{{ self.title() }}

题目简述

有一张 $n \times n$ 的网格，一开始共有 m 个格子被涂成黑色，其它所有格子都是白色的。

如果存在三个正整数 x, y, z（$1 \leq x, y, z \leq n$），满足 (x, y) 和 (y, z) 都是黑色的，那么就可以将 (z, x) 涂成黑色。

求最多能使棋盘上有多少个黑色的格子。

子任务设计

前 50 分是给直接模拟的。

算法1：模拟

每次新涂黑一个格子就重复枚举 x, y, z，复杂度为 $O\left(n^5\right)$。

优化一下，一开始先把用初始的 m 个黑色格子能得到的格子记在队列中，每次从队列中取出一个格子只需要枚举对应的 x 或 z 即可。复杂度降为 $O\left(n^3\right)$，可以获得 50 分。

算法2：正解

假设有三个互不相交的非空数集 X, Y, Z，且存在三个正整数 x, y, z，$x \in X$，$y \in Y$，$z \in Z$，$1 \leq x, y, z \leq n$。那么对于一张 $n \times n$ 的网格的最终状态（即不能再涂黑白色格子）：

- 如果 (x, y) 和 (y, z) 是黑的，那么 (z, x) 是黑的；
- 如果 (y, z) 和 (z, x) 是黑的，那么 (x, y) 是黑的；
- 如果 (z, x) 和 (x, y) 是黑的，那么 (y, z) 是黑的。

更进一步，如果对任意 $x \in X, z \in Z$，都存在 $y_0 \in Y$ 满足 (x, y_0) 和 (y_0, z) 同时为黑，那么 (z, x) 也是黑的。

可以看出，如果把一个下标同时当做行和列来考虑，那么整个问题描述会变得比较复杂。我们不妨把下标当做一个单独的元素，把一个黑色的格子当做两个元素之间产生关联。这种关联是单向的，即 (x, y) 是黑的不能得出 (y, x) 也是黑的。

可以联系到图论中的点与单向边的关系。

显然对于不同连通块，我们可以分开处理。

如果一个连通块的点集可以被划分为两个不相交的集合，且只有其中一个集合的点向另一个集合连边，每个集合内部没有连边，那么这个连通块不能产生新的边。

如果一个连通块的点集可以被划分为三个不相交的集合 X, Y, Z，且 X 中的每个点都向 Y 中某个点连边，Y 中每个点都向 Z 中某个点连边，Z 中每个点都向 X 中某个点连边（这里的"某个点"是存在的关系，对于不同的点，出边指向的点不一定相同），且每个集合内部没有连边，那么这个连通块的最终状态为 X 中所有点向 Y 中所有点连边，Y 中所有点向 Z 中所有点连边，Z 中所有点向 Y 中所有点连边。最终边数为 $|X|\,|Y| + |Y|\,|Z| + |Z|\,|X|$。

如果一个连通块不符合上面说的任意一种情况，如集合内部有连边，则最终状态为集合内部所有点对之间都有连边（包括自环）。最终边数为点数平方。

由此我们可以得到一个 $O\left(n + m\right)$ 的做法，可以通过本题。

```cpp
#include <cstdio>
using namespace std;
struct Edge{
    int to, w, next;
}e[200010];
int head[100010], col[100010], vis[100010], q[100010], cnt[3], en;
int main(){
    register int n, m, i, j, u, v, qh, qt, ecnt, clique;
    register long long ans;
    scanf("%d %d", &n, &m);
    for (en = i = 1; i <= m; ++i) {
        scanf("%d %d", &u, &v);
        e[++en] = (Edge){v, 1, head[u]};
        head[u] = en;
        e[++en] = (Edge){u, 2, head[v]};
        head[v] = en;
    }
    ans = 0;
    for (i = 1; i <= n; ++i) if (vis[i] == 0) {
        clique = ecnt = cnt[0] = cnt[1] = cnt[2] = 0;
        q[qh = qt = 0] = i;
        vis[i] = 1;
        while (qh <= qt) {
            u = q[qh];
            ++qh;
            ++cnt[col[u]];
            for (j = head[u]; j; j = e[j].next) {
                ++ecnt;
                if (vis[v = e[j].to]) {
                    if (col[v] != (col[u] + e[j].w) % 3) clique = 1;
                }
                else {
                    vis[v] = 1;
                    col[v] = (col[u] + e[j].w) % 3;
                    q[++qt] = v;
                }
            }
        }
        if (clique) ans += 1ll * (cnt[0] + cnt[1] + cnt[2]) * (cnt[0] + cnt[1] +
cnt[2]);
        else if (cnt[0] && cnt[1] && cnt[2]) ans += 1ll * cnt[0] * cnt[1] + 1ll
* cnt[1] * cnt[2] + 1ll * cnt[2] * cnt[0];
        else ans += ecnt >> 1;
    }
    printf("%lld\n", ans);
    return 0;
}
```