Jump Game 3

Friday, December 10, 2021 6:55 AM

1306. Jump Game III

Medium

Given an array of non-negative integers `arr`, you are initially positioned at `start` index of the array. When you are at index `i`, you can jump to `i + arr[i]` or `i - arr[i]`, check if you can reach to **any** index with value 0.
Notice that you can not jump outside of the array at any time.

Example 1:

Input: arr = [4,2,3,0,3,1,2], start = 5

Output: true

Explanation:
All possible ways to reach at index 3 with value 0 are:
index 5 -> index 4 -> index 1 -> index 3
index 5 -> index 6 -> index 4 -> index 1 -> index 3

Example 2:

Input: arr = [4,2,3,0,3,1,2], start = 0

Output: true

Explanation:
One possible way to reach at index 3 with value 0 is:
index 0 -> index 4 -> index 1 -> index 3

Example 3:

Input: arr = [3,0,2,1,2], start = 2

Output: false

Explanation: There is no way to reach at index 1 with value 0.

Constraints:

- 1 <= arr.length <= 5 * 104
- 0 <= arr[i] < arr.length
- 0 <= start < arr.length

From <https://leetcode.com/problems/jump-game-iii/>

HINT 1
Think of BFS to solve the problem.
Hide Hint 2

From <https://leetcode.com/problems/jump-game-iii/>

When you reach a position with a value = 0 then return true.

From <https://leetcode.com/problems/jump-game-iii/>

BFS

JAVA

```java
class Solution{
    public static boolean canReach(int[] arr, int start) {
        int n = arr.length;
        Queue<Integer> q = new LinkedList<>();
        q.add(start);

        while (!q.isEmpty()) {
            int curr = q.poll();

            if (arr[curr] == 0) return true;   // reached the target index

            // reached this index again, so not possible from this index but might be possible to reach from other direction, so check again in the queue
            if (arr[curr]<0) continue;

            if (curr + arr[curr] < n)
                q.add(curr + arr[curr]);
            if (curr - arr[curr] >= 0)
                q.add(curr - arr[curr]);

            arr[curr] = -arr[curr];   // to distinguish between index coming first time or again
        }

        return false;
    }
```

DFS

JAVA

```java
class Solution {
    public boolean canReach(int[] arr, int start) {

        if(start<0 || start>=arr.length || arr[start]<0) return false;   // terminating conditions

        if(arr[start]==0){   //reached the target
            return true;
        }

        arr[start] = -arr[start];

        return canReach(arr,start+arr[start])||canReach(arr,start-arr[start]);   //checking in both direction
    }
}
```