Q21. Subarray with sum = 0

i/p → {4, 2, -3, 1, 6}
o/p → Yes

Brute force
Simply we have to consider all the subarrays and if we find any subarray having sum = 0, we return true but if true was not returned even after going through all subarrays, then simply return false.

Time complexity = $O(n^2)$
Space complexity = $O(1)$

Optimal solution + Dry run
Here we will be using the concept of prefix sum

Ex → {4, 2, -3, 1, 6}

4
4 + 2 = 6
6 - 3 = 3
3 + 1 = 4
Hence subarray exists
4 + 6 = 10

Ex → {1, 2, 3, 0, 6}
If arr[i] == 0, then also we have to return true.

Ex → {3, -1, -2, 5}
3

3 - 1 = 2
2 - 2 = 0 → We have to return true here,
0 + 5 = 5 from starting we get the subarray

Subarray → {3, -1, -2}

3 conditions
1) sum == 0
2) arr[i] == 0
3) map[sum] == 1

} Any of the condition satisfied, simply return true.

Code

```
bool subArrayExists (int arr[], int n) {
        //Initial settings
    unordered_map<int> m;
    int sum = 0;
    // Traverse the array & find prefix sum
    for (int i=0; i<n; i++) {
        sum = sum + arr[i];
        //Checking 3 conditions
        //Any one of the conditions may be fullfilled
        if (sum==0 || m[sum]==1 ||
            arr[i]==0) {
            return true;
        }
        // Update the map
        else {
            m[sum] = 1
        }
    }
    return false;
}
```

Time complexity = O(n)
Space complexity = O(n)

The code will run even if we do not check that arr[i] == 0 as this is already included in the condition of m[sum] = 1 and hence we can use only 2 conditions.