{{ self.title() }}

题目简述

给出 $G=(V,E)$，请将 V 划分为两个集合 S,T，使得 $S\cap T=\varnothing$，$S\cup T=V$，$\forall u,v\in S,(u,v)\in E\ \ (u\neq v)$ ，$\forall u,v\in T,(u,v)\in E\ \ (u\neq v)$ ，且 $\frac{|S|(|S|-1)}{2}+\frac{|T|(|T|-1)}{2}$ 最小。如果不存在这样的划分，则输出 `-1`。

子任务设计

前 30% 分是给爆搜的。

中间有 10% 图是完全图，可以直接输出答案。

算法1：搜索

直接搜索每个点属于哪个集合，然后判断一下是否满足要求。

复杂度 $O\left(2^n m\right)$，可以进行最优化剪枝。

算法2：完全图

记 $f\left(x\right)=\frac{x(x-1)}{2}+\frac{(n-x)(n-x-1)}{2}$，可以证明当 $x=\frac{n}{2}$ 时 $f\left(x\right)$ 有最小值。

故直接输出 $f\left(\left\lfloor\frac{n}{2}\right\rfloor\right)$ 即可。

算法3：补图

由算法 2 中的分析我们可以知道，要使答案尽量小，就要使每个团的大小尽可能地靠近 $n/2$。

如果一张图只有一种分配方法，那么答案是确定的。如果一张图有多种分配方法，那每一个团都可以拆成互相独立的若干部分。在图论中，这种可拆分的性质往往与染色有关系，即我们可以认为，如果同一个团中有两个点可以属于不同的团，则这两个点在一定的染色规则下可以被染成相同的颜色，也可以被染成不同的颜色。

如果两个点之间没有连边，那么这两个点一定要属于不同的团，即被染成不同的颜色。由此我们可以得知，对给出的图 G 取补图后，得到的新图 G' 的每个连通块即是拆分的最小单位；而对每个连通块进行黑白染色使得相邻的结点的颜色不同，就可以得知每个连通块中，哪些点必须属于同一个团，而哪些点必须属于不同的团。

经过转换，问题就变成：给出若干的数对 $\left(x_i,y_i\right)$，每个数对只能恰好选 x_i 或 y_i 中的一个，求每个数对中选出的数之和最靠近 $\sum_i\left(x_i+y_i\right)$ 的值是多少。将最优化问题转化为判定性问题，即转换为每个在 $\left[0,\sum_i\left(x_i+y_i\right)\right]$ 中的整数能否用这种方法表示出来。可以记 $f\left(i,j\right)$ 表示使用了前 i 个数对能否表示出 j，这样就可以用简单的 DP 解决整个问题了。

复杂度为 $O\left(n^2\right)$，可以通过本题。DP 部分还可以用 bitset 进行优化，减小常数。

```
#include <cstdio>
#include <bitset>
```

```cpp
using namespace std;
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
bitset<2005> f;
int e[2005][2005], vis[2005], q[2005], cnt[3];
char s[2005];
int main(){
    register int n, m, i, j, u, v, qh, qt, ans;
    scanf("%d %d", &n, &m);
    for (i = 1; i <= n; ++i) {
        scanf("%s", s + 1);
        for (j = 1; j <= n; ++j) e[i][j] = (i == j ? 0 : s[j] ^ '1');
    }
    f.set(0);
    for (i = 1; i <= n; ++i) if (vis[i] == 0) {
        q[qh = qt = 0] = i;
        vis[i] = 1;
        cnt[1] = cnt[2] = 0;
        while (qh <= qt) {
            u = q[qh];
            ++qh;
            ++cnt[vis[u]];
            for (v = 1; v <= n; ++v) if (e[u][v]) {
                if (vis[v]) {
                    if ((vis[u] ^ vis[v]) != 3) {
                        puts("-1");
                        return 0;
                    }
                }
                else {
                    vis[v] = vis[u] ^ 3;
                    q[++qt] = v;
                }
            }
        }
        f = f << cnt[1] | f << cnt[2];
    }
    ans = n * n;
    for (i = 1; i <= n; ++i) if (f[i]) {
        cmin(ans, i * (i - 1) + (n - i) * (n - i - 1) >> 1);
    }
    printf("%d\n", ans);
    return 0;
}
```