Pair Sum to 0

Problem Description: Given a random integer array A, find and print the pair of elements in the array which sum to 0. Array A can contain duplicate elements.

Sample Input:

```
5
2 1 -2 2 3
```

Sample Output:

```
-2 2
-2 2
```

How to approach?

The naive approach to this problem would be to run a double nested loop and check every pair for their sum. If it's equal to 0, we print it else we move to the next iteration. The double nested loop will look like this:

```
for(i <- 0; i < arr.size() - 1; i <- i + 1):
     for(j <- i + 1; j < arr.size(); j <- j + 1):
          if(arr[i] + arr[j] = 0):
               printPair();
```

The time complexity of this method is $O(n^2)$ because of the double nested loop and the space complexity is O(1) since we are not using any extra space.

We can improve the time complexity to O(n) at the cost of some extra space.
The idea is that in the naive approach, we are checking every possible pair that can be formed but we don't have to do that.
If we iterate through the array, and we encounter some element `arr[i]`, then all we need to do is to check whether we've encountered `-arr[i]` somewhere previously in the array and if yes, then how many times. Ideally, we would want to access this information in O(1) time. For this, we can use a HashMap.

So, now we know how many times `-arr[i]` has appeared. Then we can print the pair `(-arr[i], arr[i])` {frequency of `-arr[i]`} times.

The time complexity of this approach will be O(n) and so will the space complexity.

Things to look out for!
What happens if there are multiple 0s in the array. We'll have to be careful that we don't consider `(0, 0)` where both 0s are on the same index as a valid pair. This problem has an easy fix. Just make sure that you insert/update the current element after you've printed the pairs (if any) for that iteration.

The pseudo-code for this approach is shown on the next page.

```
function pairSum(arr):
  HashMap(integer,integer) frequencyMap
  for element in arr:
    complement <-  element * -1
    if(frequencyMap contains the key ‘complement’):
      //print pair frequencyMap[complement] times
    frequencyMap[element] <- frequencyMap[element] + 1
    //frequencyMap[element] is initially 0
```