1001 布布吃地瓜题解

Gospel_rock

1 题意

给定一个 $n \times m$ 的网格，格子 (i, j) 上写着整数 $a_{i,j}$。从 $(1, 1)$ 出发，经四连通移动到 (n, m)，不能离开网格。令 S 为路径上所有格子中整数构成的集合，求所有合法路径中 $\mathrm{mex}(S)$ 的最小值。

2 核心观察：路径—屏障对偶

问题等价于：对每个非负整数 k，判断是否**每条** $(1, 1) \to (n, m)$ 的四连通路径都必须经过至少一个值为 k 的格子。如果是，则 k 一定在集合 S 中（称 k 为**不可避免的**）；否则存在某条路径绕开所有值为 k 的格子（称 k 为**可避免的**）。答案即为最小的可避免值。

判断 k 是否不可避免，依赖于网格上**路径与屏障的对偶性**——这是离散 Jordan 曲线定理在网格图上的直接推论。

2.1 边界弧的划分

将网格的边界沿周长分为两段弧，以 $(1, 1)$ 和 (n, m) 为分界点（如图 1）：

- **弧 A**（红色）：从 $(1, 1)$ 沿顶边向右到 $(1, m)$，再沿右边向下到 (n, m)。
- **弧 B**（蓝色）：从 $(1, 1)$ 沿左边向下到 $(n, 1)$，再沿底边向右到 (n, m)。

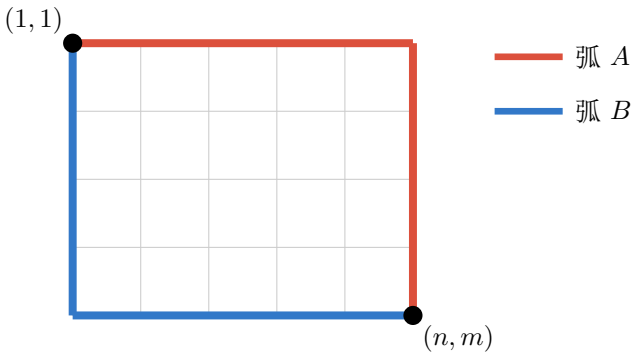

Figure 1: 边界弧的划分。$(1, 1)$ 和 (n, m) 将网格边界分成红、蓝两段弧。

2.2 对偶定理

> **定理（网格路径—屏障对偶）** 在 $n \times m$ 的网格中，不存在从 $(1, 1)$ 到 (n, m) 的四连通路径能完全避开值为 k 的格子，当且仅当值为 k 的格子中存在一个**八连通**连通分量，同时触及弧 A 和弧 B。

直觉上很好理解：任何一条从 $(1, 1)$ 到 (n, m) 的四连通路径，都会将网格”切割”为两部分——一部分靠近弧 A，另一部分靠近弧 B。若一群同值格子八连通地从弧 A 连到弧 B，则这群格子构成了一道”墙”，任何四连通路径都无法从”墙”的缝隙中穿过（四连通无法斜穿八连通的间隙），因此必须踩到墙上。

图 2 展示了一个具体例子。

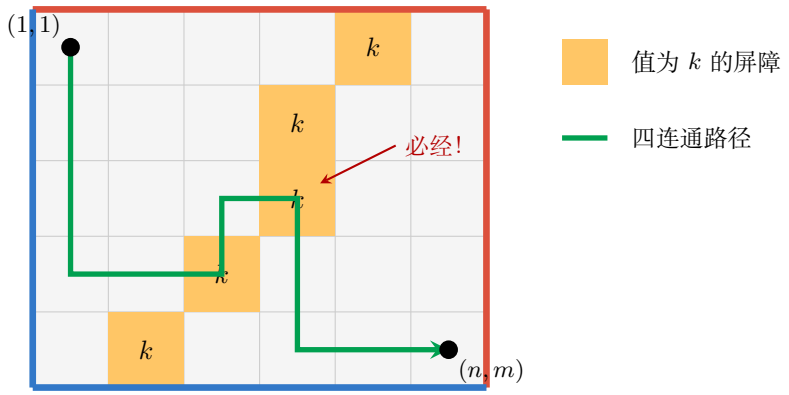

Figure 2: 值为 k 的格子（橙色）八连通地从弧 A（顶边）连到弧 B（底边），形成屏障。任何四连通路径（绿色）都必须踩到至少一个橙色格子。

3 算法流程

根据对偶定理，算法只需逐一判断每个值是否构成屏障即可。

1. 将候选集合 $C \leftarrow \{0, 1, 2, \ldots, n+m\}$。
2. 枚举弧 A 上的每个格子 (i, j)（即第 1 行的所有格子与第 m 列的所有格子），对尚未访问的格子发起**八连通 DFS**，仅扩展值相同的邻居。
3. 若 DFS 过程中到达了弧 B 上的某个格子（即第 1 列或第 n 行），说明当前值 v 构成屏障，从 C 中删除 v。
4. 所有弧 A 上的格子处理完毕后，C 中的最小元素即为答案。

为什么只需从弧 A 出发搜索？因为屏障必须**同时**触及弧 A 与弧 B，从弧 A 出发的同值连通分量若能到达弧 B，就恰好满足此条件。

3.1 复杂度分析

每个格子至多被访问一次（DFS 中标记 `vis`），因此总时间复杂度为 $O(nm)$，空间复杂度为 $O(nm)$。

4 AC 代码

```
// teamname: Gospel_rock
/**
 * Problem: 布布吃地瓜
 * Contest:
 * Judge: HDOJ
 * URL: https://acm.hdu.edu.cn/contest/problem?cid=1200&pid=1001
 * Created: 2026-04-10 20:42:20
 * Author: Gospel_rock
 * My blog: https://znzryb.com/
 *
 * Powered by AutoCp https://github.com/Pushpavel/AutoCp
 */

#include <bits/stdc++.h>
#define all(vec) vec.begin(),vec.end()
#define lson(o) (o<<1)
#define rson(o) (o<<1|1)
#define SZ(a) ((long long) a.size())
#define debug(var) cerr << #var <<" = ["<<var<<"]"<<"\n";
#define debug1d(a)    \
cerr << #a << " = [";   \
for (int i = 0; i < (int)(a).size(); i++) \
```

```
cerr << (i ? ", " : "") << a[i]; \
cerr << "]\n";
#define debug2d(a)  \
cerr << #a << " = [\n";  \
for (int i = 0; i < (int)(a).size(); i++)  \
{   \
cerr << "  [";  \
for (int j = 0; j < (int)(a[i]).size(); j++) \
cerr << (j ? ", " : "") << a[i][j];   \
cerr << "]\n";   \
}  \
cerr << "]\n";
#define cend cerr<<"\n----------\n"
#define fsp(x) fixed<<setprecision(x)

using namespace std;

using ll = long long;
using ull = unsigned long long;
using DB = double;
using i128 = __int128;
using CD = complex<double>;

static constexpr ll MAXN = (ll) 1e6 + 10, INF = (1ll << 61) - 1;
static constexpr ll mod = 998244353; // (ll)1e9+7;
static constexpr double eps = 1e-8;
const double PI = acos(-1.0);

ll lT, testcase;

/*
 *
 */
int dx[] = {1, -1, 0, 0, 1, -1, 1, -1};
int dy[] = {0, 0, 1, -1, 1, -1, -1, 1};

struct Solve {
    ll N, M;
    vector<vector<int> > maze;
    vector<vector<char> > vis;

    [[nodiscard]] bool check(ll x, ll y, ll val) const {
        if (x < 1 || y < 1) return false;
        if (x > N || y > M) return false;
        if (vis[x][y]) return false;
        if (val != maze[x][y]) return false;
        return true;
    }

    int dfs(const ll x, const ll y,ll souX,ll souY) {
        if (y == 1 || x == N) {
// #ifdef LOCAL
//             debug(x);
//             debug(y);
//             debug(souX);
//             debug(souY);
//             debug(maze[souX][souY]);
//             debug(maze[x][y]);
//             cend;
// #endif
            return 1;
        }
        int res = 0;
```

```
        for (int i = 0; i < 8; ++i) {
            ll tox = x + dx[i], toy = y + dy[i];
            if (check(tox, toy, maze[x][y])) {
                vis[tox][toy] = 1;
                res |= dfs(tox, toy,souX,souY);
            }
        }
        return res;
    }

    set<ll> st;

    Solve() {
        cin >> N >> M;
        maze.resize(N + 2, vector<int>(M + 2));
        vis.resize(N + 2, vector<char>(M + 2));
        for (int i = 1; i <= N; ++i) {
            for (int j = 1; j <= M; ++j) {
                cin >> maze[i][j];
            }
        }
// #ifdef LOCAL
//         debug2d(maze);
// #endif

        for (int i = 0; i <= N + M; ++i) {
            st.insert(i);
        }
        for (int y = 1; y <= M; ++y) {
            if (vis[1][y]) {
                continue;
            }
            vis[1][y] = 1;
            if (dfs(1, y,1,y)) {
                st.erase(maze[1][y]);
            }
        }
        for (int x = 1; x <= N; ++x) {
            if (vis[x][M]) {
                continue;
            }
            vis[x][M] = 1;
            if (dfs(x, M,x,M)) {
                st.erase(maze[x][M]);
            }
        }
        cout << *st.begin() << "\n";
    }
};

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
#ifdef LOCAL
    cout.setf(ios::unitbuf); // 无缓冲流，方便我们调试
#endif

    cin >> lT;
    for (testcase = 1; testcase <= lT; ++testcase)
        Solve solve;
    return 0;
}
```

```
/*
AC
https://acm.hdu.edu.cn/contest/view-code?cid=1200&rid=13266


*/
```