# 题目

给你一个下标从 0 开始、大小为 n x n 的矩阵 grid ，其中 n 为奇数，且 grid[r][c] 的值为 0 、1 或 2 。

如果一个单元格属于以下三条线中的任一一条，我们就认为它是字母 Y 的一部分：

从左上角单元格开始到矩阵中心单元格结束的对角线。

从右上角单元格开始到矩阵中心单元格结束的对角线。

从中心单元格开始到矩阵底部边界结束的垂直线。

当且仅当满足以下全部条件时，可以判定矩阵上写有字母 Y ：

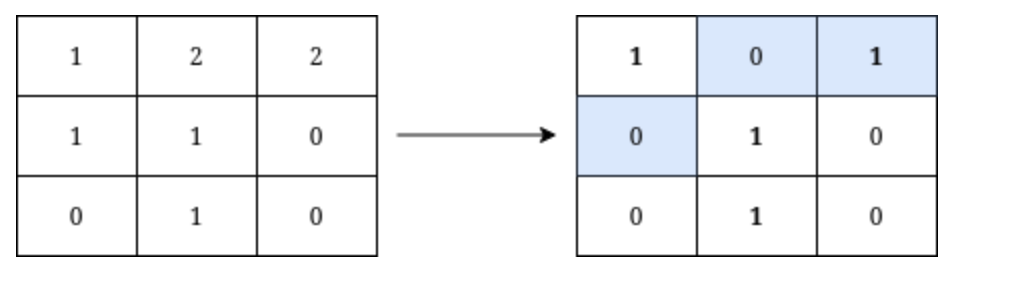
属于 Y 的所有单元格的值相等。

不属于 Y 的所有单元格的值相等。

属于 Y 的单元格的值与不属于Y的单元格的值不同。

每次操作你可以将任意单元格的值改变为 0 、1 或 2 。返回在矩阵上写出字母 Y 所需的 最少 操作次数。

示例 1：



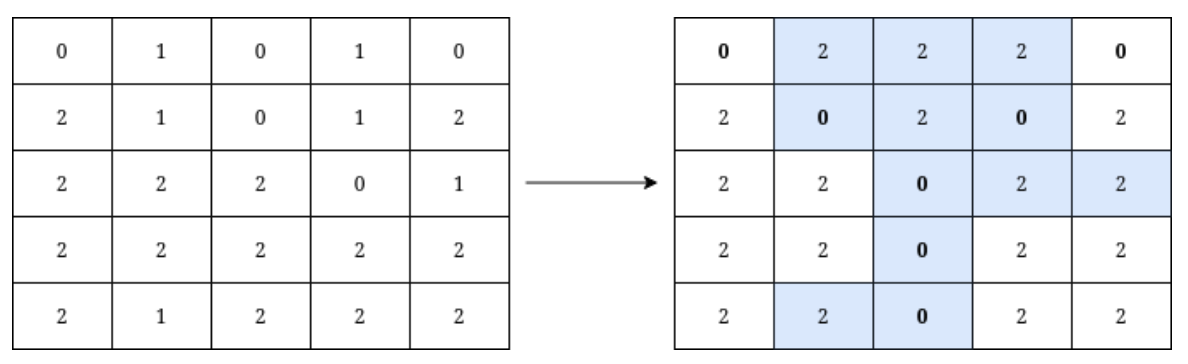
输入：grid = [[1,2,2],[1,1,0],[0,1,0]]

输出：3

解释：将在矩阵上写出字母 Y 需要执行的操作用蓝色高亮显示。操作后，所有属于 Y 的单元格（加粗显示）的值都为 1 ，而不属于 Y 的单元格的值都为 0 。

可以证明，写出 Y 至少需要进行 3 次操作。

示例 2：



输入：grid = [[0,1,0,1,0],[2,1,0,1,2],[2,2,2,0,1],[2,2,2,2,2],[2,1,2,2,2]]

输出：12

解释：将在矩阵上写出字母 Y 需要执行的操作用蓝色高亮显示。操作后，所有属于 Y 的单元格（加粗显示）的值都为 0 ，而不属于 Y 的单元格的值都为 2 。

可以证明，写出 Y 至少需要进行 12 次操作。

提示：

3 <= n <= 49

n == grid.length == grid[i].length

0 <= grid[i][j] <= 2

n 为奇数。

# 分析

要解决“矩阵写出字母Y的最少操作次数”问题，核心思路是先划分Y区域与非Y区域，再枚举所有可能的颜色组合，计算每种组合的操作次数，取最小值。

解题思路

1、明确Y区域的定义：

字母Y由三条线组成（n为奇数，中心坐标为(mid, mid)，mid = n//2）：

- 左上到中心的对角线：行号r从0到mid，列号c = r（即r == c且r <= mid）。

- 右上到中心的对角线：行号r从0到mid，列号c = n-1 - r（即c == n-1 - r且r <= mid）。

- 中心到底部的垂直线：列号c = mid，行号r从mid到n-1（即c == mid且r >= mid）。

2、统计区域颜色频次：

遍历矩阵，将单元格分为“Y区域”和“非Y区域”，分别统计两个区域中颜色0、1、2的出现次数（频次统计可快速计算后续操作次数）。

3、枚举颜色组合计算操作次数：

题目要求Y区域颜色（y\_color）与非Y区域颜色（non\_y\_color）不同，因此枚举所有y\_color != non\_y\_color的组合（共3×2=6种可能），计算每种组合的操作次数：

- Y区域操作次数 = Y区域总单元格数 - Y区域中y\_color的频次（需将非y\_color的单元格改为y\_color）。

- 非Y区域操作次数 = 非Y区域总单元格数 - 非Y区域中non\_y\_color的频次（需将非non\_y\_color的单元格改为non\_y\_color）。

- 总操作次数 = 两者之和，取所有组合中的最小值。

代码实现（含详细注释）：

class Solution {

public:

int minimumOperationsToWriteY(vector<vector<int>>& grid) {

int n = grid.size();

int mid = n / 2; // 矩阵中心坐标的行/列号（n为奇数）

// 初始化频次统计：y\_cnt[color]是Y区域中颜色color的次数，non\_y\_cnt同理

vector<int> y\_cnt(3, 0);

vector<int> non\_y\_cnt(3, 0);

int y\_total = 0; // Y区域总单元格数

// 遍历矩阵，划分Y区域与非Y区域并统计频次

for (int r = 0; r < n; ++r) {

for (int c = 0; c < n; ++c) {

bool is\_y = false;

// 检查是否属于Y区域的三条线之一

// 1. 左上到中心的对角线（r==c且r<=mid）

if (r == c && r <= mid) {

is\_y = true;

}

// 2. 右上到中心的对角线（c==n-1 -r且r<=mid）

else if (c == (n - 1 - r) && r <= mid) {

is\_y = true;

}

// 3. 中心到底部的垂直线（c==mid且r>=mid）

else if (c == mid && r >= mid) {

is\_y = true;

}

// 更新对应区域的颜色频次和总计数

int color = grid[r][c];

if (is\_y) {

y\_cnt[color]++;

y\_total++;

} else {

non\_y\_cnt[color]++;

}

}

}

int non\_y\_total = n \* n - y\_total; // 非Y区域总单元格数

int min\_ops = INT\_MAX; // 存储最少操作次数

// 枚举所有可能的y\_color和non\_y\_color组合（y\_color != non\_y\_color）

for (int y\_color = 0; y\_color < 3; ++y\_color) {

for (int non\_y\_color = 0; non\_y\_color < 3; ++non\_y\_color) {

if (y\_color == non\_y\_color) {

continue; // 跳过颜色相同的组合（不符合题目要求）

}

// 计算当前组合的操作次数

int y\_ops = y\_total - y\_cnt[y\_color]; // Y区域需修改的次数

int non\_y\_ops = non\_y\_total - non\_y\_cnt[non\_y\_color]; // 非Y区域需修改的次数

int total\_ops = y\_ops + non\_y\_ops;

// 更新最少操作次数

min\_ops = min(min\_ops, total\_ops);

}

}

return min\_ops;

}

};

代码解释

1、Y区域判定：通过三个条件精准划分Y区域（三条线），确保不遗漏或误判单元格。

2、频次统计：用两个大小为3的数组分别统计Y区域和非Y区域的颜色频次，为后续操作次数计算提供基础。

3、组合枚举：仅枚举y\_color != non\_y\_color的6种组合，避免无效计算，确保效率；通过“总单元格数 - 目标颜色频次”快速得到操作次数，逻辑直观且高效。

复杂度分析

- 时间复杂度：O(n²)，其中n为矩阵边长。遍历矩阵统计频次耗时O(n²)，枚举6种颜色组合耗时O(1)，整体线性依赖于矩阵大小。

- 空间复杂度：O(1)，仅使用固定大小的频次数组（y\_cnt、non\_y\_cnt）和几个变量，不依赖输入规模。

该方法通过“划分区域→统计频次→枚举组合”的逻辑，高效找到最少操作次数，完全满足题目中n≤49的约束。