Counting Elements

Solution

Given an integer array `arr`, count element `x` such that `x + 1` is also in `arr`.

If there're duplicates in `arr`, count them seperately.

Example 1:

```
Input: arr = [1,2,3]
Output: 2
Explanation: 1 and 2 are counted cause 2 and 3 are in arr.
```

Example 2:

```
Input: arr = [1,1,3,3,5,5,7,7]
Output: 0
Explanation: No numbers are counted, cause there's no 2, 4, 6, or 8 in arr.
```

Example 3:

```
Input: arr = [1,3,2,3,5,0]
Output: 3
Explanation: 0, 1 and 2 are counted cause 1, 2 and 3 are in arr.
```

Example 4:

```
Input: arr = [1,1,2,2]
Output: 2
Explanation: Two 1s are counted cause 2 is in arr.
```

Constraints:

- `1 <= arr.length <= 1000`
- `0 <= arr[i] <= 1000`

Hide Hint #1

Use hashset to store all elements.

Hide Hint #2

Loop again to count all valid elements.

Java

```java
// first idea : using map like 2 sums but couldnt think it through
// 2nd idea : sort arr, check consecutive elements, use 2 pointers when there are duplicate elements
// to keep track of how of them -> realised it might work but sorting is a waste of time
// 3rd idea : thought of hashset, tested idea and surprise it works. proceed to check and it really // // work. hm.... hashset work because the problem here is when we look at 1, we dont know until later if // 2 exists...by doing one pass and storing items in set, we essentially get to see the future.

class Solution {
    public int countElements(int[] arr) {

        if(arr.length == 0) return 0;
        Arrays.sort(arr);
        int prev = -1;
        int prevCount = 0;
        int count = 0;
        for(int i : arr) {
            if(i == prev) {
                prevCount++;
```

```
            }
            else {
                if(i == prev + 1) {
                    count += prevCount;
                }
                prevCount = 1;
                prev = i;
            }
        }

        return count;
    }
}
```

Custom Testcase (Contribute !)

Run Code Submit