LeetCode 403

Description

A frog is crossing a river. The river is divided into x 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 is able to cross the river by landing on the last stone. Initially, the frog is on the first stone and assume the first jump must be 1 unit.

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

Note:

The number of stones is ≥ 2 and is < 1,100.
Each stone's position will be a non-negative integer < 231.
The first stone's position is always 0.
Example 1:

[0,1,3,5,6,8,12,17]

There are a total of 8 stones.
The first stone at the 0th unit, second stone at the 1st unit,
third stone at the 3rd unit, and so on…
The last stone at the 17th unit.

Return true. 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:

[0,1,2,3,4,8,9,11]

Return false. There is no way to jump to the last stone as
the gap between the 5th and 6th stone is too large.

Thought

需要DFS的解法 - 青蛙能否从一个位置跳到下一个位置，取决于下一个位置上是否有stone，并且下一个位置还能否继续往前跳。基于每个stone都建立一个走到它的步骤，然后看它能走到哪些其他的stone，如果可以走到就在那个stone记下这个步骤大小，依次类推，看能不能走到最后一个。

Solution

```
class Solution{
  public boolean canCross(int[] stones) {
       if (stones == null || stones.length == 0) {return false;}
       int n = stones.length;
       if (n == 1) {return true;}
       if (stones[1] != 1) {return false;}
       if (n == 2) {return true;}
       int last = stones[n - 1];
       HashSet<Integer> hs = new HashSet();
       for (int i = 0; i < n; i++) {
           if (i > 3 && stones[i] > stones[i - 1] * 2) {return false;} // The two s
           hs.add(stones[i]);
       }
       return canReach(hs, last, 1, 1);
   }

   private boolean canReach(HashSet<Integer> hs, int last, int pos, int jump) {
       if (pos + jump - 1 == last || pos + jump == last || pos + jump + 1 == last)
           return true;
       }
       if (hs.contains(pos + jump + 1)) {
           if (canReach(hs, last, pos + jump + 1, jump + 1)) {return true;}
       }
       if (hs.contains(pos + jump)) {
           if (canReach(hs, last, pos + jump, jump)) {return true;}
       }
       if (jump > 1 && hs.contains(pos + jump - 1)) {
           if (canReach(hs, last, pos + jump - 1, jump - 1)) {return true;}
       }
       return false;
   }
}
```