得到了m个方程，并且这m个方程能表示所有条件。因此只要用高斯消元解这m个方程即可。时间复杂度$O(m^3)$。

算法二：

在下面的讲解中，我们忽略题目条件中的模运算，直接用等号表示条件。

算法一的主要瓶颈在于方程的变量过多，我们尝试继续挖掘一个环的性质：

$$\sum_{i=0}^{n-1} B(v_i,v_{i+1}) \cdot C(v_i,v_{i+1}) = \sum_{i=0}^{n-1} A(v_i,v_{i+1})$$

这个式子实际上可以写成

$$\sum_{i=0}^{n-1} B(v_i,v_{i+1}) \cdot C(v_i,v_{i+1}) - A(v_i,v_{i+1}) = 0$$

如果我们定义边权$D(u,v) = B(u,v) \cdot C(u,v) - A(u,v)$，这样我们可以根据$D$来计算出$C$。那么上面的限制实际上相当于每个环上的边权$D$之和为0。我们考虑环上的两个点$x,y$，那么整个环可以拆解为两段：$x \xrightarrow{1} y \xrightarrow{2} x$。那么就有

$$\sum_{(u,v)\in x\overset{1}{\to}y} D(u,v) + \sum_{(u,v)\in y\overset{2}{\to}x} D(u,v) = 0$$

我们发现$D(u,v) = -D(v,u)$，因此

$$\sum_{(u,v)\in x\overset{1}{\to}y} D(u,v) = \sum_{(u,v)\in x\overset{2}{\to}y} D(u,v)$$

也就是说，任意两条$x \to y$的路径的边权D的和是相同的。因此对于一个点u，所有$1 \to u$的路径的边权D之和都一样。那么我们不妨设$\phi(u)$表示任意一条$1 \to u$的路径的D之和。并且对于任意一条边$(u,v) \in E$，我们可以直接得到

$$D(u,v) = \phi(v) - \phi(u)$$

那么根据第二个条件，我们就可以列出$n-1$个方程（其中我们直接令$\phi(1) = 0$）。其中利用点u列出的方程为

$$\sum_{(u,v)\in E} \frac{\phi(v) - \phi(u) + A(u,v)}{B(u,v)} = 0$$

那么直接用高斯消元解即可，解完直接用ϕ来计算C的值即可。时间复杂度$O(n^3)$。注意模数比较大，需要用快速乘。

【参考程序】//陈煜翔

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void putint(T x)
{
    static char buf[45], *tail = buf;
    if (!x)
        putchar('0');
    else
    {
        if (x < 0)
            putchar('-'), x = ~x + 1;
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

typedef long long s64;
```

```
const int MaxN = 110;
const int MaxM = 4020;

int n, m;
s64 mod;

int ect, adj[MaxN];
int nxt[MaxM], to[MaxM];
s64 weight_a[MaxM], weight_b[MaxM];
s64 weight_p[MaxM], ans[MaxM];

#define foredge(u) for (int e = adj[u], v; v = to[e], e; e = nxt[e])

inline void add(s64 &x, const s64 &y)
{
    x += y;
    if (x >= mod)
        x -= mod;
}

inline void dec(s64 &x, const s64 &y)
{
    x -= y;
    if (x < 0)
        x += mod;
}

inline s64 qmul(s64 x, s64 y) //计算 x*y%mod
{
    s64 res = 0;
    for (; y; y >>= 1, add(x, x))
        if (y & 1)
            add(res, x);
    return res;
}

inline s64 qpow(s64 x, s64 y)//计算 x^y%mod
{
    s64 res = 1;
    for (; y; y >>= 1, x = qmul(x, x))
        if (y & 1)
            res = qmul(res, x);
    return res;
}

inline void addEdge(int u, int v, s64 a, s64 b, s64 p)
{
    nxt[++ect] = adj[u];
    adj[u] = ect;
    to[ect] = v;
    weight_a[ect] = a;
    weight_b[ect] = b;
```

```
    weight_p[ect] = p;
}

s64 mat[MaxN][MaxN];

inline void Gauss() //高斯消元
{
    for (int i = 1; i <= n; ++i)
    {
        int p = 0;
        for (int j = i; j <= n; ++j)
            if (mat[j][i])
            {
                p = j; //找到一行使得其第 i 个变量的系数不为 0
                break;
            }
        if (p != i)
        {
            for (int j = i; j <= n + 1; ++j)
                std::swap(mat[p][j], mat[i][j]);
        }

        s64 div = qpow(mat[i][i], mod - 2);
        for (int j = i; j <= n + 1; ++j)
            mat[i][j] = qmul(mat[i][j], div);
        for (int j = 1; j <= n; ++j)
            if (j != i && mat[j][i])
            {
                s64 t = mat[j][i];
                for (int k = i; k <= n + 1; ++k)
                    dec(mat[j][k], qmul(mat[i][k], t));
                    //用当前的行消去其他行的第 i 个变量
            }
    }
}

int main()
{
    freopen("graph.in", "r", stdin);
    freopen("graph.out", "w", stdout);

    read(n), read(m), read(mod);
    for (int i = 1; i <= m; ++i)
    {
        int x, y;
        s64 a, b, p;
        read(x), read(y), read(a), read(b);
        b = qpow(b, mod - 2); //这里 b 提前取逆元，免去不必要的运算
        p = qmul(a, b); //p 表示题目中 a/b 的值
        addEdge(x, y, a, b, p);
        addEdge(y, x, (mod - a) % mod, b, (mod - p) % mod);
    }
```

```
    mat[1][1] = 1; //强制φ(1)=0
    for (int u = 2; u <= n; ++u)
    {
        foredge(u)//对于每个点u，枚举它的出边，列方程
        {
            dec(mat[u][u], weight_b[e]);
            add(mat[u][v], weight_b[e]);
            dec(mat[u][n + 1], weight_p[e]);
        }
    }

    Gauss();
    for (int u = 1; u <= n; ++u)
    {
        foredge(u)
            if (e & 1)
            {
                s64 cur = mat[v][n + 1];
                dec(cur, mat[u][n + 1]);
                add(cur, weight_a[e]);
                ans[e] = qmul(cur, weight_b[e]);
    //根据计算出的φ，来得出边权C的值
            }
    }

    for (int e = 1; e <= ect; e += 2)
    {
        putint(ans[e]);
        putchar('\n');
    }

    return 0;
}
```