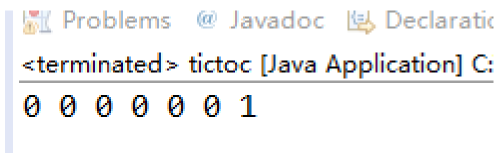
# 348. **Design Tic-Toc Toe** 一条龙游戏

Design a Tic-tac-toe game that is played between two players on a n x n grid.
You may assume the following rules:
A move is guaranteed to be valid and is placed on an empty block.
Once a winning condition is reached, no more moves is allowed.
A player who succeeds in placing n of their marks in a horizontal, vertical, or diagonal row wins the game.
Example:
Given n = 3, assume that player 1 is "X" and player 2 is "O" in the board.
TicTacToe toe = new TicTacToe(3);
toe.move(0, 0, 1); -> Returns 0 (no one wins)
|X| | |
| | | | // Player 1 makes a move at (0, 0).
| | | |
toe.move(0, 2, 2); -> Returns 0 (no one wins)
|X| |O|
| | | | // Player 2 makes a move at (0, 2).
| | | |
toe.move(2, 2, 1); -> Returns 0 (no one wins)
|X| |O|
| | | | // Player 1 makes a move at (2, 2).
| | |X|
toe.move(1, 1, 2); -> Returns 0 (no one wins)
|X| |O|
| |O| | // Player 2 makes a move at (1, 1).
| | |X|
toe.move(2, 0, 1); -> Returns 0 (no one wins)
|X| |O|
| |O| | // Player 1 makes a move at (2, 0).
|X| |X|
toe.move(1, 0, 2); -> Returns 0 (no one wins)
|X| |O|
|O|O| | // Player 2 makes a move at (1, 0).
|X| |X|
toe.move(2, 1, 1); -> Returns 1 (player 1 wins)
|X| |O|
|O|O| | // Player 1 makes a move at (2, 1).
|X|X|X|
Follow up:
Could you do better than O(n2) per move() operation?
Hint:
Could you trade extra space such that move() operation can be done in O(1)?
You need two arrays: int rows[n], int cols[n], plus two variables: diagonal, anti_diagonal.

Follow up中让我们用更高效的方法，那么根据提示中的，我们建立一个大小为n的一维数组rows和cols，还有变量对角线diag和逆对角线rev_diag，这种方法的思路是，如果玩家1在第一行某一列放了一个子，那么rows[0]自增1，如果玩家2在第一行某一列放了一个子，则rows[0]自减1，那么只有当rows[0]等于n或者-n的时候，表示第一行的子都是一个玩家放的，则游戏结束返回该玩家即可，其他各行各列，对角线和逆对角线都是这种思路：

```java
import java.util.Iterator;
import java.util.Vector;

public class tictoc {

public static class solution
{
	private int[] rows;
	private int[] cols;
	private int diag,rev_diag;//(zhewei) : diag and reverse diag 
of a NXN board
	private int sz;

public int move(int row, int col, int player)
{
	int add = player==1? 1: -1;
	rows[row]+=add;cols[col]+=add;
	if(col==row) diag+=add;
	if(row == sz-col-1) rev_diag+=add;
	if(Math.abs(rows[row])==sz || Math.abs(cols[col])==sz 
	|| Math.abs(diag)==sz || Math.abs(rev_diag)==sz)
		return player;
	return 0;
}

	solution(int n)
	{
		rows = new int[n];
		cols = new int[n];
		diag = 0;rev_diag = 0;
		sz = n;
	}
}



public static void main(String[] args) {
// TODO Auto-generated method stub
	solution s = new solution(3);
	int tmp;
	Vector<Integer> ans = new Vector<Integer>();
	tmp = s.move(0, 0, 1); ans.add(tmp);
	tmp = s.move(0, 2, 2); ans.add(tmp);
	tmp = s.move(2, 2, 1); ans.add(tmp);
	tmp = s.move(1, 1, 2); ans.add(tmp);
	tmp = s.move(2, 0, 1); ans.add(tmp);
	tmp = s.move(1, 0, 2); ans.add(tmp);
	tmp = s.move(2, 1, 1); ans.add(tmp);
```

```
        Iterator<Integer> it = ans.iterator();
        while(it.hasNext())
        {
             System.out.printf("%d ", it.next());
        }
    }

}
```



```
<terminated> tictoc [Java Application] C:
0 0 0 0 0 0 1
```