Problem 403: Frog Jump

Problem Information

Difficulty: Hard
Acceptance Rate: 46.94%
Paid Only: No
Tags: Array, Dynamic Programming

Problem Description

A frog is crossing a river. The river is divided into some number of units, and at each unit, there may or may not exist a stone. The frog can jump on a stone, but it must not jump into the water.

Given a list of `stones` positions (in units) in sorted **ascending order** , determine if the frog can cross the river by landing on the last stone. Initially, the frog is on the first stone and assumes the first jump must be `1` unit.

If the frog's last jump was `k` units, its next jump must be either `k - 1`, `k`, or `k + 1` units. The frog can only jump in the forward direction.

Example 1:

Input: stones = [0,1,3,5,6,8,12,17] **Output:** true **Explanation:** The frog can jump to the last stone by jumping 1 unit to the 2nd stone, then 2 units to the 3rd stone, then 2 units to the 4th stone, then 3 units to the 6th stone, 4 units to the 7th stone, and 5 units to the 8th stone.

Example 2:

Input: stones = [0,1,2,3,4,8,9,11] **Output:** false **Explanation:** There is no way to jump to the last stone as the gap between the 5th and 6th stone is too large.

Constraints:

* `2 <= stones.length <= 2000` * `0 <= stones[i] <= 231 - 1` * `stones[0] == 0` * `stones` is sorted in a strictly increasing order.

Code Snippets

C++:

```
class Solution {
public:
bool canCross(vector<int>& stones) {

}
};
```

Java:

```
class Solution {
public boolean canCross(int[] stones) {

}
}
```

Python3:

```
class Solution:
def canCross(self, stones: List[int]) -> bool:
```