Find if there is a subarray with 0 sum

Given an array of positive and negative numbers, find if there is a subarray (of size at-least one) with 0 sum.

Examples:

```
Input: {4, 2, -3, 1, 6}
Output: true
There is a subarray with zero sum from index 1 to 3.

Input: {4, 2, 0, 1, 6}
Output: true
There is a subarray with zero sum from index 2 to 2.

Input: {-3, 2, 3, 1, 6}
Output: false
There is no subarray with zero sum.
```

We strongly recommend to minimize the browser and try this yourself first.

A **simple solution** is to consider all subarrays one by one and check the sum of every subarray. We can run two loops: the outer loop picks a starting point i and the inner loop tries all subarrays starting from i (See this for implementation). Time complexity of this method is $O(n^2)$.

We can also **use hashing**. The idea is to iterate through the array and for every element arr[i], calculate sum of elements form 0 to i (this can simply be done as sum += arr[i]). If the current sum has been seen before, then there is a zero sum array. Hashing is used to store the sum values, so that we can quickly store sum and find out whether the current sum is seen before or not.

Following is Java implementation of the above approach.

```
// A Java program to find if there is a zero sum subarray
import java.util.HashMap;

class ZeroSumSubarray {

    // Returns true if arr[] has a subarray with sero sum
    static Boolean printZeroSumSubarray(int arr[])
    {
        // Creates an empty hashMap hM
        HashMap<Integer, Integer> hM = new HashMap<Integer, Integer>();

        // Initialize sum of elements
        int sum = 0;

        // Traverse through the given array
        for (int i = 0; i < arr.length; i++)
        {
            // Add current element to sum
            sum += arr[i];

            // Return true in following cases
            // a) Current element is 0
            // b) sum of elements from 0 to i is 0
            // c) sum is already present in hash map
            if (arr[i] == 0 || sum == 0 || hM.get(sum) != null)
               return true;

            // Add sum to hash map
            hM.put(sum, i);
        }

        // We reach here only when there is no subarray with 0 sum
        return false;
    }

    public static void main(String arg[])
    {
        int arr[] = {4, 2, -3, 1, 6};
        if (printZeroSumSubarray(arr))
            System.out.println("Found a subarray with 0 sum");
        else
            System.out.println("No Subarray with 0 sum");
    }
}
```

Output:

```
Found a subarray with 0 sum
```

Time Complexity of this solution can be considered as O(n) under the assumption that we have good hashing function that allows insertion and retrieval operations in O(1) time.

Exercise:
Extend the above program to print starting and ending indexes of all subarrays with 0 sum.