Alice and Bob take turns playing a game with **Alice starting first**.

In this game, there are n piles of stones. On each player's turn, the player should remove any **positive** number of stones from a non-empty pile **of his or her choice**. The first player who cannot make a move loses, and the other player wins.

Given an integer array piles, where piles[i] is the number of stones in the ith pile, return true*if Alice wins, or*false*if Bob wins*.

Both Alice and Bob play **optimally**.

**Example 1:**

**Input:** piles = [1]

**Output:** true

**Explanation:** There is only one possible scenario:

- On the first turn, Alice removes one stone from the first pile. piles = [0].

- On the second turn, there are no stones left for Bob to remove. Alice wins.

**Example 2:**

**Input:** piles = [1,1]

**Output:** false

**Explanation:** It can be proven that Bob will always win. One possible scenario is:

- On the first turn, Alice removes one stone from the first pile. piles = [0,1].

- On the second turn, Bob removes one stone from the second pile. piles = [0,0].

- On the third turn, there are no stones left for Alice to remove. Bob wins.

**Example 3:**

**Input:** piles = [1,2,3]

**Output:** false

**Explanation:** It can be proven that Bob will always win. One possible scenario is:

- On the first turn, Alice removes three stones from the third pile. piles = [1,2,0].

- On the second turn, Bob removes one stone from the second pile. piles = [1,1,0].

- On the third turn, Alice removes one stone from the first pile. piles = [0,1,0].

- On the fourth turn, Bob removes one stone from the second pile. piles = [0,0,0].

- On the fifth turn, there are no stones left for Alice to remove. Bob wins.

**Constraints:**

* n == piles.length
* 1 <= n <= 7
* 1 <= piles[i] <= 7