E2-F题解

(参考standard的解法)

思路

此问题可以转换为图遍历问题，其中每个格子视为图的一个节点，传送条件定义了节点之间的边。通过虚拟节点来简化同行、同列传送的实现。

具体步骤

1. **节点和虚拟节点的表示**：
   - 每个格子的位置 `(i, j)` 可视为一个节点，编号为 `id = i * m + j`。
   - 为了实现行列传送，我们为每一行和每一列创建额外的"虚拟节点"：
     - 行虚拟节点：每一行的两个颜色分别创建一个虚拟节点。例如，第 `i` 行的黑色虚拟节点编号为 `n * m + 2 * i`，白色虚拟节点编号为 `n * m + 2 * i + 1`。
     - 列虚拟节点：每一列的两个颜色分别创建一个虚拟节点。例如，第 `j` 列的黑色虚拟节点编号为 `n * m + 2 * n + 2 * j`，白色虚拟节点编号为 `n * m + 2 * n + 2 * j + 1`。
2. **构建图的边**：
   - 对于每个格子 `(i, j)`：
     - 连接到该行的同色虚拟节点。
     - 连接到该列的异色虚拟节点。
   - 每个格子通过这些虚拟节点间接连接到同行、同列的符合条件的其他格子。
3. **图遍历求解**：
   - 使用广度优先搜索（BFS）从起点 `(1,1)` 开始遍历。
   - 遇到终点 `(n, m)` 时，记录步数即为最少传送次数。
4. **输出结果**：
   - 如果终点 `(n, m)` 可以到达，则输出步数。
   - 如果 BFS 无法到达终点，则输出 `-1`。

代码实现

```
#include iostream
#include <bits/stdc++.h>
using namespace std;

int main() {
    int T;
    cin >> T;
    while (T--) {
        int n, m;
        cin >> n >> m;
```

```
        vector<vector<int>> g(n * m + 2 * (n + m));
        vector<vector<int>> a(n, vector<int>(m));

        // 读取网格颜色并建立边
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                scanf("%d",&a[i][j]);
                int id = i * m + j;
                int rowNode = n * m + 2 * i + a[i][j];
                g[id].emplace_back(rowNode);
                g[rowNode].emplace_back(id);
            }
        }

        // 列的虚拟节点建立边
        for (int j = 0; j < m; j++) {
            int colNode = n * m + 2 * n + 2 * j;
            for (int i = 0; i < n; i++) {
                int id = i * m + j;
                g[id].emplace_back(colNode + (a[i][j] ^ 1));
                g[colNode + a[i][j]].emplace_back(id);
            }
        }

        vector<int> vis(n * m + 2 * (n + m), 0);
        queue<pair<int, int>> q;
        q.push({0, 0});
        vis[0] = 1;
        int ans = -1;

        while (!q.empty()) {
            auto [cur, steps] = q.front();
            q.pop();
            if (cur == n * m - 1) {
                ans = steps / 2;
                break;
            }
            for (int next : g[cur]) {
                if (!vis[next]) {
                    vis[next] = 1;
                    q.push({next, steps + 1});
                }
            }
        }

        printf("%d\n",ans);
    }
    return 0;
}
```