# 题目

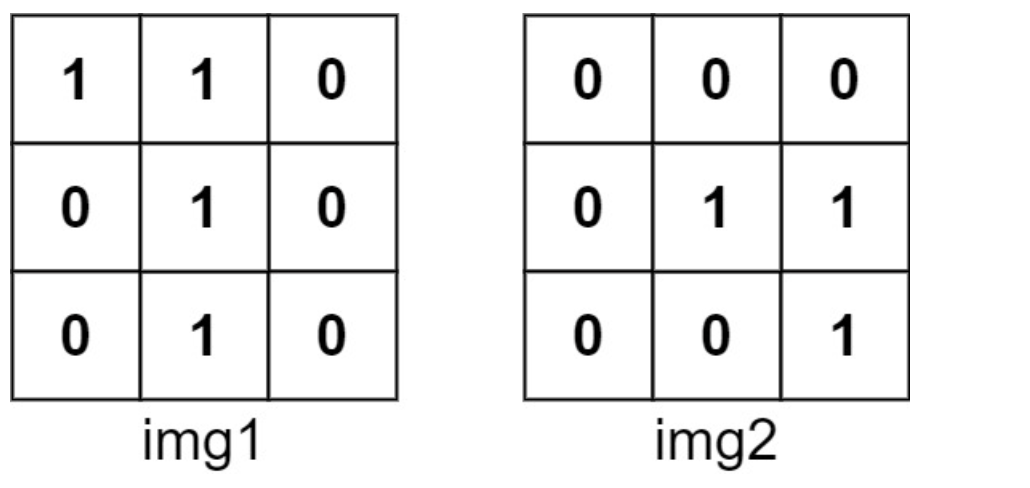
给你两个图像 img1 和 img2 ，两个图像的大小都是 n x n ，用大小相同的二进制正方形矩阵表示。二进制矩阵仅由若干 0 和若干 1 组成。

转换 其中一个图像，将所有的 1 向左，右，上，或下滑动任何数量的单位；然后把它放在另一个图像的上面。该转换的 重叠 是指两个图像 都 具有 1 的位置的数目。

请注意，转换不包括向任何方向旋转。越过矩阵边界的 1 都将被清除。

最大可能的重叠数量是多少？

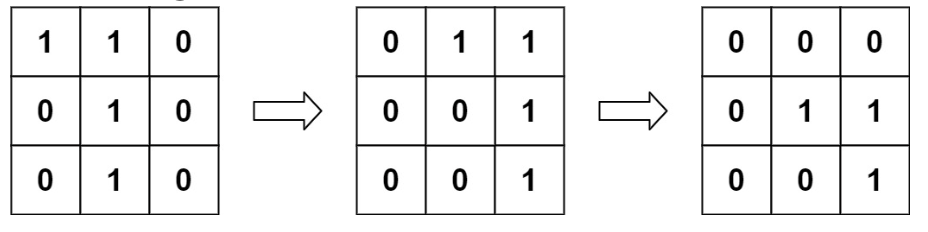
示例 1：



输入：img1 = [[1,1,0],[0,1,0],[0,1,0]], img2 = [[0,0,0],[0,1,1],[0,0,1]]

输出：3

解释：将 img1 向右移动 1 个单位，再向下移动 1 个单位。



两个图像都具有 1 的位置的数目是 3（用红色标识）。

日历

低可信度描述已自动生成

示例 2：

输入：img1 = [[1]], img2 = [[1]]

输出：1

示例 3：

输入：img1 = [[0]], img2 = [[0]]

输出：0

提示：

n == img1.length == img1[i].length

n == img2.length == img2[i].length

1 <= n <= 30

img1[i][j] 为 0 或 1

img2[i][j] 为 0 或 1

# 方法

要解决这个问题，我们需要找到两个二进制图像在平移操作下的最大重叠数量。重叠数量是指两个图像中都为1的位置的数目，平移操作包括将一个图像的所有1向左、右、上、下滑动任意单位，超出边界的1会被清除。

思路：

核心思路是通过分析两个图像中1的位置关系，找到能使最多1重叠的平移方式。具体步骤如下：

1、收集1的坐标：首先遍历两个图像，记录下所有值为1的位置坐标，分别存储在两个列表中。

2、计算位移向量：对于第一个图像中的每个1，计算它与第二个图像中每个1的相对位移（即需要将第一个图像的1移动到第二个图像的1所需的平移量）。

3、统计位移出现次数：统计每种位移出现的次数，次数最多的位移对应的次数就是最大重叠数量（因为该位移能让最多的1在两个图像中重叠）。

代码：

class Solution {

public:

int largestOverlap(vector<vector<int>>& img1, vector<vector<int>>& img2) {

int n = img1.size();

vector<pair<int, int>> ones1, ones2;

// 收集两个图像中所有1的坐标

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

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

if (img1[i][j] == 1) {

ones1.emplace\_back(i, j);

}

if (img2[i][j] == 1) {

ones2.emplace\_back(i, j);

}

}

}

// 如果有一个图像没有1，重叠数量为0

if (ones1.empty() || ones2.empty()) {

return 0;

}

// 用哈希表统计每种位移对应的重叠数量

unordered\_map<int, int> shiftCount;

int maxOverlap = 0;

for (auto& p1 : ones1) {

int x1 = p1.first, y1 = p1.second;

for (auto& p2 : ones2) {

int x2 = p2.first, y2 = p2.second;

// 计算位移(dx, dy)，用一个整数编码位移（避免使用pair作为键）

int dx = x2 - x1;

int dy = y2 - y1;

int key = dx \* 60 + dy; // 60是因为n≤30，dx和dy的范围在[-29,29]，60足够区分

shiftCount[key]++;

maxOverlap = max(maxOverlap, shiftCount[key]);

}

}

return maxOverlap;

}

};

解释

1、收集1的坐标：通过两层循环遍历图像，将所有值为1的位置坐标存储在ones1（img1中的1）和ones2（img2中的1）中。

2、处理特殊情况：如果其中一个图像没有1，则重叠数量必然为0，直接返回0。

3、计算位移并统计：对于ones1中的每个1（坐标(x1,y1)）和ones2中的每个1（坐标(x2,y2)），计算位移向量(dx, dy) = (x2-x1, y2-y1)。这个位移向量表示将img1的1移动到img2的1所需的平移量。

4、编码位移：为了方便使用哈希表统计，将位移向量(dx, dy)编码为一个整数（这里用dx \* 60 + dy，因为dx和dy的范围有限，60足够避免冲突）。

5、寻找最大值：统计每种位移出现的次数，最大值即为最大可能的重叠数量。

这种方法的时间复杂度为O(k\*m)，其中k和m分别是两个图像中1的数量（最坏情况下为O(n²)，n为图像大小），空间复杂度为O(k\*m)，在题目给定的n≤30的约束下非常高效。