K-Diff Pairs in an Array

Question:

https://leetcode.com/problems/k-diff-pairs-in-an-array/

Given an array of integers `nums` and an integer `k`, return the number of unique k-diff pairs in the array.

A k-diff pair is an integer pair `(nums[i], nums[j])`, where the following are true:

- *`0 <= i < j < nums.length`*
- *`|nums[i] - nums[j]| == k`*

Notice that `|val|` denotes the absolute value of `val`.

Example 1:

```
Input: nums = [3,1,4,1,5], k = 2
Output: 2
Explanation: There are two 2-diff pairs in the array, (1, 3) and (3, 5).
Although we have two 1s in the input, we should only return the number of unique pairs.
```

Example 2:

```
Input: nums = [1,2,3,4,5], k = 1
Output: 4
Explanation: There are four 1-diff pairs in the array, (1, 2), (2, 3), (3, 4) and (4, 5).
```

Example 3:

```
Input: nums = [1,3,1,5,4], k = 0
Output: 1
Explanation: There is one 0-diff pair in the array, (1, 1).
```

Approach:

Counted the occurrence of each number and stored it in a hashmap.
Then checked for target sum through the hashmap.
Also handled the boundary case of 0 sums in the else block at the end.

Solution:

```
def findPairs(self,nums,k):
        answ=0
        d={}
        for i in nums:
                if i in d:
                        d[i]+=1
                else:
                        d[i]=1
        answ=0
        if k>0:
                for i in d:
                        if (i+k) in d:
                                answ+=1
        else:
                for i in d:
                                answ+=(d[i]>1)
        return answ
```

Time Complexity: O(n)

Space Complexity: O(n)