# 题目

爱丽丝和鲍勃一起玩游戏，他们轮流行动。爱丽丝先手开局。

最初，黑板上有一个数字n。在每个玩家的回合，玩家需要执行以下操作：

选出任一x，满足0 < x < n且n % x == 0。

用n - x替换黑板上的数字n。

如果玩家无法执行这些操作，就会输掉游戏。

只有在爱丽丝在游戏中取得胜利时才返回 true 。假设两个玩家都以最佳状态参与游戏。

示例 1：

输入：n = 2

输出：true

解释：爱丽丝选择 1，鲍勃无法进行操作。

示例 2：

输入：n = 3

输出：false

解释：爱丽丝选择 1，鲍勃也选择 1，然后爱丽丝无法进行操作。

提示：

1 <= n <= 1000

# 分析

## 方法一：动态规划

思路：

我们可以定义一个数组 dp，其中 dp[i] 表示数字 i 的时候先手是否能获胜。

对于数字 i，爱丽丝可以选择一个约数 x，然后鲍勃将面对数字 i-x。如果鲍勃无法获胜，那么爱丽丝可以获胜；如果鲍勃能获胜，那么爱丽丝无法获胜。

因此，状态转移方程为：dp[i] = !dp[i-x]，其中 x 是 i 的约数。

最终，如果 dp[n] 为真，则爱丽丝能获胜；否则，爱丽丝无法获胜。

代码：

class Solution {

public:

bool divisorGame(int n) {

// dp[i] 表示数字 i 的情况下，先手是否能获胜

vector<bool> dp(n + 1, false);

for (int i = 2; i <= n; ++i) {

// 对于每个数字 i，遍历所有可能的 x，其中 0 < x < i 且 i % x == 0

for (int x = 1; x < i; ++x) {

// 如果存在一个 x，使得当数字变为 i - x 时，先手无法获胜，则先手可以选择 x 获胜

if (i % x == 0 && !dp[i - x]) {

dp[i] = true;

break;

}

}

}

// 返回数字 n 的情况下，先手是否能获胜

return dp[n];

}

};