# 题目

给定 X-Y 平面上的一组点 points ，其中 points[i] = [xi, yi] 。这些点按顺序连成一个多边形。

如果该多边形为 凸 多边形（凸多边形的定义）则返回 true ，否则返回 false 。

你可以假设由给定点构成的多边形总是一个 简单的多边形（简单多边形的定义）。换句话说，我们要保证每个顶点处恰好是两条边的汇合点，并且这些边 互不相交 。

示例 1：

图表, 形状

描述已自动生成

输入: points = [[0,0],[0,5],[5,5],[5,0]]

输出: true

示例 2：

图表

描述已自动生成

输入: points = [[0,0],[0,10],[10,10],[10,0],[5,5]]

输出: false

提示:

3 <= points.length <= 104

points[i].length == 2

-104 <= xi, yi <= 104

所有点都不同

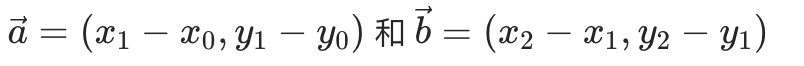
# 分析

要判断一个多边形是否为凸多边形，我们可以利用凸多边形的几何性质：\*\*凸多边形的所有内角均小于180度，且所有相邻边的转向（顺时针或逆时针）保持一致\*\*。

解题思路

1、向量叉积判断转向\*\*：对于多边形连续的三个顶点 \( p\_0, p\_1, p\_2 \)，可以通过向量叉积判断转向：

- 计算向量：



- 叉积：



- 若 cross > 0 ：逆时针转向

- 若 cross < 0 ：顺时针转向

- 若 cross = 0 ：三点共线（允许在凸多边形的边上）

2、统一转向判断：凸多边形的所有相邻边的转向必须保持一致（全为顺时针、全为逆时针，或存在共线情况）。若出现转向相反的情况，则多边形为凹多边形。

3、循环处理顶点：由于多边形是闭合的，最后一个顶点需与第一个、第二个顶点组成三角形进行判断。

代码：

class Solution {

public:

bool isConvex(vector<vector<int>>& points) {

int n = points.size();

long long prev = 0; // 存储前一个叉积的符号（避免int溢出）

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

// 取连续三个点：当前点i，下一个点(i+1)%n，下下个点(i+2)%n

int x1 = points[(i+1)%n][0] - points[i][0];

int y1 = points[(i+1)%n][1] - points[i][1];

int x2 = points[(i+2)%n][0] - points[(i+1)%n][0];

int y2 = points[(i+2)%n][1] - points[(i+1)%n][1];

// 计算叉积

long long curr = (long long)x1 \* y2 - (long long)y1 \* x2;

if (curr != 0) {

// 若当前叉积与前一个叉积符号相反，说明转向不一致，不是凸多边形

if (curr \* prev < 0) {

return false;

}

prev = curr; // 更新前一个叉积

}

}

return true;

}

};

解释：

- 叉积计算：通过向量叉积判断三个连续点的转向，叉积的符号决定了转向方向。

- 符号一致性检查：遍历所有连续三点组合，若出现叉积符号相反的情况（即curr\*prev< 0），则直接返回false。

- 处理共线情况：若叉积为0（三点共线），不改变前一个叉积的符号，继续检查后续点。

- 闭合多边形处：通过取模运算 (i+1)%n和 (i+2)%n 处理最后一个顶点与起始顶点的连接。

该算法的时间复杂度为O(n)（遍历所有顶点一次），空间复杂度为O(1)（仅使用常数空间），高效适用于大规模点集（如题目中n<=104的情况）。