题目：

Given a list of **non-negative** numbers and a target **integer** k, write a function to check if the array has a continuous subarray of size at least 2 that sums up to the multiple of **k**, that is, sums up to n*k where n is also an **integer**.

Example 1:

```
Input: [23, 2, 4, 6, 7],  k=6

Output: True

Explanation: Because [2, 4] is a continuous subarray of size 2 and sums up t
o 6.
```

Example 2:

```
Input: [23, 2, 6, 4, 7],  k=6

Output: True

Explanation: Because [23, 2, 6, 4, 7] is an continuous subarray of size 5 an
d sums up to 42.
```

Note:

1. The length of the array won't exceed 10,000.
2. You may assume the sum of all the numbers is in the range of a signed 32-bit integer.

Subscribe to see which companies asked this question.

1.时间：O（N^2）;空间：O（N）

```
class Solution {
public:
    bool checkSubarraySum(vector<int>& nums, int k) {
        if (nums.size() < 2) return false;
        const int len = nums.size();
        if (k == 0){   /*  优化 k==0 的情况  */
            return hasContinuousZero(nums);
        }
        std::vector<int> dp(nums.size() + 1, 0);
        for (int i = 1; i < dp.size(); ++i)
            dp[i] = dp[i - 1] + nums[i - 1];
        for (int i = 0; i < len; ++i){
            for (int j = i + 2; j <= len; ++j){
                int subsum = dp[j] - dp[i];
                if (subsum % k == 0) return true;
            }
        }
        return false;
    }
private:
    bool hasContinuousZero(const std::vector<int>& nums){
        for (int i = 1; i < nums.size(); ++i)
            if (nums[i] == 0 && nums[i - 1] == 0) return true;
        return false;
    }
};
```

2.时间：O（N）; 空间：O（N）

```
class Solution {
public:
    bool checkSubarraySum(vector<int>& nums, int k) {
        if (nums.size() < 2) return false;
        const int len = nums.size();
        if (k == 0) return hasContinuousZero(nums);
        std::unordered_map<int, int> hashTable;
        hashTable[0] = -1;
        int sum = 0;
        for (int i = 0; i < len; ++i){
            sum = (sum + nums[i]) % k;
            if (hashTable.count(sum)){
                if (i - hashTable[sum] > 1) return true;
            }
            else hashTable[sum] = i;
        }
        return false;
    }
private:
    bool hasContinuousZero(const std::vector<int>& nums){
        for (int i = 1; i < nums.size(); ++i)
            if (nums[i] == 0 && nums[i - 1] == 0) return true;
        return false;
    }
};
```