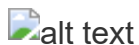


Problem 1



position: 一个总体的棋局状态

recursive algorithm: 递归算法

⚠️: 证明PSPACE和PSPACE-complete是两个不同的问题

✖️: if F can be reduced to a PSPACE-COMPLETE language, then F is a PSPACE language.

例如，不可判定语言（如停机问题）也能归约到 PSPACE 完全语言，但它不属于 PSPACE。

存在一个策略赢：存在一种下法，让对手所有下后的棋局，仍然存在策略赢

解题思路

Provement

To prove that $GM \in PSPACE$, we need to prove that it can be judged with poly-space.

A **recursive algorithm** is given below:

$GM = \exists x_i \forall y_i GM$

if 判断当前棋局，X已经赢了，那接收，消耗空间是常数级的

else 递归检测X是否存在一种下法，使得y所有落子可能之后，得到的position依然是GM，如果是，那就接受；反之

空间分析

当前棋盘状况，消耗空间为 $O(n^2)$

递归调用的空间，最多调用不超过 n^2 层，

采用递归空间复用策略，每层使用的栈空间均为常数级别（记录父层传递来的前边的落子位置），因此消耗的空间t

综上， GM can be judged with poly-space, thus $GM \in PSPACE$.

Proof

To show $GM \in PSPACE$, we need to demonstrate that GM can be decided using **polynomial space** via a recursive algorithm:

Recursive Algorithm for GM

The core logic of the algorithm is based on alternating moves (matching the quantifier structure of game strategies):

$$GM = \exists x_i \forall y_i GM'$$

where:

- x_i represents a valid move by player "X" (exists a move for X),
- y_i represents any valid move by player "O" (for all moves by O),
- GM' denotes the resulting game position after these moves.

The algorithm proceeds as follows:

1. **Base Case:** If the current position is a winning state for X (X already has 5 consecutive markers), **accept** (this check uses constant space).
2. **Recursive Step:**
 - For player X's turn: Check if **there exists at least one valid move** such that, for **all possible responses** by O, the resulting position is in GM . If yes, **accept**; otherwise, **reject**.
 - For player O's turn: Check if **for all possible valid moves** by O, there exists a response by X such that the resulting position is in GM . If yes, **accept**; otherwise, **reject**.

Space Complexity Analysis

- **Board State Storage:** The current board (an $n \times n$ grid) requires $O(n^2)$ space (to track placed markers and the next player).
- **Recursion Stack:** The maximum recursion depth is $O(n^2)$ (since there are at most n^2 empty cells, so at most n^2 moves left in the game). Each recursive stack frame uses **constant space** (only to record the move made from the parent state, no redundant storage of full board copies—we reuse space to update the board in-place).

Since the total space (board state + recursion stack) is $O(n^2)$ (a polynomial in the input size n), GM can be decided in polynomial space. Thus, $GM \in \text{PSPACE}$.

Problem 2



这题要给出一个下棋的策略，第一步落子的位置很关键，如果落的好，就能大大提高得胜概率，简化后续步骤
我用坐标(x,y)的形式来表示棋子的位置，原点设置在左上角，从1开始计数，比如左下角的白子坐标是（3，1）
X现在可以下到（1，3）的位置，既可以阻挡白子的延伸，又可以和已有的两个X子构建起联系。

进一步分情况讨论白子的落子情况及应对策略

Winning Strategy for the X-Player in Tic-Tac-Toe

Step 1: Analyze the Initial Board

The initial board (denoted as (row, column)):

- X's markers: (1,1) , (3,3)
- O's markers: (2,2) , (3,1)
- Empty cells: (1,2) , (1,3) , (2,1) , (2,3) , (3,2)

Step 2: X's First Move (Create Dual Threats)

X first places a marker at (1,3) (**1st row, 3rd column**).

This creates two incomplete "three-in-a-row" threats:

1. **1st row threat:** (1,1) + (1,3) (missing (1,2));
2. **3rd column threat:** (1,3) + (3,3) (missing (2,3)).

Step 3: Respond to All O's Defenses

O must block one of the two threats; X can win by completing the other threat:

Case 1: O blocks the 1st row threat (plays (1,2))

- O places at (1,2) to block the 1st row.
- X responds by placing at (2,3) (2nd row, 3rd column): this completes the **3rd column** ((1,3) + (2,3) + (3,3)), so X wins.

Case 2: O blocks the 3rd column threat (plays (2,3))

- O places at (2,3) to block the 3rd column.
- X responds by placing at (1,2) (1st row, 2nd column): this completes the **1st row** ((1,1) + (1,2) + (1,3)), so X wins.

Case 3: O plays an irrelevant cell (e.g., (2,1) or (3,2))

- If O places at (2,1) / (3,2) (not blocking either threat), X directly places at (1,2) or (2,3) to complete the 1st row/3rd column, so X wins.

Final Winning Strategy

1. X's first move: (1,3) (creates dual threats of 1st row and 3rd column).
2. For any defense by O, X places at the missing cell of the unblocked threat to complete three-in-a-row and win.