Print all subarrays with 0 sum

Given an array, print all subarrays in the array which has sum 0.

Examples:

```
Input:  arr = [6, 3, -1, -3, 4, -2, 2,
             4, 6, -12, -7]
Output:
Subarray found from Index 2 to 4
Subarray found from Index 2 to 6
Subarray found from Index 5 to 6
Subarray found from Index 6 to 9
Subarray found from Index 0 to 10
```

Related posts: Find if there is a subarray with 0 sum

A simple solution is to consider all subarrays one by one and check if sum of every subarray is equal to 0 or not. The complexity of this solution would be O(n^2).

A better approach is to use Hashing.

Do following for each element in the array

1. Maintain sum of elements encountered so far in a variable (say sum).
2. If current sum is 0, we found a subarray starting from index 0 and ending at index current index
3. Check if current sum exists in the hash table or not.
4. If current sum exists in the hash table, that means we have subarray(s) present with 0 sum that ends at current index.
5. Insert current sum into the hash table

Below is C++ implementation of above idea –

```
// C++ program to print all subarrays
// in the array which has sum 0
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

// Function to print all subarrays in the array which
// has sum 0
vector< pair<int, int> > findSubArrays(int arr[], int n)
{
    // create an empty map
    unordered_map<int, vector<int> > map;

    // create an empty vector of pairs to store
    // subarray starting and ending index
    vector <pair<int, int>> out;

    // Maintains sum of elements so far
    int sum = 0;

    for (int i = 0; i < n; i++)
    {
        // add current element to sum
        sum += arr[i];

        // if sum is 0, we found a subarray starting
        // from index 0 and ending at index i
        if (sum == 0)
            out.push_back(make_pair(0, i));

        // If sum already exists in the map there exists
        // at-least one subarray ending at index i with
        // 0 sum
        if (map.find(sum) != map.end())
        {
            // map[sum] stores starting index of all subarrays
            vector<int> vc = map[sum];
            for (auto it = vc.begin(); it != vc.end(); it++)
                out.push_back(make_pair(*it + 1, i));
        }

        // Important - no else
        map[sum].push_back(i);
    }

    // return output vector
    return out;
}

// Utility function to print all subarrays with sum 0
void print(vector<pair<int, int>> out)
{
    for (auto it = out.begin(); it != out.end(); it++)
        cout << "Subarray found from Index " <<
            it->first << " to " << it->second << endl;
}

// Driver code
int main()
{
    int arr[] = {6, 3, -1, -3, 4, -2, 2, 4, 6, -12, -7};
    int n = sizeof(arr)/sizeof(arr[0]);

    vector<pair<int, int> > out = findSubArrays(arr, n);

    // if we didn’t find any subarray with 0 sum,
    // then subarray doesn’t exists
    if (out.size() == 0)
        cout << "No subarray exists";
    else
        print(out);

    return 0;
}
```

Output:

```
Subarray found from Index 2 to 4
Subarray found from Index 2 to 6
Subarray found from Index 5 to 6
Subarray found from Index 6 to 9
Subarray found from Index 0 to 10
```